Example-based XAI

A Machine Teaching Approach

Brigt Håvardstun

Thesis for the degree of Philosophiae Doctor (PhD)
at the University of Bergen

Scientific environment

The work presented in this thesis was carried out in the Algorithms group at the Department of Informatics, University of Bergen, as part of the project *Machine Teaching for Explainable AI (MT4XAI)*, funded by the Research Council of Norway (grant no. 329745).

The MT4XAI project is a collaboration among three research groups: the Algorithms group and the Machine Learning group, both based in the Department of Informatics at the University of Bergen (UiB), Norway, and the Data, Machines, Intelligence & People (DMIP) team, part of the Valencian Research Institute for Artificial Intelligence (VRAIN) at the Universitat Politècnica de València (UPV), Spain. Each group is represented by at least one principal investigator. The collaboration has been active throughout the doctoral period, including a one-year research visit to the DMIP team, and is reflected in joint publications throughout this thesis.

The MT4XAI project also included industry collaboration with Equinor and Eviny. The collaboration with Eviny resulted in one of the publications included in this thesis (Paper (iv)).

Acknowledgements

First and foremost, to my supervisor, Jan Arne Telle. You have been with me since my master's, so we have had a few years to get to know one another. I want to thank you especially for the moments when you miraculously managed to help me untangle my thoughts and get to the core of an idea, and for the moments when you were direct and honest in telling me that there was nothing there. Your influence on the group around you is evident, and your openness and warmth are inspiring.

Secondly, I want to thank Cèsar Ferri, my co-supervisor. Thank you for your patience and for helping me lift my head from the minute details whenever they were not needed for the research at hand. Beyond our own research, it has been wonderful to see the part you play in shaping the group at DMIP — learning by example(s) truly works in this group — and I am grateful for the role you played in making my year in Valencia possible.

I would also like to give special thanks to everyone at DMIP at VRAIN. To each and every person — from research associates and PhD students to postdocs and professors — you made my year in Valencia unforgettable, both through the research and everything around it. Thank you for making our stay in your city what it was. Valencia will always feel like a second home.

To Behzad, Yael, Dani, Nick, Chris, Nando, and Carlos. Thank you for making each and every day a joy. From laughs during lunch to after-work get-togethers, your warmth and way of life stay with me.

To Diogo Nuno Freitas, as a fellow visitor to DMIP, collaborating with you has been a breath of fresh air. I recall so many discussions and investigations in that office; just the thought brings me joy.

José Hernández-Orallo, thank you for the many times Jan Arne and I turned up with something we found interesting about a narrow problem, and you showed us how to frame it so that it spoke to the field — or showed us that it was really an instance of

a larger question worth answering. Thank you also for your generosity with your time. You were plainly busy throughout all four years; despite this, I always felt you were available for questions, and you always gave our ideas time anyway.

To the rest of my co-authors: Pekka Parviainen, Joakim Sunde, Jan Kratochvíl, Kristian Flikka, and Félix Martí-Perez. Thank you all for our collaborations.

I would also like to thank the people on the PACE project in 2025: Martin Vatshelle, Thomas Alme Matre, Pål Grønås Drange, and Yash Hiren More.

I have greatly appreciated the collaboration and exploration on PORGS — follow-up work from the ORS paper — and would like to thank Maira Farias de Andrade Lira and Ricardo Prudêncio.

To the Equinor gang: Bjarte Johansen, Jakob Vigerust Kallestad, and Seong-Eun Cho. Through workshops and collaborations in Bergen, seeing how XAI is used in practice and taking part in research exploration with you has helped frame how I view these problems in real-world settings.

The Department of Informatics was a good place to be a student. Thank you to Pinar Heggernes, Eirik Rekve Thorsheim, and Linda Vagtskjold, who between them said yes to a great deal of student activity — including, eventually, a student bar. The sense of possibility you helped create is a large part of why I stayed for a PhD.

To the Algorithms group at UiB, I would like to thank you for our Friday lunches and gatherings; it was always amazing to see how large a group we were. I greatly appreciated the practice of learning something from one another each week, and the way it connected us despite the breadth of our research topics. For the many Fridays we shared, thank you to Boris Djalal, Emmanuel Arrighi, Emmanuel Sam, Erlend Raa Vågse, Fedor Fomin, Fredrik Manne, Hilde Jordal, Idris Elmi, Jakob Rødal Skaar, Jonas Haukenes, Juni Weisteen Bjerde, Kenneth Langedal, Kirill Simonov, Krishnan Dehaleesan, Lars Jaffke, Laure Morelle, Madhumita Kundu, Matthias Bentert, Maya Robbestad, Petr Golovach, Petra Wolf, Saket Saurabh, Sam Urmian, Svein Høgemo, Tian Bai, Tobia Gasparoni, Tomas Turek, Tuukka Korhonen, and William Wale.

I would also like to extend special thanks to Wim Van den Broeck and Yash Hiren More for always having an open door. Sadly for them and luckily for me, this included serving as the next-door guinea pigs whenever a new human survey needed testing.

To Darío Garigliotti, as the postdoc on the MT4XAI project, I want to thank you for being on this journey with me. Since Valencia, we have shared an office and shared ideas. Your calm and composed demeanour in discussions was exactly the kind of guidance a

young and eager researcher needed.

To Pål, thank you for all the little things and questions you have answered. From research discussions to future opportunities, I am grateful.

To my family. Thank you for raising me to be the person I am today, and for allowing me to always question and wonder. I am grateful to have your support, and the sense of safety you have given me has allowed me to take large leaps of faith.

Finally, to my dearest Sanne. If anyone truly knows the story behind this thesis, it's you. Meeting you convinced me that both doing a PhD and staying in Bergen were the right choices for me. While I have had my head stuck in work, you have always adapted, stepped in, and had my back. You came with me to Valencia, never complained when my mind wandered to work, and served as my alpha and omega guinea pig for all things related to this thesis. You give me hope, belief, and strength; with this thesis complete, the next chapters are ours.

Abstract

As Artificial Intelligence (AI) systems are increasingly deployed in high-stakes settings, the need for humans to understand their behaviour has grown accordingly. Example-based eXplainable AI (XAI) addresses this need by selecting informative examples that illustrate how a model behaves. This thesis investigates how machine teaching (MT) – a formal theory of teaching by examples – can serve as a theoretical grounding for example-based XAI. In MT, a *teacher* selects a set of examples so that a *learner* can uniquely identify a target concept among alternative concepts. Defining the AI to be explained as the target concept translates explanation into a teaching task: finding the examples that best distinguish this AI from other candidate concepts. The overarching goal is that the human learner, from a small set of well-chosen examples, understands the model well enough to *simulate* it – predict its behaviour on new instances.

We present a user study where MT-selected examples helped humans understand a target concept better than random examples of comparable complexity, provided the learner model aligned with the human learner. We further extend the theory to concept classes with redundant representations, and prove a conjecture on which concept classes minimise the number of realisable teaching sets.

We also explore example-based XAI for time series. A user study of our interactive model-inspection tool found no gain in participants' accuracy – the time series themselves appeared too complex – prompting a simplification technique tailored to XAI and a framework for evaluating simplification algorithms as explanations.

Bridging MT and XAI, we use teaching size to compare concept complexity across modalities (text and image) in vision-language models, and find that relative complexity is largely preserved. Finally, we introduce *PAC teaching*, replacing perfect learners with learners that make stochastic *deductive errors* when checking examples against concepts, and provide algorithms, hardness results, and an empirical validation on an LLM.

These results position machine teaching as a principled foundation for selecting explanatory examples, and identify modelling the human learner as the central challenge ahead.

Sammendrag

Kunstig intelligens (KI) brukes i dag mer og mer i høyrisikosituasjoner. Dette gir et økt behov for å forstå KI-systemer. En måte å forstå dem på er eksempelbasert forklarbar KI (XAI), hvor informative eksempler illustrerer hvordan KI-systemet oppfører seg. Denne avhandlingen undersøker hvordan maskinundervisning (MU, eng. *machine teaching*) – en formell teori for undervisning ved hjelp av eksempler – kan fungere som et teoretisk fundament for eksempelbasert XAI. I MU velger en *underviser* et sett med eksempler slik at en *student* entydig kan identifisere et gitt *målkonsept*. Både underviseren og studenten kan være et menneske eller en maskin. Lar vi målkonseptet være selve KI-systemet som skal forklares, blir forklaringsproblemet gjort om til et undervisningsproblem: å finne de eksemplene som best skiller dette KI-systemet fra andre kandidatkonsepter. Det overordnede målet er at et menneske i rollen som student, ut fra et lite sett med velvalgte eksempler, forstår KI-systemet godt nok til å kunne *simulere* det – forutsi hvordan det oppfører seg på nye instanser.

I en brukerstudie så vi at eksempler valgt med MU hjalp deltakerne til å forstå et gitt konsept bedre enn tilfeldig valgte eksempler med tilsvarende kompleksitet – forutsatt at studentmodellen samsvarte med den menneskelige studenten. Videre utvider vi teorien ved å ta hensyn til at samme konsept kan ha flere representasjoner, og beviser en formodning om hvilke konseptklasser som minimerer antallet realiserbare undervisningssett (eng. *teaching sets*).

Vi utforsker også eksempelbasert XAI for tidsserier. I en brukerstudie av vårt interaktive verktøy for modellinnsyn (eng. *model inspection*) så vi ingen bedring i deltakernes treffsikkerhet – tidsseriene i seg selv så ut til å være for komplekse. Dette ledet til en forenklingsteknikk skreddersydd for XAI og et rammeverk for å evaluere forenklingsalgoritmer som forklaringer.

Som en bro mellom MU og XAI bruker vi undervisningsstørrelse (eng. *teaching size*) til å sammenligne konseptkompleksitet på tvers av modaliteter (tekst og bilde) i modeller som håndterer begge deler, og finner at den relative kompleksiteten i stor grad bevares. Til slutt introduserer vi PAC-undervisning (eng. *PAC teaching*), der antakelsen om perfekte

studenter erstattes med studenter som gjør stokastiske deduktive feil (eng. *deductive errors*) når de sjekker eksempler mot konsepter, og vi gir algoritmer, hardhetsresultater og en empirisk validering på en stor språkmodell (LLM).

Disse resultatene etablerer maskinundervisning som et prinsipielt fundament for å velge informative eksempler, og peker på modellering av den menneskelige studenten som den sentrale utfordringen fremover.

List of publications

No.	Short name	Title
i	Irrelevant Features	XAI with machine teaching when humans are (not) informed about the irrelevant features
ii	Redundancy	When redundancy matters: Machine teaching of representations
iii	Combinatorics	On a combinatorial problem arising in machine teaching
iv	Model Inspection	XAI for time series classification: Evaluating the benefits of model inspection for end-users
v	ORS	Optimal robust simplifications for explaining time series classifications
vi	Evaluating Simplifications	Evaluating simplification algorithms for interpretability of time series classification
vii	QuickDraw	Relative drawing identification complexity is invariant to modality in vision-language models
viii	Deductive Error	Teaching and learning under deductive errors

(i) Håvardstun, B., Ferri, C., Hernández-Orallo, J., Parviainen, P., & Telle, J. A. (2023). XAI with machine teaching when humans are (not) informed about the irrelevant features. *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2023)*, LNCS 14171, 378–393. https://doi.org/10.1007/978-3-031-43418-1_23

(ii) Ferri, C., Garigliotti, D., Hernández-Orallo, J., Håvardstun, B., & Telle, J. A. (2026). When redundancy matters: Machine teaching of representations. *Machine Learning*, *115*(1), 20. https://doi.org/10.1007/s10994-025-06954-3

(iii) Sunde, J., Håvardstun, B., Kratochvíl, J., & Telle, J. A. (2024). On a combinatorial problem arising in machine teaching. *Proceedings of the 41st Interna-*

tional Conference on Machine Learning (ICML 2024), PMLR 235, 47305–47313. https://proceedings.mlr.press/v235/sunde24a.html

(iv) Håvardstun, B., Ferri, C., Flikka, K., & Telle, J. A. (2024). XAI for time series classification: Evaluating the benefits of model inspection for end-users. *Proceedings of the World Conference on Explainable Artificial Intelligence (xAI 2024)*, CCIS 2155, 439–453. https://doi.org/10.1007/978-3-031-63800-8_22

(v) Telle, J. A., Ferri, C., & Håvardstun, B. (2024). Optimal robust simplifications for explaining time series classifications. *Proceedings of the Workshop on Explainable AI for Time Series and Data Streams (TempXAI 2024), co-located with ECML PKDD 2024*, CEUR 3761, 28–43. https://ceur-ws.org/Vol-3761/paper2.pdf

(vi) Håvardstun, B., Marti-Perez, F., Ferri, C., & Telle, J. A. (2025). Evaluating simplification algorithms for interpretability of time series classification. *arXiv preprint arXiv:2505.08846, to appear in Proceedings of the World Conference on Explainable Artificial Intelligence (xAI 2026).* https://arxiv.org/abs/2505.08846

(vii) Freitas, D., Håvardstun, B., Garigliotti, D., Telle, J. A., Ferri, C., & Hernández-Orallo, J. (2025). Relative drawing identification complexity is invariant to modality in vision-language models. *Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025)*, FAIA 413, 4507–4514. https://doi.org/10.3233/FAIA251351

(viii) Telle, J. A., Håvardstun, B., & Hernández-Orallo, J. (2026). Teaching and learning under deductive errors. Preprint, submitted to the 40th Annual Conference on Neural Information Processing Systems (NeurIPS 2026). https://arxiv.org/pdf/2605.13384

Contents

Part I

Synthesis

Chapter 1

Introduction

In recent years, the use of Artificial Intelligence (AI) in high-stakes situations has fuelled the need for eXplainable AI (XAI) [Barredo Arrieta et al., 2020; Gunning and Aha, 2019]. These AI systems are typically machine learning models whose decision logic is opaque even to experts – so-called black-box models [Guidotti et al., 2018] – and throughout this thesis we use the terms AI, model and black-box model interchangeably. As a field, XAI has many different goals, from increased trust to improved human-computer interaction [Rong et al., 2024]. We will focus on the goal of understandability: how well a human understands a given model. Concretely, we will measure understanding through *simulatability*, the ability of a human to predict the model's behaviour on new instances [Doshi-Velez and Kim, 2017]. Taking this view, the tools and methods built in XAI can be viewed as 'Teachers' for the human 'Learner'.

Specifically, this thesis focuses on the subcategory of XAI referred to as example-based explanations [Barredo Arrieta et al., 2020; Molnar, 2020]. These could be typical examples (prototypes; Bien and Tibshirani, 2011), atypical examples (criticisms; Kim et al., 2016), or minimally altered examples that change the prediction (counterfactuals; Wachter et al., 2018). In general, we can say that the aim of example-based explanations is to select informative (*example*, *AI prediction*) pairs, such that a human learner gains understanding of the model.

In this thesis we evaluate how machine teaching [Goldman and Kearns, 1995; Zhu et al., 2018] – a formal theory of teaching by examples – can serve as a theoretical grounding for example-based XAI, identify where it works well, and investigate its limitations.

We start from a simple assumption about human learners: when faced with a new AI, they draw on their experience and form a mental list of possible rules or approximations for how the AI will act.

Current example-based XAI methods typically do not utilise this idea explicitly. They either focus on teaching the data distribution (prototypes) or local aspects of the decision boundaries (counterfactuals). However, under the assumption that humans generate hypotheses, the goal of XAI should be to help the learner identify the best such hypothesis.

With such a view of example-based XAI as a problem of concept identification among a range of possible hypotheses, it is clear that there are connections to machine teaching: a framework for selecting informative examples, such that a learner can identify a target concept in a given hypothesis space. Formalising this view – with the XAI system as the teacher, the human as the learner, and the AI model as the target concept – gives a framework for example-based explanations grounded in machine teaching.

Implicit in this view is the assumption that a learner who identifies the correct concept also understands it well enough to predict the model's actions on new instances. As mentioned, the empirical work in this thesis evaluates understanding through exactly such forward simulation, and Paper (viii) later considers learners for whom this link weakens.

The work presented in this thesis can be positioned along two axes, as shown in Figure 1.1: the horizontal axis spans from theoretical to empirical results, and the vertical axis from machine teaching to example-based XAI. This positioning gives three natural groups for the eight papers that make up the main contribution of this thesis. The first group, machine teaching (blue), focuses on theoretical aspects of machine teaching viewed in the XAI setting (Papers (iii) and (ii)), as well as evaluating how well humans align with idealised formal learners (Paper (i)). The second group, example-based XAI (red), works in the time series domain: we evaluated how well humans perform as active learners, freely exploring a classifier and its predictions (Paper (iv)), and tested how and when simplification can help in teaching situations (Papers (v) and (vi)). The final group (green) bridges the two fields: Paper (vii) uses teaching size to measure drawing-identification complexity in vision-language models, and Paper (viii) replaces the assumption of perfect learners with learners that make mistakes, aligning the theory more closely with what we observe in practice.

The rest of the synthesis given in Part I is organised as follows. Chapter 2 introduces the machine teaching framework for XAI, and follows up with different work on machine teaching (Papers (i)–(iii)). Chapter 3 turns to example-based XAI for time series classification (Papers (iv)–(vi)). Chapter 4 presents the two papers combining aspects of the fields (Papers (vii) and (viii)). Chapter 5 then discusses the results and their limitations, before presenting directions for future work and concluding. Part II then presents

the papers themselves.

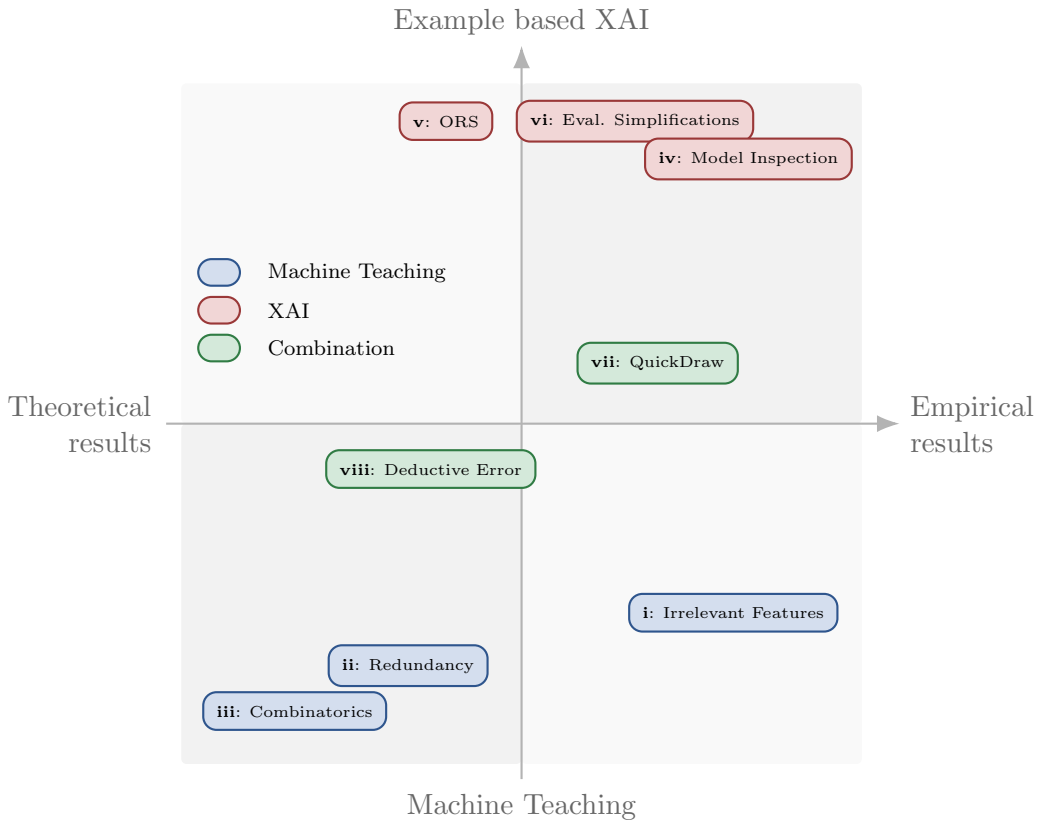


Figure 1.1: The eight papers of this thesis, positioned by field (machine teaching vs. example-based XAI) and type of result (theoretical vs. empirical).

Chapter 2

Machine Teaching

In this chapter, we present our work on machine teaching. To aid understanding, we begin with a toy example of machine teaching that will be used throughout the chapter. We then present three papers: the first on adapting machine teaching for explanations and the other two on properties of the concept class in machine teaching.

2.1 The machine teaching setting

A central goal of this thesis is to investigate how machine teaching can be used to generate example-based explanations for AI systems. In example-based eXplainable AI (XAI), explanations are provided through carefully selected examples that illustrate how a machine learning model behaves in different situations [Molnar, 2020; Ribeiro et al., 2018; van der Waa et al., 2021]. The underlying assumption is that humans, by creating their own mental models, can often understand a system more effectively from examples than from a complete description of the model itself.

Machine teaching (MT) studies how to select examples for learning [Zhu et al., 2018]. Given a target concept, the goal is to identify a small set of labelled examples – known as a *teaching set*[1] – that allows a learner to infer the target concept. Since explanations should ideally be concise, machine teaching provides a natural starting point for selecting explanatory examples.

In MT this is formalised by assuming that both teacher and learner share a common consistency matrix over a concept class C and an instance space X, with one row per concept and one column per instance. For each concept-instance pair, the consistency matrix in-

[1] Papers (ii) and (iii) use the equivalent term *witness set*.

dicates whether the instance belongs to the concept. To illustrate this setting, consider the instance space $X = \{2, 4, 5, 9\}$ and the concept class $C = \{c_{\text{even}}, c_{\text{prime}}, c_{\text{mult3}}, c_{>2}\}$, where c_{even} denotes the even numbers, c_{prime} the prime numbers, c_{mult3} the multiples of three, and $c_{>2}$ the numbers greater than two. Table 2.1 shows the binary consistency matrix describing the membership of each instance in each concept. For instance, in the row of $c_{>2}$ the column (2) has the value 0, as 2 is not part of the concept $c_{>2}$, whereas all other columns have 1, since they are part of the concept.

Table 2.1: A simple concept class used to illustrate machine teaching. An entry is 1 if the instance belongs to the concept and 0 otherwise.

Concept ↓ / Examples →	2	4	5	9
c_{even}	1	1	0	0
c_{prime}	1	0	1	0
c_{mult3}	0	0	0	1
$c_{>2}$	0	1	1	1

We denote a labelled example by (x, y), where x is an instance and $y \in \{T, F\}$ (True or False) is its label. Here, (x, T) indicates that instance x belongs to the concept, corresponding to a 1 in the consistency matrix, while (x, F) indicates that it does not, corresponding to a 0. A concept c is said to be *consistent* with a labelled example (x, y) if it assigns label y to instance x, denoted $c \models (x, y)$. More generally, a concept is consistent with a set S of labelled examples if it is consistent with every labelled example in S, denoted $c \models S$. A *teaching set* for a target concept $c^* \in C$ is a set $S \subseteq X \times \{T, F\}$ of labelled examples such that $c^* \models S$, selected by the teacher and presented to the learner. We say that S *uniquely identifies* c^* if c^* is the only concept in C consistent with S. Note that the definition of a teaching set requires the teacher to be truthful: every example is labelled according to the target concept.

Suppose the target concept is c_{prime}, i.e. the teacher aims to teach c_{prime}. A single labelled example is insufficient to uniquely identify the target concept. For example, both c_{prime} and c_{even} are consistent with $(2, T)$. However, the teaching set $\{(2, T), (4, F)\}$ uniquely identifies c_{prime}, as it is the only concept in C that labels 2 positively and 4 negatively.

More generally, every concept in Table 2.1 can be uniquely identified using two labelled examples, while no single example suffices. The goal of machine teaching is to find, for each concept, the smallest teaching set S that leads the learner L to select the target concept c^*, i.e. $L(S) = c^*$. Comparing the sizes of these teaching sets across a concept class leads to the teaching dimension (TD), defined as the size of the largest such teaching set. At its inception, Goldman and Kearns [1995] required c^*, and no other concept, to be consistent with S – unique identification, as above – so that any learner returning a concept consistent with S must return c^*; we refer to this specific case of TD as the

classical teaching dimension. Unless otherwise stated or clear from context, TD refers to the classical teaching dimension. The concept class in Table 2.1 therefore has (classical) teaching dimension 2.

2.2 Machine teaching for explanations

The first paper, Paper (i), investigates how machine teaching can be adapted to generate explanations for AI systems. More specifically, the paper studies the trade-off between two competing objectives. On the one hand, explanations should be simple and contain as few examples as possible (low *teaching complexity*). On the other hand, they should accurately communicate the behaviour of the target AI (high *explanatory fidelity*). Balancing the two is the key challenge.

In Paper (i), the AI to be explained takes the role of the target concept θ_{AI}. The XAI system is modelled as a teacher that selects explanatory examples for a learner, who in our setting is a human aiming to gain insight into the AI. The goal is to construct a small teaching set that enables the learner to build an accurate mental model of the target system.

To formalise this process, we adopt standard machine teaching terminology. The teacher T is viewed as a function from concepts to labelled example sets, with $T(\theta) = S$ denoting the teaching set used to teach concept θ. Similarly, the learner L maps labelled example sets to concepts, with $L(S) = \hat{\theta}$ denoting the concept inferred from the teaching set S. Successful teaching occurs when the learner infers the concept intended by the teacher, that is, when $L(T(\theta)) = \theta$.

Following Telle et al. [2019], we introduce a simplicity measure β over concepts, encoding a learner's preference over concepts, and a simplicity measure δ over teaching sets. The learner prefers simpler concepts when multiple concepts are consistent with the observed examples, while the teacher prefers simpler teaching sets. Since the true human learner L_H cannot be directly incorporated into the optimisation procedure, we instead employ a learner model L_M that approximates how a human reasons about examples. Finally, a loss function λ measures the discrepancy between the inferred concept θ_M and the target concept θ_{AI}. Lower values of λ correspond to higher explanatory fidelity.

Using these components, the framework searches for teaching sets that balance fidelity

and teaching complexity:

$$T(\theta_{AI}) = \underset{S:\theta_{AI} \models S}{\operatorname{argmin}}\, \delta(S) + \mu \cdot \lambda(\theta_{AI}, \theta_M) : L_M(S) = \theta_M \tag{2.1}$$
$$L_M(S) = \underset{\theta_M:\theta_M \models S}{\operatorname{argmin}}\, \beta(\theta_M)$$

Given a target AI model θ_{AI}, the teacher searches for a teaching set S such that the target model is consistent with it, i.e. $\theta_{AI} \models S$. The model of the learner L_M then receives S and selects the simplest concept θ_M that is consistent with the teaching set according to the simplicity measure β. The discrepancy between the target model and the learner's inferred concept is measured by $\lambda(\theta_{AI}, \theta_M)$, while the complexity of the teaching set is measured by $\delta(S)$. The teacher seeks a teaching set that minimises a weighted combination of these two quantities, where μ controls the trade-off between teaching complexity and fidelity.

This is the general bilevel form in which machine teaching problems are posed [Zhu et al., 2018, Eq. 2]: the teacher trades the cost of the teaching set against the discrepancy between the target and the concept the learner arrives at, subject to the learner running its own concept selection algorithm. What is specific to our setting is that the target is an AI model rather than a ground-truth concept, and that the learner minimises a preference β over concepts rather than an empirical risk.

The framework was implemented on machine learning models trained on images labelled by Boolean functions. The teaching sets produced by the framework were subsequently evaluated on human participants. Two main findings emerged from the experimental evaluation.

First, the effectiveness of the framework depends critically on the alignment between the learner model L_M and the human learner L_H. In particular, humans may focus on irrelevant features that are not represented in the learner model, leading them to form hypotheses that differ from those anticipated by the teaching algorithm.

Second, machine-teaching-generated explanations enabled participants to better understand the target concept than randomly selected examples of comparable complexity. This suggests that machine teaching can be used to identify informative examples for explanatory purposes.

The paper demonstrates that machine teaching provides a promising framework for generating example-based explanations. By explicitly balancing fidelity and teaching complexity, the framework can identify compact sets of examples that improve human un-

derstanding of an AI system. At the same time, the results highlight that the quality of the explanation depends heavily on how accurately the learner model captures human reasoning.

An important observation from this work is that a learner may reason about a concept using a different representation from the one assumed by the teacher. In Paper (i), such discrepancies often arose because humans attended to irrelevant features. However, differences in representation can also arise even when the teacher and learner agree on the underlying concept. For example, the Boolean expressions $A \wedge (B \vee C)$ and $(A \wedge B) \vee (A \wedge C)$ represent the same concept despite having different representations. Importantly, the learner can have different preferences over such equivalent representations. These preferences must be taken into account to determine which representation the learner will select when several are consistent with the teaching set. This raises the question of how machine teaching should operate when concepts admit multiple equivalent representations.

2.3 When concepts are not unique

In formal models of *machine learning* (ML), a concept class C defines the set of possible hypotheses, and the goal is to identify an unknown target concept $c^* \in C$ from randomly sampled instances $x \in X$ that are correctly labelled by the target concept. In contrast, in formal models of machine teaching, a collection of labelled examples, the so-called teaching set S, is carefully chosen by a teacher T so the learner L can identify c^* in C.

Most formal models of machine learning and machine teaching assume that C consists of distinct concepts, such that each target corresponds to a unique element in the concept class. However, this assumption is broken in situations where concepts can have multiple equivalent representations. For example, over the instance space $\mathbb{Z}$, the expressions "$x^2 > 0$" and "$x \neq 0$" are satisfied by exactly the same instances, and are thus two representations of the same concept. In such situations, the teacher may attempt to teach one representation of a concept without realising that the learner may instead infer a different, but equivalent, representation.

In Paper (ii) we explore the idea of teaching *representations*, where the teacher knows a representation and wants the learner to identify it. Different representations may belong to the same equivalence class, i.e. define the same concept, although checking this equivalence is not always possible. The paper extends the traditional machine teaching setting by considering a representation class R rather than a concept class, and

investigates how representational redundancy affects the teaching process.

The paper analyses the previously proposed *Eager* teaching algorithm [Telle et al., 2019] and introduces a new algorithm, *Greedy*, designed to address the challenges posed by representational redundancy. With Eager the learner employs Occam's razor and selects the simplest representation consistent with the teaching set, while the teacher provides the smallest teaching set sufficient to identify the target. This approach implicitly assumes that only one representation per concept needs to be learned. However, if the teacher does not know that two representations are equivalent and one representation is simpler than the other, the teacher may search indefinitely for a teaching set that distinguishes them, even though no such teaching set exists.

Surprisingly, only a tiny change to Eager is sufficient to enable learning multiple representations. When teaching a target, it is common to provide the smallest teaching set that suffices to identify it, and a learner may therefore expect the teacher to follow this strategy. Consequently, after learning a representation–teaching set pair (r, S_r), the learner can assume that neither r nor S_r will be used again for teaching. It is a function from teaching sets to representations, and already maps S_r to r. The learner can therefore remove both from consideration, and aside from this, follow the method of Eager. This simple idea forms the basis of the Greedy algorithm. Another teaching set S that is consistent with r may then be used to identify a different representation r', namely the simplest remaining representation consistent with S.

The two algorithms are evaluated empirically in six different settings, split into two groups. The first four settings form a graded family of Boolean expressions, ranging from 3-DNF with no redundancy to 3-Term DNF variants in which every concept has many equivalent representations, allowing the amount of redundancy to be varied in a controlled way. The last two, following Telle et al. [2019], both use programs written in the Turing-complete language P3, where many distinct programs compute the same function: one over the unrestricted program space, and one, small-P3, restricted to a subset small enough to analyse exhaustively.

The theoretical analysis showed that Greedy never uses larger teaching sets than Eager on concepts taught by both algorithms, while allowing multiple representations of the same concept to be taught. The empirical experiments demonstrated that Greedy teaches substantially more representations and concepts than Eager, often using smaller teaching sets. A key finding was that the benefit of Greedy depends less on the amount of redundancy and more on the newly introduced notion of *redundancy spread*, which measures how dispersed equivalent representations are in the representation ordering. Finally, for P3 programs, teaching sets were frequently smaller than the programs they

identified, illustrating the efficiency of machine teaching.

From this, we observe that the efficiency of teaching, as measured by the teaching dimension, depends not only on the learner – as seen in the difference between Eager and Greedy – but also on structural properties of the concept class, such as the amount of redundancy and the redundancy spread. This raises a natural question: can we identify concept classes that are inherently difficult to teach, regardless of the learner being used? The work in Paper (ii) led to a conjecture of exactly this form, singling out one family of concept classes as extremal in this sense. Paper (iii), presented in the next section, proves it.

2.4 A lower bound on teaching dimension

Different concept classes may require vastly different numbers of examples to teach and hence have vastly different teaching dimensions. For instance, the concept class in Table 2.1 has teaching dimension 2, but if we were to remove $c_{>2}$, the number of examples needed to uniquely identify the remaining concepts would drop to one for each: $(4, T)$ for c_{even}, $(5, T)$ for c_{prime}, $(9, T)$ for c_{mult3}. However, in addition to a given concept class, the precise number of examples needed for teaching depends on the learner-teacher protocol. Given this, it might seem difficult to say anything about which concept classes are harder than others.

We therefore search for a lower bound on the teaching dimension for any learner. To obtain a learner-independent lower bound, we focus on a property that every teacher must satisfy. Regardless of the learner, each concept must ultimately be assigned a distinct teaching set. Consequently, a concept class containing $|C|$ concepts requires at least $|C|$ distinct teaching sets. Let $|W^{\leq q}|$ denote the number of teaching sets with q or fewer examples (the notation W, for witness set, follows Papers (ii) and (iii), and Simon and Telle [2026]). Then every teaching protocol must use a value of q for which $|W^{\leq q}| \geq |C|$.

$$TD(C) \geq \min\{q : |W^{\leq q}| \geq |C|\}$$

This counting argument is, however, optimistic, as it implicitly assumes that every teaching set of size at most q can be used in teaching. In practice, this is not the case. Some labelled subsets are inconsistent with every concept in the concept class and can therefore never be used for teaching. Excluding such teaching sets leads to a stronger bound.

For an example of when a set of labelled examples is not usable for teaching, consider the concept class C from Table 2.1. The set $S = \{(2, T), (9, T)\}$ is not consistent with any concept, since no concept contains both 2 and 9. Furthermore, any teaching set containing S as a subset is also not consistent with any concept, and hence cannot be used for teaching.

Following Simon and Telle [2026], a teaching set S is *realisable* in C if there exists a concept $c \in C$ consistent with S, i.e. $c \models S$. We denote by $W_r^{\leq q}(C)$ the set of all realisable teaching sets containing at most q examples in the concept class C. Simon and Telle [2026] observed that restricting the counting argument to realisable teaching sets yields the stronger lower bound:

$$TD(C) \geq \min\{q : |W_r^{\leq q}| \geq |C|\}$$

whose right-hand side is denoted $SMN'(C)$ in Simon and Telle [2026]. Using the lower bound above, we can study which concept class maximises the lower bound on the teaching dimension. This can be formulated as asking which concept class minimises the number of realisable teaching sets of size at most q, that is $|W_r^{\leq q}|$, for every q.

Consider a concept class C with n concepts and m examples. We can view the problem as a game: construct C so as to minimise the number of realisable teaching sets. If we want to minimise the number of realisable teaching sets, a natural approach is to ensure all concepts assign the same label b to a given example x. Then every teaching set containing the labelled example $(x, \neg b)$ is not realisable. Continuing like this, it could be natural to try to make the rows have the same value for every example. If we were in the redundant representations setting considered in Paper (ii), the optimal solution would simply be to make every row identical, for example, all zeros. This would maximise agreement between concepts and minimise the number of realisable teaching sets.

In line with traditional machine teaching, we instead seek to maximise agreement while preserving uniqueness among concepts. We can still start with the same idea of having concepts agree on many examples, leaving their differences encoded only in the last few examples. Encoding the n distinct concepts in binary requires at least $k = \lceil \log_2 n \rceil$ examples; a visualisation is shown in Table 2.2.

In fact, the matrix shown in Table 2.2 arose from the work in Paper (ii), as part of an argument for a lower bound on the performance of Greedy. An earlier version of that paper conjectured this matrix to be extremal, and Paper (iii) named it $H_{n,m}$.[2] Simon and

[2]Paper (iii) writes $H_{n,k}$, with k concepts over n examples; we write $H_{n,m}$, with n concepts over m examples, to match the notation used elsewhere in this chapter.

Table 2.2: The proposed matrix $H_{n,m}$. Here illustrated for $n = 8$, where the first $m - 3$ columns are identical and the final three columns encode distinct binary identifiers.

x_1	x_2	$\cdots$	x_{m-3}	x_{m-2}	x_{m-1}	x_m
0	0	$\cdots$	0	0	0	0
0	0	$\cdots$	0	0	0	1
0	0	$\cdots$	0	0	1	0
0	0	$\cdots$	0	0	1	1
0	0	$\cdots$	0	1	0	0
0	0	$\cdots$	0	1	0	1
0	0	$\cdots$	0	1	1	0
0	0	$\cdots$	0	1	1	1

Telle [2026] later showed the conjecture to be equivalent to stating that $H_{n,m}$ minimises the number of realisable teaching sets of each size over the family $\mathcal{C}_{n,m}$. Here $\mathcal{C}_{n,m}$ denotes all concept classes with n distinct concepts over m instances, represented as binary $n \times m$ matrices with distinct rows. The equivalent conjecture can be stated as:

$$\forall q, \forall C \in \mathcal{C}_{n,m}, \quad |W_r^{\leq q}(H_{n,m})| \leq |W_r^{\leq q}(C)| \tag{2.2}$$

So the number of realisable teaching sets of any concept class of size n by m is minimised, for any q, by the $H_{n,m}$ matrix.

In Paper (iii) we prove this conjecture (Equation 2.2) true. This result can be seen as a generalisation of a theorem resolving the edge isoperimetry problem for hypercubes [Hart, 1976]. For readers interested in the original formulation of the conjecture, we briefly describe it as follows.

Rather than counting realisable teaching sets directly, the paper considers unique row projections. For a fixed subset of q columns, the rows are projected onto those columns, and the number of distinct projected rows is counted. Summing this quantity over all subsets of q columns gives the total number of unique projections over all subsets of size q.

$$m_q(M) = \sum_{Q \in \binom{[1,m]}{q}} dif(M(Q)) \tag{2.3}$$

Here, *dif* counts the number of unique rows in the matrix $M(Q)$ obtained by restricting the original matrix M to the columns in Q. However, notice that for a fixed subset Q, each distinct row in $M(Q)$ corresponds to one realisable labelling of the teaching set formed by the examples in Q. Hence, the two viewpoints, of projection and realisable

teaching sets, count the same thing. As such, the conjecture solved in Paper (iii) can be written as Equation 2.4.

$$\forall q, \forall C \in \mathcal{C}_{n,m}, \quad m_q(H_{n,m}) \leq m_q(C) \tag{2.4}$$

Chapter 3

Example-based XAI in the domain of time series

In the previous chapter we focused on different aspects of machine teaching. In this chapter we shift focus to eXplainable AI (XAI), specifically example-based XAI. Rather than developing methods for an idealised learner, we now investigate how to explain AI systems to humans, primarily end-users interacting with the system without domain or AI expertise. Following the taxonomy of interpretability evaluation [Doshi-Velez and Kim, 2017], we conduct human-grounded evaluations, where real humans perform simplified tasks that approximate the intended use case. In particular, we use forward simulation, which tests the simulatability introduced in Chapter 1: participants are first presented with labelled training examples, possibly accompanied by an additional explanation, and then shown a new set of examples and asked to assign each example the label they believe the target AI would predict. All classification tasks used in our user studies are binary, except UMD in Paper (vi), which has three classes. Chance accuracy is therefore 50%, and 33% for UMD.

We will also limit the scope to temporal data. Temporal data is data collected over time, such as sensor readings or streaming data, represented as an ordered sequence of measurements [Esling and Agon, 2012]. In general, each measurement may be multi-dimensional, but in this thesis we focus on univariate time series, where each measurement is a single real value.

Temporal data is encountered in many real-world applications, ranging from healthcare [Rajkomar et al., 2018] to cyber security [Susto et al., 2018]. A central task for such data is time series classification [Bagnall et al., 2017]. Despite its prevalence, humans often struggle to interpret temporal data [Siddiqui et al., 2019]. This motivates our focus on

developing XAI methods for time series classification and evaluating their effectiveness with end-users.

As a concrete example, we briefly describe a real-world setting from the industry partner Eviny, whose dataset is used in Paper (iv). Imagine someone charging their new electric car at a charging station. During the session they are shown a curve of the power injected into the car each minute, and due to battery status and weather conditions this curve can look strange and differ from session to session. To reduce the number of calls from customers worried that something is wrong with their car, the charging company deploys an AI system that reads the charging curve and flags anomalies. This might help slightly; however, people are seldom happy without an explanation. If the company instead deployed an XAI system showcasing similar charging sessions from known healthy car batteries, the customer could be reassured directly.

3.1 Active learners in XAI

In contrast to the passive learners typically assumed in machine teaching, humans can be active learners. There has therefore been a push to rethink how users can interact with an XAI system, for instance allowing dialogue between users and the system, and to enable *model inspection*, where users can modify input examples and immediately observe how the model's prediction changes [Delaney et al., 2021; Theissler et al., 2022; Wang et al., 2023]. Motivated by this shift towards interactive explanations, Paper (iv) explores how such interactive model inspection can be realised for time series classification.

The goal is to allow users to actively construct and refine their mental model of the classifier by testing different hypotheses. However, time series are inherently high-dimensional and difficult for humans to understand, making unrestricted exploration difficult. Small edits may produce unrealistic examples or lead users to explore regions that provide little insight into the classifier's behaviour.

To support meaningful exploration, users require guidance that both keeps their modifications within realistic regions of the data space and helps them understand how those modifications affect the classifier's behaviour. Firstly, users should receive feedback on where realistic examples lie, so as to keep the exploration grounded, rather than drifting towards implausible time series. Secondly, users should receive feedback indicating how their modifications influence the classifier's prediction and whether they are approaching the decision boundary.

Paper (iv) implements these design principles in an interactive visual tool, shown in

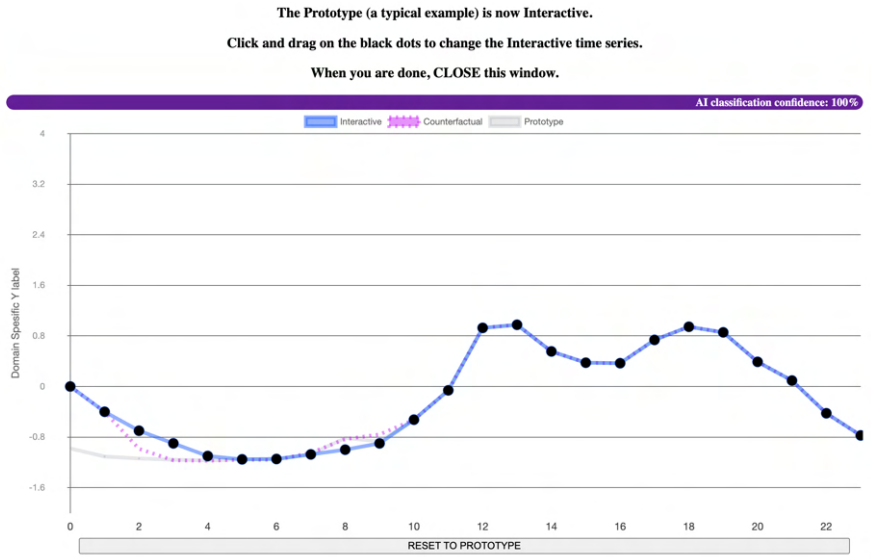


Figure 3.1: The interactive model inspection tool developed in Paper (iv), shown as it was presented during the user study with instructional text. The top displays the classifier's confidence in its current prediction. The main plot shows three time series: the Interactive series currently being edited by the participant, the Counterfactual series (displayed as a dotted line) representing an example from the opposite class, and the Prototype series (shown in grey), which served as the participants' starting point. As the participant modifies the interactive time series, the classifier's confidence, the colour of the interactive series (indicating its predicted class), and the counterfactual example are all updated in real time.

Figure 3.1. Users begin with representative prototype time series, providing a realistic starting point for exploration. The tool generates counterfactual examples using Native Guide [Delaney et al., 2021], thereby grounding the guidance in existing examples. At the same time, the classifier's prediction confidence is displayed to the user, allowing the user to infer when they are approaching the decision boundary. As the user modifies the time series, both the counterfactual and the prediction confidence are updated in real time, continually informing the user about realistic examples while revealing the location of the decision boundary.

Having developed this tool, the paper then asks whether providing these interactive capabilities improves users' understanding of the classifier. To answer this question, the tool was evaluated through a controlled user study.

The user study was conducted over three datasets: two from the UCR archive (ItalyPowerDemand and Chinatown) [Dau et al., 2019] and one from industry partner Eviny (electric vehicle charging curves). These datasets were selected based on several considerations, including their relatively short length (24 time steps, corresponding to 24 hours). The participants were split into three groups, two passive learning groups and

one active learning group.

- G_P: Only prototypes
- G_{P+CF}: Prototypes and counterfactuals
- $G_{P+CF+MI}$: Prototypes and counterfactuals with model inspection.

For each dataset, the participants first received training examples, alongside the explanation technique of their group. After training, the participants were shown new examples and asked what they believed the AI would label each instance as. The results in Table 3.1, somewhat surprisingly, show that model inspection (in this setup) did not improve the accuracy of the participants.

Table 3.1: Average accuracy over all datasets, by group.

Groups	G_P	G_{P+CF}	$G_{P+CF+MI}$
Accuracy over all datasets	$70.2\% \pm 14\%$	$65.9\% \pm 20\%$	$64.0\% \pm 20\%$

Based on the participants' response time and qualitative feedback, we will now highlight two possible explanations for the observed results. First, the average completion time was approximately five minutes per dataset across all groups, including both the training and testing phases. While this is sufficient for the passive learning groups, it raises the question of whether participants assigned to the model inspection group used sufficient time to meaningfully explore the interactive interface. Second, several participants reported that interacting with the tool was challenging. Despite deliberately selecting relatively short time series, 24 time steps may still have been too many to support effective interactive exploration by end-users.

Together, these observations suggest that interactive model inspection alone was not sufficient to improve users' understanding in our setting. While interaction may help users formulate and test hypotheses, the complexity of the time series itself remains a limiting factor.

3.2 A simplification algorithm in the context of XAI

Consider the binary time series classification problem illustrated in Figure 3.2, where the goal is to distinguish normal heartbeats from faulty heartbeats (Myocardial Infarction), using the ECG200 dataset [Olszewski, 2001]. At first glance, the time series is difficult to interpret due to numerous small fluctuations. A natural idea is therefore to

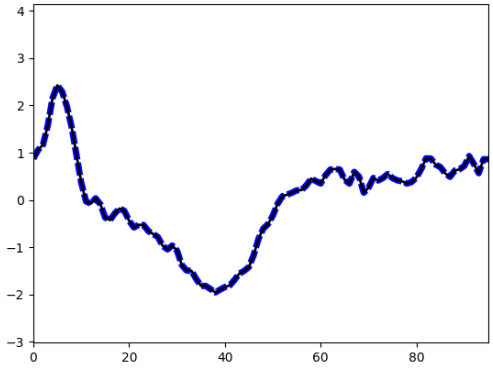


Figure 3.2: Original time series, class Blue.

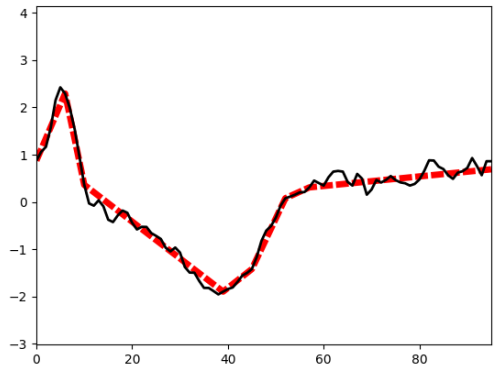


Figure 3.3: Simplified time series, class Red, overlaid on the original time series.

simplify the time series, removing unnecessary detail while preserving its overall shape. Indeed, many algorithms have been designed to compress, simplify, or segment time series [Douglas and Peucker, 1973; Esling and Agon, 2012; Keogh et al., 2004]. Figure 3.3 shows such a simplification: with far fewer data points, the resulting time series is easier to comprehend. However, when it is presented to the classifier, the predicted label changes. This indicates that the simplification has removed information that was important to the model. Although visually simpler, the resulting time series no longer provides a faithful explanation of the classifier's decision.

This issue is particularly problematic in example-based XAI, where explanations are constructed from labelled examples illustrating the behaviour of a classifier. Suppose we wish to use the labelled instance (x, b) as one such example, but first simplify it to obtain a new instance x'. If the classifier predicts a different label b' for x', then the explanation must either be presented as (x', b), contradicting the behaviour of the classifier, or as (x', b'), no longer conveying the same behaviour of the classifier that the original labelled example was selected to illustrate.

Based on these observations, it is clear that traditional simplification algorithms may be unsuitable when used for example-based XAI.

Inspired by the findings of Paper (iv), where the complexity of the prototype time series appeared to limit participants' understanding, Paper (v) investigated whether time series simplification could produce prototypes that are easier for humans to interpret.

For such simplifications to remain useful as explanations, they must first preserve the classifier's prediction, ensuring that they remain faithful representations of the original labelled example. However, preserving the prediction alone is not sufficient. Since prototypes are intended to represent the classifier's behaviour beyond a single observation,

they should correspond to a stable region of the decision space rather than isolated points whose label changes after only minor perturbations. Consequently, the simplifications should be robust to small perturbations, ensuring that they represent stable regions of the classifier's decision space.

While robustness is essential for explanations, it is not the only consideration. A useful simplification should still resemble the original time series, ensuring that the explanation reflects the observed data, while also reducing its complexity to improve human interpretability. Consequently, Paper (v) formulates time series simplification for example-based XAI as a multi-objective optimisation problem balancing robustness, similarity, and complexity.

In Paper (v), a simplification is defined by selecting a subset of the original time steps. The values at the selected time steps are retained, while the remaining values are reconstructed by linear interpolation and extrapolation. The optimisation therefore searches over all simplifications that can be generated in this way.

The paper formulates this as the optimisation problem $opt_{simp}(ts, h, \alpha, \beta, \gamma)$, where ts is the time series to be simplified and h the classifier:

$$opt_{simp}(ts, h, \alpha, \beta, \gamma): \quad \min\ \alpha \cdot err(ts, sts) + \beta \cdot k_{sts} + \gamma \cdot frag(ts, sts, h) \tag{3.1}$$

where the minimisation ranges over all simplifications sts satisfying $h(ts) = h(sts)$.

The first term, $err(ts, sts)$, measures the dissimilarity between the original and simplified time series. The second term, k_{sts}, measures the complexity of the simplification through its number of segments. Finally, $frag(ts, sts, h)$ measures the fraction of perturbations that change the classifier's prediction. Since fragility is the complement of robustness, minimising this term encourages simplifications that remain stable under small perturbations. The balancing parameters α, β, and γ allow the user to trade off similarity, complexity, and robustness depending on the intended application.

Although the search space is exponential in the number of time steps, Paper (v) exploits the structure of the optimisation problem to avoid an exhaustive search. The similarity and complexity terms decompose over the segments of a simplification and can be optimised directly by dynamic programming, whereas robustness requires querying the black-box classifier under perturbations and cannot be folded into the dynamic programming. The algorithm therefore proceeds in two steps: for a user-set parameter q, dynamic programming is first used to find the q least costly simplifications with respect to similarity and complexity; robustness is then evaluated only for these q candidates,

and the best overall is returned. This runs in time $O(n \cdot \max\{n^2, q \log n\})$.

Because robustness is examined only for the top q candidates, a simplification ranked beyond position q could in principle be better overall, its worse similarity and complexity offset by unusually good robustness. Paper (v) bounds exactly this risk: since robustness contributes at most γ to the total cost, if the gap in similarity and complexity cost between the best candidate and the qth best candidate is at least γ, no unexamined simplification can overtake the best, and the returned simplification is guaranteed to be globally optimal[1]. The user can therefore set q higher or lower, tightening the bound at the cost of higher run time.

Finally, the algorithm was evaluated on the UCR datasets Chinatown, ECG200, and ItalyPowerDemand. Rather than conducting a user study, the experiments served as a proof of concept demonstrating that robust simplifications can reveal concise, human-interpretable rules-of-thumb describing the classifier's behaviour. These results suggest that robust simplifications are a promising explanation technique, while leaving the question of whether they improve human understanding to future work.

3.3 Evaluating simplification algorithms in the context of XAI

The previous paper demonstrated that simplifying time series can produce concise, faithful prototype-based explanations. However, increasing the degree of simplification inevitably removes information from the original time series. While simpler explanations may be easier for humans to understand, overly aggressive simplifications risk changing the classifier's prediction, causing the explanation to no longer reflect the behaviour of the model.

This naturally raises the question of where the balance between simplicity and prediction preservation lies. Unlike data compression, where the objective is simply to preserve the original signal, simplifications for explainability must preserve the classifier's decision. Consequently, the usefulness of a simplification depends not only on the simplification algorithm itself, but also on the particular classifier being explained and the characteristics

[1] In Paper (v) this guarantee is settled before the classifier is queried for robustness. The same argument yields a sharper certificate, not stated in the paper: the returned simplification is globally optimal whenever its total cost is at most the similarity-and-complexity cost of the qth candidate – any simplification outside the top q already costs at least that much in similarity and complexity alone, hence at least that much in total. Since the algorithm computes robustness regardless, this is a free $O(1)$ check, potentially certifying optimality even when the guarantee stated in the paper does not apply.

of the underlying dataset.

Inspired by the observation in Paper (v) that a useful simplification should preserve the classifier's prediction, Paper (vi) introduces the notion of loyalty. Loyalty measures the fraction of time series in a dataset that receive the same prediction after simplification, and thus takes values in $[0, 1]$, where a loyalty of 1 means every prediction in the dataset survives simplification.

Since simplifications should also reduce the cognitive burden on the user, the paper further introduces a complexity metric, measuring the fraction of the original time series retained after simplification. Together, these metrics capture the trade-off between loyalty to the classifier and simplicity of the explanation, enabling systematic evaluation of simplification algorithms across different classifiers and datasets.

Using these two metrics, Paper (vi) evaluates four simplification algorithms across a diverse collection of time series datasets and classifiers. By varying the amount of simplification, each algorithm produces a *loyalty–complexity curve*, illustrating how much loyalty is sacrificed in exchange for simpler explanations. These curves make it possible to compare simplification algorithms independently of a particular parameter setting, while also revealing how the trade-off varies across different classifiers and datasets.

To investigate whether these theoretical metrics correspond to improvements in human understanding, the paper performs a human-grounded evaluation using forward simulation. Four representative operating points are selected from the loyalty–complexity curves: the original time series without simplification, the most aggressive simplification that maintains perfect loyalty (L1.0), and two intermediate operating points referred to as *Knee High* and *Knee Low*. Participants were divided into four groups using a Latin-square design, ensuring that each participant evaluated every dataset once and each simplification level once. Since what a participant learns from a simplification with loyalty λ carries over to the original time series only a λ fraction of the time, the measured accuracies are multiplied by the corresponding loyalty. The results are shown in Table 3.2.

The evaluation showed that simplifications do not universally improve interpretability. For some datasets, the original time series enabled the highest participant accuracy, while for others the simplified representations performed equally well or better. Rather than identifying a single best level of simplification, the results demonstrated that the usefulness of simplifications depends on the interaction between the classifier, the dataset, and the degree of simplification. Although the user study exhibited considerable variation between participants, the observed trends were consistent with the proposed loyalty and complexity metrics.

Table 3.2: Loyalty-adjusted participant accuracy in the forward simulation study, by dataset and simplification level: measured accuracy multiplied by the loyalty of the corresponding simplification, so the figures express understanding of the classifier on the *original* time series. Reproduced from Paper (vi); per-participant variance was substantial, and the unadjusted accuracies are given there.

Dataset	Knee Low	Knee High	L1.0	Original
Chinatown	56.6%	75.9%	76.0%	66.0%
ECG200	55.8%	69.8%	72.0%	72.5%
SonyAIBORobotSurface1	60.8%	69.1%	76.0%	78.0%
UMD	93.1%	99.0%	100.0%	96.0%

The theoretical analysis and user study were finally combined into a practical decision framework for determining when simplifications are likely to improve interpretability. Rather than recommending a particular simplification algorithm, the framework guides the practitioner through a sequence of considerations, including whether the original data is already easy to interpret, whether sufficiently loyal simplifications can be obtained, and whether the simplified representations remain understandable to humans. This framework constitutes the main contribution of the paper, providing practical guidance for deciding when simplifications should be used as explanations of time series classifiers.

Chapter 4

Bridging XAI and machine teaching

In the previous chapters, we have examined problems related to machine teaching itself and example-based XAI for time series. In this chapter, we bring these two research directions closer together. We first investigate how machine teaching can provide insight into behaviours of black-box models. Since their internal representations are inaccessible, we use relative teaching size as an external probe of whether concept difficulty is stable across representations. We then reverse the perspective and ask how standard assumptions in machine teaching can be challenged to better reflect the XAI setting.

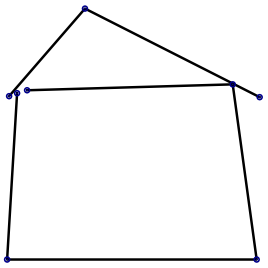
4.1 Comparing concept complexity across modalities with machine teaching

Many modern large language models are, under the hood, vision-language models (VLMs), capable of processing both images and text [Achiam et al., 2023; Team et al., 2024]. Consequently, the same concept – for example, a house – can be presented either as a bitmap image or as a textual description of a drawing, as illustrated in Figure 4.1.

This raises the question of whether the relative identification complexity of concepts for these VLMs is tied to their representation, or whether some concepts are intrinsically easier to identify than others. If concept complexity is intrinsic, then concepts that are easy to identify in one modality should remain comparatively easy in another. Since the internal representations learned by modern vision-language models are inaccessible, this hypothesis is difficult to investigate directly.

Paper (vii) approaches this question using machine teaching. Rather than analysing the

Image:



Text:

```
\draw (10, 169) -- (0, 0) -- (250, 0)
    -- (226, 178) -- (20, 172);
\draw (2, 166) -- (78, 255) -- (253,
    165);
```

Figure 4.1: Example of how the concept 'House' can be presented to a VLM in two distinct modalities. Adapted from Paper (vii).

model's internal representations directly, the paper measures how much information is required for reliable concept identification using the notion of teaching size [Telle et al., 2019]. In Paper (vii), the teaching size is defined as the number of segments.

The experiments are based on drawings of twenty concepts from the Quick, Draw! dataset [Jongejan et al., 2016]. Each drawing is repeatedly simplified using the Ramer–Douglas–Peucker (RDP) algorithm [Douglas and Peucker, 1973; Ramer, 1972], producing progressively simpler versions of the same concept. These simplified drawings are then presented to six different VLMs in two modalities: either as bitmap images or as TikZ descriptions encoding the same sequence of line segments. For each concept, model, and modality, the smallest representation that remains reliably recognised (defined as at least 50% accuracy at temperature 1) is identified, allowing the teaching size of the concept to be estimated.

The results show that, under the experimental setting considered, concepts generally require less information to be identified from bitmap images than from textual coordinate descriptions. Indeed, eight of the twenty concepts were never identified from the coordinate representation by any model. However, despite this difference in absolute teaching size, the relative ordering of concept complexity is largely preserved across modalities, though with variation across models. Concepts that require comparatively little information in one representation also tend to require comparatively little information in the other. These findings support the hypothesis that some concepts exhibit an intrinsic notion of complexity that is, at least to some extent, independent of the representation used.

4.2 On imperfect learners in machine teaching

In the formal machine teaching models considered earlier in this thesis, the learners have been assumed to be perfect. However, in this latest work with VLMs (Paper (vii)), we have what can be referred to as imperfect learners. Given an image and tasked with identifying it, the VLMs are not perfect, and will fail stochastically. Humans, for instance the participants from the user studies in Chapter 3, are also imperfect in a similar way: they fail on some consistency checks, and can do so stochastically.

Can we model how these imperfect learners fail, and by taking this into account find better ways of teaching them? To do so, we have to make assumptions about where the learners can fail. We distinguish two stages at which errors may arise: checking whether labelled examples are consistent with candidate concepts, and using the resulting consistency information to infer the target concept. We call errors in the first stage deductive errors, and errors in the second inductive errors. Thus far, most work in machine teaching has assumed there to be no deductive error, i.e. the learner has a perfect consistency-checking oracle. However, for imperfect learners, the consistency check, i.e. checking, for a given concept c and labelled example (x, b), whether $c(x) = b$, can fail.

Paper (viii) explores this problem, and defines the deductive error rate of a consistency check as $\gamma_L(c, x)$, for a learner L, concept c and example x. To illustrate why this error depends on both the concept and the example, we take inspiration from the work on time series in Papers (iv)–(vi). Consider two candidate concepts: c_1, true if the first time step is above zero, and c_2, true if the global maximum lies in the first half. Checking c_1 is easy for any time series, since only the first time step matters. Checking c_2, however, depends on the example: for a slowly and monotonically rising time series one can conclude at a glance that the answer is false, while for a time series with two bumps of similar height, one in each half, the check becomes error-prone.

As these learners fail, and do so stochastically, we cannot generally guarantee that we can teach the target concept. However, we might increase the likelihood of the learner identifying the target concept, given a good teaching set. Moreover, if there are many similar concepts, uniquely identifying the target concept might be difficult, but we might be able to ensure a high likelihood of teaching an approximately similar concept. We take inspiration from the Probably Approximately Correct (PAC) learning framework [Valiant, 1984], where the goal is to learn an approximately correct concept with high probability. In PAC learning, the stochasticity comes from random sampling of examples;

in our case, it comes from deductive errors. We thus define *PAC teaching*:

$$P[sim(L(S), c^*) \geq q] \geq p \tag{4.1}$$

For a given probability p and approximation ratio q, the goal is for the teacher to find a teaching set S such that there is a probability of at least p that the learner selects a concept $L(S)$ that is at least q-similar to the target concept. Here, $sim(c, c^*)$ measures the fraction of examples on which the two concepts agree. We also distinguish between the learner *identifying* the target concept and the learner *simulating* it: $sim(c, c^*)$ is used in the identification scenario (the raw similarity between c and c^*), while $sim_L(c, c^*)$ is used in the simulatability scenario (the expected similarity when the learner *employs* c with error). In the latter case, we assume the same deductive error rate during employment as during teaching.

In Paper (viii) we model the situation of learners with deductive errors. We present two learners: a *naive* learner and a *prudent* learner. In short, the naive learner acts as if there is no error, excluding any concept not deemed consistent with the teaching set. They then return an arbitrary remaining concept, or report failure if none remain. The prudent learner, on the other hand, is aware that errors happen, but is not calibrated on which example-concept pairs they typically make mistakes on (the teacher is). The prudent learner keeps a count of how many times they believe each concept agreed with the labelled examples in a teaching set, and selects the concept with the highest count after teaching.

We investigate three natural teaching scenarios. Firstly, for a given p and q, the teacher is tasked with minimising the size k of the teaching set that satisfies PAC teaching. Secondly, for a given p and k (maximum size of the teaching set), find the teaching set S that maximises q. Thirdly, for a given q and k, find the teaching set S that maximises p. For each scenario, the paper gives an algorithm and a matching hardness result: the last two problems are W[2]-hard [Downey and Fellows, 2013] when parametrised by the teaching set size k, while size minimisation admits no $2^{o(|X|)}$-time algorithm under the Exponential Time Hypothesis (ETH) [Impagliazzo and Paturi, 2001].

Finally, we test these ideas empirically on an LLM. We wanted to check if using teaching sets containing examples with low deductive error would correlate with low teaching error. For this, we defined a concept class over multiples of d, where each concept c_d contains all multiples of d, for $d \in \{5, 7, 11, 13, 17\}$ – distinct primes, so that no concept contains another. These values were deliberately selected to have one simple concept, c_5 (check last digit), and several harder concepts ($c_7, c_{11}, c_{13}, c_{17}$). The deductive errors were estimated by asking the LLM '*Is d a divisor of x?*', and measuring the error, thus

simulating the consistency checks needed later. We then used these examples to create teaching sets and asked '*Which of 5, 7, 11, 13 and 17 is a divisor of x?*'. The experiments used GPT-5-nano, chosen because stronger models perform these consistency checks near-perfectly, leaving little error signal to study. The results show that examples with low deductive error consistently give better performance on the teaching problem.

Chapter 5

Discussion, future directions and conclusion

In this thesis, we have investigated how machine teaching – a formal theory of teaching by examples – can serve as a theoretical grounding for example-based XAI. Framing the XAI system as the teacher, the human as the learner, and the AI model as the target concept, the selection of explanatory examples becomes a teaching problem.

On the machine teaching side, we adapted the framework to the XAI setting and found that machine-teaching-generated explanations helped participants understand a target concept better than randomly selected examples of comparable complexity – while also finding that success depends critically on the alignment between the learner model L_M and the human learner L_H (Paper (i)). We extended the formal setting to concept classes with redundant representations (Paper (ii)), and proved the conjecture posed there, establishing which concept classes minimise the number of realisable teaching sets and thereby giving a lower bound on the teaching dimension for any learner (Paper (iii)).

In the domain of time series, we developed an interactive model inspection tool and evaluated it in a user study, finding, somewhat surprisingly, that the interaction did not improve participants' understanding (Paper (iv)). Motivated by the complexity of the time series themselves, we formulated simplification for example-based XAI as an optimisation problem balancing similarity, complexity, and robustness, and gave an efficient algorithm for it (Paper (v)). We then introduced the loyalty and complexity metrics for evaluating simplification algorithms, and combined them with a user study into a practical decision framework for when simplifications should be used as explanations (Paper (vi)).

Finally, bridging the two fields, we used teaching size to compare concept complexity

across modalities in vision-language models, finding support for the hypothesis that some concepts are intrinsically easier to identify than others (Paper (vii)), and we replaced the standard assumption of perfect learners with stochastically failing ones, defining PAC teaching and providing algorithms, hardness results, and an empirical validation on an LLM (Paper (viii)).

5.1 Discussion

Across the eight papers, one theme recurs more than any other: every method for selecting explanatory examples rests on an assumption about how the human learns, whether or not that assumption is ever written down. The distinctive move of machine teaching is to make it explicit. The teacher optimises against a learner model L_M, and the quality of the resulting explanation is bounded by how well L_M describes the human learner L_H.

Seen this way, the work in this thesis tests several different models of the learner. Paper (i) uses an explicit one: a concept class of Boolean expressions together with a preference over them. Paper (iv) assumes a learner who refines a mental model through active hypothesis testing, and provides the means to do so. Papers (v) and (vi) assume a learner whose error grows with the complexity of what they are shown, and act on the representation rather than on the selection. Paper (viii) relaxes the assumption that learners deduce perfectly.

It is worth being precise about the simplification case. Simplification is not used to generate a concept class; it never feeds back into which examples are selected. It is an assumption at the level of representation: that humans reason more easily over a simpler description of a time series than over its raw values. Loyalty is then the constraint that keeps this change of representation honest with respect to the classifier, preventing a simpler picture from becoming a different picture.

Therefore, when we tested model inspection and simplification and found that they did not improve participants' accuracy, and in some cases reduced it, what failed was an assumption about the learner, not a method for selecting examples. In these studies prototypes stood in for the teaching set, and what varied between conditions was the representation, or the means of interacting with it. The difficulty this exposes – knowing which representation a human finds easiest – is the same difficulty this thesis identifies as central.

Where the machine teaching example selection itself was tested, it worked. In Paper

(i), teaching sets selected by machine teaching gave higher participant accuracy than a random baseline of matched complexity. In Paper (viii), teaching sets built from examples with low deductive error produced lower teaching error in an LLM learner than those selected without regard to it.

As is standard in machine teaching, the number of examples needed for teaching depends on both the learner and the concept class. The theoretical work of this thesis identifies which structural properties carry this dependence. In Paper (ii), the difference between Eager and Greedy is governed less by how much redundancy the representation class contains than by its redundancy spread – how the equivalent representations are distributed through the representation ordering. In Paper (iii), we prove that among all concept classes with n concepts over m instances, $H_{n,m}$ minimises the number of realisable teaching sets of every size, and therefore maximises the lower bound on the teaching dimension that holds for any learner.

Three questions remain genuinely open. The first is whether any tractable learner model captures human reasoning well enough to optimise against. The second is whether large language models can serve as proxies for humans, either as the learner model itself or as substitutes for participants when comparing explanation methods. The third is more fundamental, and concerns whether simulatability tracks understanding at all. We turn to this last question first.

5.2 Limitations

The most significant limitation of this thesis is conceptual. Throughout, understanding has been operationalised as simulatability, and measured through forward simulation. There are good reasons for this: it is measurable, and it measures understanding in the users of these tools rather than a property of the tools themselves. However, in accordance with Goodhart's law, a metric may cease to be a good one once it becomes the target. Let us see why simulatability might not be enough.

Firstly, simulatability is uniform; it only cares about how well you can predict the model's output on average. It may be more important to correctly identify the examples on which the model disagrees with the ground truth (risk analysis), or to correctly identify when the model will produce a signal for a minority class (anomaly detection). Simulatability does not capture these important aspects of understanding.

Secondly, prediction does not require comprehension. A person may predict the model's outputs by pattern matching alone, without any causal or transferable account of why

the model behaves as it does. High simulatability is therefore possible without the kind of understanding one may actually care about.

Some operational target must be chosen, and we consider simulatability the right one; but it should be read as exactly that – an operational target, not a definition of understanding.

Another important acknowledgement is that machine-teaching-selected examples were never compared against the classical methods of example-based XAI. The reason is that our domains fell on either side of a divide: either the domain was constructed for the machine teaching setting, with the concept class known by construction, or it was practical, and no method existed to extract a good concept class from it. In the latter case, prototypes played the role of the teaching set when we tested model inspection and simplification. This comparison is the most direct test the work still lacks, and the automatic construction of concept classes discussed in Section 5.3 is its prerequisite.

The user studies in this thesis also carry the limits inherent to their format: large variation between participants, limited sample sizes, and short completion times. In Paper (iv), the cost of interaction additionally scaled with the length of the series, since each time step had to be edited in turn. We refer to the individual papers for a full account.

Furthermore, the algorithm proposed in Paper (v) has never been tested in a user study. Hence, its usefulness has not been verified, despite the informative rules of thumb identified. It is also worth noting that even if ORS is confirmed to be beneficial in user studies, this might not be due to the simplification itself, but to the shift it induces towards more stable, prototypical instances.

We now turn our focus forward to what could be done next. We discuss the directions we consider most promising: modelling the human learner, interactive and adaptive teaching, and algorithmic aspects of our simplification framework.

5.3 Modelling the human learner

How to model the human learner is a question that has run through this thesis. In Paper (i) we saw that misalignment between the learner model L_M and the human learner L_H is problematic; there, L_M was a preference-based learner over a handcrafted concept class. Papers (vii) and (viii) instead used LLMs as learners, either directly (Paper (vii)) or as experimental support (Paper (viii)).

Broadly, the learner model can be viewed as a surrogate model of the human – the mirror image of XAI's surrogate models of AI systems. Learner models then fall on a spectrum, from version space learners over handcrafted concept classes to fully learned surrogates of the human. We discuss ideas along this range.

Instead of handcrafting the concept class, we could automatically extract it. For the class to remain human-like, its concepts should be simple combinations of human-interpretable features. This follows the work of Singla et al. [2014], where human-interpretable features are obtained through crowdsourcing and linear combinations of them form the candidate concepts. Extracting such features is necessarily domain-specific; for time series – the most explored practical domain of this thesis – a natural candidate is shapelets: short continuous segments that discriminate between classes by approximate containment [Ye and Keogh, 2011]. Concepts would then be short rules over these features.

Another natural approach would be to use LLMs to generate candidate rules or explanations of the AI system's behaviour [Martí-Pérez et al., 2026], which could then form the basis of a concept class. This way, what constitutes human-interpretable features and their combinations would be determined by the LLM.

A third option to explore is to train a surrogate model directly. Given data from a forward simulation user study, one could train a model on the different teaching sets and use the users' resulting answers as labels.

However, LLMs may already be good surrogate models of humans in terms of simulatability, mitigating the need to train dedicated surrogate models.

This opens two distinct uses. The first is to use an LLM directly as the learner model L_M: the teacher optimises teaching sets against the LLM, and the best set found is presented to the human. Since the teacher only ever selects the best set, the critical question is: *are the teaching sets that are best for an LLM also good for humans – and which LLMs are the best surrogates for which domains, problems, and target users (end-users or experts)?* The second use is in evaluation: forward simulation user studies are costly, and LLMs could stand in for participants when comparing explanation methods, reserving human studies for the most promising candidates. Here the LLM must be faithful across the whole range: *from poor to good teaching sets, does simulatability for the LLM correlate with simulatability for humans?*

Since some of the methods described in this section could produce huge concept classes containing many similar – or even equivalent, as in Paper (ii) – candidates, unique identification might require a very large teaching set. The natural remedy is instead to use PAC teaching, as introduced in Paper (viii), in which the aim is to teach an

approximately similar concept with high probability.

However, the concept class is only part of modelling the learner – assuming one does not use full surrogates, such as LLMs. The prior over the concepts is another aspect that needs to be modelled, and how to do this well should also be studied, although we do not provide concrete ideas on how to do so.

5.4 Interactive and adaptive teaching

Paper (iv) took a first step towards interactive teaching: the teacher reacts to the learner's current example by generating a new counterfactual. Teaching of this kind has been studied for real-world categories: Johns et al. [2015] teach humans to classify images by maintaining a probabilistic model of the student, updating it after each response, and choosing the next image accordingly, and Singla et al. [2014] address a related problem with crowd workers. What remains open is the same loop when the target concept is a classifier's decision function rather than a ground-truth category, where labels are free but the concept class to update beliefs over is precisely what is missing, as discussed in Section 5.3.

Concretely, such a loop could alternate between teaching, exploring and quizzing. First, the learner receives a teaching set from the teacher. Second, the learner explores the model based on their current mental model, a step which has no counterpart in the loops described above. Third, the teacher gives a quiz of unlabelled examples: forward simulation, used here as a feedback signal rather than as the evaluation instrument it has been throughout this thesis. The learner's answers reveal their current hypothesis and where it errs, and the teacher uses this feedback to select the next teaching set. The cycle repeats until the learner reaches a satisfactory quiz score. Notably, in Paper (iv) we found that counterfactual guiding alone did not improve participants' understanding; a guided loop of this kind is where interaction may yet pay off.

5.5 Algorithmic aspects

The ORS algorithm of Paper (v) has two steps: first, dynamic programming generates the q least costly candidate simplifications with respect to similarity and complexity; second, the robustness of each candidate is scored and the overall best is selected. For long time series, Paper (v) observed that even with $q = 10^6$, the candidates returned by the first step are nearly identical to one another: a counting argument shows that

the number of ways to place segment endpoints in the immediate vicinity of the best simplification can alone exceed q. Consequently, increasing the weight γ on robustness cannot change the outcome, and the optimality guarantee of the algorithm – which requires the cost difference between the best and the qth candidate to be at least γ – applies only for very small γ. The paper therefore suggested, as future work, a heuristic that discards marginally different simplifications during the dynamic programming.

Here we sketch this direction in more detail. Pruning near-identical candidates makes the qth candidate genuinely different from the best, and thus directly widens the range of γ for which the optimality guarantee applies. The remaining question is what discarding a candidate may cost. A natural hope is to bound this through overlapping perturbations: if the perturbations of two near-identical simplifications largely coincide, their robustness scores cannot differ by much. We explored this direction, but ran into an obstacle: perturbations are anchored at the pivot points of a simplification, and since near-identical simplifications typically select different pivot points, their perturbation sets are, by definition, disjoint.

One way forward would be to relax the perturbation model to also shift pivot points along the time axis, so that the perturbation sets of similar simplifications genuinely intersect and the bounding argument can be revived. Alternatively, one could forgo worst-case bounds altogether: we believe that sufficiently similar simplifications will tend to have similar robustness scores, and quantifying this empirically could justify pruning heuristics directly.

5.6 Conclusion

This thesis set out to evaluate machine teaching as a foundation for example-based XAI. Where it was applied, it worked. Machine-teaching-selected examples helped participants understand a target concept better than a random baseline of comparable complexity (Paper (i)). Used as an instrument rather than as an explanation method, teaching size measured the identification complexity of concepts across modalities in vision-language models, and found the relative ordering largely preserved (Paper (vii)). Extending the theory to learners that fail stochastically produced teaching sets that measurably reduced error in an LLM learner (Paper (viii)). On the theoretical side, we identified some structural properties of the concept class that influence the cost of teaching: redundancy spread (Paper (ii)) and the worst-case class $H_{n,m}$ (Paper (iii)).

What proved unreliable was not the machinery for selecting examples, but the assump-

tions about the learner that such selection is optimised against. Even in Paper (i), where the learner model was explicit, participants attended to features it did not represent; selection still helped, because the model was close enough to be worth optimising against. The assumptions that did not survive were of a different kind: that a particular interface would prompt active hypothesis testing, or that a simpler representation would be easier to reason over. These are claims about what people find easy, and we had no way to check them before running the study.

The binding constraint is therefore the learner model, and it cannot be removed. Without prior information about a particular person, what they will infer from a set of examples cannot be predicted with certainty. But it does not need to be. It needs only to be approximated well enough to be useful, and there are two routes to that: better surrogates of the population, pursued in Section 5.3, or information gathered from the learner during teaching itself, pursued in Section 5.4.

Despite this limitation, the case for the approach holds. Machine teaching is useful for example-based XAI when the learner model, the representation, and the evaluation target are all credible, and its contribution is that it forces the learner model to be stated. The question it puts at the centre of example-based XAI is not which examples to show, but whom we are teaching.

Bibliography

J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, et al. GPT-4 technical report, 2023.

A. Bagnall, J. Lines, A. Bostrom, J. Large, and E. Keogh. The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 31(3):606–660, 2017. doi: 10.1007/s10618-016-0483-9. URL `https://doi.org/10.1007/s10618-016-0483-9`.

A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 58:82–115, 2020. ISSN 1566-2535. doi: https://doi.org/10.1016/j.inffus.2019.12.012. URL `https://www.sciencedirect.com/science/article/pii/S1566253519308103`.

J. Bien and R. Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403 – 2424, 2011. doi: 10.1214/11-AOAS495. URL `https://doi.org/10.1214/11-AOAS495`.

H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh. The ucr time series archive. *IEEE/CAA Journal of Automatica Sinica*, 6(6):1293–1305, 2019. doi: 10.1109/JAS.2019.1911747.

E. Delaney, D. Greene, and M. T. Keane. Instance-based counterfactual explanations for time series classification. In *Case-Based Reasoning Research and Development - Int. Conference, ICCBR 2021*, pages 32–47. Springer, 2021.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *CoRR*, abs/1702.08608, 2017.

D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

P. Esling and C. Agon. Time-series data mining. *ACM Comput. Surv.*, 45(1), Dec. 2012. ISSN 0360-0300. doi: 10.1145/2379776.2379788. URL https://doi.org/10.1145/2379776.2379788.

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi. A survey of methods for explaining black box models. *ACM Comput. Surv.*, 51(5), Aug. 2018. ISSN 0360-0300. doi: 10.1145/3236009. URL https://doi.org/10.1145/3236009.

D. Gunning and D. W. Aha. Darpa's explainable artificial intelligence program. *AI Magazine*, 40(2):44–58, 2019. doi: https://doi.org/10.1609/aimag.v40i2.2850. URL https://onlinelibrary.wiley.com/doi/abs/10.1609/aimag.v40i2.2850.

S. Hart. A note on the edges of the n-cube. *Discrete Mathematics*, 14(2):157–163, 1976.

R. Impagliazzo and R. Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001. ISSN 0022-0000. doi: https://doi.org/10.1006/jcss.2000.1727. URL https://www.sciencedirect.com/science/article/pii/S0022000000917276.

E. Johns, O. Mac Aodha, and G. J. Brostow. Becoming the expert – interactive multi-class machine teaching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2616–2624, 2015.

J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. The Quick, Draw! - A.I. https://quickdraw.withgoogle.com/, 2016. Accessed: 2024-07-08.

E. Keogh, S. Chu, D. Hart, and M. Pazzani. Segmenting time series: A survey and novel approach. In *Data Mining in Time Series Databases*, pages 1–21. World Scientific, 2004. doi: 10.1142/9789812565402_0001. URL https://www.worldscientific.com/doi/abs/10.1142/9789812565402_0001.

B. Kim, R. Khanna, and O. O. Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/5680522b8e2bb01943234bce7bf84534-Paper.pdf.

F. Martí-Pérez, C. Bontveit, C. Ferri, and J. A. Telle. LLMs as verbal translation layer for TSC explanations. Manuscript, 2026. URL `https://xai.w.uib.no/files/2026/06/LLM_rules_for_TSC-15.pdf`.

C. Molnar. *Interpretable machine learning*. Lulu. com, 2020.

R. T. Olszewski. *Generalized feature extraction for structural pattern recognition in time-series data*. Carnegie Mellon University, 2001.

A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al. Scalable and accurate deep learning with electronic health records. *NPJ digital medicine*, 1(1):1–10, 2018.

U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. ISSN 0146-664X.

M. T. Ribeiro, S. Singh, and C. Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Y. Rong, T. Leemann, T.-T. Nguyen, L. Fiedler, P. Qian, V. Unhelkar, T. Seidel, G. Kasneci, and E. Kasneci. Towards human-centered explainable ai: A survey of user studies for model explanations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(4):2104–2122, 2024. doi: 10.1109/TPAMI.2023.3331846.

S. A. Siddiqui, D. Mercier, M. Munir, A. Dengel, and S. Ahmed. Tsviz: Demystification of deep learning models for time-series analysis. *IEEE Access*, 7:67027–67040, 2019. doi: 10.1109/ACCESS.2019.2912823. URL `https://doi.org/10.1109/ACCESS.2019.2912823`.

H. U. Simon and J. A. Telle. Matching-based parameters in machine teaching: a unified framework and hierarchy. Manuscript, 2026. URL `https://xai.w.uib.no/files/2026/05/Simon_Telle-Matching-based_Parameters.pdf`.

A. Singla, I. Bogunovic, G. Bartok, A. Karbasi, and A. Krause. Near-optimally teaching the crowd to classify. In E. P. Xing and T. Jebara, editors, *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 154–162, Beijing, China, 22–24 Jun 2014. PMLR. URL `https://proceedings.mlr.press/v32/singla14.html`.

G. A. Susto, A. Cenedese, and M. Terzi. Time-series classification methods: Review and applications to power systems data. *Big data application in power systems*, pages 179–220, 2018.

G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

J. A. Telle, J. Hernández-Orallo, and C. Ferri. The teaching size: computable teachers and learners for universal languages. *Machine Learning*, 108(8):1653–1675, 2019. doi: 10.1007/s10994-019-05821-2. URL `https://doi.org/10.1007/s10994-019-05821-2`.

A. Theissler, F. Spinnato, U. Schlegel, and R. Guidotti. Explainable AI for time series classification: A review, taxonomy and research directions. *IEEE Access*, 10:100700–100724, 2022.

L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

J. van der Waa, E. Nieuwburg, A. Cremers, and M. Neerincx. Evaluating XAI: A comparison of rule-based and example-based explanations. *Artificial Intelligence*, 291: 103404, 2021.

S. Wachter, B. Mittelstadt, and C. Russell. Counterfactual explanations without opening the black box: Automated decisions and the gdpr. *Harv. JL & Tech.*, 31:841, 2018.

Z. J. Wang, J. W. Vaughan, R. Caruana, and D. H. Chau. GAM coach: Towards interactive and user-centered algorithmic recourse. In *Proc. Conf. on Human Factors in Computing Systems,*, pages 835:1–835:20. ACM, 2023.

L. Ye and E. Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1): 149–182, 2011. doi: 10.1007/s10618-010-0179-5. URL `https://doi.org/10.1007/s10618-010-0179-5`.

X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. An overview of machine teaching, 2018. URL `https://arxiv.org/abs/1801.05927`.

Declaration on the use of generative AI

As this thesis concerns explainable AI, it is perhaps fitting that the author has also gained first-hand experience as a user of AI systems – for better or worse. While these tools sometimes saved time, on other occasions they made the work take longer without producing an obviously better result.

For transparency, the following declaration uses the categories published by the Faculty of Science and Technology at the University of Bergen as a framework for describing how generative AI was used in the synthesis of this thesis:

Category A (brainstorming and topic exploration): Claude, ChatGPT, Perplexity, and Gemini were used for literature search and exploration of topics and related work.

Category F (interpreting and discussing academic content): Claude and ChatGPT were used as discussion partners over the author's own material – to discuss and structure content, and to explore framings and formulations, which the author adopted where appropriate.

Category B (language editing): Claude and ChatGPT were used to improve the language and readability of the author's text.

In all cases, the text and ideas have been revised and fact-checked, and only included by decision of the author, who takes full responsibility for the content of this synthesis. No part of the synthesis consists of AI-generated content beyond the uses declared above.

Part II

Scientific Results

Paper I

Håvardstun, B., Ferri, C., Hernández-Orallo, J., Parviainen, P., & Telle, J. A.
Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2023), LNCS **14171**, pp. 378–393 (2023)
DOI: 10.1007/978-3-031-43418-1_23

XAI with Machine Teaching When Humans Are (Not) Informed About the Irrelevant Features

Brigt Arve Toppe Håvardstun[1( )], Cèsar Ferri[2], Jose Hernández-Orallo[2], Pekka Parviainen[1], and Jan Arne Telle[1]

[1] Department of Informatics, University of Bergen, Bergen, Norway
{brigt.havardstun,pekka.parviainen,jan.arne.telle}@uib.no
[2] VRAIN, Universitat Politècnica de València, Valencia, Spain
cferri@dsic.upv.es, jorallo@upv.es

Abstract. Exemplar-based explainable artificial intelligence (XAI) aims at creating human understanding about the behaviour of an AI system, usually a machine learning model, through examples. The advantage of this approach is that the human creates their own explanation in their own internal language. However, what examples should be chosen? Existing frameworks fall short in capturing all the elements that contribute to this process. In this paper, we propose a comprehensive XAI framework based on machine teaching. The traditional trade-off between the fidelity and the complexity of the explanation is transformed here into a trade-off between the complexity of the examples and the fidelity the human achieves about the behaviour of the ML system to be explained. We analyse a concept class of Boolean functions that is learned by a convolutional neural network classifier over a dataset of images of possibly rotated and resized letters. We assume the human learner has a strong prior (Karnaugh maps over Boolean functions). Our explanation procedure then behaves like a machine teaching session optimising the trade-off between examples and fidelity. We include an experimental evaluation and several human studies where we analyse the capacity of teaching humans these Boolean function by means of the explanatory examples generated by our framework. We explore the effect of telling the essential features to the human and the priors, and see that the identification is more successful than by randomly sampling the examples.

1 Introduction

In the field of eXplainable AI (XAI), there are multiple ways to explain humans how an AI system works, one of them being example-based XAI [16,21,24],

A preliminary version of this work was presented as a poster at AAIP@IJCLR2022. Supported by the Norwegian Research Council, project Machine Teaching for XAI.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-43418-1_23.

D. Koutra et al. (Eds.): ECML PKDD 2023, LNAI 14171, pp. 378–393, 2023.
https://doi.org/10.1007/978-3-031-43418-1_23

where the XAI system aims to find examples showing how the machine learning system acts in different situations. Machine teaching is the research area of actively selecting an optimal (e.g., minimal) set of examples so that a learner can identify a given concept or model [27]. The goal is for the teacher to find the smallest training set—known as the *teaching* or *witness* set—such that, a learning algorithm, when given the teaching set as an input, produces a target concept. In this work, we propose a framework based on machine teaching techniques where the XAI system (the teacher) provides explanatory examples to humans (the learners). The target concept is (a part of) the black-box AI system that needs explanation. The machine teaching algorithm must find a small set of labelled examples that will allow the human to build their own model of the AI system and thereby arrive at an explanation of the target concept [17,19]. We demonstrate the validity of our proposal by including some results of an experimental evaluation where we evaluate the results of teaching a black-box model to humans. Specifically, the black box to explain is an artificial neural network learned from images generated by Boolean expressions. We choose Boolean functions because the notions of prototype, centroid, anchors or boundary examples are more elusive in discrete concept classes like this, but we also use neural networks that might use some other features. We analyse the effect of giving humans information about how the examples were chosen and indications about the relevant features. The results show that our framework can generate explanatory examples useful to teach humans Boolean functions, better than sampling examples at random.

The paper is structured as follows. In Sect. 2 we review part of the literature related to XAI and machine teaching. Section 3 describes the framework we developed to generate explanatory witness sets. We instantiate that method for explaining neural network classifiers of images representing Boolean concepts in Sect. 4. Section 5 describes the experiments and human studies, and discusses the results. Finally, Sect. 6 closes the paper with conclusions and future work.

2 Machine Teaching for XAI

Explainable AI (XAI) is an active research field aiming at explaining the decisions of AI systems [16]. Machine learning is a key component of many AI systems, and therefore XAI usually focuses on explaining machine learning models [7,22].

Explainable AI must usually face several trade-offs, such as the tension between fidelity (level of coincidence between the predicted or understood behaviour of the system and the actual behaviour of the model) and comprehensibility (how much effort it takes for the human to understand) [5]. In general, making useful explanations among these tensions requires a great deal of abstraction, additionally modelling machine behaviour [20] in a way that is comprehensible to humans.

XAI approaches are divided into two families. In the first one, the goal is to extract an abstract representation of the AI system to serve as an explanation to a human. An example of this approach is extracting comprehensible rules from

models [3]. In the second family, the goal is to use examples such that humans can infer their explanation themselves, known as exemplar-based explanations. An example of this approach is using anchors or partial examples [21].

Machine teaching [26] is a research field that is sometimes considered as an inverse problem to machine learning. In machine teaching the examples are chosen wisely by a teacher to teach a concept to the learner. Figure 1 shows a situation where the teacher has the concept of reversing a list. The teacher could try to explain the concept, but the languages employed by the learner and teacher might not be the same. In this situation, as happens with humans frequently, a few examples may be more effective. In the image, the teacher sends a couple of input-output pairs to the learner, thinking that this would be useful for the learner to build and identify the concept.

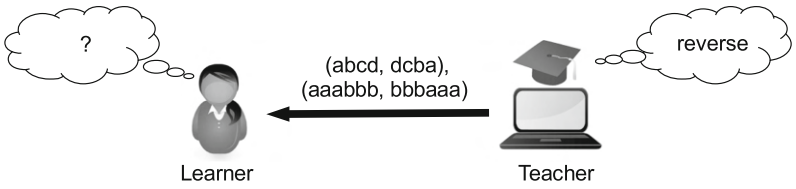


Fig. 1. Machine teaching example. The teacher tries to teach the concept of the *reverse* of a string. The teacher selects two examples carefully and shows them to the learner: the input string *abcd* being mapped into *dcba*, and the input string *aaabbb* being mapped into *bbbaaa*. The learner must infer the concept from only these two examples.

Mainly, machine teaching has been used to comprehend and depict how humans teach. An example is the analysis conducted by [12], which examines the teaching of 1D concepts (intervals) to machines, comparing a machine teaching environment with a curriculum learning environment. In both instances, the question is whether humans provide examples at the boundaries to assist the learner in replicating these boundaries or if they provide examples in clear areas so that the user can interpolate, as outlined by [1].

Our focus lies in machine teaching for the purpose of explaining concepts to humans [8]. In certain models, the teacher can interact with the learner by posing questions (e.g., [15]). On the other hand, some methods have attempted to expand the machine teaching framework by using examples to achieve explainable AI. A few proposals stray from the traditional machine teaching approach and instead utilize well-selected demonstrations in inverse reinforcement learning [9], or in the Cooperative Inverse Reinforcement Learning (CIRL) framework [6].

Yang et al. [25] evaluated the effectiveness of example-based explanations for AI using Bayesian Teaching, with a focus on high sensitivity and high specificity, and we will compare our findings to theirs. Another approach to teaching for XAI is the decomposition of the learner's hypothesis into an attention function and a decision function, as proposed by Chen et al. [2]. Ouyang [18] presents an algorithm for the Bayesian inference of regular expressions using examples. The teaching paradigm proposed is also linked to how humans communicate and how the speaker chooses the appropriate word based on their listener.

3 A MT Framework to Generate Explanatory Teaching Sets

In machine teaching, the teacher T is viewed as a function from concepts to sets of labelled examples, with $T(\theta) = S$ denoting the labelled examples S the teacher employs to teach concept θ. Likewise, the learner L is viewed as a function from sets of labelled examples to concepts, and we require that the concept guessed by the learner is compatible with the given examples S, denoted $L(S) \models S$. Correct teaching is achieved if $L(T(\theta)) = \theta$, i.e. the guessed concept is indeed the one the teacher had in mind. To achieve an efficient teaching protocol we employ simplicity β on concepts and δ on example sets (Occam's razor), as in [23]. β is shared by learner and teacher, and δ is used to prioritise simple witness set. When applying this to XAI the concept θ_{AI} can be the entire AI model to be explained or some particular substructure. To build our XAI system we employ i) an machine learning algorithm L_M modelling the human learner L_H with its simplicity prior β on guessed concepts, ii) a simplicity prior δ on example sets, and iii) a loss function λ giving a penalty for deviations of the guess θ_M from the intended θ_{AI}.

We propose a parameterised framework to generate explanatory examples from a black-box model θ_{AI}. In the framework, we explore the trade-off between fidelity (squared error of the guessed model compared to the black-box model) and teaching complexity (measured as the complexity of the set of labelled examples used as a teaching set) [14,24]. The framework is defined as:

$$T(\theta_{AI}) = \underset{S:\theta_{AI} \models S}{\operatorname{argmin}} \{\delta(S) + \mu \cdot \lambda(\theta_{AI}, \theta_M) : L_M(S) = \theta_M\} \tag{1}$$

$$L_M(S) = \underset{\theta_M:\theta_M \models S}{\operatorname{argmin}} \{\beta(\theta_M)\}$$

In these equations T is a teacher, aiming to teach a concept θ_{AI} to a human learner L_H, by finding a teaching set S such that $L_H(S) = \theta_{AI}$. To achieve automation and increase iteration speed a model L_M of L_H is used, and the teacher will therefore aim for $T(\theta_{AI}) = S$ **s.t.** $L_M(S) = \theta_{AI}$. The fidelity function becomes $1 - \lambda$ and it measures how closely the guessed concept θ_M matches the concept θ_{AI}, while the factor μ allows us to balance the influence of complexity (δ) and fidelity ($1 - \lambda$). In this work, we present an implementation[1] of Eq. 1 tested on a machine learning model trained on images generated by basic Boolean functions.

4 Obtaining Explanatory Examples from a Neural Network

In this section we discuss how the framework presented in the previous section is applied to a black-box model represented by a neural network learned from images generated by basic Boolean functions.

[1] https://github.com/BrigtHaavardstun/ExplainableAI.

4.1 The Black-Box Model θ_{AI}

For the experimental setting, we implemented our own θ_{AI}, with the task of learning a Boolean function on four variables, $\phi(A, B, C, D)$. Determining the subjective difficulty of learning Boolean functions has been addressed in the literature, see e.g. [4]. The input to θ_{AI} will be a bitmap containing a subset of letters from the alphabet $\Sigma = \{A, B, C, D\}$, with the letters present being the variables set to True. The bitmaps thus represent an example, with letters being rotated and scaled and placed randomly. This gives us the possibility of extensive training data for our AI. The output space of θ_{AI} is $\{0, 1\}$. For instance, with the concept $\phi = (A \wedge B) \vee (C \wedge D)$, we label an example 1 if ϕ evaluates to True, and 0 if ϕ evaluates to False.

We chose a Convolutional Neural Network (CNN) [13], as a common technique for images, while at the same time not interpretable by themselves, making them a good choice for generating our θ_{AI}. We implemented a CNN with 8 layers in Python using Keras and TensorFlow.

4.2 The Model of the Human L_M

For simplicity, our model L_M of the human learner will not be given bitmaps as examples. Instead, it takes as input the letters present in each image. We thus hypothesise that the human will pay attention to the letters present in the image and disregard other information such as rotation, size and position.

The hypothesis class of L_M will consist of all Boolean functions over the 4-letter alphabet. Then, given a teaching set like $S = \{(AC, 0), (AD, 0), (BD, 0), (AB, 1), (BC, 1), (CD, 1)\}$, we must decide how L_M will act. We assume a human constructs something like a partial truth table, in this case with 3 rows out of $2^4 = 16$ rows total filled with True, 3 rows filled with False, and 10 rows filled with Don't-Cares (x). Applying Occam's razor, we need to define the function β, to choose the Boolean function that is most simple and adheres to these constraints. A commonly accepted answer is the use of Karnaugh maps [11].

We use disjunctive normal form (DNF) which mimics human reasoning. To verify a positive instance you need only to confirm one clause, whereas to confirm a negative instance you always need to check all clauses. The resource-heavy task of confirming a negative compared to a positive is somewhat similar to how humans are poor at negations [10]. For each teaching set the Karnaugh map technique can find many possible DNFs, and in the spirit of K-map minimization we use the following scheme to pick the simplest. The DNFs are sorted in order by fewest clauses, and to break ties we compare clauses starting from the simplest one, using the criteria 1) fewest variables, 2) fewest negations, 3) lexicographic order. This defines β and gives us a unique Boolean formula in DNF form for each teaching set.

4.3 The Fidelity Function $1 - \lambda$

When we want to compare θ_{AI} and θ_M, we need to view the former as an approximation to some Boolean function, but also being affected by the loca-

tion, rotation, etc., of the letter. Consequently, for each subset of letters (logical example), we estimate the percentage of images containing exactly these letters that θ_{AI} evaluates to True on new images, based on the full training set. We get values like the top row in Table 1. We observe that θ_{AI} predicts some letter groups the same and is more undecided on other letter combinations.

Table 1. Top row shows the percentage of bitmaps on letters for that column for which θ_{AI} evaluates to True. Bottom row shows the truth table of $\theta_M = (A \wedge B) \vee (C \wedge \neg A)$, and $\lambda(\theta_{AI}, \theta_M) = \frac{0.2222}{16} \approx 0.0139$ is the MSE of the difference of all 16 columns, giving fidelity 0.9861.

Symbol	$\emptyset$	A	B	C	D	AB	AC	AD	BC	BD	CD	ABC	ABD	ACD	BCD	ABCD
θ_{AI} predicts	0.00	0.00	0.00	0.95	0.00	0.99	0.02	0.00	0.63	0.02	0.91	1.00	1.00	0.04	0.74	1.00
θ_M evaluates	0	0	0	1	0	1	0	0	1	0	1	1	1	0	1	1

To evaluate how well the θ_M, returned by the learner as the Boolean formula minimizing β, matches θ_{AI}, we use its truth table as in the bottom row of Table 1. We then compare the two rows (for θ_{AI} and θ_M) using Mean Square Error (MSE) to get λ and fidelity $1 - \lambda$.

We have experimented with various definitions for the complexity function, to punish large and complicated teaching sets S. The chosen δ is a simple squared sum of the number of variables present in each example, plus 0.1 for the empty set (corresponding to setting no variable to True). We thus keep low the total number of variables in all examples while simultaneously putting a high cost on a single large example. Note that the δ values are typically much higher than the λ values, so in our first set of experiments we set the multiplicative factor $\mu = 800$ when computing the aggregated score $\delta(S) + \mu \cdot \lambda(\theta_{AI}, \theta_M)$.

4.4 The Teacher *T*

The goal of the teacher is to find a teaching set explaining θ_{AI}, by iterating over potential teaching sets. For each teaching set S, we compute $L_M(S) = \theta_M$ as described earlier, and the aggregate score $\delta(S) + \mu \cdot \lambda(\theta_{AI}, \theta_M)$. During the iteration we retain the best aggregate score. For these experiments the iteration is an exhaustive search.

5 Experimental Evaluation

Given the previous setting we performed a set of experiments using different concepts and parameters to analyse the effect of several elements in the machine teaching process on explaining the behaviour of various AI models. In particular, we played with AI models trained on different sized training sets, which approximate the original Boolean function to different levels of accuracy. Depending on how well the AI is approximated by a Boolean function the trade-off parameter μ between fidelity $(1 - \lambda)$ and teaching complexity (δ) has different effects.

5.1 Generation of Teaching Sets

5.1.1 Fixed μ for Varying Models. We trained nine different Θ_{AI} models with differently sized data sets. In this first experiment, all models are trained with the ground truth $\phi = (A \wedge B) \vee C$ and the alphabet $\Sigma = \{A, B, C\}$. The data set sizes used in the experiment are: {10, 50, 100, 500, 1000, 2000, 5000, 10000, 50000}. Accordingly, we denote the different models: $\{AI_{10}, AI_{50}, AI_{100}, AI_{500}, AI_{1000}, AI_{2000}, AI_{5000}, AI_{10000}, AI_{50000}\}$.

In Table 2 we show several results. In the first row we see the expected result that the accuracy of the models wrt the original concept ϕ increases as more training examples were given to the neural network. In the next rows we show the Boolean expression that best approximates the model, with its associated highest possible fidelity $(1 - \lambda)$ over all Boolean functions. We see that the language of Boolean functions obtains a perfect match for the case of AI_{10} (because the underlying concept is very simple, always predicting True, which is a Boolean function) and almost perfect for AI_{10000} and AI_{50000} (because the number of training examples leads to a concept that is very close to ϕ). Note that also in other cases the most accurate Boolean function is ϕ (from AI_{2000} and up).

Table 2. Several Θ_{AI} models AI_t trained for size t of training examples for $\phi = AB + C = (A \wedge B) \vee C$. We first show accuracy with respect to ϕ. The next two rows show the closest Boolean expression ($AB+C$ from AI_{2000} and on) and its fidelity value $1 - \lambda$. Then we do teaching with $\mu = 800$, and show the Boolean concept taught by the system, the teaching set and its complexity, the fidelity and aggregate score.

AIs	AI_{10}	AI_{50}	AI_{100}	AI_{500}	AI_{1000}	AI_{2000}	AI_{5000}	AI_{10000}	AI_{50000}
Accuracy AB+C	62.50	72.85	78.38	81.01	88.47	91.42	94.74	98.72	99.36
Boolean with highest $1 - \lambda$	Always True	A+B+C	A+B+C	AB+ AC+BC	AB+ AC+BC	AB+C	AB+C	AB+C	AB+C
Highest $1 - \lambda$	1	0.927	0.9578	0.9226	0.9843	0.9752	0.9936	0.9994	0.9999
Model taught θ_M	Always True	A+B+C	A+B+C	A+C	AB+ AC+BC	AB+C	AB+C	AB+C	AB+C
Teaching Set S	{(∅,1)}	{(∅,0), (A,1), (B,1), (C,1)}	{(∅,0), (A,1), (B,1), (C,1)}	{(∅,0), (A,1), (C,1)}	{(A,0), (AB,1), (AC,1), (B,0), (BC,1), (C,0)}	{(A,0), (AB,1), (B,0), (C,1)}	{(A,0), (AB,1), (B,0), (C,1)}	{(A,0), (AB,1), (B,0), (C,1)}	{(A,0), (AB,1), (B,0), (C,1)}
$\delta(S)$	0.1	3.1	3.1	2.1	15	7	7	7	7
$1 - \lambda(AI_x, \theta_M)$	1	0.927	0.9578	0.9179	0.9843	0.9752	0.9936	0.9994	0.9999
$\delta + 800\lambda$	0.1	61.52	36.71	67.82	27.63	26.84	12.17	7.49	7.09

Now let us look at the next few rows showing results for the teaching framework when run with the chosen parameter $\mu = 800$. First we show the Boolean concept θ_M that is actually taught by the system and note that it is almost always equal to the Boolean concept with highest fidelity $(1 - \lambda)$ value in the 2nd row. The only exception is AI_{500} where the trade-off between δ and λ favours

the Boolean concept $A \vee C$ instead of $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$ because the teaching set for the former is much simpler ($\delta = 2.1$) than the teaching set for the latter ($\delta = 15$ as can be seen under AI_{1000}). The next rows show the teaching set employed, its δ value, the fidelity value and the aggregate score.

There are three clear cases in the table (AI_{10}, AI_{10000} and AI_{50000}) where a simple teaching set allows the teacher to convey a concept to the learner that very closely captures the model. But there are other cases, such as AI_{1000} and AI_{2000}, where the situation is less clear. For AI_{1000} the fidelity is not bad ($1 - \lambda = 1 - 0.0157 = 0.9843$) but the complexity of teaching becomes high ($\delta = 15$) so even if a sufficiently accurate concept can be taught this is at the cost of a higher effort from the learner. For AI_{2000} we see that this cost is reduced but the fidelity is worse ($1 - \lambda = 1 - 0.0248 = 0.9752$).

5.1.2 Varying μ for a Single Model. In a second experiment we trained a Θ_{AI} model on a data set of size 350 for $\phi = (A \wedge B) \vee (C \wedge D) = AB + CD$ on 4 variables/letters. The accuracy was 78.25% and the closest Boolean function, with a fidelity value $1 - \lambda$ of $1 - 0.06 = 0.94$, turned out to be $ABC + ABD + ACD + BCD$, which can be interpreted as "True if and only if at least 3 letters present". To investigate the trade-off between fidelity and complexity, teaching was done with varying values of μ, see left column in Table 3. We see that as μ increases more emphasis is put on fidelity at expense of complexity. Note that at $\mu = 3200$ the fidelity is as good as possible (i.e. highest possible $1 - \lambda$) since the teaching set at $\mu = 3200$ is optimal for that optimal θ_M so increasing μ will have no effect. Of course, this comes at the expense of a high complexity. An option worth exploring is to take the characteristics of the human user into account when deciding on the fidelity vs complexity trade-off, e.g., having a high value of μ for an expert and a low value for a non-expert.

Table 3. Results for a single AI model where the closest Boolean function turned out to be $ABC+ABD+ACD+BCD$. Teaching was done with varying values of μ, see left column. As μ increases more emphasis is put on lower fidelity $1 - \lambda = 1 - \lambda(\theta_{AI}, \theta_M)$ at expense of higher teaching complexity δ.

Range μ	Model taught θ_M	$1 - \lambda$	δ	Teaching set
16	A	0.812	1.1	{(∅, 0),(A,1)}
160–960	AC+BD	0.9119	12	{(A,0),(B,0),(C,0),(D,0),(AC,1),(BD,1)}
1120–1840	AC+BCD+AD	0.9282	30	{(A,0),(AC,1),(AD,1),(BC,0),(BD,0),(CD,0)}
1920–2400	AC+ABD+BCD	0.9344	42	{(AC,1),(AB,0),(AD,0),(BC,0),(BD,0),(CD,0), (ABD,1),(BCD,1)}
3200–∞	ABC+ABD+ ACD+BCD	0.94	60	{(AB,0),(AC,0),(AD,0),(BC,0),(BD,0),(CD,0), (ABC,1),(ABD,1),(ACD,1),(BCD,1)}

This second experiment also shows that it is not difficult to determine when the language used for the explanation leads to low fidelity and/or complex explanations. Actually, in this case, since the function captured by the AI model does not have a clean Boolean concept, we can detect that teaching will either lead to

low fidelity or complex explanation (or both). In sum, the use of the complexity of the teaching set in the trade-off is not only the right choice when doing example-based XAI but it also leads to the same insights as when the complexity of the concept is taken into account.

5.2 Different Hypothesis Spaces

This section will examine the effects of different hypothesis spaces (representation languages) between the AI model, the ground truth, the model of the learner L_M and the actual human learner L_H. In our exemplar-based explanation system, we added letter rotations, letter resizing and letter location as cognitive noise and extra features, so that we have more confounders, and motivated by these spurious variations the neural network will have error with respect to the ground truth. This makes things more realistic, with the neural network creating patterns that are not fully captured by L_M. A neural network trained on images can in principle model functions over all possible images creating an enormous hypothesis space H_{pix}. On the other hand, the ground truth labelling function is on a small set of features, i.e. the presence or absence of k letters, giving the small hypothesis space H_{p^k}, with $p = \{0, 1\}$ indicating the two possibilities of present or absent. When the actual human learner L_H is given a set of labelled images from the example space H_{pix}, they will create a rule based on the features of the images they consider relevant.

Our exemplar-based explanation system has a focus on the simplicity of examples, and so far this has been with respect to the δ function. But simplicity also comes into play when generating images with certain letters present. The simplest images contain letters that all have the same size, with no rotation and with uniform placement, and these can be used as the simplified examples most compatible with the smaller example space H_{p^k}. The research question we want to address with the following experiment is whether using such simplified examples helps align the hypothesis spaces.

The experiment will compare the options for aligning the hypothesis spaces, by three groups that are given the teaching sets in different formats

- Group I: Use original images.
- Group II: Use simplified images, without irrelevant features.
- Group III: Use original images, but alert learners to essential features

We created a 2AFC (two-alternative forced choice) survey. Participants were shown a teaching set of carefully selected images and the (binary) classification of these images into Box 1 or Box 2 (True/False). They were then shown a test set of unclassified images and tasked to classify each image into one either Box 1 or Box 2. In Fig. 2, we display how these tasks were presented to the participants.

The teaching sets were selected according to the system presented in the earlier sections. The test sets were randomly selected with the restriction that exactly one image should contain the same letter combination as one in the teaching set.

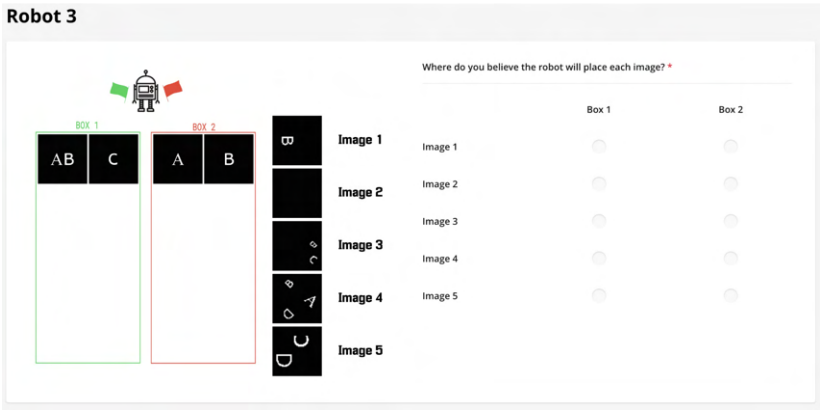


Fig. 2. Participants in group II were presented with this screen when tested on the formula ϕ_3 (C or both A and B). Note the teaching set for Box 1 (True) and Box 2 (False) have images without the noise (rotation, resizing and relocation) found in the five images of the Test Set. For Groups I and III also the teaching set had noise. For group III the following text was displayed prominently: "NB: Size, placement and rotation do not matter. Focus on present/absent letters."

In total, we trained six AI models (called 'robots' in the survey) to be tested. All of them were trained to a high degree of accuracy. Each robot was trained on a different Boolean expression ϕ.

We asked each participant to classify five test instances for each Boolean expression ϕ. The participant's answer to the ith test is denoted $p(\phi, i)$. The correct answer to each test is given by $\phi(i)$. To calculate a participant's score for a single Boolean expression, we use the following formula: $p(\phi) = \frac{1}{6}\sum_{i=1}^{6}[p(\phi, i) = \phi(i)]$ where: $p(\phi, i) = \phi(i)$ is 1 if the participant's answer is correct and 0 otherwise. We then calculate the score of the jth participant across all Boolean expressions as follows: $p_j = \frac{1}{6}\sum_{i=1}^{6} p(\phi_i)$. Here, $p(\phi_i)$ represents the participant's score for the ith Boolean expression.

In total, we had 42 voluntary participants, who were master students, doctoral students or faculty in informatics, none of whom received compensation. The participants were presented with the survey and freely choose to participate. The participants were randomly assigned to the groups, with 12 participants in group I, 17 in group II, and 13 in group III.

The average scores for Groups I, II, and III were 0.564, 0.664, and 0.732, respectively. Furthermore, we observed that group III had the highest average score on 5 out 6 test instances. Though the results are not conclusive due to the small sample size[2], we decided to move on with option III, as we know that the teacher should do something to align the hypothesis spaces of L_M and L_H to achieve efficient learning. Next, we look at the quality of the teaching sets, by comparing our teacher to a teacher randomly selecting teaching sets of similar complexity, with both presenting the teaching sets as group III.

[2] We conducted t-tests for all pairs of groups to test whether means differ statistically significantly and got p-values 0.0297, 0.0013, and 0.0747 for pairs (I, II), (I, III) and (II, III), respectively.

5.3 Compare to Teaching Sets Chosen Randomly

In this section, we discuss a second survey where we compare the teaching sets given by our exemplar-based explanation system (the smart teacher) to a system where the teaching sets are chosen randomly but correctly labelled and without repetitions (the random teacher).

To make the comparison of the random teacher and smart teacher fair, both will present their teaching sets as in group III. For a given Boolean formula, we first find the teaching set S_S used by the smart teacher, and then we ensure that the random teaching set S_R has a complexity (δ^*-value[3]) close to S_S (i.e. within some $\pm\epsilon$ additive difference) by choosing S_R as follows, while making sure that no letter combination is repeated in S_R:

1. while $\delta(S_R) < \delta(S_S) - \epsilon \rightarrow$ add a random new image to S_R
2. if $\delta(S_S) - \epsilon \leq \delta(S_R) \leq \delta(S_S) + \epsilon \rightarrow$ use S_R
3. if $\delta(S_S) + \epsilon < \delta(S_R)$ then set $S_R = \emptyset$ and restart from 1.

To avoid bias from the previous survey, we changed most Boolean expressions for the new survey. They were (again) chosen with a variation in terms of expected difficulty, see Table 4. The formulas used can be found in Table 4.

Table 4. Boolean expressions used in the second survey.

Nr	Prior L_H	Short description	Boolean expression
ϕ_1	High	Less Than Two Letters	$(\neg A \wedge \neg B \wedge \neg D) \vee (\neg B \wedge \neg C \wedge \neg D) \vee$ $(\neg A \wedge \neg C \wedge \neg D) \vee (\neg A \wedge \neg B \wedge \neg C)$
ϕ_2	Medium	A or both B and D	$(B \wedge D) \vee A$
ϕ_3	Medium	B or D	$B \vee D$
ϕ_4	High	Exactly One Letter	$(A \wedge \neg B \wedge \neg C \wedge \neg D) \vee (\neg A \wedge B \wedge \neg C \wedge \neg D) \vee$ $(\neg A \wedge \neg B \wedge C \wedge \neg D) \vee (\neg A \wedge \neg B \wedge \neg C \wedge D)$
ϕ_5	Medium	No D	$\neg D$

Both groups G_S given smart teaching sets and G_R given random teaching sets will be shown the same test sets, and we now discuss how to generate the testing sets. When generating testing sets, we want them to be fair with regards to both teaching sets S_S and S_R so that none of them get an unfair advantage. Define $l(S)$ to be the set of letter combinations in the teaching set S, and define X to be the set of all images. We generate test sets for S_S and S_R by choosing images from X as follows, while ensuring that each letter combination appears at most once:

1. As long as there are new letter combinations in $l(X)/(l(S_R) \cup l(S_S))$, choose such an image at random.
2. Otherwise, fill the test set with images from the set of letter combinations in $l(S_R) \cap l(S_S)$, chosen randomly

[3] We use $\delta^* =$ *Number of present letters.*

We select a test set of size five by the above protocol, for each Boolean expression, see Table 6. We will thus be able to make a fair comparison between the random and smart teaching sets.

Table 5. Teaching sets used for G_S on left and G_R on right (B1=Box 1, B2=Box 2).

Nr	Teaching set group G_S	$\delta^*(G_S)$	Teaching set group G_R
ϕ_1	B1:{ A , B , C , D } B2:{ AB , AC , AD , BC , BD , CD }	16	B1:{ ∅ , A , B , C } B2:{ AC , ACD , AD , BCD , CD }
ϕ_2	B1:{ A , BD } B2:{ B , D }	5	B1:{ A , AC } B2:{ B }
ϕ_3	B1:{ B , D } B2:{ ∅ }	2.1	B1:{ D } B2:{ C }
ϕ_4	B1:{ A , B , C , D } B2:{ ∅ , AB , AC , AD , BC , BD , CD }	16.1	B1:{ A , B , C } B2:{ AB , ABC , ABD , AD , BCD }
ϕ_5	B1:{ ∅ } B2:{ D }	1.1	B1:{ ∅ } B2:{ BD }

Table 6. Test sets used for both G_S and G_R.

Nr	Test set
ϕ_1	{ ABC , ABCD , ABD , C , AD }
ϕ_2	{ ABD , ACD , C , ∅ , AD }
ϕ_3	{ ABC , BD , ABCD , AB , ACD }
ϕ_4	{ ACD , ABCD , B , AD , C }
ϕ_5	{ AC , CD , A , ABC , BC }

5.4 Overall Results

We will now discuss the results of the second survey. In total, we had 56 participants, none of them overlapping with the previous test. The participants were students in a university-level informatics course. The participants were randomly assigned into two groups, with 22 participants in group G_S and 34 in group G_R.

We start by looking at each group's average score for each ϕ_i. The results are shown in Fig. 3. Our initial observation is that the group G_S has average accuracy over all 5 Boolean expressions of 0.809 versus 0.699 for the group G_R, suggesting that the smart teaching sets have an advantage. This difference is statistically significant ($p = 0.00016$ from t-test). There are two cases where G_R exhibits slightly higher average accuracy than G_S, namely for ϕ_1 and ϕ_4. Notice these concepts are the ones we classified to have high prior for L_H in Table 4 and if we look at Table 5 we see these are also the concepts where our automatic system generates teaching sets with large size (δ-value). When the system is

Task	Group G_S	Group G_R	$G_S + G_R$
ϕ_1	0.882	0.924	0.908
ϕ_2	0.954	0.759	0.836
ϕ_3	0.827	0.547	0.657
ϕ_4	0.873	0.888	0.882
ϕ_5	0.509	0.376	0.428
Total	0.809	0.699	0.742

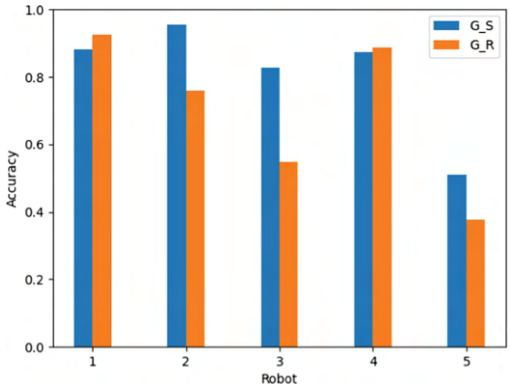


Fig. 3. The table shows average accuracy in Groups G_S, G_R and $G_S + G_R$, for each ϕ_i and Total. The bar plot shows average accuracy in G_S, G_R for each ϕ_i.

aligned[4], as with ϕ_2 and ϕ_3, our system achieves substantially higher accuracy than the random teacher (Fig. 4).

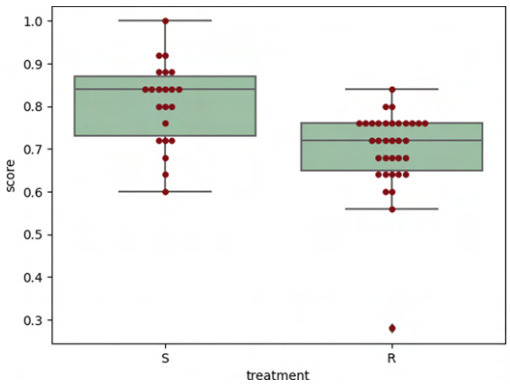


Fig. 4. Boxplot showing results of the two survey groups, to the left teaching sets with the exemplar-based explanation system, and to the right the random teaching sets. There is a clear difference between the groups.

Table 7 gives information on the most common answers for each robot. The most common answer vector of group G_S is the correct one for 4 of the 5 ϕ_is, while for G_R it is the correct one for 3 of the 5.

5.4.1 Detailed Discussion of ϕ_4 (Exactly One Letter in Box 1). Note in Table 7 that for ϕ_4 a full 85 % of the participants in G_R had all answers correct, whereas this drops to 64 % for group G_S. We believe this is because the smart teaching set happens to be compatible with the (wrong) concept 'Odd Number of Letters'. Thus when shown the test set containing 'ACD' almost a third (7/22) of those thought with smart teaching set made a wrong choice,

[4] We say that the system is aligned when the prior of L_M is similar to the prior of L_H.

Table 7. The most common answer for both groups. We show how correct the answer is, with 1.0 being all 5 tests correct, and we also show how common it is.

Concept	Most common answer G_S	Score [0..1]	Fraction of participants G_S	Most common answer G_R	Score [0..1]	Fraction of participants G_R
ϕ_1	[2,2,2,1,2]	1.0	59%	[2,2,2,1,2]	1.0	76%
ϕ_2	[1,1,2,2,1]	1.0	82%	[1,1,2,2,1]	1.0	44%
ϕ_3	[1,1,1,1,1]	1.0	68%	[2,1,1,2,1]	0.6	29%
ϕ_4	[2,2,1,2,1]	1.0	64%	[2,2,1,2,1]	1.0	85%
ϕ_5	[2,2,2,2,2]	0.2	41%	[2,2,2,2,2]	0.2	53%

while less than a tenth (3/34) of those taught with the random teaching set selected the wrong box. The teaching sets are in Table 5. This is why we believe the random teaching set is slightly better (accuracy 0.888 vs 0.873, see Fig. 3) for formula ϕ_4 where the procedure built on Karnaugh map used in our system generates a very large smart teaching set.

We also asked participants for how they themselves would explain what they thought each robot was doing. This information is useful to elucidate why the smart teaching set does worse than the random teaching set on ϕ_4.

In the group G_S (the smart teaching set), 10 of the 22 subjects did not write any explanation while 12 subjects had an explanation. 7 people answered wrong for test 'ACD' and 3 of these had no explanation, whereas the other 4 confirm our suspicion that they are focusing on odd/even numbers of letters.

In the group G_R (the random teaching set) 27 subjects had an explanation. 3 people answered wrong for test 'ACD' and 2 of these had no explanation, whereas the 3rd had an explanation that actually should have led the subject to classify 'ACD' correctly.

6 Conclusions

The results of the paper are indeed promising and have the potential to advance the field of explainable AI. Our proposed framework based on machine teaching can effectively teach complex functions to humans using explanatory examples, with a clear advantage over choosing the examples randomly. These findings demonstrate that machine teaching is a valid approach for exemplar-based explainable AI, but also that the expectations on the features and the priors of the humans is critical to get effective explanations from as few examples as possible. As future work, we propose the study of L_M models better aligned with humans. Also, we are considering the use of teaching examples generated by recent language models.

References

1. Basu, S., Christensen, J.: Teaching classification boundaries to humans. In: Twenty-Seventh AAAI Conference on Artificial Intelligence (2013)
2. Chen, Y., Mac Aodha, O., Su, S., Perona, P., Yue, Y.: Near-optimal machine teaching via explanatory teaching sets. In: International Conference on Artificial Intelligence and Statistics, pp. 1970–1978 (2018)
3. Domingos, P.: Knowledge discovery via multiple models. Intell. Data Anal. **2**(1–4), 187–202 (1998)
4. Feldman, J.: Minimization of Boolean complexity in human concept learning. Nature **407**(4), 630–633 (2000)
5. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM Comput. Surv. **51**(5), 93 (2018)
6. Hadfield-Menell, D., Russell, S.J., Abbeel, P., Dragan, A.: Cooperative inverse reinforcement learning. In: NIPS, pp. 3909–3917 (2016)
7. Hernández-Orallo, J.: Gazing into clever hans machines. Nat. Mach. Intell. **1**(4), 172 (2019)
8. Hernández-Orallo, J., Ferri, C.: Teaching and explanations: aligning priors between machines and humans. In: Human-Like Machine Intelligence, pp. 171–198 (2021)
9. Ho, M.K., Littman, M., MacGlashan, J., Cushman, F., Austerweil, J.L.: Showing versus doing: teaching by demonstration. In: NIPS, pp. 3027–3035. Curran (2016). www.papers.nips.cc/paper/6413-showing-versus-doing-teaching-by-demonstration.pdf
10. Hoosain, R.: The processing of negation. J. Verbal Learn. Verbal Behav. **12**(6), 618–626 (1973). https://doi.org/10.1016/S0022-5371(73)80041-6, www.sciencedirect.com/science/article/pii/S0022537173800416
11. Karnaugh, M.: The map method for synthesis of combinational logic circuits. Trans. Am. Inst. Electr. Engineers Part I: Commun. Electron. **72**(5), 593–599 (1953). https://doi.org/10.1109/TCE.1953.6371932
12. Khan, F., Mutlu, B., Zhu, J.: How do humans teach: on curriculum learning and teaching dimension. In: NIPS, pp. 1449–1457 (2011)
13. Lecun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998). https://doi.org/10.1109/5.726791
14. Lipton, P.: Contrastive explanation. Roy. Inst. Phil. Suppl. **27**, 247–266 (1990)
15. Liu, W., Dai, B., Li, X., Liu, Z., Rehg, J.M., Song, L.: Towards black-box iterative machine teaching. arXiv preprint arXiv:1710.07742 (2017)
16. Molnar, C.: Interpretable machine learning. https://lulu.com/ (2020)
17. Ortega, A., Fierrez, J., Morales, A., Wang, Z., Ribeiro, T.: Symbolic AI for XAI: evaluating LFIT inductive programming for fair and explainable automatic recruitment. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 78–87 (2021)
18. Ouyang, L.: Bayesian inference of regular expressions from human-generated example strings. arXiv:1805.08427 (2018)
19. Pisano, G., Ciatto, G., Calegari, R., Omicini, A.: Neuro-symbolic computation for xai: towards a unified model. In: WOA, vol. 1613, p. 101 (2020)
20. Rahwan, I., et al.: Machine behaviour. Nature **568**(7753), 477 (2019)
21. Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-precision model-agnostic explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)

22. Samek, W., Müller, K.-R.: Towards explainable artificial intelligence. In: Samek, W., Montavon, G., Vedaldi, A., Hansen, L.K., Müller, K.-R. (eds.) Explainable AI: Interpreting, Explaining and Visualizing Deep Learning. LNCS (LNAI), vol. 11700, pp. 5–22. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-28954-6_1
23. Telle, J.A., Hernández-Orallo, J., Ferri, C.: The teaching size: computable teachers and learners for universal languages. Mach. Learn. **108**, 1653–1675 (2019). https://doi.org/10.1007/s10994-019-05821-2
24. van der Waa, J., Nieuwburg, E., Cremers, A., Neerincx, M.: Evaluating XAI: a comparison of rule-based and example-based explanations. Artif. Intell. **291**, 103404 (2021)
25. Yang, S.C.H., Vong, W.K., Sojitra, R.B., Folke, T., Shafto, P.: Mitigating belief projection in explainable artificial intelligence via Bayesian teaching. Sci. Rep. **11**(1), 9863 (2021). https://doi.org/10.1038/s41598-021-89267-4. www.nature.com/articles/s41598-021-89267-4
26. Zhu, X.: Machine teaching: an inverse problem to machine learning and an approach toward optimal education. In: AAAI, pp. 4083–4087 (2015)
27. Zhu, X., Singla, A., Zilles, S., Rafferty, A.N.: An overview of machine teaching (2018). arxiv.org/abs/1801.05927

Paper II

Ferri, C., Garigliotti, D., Hernández-Orallo, J., Håvardstun, B., & Telle, J. A.
Machine Learning, **115**(1), art. 20 (2026)
DOI: 10.1007/s10994-025-06954-3

Machine Learning (2026) 115:20
https://doi.org/10.1007/s10994-025-06954-3

When Redundancy Matters: Machine Teaching of Representations

Cesar Ferri[1] · Dario Garigliotti[2] · Jose Hernandez-Orallo[1] · Brigt Håvardstun[2] · Jan Arne Telle[2]

Received: 6 February 2025 / Revised: 28 August 2025 / Accepted: 10 December 2025 /
Published online: 8 January 2026

Abstract
In traditional machine teaching, a teacher needs to teach a concept to a learner by means of a finite set of examples, the witness set. But concepts can have many equivalent representations. This redundancy strongly affects the search space, to the extent that teacher and learner may not be able to easily determine the equivalence class of each representation. In this common situation, instead of teaching concepts, we explore the idea of teaching representations. We work with several teaching schemas that exploit representation and witness *size* (Eager, Greedy and Optimal) and analyze the gains in teaching effectiveness, both theoretically, and also experimentally for languages where redundancy can vary (DNF expressions and Turing-complete P3 programs). Our theoretical and experimental results indicate that there are various types of redundancy, related, e.g,. to the *spread* of the redundant representations, handled better by the new Greedy schema introduced here than by the Eager schema. For P3 programs witness sets found by Greedy are usually smaller than the programs they identify, corroborating previous results that conveying information efficiently is a leitmotif of machine teaching.

Keywords Machine teaching · Concept representations · Redundancy

1 Introduction

In formal models of *machine learning* (ML), there is a concept class C of possible hypotheses, an unknown target concept $c^* \in C$ and training data given by correctly labelled random examples. In formal models of *machine teaching* (MT), a collection of labelled examples,

Editor: Henry Gouk.

✉ Dario Garigliotti
dario.garigliotti@uib.no

[1] VRAIN, Universitat Politècnica de València, Valencia, Spain

[2] Department of Informatics, University of Bergen, Bergen, Norway

also called a witness set w, is instead carefully chosen by a teacher T so the learner L can identify c^* in C, thus $w = T(c^*)$ and $c^* = L(w)$, see e.g. Goldman and Kearns (1995). Recently, machine teaching has been applied in various fields like pedagogy (Shafto et al., 2014), trustworthy AI (Zhu et al., 2018), reinforcement learning (Zhang et al., 2021), active learning (Wang et al., 2021) and explainable AI (Yang et al., 2021; Håvardstun et al., 2023). Most theoretical models of MT (and of ML) assume that the class C is a set of different concepts; the goal is then to identify or approximate the target concept in the class. In MT, this assumption is broken when concepts are represented in a language that can have several expressions for the same concept, such as "$|x_3| \cdot |x_5| > 0$" and "$x_3 \neq 0 \wedge x_5 \neq 0$". *A teaching model can focus on teaching the first expression while not realising that the second—quivalent—expression may be preferable.*

Concepts are expressed in a representational language, and most languages have redundancy, with several representations mapping to the same concept; this is a common kind of *language bias*. For some languages, teacher and learner may not know whether two representations are equivalent—it may not even be computable. Hence, we may end up teaching more than one representation of a concept. For example, the 'powers of 2', the 'number of subsets of an n-element set', and the 'number of n-bit binary strings', all denote the same subset of natural numbers. In many practical situations, representations rather than concepts matters, e.g., prompts for answers (Fernando et al., 2023) or programs for behaviours (Finnie-Ansley et al., 2022).

Having redundant representations is common in most languages with which humans and machines represent concepts, from natural languages to artificial neural networks. Redundancy has been vindicated as a way to accomplish resilience—see e.g. Vardi (2022) referring to von Neumann: "Resilience via redundancy is one of the great principles of computer science, which deserves more attention!" - by analysing kinds and levels of redundancy. Hence, studying redundancy, in particular how the relation between representations and concepts affects teaching, has important implications for and beyond teaching. This holds also for machine learning [see, e.g., Hadley and Cardei (1999)], where notions of syntactic, semantic and search bias (any language bias) are explored to make one language more suitable for learning than another [see, e.g., Muggleton and De Raedt (1994); Whigham (1996), and Nédellec et al. (1996)]. The issue is important in any area where there is a non-injective correspondence from concepts to representations, such as explainable AI. Several notions of redundancy, density or compactness in representations have been studied (Yu, 1988; Enflo et al., 1994; Alpuente et al., 2010), yet without focus on quantifying redundancy or concentration over equivalence classes, as we define them in this paper.

We explore the idea of teaching *representations*, where the teacher knows a representation and wants the learner to identify it. Different representations r_1 and r_2 can be in the same equivalence class, i.e. defining the same concept c, denoted by $[r_1] = [r_2] = c$. We model this much as in the traditional *concept* teaching scenario, but now by a set of representations R (e.g. all regular expressions on alphabet $\{a, b\}$) over a domain X (e.g., all finite strings over $\{a, b\}$). Each $r \in R$ splits X into negative and positive examples and can be viewed as a function $r : X \to \{0, 1\}$ (e.g. $r(abb) = 1$ if *abb* is generated by r and $r(abb) = 0$ otherwise). If $[r_1] = [r_2]$ then as functions over the domain r_1 and r_2 are equal, i.e. they split X equally (e.g., $r_1 = (aa|ab|ba|bb)^*$ and $r_2 = ((a|b)(a|b))^*$ both generate the set of strings of even length). The teacher T maps $r \in R$ injectively to a finite set of labelled examples $T(r) = w_r = \{(x_1, b_1), ..., (x_n, b_n)\}$ consistent with r, i.e. with each $x_i \in X$ and

$r(x_i) = b_i$, called the witness for r. In case of teaching only with positive examples we will have $b_i = 1$ for all examples in any witness. As usual in machine teaching, the goal is to teach using witnesses on the smallest possible number of labelled examples.

We assume the learner has simplicity functions on the set of representations and on the set of labelled examples. A similar assumption appears in various Machine Teaching models, e.g., the Preference-Based teaching model (Gao et al., 2017) assumes an ordering on concepts and the Recursive teaching model (Zilles et al., 2011) assumes a partial ordering on witnesses. Furthermore, in the teacher/learner algorithm of Telle et al. (2019), which we call *Eager*, the learner employs an ordering on concepts called the learning prior, while the teacher wants to minimize the 'teaching size,' which implicitly determines an order on witnesses. In the Eager algorithm, the learner, when given a witness, will employ Occam's razor and guess the simplest consistent representation, and the teacher will always provide the smallest witness that suffices to learn the target representation. Note this implies the learner will learn at most one representation per concept, namely the simplest one. But *the main issue is that if the teacher does not know that* r_1 *and* r_2 *are equivalent, and* r_1 *is simpler than* r_2 *, then the teacher will be looking for a witness set for* r_2 *that distinguishes it from* r_1 *, something the teacher will never find.* In this work, we operationalize the simplicity function over representation set R by a natural size function $s(r)$ defined in terms of the syntax in which a representation $r \in R$ is expressed (see Sect. 2).

When teaching a target it is quite common to give the smallest observation that suffices to learn it, and the learner could actually expect this from the teacher, and thus rule out representations that could have been taught with a witness smaller than what was given. Herein lies the simple idea behind the slight change to Eager into what we call the *Greedy* algorithm. Now, after having taught a representation r using its witness w_r then *rand* w_r *are removed from consideration* so that another witness w consistent with r can later be used to teach another representation, some r' which is the simplest one remaining that is consistent with w. Note that we then may or may not have $[r] = [r']$. For this to work we must assume that the learner, when given a witness, will know the teacher behaves in the above way. See Sect. 2 for the exact protocol formalizations.

We also compare with a mapping that we call *Optimal*, being equal to the graph-theoretical optimum (minimizing the maximum number of labelled examples used by a witness) under the minimal constraint that a representation must be taught by a consistent witness and that two distinct representations must be taught by two distinct witnesses. Section 3 shows several theoretical results about the strengths of Eager, Greedy and Optimal. In particular, on concepts taught by both Eager and Greedy, the latter never uses a larger witness, and often a smaller witness. However, Theorem 3 shows that for any concept class, adding redundant representations with small spread in the order, can level out the difference between Eager and Greedy.

Moreover, there are situations where the max size witness used by Greedy is much larger than the Optimal. In arguing for a lower bound on the performance of Greedy we arrived, in an earlier arxiv version of the current paper (Ferri et al., 2024), at a conjecture about when the performance of Greedy was the worst possible, and this conjecture was confirmed by Sunde et al. (2024). In a further study, the parameter GMN(C) (Greedy Matching Number of a 0–1 matrix C, where we can view C as the biadjacency matrix of the bipartite consistency graph) was defined and studied by Simon and Telle (2025). This is the largest possible value $\max_{c \in C} |T(c)|$ of a teacher mapping T returned by the Greedy algorithm, over all possible

witness orderings. The interested reader is referred to Simon and Telle (2025) for an in-depth study of several related combinatorial parameters.

In Sect. 4 we report on experiments with Eager, Greedy and Optimal, on several representation languages with various kinds of redundancy.

The main finding is that *while Greedy uses more witness sets to teach all representations, it uses them more effectively to teach more concepts than Eager, oftentimes also teaching commonly taught concepts by a smaller witness.* The experiments were further chosen to answer this question: What characterizes a language where Greedy teaches many common concepts by a smaller witness than Eager? We show that this is not directly related to the *amount* of redundancy, but rather to the *spread* of the redundant representations in the order (see Fig. 3). Also, for P3 programs, the witness sets used by Greedy are often smaller than the programs they identify (see Fig. 4). A similar behavior on fewer programs was noted for Eager by Telle et al. (2019).

Eager and Greedy are two extremes (one teaching a single representation per concept, the other attempting to teach all representations) in a spectrum of approaches that can shed light to important phenomena such as the effect of redundancy in inductive search, the actual bias in size-related priors and the relevance of syntax over semantics in explanations. There are many practical situations where representations rather than concepts matters. For instance, the Promptbreeder from GoogleMind (Fernando et al., 2023) that generates various prompts (representations) to get the same answer (concept), or the use of Codex in teaching (Finnie-Ansley et al., 2022) which produces various programs (representations) having the same behaviour (concept). Having said that, our main contribution is not practical, it is rather the insight about highlighting the distinction between teaching concepts and teaching representations, and showing theoretically and experimentally how small the witness sets can be, given the existing redundancy.

2 Machine teaching definitions

Various models of machine teaching have been proposed, e.g., the classical teaching dimension model (Goldman & Kearns, 1995), the optimal teacher (Balbach, 2008), recursive teaching (Zilles et al., 2011), preference-based teaching (Gao et al., 2017), or no-clash teaching (Fallat et al., 2023). These models differ mainly in the restrictions they impose on the learner and the teacher to avoid collusion. The common goal is to keep the *teaching dimension*, i.e., the size of the largest witness, $\max_{c \in C} |T(c)|$, as small as possible.

In all these models, C is a set of concepts, with each $c \in C$ a subset of a domain X of examples. We now allow more than one *representation* for the same concept, thus with a set R of representations constituting a multiset of the concepts C. The teacher $T : R \to W$ is now a partial injective mapping from representation $r \in R$ to a set of labelled examples $w \in W$ (we will assume $W \subseteq 2^{X \times \{0,1\}}$ so that w is a finite set of negatively or positively labelled examples from X) and the learner $L : W \to R$ is a partial mapping in the opposite direction. Note that W can be restricted in various ways, e.g., in some experiments of Sect. 4.2 each element of W has at most 5 examples and these are always positively labelled. As usual in MT and ML, it is assumed that teacher and learner share the consistency graph G_R on vertex set $R \cup W$ with a representation $r \in R$ adjacent to a witness $w \in W$ (and thus $rw \in E(G_R)$ an edge) if r and w are consistent, i.e., with positive (resp. negative)

examples in w being members (resp. non-members) of r. Naturally, r must be consistent with $T(r)$ and $L(w)$ must be consistent with w, and a successful teacher-learner pair must have $L(T(r_1)) = r_2$ such that $[r_1] = [r_2]$. In what follows, we assume L and T use the same representation language and we impose that $L(T(r)) = r$.

From a graph theoretical point of view, the teacher and learner mappings are matchings in G_R between representations and sets of labelled examples. Two vertices are called twins if they have the same set of neighbours. Usually there is a bijection between the equivalence classes of twins of R in G_R, that we denote by R/W, and the set of concepts C. However, if the witness set W is too sparse then two representations forming distinct concepts may be consistent with the exact same subset of W and in that case R/W will be a coarsening of C.

We will be comparing two models for teaching representations, that we call Eager and Greedy. Both assume some natural size functions on representations R and on witnesses W, and use these to arrive at two total orderings $\prec_R$ on R and $\prec_W$ on W. When the representations in R are expressed in some description language, which can be English or a programming language or some set of mathematical formulas, then a representation is a finite string of symbols drawn from some fixed alphabet. We assume a total order on this alphabet which is used to derive a lexicographic order on strings over this alphabet, with shorter strings always smaller, also called shortlex. Analogously, we can define a total ordering on witnesses.

The input to these algorithms is then an ordered consistency graph denoted by the 3-tuple $(G_R, \prec_R, \prec_W)$. We say that a vertex (or representation/witness) appears earlier than another, or that it is simpler than another, if it appears earlier in these orderings. In the Eager model we have $L(w) = r$ for r being the earliest representation (in the order $\prec_R$) consistent with w. The teacher, knowing that the learner behaves in this way, will construct the mapping $T : R \to W$ iteratively as follows:

Eager: go through all of W in the order of $\prec_W$, and for a given w find the earliest $r \in R$ with $rw \in E(G_R)$, and if $T(r)$ is not yet defined then set $T(r) = w$

Telle et al. (2019) use the Eager model to teach programs in the Turing-complete language P3, by witnesses containing specified I/O-pairs. Many P3 programs have equivalent I/O-specifications, and we can observe that in the Eager model only the earliest representation (program) of any concept (I/O-specification) is taught. In order to teach all representations we introduce the Greedy model, where we make a slight change to the way the teacher constructs its mapping, as follows:

Greedy: go through all of W in the order of $\prec_W$, and for a given w find the earliest $r \in R$ with $rw \in E(G_R)$ such that $T(r)$ is not yet defined, and if such r exists then set $T(r) = w$

Also the learner must slightly change its protocol in the Greedy model, by removing a representation r from further consideration once it has been taught. In other words, given witness w search for the earliest consistent representation that has not yet been taught, call this r and set $L(w) = r$. Teaching and Learning in the Greedy model is thus done following an order on witnesses, similar to what happens in the Recursive teaching model of Zilles et al. (2011) (teach sequentially and remove concepts that have already been taught), and using also an order on the representations, similar to what is done in the Preference-Based teaching model of Gao et al. (2017) (return the most preferred remaining concept consistent with a given witness).

In classical machine teaching, the goal is to minimize the teaching dimension of a concept class, i.e. to find a legal teacher mapping $T : C \to W$ where the maximum of $|T(c)|$

over all $c \in C$ is minimized. When there is a total mapping $T : R \to W$ we can do the same for teaching representations, except we use *size*($T(c)$) rather than cardinality, as in Optimal-1 defined below. However, a total mapping $T : R \to W$ may not exist, and in that case we define an alternative measure called Optimal-2 (see below) being the maximum number of representations covered when constrained to partial teacher mappings that cover the maximum number of concepts. We thus define Optimal in two flavors. Note that we do not intend this as a bona fide teacher/learner protocol, rather it is used to compare how far the Eager and Greedy teaching strategies are from the optimal achievable. Nevertheless, there will be some 'optimal' ordering of the witnesses, such that if the Greedy teacher follows this ordering then it would result in the Optimal mapping. Assuming Optimal returns a mapping $T_{opt} : R \to W$ and we have a total ordering $\prec_R$ on R, then the ordering of W given by: w earlier than w' if and only if $T_{opt}^{-1}(w)$ is earlier than $T_{opt}^{-1}(w')$ in $\prec_R$, is such an 'optimal' ordering.

Optimal-1 is the minimum max size witness over all teacher mappings, with the sole restriction that the teacher mapping is injective and assigns a consistent witness to every representation. This value can be computed in polynomial time by a binary search for the smallest k s.t. the induced subgraph G_R^k of G_R, on R and all witnesses of size at most k, has a matching saturating R i.e. containing all of R. See (Simon & Telle, 2024, Theorem 38), where this parameter is called SMN (Saturating Matching Number), for a faster algorithm.

As Optimal-1 is undefined when G_R has no matching saturating R, we define *Optimal-2* as the maximum number of representations covered by any matching that maximizes the number of concepts covered. It is not clear that this number is computable in polynomial time, but for most graphs in the experimental section we could compute it.

3 Formal comparisons

We show several results comparing the performance of the three algorithms Eager, Greedy and Optimal, starting with Fig. 1 giving the simplest possible consistency graph where Eager, Greedy and Optimal-1 behave differently. Let us explain Fig, 1. Since w_2, w_3 are adjacent to r_1 that is taught using w_1, Eager will not use them. Greedy will use w_2 but

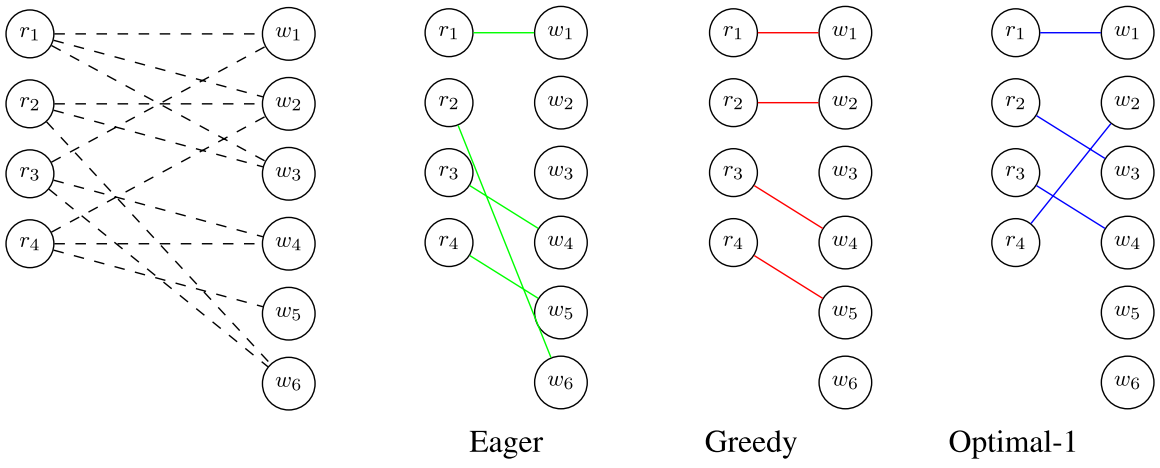


Fig. 1 Simplest possible ordered consistency graph ($\prec_R$ and $\prec_W$ given by the indices; witness w_i has size i) showing that Eager, Greedy and Optimal-1 can require different max teaching sizes (witnesses of size 6, 5 and 4 respectively)

not w_3 as it is only consistent with r_1, r_2 that have already been taught using w_1, w_2. Optimal-1 is not constrained to follow the given orderings, and finds a teacher mapping where the least simple witness has index as low as 4. Note that if Greedy uses the ordering $w_1 < w_3 < w_4 < w_2$ then it finds this Optimal-1 ordering. This raises the question of how far from optimal the Greedy teacher mapping can be, motivating the study of the GMN (Greedy Matching Number) in Simon and Telle (2025).

The teacher mapping $T : R \to W$ for Greedy, as defined in the previous section, is computed by an iterative procedure whose outer loop follows the witness ordering, and with an inner loop following representation ordering. Let us call it $T_W : R \to W$. Consider the alternative $T_R : R \to W$ which switches the role of these two orderings: go through R in the order of $\prec_R$, and for a given representation r find the earliest $w \in W$ with $rw \in E(G_R)$ such that $T_R^{-1}(w)$ is not yet defined, then set $T_R(r) = w$ and continue with the next representation (if no such w exists then drop this r).

Theorem 1 The teacher mappings returned by Greedy following $\prec_W$ and the alternative following $\prec_R$ are the same.

Proof We prove, by induction on $\prec_W$, the statement (*): "for any $w \in W$ if $T_W(r) = w$ then also $T_R(r) = w$." Let w_1 be the earliest witness. We have $T_W(r') = w_1$ and when assigning T_R clearly w_1 is the earliest neighbor of r' with $T_R^{-1}(w_1)$ undefined so that $T_R(r') = w_1$. For the inductive step, we assume the statement (*) for all witnesses earlier than w, and assume that $T_W(r) = w$. By the inductive assumption (*) we know that no witness earlier than w will by T_R be assigned to r. This means that when assigning $T_R(r)$ then w is the earliest neighbor of r with $T_R^{-1}(w)$ undefined, and thus we have $T_R(r) = w$ as desired. □

Since the Greedy teaching is only a slight change to the Eager teaching, it is easy to see that for any $r \in R$, if Eager assigns $T_E(r) = w_E$ then Greedy will also assign some $T_G(r) = w_G$, and in the order $\prec_W$ we could have w_G earlier than w_E, but not the other way around.

Theorem 2 On representations/concepts taught by Eager, Greedy will never use a larger witness.

Proof This since $T_E(r) = w_E$ implies that r is the earliest representation consistent with w_E, so when the Greedy teacher mapping is computed then at reaching w_E the first representation tried is r, and if $T_G(r)$ has already been defined it will be for an earlier and thus no larger witness. □

We show that for any concept class we can add redundant copies of representations consecutive in $\prec_R$ to make Eager and Greedy perform almost identical.

Theorem 3 For any concept class R (i.e. each concept has a unique representation) on witness set W and orders $\prec_R, \prec_W$, we can add less than a total of $|W|$ copies of representations to get R', and make $\prec_{R'}$ an extension of $\prec_R$ with all copies consecutive, such that Eager

teaches the same on $(G_R, \prec_R, \prec_W)$ as on $(G_{R'}, \prec_{R'}, \prec_W)$ and using the same witnesses as Greedy on $(G_{R'}, \prec_{R'}, \prec_W)$ on all common concepts.

Proof Consider the ordered consistency graph $(G_R, \prec_R, \prec_W)$ and for each representation $r \in R$ compute $f_r = |\{w \in W : rw \in E(G_R) \wedge (r'w \in E(G_R) \Rightarrow r \prec_R r')\}|$, i.e. the number of witnesses for which r is the earliest neighbor. Let R' be the representation class corresponding to the ordered consistency graph $(G_{R'}, \prec_{R'}, \prec_W)$ where for each $r \in R$ we add $f_r - 1$ copies of r and extend the order $\prec_R$ to $\prec_{R'}$ by putting all the copies consecutively right after r, so there are f_r consecutive copies total of r. Note that, as the sum of f_r over all $r \in R$ is $|W|$ (each witness has a single earliest r) we add less than $|W|$ new representations.

For each $w \in W$ define r_w to be the earliest neighbor of w in G_R and let r'_w be the earliest neighbor in $G_{R'}$. For simplicity, when we say $G_R, G_{R'}$ we mean the ordered consistency graphs. We prove the following loop invariant on any witness $w \in W$ by induction over $\prec_W$: "knowing r_w has f_{r_w} witnesses for which it is the earliest, let w be the kth such witness. Both on G_R and $G_{R'}$, if $k = 1$ then Eager will assign r_w to w, and if $k > 1$ Eager will not use w. Greedy on $G_{R'}$ will assign r to w for r being the kth copy of r_w in the $\prec_{R'}$ order." The loop invariant is trivially true for the first witness. For the inductive step, we know r_w is the earliest neighbor of w and that w shares this property with f_{r_w} witnesses. If w is the first of these witnesses, then it is clear that both Eager and Greedy will assign r_w to w, as the loop invariant tells us no earlier witness has been assigned to r_w. If w is the kth copy of r_w for $k \geq 2$ then Eager will not use w, neither on G_R nor $G_{R'}$, while for Greedy on $G_{R'}$ as there are at least k copies of r_w all in consecutive order then Greedy will assign w to the kth copy as it must be available by the loop invariant. We have thus proven the loop invariant, and this implies the statement of the Theorem as it encompasses all representations taught by both algorithms. □

Greedy can be worse than Optimal, when size is cardinality. In Fig. 2, there are 4 concepts c_0, c_1, c_2, c_3 and 2 ground elements x_1, x_2. Witnesses consist of labelled examples and size is just cardinality (the number of examples in a witness). There are thus 4 witnesses of size 1, and the optimal mapping given by the blue dotted lines uses these 4, but Greedy assigns the red mapping, which uses a witness of cardinality two.

3.1 Optimal-1 can be more than the trivial bound, for size being cardinality

In this subsection we assume the size of a witness is its cardinality (number of examples), as in teaching dimension.

Note that for set of representations R and ground domain X, defining $triv_{pos}(R)$ (the 'trivial' bound for positively labelled examples) to be the smallest value k such that $\sum_{i=0}^{k} \binom{|X|}{i} \geq |R|$ then for witnesses on positively labelled examples Optimal-1 must use a witness of cardinality (size) $triv_{pos}(R)$, since the sum is the number of witnesses on positively labelled examples of cardinality up to $triv_{pos}(R)$, and we need that many witnesses to cover R.[1] However, there are concept classes where we need more, e.g. take $X = \{1, 2, 3, 4\}$ and $R = \{\emptyset, 1, 2, 3, 4, 12, 14, 24, 124\}$. Note $|R| = 9$, $|X| = 4$ and

[1] In Simon and Telle (2025) the parameter $triv_{pos}(R)$ is given the name GMN'

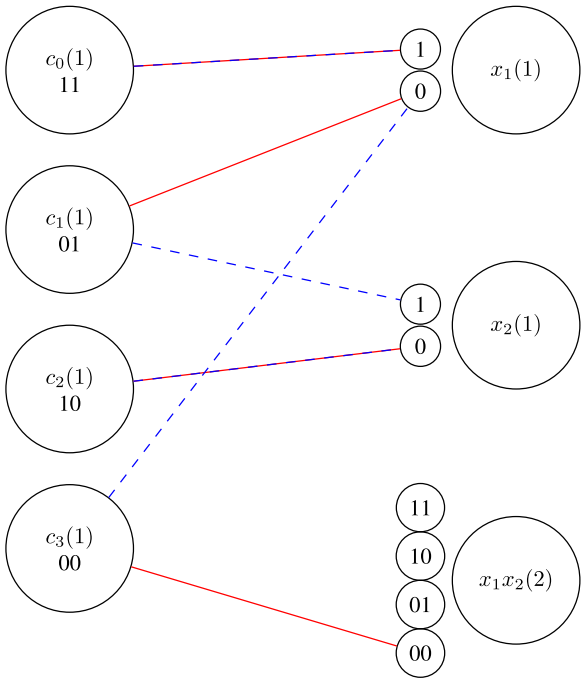


Fig. 2 Comparison of Greedy versus Optimal, when size is cardinality

thus $triv_{pos}(R) = 2$. In the bipartite consistency graph limited to witnesses of cardinality at most 2 we have 9 vertices on the *R*-side and we have 11 vertices on the other side corresponding to all subsets of *X* of size at most 2, but 3 of these latter vertices, corresponding to $\{13, 23, 34\}$, have no neighbors in *R* and thus since $11 - 3 < 9$ no matching saturating *R* can exist.

Similarly, for witnesses on labelled examples, define $triv_{lab}(R)$ (the trivial bound for labelled examples) to be the smallest value *k* such that $\sum_{i=0}^{k} \binom{|X|}{i} * 2^i \geq |R|$. We show a case where Optimal-1 must use a witness of cardinality (size) larger than $triv_{lab}(R)$. Assume a set of concepts *R* and ground elements $X = \{a, b, c, d, e, f, g\}$. Let $r_1 = \{a, b\}$ and let $r_2 = \{b, c\}$. Now, from the subset $\{d, e, f, g\}$ select 13 unique subsets, and let these be $\{r_3, r_4, ..., r_{15}\}$. By letting $R = \{r_1, r_2, ..., r_{15}\}$ we have $triv_{lab}(R) = 1$. However, the only representations consistent with the witness sets, of cardinality one, on the positive examples (*a*, 1), (*b*, 1), (*c*, 1) are r_1 and r_2. Hence at most two of these three witness sets can be used to teach a representation. With 15 representations, and 15 witness sets of size ≤ 1, and at least one unused witness set, we know we will have to use a witness set of cardinality ≥ 2.

3.2 Greedy can be arbitrarily worse than Optimal-1

We now know that Eager can never be better than Greedy, and that Optimal-1 sometimes leaves small witnesses unused. In the experimental section, we will see that Greedy is often close to Optimal-1 , but we now show that there are also cases where Greedy is far worse than Optimal-1 , in that the mapping it finds will use a witness of larger maximum size.

The most restricted case is when the size of a witness is equal to its cardinality, i.e. equal to the number of examples it contains, so-called teaching dimension, and even then Greedy can do worse than Optimal-1 , as in Fig. 2. In this restricted case we are not able to show that it can do worse by an arbitrary amount, and must leave this as an open question. However,

for some less restricted size functions, when we allow the size of ground elements to vary, we can show the following.

Theorem 4 When there is a bound on the number of witnesses of any size, then for any *t*, there exists a size ordered consistency graph where Greedy uses a witness of size G_{max} while Optimal-1 will not use a witness of size larger than O_{max} and where $G_{max} - O_{max} \geq t$.

Proof We prove it by construction, for a graph where we have only one representation for each concept. Assume there are at most q witnesses of any size. Let concept ordering be $c_1, c_2, ...c_k$, and let witness ordering be $w_1, w_2, w_3, ...w_{qt+k}, ...$ with $s(w_i) \leq s(w_{i+1})$, for all *i*. Let c_1 be consistent with w_1 and w_2, and let c_2 be consistent with w_1 and w_{qt+k} (but not consistent with any witness in between, which could happen if e.g. there are that many ground elements) and let c_i for $3 \leq i \leq k$ be consistent with $w_i, w_{i+1}, ..., w_{qt+i}$. The Optimal matching will assign c_1 to w_2 and c_2 to w_1 and c_i to w_i for any $3 \leq i \leq k$, thus using max size witness with index *k*. Greedy will assign almost the same matching except it starts with c_1 to w_1 and then c_2 to w_{qt+k} since this is the only one available, thus using max size witness with index $qt + k$. As we have at most q witnesses of each size we will have $s(w_{qt+k}) - s(w_k) \geq t$, and we are done with the proof. □

4 Experiments

In this section we observe quantitatively how each teaching protocol behaves in a variety of languages with various types of redundancy. We consider 6 distinct languages, whose characteristics are summarized in Table 1. These range from boolean expressions in 3-DNF with no redundancy, to cases of 3-Term DNF with varying redundancy, and two examples over the universal language P3.

Let us first describe how we measure redundancy, and then the languages. To estimate how redundant a language is, we define *Redundancy* as $1 - Uniqueness$, where *Uniqueness* is computed as the average for all concepts of $1/|R_c|$, with R_c the set of all equivalent representations for concept *c*. Formally:

Table 1 Domain features: 'Witness' is the constraint about the size of each witness set

Domain	$\|R\|$	$\|C\|$	Witness	$\|W\|$	Redundancy	Redund. Spread
3-DNF	256	256	$\|w\| \leq 5$	3488	0	15.13
3-Term DNF	2952	246	$\|w\| \leq 5$	3488	0.727	13.28
3-Term DNF with permutations	79,158	246	$\|w\| \leq 5$	3488	0.9896	11.41
			$\|w\| = 5$	1792	0.9896	7.04
3-Term DNF with permutations and duplicates	158,316	246	$\|w\| \leq 5$	3488	0.9948	10.42
P3	$1.9 * 10^9$	?	$s(w) \leq 6$	6548	?	?
Small-P3	1267	106*	$s(w) \leq 4$	260	0.331	2.97

For P3 we do not know G_R nor *C*. (*For small-P3 we do not know $|C|$, so we show $|R/W|$.)

$$\text{Redundancy} = 1 - \frac{1}{|C|} \sum_{c \in C} \frac{1}{|R_c|} \ . \tag{1}$$

We now define a measure called Redundancy Spread that will better evaluate the effect of redundancy. Note Theorem 3 suggests that Eager performs well compared to Greedy when, for each witness *w*, in the set R^i_w of the *i* earliest representations consistent with *w* there are few different concepts in R^i_w. Assuming the $\prec_W$ ordering is $w_1, w_2, \ldots$ we therefore define *Redundancy Spread* as the average number of different concepts in $R^i_{w_i}$. We use $R^i_{w_i}$ for w_i because at most *i* values of $T : R \to W$ have been defined when Greedy or Eager consider w_i. Formally, if for each witness w_i we let $r^j_{w_i}$ be the *j*-th representation consistent with w_i, and recalling that $[r]$ is the concept equivalence class *r* belongs to, the definition of Redundancy Spread becomes:

$$\text{Redundancy Spread} = \frac{1}{|W|} \sum_{w_i \in W} \left| \left\{ [r^j_{w_i}] : j \in [1 \ldots i] \right\} \right| .$$

4.1 Boolean expressions

We experiment with Boolean expressions, since in this domain it is possible to decide whether two representations belong to the same equivalence class, i.e., correspond to the same concept. Furthermore, in this language, we are able to measure the degree of redundancy present in the set of representations. In all our experiments, we consider the ordered alphabet of 3 Boolean variables $\{a, b, c\}$ and every representation in *R* is expressed in DNF (Disjunctive Normal Form), i.e. OR of AND-terms. Every variable occurs at most once in a term, possibly preceded by the negation operator $\neg$. For example, $(a \wedge b \wedge \neg c)$ is a valid ground term.

Let us first describe the witnesses, which are similar across all experiments. Each example is a pair consisting of a binary string of length three (the input) and a bit (the output), such as (001, 1) indicating that when $a = 0, b = 0, c = 1$ the value of the function should be 1. There are thus 16 witnesses of cardinality one, and $16 * 14/2 = 112$ witnesses of cardinality two, as we do not allow for self-incongruent witnesses and the order of examples is irrelevant. For these experiments we use witnesses of cardinality up to 5, and only positive examples, which implies $|W| = 3448$.

$\prec_W$: The witnesses in the collection *W* are ordered by their increasing teaching size. The *size function* $s : W \to \mathbb{N}$ is defined for a witness *w* as the number of examples in *w* times 4 (the number of bits) plus the number of 1 s in the inputs of the examples. This scheme may be useful for a situation with many variables, where it could be more efficient to just identify the position of the 1 s in a long binary string.

In the first experiment, *3-DNF*, we build a scenario without redundancy. We consider all 256 Boolean functions on three variables as the set $R = C$, as follows. A Boolean function ϕ on *a*, *b*, *c* is uniquely defined by a truth table that gives the truth values for the 8 possible truth assignments to the three variables. For any assignment (v_a, v_b, v_c) to (*a*, *b*, *c*) where $\phi(v_a, v_b, v_c) = 1$ we include the corresponding term in the representation r_ϕ of ϕ in *R*. For instance, if ϕ is logically equivalent to "*a* and *b*" then this is represented as $r_\phi = (a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c)$.

In our second experiment, *3-term DNF*, we introduce redundancy in R, and do this in a way that will shrink the concept space to have smaller cardinality $|C| = 246$. Every representation is now a disjunction of up to 3 ground terms, and each term contains one, two or three distinct variables, each possibly negated. For a single term we have six representations of one variable, $12 = 6 * 4/2$ representations of two variables, and 8 representations for three variables, thus 26 in total. For two terms we have $26 * 25/2 = 325$ representations, and for three terms we have $26 * 25 * 24/3! = 2,600$. We also have a single representation (False) with no terms, and thus $|R| = 1 + 26 + 325 + 2,600 = 2,952$.

For the third experiment, *3-term DNF with permutations*, we want to increase Redundancy and decrease Redundancy Spread. The set of representations is similar to 3-term DNF, but here we allow for any permutation of variables within each term. For example, if $(b \wedge \neg c) \in R$, also $(\neg c \wedge b) \in R$. This drastically increases the cardinality of R, while Redundancy also increases substantially and Redundancy Spread decreases. This third experiment is carried out also against a smaller witness set, with witnesses of cardinality 5 only, to see the effects of a large decrease in Redundancy Spread.

Finally, the last Boolean experiment, *3-term DNF with permutations and duplicates*, is similar to the previous one, 3-term DNF with permutations, but here each representation occurs consecutively twice in the ordered collection R, for yet another variation of redundancy.

$\prec_R$: Representations in R are ordered by increasing size $s(r)$ defined as the number of literals in r plus the number of negated variables plus the number of disjunction symbols; e.g., $s((a \wedge b \wedge c) \vee (a \wedge b \wedge \neg c)) = 8$. Two representations of same size are further ordered by the increasing number of terms, and by the $\langle a, b, c, \neg, \vee \rangle$-induced lexicographic order.

4.2 Universal language

We also analyze the new teaching frameworks for the Turing-complete language P3. P3 is a simple string manipulation language inspired by P", introduced by Corrado Böhm ((Böhm, 1964)), the first GOTO-less imperative language proved Turing-complete, i.e., universal. The most popular variant (Brainfuck) has 8 instructions in total. We employ another version called P3, also universal and having just 7 instructions: $<> + - []$o. We consider P3-programs that take binary inputs and generate binary outputs. A representation is a P3 program, such as the following one, which performs the left shift operation (e.g., on input 10010 we get output 00101): $> [$o $>] < [<] >$ o. Regarding the semantics of P3, we refer to the work by Telle et al. (2019).

A witness is a set of pairs of binary strings defining an input and an output, and we use only positive examples. For example, the witness $w = \{\langle 0100, 00\rangle, \langle 001, \rangle, \langle 00, 00\rangle\}$ is consistent with any program that on input 0100 outputs 00, on input 001 has no output and on input 00 outputs 00.

$\prec_W$: We use a simple encoding of example sets (the number of bits) as the size function $s(w)$, and break ties in a deterministic way. The witness w above has size $s(w) = 13$.

$\prec_R$: The lexicographic order on instructions $<> + - []$o defines the $\prec_R$ length-lexicographic P3 programs ordering.

We perform two experiments, called P3 and small-P3. In P3 we repeat the experiment from Telle et al. (2019). Note that computing equivalence classes of programs is undecidable in general, and we cannot decide if a program enters an infinite loop. We limit the num-

ber of timesteps for the computation of each program before breaking. In P3 experiment, the set of representations/programs R is a potentially infinite set, but Eager and Greedy end up never going beyond the $1.9 * 10^9$ first programs. The witness set W contains all sets of size at most 6. As we cannot compute the consistency graph and run any version of Optimal, we conduct another experiment, *small-P3*.

In small-P3 we start with all self-congruent witnesses of size at most 4, and the first 10,000 programs in $\prec_R$. We filter out every witness without consistent program and every program without consistent witness. A set of 1,267 programs is then R, and a set of 260 witnesses is W. We compute the consistency graph G_R and use R/W as the concept class, i.e., considering two programs equivalent if they are consistent with the exact same subset of W.

4.3 Experimental results

The main characteristics of the chosen languages is given in Table 1, while Table 2 shows the performance of Eager, Greedy and Optimal in terms of the number of representations taught, the number of concepts taught, the maximum size of a used witness, and the maximum index of a used witness. Additionally, Table 3 reports on the percentage of cases where Greedy is able to teach concepts with simpler witnesses (lower max index and lower max size) than those used by Eager, and Fig. 3 shows that Redundancy Spread to a large degree explains this behaviour.[2]

Table 2 Results for the teaching frameworks across all domains

Domain	Witness	Algorithm	Reprs. taught	Concepts taught	Max. $\{s(w)\}$	Max. $\{i(w)\}$
3-DNF	$\lvert w\rvert \leq 5$	Eager	219	219	30	3488
		Greedy	256	256	16	328
		Optimal-1	256	256	16	256
3-Term DNF	$\lvert w\rvert \leq 5$	Eager	170	170	30	3466
		Greedy	2895	246	30	3481
		Optimal-1	2952	246	28	2952
3-Term DNF with permutations	$\lvert w\rvert \leq 5$	Eager	170	170	30	3466
		Greedy	3488	189	30	3488
		Optimal-2	3488	246	30	3488
	$\lvert w\rvert = 5$	Eager	170	170	30	1770
		Greedy	1792	177	30	1792
		Optimal-2	1792	246	30	1792
3-Term DNF with permutations and duplicates	$\lvert w\rvert \leq 5$	Eager	170	170	30	3466
		Greedy	3488	178	30	3488
		Optimal-2	3488	246	30	3488
P3	$s(w) \leq 6$	Eager	2032	2032	6	6512
		Greedy	6548	?	6	6548
small-P3	$s(w) \leq 4$	Eager	53	53	4	202
		Greedy	225	65	4	260
		Optimal-2	≥ 214*	106	4	260

$s(w)$ is the size of a witness w; $i(w)$ is the index of witness w in the order $\prec_W$. (*For small-P3, 'Reprs. taught' by Optimal-2 is only a lower bound.)

[2] The materials to reproduce the experimental results are stored in a downloadable directory shared at https://bit.ly/MLJournal-MTOfReprs-Supplementary .

Table 3 Greedy beating Eager: common concepts with earlier (lower max index) and simpler (smaller size) witness

Domain	Witness	% Witness Index Lower	% Witness Size Smaller
3-DNF	$\|w\| \leq 5$	94.98	94.06
3-term DNF	$\|w\| \leq 5$	97.65	97.06
3-Term DNF with permutations	$\|w\| \leq 5$	90.59	88.24
	$\|w\| = 5$	40.00	30.00
3-Term DNF w/ permuts. and duplicates	$\|w\| \leq 5$	84.12	81.18
P3	$s(w) \leq 6$	13.98	6.55
small-P3	$s(w) \leq 4$	24.53	18.87

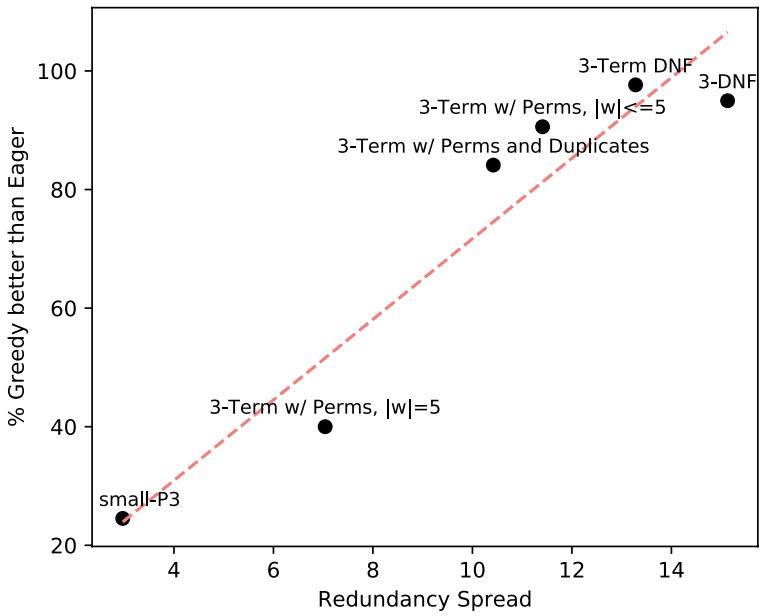


Fig. 3 Relation between *Redundancy Spread* and *% Greedy better than Eager* in terms of using a witness of lower max index. Data from Tables 1 and 3. The red line is the best fit linear approximation

As Table 2 shows, for most of the Boolean experiments, Greedy manages to teach as many representations as the amount of witness sets available. We can observe that Greedy is always able to teach more concepts than Eager, a method specialized in teaching unique concepts. Greedy teaches many more concepts and representations often by making use of witnesses that are larger or later in the order. Yet, even in scenarios without redundancy, such as 3-DNF, where Eager might be considered suitable, Greedy still teaches some more concepts, with earlier and smaller witnesses. Greedy performs almost as good as Optimal-1 and Optimal-2 in terms of the number of representations taught and the largest size and highest index of witnesses used. However, Optimal-2 (used rather than Optimal-1 whenever not all of R can be covered) always covers a higher concept number than Greedy. Figure 3 shows the correlation between the percentage of cases where Greedy outperforms Eager in terms of lower max index on used witness and the Redundancy Spread. It confirms Redundancy Spread (that can be computed without running either Eager or Greedy) as a strong metric for predicting that Greedy will outperform Eager.

Analyzing now the universal language domain, we see that Greedy allows to teach over three times more (almost five for small-P3) programs than Eager. For P3 we do not have the consistency graph and thus cannot compute any version of Optimal. Quite remarkably,

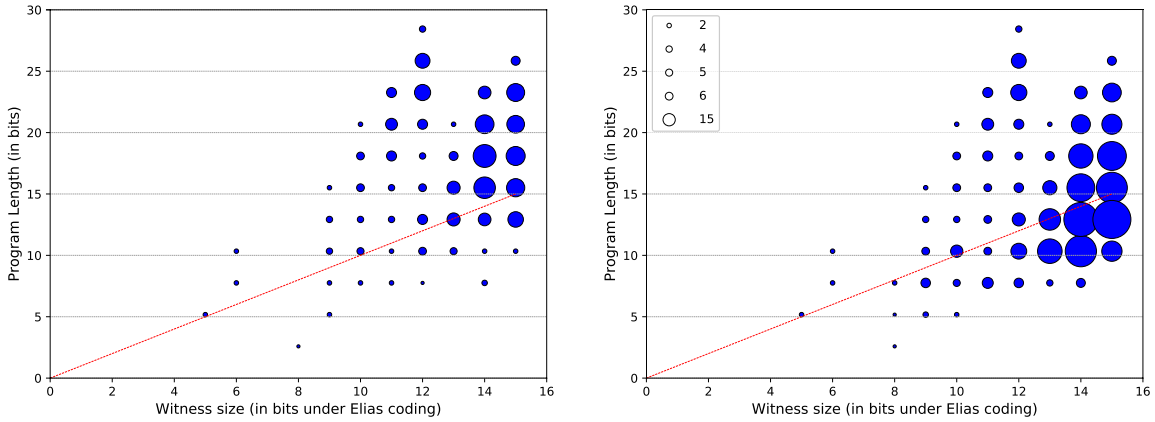


Fig. 4 Eager (left) versus Greedy (right): Program length versus witness size, using Elias coding (Elias, 1975)

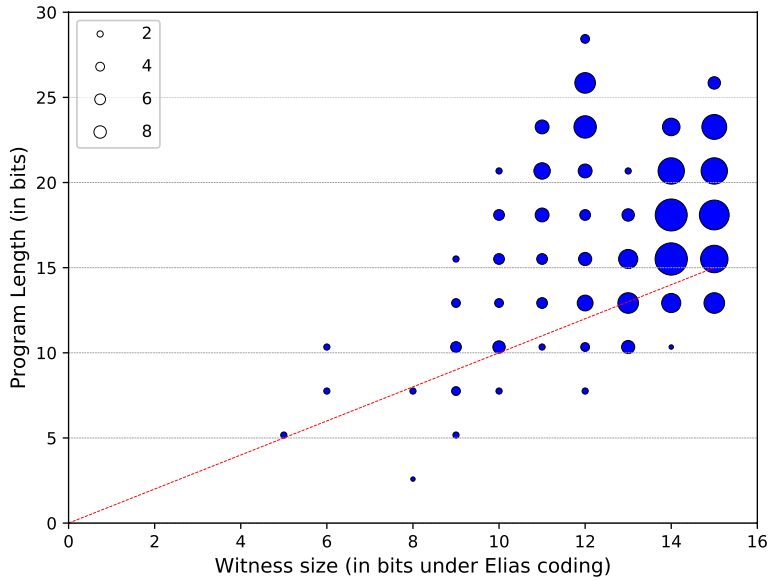


Fig. 5 Program length versus witness size of Greedy, using Elias coding iciteDBLP:journalsspstitspsElias75, for the programs taught by both Eager and Greedy. Circles above the unit diagonal denote witness smaller than program, with size of circle corresponding to the number of programs

in Fig. 4 we see that for P3 programs the witness sets of Greedy are usually smaller than the programs they identify. A similar behavior for fewer programs was noted for Eager by Telle et al. (2019). The number of programs taught by the Greedy algorithm is significantly higher than Eager (6548 versus 2032). When we focus on the performance of Greedy on the intersection of programs that are taught by both strategies (Fig. 5), we can better appreciate how the Greedy algorithm is able to teach concepts using witness sets of smaller size.

For small-P3 we use R/W as a proxy for C and note that Optimal-2 is able to cover all these 106 concepts, but seemingly at the price of fewer representations (214 versus 225 for Greedy). However, note that our implementation of Optimal-2 is actually only able to optimize the number of concepts covered. For the further maximization of representations that Optimal-2 requires we were not able to find a polynomial-time algorithm, and we must leave it as an open problem whether this is NP-hard. On concepts common to Eager and Greedy, the latter finds smaller witnesses on a relatively low percentage of the total for

Small-P3 (see Fig. 3) but even lower for P3. Since for P3 there is no consistency graph, we cannot compute Redundancy Spread, but we hypothesize that the small values of 13.98% (lower max index) and 6.55% (lower max size) where Greedy beats Eager, are due to an even lower Redundancy Spread of its consistency graph.

5 Discussion

The notion of redundancy we explore is very old, dating back to numeral systems (e.g., IIII and IV being two alternative representations of the number 4 in Roman numerals). Leibniz was interested in prime decomposition as a tool for his *characteristica universalis* in which the representations and concepts could have a bijective mapping. In numeration systems, this is possible. Indeed, it was Böhm himself (the 'father' of P3) who proved that every natural number has a unique representation in bijective base-k ($k \geq 1$) (Böhm, 1964). However, we also know that in many other languages, such as Turing-complete languages, this is not possible. And even for some other formal languages for which the equivalence classes can be calculated effectively, using a canonical representative for each concept would lead to languages that are very cryptic, such as a numbering of an enumeration—a coding—of all the equivalence classes. This is the first time, to our knowledge, that the problem of teaching has been formally studied without assuming the bijection of representations and concepts, or even assuming that teacher and learner can calculate the equivalence classes.

The more realistic view of teaching adopted in this paper modifies some key elements of the traditional machine teaching setting and gives more relevance to the representation language. Given the high predictiveness for some teaching indicators, the new metric of redundancy spread, introduced in this paper, could be used to anticipate what language representations are better than others for teaching and learning. The size of representations becomes a natural way of imposing an order on them, which can be used in the teaching protocols. Some of the protocols seen here, such as Greedy, are computationally expensive. However, none of them requires the calculation of equivalence classes, which may also be expensive—or undecidable. The key finding is that *Greedy, a protocol designed to teach all representations, ends up teaching more concepts than Eager, which aims at only teaching concepts.* Our results reveal a wide spectrum between these two protocols, and future work could explore other teaching protocols for representations that are somewhat in between, or have better theoretical or statistical properties.

Author Contributions J.A.T. and J.H.O. wrote the main basis of the manuscript. J.A.T. and D.G. prepared figure 1; B.H. prepared figures 2 and 3; C.F. and D.G. prepared figures 4 and 5. All authors reviewed the manuscript.

Funding Open access funding provided by University of Bergen (incl Haukeland University Hospital). This work was funded by the Norwegian Research Council grant 329745 "Machine Teaching for Explainable AI," CIPROM/2022/6 (FASSLOW) funded by Generalitat Valenciana, the EC H2020-EU grant agreement No. 952215 (TAILOR), and Spanish grant PID2021-122830OB-C42 (SFERA) funded by MCIN / AEI / 10.13039/501100011033 and "ERDF A way of making Europe."

Data Availability The materials to reproduce the experimental results are stored in a downloadable directory shared at https://bit.ly/MLJournal-MTOfReprs-Supplementary .

Declarations

Conflict of interest The authors declare no Conflict of interest.

References

Alpuente, M., Comini, M., Escobar, S., Falaschi, M., & Iborra, J. (2010). A compact fixpoint semantics for term rewriting systems. *Theoretical Computer Science, 411*(37), 3348–3371.

Balbach, F. J. (2008). Measuring teachability using variants of the teaching dimension. *Theoretical Computer Science, 397*(1–3), 94–113.

Böhm, C. (1964). On a family of Turing machines and the related programming language. *ICC Bulletin, 3*(3), 187–194.

Elias, P. (1975). Universal codeword sets and representations of the integers. *IEEE Transactions on Information Theory, 21*(2), 194–203.

Enflo, P., Granville, A., Shallit, J., & Yu, S. (1994). On sparse languages l such that ll= σ. *Discrete Applied Mathematics, 52*(3), 275–285.

Fallat, S., Kirkpatrick, D., Simon, H. U., Soltani, A., & Zilles, S. (2023). On batch teaching without collusion. *Journal of Machine Learning Research, 24*, 1–33.

Fernando, C., Banarse, D., Michalewski, H., Osindero, S., & Rocktäschel, T. (2023). Promptbreeder: Self-referential self-improvement via prompt evolution. *CoRR*, arXiv:abs/2309.16797.

Ferri, C., Garigliotti, D., Håvardstun, B. A. T., Hernández-Orallo, J., & Telle, J. A. (2024). When redundancy matters: Machine teaching of representations. *CoRR*, arXiv:abs/2401.12711.

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., & Prather, J. (2022). The robots are coming: Exploring the implications of OpenAI codex on introductory programming. In J. Sheard & P. Denny (Eds.), *ACE '22: Australasian computing education conference, virtual event, Australia, February 14–18, 2022* (pp 10–19). ACM.

Gao, Z., Ries, C., Simon, H. U., & Zilles, S. (2017). Preference-based teaching. *Journal of Machine Learning Research, 18*, 31:1-31:32.

Goldman, S. A., & Kearns, M. J. (1995). On the complexity of teaching. *Journal of Computer and System Sciences, 50*(1), 20–31.

Hadley, R. F., & Cardei, V. C. (1999). Language acquisition from sparse input without error feedback. *Neural Networks, 12*(2), 217–235.

Håvardstun, B. A. T., Ferri, C., Hernández-Orallo, J., Parviainen, P., & Telle, J. A. (2023). XAI with machine teaching when humans are (not) informed about the irrelevant features. In Koutra, D., Plant, C., Rodriguez, M. G., Baralis, E., and Bonchi, F., editors, *Machine Learning and Knowledge Discovery in Databases: Research Track. ECML PKDD 2023*.

Muggleton, S., & De Raedt, L. (1994). Inductive logic programming: Theory and methods. *The Journal of Logic Programming, 19*, 629–679.

Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F., & Tausend, B. (1996). Declarative bias in ILP. *Advances in Inductive Logic Programming, 32*, 82–103.

Shafto, P., Goodman, N. D., & Griffiths, T. L. (2014). A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive Psychology, 71*, 55–89.

Simon, H. U., & Telle, J. A. (2024). MAP- and mle-based teaching. *Journal of Machine Learning Research, 25*, 96:1-96:34.

Simon, H. U., & Telle, J. A. (2025). The hierarchy of saturating matching numbers. *CoRR (to appear at Eurocomb 2025)*, arXiv:abs/2503.14061.

Sunde, J., Håvardstun, B. A. T., Kratochvíl, J., & Telle, J. A. (2024). On a combinatorial problem arising in machine teaching. In *Forty-first international conference on machine learning, ICML 2024, Vienna, Austria, July 21–27, 2024*. OpenReview.net.

Telle, J. A., Hernández-Orallo, J., & Ferri, C. (2019). The teaching size: Computable teachers and learners for universal languages. *Machine Learning, 108*(8–9), 1653–1675.

Vardi, M. Y. (2022). *Efficiency vs. resilience: Lessons from COVID-19*, pp. 285–289. Springer International Publishing, Cham.

Wang, C., Singla, A., & Chen, Y. (2021). Teaching an active learner with contrastive examples. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W., editors, *Advances in neural information processing systems 34: Annual conference on neural information processing systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17968–17980.

Whigham, P. A. (1996). Search bias, language bias, and genetic programming. *Genetic Programming, 1996*, 230–237.

Yang, S.C.-H., Vong, W. K., Sojitra, R. B., Folke, T., & Shafto, P. (2021). Mitigating belief projection in explainable artificial intelligence via Bayesian teaching. *Scientific Reports, 11*(1), 1–17.

Yu, S. (1988). Can the catenation of two weakly sparse languages be dense? *Discrete Applied Mathematics, 20*(3), 265–267.

Zhang, X., Bharti, S. K., Ma, Y., Singla, A., & Zhu, X. (2021). The sample complexity of teaching by reinforcement on q-learning. In *Thirty-fifth AAAI conference on artificial intelligence, AAAI 2021, thirty-third conference on innovative applications of artificial intelligence, IAAI 2021, the eleventh symposium on educational advances in artificial intelligence, EAAI 2021, Virtual Event, February 2–9, 2021*, pp. 10939–10947. AAAI Press.

Zhu, X., Singla, A., Zilles, S., & Rafferty, A. N. (2018). An overview of machine teaching. *arXiv preprint* arXiv:1801.05927.

Zilles, S., Lange, S., Holte, R., & Zinkevich, M. (2011). Models of cooperative teaching and learning. *Journal of Machine Learning Research, 12*(Feb), 349–384.

Paper III

Sunde, J., Håvardstun, B., Kratochvíl, J., & Telle, J. A.
Proceedings of the 41st International Conference on Machine Learning (ICML 2024), PMLR **235**, pp. 47305–47313 (2024)
URL: proceedings.mlr.press/v235/sunde24a.html

On a Combinatorial Problem Arising in Machine Teaching

Joakim Sunde [1] **Brigt Håvardstun** [1] **Jan Kratochvíl** [2] **Jan Arne Telle** [1]

Abstract

We study a model of machine teaching where the teacher mapping is constructed from a size function on both concepts and examples. The main question in machine teaching is the minimum number of examples needed for any concept, the so-called teaching dimension. A recent paper (Ferri et al., 2024) conjectured that a worst case for this model, as a function of the size of the concept class, occurs when the consistency matrix contains the binary representations of numbers from zero and up. In this paper we prove their conjecture. The result can be seen as a generalization of a theorem resolving the edge isoperimetry problem for hypercubes (Hart, 1976). Our proof is based on a generalization of a lemma of (Graham, 1970).

1. Introduction

In formal models of *machine learning* (Valiant, 1984) we have a concept class C of possible hypotheses, an unknown target concept $c^* \in C$ and training data given by correctly labelled random examples. The concept class C is given by a binary matrix M whose rows are concepts and whose column set is the domain of examples X, with $M(c,x) = 1$ if c is consistent with x labelled positively, i.e. with $(x,1)$ rather than with $(x,0)$. In formal models of *machine teaching* a set of labelled examples w called a *witness* is instead carefully chosen by a teacher T, i.e. $T(c^*) = w$, so the learner can reconstruct c^*. The common goal is to keep the *teaching dimension*, i.e., the cardinality of the witness set, $\max_{c \in C} |T(c)|$, as small as possible. In recent years, the field of machine teaching has seen various applications in fields like pedagogy (Shafto et al., 2014), trustworthy AI (Zhu et al., 2018), reinforcement learning (Zhang et al., 2021), active learning (Wang et al., 2021) and explainable AI (Yang et al., 2021).

Various models of machine teaching have been proposed, e.g. the classical teaching dimension model (Goldman & Kearns, 1995), the optimal teacher model (Balbach, 2008), recursive teaching (Zilles et al., 2011), preference-based teaching (Gao et al., 2017), no-clash teaching (Fallat et al., 2023), and probabilistic teaching (Ferri et al., 2022). In (Telle et al., 2019) a model focusing on teaching size is introduced, and in (Ferri et al., 2024) an algorithm called Greedy constructing the teacher mapping in this model is given.

Greedy assumes two total orderings $\prec_C$ on C and $\prec_X$ on X, with $\prec_X$ extended to $\prec_W$ on subsets of labelled examples $W = 2^{X \times \{0,1\}}$ by shortlex ordering. In the Greedy algorithm the teacher defines its mapping iteratively: go through W in the order of $\prec_W$, and for a given witness $w = \{(x_1,b_1)...(x_q,b_q)\}$, find the earliest (in $\prec_C$ order) $c \in C$ consistent with w (i.e. with $M(c,x_i) = b_i$ for all $1 \leq i \leq q$) such that $T(c)$ is not yet defined, then set $T(c) = w$ and continue with next witness (or drop this w if no such c exists).

To compare the teaching dimension achievable by Greedy to that of other models, the authors of (Ferri et al., 2024) argued as follows when a large witness is used: If Greedy assigns $T(c) = w$ for some $w = \{(x_1,b_1)...(x_q,b_q)\}$, then we may ask why was c not assigned to a smaller witness? Assuming there are $|X| = n$ examples, any subset $Q \subseteq X$ of size $q-1$ when labelled consistent with c has already been tried by Greedy, and hence some other concept must already have been assigned to any such Q, and all these concepts are distinct. This means we must have taught $\binom{n}{q-1} = k$ other concepts already. But then we have already taught at least $k+1$ concepts and we can again ask why were any of these not taught by a smaller witness of size $q-2$? It must be that any such witnesses (labelled to be consistent with some concept among the $k+1$ we already have) must have been used to teach other, again distinct, concepts.

Note that, to verify how many distinct witnesses exist, corresponding to new concepts, that are labelled consistently with one of these $k+1$ concepts, one must sum up the number of distinct rows when projecting on $q-2$ columns, for all choices of these columns. Note that the number of distinct rows, i.e., witnesses, and hence the number of con-

[1]Department of Informatics, University of Bergen, Bergen, Norway [2]Department of Applied Mathematics, Faculty of Mathematics and Physics, Charles University, Praha, Czech Republic. Correspondence to: Joakim Sunde <Joakim.Sunde@uib.no>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

cepts, when projecting on $q-2$ columns, for all choices of these columns, depends on the matrix M one does the projection on. The authors of (Ferri et al., 2024) wanted to find the matrix M minimizing the sum of unique rows after doing the projection, thus arriving at the following combinatorial question. What is the binary matrix M on k distinct rows and n columns that would give the smallest sum when projecting on q columns? They conjectured that this was achieved by the matrix $H_{n,k}$ consisting of the k rows corresponding to the binary representations of the numbers between zero and $k-1$, with leading 0s to give them length n. In this paper we prove this conjecture.

Consider the binary consistency graph G_C on the set of concepts versus the set W of subsets of labelled examples, with a concept c adjacent to $w \in W$ if c is consistent with each labelled example in w. We can view the Greedy Matching algorithm as working on G_C. Note that the above-mentioned sum for a matrix M when projecting on q columns (called $m_q(M)$ in the next section) is then the number of W-vertices on q examples that have at least one neighbor among the concepts. Since we prove that $H_{n,k}$ minimizes this value for all q, it means that it minimizes the number of W-vertices having a neighbor in the consistency graph, over all concept classes on k concepts over a domain of size n. As the consistency graph is of importance in machine teaching, this is an indication that our result has a general relevance in that field.

When $q = n-1$ this minimization question is equivalent to asking for the induced subgraph on k vertices of the hypercube of dimension n having the maximum number of edges, for the following reason. The rows of the k by n binary matrix M are viewed as k vertices of the hypercube of dimension n, labelled in the standard way, with two vertices adjacent iff their labels differ in exactly one coordinate. When $q = n-1$ we have $\binom{n}{n-1} = n$ choices for the projection on q columns and each such projection leaves out exactly one column (and a column corresponds to a dimension of the hypercube). Each such projection could give at most k unique rows, so the maximum achievable sum of unique projection rows is k times n. The main observation when $q = n-1$ is the following: three or more rows cannot have the same projection row, but two rows can, and two rows of M give the same projection row (when leaving out a column/dimension) if and only if the corresponding pair of vertices are adjacent (across the dimension we left out), and thus when $q = n-1$, the sum of unique projection rows for M is k times n minus the number of edges induced in the hypercube. Thus, a matrix minimizing the sum of unique projection rows for $q = n-1$ will also maximize the number of induced edges in the hypercube of dimension n.

The question of finding the matrix achieving the maximum mentioned above is called the *edge isoperimetry problem for the hypercube*. This has been shown (Hart, 1976) to be achieved by $H_{n,k}$, and the edge isoperimetry of the hypercube has been studied extensively in (McIlroy, 1974; Delange, 1975; Hart, 1976; Greene & Knuth, 1990; Agnarsson, 2013) to name a few articles. The result we give in this paper is thus a generalization of the edge isoperimetry problem on the hypercube, as we show that $H_{n,k}$ is the solution not only when $q = n-1$, but for all values of $1 \leq q \leq n$.

The rest of our paper is organized as follows. In Section 2 we give the formal definition of the conjecture. In Section 3 we show that the conjecture would be settled if we could prove a stronger theorem. Then in Section 4 we prove this stronger theorem, using a generalization of an old result from (Graham, 1970).

2. Statement of the main theorem

Let M be a $k \times n$ binary matrix whose all k rows are distinct. Let $\mathcal{M}_{n,k}$ be the set of all such matrices. For any binary matrix A, let $\mathrm{dif}(A)$ denote the number of unique rows in the matrix A. For $Q \subseteq \{1, 2, ..., n\}$, let $M(Q)$ be the submatrix of $M \in \mathcal{M}_{n,k}$ formed by taking the columns with indices from Q. Finally for integers a and b where $a \leq b$ let $[a, b] = \{a, a+1, ..., b\}$. Our main interest is the number

$$m_q(M) = \sum_{Q \in \binom{[1,n]}{q}} \mathrm{dif}(M(Q))$$

which is the sum of the numbers of unique rows for each submatrix of M created by picking a subset of the columns of size q. For fixed positive integers k, n and q, we are interested in finding a matrix $M \in \mathcal{M}_{k,n}$ with the minimum value of $m_q(M)$. Let $m_q(n, k)$ be this minimum value, i.e.,

$$m_q(n, k) = \min_{M \in \mathcal{M}_{n,k}} m_q(M).$$

We show that the $k \times n$ binary matrix $H_{n,k}$ whose rows are the binary representations of all numbers between zero and $k-1$ achieves this minimum value of $m_q(n, k)$.

Example 1.

$$H_{4,5} = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \end{pmatrix} \cong \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

with $m_1(H_{4,5}) = 7$, $m_2(H_{4,5}) = 16$, $m_3(H_{4,5}) = 15$ and $m_4(H_{4,5}) = 5$.

It will be useful for us to use the following recursive definition of $H_{n,k}$. Note that the invariants we are interested in

remain unchanged by permutations of rows or columns of the matrices under consideration. In this sense we consider all such matrices equivalent. Let $\mathbf{0}$ be the all 0 row vector and let $\mathbf{0}^T$ be the all 0 column vector, and similarly for $\mathbf{1}$ and $\mathbf{1}^T$. Then

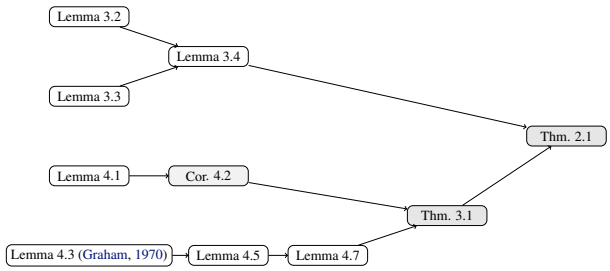
$$H_{n,k} = \begin{cases} \mathbf{0} & k = 1 \\ \begin{pmatrix} H_{n-1,\lceil \frac{k}{2} \rceil} & \mathbf{0}^T \\ H_{n-1,\lfloor \frac{k}{2} \rfloor} & \mathbf{1}^T \end{pmatrix} & k > 1. \end{cases}$$

Let $h_q(n,k) = m_q(H_{n,k})$. Our goal is thus to prove the following theorem.

Theorem 2.1. *For any positive integers q, n, k where $q \leq n$ and $k \leq 2^n$,*

$$m_q(n,k) = h_q(n,k).$$

Here is a diagram showing how we will prove Theorem 2.1.



3. A sufficient condition

The goal of this section is to prove that the following theorem (whose proof we leave to the next section) implies Theorem 2.1.

Theorem 3.1. *For any positive integers q, n, k where $q \leq n$ and $k \leq 2^n$,*

$$\min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} h_q(n,x) + h_{q-1}(n-1, k-x)$$

$$= h_q(n, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor).$$

which is just stating that the minimum value of the expression on the left occurs when $x = \lceil \frac{k}{2} \rceil$.

Lemma 3.2. *The h numbers satisfy the recurrence relation*

$$h_q(n,1) = \binom{n}{q}$$

and

$$h_q(n,k) = h_q(n, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor)$$

for $k > 1$.

Proof. Let Q be a q-element subset of the column-index set $\{1, 2, \ldots, n\}$. If $n \notin Q$, then each of the bottom $\lfloor \frac{k}{2} \rfloor$ rows of $H_{n,k}(Q)$ appears as a row among the $\lceil \frac{k}{2} \rceil$ top ones, and hence $m_q(H_{n,k}(\{1, 2, \ldots, n-1\})) = m_q(H_{n-1,\lceil \frac{k}{2} \rceil}) = h_q(n-1, \lceil \frac{k}{2} \rceil)$. Since the value of the last column (the n-th column) is 0 for the $\lfloor \frac{k}{2} \rfloor$ rows and 1 for the rest we have that if $n \in Q$, every row from the bottom $\lfloor \frac{k}{2} \rfloor$ rows of $H_{n,k}(Q)$ differs from any row from the $\lceil \frac{k}{2} \rceil$ top ones, and so the sum over those Q that contain n contributes exactly $m_{q-1}(H_{n-1,\lceil \frac{k}{2} \rceil}) + m_{q-1}(H_{n-1,\lfloor \frac{k}{2} \rfloor}) = h_{q-1}(n-1, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor)$. Thus

$$h_q(n,k)$$

$$= h_q(n-1, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor).$$

This can be slightly simplified as follows. Note that $h_q(n-1, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lceil \frac{k}{2} \rceil)$ is exactly the contribution of the $\lceil \frac{k}{2} \rceil$ top rows of $H_{n,k}$ to $m_q(H_{n,k})$, i.e., $m_q((H_{n-1,\lceil \frac{k}{2} \rceil} \mathbf{0}^T))$ what equals $m_q((\mathbf{0}^T H_{n-1,\lceil \frac{k}{2} \rceil})) = m_q(H_{n,\lceil \frac{k}{2} \rceil}) = h_q(n, \lceil \frac{k}{2} \rceil)$ and the claim follows. □

Lemma 3.3. *For any positive integers q, n, k where $q \leq n$ and $k \leq 2^n$,*

$$m_q(n,k) \geq \min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} m_q(n,x) + m_{q-1}(n-1, k-x)$$

Proof. Let $A \in \mathcal{M}_{n,k}$ be a matrix that minimizes m_q over $\mathcal{M}_{n,k}$, i.e., it satisfies $m_q(A) = m_q(n,k)$.

If $k = 1$, then every $Q \in \binom{[1,n]}{q}$ contributes 1 to the sum $\sum_Q \mathrm{dif}(M(Q))$, and hence $m_q(A) = \binom{n}{q}$.

Let $k > 1$. Suppose w.l.o.g. that the last column contains both 0's and 1's and that the number of 1's does not exceed the number of 0's. Let y be the number of 0's in it, and assume that the 0's are in rows $1, \ldots, y$ and the 1's in rows $y+1, \ldots, k$, with $y \geq k - y$, i.e., $y \geq \lceil \frac{k}{2} \rceil$. Let T be the submatrix of A determined by rows $1, \ldots, y$ and columns $1 \ldots, n-1$, and let B be the submatrix determined by rows $y+1, \ldots, k$ and columns $1, \ldots, n-1$, i.e.,

$$A = \begin{pmatrix} T & \mathbf{0}^T \\ B & \mathbf{1}^T \end{pmatrix}.$$

We further denote by $T^* = (T \ \mathbf{0}^T)$ the submatrix of A formed by its top y rows.

We first observe that

$$\sum_{Q \in \binom{[1,n]}{q} : n \notin Q} \mathrm{dif}(A(Q)) \geq \sum_{Q \in \binom{[1,n-1]}{q}} \mathrm{dif}(T(Q)) \quad (1)$$

since in this case we do not include the n-th column in Q. Because the n-th column is not included, we observe that any unique row projection counted by $\sum_{Q\in\binom{[1,n-1]}{q}} \text{dif}(T(Q))$ will be a subset of the unique row projections counted by $\sum_{Q\in\binom{n}{q}:n\notin Q} \text{dif}(A(Q))$.

We also see that

$$\sum_{Q\in\binom{[1,n]}{q}:n\in Q} \text{dif}(A(Q)) \geq \sum_{Q'\in\binom{[1,n-1]}{q-1}} \text{dif}(T(Q')) + \sum_{Q'\in\binom{[1,n-1]}{q-1}} \text{dif}(B(Q')) \tag{2}$$

since when $n \in Q$, each row leading to a unique projection in A, the entire row, except that last column, was in T or in B. As we know that each projection of rows in B will differ from rows in T in at least the n-th column, we can count up the number of unique projections in T and B separately, using Q' of size $(q-1)$ as we will increase the size by 1 when we add back n.

We see that $m_q(T^*) \geq m_{q-1}(n-1, k-y)$ since for a given T^* for n columns and y rows, the m_q value will be greater or equal to the minimum value over all matrices of size (n, y). We also see that $m_{q-1}(B) \geq m_{q-1}(n-1, k-y)$ using the same idea. Given a matrix B of size $(n-1, k-y)$, the m_q value of this matrix will be larger or equal to the minimum value over all matrices of size $(n-1, k-y)$. Combining these we get the inequality

$$m_q(T^*) + m_{q-1}(B) \geq m_{q-1}(n-1, k-y) + m_{q-1}(n-1, k-y) \tag{3}$$

.

Finally we need the inequality

$$m_q(n, y) + m_{q-1}(n-1, k-y) \geq \min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} m_q(n, x) + m_{q-1}(n-1, k-x). \tag{4}$$

To show the soundness of this inequality we observe that when $x = \lceil \frac{k}{2} \rceil$ we have $x \leq y$, as $x = \lceil \frac{k}{2} \rceil \leq y$. We also have $y \leq k-1$, as we assume that the last column has both 0's and 1's. When $x = k-1$, we have $y \leq x$, as we do the minimization over all possible values of x in this range, we know that we are evaluating $x = y$ as well. Hence the minimum will be equal to or less than $m_q(n, y) + m_{q-1}(n-1, k-y)$.

Using these four relations we can now finish the proof.

$$m_q(A) = \sum_{Q\in\binom{[1,n]}{q}:n\notin Q} \text{dif}(A(Q)) + \sum_{Q\in\binom{[1,n]}{q}:n\in Q} \text{dif}(A(Q)) \geq$$

(by combining (1) and (2))

$$\geq \sum_{Q\in\binom{[1,n-1]}{q}} \text{dif}(T(Q)) + \sum_{Q'\in\binom{[1,n-1]}{q-1}} \text{dif}(T(Q')) + \sum_{Q'\in\binom{[1,n-1]}{q-1}} \text{dif}(B(Q')) =$$

$$= \sum_{Q\in\binom{[1,n-1]}{q}} \text{dif}(T(Q)) + \sum_{Q\in\binom{[1,n]}{q},n\in Q} \text{dif}(T^*(Q)) + \sum_{Q'\in\binom{[1,n-1]}{q-1}} \text{dif}(B(Q')) =$$

$$= m_q(T^*) + m_{q-1}(B) \geq$$

(by (3))

$$\geq m_q(n, y) + m_{q-1}(n-1, k-y) \geq$$

(and by (4))

$$\geq \min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} m_q(n, x) + m_{q-1}(n-1, k-x).$$

□

Lemma 3.4. *Theorem 3.1 implies Theorem 2.1*

Proof. Certainly $m_q(n, k) \leq h_q(n, k)$, we prove the other inequality by induction on k. The base case $k = 1$ follows from $m_q(n, 1) = h_q(n, 1) = \binom{n}{q}$.

Suppose $k > 1$. Lemmas 3.2 and 3.3 imply that

$$m_q(n, k) \geq \min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} m_q(n, x) + m_{q-1}(n-1, k-x) \geq$$

(by the induction hypothesis)

$$\geq \min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} h_q(n, x) + h_{q-1}(n-1, k-x) =$$

(by Theorem 3.1)

$$= h_q(n, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor) =$$

(by Lemma 3.2)

$$= h_q(n, k).$$

□

4. Proving Theorem 3.1

In this section we will prove Theorem 3.1 by showing that $h_q(n,x)$ "increases" at least as fast as $h_{q-1}(n-1,k-x)$ "decreases" when x starts at $\lceil \frac{k}{2} \rceil$ and increases until $k-1$. To be more precise, we will show that

$$\begin{aligned} &h_q(n, \lceil \frac{k}{2} \rceil + j) - h_q(n, \lceil \frac{k}{2} \rceil) \\ &\geq h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor) - h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor - j) \end{aligned} \tag{5}$$

for any $j \geq 1$ such that $\lceil \frac{k}{2} \rceil + j \leq k-1$. Since the above inequality is equivalent to

$$\begin{aligned} &h_q(n, \lceil \frac{k}{2} \rceil + j) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor - j) \\ &\geq h_q(n, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor), \end{aligned}$$

it follows straightforwardly that the minimum value of $h_q(n,x) + h_{q-1}(n-1,k-x)$ over $x \in [\lceil \frac{k}{2} \rceil, k-1]$ is attained by $x = \lceil \frac{k}{2} \rceil$.

We first need to understand the behavior of the $h_q(n,k)$ numbers as k increases or decreases. Let $|x|$ denote the Hamming weight (number of 1's) in the binary representation of integer x. We recall that the binomial coefficient $\binom{n}{k}$ by definition evaluates to 0 when $k < 0$ or $k > n$. Similarly, we define the boundary values of $h_q(n,k)$ for $q = 0$ and $k = 0$ as $h_0(n,k) = 1$ (for $k > 0$) and $h_q(n,0) = 0$.

Lemma 4.1. *For any integers x, q, n such that $0 \leq x \leq 2^n - 1$ and $0 \leq q \leq n$, we have*

$$h_q(n, x+1) = h_q(n,x) + \binom{n-|x|}{q-|x|},$$

and for integers x, q, n such that $1 \leq x \leq 2^{n-1}$ and $1 \leq q \leq n$, we have

$$h_{q-1}(n-1, x-1) = h_{q-1}(n-1,x) - \binom{n-1-|x-1|}{q-1-|x-1|}.$$

Proof. We prove the first formula, the second one then follows directly by applying the first one for $x-1, q-1$ and $n-1$.
For the boundary values of q and x, we have $h_0(n,1) = 1 = 0+1 = h_0(n,0) + \binom{n}{0}$, $h_0(n,x+1) = 1 = 1+0 = h_0(n,x) + \binom{n-|x|}{-|x|}$ for $x \geq 1$, and $h_q(n,1) = \binom{n}{q} = 0 + \binom{n}{q} = h_q(n,0) + \binom{n-|0|}{q-|0|}$.
For the notrivial cases, suppose that $q \geq 1$ and $x \geq 1$. The only difference between $H_{n,x}$ and $H_{n,x+1}$ is that $H_{n,x+1}$ has one extra row, which is the binary representation of x with zeroes padded to the left if needed. Let S be the set of column indices where the last row of $H_{n,x+1}$ has a 1.
We first observe that $\text{dif}(H_{n,x}(Q)) = \text{dif}(H_{n,x+1}(Q))$ whenever $S \not\subseteq Q \subseteq \{1,2,\ldots,n\}$. To see this let $i \in S \backslash Q$ and y be the number with binary representation having the same entry as x in the positions belonging to Q and 0's in all other positions. Then $y < x$ and $H_{n,x}$ contains a row which is the binary representation of y. Since this row of $H_{n,x}$ is equal to the last row of $H_{n,x+1}$ when only looking at the columns with indices in Q, $\text{dif}(H_{n,x}(Q)) = \text{dif}(H_{n,x+1}(Q))$.
Then we see that $\text{dif}(H_{n,x+1}(Q)) = \text{dif}(H_{n,x}(Q)) + 1$ whenever $S \subseteq Q$. This is because there is no row in $H_{n,x}$ where all the columns with indices in S are equal to 1, since the number of this row would be greater or equal to x.
So we are left with counting how many subsets Q of $\{1,...,n\}$ satisfy $S \subseteq Q$ and $|Q| = q$. This is exactly $\binom{n-|S|}{q-|S|} = \binom{n-|x|}{q-|x|}$.

□

Corollary 4.2. *For any integers q, n, x, j such that $0 \leq q \leq n$, $0 \leq x$, $1 \leq j$ and $x + j \leq 2^n$, we have*

$$h_q(n, x+j) = h_q(n,x) + \sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|}.$$

Moreover, whenever $1 \leq q \leq n$ and $1 \leq j \leq x \leq 2^{n-1}$, we have

$$h_{q-1}(n-1, x-j) = h_{q-1}(n-1,x) - \sum_{i=x-j}^{x-1} \binom{n-1-|i|}{q-1-|i|},$$

and whenever $1 \leq q \leq n$ and $1 \leq j \leq x-1 \leq 2^{n-1}$, we have

$$\begin{aligned} &h_{q-1}(n-1, x-j-1) \\ &= h_{q-1}(n-1, x-1) - \sum_{i=x-j-1}^{x-2} \binom{n-1-|i|}{q-1-|i|}. \end{aligned}$$

Proof. The first two formulae follow from Lemma 4.1 by induction on j, the third formula follows from the second by substituting $x-1$ for x. □

In view of this corollary, the inequality (5) is equivalent to the claim that our goal is to prove that

$$\sum_{i=\lceil \frac{k}{2} \rceil}^{\lceil \frac{k}{2} \rceil + j - 1} \binom{n-|i|}{q-|i|} \geq \sum_{i=\lfloor \frac{k}{2} \rfloor}^{\lfloor \frac{k}{2} \rfloor - 1} \binom{n-1-|i|}{q-1-|i|}$$

holds true for all feasible q, n, k and j.

We first show some useful properties of Hamming weights which extend the following lemma from (Graham, 1970) whose proof was finalized in (Jones & Torrence, 1999).

Lemma 4.3. *((Graham, 1970; Jones & Torrence, 1999)) Let s, t be non-negative integers. Then there exists a bijective mapping $\theta : [0, r] \to [s, s + r]$ such that $|\theta(k)| \geq |k|$ for every $k \in [0, r]$.*

We will need a generalization of this lemma whose proof depends on the following observation:

Observation 4.4. Let $x \geq t$ be non-negative integers. Then $|x - t| \geq |x| - |t|$.

Proof. This follows directly from the standard subtraction algorithm for integers in binary representation. □

Lemma 4.5. *Let s, r, t be non-negative integers such that $r, t \geq 1$ and $s \geq r + t - 1$. Denote by $T = [s, s + r - 1]$ and $B = [s - r - t + 1, s - t]$. Then there exists a bijective mapping $\theta : T \to B$ such that $|\theta(x)| \geq |x| - |t|$ for all $x \in T$.*

Proof. Our proof works by induction on r. When $r = 1$, we have $T = \{s\}$ and $B = \{s - t\}$. The only possible mapping θ then simply maps s to $s - t$ and we see that $|\theta(s)| = |s - t| \geq |s| - |t|$ by Observation 4.4. Thus, the base case $r = 1$ is established for all values of $t \geq 1$.

Let $r > 1$. Create two matrices with r rows each

$$M_T = \begin{pmatrix} \overrightarrow{s + r - 1} \\ \vdots \\ \overrightarrow{s + 1} \\ \overrightarrow{s} \end{pmatrix} \text{ and } M_B = \begin{pmatrix} \overrightarrow{s - t} \\ \overrightarrow{s - t - 1} \\ \vdots \\ \overrightarrow{s - r - t + 1} \end{pmatrix}$$

where $\overrightarrow{x}$ is the base 2 representation of x as a binary vector with 0-s padded to the left so that all vectors have the same length. Finally let $M = \begin{pmatrix} M_T \\ M_B \end{pmatrix}$.

Reformulating the lemma in this matrix context we seek a bijective mapping θ of the rows of M_T to the rows of M_B such that $|\theta(x)| \geq |x| - |t|$ holds true for every row x of M_T. (With a slight abuse of notation we write $\theta : M_T \to M_B$.) The induction hypothesis states that this holds true, for this value of t, if the number of rows of each matrix is less than r.

Without loss of generality we may assume that the first (leftmost) column of M contains at least one 0 and at least one 1 (since we could disregard this column otherwise). Then if we look at the first column of M, there will be a point where a 1 appears for the first time, when moving through the rows from the bottom row up. This could happen either in the M_T part or in the M_B part of the matrix. We will deal with these 2 cases separately.

Case 1 (The first leftmost 1 appears in the M_T part of the matrix) We divide both the M_T and M_B matrices further and write M as

$$M = \begin{pmatrix} T_1 \\ T_2 \\ B_2 \\ B_1 \end{pmatrix}$$

where the bottom row of T_1 is the row where the first 1 appears (thus that row is $100...0$), with T_2 being the remainder of M_T, and we let B_1 have the same number of rows as T_1. We will map T_1 to B_1 and T_2 to B_2. Since T_2 and B_2 have fewer rows than r (since T_1 and B_1 always have at least one row) and are on the form specified by the lemma since the smallest number in T_2 are the same as in T and the largest number in B_2 is the same as in B and we simply deleted some of the largest/smallest numbers of T and B to create T_2 and B_2 respectively so it will still be an interval. It follows by the induction hypothesis applied to T_2, B_2 and r as the number of rows of T_2, B_2 that, for the same value of t, there exists the required mapping $\theta_1 : T_2 \to B_2$. Note also that this is vacuously true if T_2 and B_2 are empty. Now if we ignore the first column of T_1, then $T_1(\{2, 3, ...\})$ is the binary representation of the numbers $0, 1, ..., |T_1| - 1$. So by Lemma 4.3 there is a mapping $\theta_2 : T_1(\{2, 3, ...\}) \to B_1$ such that $|\theta_2(x)| \geq |x|$ for every x. Adding back the first column of T_1 and using the same mapping between the rows as θ_2, we get a mapping $\theta_3 : T_1 \to B_1$ where $|\theta_3(x)| \geq |x| - 1$ for every x (since the Hamming weight of x increases by 1). Clearly $|x| - 1 \geq |x| - |t|$ when $t \geq 1$, hence combining θ_1 and θ_3 gives us a bijective mapping $\theta : T \to B$ with the required properties.

Case 2 (The first leftmost 1 appears in the M_B part) We divide the matrix in a similar way as in the first case

$$M = \begin{pmatrix} T_1 \\ T_2 \\ B_2 \\ B_1 \end{pmatrix}$$

so that the bottom row of B_2 is the row where the first 1 appears in the leftmost column (so this row is $100...0$) and we let T_2 have the same number of rows as B_2. For a binary vector x let $\overline{x}$ be the complement of x, so $\overline{x} = (1, 1, 1, ..., 1) - x$. For a binary matrix A, let $\overline{A}$ be the matrix whose rows are the complements of the rows of A.

By the induction hypothesis, as there are fewer rows and we have the same value of t, there is a mapping $\theta_1 : T_2 \to B_2$ with the required properties. The top row of B_1 is $(011...1)$ so if we look at $\overline{B_1}$, the top row will be $(100...0)$, the next row will be $(100...01)$ and so on, meaning that if we ignore the first column we are counting up from 0 in binary. Lemma 4.3 then gives us a mapping $\theta_2 : \overline{B_1(\{2, 3, ...\})} \to \overline{T_1}$ with $|\theta_2(x)| \geq |x|$ for every x. Adding back the first column but keeping the row mapping we get a mapping $\theta_3 : \overline{B_1} \to \overline{T_1}$

where $|\theta_3(x)| \geq |x| - 1$. Now define $\theta_4 : B_1 \to T_1$ by $\theta_4(x) = \overline{\theta_3(\overline{x})}$ and let $||x||$ be the length of the vector x.

$$\begin{aligned} |\theta_4(x)| - |x| &= |\overline{\theta_3(\overline{x})}| - |x| \\ &= ||x|| - |\theta_3(\overline{x})| - (||x|| - |\overline{x}|) \\ &= |\overline{x}| - |\theta_3(\overline{x})| \qquad (6) \\ &\leq 1 \end{aligned}$$

To get (6) we use the fact that $|\theta_3(x)| \geq |x| - 1$.

We will now look at the inverse mapping $\theta_4^{-1} : T_1 \to B_1$. For any $y \in T_1$, there is an $x \in B_1$ such that $\theta_4(x) = y$. We just showed that $|\theta_4(x)| - |x| \leq 1$ which is the same as $|y| - |x| \leq 1$ which we can rewrite as $|y| - |\theta_4^{-1}(y)| \leq 1$. Multiplying both sides by -1 we get $|\theta_4^{-1}(y)| \geq |y| - 1$. Combining θ_4^{-1} and θ_1 in the natural way we get the desired mapping $\theta : T \to B$ satisfying $|\theta(x)| \geq |x| - 1 \geq |x| - |t|$ for every x.

Thus the lemma is proven for any $r, t \geq 1$ and any $s \geq r + t - 1$. □

We are now set to prove that the sum in the first formula of the Corollary 4.2 is always larger than the sums in the second and third formulae. We will need the following well known observation:

Observation 4.6. For any integers n, k, j such that $0 \leq j \leq k \leq n$,

$$\binom{n}{k} \geq \binom{n-j}{k-j}.$$

Lemma 4.7. *For all positive integers q, n, x, j such that $x + j - 1 \leq 2^n$ and $x - j \geq 0$,*

$$\sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|} \geq \sum_{i=x-j}^{x-1} \binom{n-1-|i|}{q-1-|i|},$$

and if $x - j \geq 1$, then

$$\sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|} \geq \sum_{i=x-j-1}^{x-2} \binom{n-1-|i|}{q-1-|i|}.$$

Proof. We show that there exists a bijection θ from $[x, x + j - 1]$ to $[x - j, x - 1]$ (or to $[x - j - 1, x - 2]$, respectively) such that $|\theta(i)| \geq |i| - 1$ for all i. This will prove the lemma since then for every term $\binom{n-|i|}{q-|i|}$ in the sum of the left hand side there will be a corresponding term in the sum on the right side

$$\binom{n-1-|\theta(i)|}{q-1-|\theta(i)|}$$

and we see that by Observation 4.6

$$\binom{n-1-|\theta(i)|}{q-1-|\theta(i)|} \leq \binom{n-1-(|i|-1)}{q-1-(|i|-1)} = \binom{n-|i|}{q-|i|}.$$

So if such a bijection exists, then for every term in the sum on the left hand side there will be a unique element in the sum on the right hand side which is no greater than the element on the left hand side. So then the sum on the left hand side must be greater than or equal to the sum on the right hand one. Now we just need to show that there exist such bijections θ. We will use Lemma 4.5.

Case 1. Let $s = x, r = j$ and $t = 1$. Then by Lemma 4.5 there is a mapping $\theta : [x, x + j - 1] \to [x - 1, x - j]\}$ such that $|\theta(i)| \geq |i| - |t| = |i| - 1$ for every i, which is the first bijection we wanted.

Case 2. If we set $t = 2$, then Lemma 4.5 gives us a mapping $\theta : [x, x + j - 1] \to [x - 2, x - j - 1]$ such that $|\theta(i)| \geq |i| - |t| = |i| - 1$ for every i, which is the second bijection we wanted. □

We are now ready to prove Theorem 3.1.

Theorem 3.1. *For any positive integers q, n, k where $q \leq n$ and $k \leq 2^n$,*

$$\min_{\lceil \frac{k}{2} \rceil \leq x \leq k-1} h_q(n, x) + h_{q-1}(n-1, k-x)$$

$$= h_q(n, \lceil \frac{k}{2} \rceil) + h_{q-1}(n-1, \lfloor \frac{k}{2} \rfloor).$$

Proof. We consider the two cases depending on whether k is either even or odd.

Case 1 - k is even. Set $x = \frac{k}{2}$ and rewrite the left hand side of Theorem 3.1 as

$$\min_{0 \leq j \leq x-1} h_q(n, x + j) + h_{q-1}(n-1, x-j)$$

Then by the first and second part of Corollary 4.2 whenever $1 \leq j \leq x - 1$, we can rewrite the expression which is minimized as

$$h_q(n, x) + \sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|} + h_{q-1}(n-1, x)$$

$$- \sum_{i=x-j}^{x-1} \binom{n-1-|i|}{q-1-|i|}$$

By Lemma 4.7 we have $\sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|} \geq \sum_{i=k-j}^{x-1} \binom{n-1-|i|}{q-1-|i|}$ for any $1 \leq j \leq x - 1$, which means that the smallest value occurs when $j = 0$.

Case 2 - k is odd. Set $x = \lceil \frac{k}{2} \rceil$ and rewrite the expression of the left hand side of Theorem 3.1 as

$$\min_{0 \leq j \leq x-1} h_q(n, x + j) + h_{q-1}(n-1, x-1-j).$$

By the first and third part of Corollary 4.2 we can rewrite the part which is minimized for $1 \leq j \leq x - 1$ as

$$h_q(n, x) + \sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|} + h_{q-1}(n-1, x-1)$$

$$- \sum_{i=x-j-1}^{x-2} \binom{n-1-|i|}{q-1-|i|}.$$

By Lemma 4.7 we have $\sum_{i=x}^{x+j-1} \binom{n-|i|}{q-|i|} \geq \sum_{i=k-j-1}^{x-2} \binom{n-1-|i|}{q-1-|i|}$ for $1 \leq j \leq x-1$ so the smallest value of the expression will occur for $j = 0$.

□

By Lemma 3.4 this proves Theorem 2.1.

5. Conclusion

We have proven a conjecture of (Ferri et al., 2024), identifying a binary matrix M minimizing $m_q(M)$, the sum of the numbers of distinct rows over all submatrices on q columns. Let us further consider the complexity of computing $m_q(M)$ when the binary matrix M with k rows and n columns is given as input. There is a straightforward algorithm with runtime $O(n^q kq \log k)$. The question arises if computing $m_q(M)$ is FPT (Fixed Parameter Tractable, see Cygan et al. (2015)) when parameterized by q. In other words, is there an algorithm whose runtime is polynomial in the size of M, with any superpolynomial dependency restricted to q only? We leave this as an open problem.

Acknowledgements

Thanks to the reviewers for their careful reading and helpful suggestions.

Impact Statement

Advancing the field of machine teaching.

References

Agnarsson, G. On the number of hypercubic bipartitions of an integer. *Discrete Mathematics*, 313(24):2857–2864, 2013.

Balbach, F. J. Measuring teachability using variants of the teaching dimension. *Theoretical Computer Science*, 397 (1-3):94–113, 2008.

Cygan, M., Fomin, F. V., Kowalik, L., Lokshtanov, D., Marx, D., Pilipczuk, M., Pilipczuk, M., and Saurabh, S. *Parameterized Algorithms*. Springer, 2015. ISBN 978-3-319-21274-6. doi: 10.1007/978-3-319-21275-3. URL https://doi.org/10.1007/978-3-319-21275-3.

Delange, H. Sur la fonction sommatoire de la fonction" somme des chiffres". *Enseign. Math.*, 21(2):31–47, 1975.

Fallat, S., Kirkpatrick, D., Simon, H. U., Soltani, A., and Zilles, S. On batch teaching without collusion. *Journal of Machine Learning Research*, 24:1–33, 2023.

Ferri, C., Hernández-Orallo, J., and Telle, J. A. Non-cheating teaching revisited: A new probabilistic machine teaching model. In Raedt, L. D. (ed.), *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI 2022, Vienna, Austria, 23-29 July 2022*, pp. 2973–2979. ijcai.org, 2022. doi: 10.24963/IJCAI.2022/412. URL https://doi.org/10.24963/ijcai.2022/412.

Ferri, C., Garigliotti, D., Håvardstun, B. A. T., Hernández-Orallo, J., and Telle, J. A. When redundancy matters: Machine teaching of representations, 2024. arXiv:2401.12711.

Gao, Z., Ries, C., Simon, H. U., and Zilles, S. Preference-based teaching. *Journal of Machine Learning Research*, 18:31:1–31:32, 2017. URL http://jmlr.org/papers/v18/16-460.html.

Goldman, S. A. and Kearns, M. J. On the complexity of teaching. *Journal of Computer and System Sciences*, 50 (1):20–31, 1995.

Graham, R. L. On primitive graphs and optimal vertex assignments. *Annals of the New York academy of sciences*, 175(1):170–186, 1970.

Greene, D. H. and Knuth, D. E. *Mathematics for the Analysis of Algorithms*, volume 504. Springer, 1990.

Hart, S. A note on the edges of the n-cube. *Discrete Mathematics*, 14(2):157–163, 1976.

Jones, J. C. and Torrence, B. F. The case of the missing case: the completion of a proof by rl graham. *Pi Mu Epsilon Journal*, 10(10):772–778, 1999.

McIlroy, M. D. The number of 1's in binary integers: bounds and extremal properties. *SIAM Journal on Computing*, 3 (4):255–261, 1974.

Shafto, P., Goodman, N. D., and Griffiths, T. L. A rational account of pedagogical reasoning: Teaching by, and learning from, examples. *Cognitive Psychology*, 71:55 – 89, 2014. ISSN 0010-0285. doi: https://doi.org/10.1016/j.cogpsych.2013.12.004. URL http://www.sciencedirect.com/science/article/pii/S0010028514000024.

Telle, J. A., Hernández-Orallo, J., and Ferri, C. The teaching size: computable teachers and learners for universal languages. *Machine Learning*, 108(8-9):1653–1675, 2019.

Valiant, L. G. A theory of the learnable. *Commun. ACM*, 27 (11):1134–1142, 1984. doi: 10.1145/1968.1972. URL https://doi.org/10.1145/1968.1972.

Wang, C., Singla, A., and Chen, Y. Teaching an active learner with contrastive examples. In Ranzato, M., Beygelzimer, A., Dauphin, Y. N., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 17968–17980, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/958adb57686c2fdec5796398de5f317a-Abstract.html.

Yang, S. C.-H., Vong, W. K., Sojitra, R. B., Folke, T., and Shafto, P. Mitigating belief projection in explainable artificial intelligence via bayesian teaching. *Scientific reports*, 11(1):1–17, 2021.

Zhang, X., Bharti, S. K., Ma, Y., Singla, A., and Zhu, X. The sample complexity of teaching by reinforcement on q-learning. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pp. 10939–10947. AAAI Press, 2021. doi: 10.1609/AAAI.V35I12.17306. URL https://doi.org/10.1609/aaai.v35i12.17306.

Zhu, X., Singla, A., Zilles, S., and Rafferty, A. N. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

Zilles, S., Lange, S., Holte, R., and Zinkevich, M. Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12(Feb):349–384, 2011.

Paper IV

Håvardstun, B., Ferri, C., Flikka, K., & Telle, J. A.
Proceedings of the World Conference on Explainable Artificial Intelligence (xAI 2024),
CCIS **2155**, pp. 439–453 (2024)
DOI: 10.1007/978-3-031-63800-8_22

XAI for Time Series Classification: Evaluating the Benefits of Model Inspection for End-Users

Brigt Håvardstun[1( )], Cèsar Ferri[2], Kristian Flikka[3], and Jan Arne Telle[1]

[1] Department of Informatics, University of Bergen, Bergen, Norway
{brigt.havardstun,Jan.Arne.Telle}@uib.no
[2] VRAIN, UPV-Polytechnic University of Valencia, Valencia, Spain
cferri@dsic.upv.es
[3] Eviny AS, Bergen, Norway
Kristian.Flikka@eviny.no

Abstract. We present an XAI tool for time series classification providing model-agnostic instance-based post-hoc explanations, by means of prototypes and counterfactuals. Additionally, our tool allows for model inspection on instances generated by the user, to navigate the boundary between classes. This will allow the user to test and improve their hypotheses when formulating a model of the black box classification. We perform a human-grounded evaluation with forward simulation, to contribute a quantitative end-user evaluation to the field of XAI for time series.

Keywords: Time series · Human-grounded evaluation · Instance-based explanations

1 Introduction

Temporal data is encountered in many real-world applications ranging from patient data in healthcare [15] to the field of cyber security [20]. Deep learning methods have been successful in time series classification [11,15,20], but such methods are not easily interpretable, and often viewed as black boxes, which limits their applications when user trust in the decision process is crucial. To enable the analysis of these black-box models we revert to post-hoc interpretability. Recent research has focused on adapting existing methods to time series, both specific methods like SHAP-LIME [8], Saliency Methods [10] and Counterfactuals [4], and also combinations of these [16].

However, compared to images and text, time series data are not intuitively understandable to humans [18]. This makes interpretability of time series extra demanding, both when it comes to understanding how users will react to the provided explanations and also to the evaluation of its usefulness. Nevertheless,

L. Longo et al. (Eds.): xAI 2024, CCIS 2155, pp. 439–453, 2024.
https://doi.org/10.1007/978-3-031-63800-8_22

as humans learn and reason by forming mental representations of concepts based on examples, and any machine learning model has been trained on data, then data e.g. in the form of prototypes and counterfactuals is indeed the natural common language between the user and this model. In addition, several studies have highlighted the need to rethink new ways of interaction with an XAI algorithm, to allow for a dialogue between explainer and explainee, and to enable model inspection at will [1,13,17].

Hence, we advance an XAI time series tool that on the one hand provides users with instances of both prototype and counterfactual time series, and on the other hand lets the user generate their own instances for classification. This enables active learning and allows the user to test and improve their hypotheses when formulating a mental model of the black box classification.

We perform a quantitative user evaluation, thereby meeting a demand from the XAI research community [4,21,22], to measure how prototypes, counterfactuals and interactivity increases the understanding that the user has of the black box classification. Using the taxonomy of interpretability evaluation [5] what we do is a Human-Grounded Evaluation with Forward Simulation.

We use 3 datasets having a binary classification (e.g. 24-h power demand of a household in Winter versus Summer) and for each dataset we train an ML model. Note that it is generally more difficult to explain the classification of an AI model than the classification of a real-world dataset. The reason for this is that the dataset contains only real instances, whereas the AI model classifies *any* instance, also artificial ones. In this work, for prototypes we use real instances from the dataset, whereas we produce counterfactuals by combining real instances with artificial ones, based on the NativeGuide method [4]. For tests we have chosen to use real instances.

In the remainder of this paper, we first discuss related work, then we give definitions, followed by a description of the tool and a discussion of the user evaluation results.

2 Related Work

Research on XAI for time series classification has progressed similarly to XAI in general, with earlier work focused on feature-importance rather than on instance-based methods using prototypes and counterfactuals. A comprehensive survey of XAI methods for time series can be found in [21]. Our own work deals with model-agnostic instance-based post-hoc explanations, by means of prototypes and counterfactuals. In [14], the author offers a survey of work on instance-based explanations for XAI mainly in the image domain, defining and classifying the various approaches in the literature. In this paper we evaluate the use of instance-based explanations for time series by end-users. The work [7] studied how non-experts handled post-hoc example-based explanations, however not in the time series domain. They found that even though these do assist users with correct judgement, people have significant difficulties dealing with misclassifications in an unfamiliar domain. Thus we should maybe not expect very high accuracy

from our own evaluations on end-users, as time series are notoriously hard for humans to interpret [18].

Interactivity and model inspection have recently been seen as important in XAI. The paper [2] argues that interactivity in XAI is a core value in the interface between the model and the user, and that a user study is needed for a qualitative evaluation. In [22], the authors develop an interactive XAI tool for loan applications that allows users to experiment with hypothetical input values and inspect their effect on model outcomes, and perform a user evaluation on MTurk.

User evaluations of XAI systems come in various forms. A taxonomy of interpretability evaluation, from the gold standard of Application-oriented evaluations, to Human-grounded evaluations as we perform in this work, to Functionally-grounded evaluations that do not require human experimentations is developed in [6]. For time series it seems the latter approach is the more common. The authors in [16] apply several XAI methods previously used on image and text domain to time series, and introduces verification techniques specific to times series, in particular a perturbation analysis and a sequence evaluation, but they do not include any user evaluation of their systems. Likewise, [9] presents a Python package to provide a unified interface to interpretation of time series classification, but no user evaluation.

The work of [4] provides a method for generating counterfactuals for time series classifiers, called Native Guide, that applies Class Activation Mappings [23] to select discriminative areas for modification. They end their paper by arguing that 'Given the ubiquitous nature of time series data and the frequent requirement for explanation, it is clear that experiments with human users and CBR solutions have much to offer in future work.' The survey [21] says about Native Guide that '...evaluating this promising approach involving end users could be promising for future work'. Indeed this is a part of what we do in the current paper as the counterfactuals we show users are exactly the ones generated by Native Guide.

3 Definitions

Let us present formal definitions for Time Series Classification (TSC) and recall basic notions.

Staying consistent with earlier notation [4,21] a time series $T = \{t_1, t_2, \ldots, t_m\}$ is an ordered set of m real-valued observations (or time steps). A time series dataset $D = \{T_1, T_2, ..., T_n\} \in R^{n \times m}$ is a collection of such time series where each time series has a class label c forming a vector of class labels. In this paper we consider only binary classification tasks. Given such a dataset, Time Series Classification is the task of training a mapping b from the space of possible inputs to a probability distribution over the class values. Thus, a black-box classifier $b(T)$ takes a time series T as input and predicts a probability output over the class values. Given a to-be-explained time series T, with predicted label $b(T) = c$ from the black-box classifier, a counterfactual explanation aims to find

how T needs to (minimally) change to some T' for the system to classify it alternatively, as $b(T') = c' \neq c$. We refer to T' as a counterfactual explanation for T, without having to specify the target class since we consider only binary classification tasks. The minimality criterion usually refers to a notion of distance (proximity) between time series, but another criteria property can be sparsity (that T and T' differ on few data points, or on few contiguous sequences of data points) and plausibility (that the instance is not an outlier).

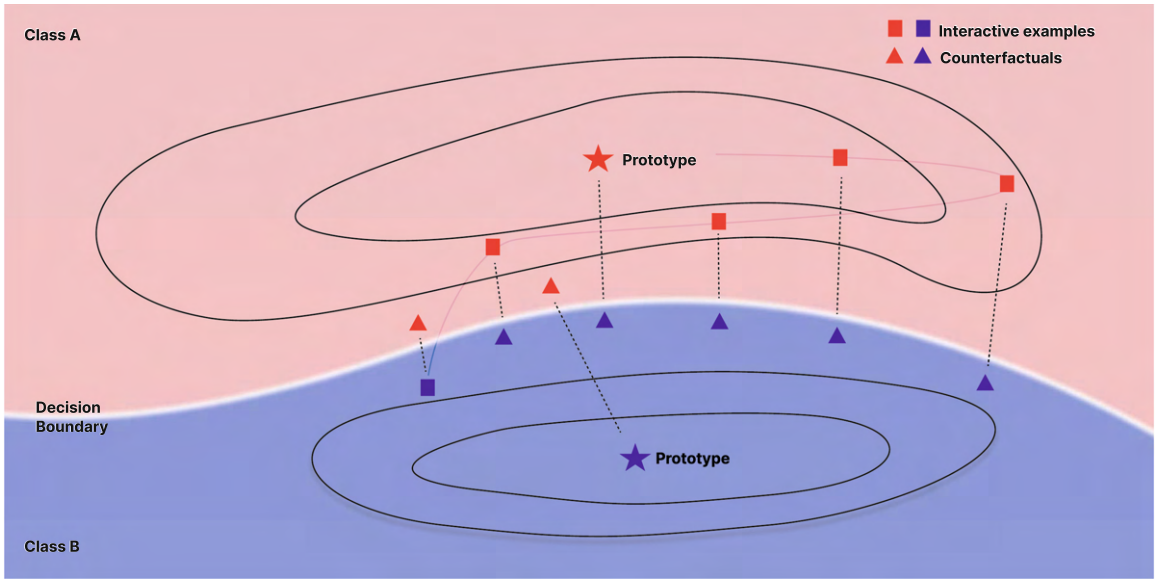


Fig. 1. Explanation types illustrated in a two-class decision space. The two prototypes are representatives of each class, positioned in the middle of the data density contour map. A path of interactive examples and their counterfactuals is shown to illustrate a possible user's journey.

Prototypes are time series exemplifying the main aspects responsible for a classifier's specific decision outcome. It can be a real instance (which is what we opt for in our tool) sampled from the dataset that is important and meaningful because it summarizes the shape of many other similar instances, or a synthetic one, for example a cluster centroid or an instance generated by following some ad-hoc processes. See Fig. 1.

4 Prototypes, Counterfactuals, and User-Model Inspection

In this section we introduce our tool for XAI on time series classification providing model-agnostic instance-based post-hoc explanation, by means of prototypes and counterfactuals. Additionally, one of our contributions is to allow for model inspection on instances generated by the explainee. This interactive tool allows the user to themselves navigate the boundary between classes. Starting from a

prototype, while simultaneously seeing a counterfactual (see Fig. 1) the user can change individual time points at will, and see the resulting classification. This will allow the user to test and improve their hypotheses when formulating a model of the black box classification.

In the rest of this section we describe our tool and start by giving an argument for its motivation, coming from the industry side. We then give a description of the algorithms for generating prototypes plus counterfactuals, and a discussion of how to present these to the user on-screen. We end by reporting on the design choices related to the interactive part, that allows the user to change individual time points and do model inspection.

4.1 The Need for Trust

When presenting output from black box ML systems to non-trained users, the need for explanations arises from at least two angles: The user must trust the results, and the user must, to a sufficient degree understand the results. To use a real-world example from energy companies [19], if a system based on charging data tells a car owner that their car seems faulty, or tells a home owner that their power consumption deviates from expectations - they would need to trust their correctness, otherwise the message will be ignored. When trust is established, the next step is usually to fix the situation. If the car owner understands the reason for their car charging being classified as faulty, he/she can bring it to the vendor or repair shop with concrete information that can be used to fix it. Similarly, if the home owner understands why their consumption is classified as deviating - they could perhaps fix a broken appliance - or adjust their consumption to a more favourable price pattern. These are situations encountered at Eviny, a Norwegian energy company collaborating on the present tool. The more complicated the data and classifications are (for example consisting of several series of data measuring power, temperature, etc. - and/or having temporal patterns like slowly decreasing trends), the more challenging will the explanation be. Thus - exploring different approaches and tools for explanations for non-trained users is of great use when companies plan on applying machine learning on complex data. Without trust the models will be ignored, and without an understanding of what the actual problem is no appropriate action can be taken. Of course, one can ask how much benefit non-trained users can gather from example-based explanations in the realm of time series, and this is indeed one of the questions guiding the user evaluations done in this work.

4.2 Generating and Presenting Prototypes and Counterfactuals

We have opted to use cluster centers as our prototypes. To ensure the prototypes we selected have high plausibility, we use a k−medoids algorithm to find the prototypes, see [12]. Specifically we used KMedoids from sklearn_extra.cluster package. We used default configuration of KMedoids, with Euclidean distance as the distance metric.

There is a growing consensus that counterfactuals provide robust and informative explanations to a query time series whose classification is to be explained. In the time series domain the visualization of counterfactuals is straightforward. We have opted to use a novel method for generating counterfactuals for time series called Native Guide, developed recently by Delaney et al. [4]. This method extracts counterfactual time series, named Native Guides, starting from initial training data. Starting from the query time series whose classification is to be explained, the Native Guide method starts by finding a counterfactual time series belonging to the dataset that is close to the query. This Native series are then adapted in a Guided way to generate novel counterfactuals, following four identified key properties for good counterfactuals: proximity, sparsity, plausibility, and diversity. The Native Guide counterfactual generation method uses Class Activation Mappings to Guide the counterfactual generation from the Native series. The use of CAM in itself puts some limitations on the AI model used, e.g. having the last layer be global average pooling, so in that sense is not completely model-agnostic. To ensure compatibility between the model doing the classification and the Native Guide counterfactual generator, we therefore closely follow the time series classification model implementation of Delaney et al. available here.

When presenting time series to the user we have opted to use two colours for the two classes, namely Blue and Pink. Since a counterfactual, say Blue, will be used to explain the classification of a given query Pink time series, we have chosen to present both at the same time to the user. Thus, we plot the query with Pink lines, and then show only the deviation of the counterfactual by Blue dotted lines. See Figs. 2, 3, 4 and 5 which show also that the user is allowed to make interactive changes, as described in the next subsection. Note that the y-values in the figures represents the normalized values for each dataset.

4.3 Allowing Interaction and Model Inspection

A central aspect of our tool is that it allows the user to alter individual data points and do model inspection. In real-time the tool will update the model classification by changing the color of the time series if the classification changes, showing the model confidence in this classification. It will also update to a new counterfactual. This enables active learning and allows the user to test and improve their hypotheses when formulating a mental model of the black box classification. In Figs. 2, 3, 4 and 5, we see 4 screen shots of an actual session with the tool: Fig. 2) User starts with a Pink prototype and a counterfactual. Figure 3) Makes changes to left end of series so that confidence (top bar) of model classification drops, plus new counterfactual, but same color/class. Figure 4) Makes further changes and now the model classification switches. Figure 5) Last changes made by user and confidence of model classification increases. Note how the user can explore their understanding of the classification by progressive changes to the current time series. Compare also with Fig. 1.

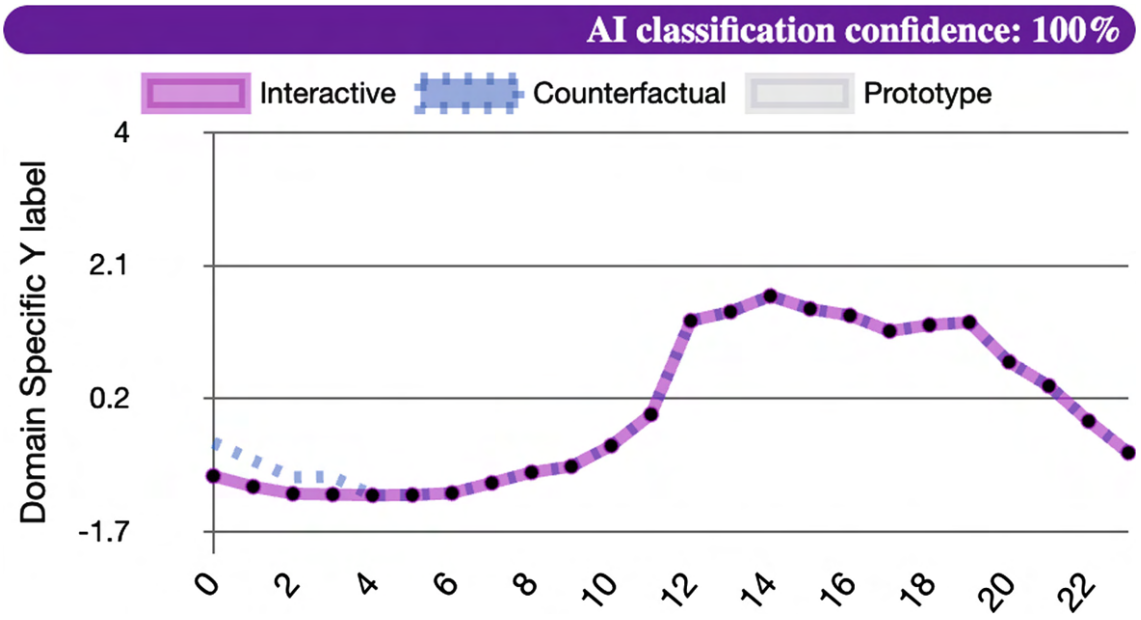


Fig. 2. Start in Pink prototype. (Color figure online)

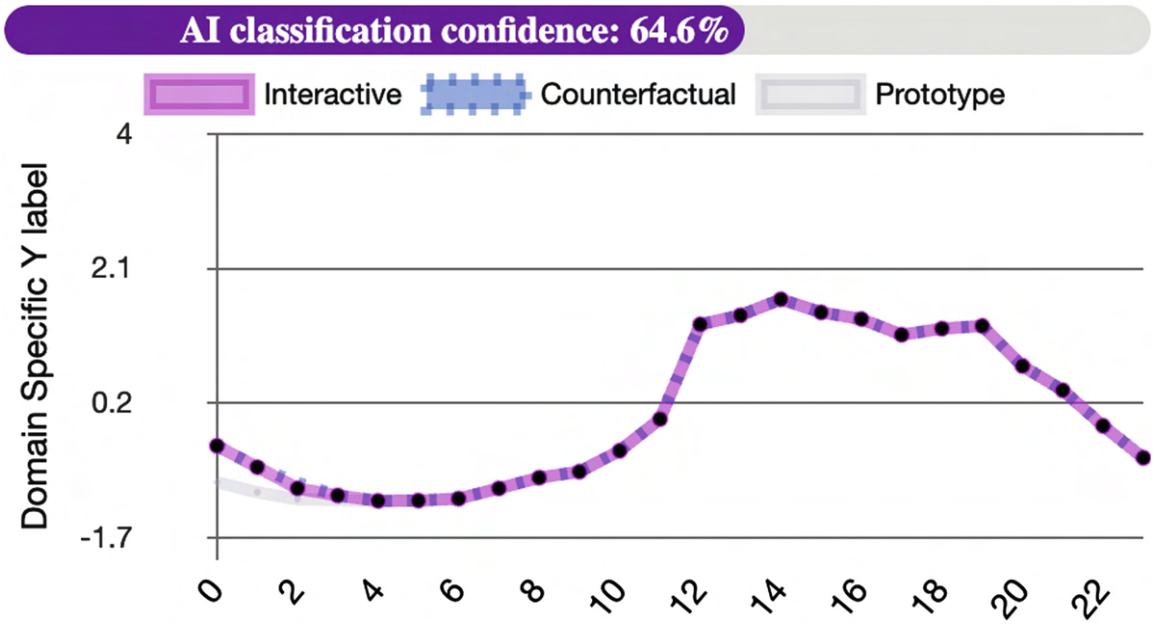


Fig. 3. Changes at left end, still Pink. (Color figure online)

5 End-User Evaluation with Forward Simulation

We have not before seen such an interactive tool for time series in the research literature. As time series are notoriously difficult for human users, most of the human evaluations of XAI methods done so far have been qualitative and involving domain experts. Our goal in this work has been to add the new dimension of quantitative user evaluations involving end-users, i.e. non-experts, on XAI for time series. In this section we present this user evaluation. We start by presenting

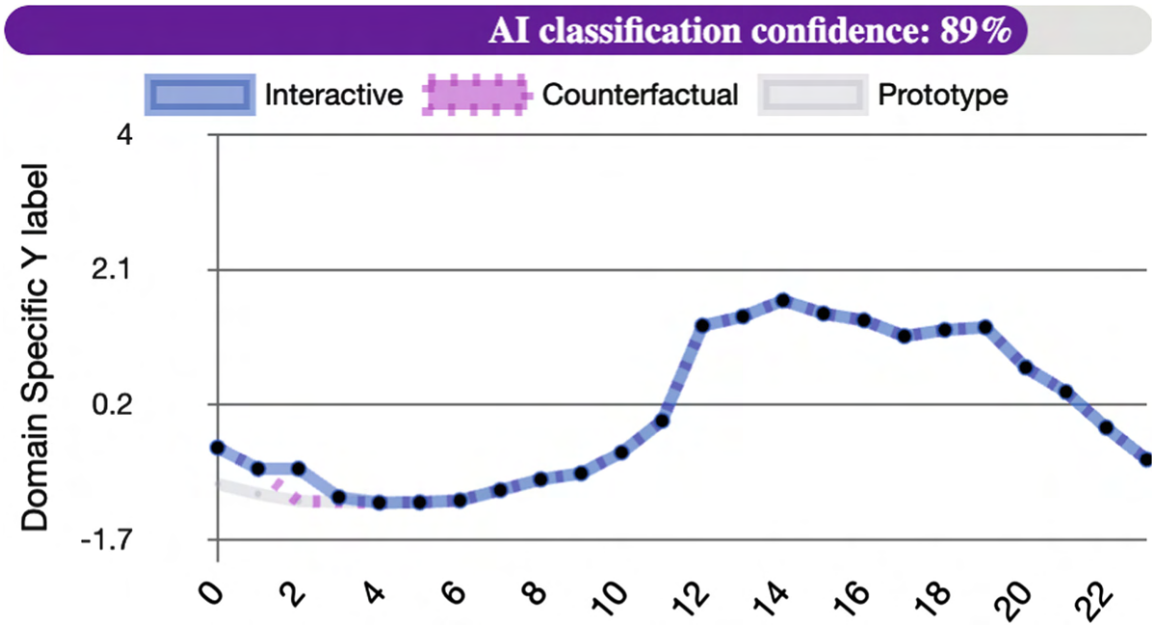


Fig. 4. Further changes, now Blue. (Color figure online)

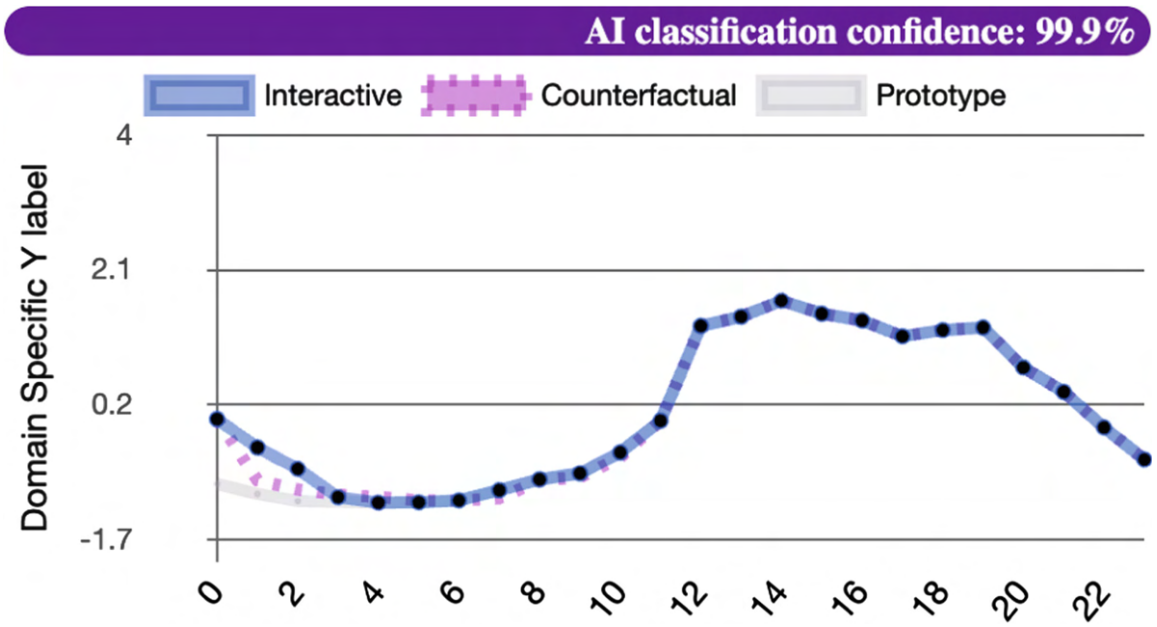


Fig. 5. Last changes, still Blue. (Color figure online)

the discussion leading up to the 3 time series datasets chosen for the evaluation. We then discuss the 3 distinct survey groups that will be given different abilities in the training stage. We end by giving the results of the evaluation, on each pairing of dataset and survey group, and a discussion of their significance.

Our tool is designed for univariate datasets. Moreover, since our test users are not domain experts and time series are notoriously non-intuitive to us humans we had some desiderata when choosing datasets for our survey. Firstly, we wanted univariate datasets with a binary classification, Secondly, we wanted datasets on

time series with not too many data points. Thirdly, we wanted datasets where it is fairly easy for a non-expert to understand the domain. Datasets satisfying these criteria will increase the prospective of providing constructive quantitative feedback, and also they form the more common situation where an explanation would potentially be provided for end-users. However, it is important to note that apart from the above constraints we did not want simple datasets where the binary classification is particularly easy or could be described (e.g. by ourselves) in any straightforward way. We chose the following three datasets satisfying the above criteria:

- From UCR [3]: Italy Power Demand. This dataset shows power demand in Italian households over a 24-h time period, and classifies these into Winter (October-March) and Summer (April-September).
- From UCR [3]: Chinatown. This dataset shows the number of pedestrians on a particular street corner of Chinatown in Melbourne over a 24-h time period, and classifies these into Weekend (Sat-Sun) and Weekdays (Mon-Fri).
- From Eviny: Car Charging. This dataset shows the power demand at a particular charging station for electric vehicles over a 24-h period, and classifies these into Weekend (Sat-Sun) and Weekdays (Mon-Fri).

Table 1. Information of the datasets employed in the evaluation. We also include information about the proportion of data employed for training and testing the data, and the accuracy obtained by the learned model.

Dataset	Number of instances	Class distribution	Train/Test	Accuracy
ItalyPowerDemand	363	104/259	6%/94%	98%
ChinaTown	1096	547/549	5%/95%	96%
Car Charging	365	269/96	75%/25%	57%

Some details of the datasets employed in the experiments can be found in Table 1. We also include the accuracy obtained by the trained models. In the datasets we used the split between training and test defined and the repository except from Car charging where we employed a random selection of 75% train and 25% test.

As mentioned earlier, the three main components used in our interactive XAI system for time series are prototypes (PT), counterfactuals (CF), and model inspection (MI) with user-generated instances. Our survey groups will enter a 3-stage process, as follows:

- Intro: A short introduction is given to the relevant components, the dataset, training stage, and testing stage.
- Training: Group dependent.
- Testing: 10 randomly selected time series from the dataset are shown, and for each one the user must guess the AI model classification. See Figure below.

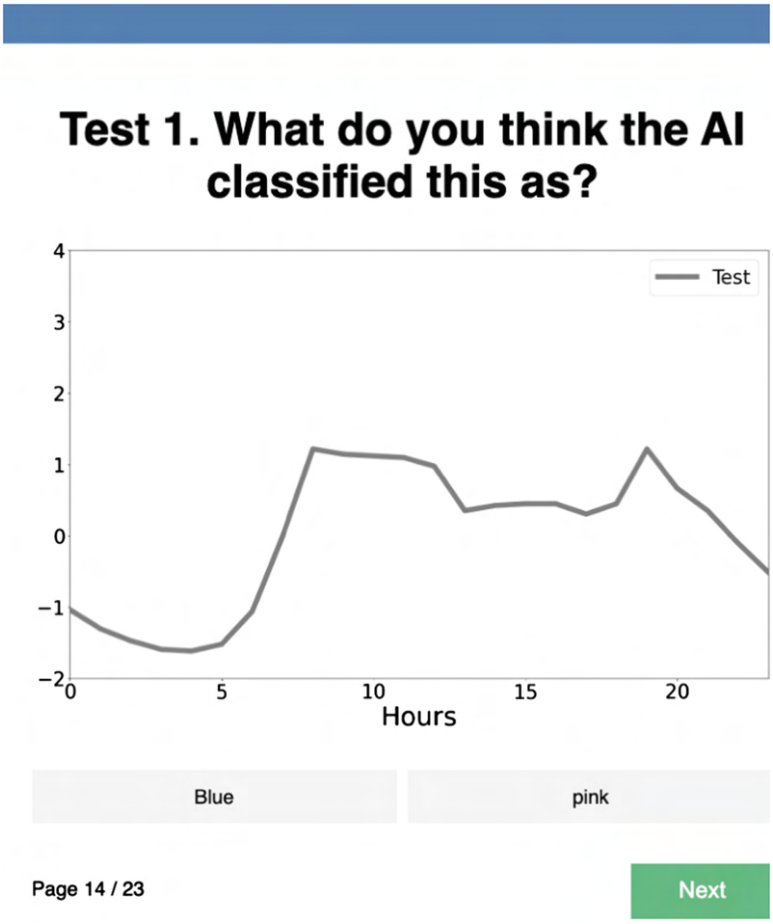


Fig. 6. Test case for ItalyPowerDemand.

To cover the distribution of time series within each class, we wanted to show users more than one prototype for each class. However, we did not want to overload the user with information, hence we opted to have 6 prototypes in total, three for each class. We will have three survey groups (PT, PT+CF, PT+CF+MI) depending on what is made available to users in the Training stage:

- PT: One at a time, the user is shown 3 prototypes from class A, then 3 from class B, and finally shown a screen with all 6 prototypes.
- PT+CF: One at a time, the user is shown 3 prototypes from class A together with a counterfactual from class B, then 3 converse pairs, and finally shown a screen with all 6 pairs.
- PT+CF+MI: The user is shown same as PT+CF but model inspection is permitted, with prototypes being interactive to allow iterative changes of any chosen time points, and the ensuing classification and also new counterfactual continually updated.

In Figs. 2, 3, 4 and 5 we see how a user in group PT+CF+MI is shown a pair consisting of a prototype and a counterfactual and is allowed to modify data points. A user in group PT+CF is shown a single such pair consisting

of prototype and counterfactual but without the ability to make modifications, while a user in group PT is shown only the prototype. In the last part of the training stage the users are shown all 6 prototypes/counterfactuals on one screen, to easily make a comparison, as in Fig. 7 for group PT+CF+MI.

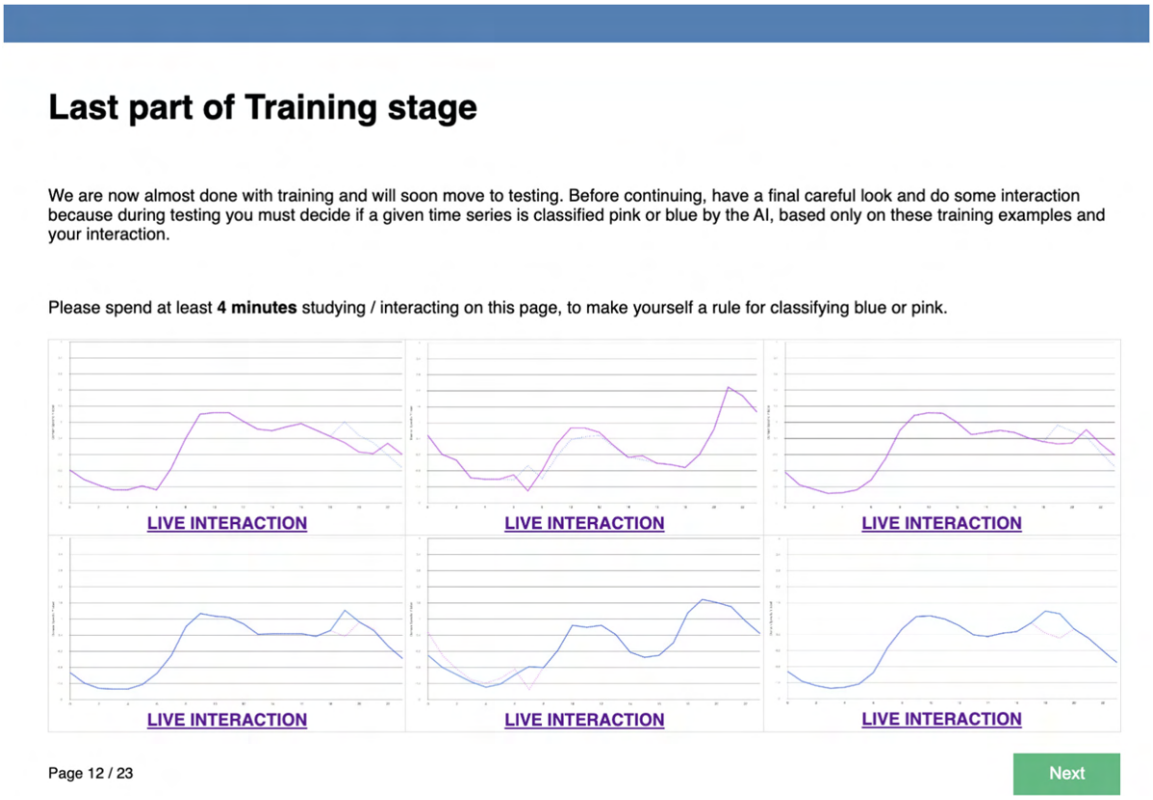


Fig. 7. Blue and pink prototypes with counterfactuals, for ItalyPowerDemand. (Color figure online)

5.1 Evaluation Results

Let us present the results of the user survey. In total, we had 65 voluntary participants, who were either students in a university-level informatics course, or researchers in informatics, thus with experience in using PCs, none of whom received compensation. The participants were presented with the survey and freely chose to participate. The participants were randomly divided into the 3 survey groups. After introduction and training they were asked to answer 10 test questions from each of up to 3 datasets. Those who spent less than a minimum amount of time (less than 1 min for training and testing combined on a single dataset) were discarded, as we considered that it would be impossible to even read the provided information in that time.

In Table 2 we see the accuracy and number of tests (i.e. number of participants times 10) for each of the three survey groups, on each of the three datasets. Accuracy is the percentage of correct test answers. The rightmost column gives the aggregated information for each survey group, and the bottom row the aggregated information for each dataset. The value in the bottom right corner shows that the overall accuracy was 66.7%, thus clearly better than a random guess, and satisfactory given the complicated nature of these time series classifications.

Let us compare the performance between the three survey groups. Perhaps surprisingly, on aggregate we see that survey group PT that was only shown the prototypes, did slightly better than the other two, with accuracy of 70.2% versus 65.9% and 64%. This could indicate that non-expert users are not necessarily able to use the extra information provided by counterfactuals and model inspection in a meaningful way. It could also mean that a 'rule-of-thumb' that appears useful based only on prototypes (or prototypes + counterfactuals) actually works well on a high percentage, say 80%, of test instances. However, after close model inspection a user in group PT+CF+MI may discover that this rule is not precise (as it fails on several instances) leaving the user to discard this rule-of-thumb, and subsequently actually performing worse on the tests. A final possible explanation for why group PT+CF+MI did not achieve higher accuracy is the fact that the average time spent by users in total on all 3 datasets was about the same, around 15 min: group PT 954 s, group PT+CF 934 s, group PT+CF+MI 966 s. However, it seems natural that a user doing model inspection to learn a better rule should have spent more time than one who cannot do model inspection, making us question how careful the users in group PT+CF+MI were. On the other hand, there is not a clear trend when we compare accuracy versus time spent for users in this group.

Let us turn to comparing between the 3 datasets. Interestingly, all 3 survey groups had the highest accuracy on Chinatown and the lowest accuracy on CarCharging. We asked some users about rules they had been using for their own mental classification, and the most mentioned rule was for Chinatown, something like the following: 'if there is a sharp dip at the beginning of the series then it is Blue, and otherwise Pink'. Compare this rule to Figs. 2, 3, 4 and 5 from the Chinatown dataset. See also Figs. 7 and 6 which for ItalyPowerDemand gives all prototypes and counterfactuals, plus an example of a test, to get an impression of how difficult the classification indeed is for this dataset. For the test shown in Fig. 6 the average accuracy was 59.3%. Note the users could not navigate back to see the prototypes when answering the tests.

Table 2. Overview of accuracy and number of tests by survey group and dataset.

Survey ID	Data	ItalyPowerDemand	Chinatown	CarCharging	All 3 datasets
PT	accuracy	$65.9\% \pm 10$	$79.4\% \pm 13$	$65.3\% \pm 14$	$70{,}2\% \pm 14$
	tests	170	170	170	510
PT+CF	accuracy	$61.9\% \pm 17$	$81.3\% \pm 13$	$54.4\% \pm 21$	$65.9\% \pm 20$
	tests	160	160	160	480
PT+CF+MI	accuracy	$60.2\% \pm 16$	$76.2\% \pm 22$	$55.7\% \pm 13$	$64.0\% \pm 20$
	tests	240	210	230	680
All groups	accuracy	$62.7\% \pm 15$	$79.0\% \pm 17$	$58.4\% \pm 17$	$66.7\% \pm 19$
	tests	570	540	560	1670

6 Conclusion

As companies and controllers must cope with the EU regulations in form of GDPR and the AI Act, explainability that allows model inspection by end-users may very well become the target for developers. In domains where we humans have poor intuition, such as time series, this may pose several challenges. In this work we have contributed a tool for XAI on time series classification that allows such model inspection. The evaluation results show that users can then successfully perform a difficult forward simulation test. However, to attain the full benefits from model inspection for such a complicated domain, it seems necessary to have highly motivated users that are willing to spend more time with such a tool, in order to form better mental models of the black-box classification. As future work, we propose to explore the use of simplified versions of time series prototypes generated by Machine Teaching techniques to explain time series classification models.

Acknowledgement. This paper is part of NRF project 329745 Machine Teaching For XAI. We thank the anonymous reviewers for their comments and the volunteers who participated in the experiments. This work was funded by ValGrai, CIPROM/2022/6 (FASSLOW) and IDIFEDER/2021/05 (CLUSTERIA) funded by Generalitat Valenciana, the EC H2020-EU grant agreement No. 952215 (TAILOR), US DARPA HR00112120007 (RECoG-AI) and Spanish grant PID2021-122830OB-C42 (SFERA) funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

Disclosure of interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abdul, A.M., Vermeulen, J., Wang, D., Lim, B.Y., Kankanhalli, M.S.: Trends and trajectories for explainable, accountable and intelligible systems: an HCI research agenda. In: Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems. CHI 2018, p. 582. ACM (2018)

2. Beretta, I., Cappuccio, E., Manerba, M.M.: User-driven counterfactual generator: A human centered exploration. In: Conference on eXplainable Artificial Intelligence (xAI-2023), pp. 83–88. CEUR Workshop Proceedings (2023)
3. Dau, H.A., et al.: The UCR time series classification archive (2018)
4. Delaney, E., Greene, D., Keane, M.T.: Instance-based counterfactual explanations for time series classification. In: Sánchez-Ruiz, A.A., Floyd, M.W. (eds.) ICCBR 2021. LNCS (LNAI), vol. 12877, pp. 32–47. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-86957-1_3
5. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. CoRR **abs/1702.08608** (2017)
6. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
7. Ford, C., Keane, M.T.: Explaining classifications to non-experts: an XAI user study of post-hoc explanations for a classifier when people lack expertise. In: Rousseau, J.J., Kapralos, B. (eds.) ICPR 2022. LNCS, vol. 13645, pp. 246–260. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-37731-0_15
8. Guillemé, M., Masson, V., Rozé, L., Termier, A.: Agnostic local explanation for time series classification. In: 31st IEEE International Conference on Tools with Artificial Intelligence. ICTAI 2019, Portland, OR, USA, 4–6 November 2019, pp. 432–439. IEEE (2019). https://doi.org/10.1109/ICTAI.2019.00067
9. Höllig, J., Kulbach, C., Thoma, S.: Tsinterpret: a Python package for the interpretability of time series classification. J. Open Source Softw. **8**(87), 5220 (2023)
10. Ismail, A.A., Gunady, M.K., Bravo, H.C., Feizi, S.: Benchmarking deep learning interpretability in time series predictions. In: Annual Conference on Neural Information Processing Systems 2020. NeurIPS 2020, 6–12 December 2020, Virtual (2020)
11. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Min. Knowl. Disc. **33**(4), 917–963 (2019)
12. Kaufman, L., Rousseeuw, P.J.: Finding Groups in Data: An Introduction to Cluster Analysis. Wiley, New York (1990). https://doi.org/10.1002/9780470316801
13. Miller, T.: Explanation in artificial intelligence: insights from the social sciences. Artif. Intell. **267**, 1–38 (2019)
14. Poché, A., Hervier, L., Bakkay, M.C.: Natural example-based explainability: a survey. In: Longo, L. (ed.) xAI 2023. CCIS, vol. 1902, pp. 24–47. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-44067-0_2
15. Rajkomar, A., et al.: Scalable and accurate deep learning with electronic health records. NPJ Digit. Med. **1**(1), 1–10 (2018)
16. Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., Keim, D.A.: Towards a rigorous evaluation of XAI methods on time series. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), pp. 4197–4201. IEEE (2019)
17. Shneiderman, B.: Human-centered artificial intelligence: reliable, safe & trustworthy. Int. J. Hum.-Comput. Interact. **36**(6), 495–504 (2020)
18. Siddiqui, S.A., Mercier, D., Munir, M., Dengel, A., Ahmed, S.: Tsviz: demystification of deep learning models for time-series analysis. IEEE Access **7**, 67027–67040 (2019). https://doi.org/10.1109/ACCESS.2019.2912823
19. Su, L., Zhang, S., McGaughey, A.J., Reeja-Jayan, B., Manthiram, A.: Battery charge curve prediction via feature extraction and supervised machine learning. Adv. Sci. **10**(26), 2301737 (2023)

20. Susto, G.A., Cenedese, A., Terzi, M.: Time-series classification methods: review and applications to power systems data. In: Big Data Application in Power Systems, pp. 179–220 (2018)
21. Theissler, A., Spinnato, F., Schlegel, U., Guidotti, R.: Explainable AI for time series classification: a review, taxonomy and research directions. IEEE Access **10**, 100700–100724 (2022)
22. Wang, Z.J., Vaughan, J.W., Caruana, R., Chau, D.H.: GAM coach: towards interactive and user-centered algorithmic recourse. In: Proceedings of Conference on Human Factors in Computing Systems, pp. 835:1–835:20. ACM (2023)
23. Zhou, B., Khosla, A., Lapedriza, À., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2016, Las Vegas, NV, USA, 27–30 June 2016, pp. 2921–2929. IEEE Computer Society (2016)

Paper V

Telle, J. A., Ferri, C., & Håvardstun, B.
Proceedings of the Workshop on Explainable AI for Time Series and Data Streams (TempXAI 2024), co-located with ECML PKDD 2024, CEUR **3761**, pp. 28–43 (2024)
URL: https://ceur-ws.org/Vol-3761/paper2.pdf

Optimal Robust Simplifications for Explaining Time Series Classifications

Jan Arne Telle[1], Cèsar Ferri[2] and Brigt Håvardstun[1]

[1]Department of Informatics, University of Bergen, Norway
[2]VRAIN, Universitat Politècnica de València, Spain

Abstract

Given a Time Series Classifier (TSC) and three parameters that balance between error, simplicity and robustness, we define an optimization problem over all possible ways of simplifying a given time series ts into straight-line segments. Robustness is fixed as the fraction of perturbations that have the same classification as ts under the TSC, and we introduce a novel method for generating a set of perturbations where this information is easy to visualize. We prove that under some mild conditions on the TSC and the three parameters, we can find the optimal solution in time polynomial in the length of ts, by first doing dynamic programming to solve for error and simplicity, and then adding robustness. We test the resulting Optimal Robust Simplification (ORS)-Algorithm on binary TSCs for three datasets from UCR. We apply the ORS-Algorithm to prototypes of the classes, with varying parameters, to evaluate its power as an explanatory tool for the trained classifiers. We also provide a tool for visualizing the robustness information. We believe the resulting insights show the usefulness of the Optimal Robust Simplifications in explaining TSCs.

Keywords

Time Series, XAI, Simplification, Explainability

1. Introduction

Temporal data is encountered in many real-world applications ranging from patient data in healthcare [1] to the field of cyber security [2]. Deep learning methods have been successful [3, 1, 2] for TSC - Time Series Classification - but such methods are not easily interpretable, and often viewed as black boxes, which limits their applications when user trust in the decision process is crucial. To enable the analysis of these black-box models we revert to post-hoc interpretability. Recent research has focused on adapting existing methods to time series, both specific methods like SHAP-LIME [4], Saliency Methods [5] and Counterfactuals [6], and also combinations of these [7].

As humans learn and reason by forming mental representations of concepts based on examples, and any machine learning model has been trained on data, then we believe that data e.g. in the form of prototypes and counterfactuals is indeed the natural common language between the user and this model. However, the basic problem is that compared to images and text, time series data are not intuitively understandable to humans [8]. This makes interpretability of time series extra demanding, both when it comes to understanding how users will react to the provided explanations and to predict what explanatory tools are best. For example, in [9] a tool was given for explainability of a TSC, that allowed model inspection so the user could form their own mental model of the classifier. However, a user evaluation showed that non-expert end-users were not able to make use of this freedom, supporting the notion that time-series are particularly non-intuitive for humans.

Theissler et al [10] give an intriguing taxonomy of XAI for TSC, divided into those focused on (i) Time-points (e.g. SHAP and LIME) or (ii) Subsequences (e.g. Shapelets) or (iii) Instances (Prototypes, CF, Feature-based). Explaining a classifier by instances like prototypes has a definite appeal, but it is a challenge how to highlight the features important for the classification, while at the same time simplifying the instance to mitigate the non-intuitive aspects of this domain, as we attempt in this work. Theissler et al [10] review the literature on XAI for TSC and of the 9 papers they discuss on Instance-based explanations using Prototypes or Features, it is remarkable that not a single one is

TempXAI: Explainable AI for Time Series and Data Streams Tutorial-Workshop, September 9, 2024, Vilnus, Lithuania

specialized for Local explanations, rather they are all Global. In this work we fill this gap, and introduce a way of *simplifying* a time series by straight-line segments, and doing this in a way that is *robust* with respect to the classification of a given TSC h.[1] Our ORS-algorithm, for Optimal Robust Simplifications, is model-agnostic, with robustness of a simplification being the fraction of local perturbations that retains the classification under TSC h. There are various ways of perturbing times series, see e.g. [11, 12, 13]. In this work we develop a different method of computing a set of perturbations, both since we start with a simplification on straight-line segments and also because we want the aggregate information about their classifications to be visualized in an informative way. When computing robustness we start with a straight-line simplification, and any perturbation must to a certain degree keep that position and shape, while it is also allowed to deviate as the perturbation moves inside a band around the simplification, see Figure 8.

The ORS-Algorithm computing a robust simplification sts of a given times series ts, under a TSC h, has three parameters that allows to balance between error (Squared Euclidean distance between ts and sts), simplicity (number of segments of sts), and robustness (what fraction of perturbations of sts have same classification as ts by h), and sts is defined as the minimization over an objective function taking all this into account, see Definition 1. Perhaps surprisingly, in Theorem 4 we show that under some mild conditions on h and the three parameters, we can find the optimal solution in time polynomial in the length of the original time series ts, by first solving for error and simplicity, and then adding robustness. Note that our algorithm allows not only the balance between error, simplicity and robustness to change, but also their computation, i.e. the algorithm is agnostic to the particular methods used to calculate these values. Hence Squared Euclidean distance can be replaced by some other distance function as long as it is amenable to the dynamic programming, simplicity can depend on other aspects than number of segments, and robustness can be computed by other types of perturbations.

In the rest of this paper we first discuss related work and cover some standard definitions before presenting the ORS-Algorithm together with proof of correctness and runtime. We then discuss the usefulness of the robust simplifications, with a focus in this paper on time series that are discrete, univariate, evenly sampled, aperiodic, short, and non-stationary, based on UCR datasets Chinatown, Italy Power Demand and ECG200. We also provide an aid to visualize the robustness information. Our findings suggest that the robust simplifications of a prototype ts can be used to extract explanations of the classification done by the TSC, as they simplify ts to the point where human intuition can more easily be applied. We conclude the paper by discussing some directions for future work.

2. Related Work

Several techniques have been applied to generate explanations from TSC models [10, 14]. Specifically, techniques previously used on Convolutional Neural Networks or Recurrent Neural Networks have been applied when these techniques have been used for TSC. For instance, the authors in [7] apply to time series several XAI methods previously used on image and text domain. They also introduce verification techniques specific to times series, in particular a perturbation analysis and a sequence evaluation. Related to the datasets used in our paper, [15] presents a user study of XAI methods to explain the ECG200 dataset from UCR.

Another alternative is to produce explanations through examples, and these can be specifically utilized in the time series domain. One type of this explanation method involves giving the nearest example from the training dataset that acts as a prototype to depict the normal behavior of a similar sample [16]. A method to generate prototypes for time series data using an autoencoder is presented in [17]. The main novelty of the work is the method to generate diverse prototypes.

Given that in many cases raw time series can be too complex for humans, several studies have tried to employ simplified versions as explanations. In [18], the authors propose a dimensionality reduction technique for time series, called Piecewise Aggregate Approximation, as a tool for similarity search

[1] Note that if one pieces together several such local explanations, say for prototypes of each class, then this can still form a global explanation of the given TSC h.

in large time series databases. The work [19] presents a review of piecewise linear approximations employed for time series data, and the authors also propose a method named SWAB (based on the Sliding Window and Bottom-up approaches). Finally, in [20] Camponogara and Nazari introduce a range of piecewise-linear models and algorithms for unknown functions. Another option is to segment time series into fixed-width windows and employ these to justify decisions. In [21] Schlegel et al propose TS-MULE, a model explanation method working to segment and perturb time series data and extending LIME. A study on the effect of segmentation on time series, in particular in the field of finance for the use of pattern matching was presented in [22]. In [23], Si and Yin also apply segmentation to financial time series data, now as a preprocessing step to locate technical patterns.

Perturbations have also been previously employed for XAI and time series. In [11], the authors propose a framework to generate explanations for time series classifications based on a generative model to create with-in-distribution perturbations. Enguehard, in [12], introduces a technique to explain multivariate time series predictions using a perturbation-based saliency method. Recently, the approach ContraLSP has been detailed in [13]. This method is based on a locally sparse model that produces counterfactual samples to build uninformative perturbations but keeps distribution using contrastive learning.

Some tools have been proposed to allow users to interact with time series and, in that way, to get some insights about their classification, like [24] where model inspection is the key aspect. In [25], the authors develop an interactive XAI tool for loan applications that allows users to experiment with hypothetical input values and inspect their effect on the outcomes of the model and perform a user evaluation on MTurk. [26] presents a Python package to provide a unified interface for the interpretation of time series classification.

Some papers have developed a similar approach to our proposal. Im [27], Tang et al. presents the method Dual Prototypical Shapelet Networks (DPSN) for few-shot time-series classification. This method trains a neural network from few examples and also interprets the model from two levels of granularity: a global overview with representative time series examples, and local highlights with discriminative shapelets. Another related paper for sequence learning is [28]. The work proposes ProSeNet a RNN-based method for deep sequence modeling. The approach combines prototype learning and RNNs to construct models with high accuracy and interpretability. The method considers three criteria in constructing prototypes: Simplicity, the prototypes can be subsequences with only the key events determining the output; Diversity, redundant prototypes should be avoided; Sparsity, for each example to explain only a few prototypes are shown to avoid long explanations. The authors show the validity of ProSeNet for time series using the MIT-BIH Arrhythmia ECG dataset.

3. Standard definitions

Let us present formal definitions for Time Series Classification (TSC) and recall basic notions.

Staying consistent with earlier notation [10, 6] a time series $T = \{t_1, t_2, \ldots, t_m\}$ is an ordered set of m real-valued observations (or time steps). Note we may also view each t_i as a pair consisting of an x-value (the time) and a y-value (the observation). A time series dataset $D = \{T_1, T_2, ..., T_n\} \in R^{n \times m}$ is a collection of such time series where each time series has a class label c forming a vector of class labels. In this paper we consider only binary classification tasks. Given such a dataset, Time Series Classification is the task of training a mapping b from the space of possible inputs to a probability distribution over the class values. Thus, a black-box classifier $b(T)$ takes a time series T as input and predicts a probability output over the class values.

Prototypes are time series exemplifying the main aspects responsible for a classifier's specific decision outcome. It can be a real instance (which is what we opt for) sampled from the dataset that is important and meaningful because it summarizes the shape of many other similar instances, or a synthetic one, for example a cluster centroid or an instance generated by following some ad-hoc processes.

4. An Algorithm for Optimal Robust Simplifications

In this section we give the ORS-Algorithm that takes a binary Time Series Classifier h and a univariate times series ts, on n points $p_1, ..., p_n$ given by increasing x-values, and computes the optimal robust simplification $opt_{simp}(ts, h, \alpha, \beta, \gamma)$, where α, β, γ are factors freely chosen to balance between error, simplicity and robustness. This simplification will select $k+1$ of the n points $p_{i_1}, p_{i_2}, ..., p_{i_{k+1}}$ with $i_j < i_{j+1}, \forall j$ to form a simplified time series sts on k segments by drawing a line between each consecutive pairs of the $k+1$ points and by letting the first segment and the last segment extend to the x-values of p_1 and p_n, respectively. See Figure 1.

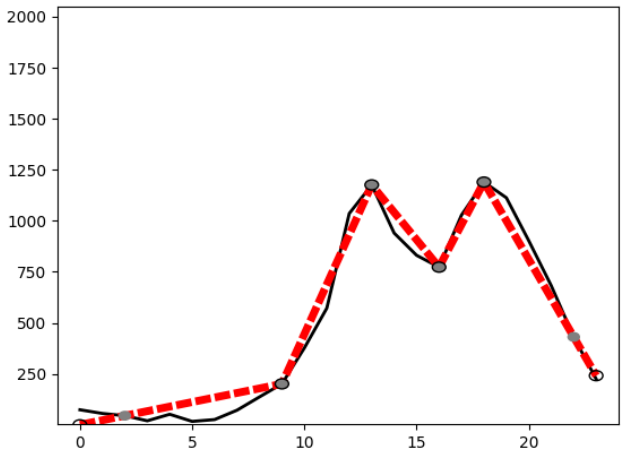


Figure 1: Original time series ts in black, with selected points given by 6 grey circles, resulting in a simplification sts in red with 5 segments.

Definition 1. *We define $opt_{simp}(ts, h, \alpha, \beta, \gamma)$ formally as the minimum, over all $2^n - n - 1$ possible choices of $2 \leq k \leq n$ points that give a simplification sts with $h(ts) = h(sts)$, of the value*

$$\alpha \cdot err(ts, sts) + \beta \cdot k_{sts} + \gamma \cdot frag(ts, sts, h)$$

comparing the original time series ts and the simplification sts on $k_{sts} = k - 1$ segments defined by these k points.

The optimal simplification thus reflects a selected balance of error, simplicity and robustness achieved by freely choosing the 3 factors α, β, γ:

- α for error: $err(ts, sts)$ is Squared Euclidean distance between ts and sts, i.e. the sum over all x-values, of the square of the difference between the corresponding y-value of ts and sts.
- β for simplicity: k_{sts} is the number of segments
- γ for robustness: $frag(ts, sts, h)$ is a measure of how robust the classifier h is on perturbations of sts, measured by the fraction of perturbations that do not fall in the same class as ts, thus in the range $[0, 1]$ with 0 being most robust ($frag$ is short for fragility and can be viewed as (1-robustness)). See subsection 4.2.2 for the definition of the perturbations used.

Perhaps surprisingly, under some mild assumptions this complicated objective function still allows an algorithm that finds the optimal in time polynomial in n. In this section we describe this algorithm in detail.

4.1. If no robustness: Dynamic Programming

We first solve the case of $\gamma = 0$, i.e. without giving any weight to robustness. Firstly, Least Squares is a foundational problem in statistics and numerical analysis, given points $p_1, p_2, ..., p_n$ in the plane find the straight line that minimizes the squared error. In the more general Segmented Least Squares (SLS) problem we ask for a sequence of straight-line segments that minimizes cost, which is some balance of error and simplicity (number of segments). The SLS problem is famously solvable in polynomial time by a textbook Dynamic Programming Algorithm based on the following recurrence for $OPT(j)$, the minimum cost for points $p_1, ..., p_j$:

$$OPT(j) = \begin{cases} 0, & \text{if } j = 0 \\ \min_{1 \le i \le j}\{lse(i,j) + \beta + OPT(i-1)\} & \text{else} \end{cases}$$

where $lse(i, j)$ is the least square error over all single lines that cover p_i to p_j, and β is the the cost of one extra segment, with value of β chosen to balance between error and simplicity. This problem formulation differs from our setting in 3 important ways:

- We want a continuous set of segments, with each segment starting and ending in given points.
- We want the first (resp. last) segment to be defined by any two points p_i, p_j and the line drawn between them continued until it hits the x-value of p_1 (resp. of p_n).
- We want robustness to perturbations as part of the objective function.

The first two differences are relatively easy to accommodate, as follows:

$$OPT(j) = \begin{cases} 0, & \text{if } j = 0 \\ \min_{1 \le i \le j}\{[\alpha \cdot err(i,j) + \beta + OPT(i)], [\alpha \cdot err(1,i,j) + \beta]\} & \text{if } j < n \\ \min_{1 \le i \le n} \min_{i+1 \le k \le n}\{[\alpha \cdot err(i,k,n) + \beta + OPT(i)], & \\ \qquad [\alpha \cdot err(1,i,k,n) + \beta]\} & \text{if } j = n \end{cases}$$

where

- $err(i, j)$ is the Squared Euclidean distance between the straight line from p_i to p_j and the part of the original time series ts given by points $p_i, ..., p_j$
- $err(i, j, n)$ is the same except the straight line continues past p_j to the x-value of p_n and we compare to the final part of ts given by $p_i, ..., p_n$
- $err(1, i, j)$ is similar as above except the line continues in the other direction to the x-value of p_1 and we compare to the initial part of ts $p_1, ..., p_j$
- $err(1, i, j, n)$ compares ts to a single line covering all x-values and going through points p_i, p_j

For the case of $opt_{simp}(ts, h, \alpha, \beta, \gamma)$ where $\gamma = 0$ i.e. where robustness does not play a part, the argument for correctness of the above recurrence is similar to the textbook argument for correctness of the recurrence of SLS, so we do not repeat it here, except to note the new parts being that

- We use $OPT(i)$ rather than $OPT(i-1)$ since now for two consecutive segments the endpoint of the first is the startpoint of the next.
- When defining $OPT(j)$ we use $err(1, i, j)$ so first segment can end in j
- When defining $OPT(n)$ we use $err(i, k, n)$ so last segment can choose any pair p_i, p_k, and $err(1, i, k, n)$ to allow simplification with a single segment.

Transforming this recurrence into a dynamic programming algorithm is straightforward, by first computing each of the $O(n^2)$ error terms in time $O(n)$ each, followed by a nested loop computing the n $OPT(j)$ values in $O(n)$ time each. Retrieving the optimal solution is done by a second pass over the values stored, also standard. This gives the following result.

Theorem 1. *Given a time series ts of length n, a binary TSC h, and factors α, β, γ that balance between error, simplicity and robustness. For the case of $\gamma = 0$ (no robustness) we can compute the optimal simplification achieving the minimum $opt_{simp}(ts, h, \alpha, \beta, \gamma = 0)$ in time $O(n^3)$ by dynamic programming.*

4.2. Accommodating Robustness: DP with Heaps

Dynamic programming relies on the property that an optimal solution to a larger problem consists of optimal solutions to subproblems. Robustness depends on the TS Classifier h and we cannot expect it to satisfy this property, since its classification on only a part of a time series will not in general correspond to its classification on the full time series. Thus, we cannot incorporate robustness as part of the DP scheme. Instead, our algorithm for Optimal Robust Simplifications (ORS), the **ORS-Algorithm**, uses a 2-stage approach:

Stage 1 Use DP to compute a 2-dimensional table of size $n \times q$ that in cell $D[j,k]$ stores the kth least costly simplification for the points $p_1, ..., p_j$ of the input time series ts, considering only error and simplicity but not robustness, as in above DP.

Stage 2 Consider the q least costly simplifications $sts_1, ..., sts_q$, ordered by non-decreasing cost under error and simplicity as stored in $D[n,1]..D[n,q]$ respectively. Compute robustness for any simplification that h classifies same as ts.
The ORS-Algorithm then returns the simplification having lowest overall total cost, i.e. the sts minimizing the value of value of $\alpha \cdot err(ts, sts) + \beta \cdot k_{sts} + \gamma \cdot frag(ts, sts, h)$

Pseudo-code is given in the Appendix. We will here give a high-level explanation, but let us first state the following formal result.

Theorem 2. *Let the difference in cost between sts_1 and sts_q, as computed by Stage 1, be d. For any value of $\gamma \leq d$ the simplification returned by the ORS-Algorithm will then be an optimal one achieving $opt_{simp}(ts, h, \alpha, \beta, \gamma)$.*

Proof 1. *We prove the statement by contradiction. Let the simplification sts returned by the ORS-Algorithm have k_{sts} segments and assume there is another simplification sts' with $k_{sts'}$ segments such that*

$$\alpha \cdot err(ts, sts') + \beta \cdot k_{sts'} + \gamma \cdot frag(ts, sts', h) <$$

$$\alpha \cdot err(ts, sts) + \beta \cdot k_{sts} + \gamma \cdot frag(ts, sts, h)$$

Since Stage 2 optimized this value over all simplifications $sts_1, ..., sts_q$ we cannot have sts' among these, and thus by the assumption in the statement of the theorem we have $\alpha \cdot err(ts, sts') + \beta \cdot k_{sts'} \geq d + \alpha \cdot err(ts, sts_1) + \beta \cdot k_{sts_1}$, with k_{sts_1} the number of segments of sts_1. Thus we get

$$d + \alpha \cdot err(ts, sts_1) + \beta \cdot k_{sts_1} + \gamma \cdot frag(ts, sts', h) <$$

$$\alpha \cdot err(ts, sts) + \beta \cdot k_{sts} + \gamma \cdot frag(ts, sts, h) \leq$$

$$\alpha \cdot err(ts, sts_1) + \beta \cdot k_{sts_1} + \gamma \cdot frag(ts, sts_1, h)$$

with the latter inequality following since sts was the minimum over $sts_1, ..., sts_q$. Rearranging, this gives $d < \gamma \cdot frag(ts, sts_1, h) - \gamma \cdot frag(ts, sts', h) + ((\alpha \cdot err(ts, sts_1) + \beta \cdot k_{sts_1}) - (\alpha \cdot err(ts, sts_1) + \beta \cdot k_{sts_1}))$ with the latter term being zero and thus

$$d < \gamma \cdot (frag(ts, sts, h) - frag(ts, sts', h))$$

Note we have $frag(\cdot)$ in the range $[0, 1]$ which means the above implies that $\gamma > d$, contradicting the assumption in the statement of the theorem. □

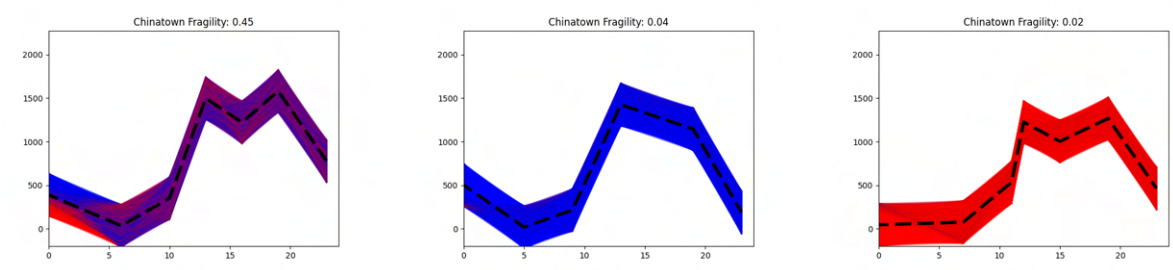
4.2.1. Important details of Stage 1.

We describe some key details of how to implement Stage 1, computing a table of size $n \times q$ with $D[j,k]$ storing the kth least costly simplification of points $p_1, ..., p_j$ under error and simplicity only. The difference to the earlier DP is that we compute not only the least costly but the q least costly with $q > 1$. After computing error terms we run an outer loop from $j = 1$ to n with two inner loops one after the other. The first loop from $i = 1$ to $j - 1$ is similar to the previous DP, computing the cost $D[i,1] + err(i,j)$ of having the last segment go from p_i to p_j, but now adding all these values to a heap H_j. The second loop from $k = 1$ to q fills the $D[j,k]$ entries by extracting the minimum element from H_j, say this element is $D[i,r] + err(i,j)$, then first set $D[j,k]$ to this element, but most importantly also add to the heap H_j the new value $D[i,r+1] + err(i,j)$ since the $r+1$st least costly solution up to p_i may be lower than many other solutions in the heap. For details, in particular regarding the edge-cases which are cumbersome to implement properly, see the pseudo-code in the appendix.

4.2.2. Definition of Perturbations and Robustness in Stage 2.

A simplified time series sts on k segments is computed by choosing $k+1$ points $p_{i_1}, ..., p_{i_{k+1}}$ from an original time series ts on points $p_1, ..., p_n$. When computing the robustness of sts focus on the following $k+1$ *pivot points* $s_1, p_{i_2}, ..., p_{i_k}, s_2$ of sts (note the first and last may not be points of ts) where s_1 (and s_2) is the y-value of sts that corresponds to the smallest (and largest) x-value, where smallest is same as x-value of p_1 (and largest is same as x-value of p_n). In Figure 1 s_1 is the leftmost circled point and s_2 the rightmost circled point, at x-values 0 and 23 respectively. For these $k+1$ pivot points of sts we allow an increase or decrease of its y-value by an ϵ which is 10% of the y-range of the dataset the classifier h was trained on, i.e. $\epsilon = 0.1 * (y_{max} - y_{min})$ with y_{max} and y_{min} the maximum and minimum y-value in this dataset.

We compute 10.000 random perturbations of sts, each consisting of k segments anchored at the same x-values as the $k+1$ pivot points of sts but with the y-value of each pivot point shifted by an amount drawn independently for the $k+1$ values uniformly at random in the range $[-\epsilon, +\epsilon]$. Thus the perturbation will lie inside a band around sts of height $2 \cdot \epsilon$ at each pivot point. See Figure 2. We then use h to classify each of the perturbations, and define the $frag$-value of sts as the fraction of the 10.000 random perturbations that have the opposite classification as ts.



(a) $frag = 0.45$ blue. Not robust.

(b) $frag = 0.04$ blue. Very robust.

(c) $frag = 0.02$ red. Very robust.

Figure 2: Visualizing robustness of the three simplified time series in dotted black lines. All perturbations consist of straight-line segments that are forced to lie inside the colored band, with pivot points at same x-values as the simplification, but different y-values. We compute several perturbations of sts by randomly altering the y-values of the $k_{sts}+1$ pivot points inside the narrow band, e.g. in (b) sts in black has 5 segments and the 6 pivot points of the perturbations are at x-values 0,5,9,13,18,24. Each perturbation is drawn with a color according to its classification by the classifier h. Thus if many have same color as ts then robustness is good (and $frag$-value low).

Theorem 3. *The runtime of the ORS-ALgorithm is* $O(n \cdot \max\{n^2, q \log n\})$.

Proof 2. *The initialization of error terms is $O(n^3)$ as in the first DP version. We use binary heaps having $O(\log n)$ insertion and extraction operations. Then, for each of the n values of $1 \leq j \leq n$ we have two for-loops, one inserting at most n values into heap H_j (for time $O(n \log n)$), and the other extracting and inserting at most q values of H_j (for time $O(q \log n)$). Note there are never more than n elements in any heap H_j as the first loop never inserts more than n elements and the latter loop extracts one element and inserts at most one new element. This completes the proof.*

Theorem 4. *The optimal simplification $opt_{simp}(ts, h, \alpha, \beta, \gamma)$ can be computed in polynomial time for a time series ts on n points, a binary TSC h, and balancing factors α, β, γ, if there exists a constant c, such that the difference is $\geq \gamma$ between the least costly simplification that does not take robustness into account and the n^cth least costly one.*

Proof 3. *This follows from Theorems 2 and 3 by running the ORS Algorithm with $q = n^c$.*

5. Usefulness of the robust simplifications

We have implemented [2] the ORS-Algorithm and in this section we illustrate its use for explaining a Time Series Classifier. The algorithm is designed for classifiers working on univariate discrete time series with a binary classification. Moreover, we believe its use is more easily evaluated if the times series are evenly sampled, aperiodic and non-stationary. However, it is important to note that apart from the above constraints we did not want simple datasets where the binary classification is particularly easy or could be described (e.g. by ourselves) in any straightforward way. Based on these criteria we have chosen to focus on three datasets.

- From UCR [29]: Chinatown. This dataset shows the number of pedestrians on a street corner of Chinatown in Melbourne over a 24-hour time period, and classifies these into Weekend and Weekdays.
- From UCR [29]: ECG200. This dataset traces the electrical activity recorded in 96 time steps during one heartbeat and classifies the time series into a normal heartbeat and a Myocardial Infarction.
- From UCR [29]: Italy Power Demand. This dataset shows power demand in Italian households over a 24-hour time period, and classifies these into Winter (October-March) and Summer (April-September).

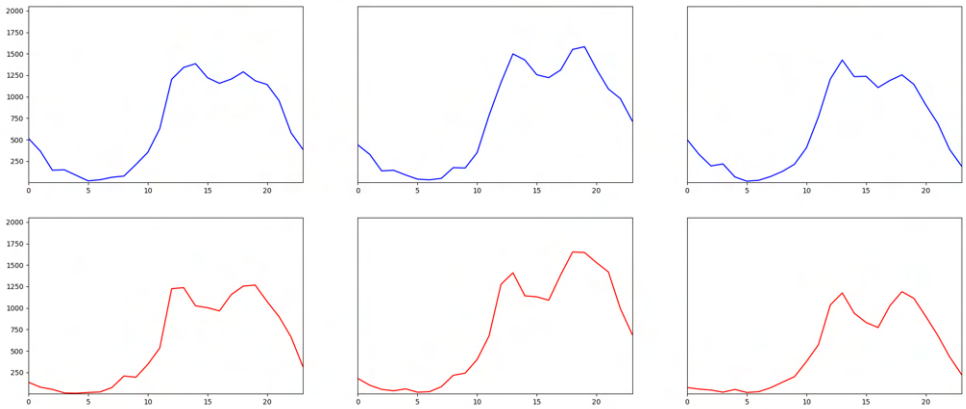


Figure 3: Six prototypes from Chinatown. Hard to make a rule-of-thumb.

[2] Available on Github: https://github.com/BrigtHaavardstun/kSimplification

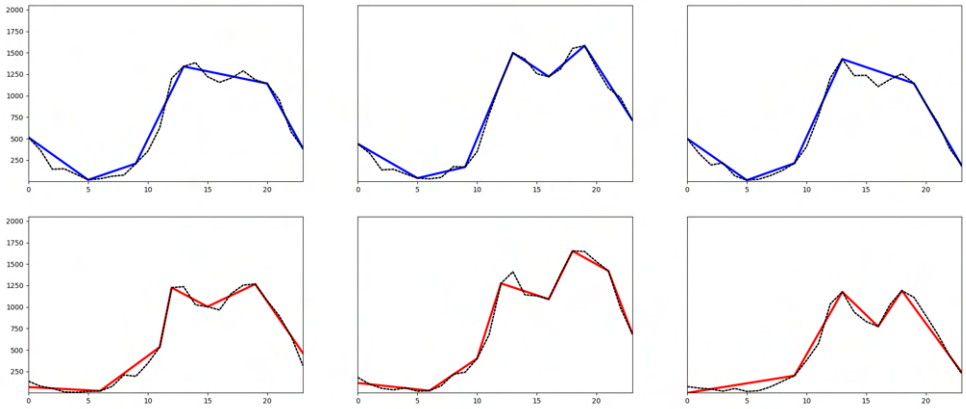


Figure 4: Same six prototypes from Chinatown, now shown in dotted black lines, with robust simplifications making it easier to extract a rule-of-thumb.

For each of these datasets we trained a fully convolutional neural network (FCN), originally proposed by Wang et al. [30]. Specifically we implemented a model closely following the work by Delaney et al. [6]. To select prototypes we used SPOTGreedy by Gurumoorthy et al. [31], implemented in the python package InterpretML[32]. We then ran the robust simplification algorithm on the resulting prototypes, with varying balancing factors, to evaluate their power as explanatory tools for the trained classifiers. In this section we focus on three aspects of the robust simplifications to argue that they are helpful explanations of the classifier model.

- Showing prototypes together with their simplifications is more informative than showing prototypes only
- Varying the balance of error vs simplicity vs robustness will highlight different aspects of the prototypes
- Extracting a rule for class membership from this information can give simple and powerful explanations

We cover these aspects in separate subsections.

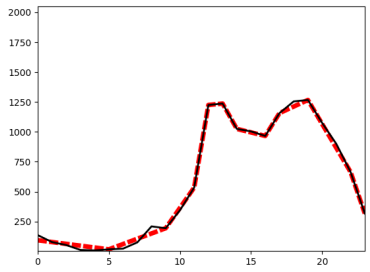
5.1. Simplifications are informative as explanations

5.1.1. Chinatown

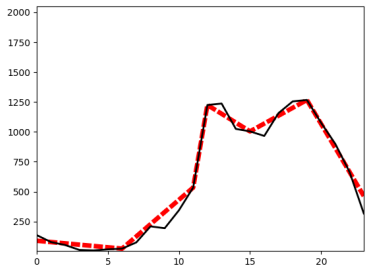
For the Chinatown dataset we have computed 3 prototypes from each of the two classes, that we call Red and Blue. These 6 prototypes are presented in Figure 3. From these prototypes only it seems difficult to identify with any kind of certainty a rule-of-thumb that can be used to decide the classification of new instances. On the next figure, Figure 4, we show a single simplification added to each of the prototypes. These simplifications are chosen with a well-balanced mix of error, simplicity and robustness, choosing each of α, β, γ in the mid-range of values [3] These simplifications function as explanations for the classification of each of the prototypes, as they also incorporate the robustness to 10.000 perturbations, and as such they allow a rule-of-thumb to be more readily guessed at. Looking at the first segment of each of the robust simplifications it is striking that the blue slopes are quite sharply downwards, whereas none of the red are. A natural guess is as follows:

- Rule-of-thumb for Chinatown classification: if the time series starts by a noticable downward slope then it is Blue, otherwise Red

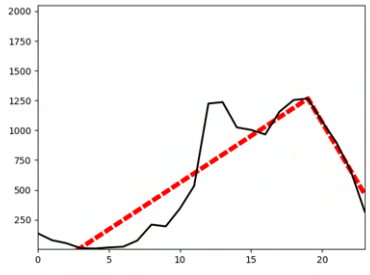
[3] Exact values used for parameters can be found on the github page. Here we only refer to them as Low-Mid-High to keep the presentation simple.



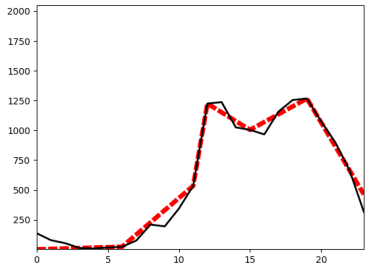
(a) 11 segs: Mid α - Low β - Mid γ



(b) 6 segs: Mid α - Mid β - Mid γ

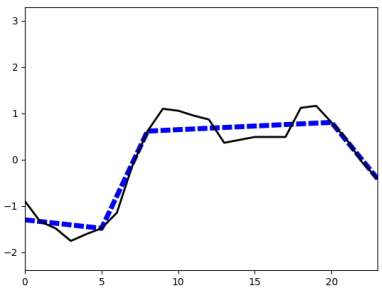


(c) 2 segs: Mid α - High β - Mid γ

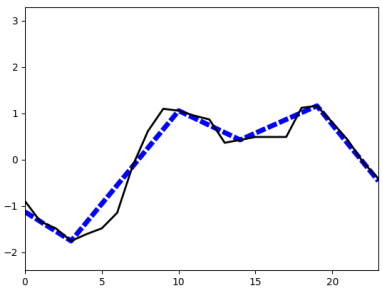


(d) 6 segs: Mid α - Mid β - High γ

Figure 5: Prototype from Chinatown in black and four distinct simplifications of it in red. From (a) to (b) to (c) we see that increasing the importance of simplicity from low to middle to high will decrease the number of segments from 11 to 6 to 2, and increase the Euclidean distance. In (d) we see that increasing the importance of robustness, when comparing to (b) which also has 6 segments, changes the slope of the first segment even though this increases the Euclidean distance, which is significant information about the red classification.



(a) 4 segs: Mid α - Mid β - Mid γ



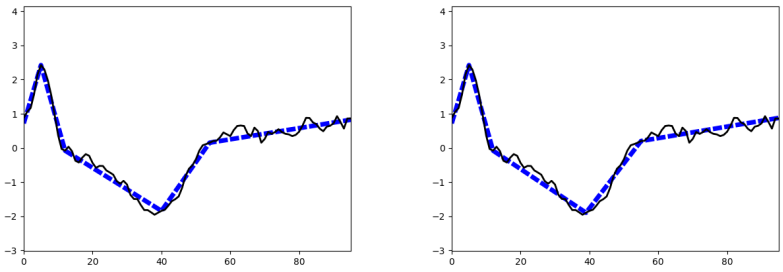
(b) 5 segs: Mid α- Mid β - High γ

Figure 6: Prototype from ItalyPowerDemand in black and two distinct simplifications of it in blue. Note that both have the same balancing factor for error and simplicity, but (a) has 4 segments while (b) has a higher factor of robustness that yields 5 segments. Simplifying, we could say that a segment in (a) has been replaced by two segments in (b) to give two peaks. This indicates that those two peaks at or around those x-values is important for the robustness of the Blue classification.

5.2. Varying balancing factors

5.2.1. Chinatown

In Figure 5 we see the effect of varying the balancing factors for a Red prototype in the Chinatown dataset. We see that increasing the importance of simplicity from low to middle to high will decrease the number of segments, while naturally also increasing the Euclidean distance to the original time series. We also see that increasing the importance of robustness changes the slope of the first segment from slightly downward (in the other 6-segment simplification) to slightly upward. This means that



(a) 5 segs: Mid α - Mid β - Mid γ **(b)** 5 segs: Mid α - Mid β - High γ

Figure 7: Prototype from ECG200 in black and two distinct simplifications of it in blue. Note that even though the factor for robustness is much higher in (b) than (a), the two simplifications are almost identical (difference being that in (a) the third segment ends in $x = 40$ whereas in (b) it ends in $x = 39$).

a higher percentage of time series with first segment having slightly downward slope in this vicinity are classified as Blue, as compared to the percentage classified Blue having slightly upward slope. This change is significant information, and it holds even in the presence of a higher Euclidean distance to the original prototype. Note that this observation constitutes supporting evidence for the rule-of-thumb guessed at previously for Chinatown.

5.2.2. Italy Power Demand

In Figure 6 we see a single prototype from ItalyPowerDemand in black and two robust simplifications of it in blue. Both simplifications have the same balancing factor for error and simplicity, but (a) has 4 segments while (b) has 5 segments resulting from an increase in the factor for robustness. Simplifying, we could say that a single segment in (a) has been replaced by two segments in (b) to give two peaks. This indicates a rule-of-thumb for the classifier trained on ItalyPowerDemand, namely that two such peaks at or around those x-values is important for the Blue classification.

5.2.3. ECG200

In Figure 7 we see a prototype from ECG200 in black and two distinct simplifications of it in blue. Note that even though the factor for robustness is much higher in (b) than (a), the two simplifications are (almost) identical, both on 5 segments. To explain this lack of significance of robustness we turn to Theorem 2 which says that the difference between the best cost and qth best cost output by Stage 1 of the ORS Algorithm, may limit the values of γ for which Stage 2 returns the optimal robust simplification. For the simplifications of Figure 7 we ran Stage 1 with the high value of $q = 1.000.000$ and found this difference to be 0.03630-0.02884=0.00746. Thus by Theorem 2 the simplification in (a) may not be optimal for that value of $\gamma = 0.01 > 0.00746$, whereas the one in (b) is likely not optimal, as $\gamma = 0.1 >> 0.00746$. Let us explain why we believe this happens. This time series has a length of 96, and note the DP to get 5 segments optimizes over all possible ways of choosing 6 points from 96. Note that there are many such choices of 6 points that give 5 segments similar to the ones in Figure 7. Counting the number of ways of choosing these 6 points, starting with the rightmost point and going left, a back-of-the-envelope calculation estimates this to be in the ballpark of 35*20*10*10*3*5 which is over 1 million. Thus, we take this as evidence that almost all the q best simplifications returned by the dynamic programming of Stage 1, that do not take robustness into account, are very close to these 5 segments. Thus increasing the importance of robustness cannot help, as the result will always be a simplification similar to the one given. We leave for future work to deal with such cases, for example by a heuristic during the DP that discards simplifications that are only marginally different from others.

5.3. Evaluating a classification rule guessed from robust simplifications

5.3.1. Chinatown

For the Chinatown classification we made a rule-of-thumb based on information gathered from the explanations of the classifications given for 6 prototypes, by means of the robust simplifications. To check if this rule-of-thumb is actually useful we need to make it more specific. Looking at the values on the y-axis for Figure 4 we see that the first segments of the 3 Blue simplifications are lines starting at point $(0, 500)$ and decreasing to about $(5, 0)$ while the first segments of the Red simplifications are only very slightly decreasing or increasing. Thus, the difference between the y-values at $x = 0$ and $x = 5$ would distinguish between these Red and Blue prototypes. We guess that this difference also distinguishes between many other time series in the dataset, as this insight is gathered from the robust simplifications. Taking the halfway point of 250 between the y-values of 500 and 0 as the cutoff the simple classification rule becomes

- Simple Rule for Chinatown: if the y-value at $x = 0$ minus the y-value at $x = 5$ is larger than 250 then ts is classified Blue, otherwise Red

The Chinatown dataset has 363 time series. We checked their classification by the model and compared to the Simple Rule. Indeed, the Simple Rule classifies all but 1 of the model-classified Blue time series as Blue, and it classifies all but 19 of the Red time series as Red, for an accuracy of 94.5 $\%$. Note this is accuracy with respect to the model, not the ground truth, as the Simple Rule is trying to explain the model. Let us mention that the accuracy of the model wrt ground truth is 96.4 %, while accuracy of the simple rule wrt ground truth is 97 %.

Finally, let us also mention that if we raise the cutoff in the Simple Rule from 250 to 325 it actually achieves an accuracy of 99.7 % (with accuracy of the simple rule vs ground truth down to 96.7%) failing on only one instance. In retrospect, we could ask if there was any information available to us from the robust simplifications that would point to this higher cutoff. In fact, consider Figure 5 (a) that visualizes the robustness of a simplification whose first segment goes from $(0, 400)$ to $(7, 50)$. Note the red part of the band around the first segment stretches up above $(0, 250)$. This should have made us wonder if the cutoff for the simple rule should be higher than 250. We could have made a test of this, for example by constructing a time series whose first segment goes from $(0, 300)$ to $(5, 0)$ and visualize its robustness. See Figure 8, where we have done exactly this, and pay close attention to the colourings of the band around the first segment. This band is predominantly blue down to somewhere above $(0, 300)$, and already then it turns red. This in itself could have made us guess a higher cutoff than 250 for the Simple Rule. We believe these insights show the usefulness of the robust simplifications and their visualization.

5.3.2. ItalyPowerDemand

In the previous subsection we saw that two peaks in the mid-afternoon seemed to signify Blue class for the classifier on the ItalyPowerDemand dataset. Looking more closely at Figure 6 we note that the difference between the high part of the peaks and the low part is about 0.5. We also note from the original prototype that the first peak is centered at 10, the second peak at 18, and that in (b) the bottom of the simplification is centered at 14. From this we extract the following rule:

- Simple Rule for ItalyPowerDemand: let $max1$ = maximum of the y-values over $x = 9, 10, 11$, let $min2$= minimum of the y-values over $x = 13, 14, 15$ and $max3$ = maximum of the y-values over $x = 17, 18, 19$. If $max1 - min2 \geq 0.5$ and $max3 - min2 \geq 0.5$ then classify this instance Blue, else Red.

The dataset ItalyPowerDemand has 1096 time series. We checked their classification by the model and compared to the Simple Rule. Indeed, the Simple Rule classifies all but 50 of the model-classified Blue correctly, and all but 49 of the Red correctly, for an accuracy of 91%. Let us mention that accuracy of the model wrt ground truth is 96.3%, and of the Simple Rule wrt Ground Truth 89.6%. Note that

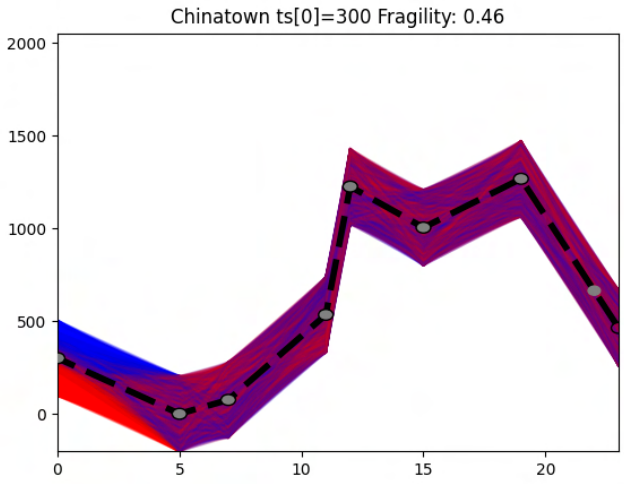


Figure 8: Visualization of robustness of a time series starting at $(0, 300)$ suggesting that a cutoff somewhat higher than $(0, 250)$ for Blue vs Red would have been a better guess for the Simple Rule for Chinatown.

classifiers of ItalyPowerDemand have previously been shown hard to explain to end-users [24]. Thus, the Simple Rule, developed by examining the robust simplifications in Figure 6, has a surprisingly high accuracy.

6. Conclusions and Future Work

In this paper, we addressed the challenge of interpretability in Time Series Classification (TSC) by introducing the Optimal Robust Simplifications (ORS) algorithm. Our approach focuses on simplifying time series data into straight-line segments while ensuring that these simplifications remain robust with respect to the classifications made by a given TSC model.

Our ORS-Algorithm is model-agnostic and allows a balance between error (fidelity with respect to the original instance), simplicity (complexity of the simplification), and robustness (fraction of local perturbations that retain the correct classification). We prove that the ORS-Algorithm, under certain mild conditions, finds the optimal solution in polynomial time, by a first dynamic programming stage solving for error and simplicity, and then incorporating robustness into the solution. This simplification process aids in making time series data more intuitive and interpretable for humans.

Our experimental results, based on UCR datasets Chinatown, Italy Power Demand, and ECG200, confirm the practical utility of our approach. The robust simplifications provided by the ORS algorithm make time series data more accessible to human intuition, facilitating better user understanding and trust in the TSC models. In conclusion, our work contributes a novel method for enhancing the interpretability of TSC models through robust simplifications.

Future research could explore other ways to compute the factors employed in the algorithm. For instance, the squared Euclidean distance could be replaced by some other distance function, or simplicity can depend on other aspects than only the number of segments, and robustness can be computed by other types of perturbations. Finally, we plan to conduct experiments including human studies to demonstrate empirically the benefits of using optimal robust simplifications to explain TSC models and compare to related XAI methods.

References

[1] A. Rajkomar, E. Oren, K. Chen, A. M. Dai, N. Hajaj, M. Hardt, P. J. Liu, X. Liu, J. Marcus, M. Sun, et al., Scalable and accurate deep learning with electronic health records, NPJ digital medicine 1

(2018) 1–10.
[2] G. A. Susto, A. Cenedese, M. Terzi, Time-series classification methods: Review and applications to power systems data, Big data application in power systems (2018) 179–220.
[3] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data mining and knowledge discovery 33 (2019) 917–963.
[4] M. Guillemé, V. Masson, L. Rozé, A. Termier, Agnostic local explanation for time series classification, in: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019, Portland, OR, USA, November 4-6, 2019, IEEE, 2019, pp. 432–439. URL: https://doi.org/10.1109/ICTAI.2019.00067. doi:10.1109/ICTAI.2019.00067.
[5] A. A. Ismail, M. K. Gunady, H. C. Bravo, S. Feizi, Benchmarking deep learning interpretability in time series predictions, in: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual, 2020.
[6] E. Delaney, D. Greene, M. T. Keane, Instance-based counterfactual explanations for time series classification, in: Case-Based Reasoning Research and Development - Int. Conference, ICCBR 2021, Springer, 2021, pp. 32–47.
[7] U. Schlegel, H. Arnout, M. El-Assady, D. Oelke, D. A. Keim, Towards a rigorous evaluation of xai methods on time series, in: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), IEEE, 2019, pp. 4197–4201.
[8] S. A. Siddiqui, D. Mercier, M. Munir, A. Dengel, S. Ahmed, Tsviz: Demystification of deep learning models for time-series analysis, IEEE Access 7 (2019) 67027–67040. URL: https://doi.org/10.1109/ACCESS.2019.2912823. doi:10.1109/ACCESS.2019.2912823.
[9] B. Håvardstun, C. Ferri, J. A. Telle, An interactive tool for interpretability of time series classification, in: European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2024. To be published.
[10] A. Theissler, F. Spinnato, U. Schlegel, R. Guidotti, Explainable AI for time series classification: A review, taxonomy and research directions, IEEE Access 10 (2022) 100700–100724.
[11] H. Meng, C. Wagner, I. Triguero, Explaining time series classifiers through meaningful perturbation and optimisation, Information Sciences 645 (2023) 119334. URL: https://www.sciencedirect.com/science/article/pii/S0020025523009192. doi:https://doi.org/10.1016/j.ins.2023.119334.
[12] J. Enguehard, Learning perturbations to explain time series predictions, in: International Conference on Machine Learning, PMLR, 2023, pp. 9329–9342.
[13] Z. Liu, Y. ZHANG, T. Wang, Z. Wang, D. Luo, M. Du, M. Wu, Y. Wang, C. Chen, L. Fan, Q. Wen, Explaining time series via contrastive and locally sparse perturbations, in: The Twelfth International Conference on Learning Representations, 2024. URL: https://openreview.net/forum?id=qDdSRaOiyb.
[14] T. Rojat, R. Puget, D. Filliat, J. Del Ser, R. Gelin, N. Díaz-Rodríguez, Explainable artificial intelligence (xai) on timeseries data: A survey, arXiv preprint arXiv:2104.00950 (2021).
[15] C. Obermair, A. Fuchs, F. Pernkopf, L. Felsberger, A. Apollonio, D. Wollmann, Example or prototype? learning concept-based explanations in time-series, in: Asian Conference on Machine Learning, PMLR, 2023, pp. 816–831.
[16] Z. Geler, V. Kurbalija, M. Ivanović, M. Radovanović, Weighted kNN and constrained elastic distances for time-series classification, Expert Systems with Applications 162 (2020) 113829.
[17] A. H. Gee, D. García-Olano, J. Ghosh, D. Paydarfar, Explaining deep classification of time-series data with learned prototypes, 2019. URL: https://ceur-ws.org/Vol-2429/paper3.pdf.
[18] E. Keogh, K. Chakrabarti, M. Pazzani, S. Mehrotra, Dimensionality reduction for fast similarity search in large time series databases, Knowledge and information Systems 3 (2001) 263–286.
[19] E. Keogh, S. Chu, D. Hart, M. Pazzani, Segmenting time series: A survey and novel approach, in: Data mining in time series databases, World Scientific, 2004, pp. 1–21.
[20] E. Camponogara, L. F. Nazari, Models and algorithms for optimal piecewise-linear function approximation, Mathematical Problems in Engineering 2015 (2015) 876862.
[21] U. Schlegel, D. V. Lam, D. A. Keim, D. Seebacher, Ts-mule: Local interpretable model-agnostic

explanations for time series forecast models, 2021. arXiv:2109.08438.
[22] Y. Wan, X. Gong, Y.-W. Si, Effect of segmentation on financial time series pattern matching, Applied Soft Computing 38 (2016) 346–359. URL: https://www.sciencedirect.com/science/article/pii/S1568494615006341. doi:https://doi.org/10.1016/j.asoc.2015.10.012.
[23] Y.-W. Si, J. Yin, Obst-based segmentation approach to financial time series, Engineering Applications of Artificial Intelligence 26 (2013) 2581–2596. URL: https://www.sciencedirect.com/science/article/pii/S0952197613001723. doi:https://doi.org/10.1016/j.engappai.2013.08.015.
[24] B. Håvardstun, C. Ferri, K. Flikka, J. A. Telle, XAI for time series classification: Evaluating the benefits of model inspection for end-users, in: Explainable Artificial Intelligence, 2024. URL: https://xaiworldconference.com/2024/, to be published.
[25] Z. J. Wang, J. W. Vaughan, R. Caruana, D. H. Chau, GAM coach: Towards interactive and user-centered algorithmic recourse, in: Proc. Conf. on Human Factors in Computing Systems„ ACM, 2023, pp. 835:1–835:20.
[26] J. Höllig, C. Kulbach, S. Thoma, Tsinterpret: A python package for the interpretability of time series classification, J. Open Source Softw. 8 (2023) 5220.
[27] W. Tang, L. Liu, G. Long, Interpretable time-series classification on few-shot samples, in: 2020 International Joint Conference on Neural Networks (IJCNN), IEEE, 2020, pp. 1–8.
[28] Y. Ming, P. Xu, H. Qu, L. Ren, Interpretable and steerable sequence learning via prototypes, in: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, 2019, pp. 903–913.
[29] H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, Hexagon-ML, The UCR time series classification archive, 2018.
[30] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, CoRR abs/1611.06455 (2016). URL: http://arxiv.org/abs/1611.06455. arXiv:1611.06455.
[31] K. S. Gurumoorthy, P. Jawanpuria, B. Mishra, Spot: A framework for selection of prototypes using optimal transport, 2021. arXiv:2103.10159.
[32] I. Contributors, Interpretml: Fit interpretable models. explain blackbox machine learning., GitHub repository (2023). URL: https://github.com/interpretml/interpret.

7. Appendix

7.1. Pseudo code for DP algorithm

Define a 2-dimensional DP table $D[j, p]$ that stores the pth least costly simplification for the points $p_1, ..., p_j$ ending at p_j. Set $D[1, 1] = 0$.
Compute all $D[j, p]$ for $p \geq 1$ as follows:
for $j = 1$ to $n - 1$ **do**
    Initialize an empty heap H_j to store triples (value, index, rank) with value being the key.
    for $i = 1$ to $j - 1$ **do**
        H_j.add$(\alpha \cdot err(i, j) + \beta + D[i, 1], i, 1)$
        if $i \neq 1$ **then**
            H_j.add$(\alpha \cdot err(1, i, j) + \beta, i, -1)$
        end if
    end for
    for $p = 1$ to q **do**
        $(v_m, index_m, rank_m) = pop_{min}(H_j)$
        $D[j, p] = v_m$
        if $rank_m \neq -1$ **then**
            H_j.add$(\alpha \cdot err(index_m, j) + \beta + D[index_m, rank_m + 1], index_m, rank_m + 1)$
        end if
    end for
end for
To ensure that the last segment is not required to stop at p_n, but instead can be a continuation of two other points p_i and p_j, go over all pairs of points and evaluate the error if we continue them to the end.
Store these possibilities on a heap H_n and extract the q best, similar to what we did previously.
Initialize an empty heap H_n to store quadruples (value, index, rank, end) with value being the key.
for $j = 1$ to n **do**
    for $i = 1$ to $j - 1$ **do**
        H_n.add$(\alpha \cdot err(i, j, n) + \beta + D[i, 1], i, 1, j)$
        if $i \neq 1$ **then**
            H_n.add$(\alpha \cdot err(1, i, j, n) + \beta, i, -1, j)$
        end if
    end for
end for
for $p = 1$ to q **do**
    $(v_m, index_m, rank_m, end_m) = pop_{min}(H_n)$
    $D[n, p] = v_m$
    if $rank_m \neq -1$ **then**
        H_n.add$(\alpha \cdot err(index_m, end_m, n) + \beta + D[index_m, rank_m + 1], index_m, rank_m + 1, end_m)$
    end if
end for

Paper VI

Håvardstun, B., Marti-Perez, F., Ferri, C., & Telle, J. A.
To appear in *Proceedings of the World Conference on Explainable Artificial Intelligence (xAI 2026)*. Author's accepted manuscript.
URL: https://arxiv.org/abs/2505.08846

Evaluating Simplification Algorithms for Interpretability of Time Series Classification

Brigt Håvardstun[1][0009−0007−1034−1422], Félix Martí-Perez[2][0009−0003−5736−3523], Cèsar Ferri[2][0000−0002−8975−1120], and Jan Arne Telle[1][0000−0002−9429−5377]

[1] Department of Informatics, University of Bergen, Bergen, Norway
{brigt.havardstun, jan.arne.telle}@uib.no
[2] VRAIN, Universitat Politècnica de València, València, Spain
fmarper@inf.upv.es, cferri@dsic.upv.es

Abstract. In this work, we introduce metrics to evaluate the use of simplified time series in the context of interpretability of a TSC - a Time Series Classifier. Such simplifications are important because time series data, in contrast to text and image data, are not intuitively understandable to humans. These metrics are related to the *complexity* of the simplifications - how many segments they contain - and to their *loyalty* - how likely they are to maintain the classification of the original time series. We focus on simplifications that select a subset of the original data points, and show that these typically have high Shapley value, thereby aiding interpretability. We employ these metrics to experimentally evaluate four distinct simplification algorithms, across several TSC algorithms and across datasets of varying characteristics, from seasonal or stationary to short or long. We subsequently perform a human-grounded evaluation with forward simulation, that confirms also the practical utility of the introduced metrics to evaluate the use of simplifications in the context of interpretability of TSC. Our findings are summarized in a framework for deciding, for a given TSC, if the various simplifications are likely to aid in its interpretability.

Keywords: Interpretability · Time Series Classification · Simplifications

1 Introduction

Temporal data is encountered in many real-world applications ranging from patient data in healthcare [27] to the field of cyber security [34]. Deep learning methods have been successful for TSC - Time Series Classification - [19,27,34] but such methods are not easily interpretable, and often viewed as black boxes, which limits their applications when user trust in the decision process is crucial. To enable the analysis of these black-box models we revert to post-hoc interpretability. Recent research has focused on adapting existing methods to time series, where we can find specific methods like SHAP-LIME [12], Saliency Methods [18], and Counterfactuals [5,**?**], and also combinations of these [30].

As humans learn and reason by forming mental representations of concepts based on examples, and any machine learning model has been trained on data,

then we believe that data e.g. in the form of prototypes and counterfactuals is indeed the natural common language between the user and this model. However, the basic problem is that compared to images and text, time series data are not intuitively understandable to humans [33]. This makes interpretability of time series extra demanding, both when it comes to understanding how users will react to the provided explanations and to predict what explanatory tools are best. For example, in [13] a tool was given for explainability of a TSC, that allowed model inspection so the user could form their own mental model of the classifier. However, a user evaluation showed that non-expert end-users were not able to make use of this freedom, supporting the notion that time-series are particularly non-intuitive for humans.

In [37], authors give an intriguing taxonomy of XAI for TSC, divided into those focused on (i) Time-points (e.g. SHAP and LIME) or (ii) Subsequences (e.g. Shapelets) or (iii) Instances (Prototypes, CF, Feature-based). Explaining a classifier by instances like prototypes has a definite appeal, but it is a challenge how to highlight features important for the classification, while at the same time simplifying the instance to mitigate the non-intuitive aspects of this domain. In [37] the authors also review the literature on XAI for TSC and of the 9 papers they discuss on Instance-based explanations using Prototypes or Features, it is remarkable that not a single one is specialised for Local explanations, rather they are all Global. The algorithms we evaluate in this paper fill this gap, as they are ways of *simplifying* a time series by straight-line segments[3]. We focus on Piecewise-linear sketches since they keep every retained point on the true time-and-value axes, so users can read magnitudes and slopes directly, without translating bins or symbols. This matches the canonical line-chart format that dashboards and decades of perception studies show that people understand quickly and accurately [2,14].

We introduce metrics to evaluate the use of algorithms computing simplified time series in the context of interpretability of a TSC. This addresses the key challenge that time series data, in contrast to text and image data, are not intuitively understandable to humans. These metrics are related to the *complexity* of the simplifications produced by the algorithms - how many data points they maintain as a percentage of the original - and to their *loyalty* - how likely they are to maintain the classification of the original time series. We first employ these metrics to theoretically and experimentally evaluate four distinct simplification algorithms, across several TSC algorithms and across datasets of varying characteristics, from seasonal or stationary to short or long. We also employ the metrics in a practical setting, doing a quantitative user evaluation, thereby meeting a demand from the XAI research community [5,37,40], to measure how simplifications increases the understanding that the user has of the black box classification. Using the taxonomy of interpretability evaluation from [8], what we do is a Human-Grounded Evaluation with Forward Simulation, to evaluate

[3] Piecing together several such local explanations, say for prototypes of each class, this can still form a global explanation of the given TSC.

if simplifications are better than full time series, when using prototypes to teach the classification used by a TSC.

Our findings suggest that using simplifications for interpretability of TSC is in some cases much better than using the original time series. However, we discover various cases where the trade-off between complexity and loyalty, or some other characteristics of the given dataset and TSC, are such that simplifications are not very useful. This happens for example when the original time series have several data points lying on an almost straight line, in which case the human eye seems to perform the simplification automatically, and the user interprets the TSC equally well from original time series as from loyal simplifications. In short, our findings suggest that simplifications can be useful when i) the TSC is not itself easily interpretable on the original domain, ii) the simplifications are loyal but not complex and dissimilar to the original, and moreover iii) the simplified domain is understandable under labels from TSC on the original domain.

In the rest of this paper we first discuss related work and cover some standard definitions before presenting the metrics we will use, the four simplification algorithms we will evaluate, the TSCs we will employ and the 40 UCR datasets we use with their varying characteristics. We then show the results of our theoretical experiments, first comparing the four algorithms over a variety of dataset characteristics, and then focusing on the best performing algorithms and evaluating how useful they may be for interpretability. We then report on the user study, and based on the lessons learned we give a framework for deciding if simplifications are likely to aid in interpretability of a given TSC.

2 Related Work

Several techniques have been applied to generate explanations from TSC models [37,29]. Specifically, techniques previously used on Convolutional Neural Networks or Recurrent Neural Networks have been applied for TSC. For instance, the authors in [30] apply to time series several XAI methods previously used on image and text domain. They also introduce verification techniques specific to times series, in particular a perturbation analysis and a sequence evaluation.

Another alternative is to produce explanations through examples, and these can be specifically utilised in the time series domain. One type of this explanation method involves giving the nearest example from the training dataset that acts as a prototype to depict the normal behaviour of a similar sample [11]. A method to generate prototypes for time series data using an autoencoder is presented in [10]. The main novelty of the work is the method to generate diverse prototypes.

Given that in many cases raw time series can be too complex for humans, several studies have tried to employ simplified versions as explanations. In [21], the authors propose a dimensionality reduction technique for time series, called Piecewise Aggregate Approximation, as a tool for similarity search in large time series databases. In [22], a comprehensive survey on time series segmentation, the authors highlight that the optimal number of segments often depends on the underlying patterns in the data and the specific objectives of the analysis. Also,

in [1], the authors introduce a range of piecewise-linear models and algorithms for unknown functions. Another option is to segment time series into fixed-width windows and employ these to justify decisions. In [31], the authors propose TS-MULE, a model explanation method working to segment and perturb time series data and extending LIME. A study on the effect of segmentation on time series, in particular in the field of finance for the use of pattern matching was presented in [39]. In [32], the researchers also apply segmentation to financial time series data, now as a preprocessing step to locate technical patterns. Similarly, in [42], the authors employ financial time series segmentation and introduce the Optimal Multi-Segment Linear Regression (OMSLR) algorithm. Their approach aims to minimise the global mean square error (between the simplification and the original time series) for a given number of segments k.

3 Simplifications and Metrics

We first recall basic notions and then introduce the simplifications and metrics that we will study in this paper.

Staying consistent with earlier notation [37,5] a time series $T = \{t_1, t_2, \ldots, t_n\}$ is an ordered set of n real-valued observations (or time steps). Note we may also view each $t_i = (x_i, y_i)$ as a pair consisting of an x-value (the time) and a y-value (the observation), where $x_i < x_{i+1}$, and often $x_i = i$. A time series dataset $D = \{(T_1, c_1), (T_2, c_2), ..., (T_m, c_m)\}$ is a collection of such time series where each time series $T_i \in R^m$ has a class label c_i. Thus, for binary classification tasks we would have $c_i \in \{0, 1\}$. Given such a dataset D, Time Series Classification (TSC) is the task of training a mapping C_D from the space of possible inputs to a predicted class. Interpretability of TSC concerns giving a human user an understanding of how such a trained TSC decides on a predicted class. Exemplar-based interpretability employs showing examples of classifications made by the TSC, and this will allow the user to form their own mental model of the classifier. Prototypes are time series exemplifying the main aspects responsible for a classifier's specific decision outcome. Showing prototypes is well-known in exemplar-based interpretability.

For TSC interpretability, in particular for long time series, such prototypes often have too much information, and we therefore focus on what we call simplifications. Such simplifications have been common practice in data mining and other related fields, and hence a wide range of simplification techniques have been constructed to suit different needs. In our work we focus on trustworthy and faithful simplifications. In the Experiments and Results section, we empirically show that simplifications that select a subset of original time steps as their simplifications are more faithful to important time steps.

Hence for the purposes of this paper a *simplification* of a time series $ts = \{t_1, t_2, \ldots, t_n\}$ on $|ts| = n$ time steps consists of dropping some of the time steps, thus keeping only time steps at a subset $S \subseteq \{1, 2, \ldots, n\}$ of time points and replacing the rest by the points lying on the straight-line segments given by the kept points, thus resulting in a new time series $sts = \{t'_1, t'_2, \ldots, t'_n\}$ with

$t'_i = t_i = (x_i, y_i)$ for $i \in S$ and otherwise $t'_i = (x_i, y'_i)$ with y'_i the value at x_i of the straight line between t_j and t_k for $j < k$ and $\{j, j+1, ..., k\} \cap S = \{j, k\}$. We will say that the simplification sts consists of $|S| - 1$ segments, and we denote its number of segments by $||sts||$. The selection strategy of the specific subset of time steps depends on the algorithm and parameters used. See in Figure 1 an example showing a simplification with 4 segments obtained by keeping 5 time steps (at times 0,9,13,19,23) of an original time series on 24 time steps (at times 0 to 23).

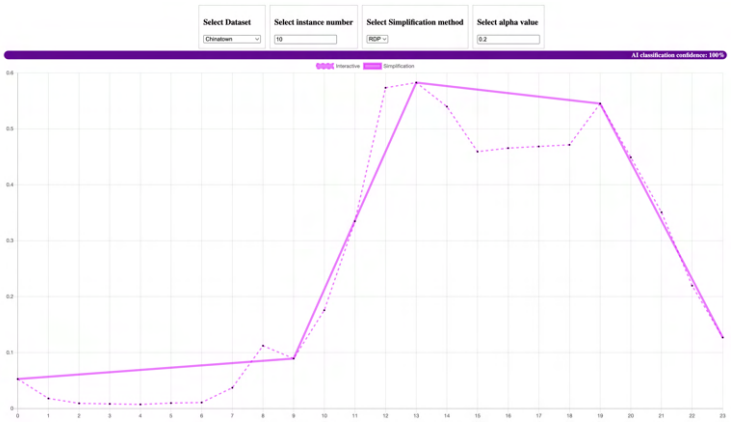


Fig. 1. An original time series (dotted line) and the simplification on 4 segments produced by RDP with parameter $\alpha = 0.2$ (used to set the level of simplification)

.

A simplification algorithm for time series called A takes as input an original time series ts and outputs a simplification $A(ts)$ on $||A(ts)||$ segments. Note that the length of $A(ts)$ is the same as the length of ts so a classifier trained for time series of that length can be used on the simplification $A(ts)$.

Definition 1. *To evaluate the usefulness of a simplification algorithm A for interpretability of a TSC C_D, we introduce the following metrics. Let P be a representative subset of inputs, chosen randomly according to the expected distribution of inputs to C_D, and let these original inputs have length n:*

- *Complexity: :* $\frac{1}{n}\frac{1}{|P|}\sum_{ts \in P} ||A(ts)|| + 1$*, the expected number of time points kept in the simplification, as a fraction of the length of the original*
- *Loyalty:* $\frac{1}{|P|}|\{ts \in P : C_D(ts) = C_D(A(ts))\}|$*, the expected probability that the simplification has the same classification as the original*
- *Kappa loyalty:* $\frac{p_0 - p_e}{1 - p_e}$ *where* $p_0 = \frac{1}{|P|}\sum_{ts \in P} 1[C_D(ts) = C_D(A(ts))]$ *is the relative observed agreement, and* $p_e = \sum_{c \in C} P_{CD}(c) \cdot P_{CD(AC)}(c)$ *is the hypothetical probability of chance agreement*

Thus complexity is a value between $\frac{2}{n}$ (when all simplifications of instances in P are on a single segment) and 1 (when all simplifications are the trivial one which does not remove any time step from the original), while loyalty is a value between 0 (when all simplifications are classified by C_D as different from the original ones) and 1 (when complete agreement, all simplifications are classified the same as the original ones). The Kappa loyalty [3] will have a maximum value of 1 when we have complete agreement, a value of 0 if the agreement is no better than chance, and below 0 if a tendency for different classes. Kappa was used as a performance metric as it accounts for agreement occurring by chance, providing a more robust measure than accuracy, particularly in the presence of class imbalance. Note that when using a simplification algorithm for interpretability of TSC, showing the simplifications to a user, then low complexity should allow a more easy focus on important features. Regarding loyalty, let us first emphasize that what we intend to teach the user are the patterns discovered by the TSC in classifying original time series. If we show the user simplified time series with Kappa loyalty 1 then nothing has been lost, since whatever the user learns from the simplifications carries over to the original. Note that we can expect this to happen for some complexity below 1, when all such simplifications retain the classification of the originals. However, if we show the user simplifications with loyalty $\lambda < 1$ then what the user learns does not fully carry over to the original, and we will take this into account when evaluating its utility. When applying simplifications we are thus searching for a sweet-spot between high loyalty and low complexity, to maximize the users ability to learn the patterns actually used by the TSC.

4 Simplification Algorithms

We briefly describe the four simplification algorithms we have chosen to focus on. Although they are quite distinct, they share a number of properties, which will allow for a fair comparison:

- their input consists of an original time series, given by n points
- their output is a simplification given by a subset of input points
- they have a parameter α and increasing its value will monotonically decrease number of segments in output, from $n-1$ to 1.

4.1 Ramer-Douglas-Peuker

RDP is a recursive algorithm introduced in [28,9]. Given n points $p_1, ..., p_n$ select a point p_d with the largest distance to the straight line between the first point p_1 and last point p_n. If the distance to p_d is no more than α, then we are in a base case and return the two points p_1 and p_n. If the distance to p_d exceeds α, then recursively process the two sequences of points formed by $p_1, ..., p_d$ and $p_d, ..., p_n$. The implementation ([15]) used for RDP has runtime $O(n \log n)$.

4.2 Visvalingam–Whyatt algorithm

VW is an iterative algorithm introduced in [38]. It computes areas of triangles formed by successive triples of points a, b, c; in each iteration, the middle point b with the smallest associated triangle is removed, the area of neighbouring triangles is recomputed, and the process is repeated. The process stops when the area of the smallest triangle exceeds α. The implementation of VW we use [17] has runtime $O(n \log n)$.

4.3 Bottom-Up

BU is based on an iterative algorithm as described in [20] with the same name. We slightly modified the algorithm from [20] so that it adheres to the simplification algorithms we study here - selecting subsets of original time steps. Initially each time point is viewed as a trivial seg (a special segment), and in each iteration we merge two consecutive segs A and B (if A ends in time step k then B starts in time step $k+1$) with the merged seg AB represented by a straight line going from the start of A to the end of B (in original BU the segment AB would be represented by the Least Square Fit over this area). The two segs merged into some AB in any iteration should have lowest error, which is the sum over each time point t within AB, of the absolute value of the difference between the value of the original time series at t and the value of the AB line at t. We stop when the lowest error exceeds α, with s segs $S_1, S_2, ..., S_s$ such that if S_i ends in k then S_{i+1} starts in $k+1$, and we return the corresponding simplified time series of length $2s$. While the resulting simplification has only s segs, if we add the gaps between consecutive segs we get $2s-1$ ordinary segments. Our implementation of BU has runtime $O(n \log n)$.

4.4 Optimal Simplification

OS is a dynamic programming algorithm introduced in [36]. It finds the piecewise linear simplification (extending the first and last of its segments to the start and end respectively) that minimizes the sum of: Squared Euclidean Distance to the original time series plus α times the number of segments used. Our implementation of OS has runtime $O(n^3)$. Note that, in contrast to the other three algorithms, it finds the optimal simplification for a given α, rather than just being a greedy algorithm, however this leads to it being more time consuming.

4.5 Normalized complexity parameter

To ensure uniformity we alter each of the four simplification algorithms so that it has a normalized complexity parameter called α_c. This parameter will range between a minimum of 0 (corresponding to a high value of α giving a small number of segments as output) and a maximum of 1 (corresponding to a low value of α giving a high number of segments as output).

5 Datasets and Classifiers

5.1 Datasets Selection

The UCR repository [4] contains 128 datasets for univariate TSC. Their features cover problems from binary up to 52-class classification, with time series lengths ranging from 15 to 2,700 data points. Given that the datasets come from a wide range of domains, such as sensors, medical devices, or traffic, the characteristics of these datasets are diverse. Most datasets come with predefined splits for training, and test sets.

To keep runtime manageable, we included only UCR datasets with time series of length less than 200 data points. This has resulted in 40 datasets of varying length, class and domain. See Table 4 for details on the 40 chosen datasets. Additionally, we categorize each dataset by stationarity, seasonality, and entropy, enabling in-depth analysis of how different simplification algorithms perform under varying data characteristics.

5.2 Feature Characterisation

The Augmented Dickey Fuller (ADF) test [7] is a statistical test used to determine the presence of a unit root in the series, thus determining if a time series is non-stationary. We set the significance level at 0.05 and apply the test to each instance: if the p-value is below it, we reject the null hypothesis that a unit root is present in the time series and label that instance "stationary"; otherwise, we label it "non-stationary." At the dataset level, we declare a dataset "stationary" if at least 80% of its instances are stationary (supermajority), "non-stationary" if at most 50% are stationary (simple-majority), and "partially-stationary" when the proportion lies between these thresholds [41].

The autocorrelation function (ACF) measures the correlation of a time series with a lagged version of itself. We utilise an extension that uses Fast Fourier transform (FFT) convolution to make ACF faster and more suitable for longer time series. A dataset is considered to be seasonal if most of its instances have a correlation coefficient greater than 0.4, thus being seasonal. This cut-off was decided empirically by visually inspecting the time series.

Approximate entropy [26] is a technique used to quantify the unpredictability of fluctuations in time series. This returns a continuous value between 0 (low entropy) and 1 (high entropy), which will determine the entropy of the time series. To obtain the entropy of a dataset, we obtain the mean entropy over all instances. To facilitate the analysis, we discretize the entropy scores into three equal-frequency bins. We determine the breakpoints to be 0.23 and 0.33. Instances with entropy up to 0.23 are labelled as low entropy, those between 0.23 and 0.33 as medium entropy, and those above 0.33 as high entropy.

5.3 Classifiers

Different types of classification algorithms are used for TSC. Throughout the literature, the most common algorithms are:

- Logistic Regression: A linear model that estimates the probability of each class using weighted input features; it is simple, interpretable (white-box), and best suited for linearly separable time series. We use scikit-learn's LogisticRegression [25] with its default L2 penalty and solver.
- Decision Trees: A non-linear, rule-based model (white-box) that recursively partitions the feature space to make decisions. Capable of handling complex relationships and capturing non-linearities in time series, but prone to overfitting. Implemented via scikit-learn's DecisionTreeClassifier [25] using the Gini impurity criterion and no maximum-depth constraint.
- k-Nearest Neighbour (kNN): A non-parametric method that classifies a time series based on the majority label among its closest examples in the training set. It can be interpretable (white-box), but its decisions can become less transparent with higher dimensionality. We leverage tslearn's KNeighborsTimeSeriesClassifier [35], configured with k = 5 and distance-weighted voting.
- Convolutional Neural Networks (CNN): A deep learning architecture that automatically extracts and learns hierarchical features through convolutional filters is highly effective for discovering patterns in time series data. However, it is considered a black-box due to its complex and opaque decision processes. Our networks are built in PyTorch consisting of three 1D convolutional layers, a global pooling layer, and a final fully-connected classification layer; trained with the Adam optimizer and cross-entropy loss.
- Linear Classifiers using Random Convolutional Kernels (Rocket): Generating a large number of random convolutional kernels to transform time series and then using the transformed features to train a linear classifier, has been shown to achieve state-of-the-art accuracy for TSC. The Rocket [6] method is in addition significantly faster than any other method of comparable accuracy. We use the RocketClassifier implemented in Sktime[23]

We first check all five classification algorithms to see which is most suited. Prior to training, we normalize both training and test sets. For datasets that do not have a validation set, we split off 20% of the training data and use it for validation. We then fit each model on the normalized training data using default hyperparameters, and validate its performance using the validation set. We evaluate these five widely-used TSC algorithms, which span the spectrum from simple, interpretable white-box models to black-box deep learners. The accuracy on the validation set shows that Rocket emerges as the best-performing model across our benchmark. Interpretability of Rocket for TSC is a challenge because it uses thousands of random convolutional kernels making it difficult to understand which features are truly important. Hence we view Rocket as a black-box model where interpretability is key, and focus exclusively on it in the rest of the paper, as it is a state-of-the-art TSC and interesting for applying simplifications.

6 Experiments and Results

It follows from the previous discussion (see Simplifications and Metrics section), where we introduced our metrics, that for interpretability of a classifier C_D we are seeking simplification algorithms that have low complexity and yet at the same time high loyalty, or rather high kappa loyalty taking into account imbalances in the distribution between classes.

We start this section with a small experiment[4] showing that simplifications that use a subset of the original time points as pivot points, rather than introducing pivot points that do not exist in the original time series, are more faithful and will improve trustworthiness, as the pivot points retained are often high-impact time steps. In the following part of this section, we then investigate which of the four simplification algorithms have the best complexity versus loyalty performance, and we will do this over a variety of characteristics of dataset, while focusing exclusively on the Rocket classifiers (see Datasets and Classifiers section). In the last part of this section, we focus on the simplification algorithms having the best performance and show that the actual values we get for loyalty versus complexity are very good.

6.1 Faithfulness of the simplification algorithms

We conduct a small experiment comparing the four simplification algorithms RDP, OS, VW, BU with the two algorithms SLS (Segmented Least Squares) and PAA (Piecewise Aggregate Approximation) - that are not restricted to selecting a subset of original data points - to see how faithful they are in retaining as pivots those data points that are most important for the classification. SLS is a Piecewise Linear Approximation, but it uses the least square method to find the straight line minimizing Euclidean distance over a segment which may lead it to create new y-values for the pivot points, instead of selecting subsets of the original data points. PAA creates fixed sized windows, and replaces this area with the aggregate mean which again may lead to not choosing original data points.

It is well known that Shapley values accurately describe how important each data point is for classification of a time series, and so Shapley values are our focus. We measure two things: - how often the data point with maximum Shapley value is chosen as pivot, and - how many data points have higher Shapley value than the chosen pivot point of highest Shapley value. Our experiment focused on three datasets of different lengths Chinatown (length 24), SonyAIBORobotSurface1 (length 70) and ECG200 (length 96), using the Rocket classifier. For a fair comparison we ensure that all six algorithms output approximately the same number of segments (7 out of 24 for Chinatown, 9 of 70 for Sony, and 12 of 96 for ECG200). For each dataset and each algorithm, we iterate over each of

[4] Code for this and later experiment can be found at `https://github.com/femartip/XAI_time_series`

the 100 selected time series in the dataset and collect the data necessary for our observations, using KernelExplainer[24] to compute Shapley values.

As shown in Table 1, RDP achieves the highest average performance, selecting the data point of highest Shapley value 28% of the time, and on expectation there will be 4 data points having higher Shapley value than the highest chosen. As expected, the four simplification algorithms in our study significantly outperform the two alternative methods that are not restricted to picking original data points. Indeed, SLS selects the most important data point only 10% of the time, while PAA fails to do this entirely. However, it happens over 20% of the time that SLS picks a pivot point at a time step x where the (x, y) data point has highest Shapley value, with SLS using some (x, y') as pivot. In these cases, the average error of SLS in the y-coordinate is not more than 0.02, which may often be acceptable. For PAA the similar values are not quite so good, being 10% and 0.94, respectively. We conclude that our choice of looking at simplification algorithms that retain original data points is sound when viewed from the perspective of wanting to include pivot points important for the classification.

Table 1. Assuming (x, y) has highest Shapley value, MAX SHAP gives expectation that (x, y) is selected as pivot, while CorrectX is expectation that (x, y') is a pivot for any y', and Dist gives expected deviation of such y' from y, thus lower is better. Rank shows Shapley rank of the pivot with highest Shapley value, thus low value is better.

Algo	MAX SHAP ↑	CorrectX ↑	Dist ↓	Rank ↓
RDP	**28.0**%	**28.0**%	0.0	4.68
OS	24.0%	24.0%	0.0	**4.24**
VW	20.0%	20.0%	0.0	4.78
BU	20.0%	20.0%	0.0	4.78
SLS	10.0%	20.0%	0.02	8.0
PAA	0.0%	10.0%	0.94	10.7

6.2 Comparing four algorithms

For the main experiments we have the 4 simplification algorithms BU, OS, VW and RDP with normalized complexity parameter α_c, and we also have a Rocket classifier C_D for each of the 40 normalized time series datasets D. The Rocket classifier achieves an average accuracy of 88% across all datasets in the test sets. For each pair of a simplification algorithm A and a classifier C_D (for a dataset D) we want to evaluate the interplay between loyalty and complexity, and do this as follows:

– Pick 100 instances P at random from the dataset D, preserving the original fraction of instances from each class

- Loop over 100 values of the complexity parameter α_c, from 0 to 1 in steps of 0.01.
- For each instance $ts \in P$ and each value of α_c, we run A to get the simplification $sts = A(\alpha_c, ts)$.
- We record two things:
  - $(||sts|| + 1)/|ts|$, to compute complexity
  - $C_D(ts) == C_D(sts)$, (True or False) to compute loyalty
- For each value of α_c, over all 100 $ts \in P$, we then compute the complexity, loyalty and kappa loyalty as per Definition 1:
  - the average complexity in the 100 simplifications having this value of α_c
  - the percentage of loyal simplifications, also called agreement
  - Cohens kappa value for the loyalty agreement (which takes into account unbalanced classes)

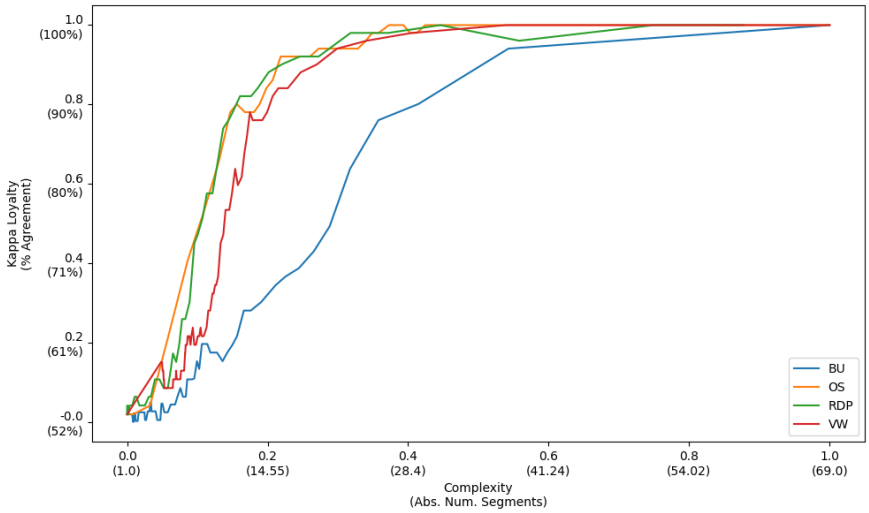


Fig. 2. SonyAIBORobotSurface1 dataset: Plot of Kappa loyalty (percent loyalty in parenthesis) versus Complexity (average number of segments in parenthesis) of the Rocket classifier, for all 4 simplification algorithms.

After doing the above we have, for each triple (A, α_c, C_D) of algorithm, complexity parameter and classifier, 100 values of kappa loyalty and complexity, one for each instance in P. For each algorithm and classifier pair (A, C_D), to visualize the results of these 10.000 simplifications (100 instances in P times 100 distinct values of α_c) we plot a curve. See Figure 2 showing kappa loyalty (loyalty in parenthesis) versus complexity (number of segments in parenthesis)

Table 2. AUC values for the 4 algorithms on the Rocket classifier. In the top row (Mean), we see that on average over all 40 datasets, OS achieves the highest value. The other rows display an average across datasets of varying characteristics.

Metric	**OS**	**RDP**	**BU**	**VW**	**Datasets**
Mean	**89.57**	88.65	82.23	87.92	40(all)
Binary	**91.36**	90.71	86.49	89.72	18
Multiclass (>2)	**88.10**	86.97	79.63	86.45	22
Stationary	**85.06**	83.97	79.32	82.08	7
Non-Stationary	**90.40**	89.55	82.71	89.10	30
Partially-Stationary	**91.71**	90.62	84.22	89.76	3
Seasonal	**95.90**	94.73	90.21	95.52	8
Non-Seasonal	**87.98**	87.13	80.08	86.02	32
Low-Entropy	**91.83**	90.01	84.57	90.57	14
Medium-Entropy	**88.53**	87.59	79.41	86.09	12
High-Entropy	88.19	**88.20**	82.24	86.84	14

for all 4 simplifications algorithms on Rocket classifiers C_D on the representative datasetSonyAIBORobotSurface1. Note that loyalty stays high even for quite low complexity values, and this is typical for most of the 40 datasets.

To compare the performance of the four simplification algorithms, we measure the AUC - Area Under Curve - as a value between 0 and 100, of the 40 such kappa loyalty versus complexity curves for each simplification algorithm. In Table 2 we show in the first row the average AUC for each of the four algorithms, and it is clear that OS and RDP have the best performance on average. The table also shows the average performance on classifiers for datasets of various characteristics, like Number of Classes, Stationarity, Seasonality and Entropy. We see that datasets that are Non-Stationary, Seasonal and with Low Entropy achieve higher AUC values.

We also measured performance of the four simplification algorithms at fixed loyalty values of 0.8, 0.85, 0.9, 0.95 and 1.0. In Table 3 we see that again OS has the best performance, with a resulting best average complexity of 0.15 at loyalty 0.9 and 0.35 at loyalty 1.0, which means that the time points kept in the simplification at these high loyalty values is on average 15% and 35%. This is a very good indicator that simplifications are useful in interpretability of TSC, in particular the fact that OS simplifications keeping only 35% of the points often have loyalty 1.0, i.e. with simplified time series not altering its classification. In Table 4 we show the average number of segments of the simplifications, at various loyalty values, for both OS and RDP, for all 40 datasets.

6.3 A closer look at OS and RDP

We have seen that the OS algorithm has the best average performance and also that RDP has the best average performance out of the three faster $O(n \log n)$ algorithms. For these two algorithms, OS and RDP, we consider the following

Table 3. Average Complexity values for the 4 algorithms on the Rocket classifiers. Over all the loyalty values we see that OS performs best and RDP second best.

Loyalty	**OS**	**RDP**	**BU**	**VW**
0.8	**0.11**	0.12	0.18	0.12
0.85	**0.13**	0.14	0.21	0.14
0.9	**0.15**	0.17	0.27	0.17
0.95	**0.19**	0.23	0.37	0.24
1.0	**0.35**	0.56	0.74	0.64
No Simp.	1.0	1.0	1.0	1.0

question over a variety of dataset characteristics: for exactly what values of loyalty is the complexity low enough that the user may benefit from seeing the simplifications?

Let us also mention that we have been experimenting with a tool for interpretability of a TSC C_D using the OS and RDP simplification algorithms.

The user selects an algorithm, say RDP, and a loyalty value, say 90%, and the tool will first find the minimum α_c value so that RDP with this parameter value has expected loyalty at least 90%. In the next step, prototypes for the classification of dataset D by C_D are computed, say two sets P_0 and P_1 for D being binary. In the final step, instead of showing these prototypes to the user, we show the simplifications $RDP(\alpha_c, ts)$, for each ts in P_0 and P_1, together with the class P_0 or P_1 that it belongs to. This way the user can make his or her own mental model of how C_D makes its classification. As stated above, the main question is if the number of segments in these simplifications is small enough that the user is willing to pay the price of the loyalty being only 90%. Clearly, this will depend on the length of the original time series, with shorter time series having fewer segments in their simplifications. In Figure 3 we address this question, by showing complexity (fraction of time points kept in the simplification) across various times series lengths and how this varies as the loyalty values vary, for OS simplifications. Even for loyalty 1.0 we see that the simplifications mostly result in a complexity of less than 0.6, i.e. a 40% reduction in the number of time points. This is an indicator that simplifications are useful in interpretability of TSC, and may be caused by the original time series having several consecutive data points lying on almost straight lines. Note the steep value at time series length 166, which corresponds to the dataset ChlorineConcentration. In Table 4 we see that this dataset indeed has a very high number of segments for loyalty values (times 100) of 90, 95 and 100.

7 End-user evaluation with forward simulation

As time series are notoriously difficult for human users, most of the human evaluations of XAI methods done so far have been qualitative and involving domain experts. Our goal in this work has been to add the new dimension of

Table 4. 40 UCR datasets. The first 5 columns denote Cl. (class length), Len. (length of time series) , Stat. (stationarity), Seas. (seasonality) and Entr. (entropy). The next 4 show average number of segments in the simplifications produced by OS, on the 100 instances chosen randomly, for loyalty values (times 100) of 90 to 100. The last 3 columns show the same for RDP.

Name	**Cl.**	**Len.**	**Stat.**	**Seas.**	**Entr.**	**OS 90**	**OS 95**	**OS 100**	**RDP 90**	**RDP 95**	**RDP 100**
Adiac	37	176	True	True	0.27	28.5	43.4	66.1	94.6	134.8	175.0
BME	3	128	False	True	0.12	6.3	6.6	7.6	5.7	6.4	17.0
CBF	3	128	False	False	0.83	32.8	35.0	43.1	20.4	23.3	32.9
Chinatown	2	24	False	False	0.23	2.3	2.4	8.3	4.1	4.6	12.3
ChlorineConct	3	166	True	False	0.78	109.4	118.0	148.0	127.9	146.4	165.0
Crop	24	46	False	False	0.37	15.7	19.2	29.0	15.9	24.5	41.2
DistalPhxOutlAge	3	80	False	False	0.27	10.8	13.7	58.7	8.7	17.2	70.9
DistalPhxOutlCor	2	80	False	False	0.30	9.7	10.5	59.8	9.9	13.6	72.0
DistalPhxTW	6	80	False	False	0.27	18.1	22.6	36.5	17.8	28.9	69.8
ECG200	2	96	False	False	0.46	4.6	7.0	20.4	5.2	6.3	40.4
ECG5000	5	140	False	False	0.24	4.7	5.9	9.5	5.8	6.2	29.9
ECGFiveDays	2	136	Partial	False	0.20	6.1	6.6	8.8	6.6	6.9	7.8
ElectricDevices	7	96	Partial	False	0.29	25.8	31.4	48.4	28.0	37.7	50.4
FaceAll	14	131	True	False	0.58	21.7	30.6	73.5	20.9	30.0	129.3
FacesUCR	14	131	True	False	0.55	16.3	18.7	28.3	19.4	22.1	26.4
GunPoint	2	150	False	True	0.11	5.3	6.0	7.8	6.2	7.2	10.1
GunPointAgeSpan	2	150	False	True	0.13	4.6	5.4	14.9	4.9	8.1	148.9
GunPointMalVsFem	2	150	False	True	0.12	5.1	5.5	8.9	4.6	5.8	6.6
GunPointOldVsYou	2	150	False	True	0.12	5.4	6.4	8.7	6.7	7.7	13.9
ItalyPowerDemand	2	24	False	False	0.24	4.4	4.9	12.6	4.2	5.2	14.4
MedicalImages	10	99	False	False	0.18	9.0	12.6	35.3	17.8	42.0	98.0
MiddlePhxOutlAge	3	80	False	False	0.23	17.5	35.3	49.0	29.6	35.9	77.9
MiddlePhxOutlCor	2	80	False	False	0.24	17.8	19.1	28.8	17.8	22.9	77.3
MiddlePhxTW	6	80	False	False	0.25	26.6	31.6	64.1	28.1	35.4	71.2
MoteStrain	2	84	False	False	0.31	4.0	7.6	15.2	6.6	7.8	82.2
PhalangesOutlsCor	2	80	False	False	0.25	11.8	13.7	24.1	16.2	19.4	30.7
Plane	7	144	False	False	0.35	8.2	8.6	9.0	9.0	9.3	14.9
PowerCons	2	144	False	False	0.36	12.4	20.2	48.5	9.5	16.7	133.2
ProxPhxOutlAge	3	80	Partial	False	0.24	9.6	17.8	39.6	13.6	21.8	65.2
ProxPhxOutlCor	2	80	False	False	0.23	16.2	22.3	41.1	20.5	28.2	63.3
ProxPhxTW	6	80	False	False	0.23	10.8	16.0	30.8	14.7	25.6	67.8
SmoothSubspace	3	15	False	False	0.04	5.5	6.7	9.9	6.0	7.0	10.9
SonyAIBORobotS1	2	70	True	False	0.35	11.6	15.6	26.4	11.6	16.0	31.3
SonyAIBORobotS2	2	65	True	False	0.38	7.7	8.7	12.6	8.1	9.2	22.6
SwedishLeaf	15	128	False	True	0.42	11.2	15.5	53.0	11.1	17.1	43.1
SyntheticControl	6	60	False	False	0.46	10.1	12.4	19.3	8.6	9.7	20.6
TwoLeadECG	2	82	False	False	0.33	6.7	8.4	10.9	7.9	8.5	9.4
TwoPatterns	4	128	True	False	0.70	9.7	10.4	11.0	7.4	7.9	12.5
UMD	3	150	False	True	0.12	7.0	7.9	24.5	7.3	9.8	34.7
Wafer	2	152	False	False	0.13	7.4	8.1	11.0	8.4	9.0	14.3

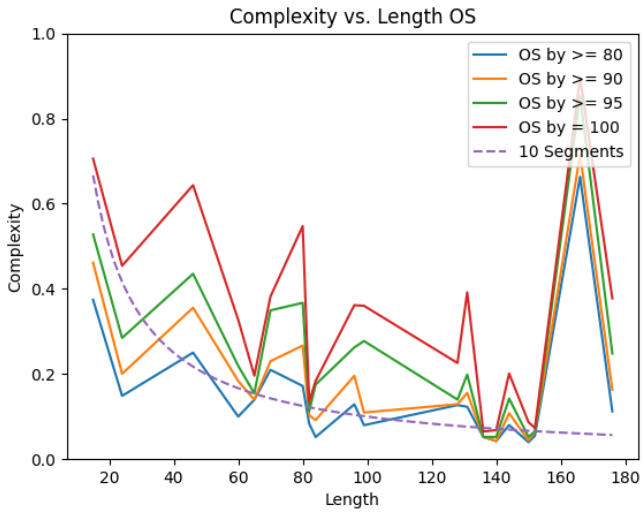


Fig. 3. OS Algorithm: Complexity versus time series length for 4 loyalty values (times 100) from 80% to 100%, over all 40 datasets. The dotted line shows where simplifications on 10 segments would lie. Note that even with loyalty 1.0 we usually achieve complexity below 0.6. The steep value at length 166 is for the ChlorineConcentration dataset.

user evaluations involving end-users, i.e. non-experts, on the use of simplifications for interpretability of TSCs. In this section we present an end-user evaluation with forward simulation, i.e. where participants will be trained on prototypes from each of the TSC classes, and then tested by being asked to guess classes of random instances. We first present the 4 datasets chosen for the evaluation, discuss the format of the training and testing stages, and give the results of the evaluation.

Table 5. Summary of the datasets employed in the evaluation. For each dataset, we report the number of classes, the length, the test accuracy (wrt Ground Truth) obtained by the Rocket model, and the selected loyalty thresholds Loy and number of segments (#Segs) corresponding to Knee Low and Knee High.

Dataset	**Cl.**	Length	**acc**	**K.Low:Loy(#Seg)**	**K.High:Loy(#Seg)**
Chinatown	2	24	98	0.72(1.0)	0.92(2.4)
ECG200	2	96	96	0.82(3.1)	0.97(7.7)
SonyAIBORobotSurface1	2	70	90	0.90(11.6)	0.96(15.9)
UMD	3	150	94	0.95(7.9)	0.99(13.6)

Since our survey participants are not domain experts and time series are notoriously non-intuitive to humans we want univariate datasets with a binary or ternary classification. Moreover, we want to cover a good spread of character-

istics, also regarding Stationarity, Seasonality and Entropy. We chose the four datasets in Table 5. The majority of the 40 UCR datasets are neither Stationary nor Seasonal, and both Chinatown and ECG200 belong here, with the first having low entropy and the second having high entropy. The chosen Sony dataset is Stationary but not Seasonal, whereas UMD is the opposite. Further details of these datasets can be found in Table 5. We also include the accuracy obtained by the Rocket TSC on the test set. For these datasets, we used the training/test split defined in the repository.

We also chose four simplification levels for the survey, as follows. Firstly, we are comparing against original time series, so we need a level with No simplification. Secondly, we use the simplification with lowest complexity that still achieves max loyalty of 1.0, and call this L1.0. Thirdly, looking for a sweet spot of loyalty versus complexity we inspect the AUC curves and pick points where the second derivative seems to change sign. The highest such point we call Knee High, and the second highest is called Knee Low. As an example, Figure 4 shows the complexity-loyalty curve for Chinatown using the OS algorithm with the 4 chosen simplification levels highlighted. These loyalty values are also given in Table 5.

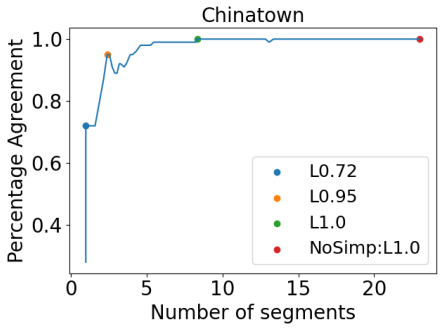


Fig. 4. Complexity-loyalty curve for Chinatown using the OS algorithm. Four points have been manually selected, covering: the original dataset without simplification, the point of lowest complexity for a loyalty of 1.0, one point we call High Knee, and one point we call Low Knee.

We thus have 16 different user configurations, 4 datasets by 4 simplification levels, as shown in Table 6. Each participant will be shown 4 user configurations, ensuring that these are from 4 distinct datasets and 4 distinct simplification values, i.e. with 4 participant groups forming a 4 by 4 Latin square. For each of these 4 configurations the participant is shown one page with 3 prototypes per class, and at the same time one page with 10 random test instances, see Figure 8 in the Appendix for an example. Prototypes are selected by applying KMedoids with Dynamic Time Wrapping (DTW) obtaining the medioids of each class [16].

It is important to note the following. For TSC C, simplification algorithm A_α and an original time series ts taken from dataset D, when we show the user

a prototype or test instance $A_\alpha(ts)$ we consider it to be in class $C(ts)$. The loyalty λ of the triple (C, A_α, D) is the probability that $C(A_\alpha(ts)) = C(ts)$. Thus, when $\lambda < 1.0$ those two classifications can differ which could lead the user to misinterpret properties of the classifier on original time series, and we will take this into account when evaluating our results.

7.1 Evaluation results

In total, we had 19 voluntary participants, who were students in a university-level informatics course, none of whom received compensation. The participants were presented with the survey and freely chose to participate. The participants were randomly divided into the 4 survey groups. They were asked to answer 10 test questions on each of 4 configurations. We ensured that participants spent at least 15 minutes on the complete survey. In Table 6 we see the absolute accuracy, i.e. the percentage of correct test answers, on each of the 16 configurations.

Table 6. Evaluation Results. Mean human accuracies (in %) for each dataset and simplification level. Values are given as mean ± half of the 95 % bootstrap confidence interval (CI) across participants.

Dataset	**Knee Low**	**Knee High**	**L1.0**	**No Simp.**
Chinatown	78.0 % ± 6.0	82.5 % ± 3.8	76.0 % ± 15.0	66.0 % ± 14.0
ECG200	68.0 % ± 12.0	72.0 % ± 12.0	72.0 % ± 8.0	72.5 % ± 3.8
SonyAIBORobotSurface1	67.5 % ± 3.0	72.0 % ± 10.0	76.0 % ± 15.0	78.0 % ± 10.0
UMD	98.0 % ± 3.0	100.0 % ± 0.0	100.0 % ± 0.0	96.0 % ± 6.0

As discussed earlier, our goal is to teach the patterns discovered by the TSC C in classifying the *original time series*. However, when we consider the class label of an original time series attached to a *simplified time series* with loyalty $\lambda < 1$ then what the user learns does not fully carry over to the original. For example, if $\lambda = 0.8$ then there is 80% chance that the simplified instance $A(ts)$ is classified the same as the original instance ts, so whatever the user learns has an 80% chance of being useful for understanding the patterns used by C in classifying the original time series. Hence we need to multiply the absolute user accuracies by the loyalty values to obtain the correct loyalty-adjusted accuracy with regards to the goal of understanding the TSC on the original time series.

Table 7 shows the loyalty-adjusted accuracies of the human subjects, on the 16 configurations, with values in bold showing the max scores. For Chinatown simplifications seem to be useful, either at the Knee High loyalty value 0.92, or at L1.0 loyalty value 1.0. However, for the other three datasets ECG200, SonyAIBORobotSurface1 and UMD, simplifications do not seem to help. In the next section we reason carefully about why simplifications help or not for these four datasets, and use this as a starting point for a general discussion involving also other datasets among the 40 covered.

Table 7. Evaluation results. Mean loyalty-adjusted human accuracies (in %) for each dataset and simplification level. These are achieved by multiplying absolute accuracy from Table 6 with loyalty from Table 5.

Dataset	Knee Low	Knee High	L1.0	No Simp.
Chinatown	56.6	75.9	**76.0**	66.0
ECG200	55.8	69.8	72.0	**72.5**
SonyAIBORobotSurface1	60.8	69.1	76.0	**78.0**
UMD	93.1	99.0	**100.0**	96.0

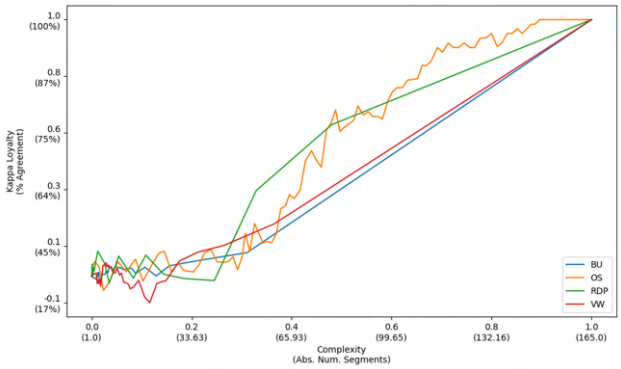


Fig. 5. ChlorineConcentration: Plot of kappa loyalty versus Complexity of the Rocket classifier, for all 4 simplifcation algorithms.

8 A framework for evaluating the utility of simplifications

In this section we present a framework giving a rule-of-thumb to decide if a simplification algorithm A (over various α values) is useful for interpretability of a TSC C on a dataset D. The four datasets covered in our user study, together with other selected datasets among the 40 investigated, will act as instructive in explaining our reasoning that leads to the flowchart presented in Figure 7.

UMD dataset: Human accuracy on the original time series (No simpl) is a very high value of 96% and so we cannot hope that simplifications improves this by much. Figure 8 in the Appendix shows the user survey for this configuration and it is indeed easy for the human eye to see the patterns of the classifier. This observation is generalized and forms the first box in the flowchart in Figure 7.

ChlorineConcentration dataset: This dataset is responsible for the steep value in Figure 3. Also table 4 and also Figure 5 show that as we decrease loyalty the complexity values remain high. In that case we do not expect loyalty-adjusted accuracies to be better for simplifications than for No simplifications, as few data points have been removed. This observation is generalized in the second box of Figure 7.

ECG200 dataset: Figure 6 shows a prototype time series from the survey su-

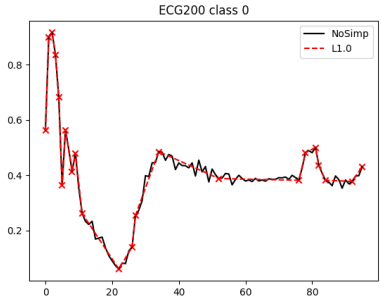


Fig. 6. Figure showing the overlap between NoSimplification in black and the simplification at L1.0 in red. This instance is one of three prototypes from class 0 of ECG200 shown to the participants in the user survey, with some users seeing the black and other users seeing the red

Table 8. Human Proxy Classifier. Accuracy/loyalty-adjusted accuracy for logistic regressor trained on the same task as human participants face in the survey. Note that we expand training and testing sets for the logistic regressor compared to the human survey.

Dataset	Knee Low	Knee High	L1.0	NoSimp
Chinatown	100% / 72%	87% / 82%	93% / 93%	93% / 93%
ECG200	80% / 66%	77% / 74%	80% / 80%	80% / 80%
SonyAIBORobotSurface1	50% / 45%	50% / 48%	50% / 50%	50% / 50%
UMD	83% / 79%	83% / 82%	83% / 83%	83% / 83%

perimposed with its L1.0 simplification. Even though the latter has retained only 22 of the original 96 data points, the Euclidean distance (error) between the two is very low. This means that a human user will probably not do much worse on No Simpl than on L1.0, as the human eye in any case will tend to straighten out a series of consecutive almost collinear points. This observation is generalized in the third box of Figure 7.

SonyAIBORobotSurface1 dataset: This dataset passes the first three boxes in the flowchart, but the human accuracy levels in Tables 6 and 7 are highest for No simplification. Could we have discovered this was likely to happen without doing a human user study? Yes, we could train a simple classifier (as proxy for human user) on the exact same task facing the human user in the various configurations and see how well it does. Table 8 shows results of a logistic regressor on the 16 configurations in the survey. Note the chosen Sony dataset is the only one of consistently low accuracy. We assume a human would also have problems and this observation is generalized in the fourth box of Figure 7.

Chinatown dataset: It passes all the tests mentioned above and therefore our rule-of-thumb tells us that the simplifications are probably useful for human interpretability.

Let us remark right away that for simplicity of presentation purposes we have made the above discussion and the flowchart in Figure 7 somewhat simpler than reality dictates. Rather then clearcut Yes/No answers there will sometimes be propensities. For a concrete triple of simplification algorithm A, TSC C and dataset D, these propensities may or may not add up to a recommendation for using the simplifications in interpretation.

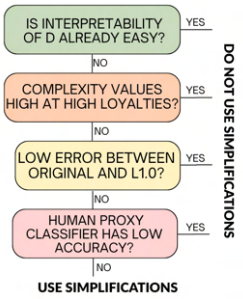


Fig. 7. A flowchart to estimate if a certain simplification algorithm A over various parameter values α will improve the interpretability of a TSC C on a dataset D.

9 Conclusion

We have introduced metrics like complexity and loyalty, see Definition 1, to evaluate the use of simplification algorithms in interpretability of Time Series Classifiers. Human perceptual studies [2,14] show that people read line-slope and point-position with near-bar accuracy, whereas decoding symbols or averages is slower and less precise, and this supports our choice of using simplifications. We have applied these metrics to evaluate four simplification algorithms on Rocket classifiers for 40 datasets from the UCR repository [4]. The theoretical results show that the OS (Optimal Simplification) and RDP (Ramer-Douglas-Peuker) algorithms have the best performance[5]. The OS algorithm has time complexity cubic in the length n of the original time series, and hence the $O(nlogn)$ RDP algorithm would be preferable for very long time series.

Our results contribute to an understanding of when simplifications can be useful for interpretability of TSC. We performed a practical user evaluation with forward simulation. We used our findings to arrive at a flowchart that employs the introduced metrics of complexity and loyalty to estimate whether simplifications would be useful for interpretability in a given situation.

[5] The BU (Bottom-Up) algorithm was designed to minimise the number of segments that go over more than a single time step. However, for the purpose of the comparisons in this paper we needed to count also for BU the segments on a single time step. This explains why BU performs worse than the other algorithms.

References

1. Camponogara, E., Nazari, L.F.: Models and algorithms for optimal piecewise-linear function approximation. Mathematical Problems in Engineering **2015**(1), 876862 (2015)
2. Cleveland, W.S., McGill, R.: Graphical perception: Theory, experimentation, and application to the development of graphical methods. Journal of the American Statistical Association **79**(387), 531–554 (1984)
3. Cohen, J.: A coefficient of agreement for nominal scales. Educational and Psychological Meas. **20**, 37–46 (1960)
4. Dau, H.A., Keogh, E., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Yanping, Hu, B., Begum, N., Bagnall, A., Mueen, A., Batista, G., Hexagon-ML: The UCR time series classification archive (2018)
5. Delaney, E., Greene, D., Keane, M.T.: Instance-based counterfactual explanations for time series classification. In: Case-Based Reasoning Research and Development - Int. Conference, ICCBR 2021. pp. 32–47 (2021)
6. Dempster, A., Petitjean, F., Webb, G.I.: ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. Data Min. Knowl. Discov. **34**(5), 1454–1495 (2020)
7. Dickey, D., Fuller, W.: Distribution of the estimators for autoregressive time series with a unit root. Journal of the American Statistical Association **74** (06 1979)
8. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
9. Doublas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a dikgitized line of its caricature. Cartographica **10**(2), 112–122 (1973)
10. Gee, A.H., Garcia-Olano, D., Ghosh, J., Paydarfar, D.: Explaining deep classification of time-series data with learned prototypes. In: CEUR workshop proceedings. vol. 2429, p. 15. NIH Public Access (2019)
11. Geler, Z., Kurbalija, V., Ivanović, M., Radovanović, M.: Weighted kNN and constrained elastic distances for time-series classification. Expert Systems with Applications **162**, 113829 (2020)
12. Guillemé, M., Masson, V., Rozé, L., Termier, A.: Agnostic local explanation for time series classification. In: 31st IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2019. pp. 432–439 (2019)
13. Håvardstun, B., Ferri, C., Telle, J.A.: An interactive tool for interpretability of time series classification. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 399–403. Springer (2024)
14. Heer, J., Bostock, M.: Crowdsourcing graphical perception: using mechanical turk to assess visualization design. In: Proceedings of the SIGCHI conference on human factors in computing systems. pp. 203–212 (2010)
15. Hirschmann, F.: RDP: Pure python implementation of the ramer-douglas-peucker algorithm (2016), pyPI package version 0.8
16. Holder, C., Guijo-Rubio, D., Bagnall, A.: Clustering time series with k-medoids based algorithms. In: International Workshop on Advanced Analytics and Learning on Temporal Data. pp. 39–55. Springer (2023)
17. Hügel, S.: Simplification (2025), `https://github.com/urschrei/simplification`
18. Ismail, A.A., Gunady, M.K., Bravo, H.C., Feizi, S.: Benchmarking deep learning interpretability in time series predictions. In: NeurIPS 2020 (2020)

19. Ismail Fawaz, H., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data mining and knowledge discovery **33**(4), 917–963 (2019)
20. Keogh, E., Chu, S., Hart, D., Pazzani, M.: An online algorithm for segmenting time series. In: Proceedings 2001 IEEE International Conference on Data Mining. pp. 289–296 (2001)
21. Keogh, E., Chakrabarti, K., Pazzani, M., Mehrotra, S.: Dimensionality reduction for fast similarity search in large time series databases. Knowledge and information Systems **3**, 263–286 (2001)
22. Keogh, E., Chu, S., Hart, D., Pazzani, M.: Segmenting time series: A survey and novel approach. Data mining in time series databases **57**(2004), 1–22 (2004)
23. Löning, M., Bagnall, A.J., Ganesh, S., Kazakov, V., Lines, J., Király, F.J.: sktime: A unified interface for machine learning with time series. CoRR **abs/1909.07872** (2019), `http://arxiv.org/abs/1909.07872`
24. Lundberg, S.M.: shap.deepexplainer. `https://shap.readthedocs.io/en/latest/generated/shap.KernelExplainer.html` (2018), accessed: 2025-08-01
25. Pedregosa, F., et al.: Scikit-learn: Machine learning in Python **12**, 2825–2830 (2011)
26. Pincus, S.M.: Approximate entropy as a measure of system complexity. Proceedings of the National Academy of Sciences **88**(6), 2297–2301 (1991)
27. Rajkomar, A., Oren, E., Chen, K., Dai, A.M., Hajaj, N., Hardt, M., Liu, P.J., Liu, X., Marcus, J., Sun, M., et al.: Scalable and accurate deep learning with electronic health records. NPJ digital medicine **1**(1), 1–10 (2018)
28. Ramer, U.: An iterative procedure for the polygonal approximation of plane curves. Computer Graphics and Image Processing **1**(3), 244–256 (1972)
29. Rojat, T., Puget, R., Filliat, D., Del Ser, J., Gelin, R., Díaz-Rodríguez, N.: Explainable artificial intelligence (xai) on timeseries data: A survey. arXiv preprint arXiv:2104.00950 (2021)
30. Schlegel, U., Arnout, H., El-Assady, M., Oelke, D., Keim, D.A.: Towards a rigorous evaluation of xai methods on time series. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 4197–4201. IEEE (2019)
31. Schlegel, U., Lam, D.V., Keim, D.A., Seebacher, D.: Ts-mule: Local interpretable model-agnostic explanations for time series forecast models (2021)
32. Si, Y.W., Yin, J.: Obst-based segmentation approach to financial time series. Engineering Applications of Artificial Intelligence **26**(10), 2581–2596 (2013)
33. Siddiqui, S.A., Mercier, D., Munir, M., Dengel, A., Ahmed, S.: Tsviz: Demystification of deep learning models for time-series analysis. IEEE Access **7** (2019)
34. Susto, G.A., Cenedese, A., Terzi, M.: Time-series classification methods: Review and applications to power systems data. Big data application in power systems pp. 179–220 (2018)
35. Tavenard, R., Faouzi, J., Vandewiele, G., Divo, F., Androz, G., Holtz, C., Payne, M., Yurchak, R., Rußwurm, M., Kolar, K., Woods, E.: Tslearn, a machine learning toolkit for time series data. Journal of Machine Learning Research **21**(118) (2020)
36. Telle, J.A., Ferri, C., Håvardstun, B.A.T.: Optimal robust simplifications for explaining time series classifications. In: Workshop on Explainable AI for Time Series and Data Streams (TempXAI 2024). CEUR Workshop Proceedings, vol. 3761, pp. 28–43 (2024)
37. Theissler, A., Spinnato, F., Schlegel, U., Guidotti, R.: Explainable AI for time series classification: A review, taxonomy and research directions. IEEE Access **10**, 100700–100724 (2022)
38. Visvalingam, M., Whyatt, J.D.: Line generalisation by repeated elimination of points. The cartographic journal **30**, 46–51 (1993)

39. Wan, Y., Gong, X., Si, Y.W.: Effect of segmentation on financial time series pattern matching. Applied Soft Computing **38**, 346–359 (2016)
40. Wang, Z.J., Wortman Vaughan, J., Caruana, R., Chau, D.H.: GAM coach: Towards interactive and user-centered algorithmic recourse. In: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems. pp. 1–20 (2023)
41. Wikipedia contributors: Supermajority — Wikipedia, the free encyclopedia (2025), `https://en.wikipedia.org/wiki/Supermajority`, [Online; accessed 7-May-2025]
42. Wu, C.J., Zeng, W.S., Ho, J.M.: Optimal segmented linear regression for financial time series segmentation. In: 2021 International Conference on Data Mining Workshops (ICDMW). pp. 623–630. IEEE (2021)

10 Appendix. End-user evaluation

Survey conducted with humans. Figure 8 shows both instances with labels used for the users as reference, and unlabelled instances that the users were asked to classify.

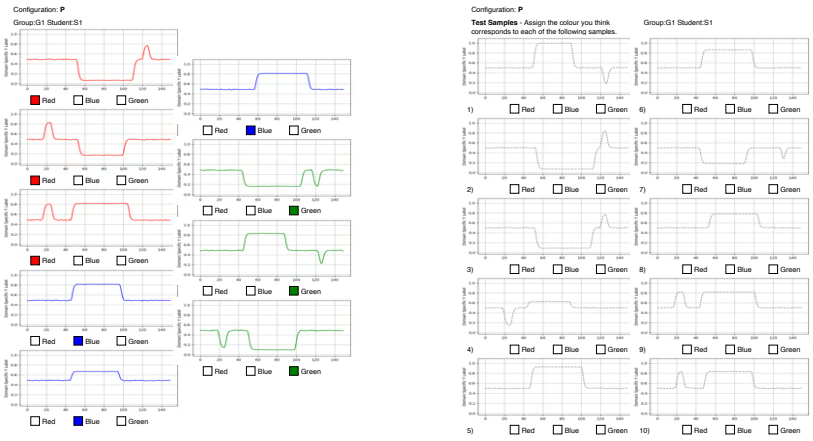


Fig. 8. UMD dataset showing prototypes and test instances, No Simplification. Example of the layout presented in the user survey, except that participants saw this on larger printouts. It contains nine sub-figures corresponding to the labelled prototypes used as "training" instances for the subjects and the ten "testing" instances. Note that the subjects were shown the two pages with training and testing side-by-side, making it easier to do pattern matching. For the interested reader we list the correct class classification of the test instances: 1:Green, 2:Red, 3:Red, 4: Green, 5:Blue, 6:Blue, 7:Green, 8:Blue, 9:Red, 10:Red

Paper VII

Freitas, D., Håvardstun, B., Garigliotti, D., Telle, J. A., Ferri, C., & Hernández-Orallo, J.
Proceedings of the 28th European Conference on Artificial Intelligence (ECAI 2025), FAIA **413**, pp. 4507–4514 (2025)
DOI: 10.3233/FAIA251351
Extended version with appendix: arXiv:2505.10583

ECAI 2025
I. Lynce et al. (Eds.)

doi:10.3233/FAIA251351

Relative Drawing Identification Complexity Is Invariant to Modality in Vision-Language Models

Diogo Freitas[a,b,c,*], Brigt Håvardstun[d], Darío Garigliotti[d], Jan Arne Telle[d], Cèsar Ferri[e] and José Hernández-Orallo[e,f]

[a]Interactive Technologies Institute (ITI/LARSyS), Funchal, Portugal
[b]Faculty of Exact Sciences and Engineering, University of Madeira, Portugal
[c]NOVA Laboratory for Computer Science and Informatics, Caparica, Portugal
[d]Department of Informatics, University of Bergen, Norway
[e]Valencian Research Institute for Artificial Intelligence, Universitat Politècnica de València, Spain
[f]Leverhulme Centre for the Future of Intelligence, University of Cambridge, United Kingdom
ORCID (Diogo Freitas): https://orcid.org/0000-0002-2351-8676, ORCID (Brigt Håvardstun): https://orcid.org/0009-0007-1034-1422, ORCID (Darío Garigliotti): https://orcid.org/0000-0002-0331-000X, ORCID (Jan Arne Telle): https://orcid.org/0000-0002-9429-5377, ORCID (Cèsar Ferri): https://orcid.org/0000-0002-8975-1120, ORCID (José Hernández-Orallo): https://orcid.org/0000-0001-9746-7632

Abstract. Large language models have become multimodal, and many of them are said to integrate their modalities using common representations. If this were true, a drawing of a car as an image, for instance, should map to a similar area in the latent space as a textual description of the strokes that form the drawing. To explore this in a black-box access regime to these models, we propose the use of machine teaching, a theory that studies the minimal set of examples a teacher needs to choose so that the learner captures the concept. In this paper, we evaluate the complexity of teaching vision-language models a subset of objects in the Quick, Draw! dataset using two presentations: raw images as bitmaps and trace coordinates in TikZ format. The results indicate that image-based representations generally require fewer segments and achieve higher accuracy than coordinate-based representations. But, surprisingly, the teaching size usually ranks concepts similarly across both modalities, even when controlling for (a human proxy of) concept priors, suggesting that the simplicity of concepts may be an inherent property that transcends modality representations.

1 Introduction

As children, when we transform images of the world into drawings and other simplified sketches, we have the intuition that some objects are simpler than others [5, 18]. For instance, six segments are enough to represent a `house` that everybody can recognize, while a bit more is necessary to represent a `cat`. This intuition is epitomized by some guessing games where one person picks a concept from a card deck and has to draw something quickly for their team to identify the concept. We can easily describe and recognize some very simple visual concepts, such as letters, with verbalized descriptions. For instance, the letter `T` is a horizontal segment on top of a vertical segment. However, humans struggle to describe more complex shapes with verbal descriptions [26] or objects, such as a `cat`, using a series of segments.

Table 1: The simplest drawings (applying RDP algorithm on an original drawing) identified for the concept `cat`.

Model	Original (images)	Simplified (images)	Original (coordinates)	Simplified (coordinates)
Claude				
Gemini				
GPT-4 Turbo				
GPT-4o				
Llama				
Pixtral				

Large Language Models (LLMs) can identify objects from a textual representation of their coordinates [3]. Thus, we aim to discover whether this understanding maps to similar capabilities for the multimodal versions of these models. Also, we do not know whether this is independent of the modality. We ask two research questions:

- Q1 (*Absolute Invariance*): If we randomly sample a concept from a concept class, $c \in C$, would it take the same number of segments to identify it if represented as a bitmap drawing as if represented as a set of coordinates?
- Q2 (*Relative Invariance*): If we randomly sample two concepts from a concept class, $c_1, c_2 \in C$, and c_1 requires fewer segments than c_2 when represented as a bitmap, will this order prevail when expressed as coordinates?

Question Q1 refers to whether a concept represented as a bitmap drawing is easier or harder to recognize than the same concept as coordinates in text, while question Q2 is about the relative ranking. For instance, consider that c_1 is a `house` and c_2 is a `cat`. In Figure 1, if a `house` is easier than a `cat` when using the bitmap of the drawing

* Corresponding author. Email: diogo.freitas@staff.uma.pt.

(top of the figure), is it also easier when represented as segment coordinates (bottom of the figure)? This is the *relative invariance*. Note that we are not comparing with photographic images of the object since other features would come into play, such as a striped texture to distinguish a `tiger` from other felines. Such distinctions are particularly evident in machine vision systems [11].

However, how can one determine the notion of simplicity of a concept from its drawings? The idea we pursue in this paper is based on the field of machine teaching [36], and in particular, the notion of teaching minimality. A concept is as simple as a teacher can communicate the concept to a learner with as little information as possible. This captures our intuition that a `house` needs six segments while a `cat` needs more segments. Given a concept, the teacher has to find the simplest drawing in terms of the number of straight-line segments—the teaching size—that enables the learner to consistently recognize the concept. We use two different types of language representations (bitmaps of the drawing and coordinates in TikZ code) to present the concepts to the learner. Multiple models, including Generative Pretrained Transformer (GPT)-4 [1], Llama [13], Gemini [29], Pixtral, and Claude, are employed as the "learners". The resulting collection of the simplest images identified, across all concepts, all modalities, and all models, is intriguingly diverse. As a preview of our findings, see Table 1, showing the simplest identified images for the concept `cat`.

It is also important to note that priors play a role in machine teaching. When in doubt, the learner will more likely associate the evidence with the most common concept (e.g., a `house` is more common than an `envelope`). Accordingly, a Bayesian prior will be used to disentangle this effect when looking at the concept simplicity rankings.

The contributions of this paper are:

- A novel machine teaching framework for evaluating the complexity of concepts, which can be applied to drawings in coordinate- and image-based modalities.
- Use of the teaching size specifically to evaluate how simply and effectively the concept can be taught across both modalities.
- A comparison of both modalities across multiple models, including GPT-4, Llama, Gemini, Pixtral, and Claude, according to the number of concepts identified, accuracy, frequency of errors, and teaching size.
- A way to disentangle the effect of the learner's prior knowledge in the concept identification task.

These contributions are generic and can be applied to other problems and modalities. In our particular case, we show that bitmaps are more efficient than coordinates, but surprisingly, the order of complexity between the concepts is preserved to some extent. This suggests that either the representations of both modalities are tightly connected in the latent space of the model, or the simplicity of concepts is an inherent property that transcends modalities.

2 Related Work

Drawing (or Sketches) Recognition: Eitz et al. [8] provided a dataset of human drawings, including 250 concepts and 20,000 drawings. They introduced a support vector machine model to recognize these drawings and observed that humans outperformed its performance. Since then, AI models have been closer or even achieved higher accuracy than that of human classification for drawing recognition (e.g., Schneider and Tuytelaars 24, Yu et al. 34, Zhang et al. 35, Yang et al. 33). Using the *Quick, Draw!* dataset, Ha and Eck [14] proposed `sketch-rnn`, a model designed to create drawings of common objects that resemble those drawn by humans. A similar version of this model has also shown capabilities in drawing recognition [2]. Other neural approaches studied for this task include convolutional neural networks [16], and graph neural networks applied over drawings represented as graphs [32].

Drawing Capacities of LLMs: Sharma et al. [25] assess the visual abilities of different language models. They conduct experiments that prompt the models to create code that draws images based on text descriptions and improve image generation code iteratively through text feedback. They show that: (a) LLMs possess limited ability to recognize concepts represented in code, and (b) these models sometimes fail to recognize concepts that they can accurately draw. Note that the authors addressed the problem as a multi-class classification problem. Moreover, the online interface for collecting human drawings limits components to basic shapes like ellipses, possibly restricting participants' ability to create complex drawings. In their initial experiments with GPT-4, Bubeck et al. [3] present an example of drawing generation, showcasing text-to-image capabilities using TikZ. They show tasks such as GPT-4 drawing a unicorn and constructing TikZ code through a multi-step prompt process. In another study, Pourreza et al. [21] introduce the *Painter*, a modified LLM that creates drawings using virtual brush strokes based on user-provided text descriptions. Additionally, Cai et al. [4] evaluated GPT-4's ability to understand visual data in SVG format across various visual tasks, including image classification, visual reasoning, and image generation. Vinker et al. [31] propose *SketchAgent*, showing that while LLMs iteratively generate sketches, they struggle with spatial reasoning.

Machine Teaching: Machine teaching is a research area that focuses on identifying the optimal set of examples that allow a learner (e.g., a human or a machine) to identify a given concept [36]. To illustrate the underlying idea of machine teaching, assume the teacher wants the learner to identify the concept of prime numbers. To achieve this, the teacher uses the set $S_1 = \{2, 3, 5, 7, 11, 13\}$ and succeeds. However, would it not be enough for the learner just to see the smaller set $S_2 = \{19, 23\}$? Of course, that depends on the learner. In general, optimal teaching will depend on the model the teacher has of the learner. Machine teaching presents an alternative framework to machine learning (where examples are not chosen but sampled from a distribution) to answer the question of whether some concepts are inherently more complex than others. The connections between machine teaching and computational learning theory are strong; see, e.g., the works by Doliwa et al. [6] or Moran and Yehudayoff [19], with machine teaching putting the emphasis on the minimal evidence that distinguishes the concept from all the rest. To determine how easy it is to teach a concept, the teaching dimension [36]—the minimum number of examples the learner needs to identify a concept—was traditionally used. Telle et al. [30] introduced a new metric named teaching size. This metric puts the focus on the sum of the sizes of the examples needed to identify a concept, rather than only the number of examples.

3 Methods

The drawings used in this work come from the *Quick, Draw!* dataset [15, 14], which includes over 50 million drawings of 345 concepts. Collected by Google Creative Lab via an interactive game, participants had 20 seconds to draw a concept while a neural network attempted real-time recognition. The dataset is the largest collection of doodles in the world, with contributions from more than 15 million participants.

Each drawing in the Simplified Drawing files that we use is stored as vectors of distinct pen strokes, i.e., distinct continuous movements

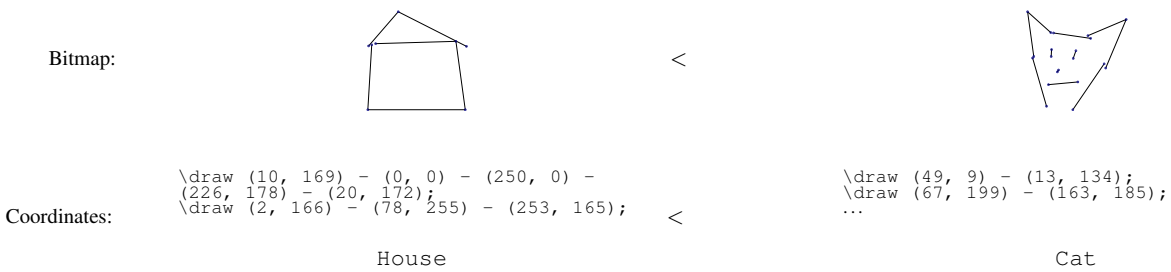


Figure 1: In this paper, we address two research questions. First, Q1 (absolute invariance): When using a vision-language model, are bitmaps (top) equally efficient representations for drawings as coordinates (bottom)? The second question is Q2 (relative invariance): Is the order (left vs. right) of simplicity preserved across modalities?

of the pen without lifting. Each stroke s_i is represented by a sequence of (x, y) coordinates $\{(x_{i1}, y_{i1}), (x_{i2}, y_{i2}), \ldots, (x_{in}, y_{in})\}$. Note that each pair of consecutive points in a stroke creates a segment. Additionally, for each drawing, a binary flag r indicates whether the game's neural network correctly recognized the concept.

The following sections cover concept selection, corresponding drawings, learners, the machine teaching setting, and the drawing selection conducted before testing the framework.

3.1 Teaching Size

Let D denote an infinite space of possible drawings (and their simplifications, as will be explained later), and let C be a set of concepts. We use D_c to denote all the drawings of a concept $c \in C$. For any given concept $c \in C$, the objective is to identify the simplest drawing $S \in D_c$ (represented as S^m with modality m being either bitmap or coordinates) such that a learner L successfully learns c with a probability of at least ρ over N independent trials (i.e., recognition consistency). The *teaching size (TS)* of c for the modality m can then be defined as follows:

$$\mathrm{TS}_{\rho,N,m}(c) = \min_{S \in D_c} |S^m| \text{ s.t. } \sum_{1}^{N} \mathbb{1}\left[L(S^m) = c\right] \geq \rho \cdot N, \quad (1)$$

where $\mathbb{1}[\cdot]$ is the indicator function, which equals 1 if the learner L correctly identifies concept c from the drawing S^m, and 0 otherwise.

We argue that a good metric for assessing the simplicity of a given drawing d can be based on the number of segments it contains. This is represented by $|S^m|$ in the above equation. This metric is intrinsic to the drawing itself, thereby avoiding dependencies on the length or verbosity of the instructions used to generate it, such as in a descriptive language like TikZ.

We also note here that while our implementation of teaching size is grounded in segment count for drawings, the framework itself is more general. Teaching size, as a proxy for descriptive complexity, can be adapted to other domains using modality-appropriate metrics.

3.2 Concepts

In our work, if the expected concept is `car` and the identified concept is `police car`, the identification is still considered correct because `police car` is a specific type of `car`, i.e., it is a semantically related prediction. This approach is similar to the one followed by Lamb et al. (2020). This means that if a specific sub-concept, or *hyponym*, is identified, it should still be seen as a correct identification as long as it falls under the more general expected concept. For a concept c, such as `car`, we consider a set of hyponyms $h(c)$ that corresponds to a set of concepts with a more specific meaning than c, e.g., `police car` belongs to $h($`car`$)$. For this study, we want a set of concepts that ensures that in the set of their hyponyms, there is no overlap, i.e., for any two concepts c_i, c_j, we have $h(c_i) \cap h(c_j) = \emptyset$. This rules out certain pairs of concepts available in the *Quick, Draw!*, like `van` and `car`, and it enhances the clarity and robustness of the study. We thus select the following subset of 20 concepts from the 345 concepts available in *Quick, Draw!*, with no overlap among their hyponyms: `apple`, `banana`, `car`, `cat`, `computer`, `cup`, `door`, `envelope`, `fish`, `grass`, `hockey puck`, `house`, `key`, `radio`, `string bean`, `sun`, `sword`, `television`, `The Great Wall of China` and `tree`.

In Table 3 in the Appendix [9], we list each concept from the dataset and the accepted hyponyms that are considered correct. This correspondence is established by human inspection and after the execution of the drawing selection phase (cf. Sect. 3.6) and the machine teaching framework experiments, with the results then analyzed based on these mappings.

3.3 Drawings

After choosing the concepts to study, we only include drawings that the game's neural network correctly identified (i.e., $r = 1$) in our research. For every concept, approximately 50 drawings are selected by a proportional random stratified sampling method [27], which groups drawings into bins based on their number of segments. (This number is approximate, as there may be rounding errors when calculating the number of samples for each bin according to its proportion.) The bin width was obtained using the minimum bin width between the Sturges's rule and the Freedman–Diaconis Estimator, ensuring that drawings of any concept are represented in a way that reflects the distribution of stroke counts for all correctly identified drawings of that concept in the dataset.

To simplify the drawings in our study, we employ the Ramer–Douglas–Peucker (RDP) algorithm [22, 7] on each stroke s of a given drawing d. RDP reduces the number of segments in each stroke while preserving its overall shape. Specifically, given a stroke s with a sequence of points $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, the RDP algorithm iteratively selects the most distant point (x_d, y_d) from the line segment connecting the first and last points of the stroke. If this distance is below a predefined threshold ϵ, then this stroke is simplified to a single segment $\{(x_1, y_1), (x_n, y_n)\}$ on the first and last points. However, if the distance to (x_d, y_d) exceeds ϵ, the algorithm keeps this point and recursively processes the two sequences of points formed by $\{(x_1, y_1), \ldots, (x_d, y_d)\}$ and $\{(x_d, y_d), \ldots, (x_n, y_n)\}$. This ensures that the essential characteristics of the stroke, up to distance ϵ, are preserved. This process continues until all points in the stroke fall within the threshold, resulting in a simplified representation of the stroke with fewer segments. By incrementing the threshold parameter,

from an initial value of $\epsilon = 2$ [1], until each stroke is reduced to one segment, we generate simplified versions of each original drawing associated with a given concept c, resulting in new drawings $\{d\}_\epsilon \subseteq D_c$. Figure 2 illustrates a drawing simplification.

We note here that image and coordinate representations are generated differently, but both encode the same visual information. While not equivalent in all respects, the coordinates in TikZ are a form of structured data that, by reflecting a sequence of drawing actions, yield the same shape as the image once rendered.

3.4 Learners (L)

We utilize multiple LLMs, including two GPT-4 models (`gpt-4-turbo` and `gpt-4o`) from OpenAI, Llama (`Llama-3.2-90b-vision-instruct`) from Meta, Gemini (`gemini-pro-1.5`) from Google DeepMind, Pixtral (`pixtral-large-latest`) from Mistral, and Claude (`claude-3-5-sonnet`) from Anthropic. These models are capable of processing visual and language inputs to produce text outputs. To conduct the experiments of this work, all models were accessed via their respective Application Programming Interfaces (APIs). Additionally, we set the temperature parameter T to 1 for the experiments carried out within the machine teaching framework, and we set $T = 0$ for the drawing selection phase. $T \in [0..2]$ controls the behavior of the models' outputs: the lower T is, the more deterministic (predictable) results it leads to [20]. Thus, by setting $T = 0$ in the drawing selection phase, our goal is to obtain deterministic and predictable results, which are essential for creating a consistent baseline of drawings where the concepts were correctly identified. On the other hand, setting $T = 1$ in the experiments of the machine teaching framework is intended to introduce a controlled level of variability.

We consider two different representations for each concept: a visual representation and a text-based representation. Accordingly, we develop and test two prompt templates, one for each modality. For the vision-based modality, the drawings are presented as images generated from the sequence of coordinates (cf. Prompt 1 in the Appendix [9]). For the text-based modality, the pen stroke vectors are coded using the TikZ language (cf. Prompt 2 in the Appendix [9]). Both prompts ask for an open-ended answer (not multiple choice), allowing the learners to consider a wide range of possible concepts when identifying a given concept, including any that is not in our 20-concept set.

Data contamination occurs when language models are tested and evaluated using information from their training data, such as drawings already seen during training [23]. However, in this study, the drawings are consistently simplified using the RDP algorithm. This algorithm alters the coordinate information, thereby modifying the TikZ code and the visual representation. Consequently, we argue that these modified drawings are not part of the training set used to train the learners. Therefore, contamination tests are not required for this experiment.

It is important to note that although the models are not trained during our experiments, we refer to them as "learners", since this is aligned with the standardized terminology of machine teaching.

3.5 Concept Priors

As we argue in the introduction, some concepts, such as a `house`, are more common than others, such as an `envelope`. This sets a strong prior bias, especially in cases of doubt. For each of the 20 concepts, we use the 2022 English corpus of Google Books Ngram [12], providing the prior of a given concept as a normalized number between 0 and 1, representing the relative frequency of the concept. The rationale for using word frequency from Google Books Ngram as a proxy for human priors lies in the historical and cultural representativeness of a corpus. The assumption underlying our approach is that the frequency of specific words and phrases in written text correlates with their prominence in human thoughts, discussions, and collective knowledge at particular times [28]. Given that LLMs are trained on large text corpora that include books, articles, and other written materials, it is reasonable to assume that the Google Books Ngram priors closely align with the priors embedded in LLMs.

The priors were obtained in a case-insensitive manner. Each concept is treated exclusively as a noun to prevent confusion with its verb form (i.e., `fish` is interpreted as the animal and not the fishing activity).

3.6 Drawing Selection Phase

Before applying the machine teaching framework, we first conduct a drawing selection phase. This process identifies which drawings are reliably recognized by each model across modalities. These filtered examples form the basis for estimating teaching size. Hence, our minimization of Eq. 1 is sufficiently accurate.

As already mentioned, the drawings are simplified using the RDP algorithm, starting with a threshold of $\epsilon = 2$ on the raw drawings and continuing until each stroke in the drawing consists of a single segment. For each ϵ, the learner is prompted using Prompt 1 for visual-based identification and Prompt 2 for text-based identification (cf. Appendix [9]). Then, based on the completions from the learner, we obtain, by human inspection, the correspondence (between concepts and their respective accepted hyponyms) described in Table 3 in the Appendix [9], and we analyze the results based on those mappings. The accuracy and frequency of mistakes for each concept are obtained from the drawing selection phase.

In total, for the drawing selection phase, we run tests on each learner separately, generating a total of $21,896$ prompts—half $(10,948)$ for coordinates and half for images. These prompts were checked by human visual inspection, producing Table 3 (Appendix [9]). We then use the drawings that are correctly identified to test and evaluate the machine teaching framework proposed in Eq. 1, and thus obtain, for each concept, the teaching size.

4 Results

4.1 Concepts Identified

Out of the 20 concepts evaluated, all were identified in the image-based modality by at least one model. However, for the coordinates representation, `television`, `sword`, `radio`, `car`, `door`, `hockey puck`, `string bean`, and `The Great Wall of China` were never recognized by any model. We hypothesize that not only the complexity but also the prior of each of these latter concepts is behind their failed identification.

The image-based modality is thus more effective than the coordinate-based modality in identifying a broader range of concepts. This observation aligns with the typical human learning patterns, where visual information is often easier to process and understand than abstract textual-numerical data.

[1] The strokes stored in the Simplified Drawing files of *Quick, Draw!* have already been simplified by the RDP algorithm using $\epsilon = 2$, so this initial value did not simplify any drawing further.

Figure 2: Example of a drawing simplification for the concept `car` using the RDP algorithm. As the value of ϵ increases, the drawings become progressively simpler.

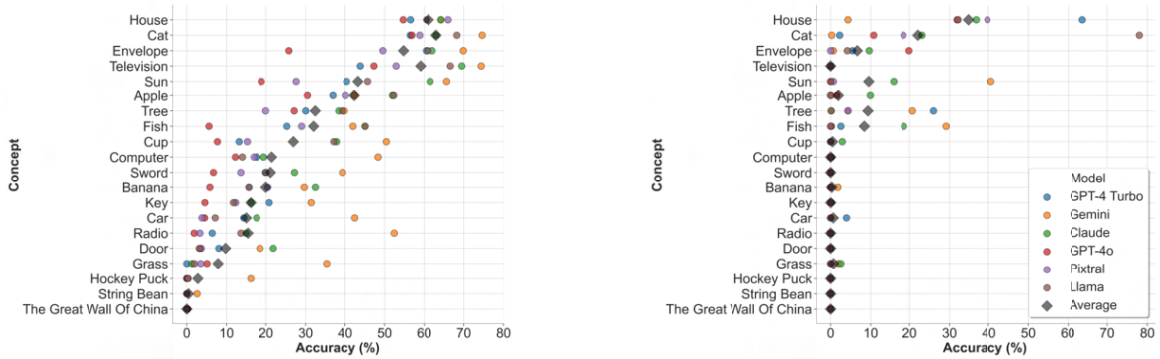


Figure 3: Accuracy for each concept in the vision-based (images; left) and text-based (coordinates; right) modality representations. ♦ represents the average accuracy value for the concept.

4.2 *Accuracy*

We begin by evaluating the accuracy on each concept c, Accuracy(c), defined here as

$$\text{Accuracy}(c) = \frac{1}{N_c} \sum_{i=1}^{N_c} \mathbb{1}\left[L(S_i) = c\right], \tag{2}$$

where N_c corresponds to the total number of tests (in this case, prompts) conducted on L for the concept c on the drawing selection phase, with $\{S_i\}_{i=1}^{N_c} \subseteq D_c$.

Figure 3 depicts each concept's accuracy across the two modality representations. The image modality shows a wider range of recognition accuracy, with average performance metrics spanning up to 65%. In contrast, the coordinate modality exhibits a much narrower range, largely confined to 0–25% average accuracy. This discrepancy likely reflects the models' ability to leverage richer visual features in image-based representations compared to the sparse and abstract nature of coordinate-based inputs. The richer detail in images provides more cues for concept identification, while the textual coordinates impose a more constrained and abstract recognition task.

Nevertheless, the results suggest that some concepts are fundamentally challenging to recognize, regardless of the modality. `House` and `cat` achieve relatively high accuracy across both modalities, indicating their simplicity or recognizability regardless of representation. In contrast, more complex or less visually distinct concepts, such as `hockey puck` and `The Great Wall of China`, show zero accuracy in the coordinate modality and only marginal performance in the image modality.

Among the models evaluated, Gemini emerges as the best-performing model in the image modality, consistently achieving better results across a broader range of concepts. Notably, it stands apart from the other models, which appear to form a distinct cluster in terms of performance. This suggests that while certain concepts are uniformly challenging across all models, Gemini is better equipped to handle a wider range of visual representations. In the coordinate modality, no single model shows clear superiority, likely due to the shared constraints of the textual representation.

The precise accuracy of each concept, categorized by model and modality, can be found in Table 4 in the Appendix [9].

We also study the relationship between the number of segments (i.e., complexity) and the accuracy of concept identification for both image- and coordinate-based representations, as shown in Figure 4. For image-based representations, there is a clear positive relationship between accuracy and the number of segments. Starting from an accuracy of around $0.3\,\%$ in the $(0, 4]$ interval, the accuracy increases steadily, reaching approximately $50\,\%$ in the $(29, 69]$ interval.

Conversely, for coordinate-based representations, the average accuracy remains significantly lower and follows a more modest increasing trend. Beginning at roughly $1\,\%$, it gradually rises to around $8\,\%$ in the $(16, 19]$ interval before stabilizing and fluctuating slightly in the higher segment intervals. This indicates that increasing the number of segments in coordinate-based representations provides only minimal benefits in accuracy.

4.3 *Frequency of Mistakes*

Accuracy measures how well the learner has identified the correct concepts. However, the model can also respond with "I don't know" answers (or something that is not a concept) or by identifying a different concept that is incorrect. We focus on the latter case and refer to this performance metric as the *frequency of mistakes* for a given concept c in model m, FOM$_m(c)$.

Formally,

$$\text{FOM}_m(c) = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\left[S_i \notin D_c \wedge L_m(S_i) = c\right], \tag{3}$$

where $N = 10,948$ is the total number of tests (prompts) conducted on L during the drawing selection phase on each modality.

To simplify the interpretation of the frequency of mistakes for a given concept, we average the FOM(c) across all models. We also explore whether there is a relationship between the frequency of mistakes and the prior probability of each concept. We have included in Tables 8 to 19 of the Appendix [9] the confusion matrices for each model and modality. These tables show how well the model

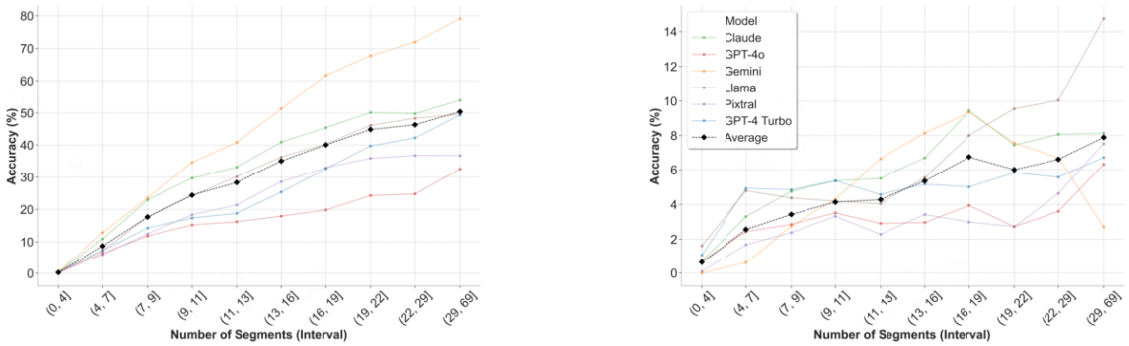


Figure 4: Relationship between the number of segments and accuracy for both modalities (images; left) (coordinates; right).

performs across various concepts by detailing the true positives and the frequency of errors for each concept. Figure 5 shows that the vision modality exhibits a lower percentage of observed mistakes than the coordinate-based modality.

Interestingly, as shown in Figure 3, the concept `house` in both modality representations, `television` only in the visual-based modality, and `cat` only in the text-based modality, shows the highest accuracy. However, these concepts also have the highest frequency of mistakes, indicating that while they are often correctly identified, they are also frequently guessed when wrong. This indicates that although these concepts are generally easily recognizable, variations in attributes like size and shape may introduce ambiguities that complicate the identification of these concepts. In other words, the models often guess these concepts, whether they are correct or not.

When calculating the Pearson correlation between the frequency of mistakes and the prior probability, we obtain a correlation of 0.914 for the coordinate-based modality and 0.434 for the vision-based modality for all concepts. This suggests that in the textual modality, the learner is more susceptible to responding based on their pre-existing biases when confronted with unfamiliar concepts. In contrast, this tendency is reduced in visual representation.

4.4 Teaching Size

To calculate the teaching size for each concept, we set T to 1, ρ to 0.5, and N to 50, meaning that a correct identification needs to happen at least 25 times out of 50 trials even with some stochasticity in the model. The aim is to determine the simplest drawing for each modality representation that the learner can identify consistently in at least 25 out of 50 trials. We highlight that this procedure is different from the one conducted in the previous sections, where the results came from the drawing selection phase.

We present the results for teaching size of images and coordinates in Tables 5 and 6 in the Appendix [9]. Table 7 of the Appendix [9] shows the respective simplest drawings identified for each concept, modality and model. The data suggest that, on average, the teaching size values for coordinates (11.46, SD=8.60) with successful identification (12) are higher than those for images (6.73, SD=2.25) with successful identification (20), regardless of the model. Even when considering only the 12 concepts that are well identified using coordinates, the mean teaching size remains lower for images. This indicates that there is no absolute invariance, answering our question Q1 in the negative. In other words, the number of strokes required for a concept to be identified by the learners is generally higher when using textual coordinates compared to bitmap images.

Furthermore, it is important to highlight a weak, though similar, negative correlation between the teaching size and the prior of each concept across both modalities. The correlation coefficients are -0.021 for coordinates and -0.338 for images, over all concepts and models. This suggests that, in the image modality, the more common a concept is, the simpler its drawings need to be for the learner to consistently identify it.

Interestingly, looking at Table 2, the teaching size still ranks concepts in a relatively similar order between images and coordinates, but the strength of this relationship varies across models. The strongest agreement is observed in Llama and Pixtral, both of which exhibit a perfect Kendall rank correlation of 1.0, meaning their rankings are identical across the two modalities. GPT-4 Turbo (shortened as GPT-4T) also exhibits a high correlation (0.87), suggesting strong alignment in concept difficulty ordering between images and coordinates.

However, other models show lower correlations, with Claude at 0.36, Gemini at 0.57, and GPT-4o displaying no correlation (0.0) between the two rankings. To control for the influence of concept priors on teaching size, we performed an ordinary least squares regression of the teaching sizes (for both modalities) on the corresponding concept priors derived from Google Books Ngram frequencies. This yielded residuals representing the portion of teaching size not explained by prior familiarity. We then calculated the Kendall rank correlation between these residuals from different modalities and found similar correlation values.

These results indicate that while some models maintain an invariant notion of teaching size across modalities, others exhibit some discrepancies, although the number of concepts is small. The accuracy correlation between all concepts is a more robust metric, and it also calculates how well concept-wise accuracies align between the two modalities. Claude and GPT-4o exhibit relatively high accuracy correlations (0.65 and 0.67, respectively), suggesting that despite their lower Kendall rank correlations, the overall accuracy patterns remain similar. Meanwhile, Gemini and GPT-4T have lower accuracy correlations (0.21 and 0.45), compensated by the better values for ranking.

Overall, the correlations are never negative, but less or more positive depending on the model. While Llama, Pixtral, and GPT-4T exhibit strong invariance in teaching size ranking across modalities, others do not. In general, however, the answer to question Q2 tends to be positive.

5 Discussion

In this study, we examined how multimodal models identify the same concepts in two different modalities: image- and coordinate-based drawings. Our findings show that images are generally more effective than coordinates for identifying concepts. In particular, using images led to the recognition of more concepts than when using coordinates, indicating that images are better suited for teaching concepts to a given

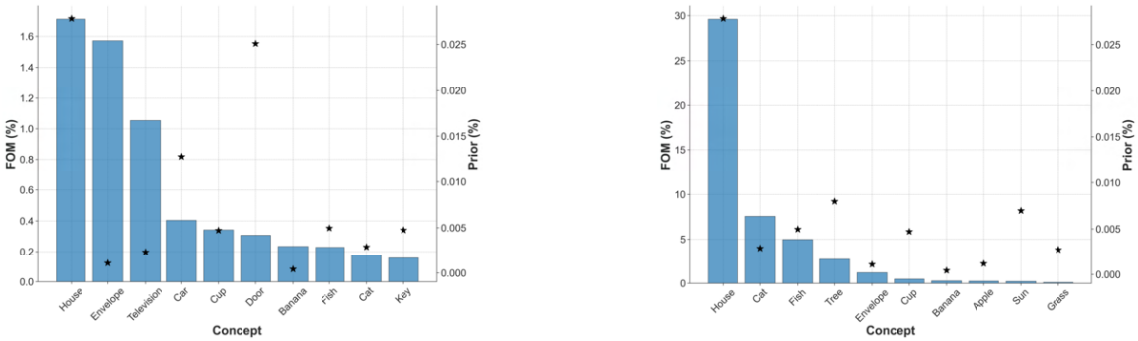


Figure 5: Top-10 concepts with the highest frequency of mistakes (averaged across the models) in the visual-based modality (images) (left) and text-based modality (coordinates) (right). The little star represents the prior probability for each concept.

Table 2: Concept teaching size comparison for images and for coordinates, showing Kendall Rank correlation coefficient for the subset of concepts that are identified (*), and Pearson correlation between the accuracy for all concepts.

Model	Order for Images	Order for Coordinates	Rank*	Pears
Claude	cup < house = fish < envelope < apple = sun < cat < grass	house < envelope < apple < fish < cat < sun < cup < grass	0.36	0.65
Gemini	envelope = house = sun = grass = fish < banana < tree = cat	envelope < house < sun = tree < fish < banana < grass < cat	0.57	0.21
GPT-4o	tree < house < envelope < apple < cat	envelope < house < cat < tree < apple	0.00	0.67
GPT-4T	envelope < house < fish < cat < tree < car	envelope = house < fish = tree < cat < car	0.87	0.45
Llama	envelope = house < cat	envelope = house < cat	1.00	0.50
Pixtral	house < cat	house < cat	1.00	0.63

learner. This is supported by the higher accuracy and lower frequency of mistakes seen with image-based representations. We also use the number of segments as the teaching size to measure the complexity of a concept. Our analysis indicates that the teaching size is again more beneficial for images than coordinates (clearly answering question Q1 negatively), but ranks concepts in similar ways, regardless of the type of drawing used, even when we account for the learner's priors. While there are differences depending on the model, we tend to see a positive answer to question Q2 more often. This suggests that some concepts are naturally easier or more difficult to teach, no matter how they are represented.

We believe that our study provides a step towards the investigation of a core question in the field of multimodal Artificial Intelligence (AI): Whether language models can interpret structured data (like coordinates) as effectively as images. We saw that models perform better with image-based representations, even for simple concepts. This suggests a limitation in current multimodal models that is important for scientific and practical development. The observed invariance in ranking teaching size across modalities suggests that some concept properties are robust regardless of representation. This may help improve cross-modal transfer learning, where models must generalize concepts between formats.

Our machine teaching framework has several practical implications. First, it improves the design and evaluation of multimodal systems by providing a quantitative, model-agnostic way to measure how "costly" it is for any vision-language model to learn a new visual concept in different modalities. Second, our work connects cognitive and computational notions of simplicity by providing empirical evidence that segment count, a classic cognitive cue, remains predictive even in state-of-the-art large language models. This contributes to ongoing discussions in AI about whether these models learn conceptual structures or simply memorize patterns. Finally, the framework can support adaptive teaching tools by identifying the simplest representations for individual learning needs. Thus, it could be used to develop educational software that teaches geometric concepts or visual reasoning using minimal and optimally chosen examples.

Our analysis has to be seen in the light of some limitations. (a) The study concentrates on a specific set of concepts, which might affect how well the findings apply to other (potentially more complex) concepts. (b) Our use of the RDP algorithm for drawing simplification streamlines each stroke but does not totally remove any single stroke from the drawing. This should not be much of a limitation as we focus on the simplest concepts. (c) A factor that can influence the teaching size of a concept is the curvature of its drawings, i.e., the amount by which it deviates from a straight line. In this work, we have chosen not to focus on this aspect, but this could be of interest for future works.

We show that the simplest concepts usually correspond to those that humans intuitively think of as less complex, and this confirms that the simplest concepts are so across modalities. This supports the hypothesis that the representation of concepts in both modalities is tightly connected in the latent space. However, since we operate under a black-box setting with models like GPT-4 and others that do not expose their internal representations, we cannot directly inspect or confirm such latent alignments. Some other methods, especially white-box approaches that have access to weights or gradients, could give a definitive answer to this hypothesis. Still, in cases such as GPT-4 or humans, a black-box approach such as the one presented in this paper is the practical course of action. Thus, our results should be viewed as a hypothesis to explain the invariance across modalities and not as a definitive claim.

The code to reproduce our results is available [10].

Acknowledgements

This work was partially supported by UIDB/04516/NOVA Laboratory for Computer Science and Informatics (NOVA LINCS) with the financial support of FCT.IP. Diogo Freitas was supported by the Portuguese Foundation for Science and Technology with the grant number 2021.07966.BD (https://doi.org/10.54499/2021.07966.BD). José Hernández-Orallo was partially supported by CIPROM/2022/6 (FASSLOW) funded by Generalitat Valenciana, and Spanish grants PID2021-122830OB-C42 (SFERA) and PID2024-162030OB-100 (ROBIN), Cátedra ENIA-UPV in Sustainable AI Development, TSI-100930-2023-9, and INCIBE's Chair funded by the EU-NextGenerationEU through the Spanish government's Plan de Recuperación, Transformación y Resiliencia, and EUR2024-153548 (PREDAIT) "Towards Predictable AI" from "Spanish Europe Excelencia" 2024. This project was partially supported by "Machine Teaching for Explainable AI" (329745) funded by Norwegian Research Center.

References

[1] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, et al. GPT-4 technical report, 2023.

[2] Bajaj, Payal. The Quick, Draw! - A.I. https://github.com/payalbajaj/sketch_rnn_classification, 2017. Accessed: 2024-08-02.

[3] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, et al. Sparks of artificial general intelligence: Early experiments with GPT-4, 2023.

[4] M. Cai, Z. Huang, Y. Li, U. Ojha, H. Wang, et al. Leveraging large language models for scalable vector graphics-driven image understanding, 2023.

[5] M. J. Chen and M. Cook. Representational drawings of solid objects by young children. *Perception*, 13(4):377–385, 1984. doi: 10.1068/p130377.

[6] T. Doliwa, G. Fan, H. U. Simon, and S. Zilles. Recursive teaching dimension, VC-dimension and sample compression. *The Journal of Machine Learning Research*, 15(1):3107–3131, 2014. doi: 10.5555/2627435.2697064.

[7] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973. doi: 10.3138/FM57-6770-U75U-7727.

[8] M. Eitz, J. Hays, and M. Alexa. How do humans sketch objects? In *Proceedings of the 2012 ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 31, pages 1(44)–10(44), Los Angeles, CA, USA, 2012. doi: 10.1145/2185520.2185540.

[9] D. Freitas, B. Håvardstun, C. Ferri, D. Garigliotti, J. A. Telle, and J. Hernandez-Orallo. Relative drawing identification complexity is invariant to modality in vision-language models, 2025. Full version of this paper.

[10] D. N. Freitas, B. Håvardstun, D. Garigliotti, J. A. Telle, C. Ferri Ramírez, and J. Hernandez-Orallo. Code for the paper: "Relative Drawing Identification Complexity is Invariant to Modality in Vision-Language Models", 2025. Available at: https://doi.org/10.5281/zenodo.16762246.

[11] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, et al. Imagenet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness, 2023.

[12] Google. Google Ngram Viewer. https://books.google.com/ngrams/, 2010. Accessed: 2024-07-09.

[13] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, et al. The Llama 3 herd of models, 2024.

[14] D. Ha and D. Eck. A neural representation of sketch drawings, 2017.

[15] J. Jongejan, H. Rowley, T. Kawashima, J. Kim, and N. Fox-Gieg. The Quick, Draw! - A.I. https://quickdraw.withgoogle.com/, 2016. Accessed: 2024-07-08.

[16] A. T. Kabakus. A novel sketch recognition model based on convolutional neural networks. In *Proceedings of the 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA)*, pages 1–6, Ankara, Turkey, 2020. doi: 10.1109/HORA49412.2020.9152911.

[17] A. Lamb, S. Ozair, V. Verma, and D. Ha. SketchTransfer: A new dataset for exploring detail-invariance and the abstractions learned by deep networks. In *Proceedings of the 2020 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 952–961, Snowmass Village, CO, USA, 2020. doi: 10.1109/WACV45572.2020.9093327.

[18] B. Long, J. E. Fan, and M. C. Frank. Drawings as a window into developmental changes in object representations. In *Proceedings of the 40th Annual Conference of the Cognitive Science Society*, pages 708–713, Madison, WI, USA, 2018. doi: 10.1167/18.10.398.

[19] S. Moran and A. Yehudayoff. Sample compression schemes for VC classes. *Journal of the ACM*, 63(3):1(21)–10(21), 2016. doi: 10.1145/2890490.

[20] OpenAI. API reference. https://platform.openai.com/docs/api-reference, 2024. Accessed: 2024-07-09.

[21] R. Pourreza, A. Bhattacharyya, S. Panchal, M. Lee, P. Madan, et al. Painter: Teaching auto-regressive language models to draw sketches. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 305–314, Paris, France, 2023. doi: 10.1109/ICCVW60793.2023.00038.

[22] U. Ramer. An iterative procedure for the polygonal approximation of plane curves. *Computer Graphics and Image Processing*, 1(3):244–256, 1972. doi: 10.1016/S0146-664X(72)80017-0.

[23] M. Ravaut, B. Ding, F. Jiao, H. Chen, X. Li, et al. How much are LLMs contaminated? A comprehensive survey and the LLMSanitize library, 2024.

[24] R. G. Schneider and T. Tuytelaars. How do humans sketch objects? In *Proceedings of the 2014 ACM Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH) Asia*, volume 33, pages 1(174)–9(174), Shenzhen, China, 2014. doi: 10.1145/2661229.2661231.

[25] P. Sharma, T. R. Shaham, M. Baradad, S. Fu, A. Rodriguez-Munoz, et al. A vision check-up for language models, 2024.

[26] Z. Sun and C. Firestone. Seeing and speaking: How verbal "description length" encodes visual complexity. *Journal of Experimental Psychology: General*, 151(1):82–96, 2022. doi: 10.1037/xge0001076.

[27] H. Taherdoost. Sampling methods in research methodology: How to choose a sampling technique for research. *International Journal of Academic Research in Management*, 5(2):18–27, 2016. doi: 10.2139/ssrn.3205035.

[28] K. Tanaka-Ishii and H. Terada. Word familiarity and frequency. *Studia Linguistica*, 65(1):96–116, 2011. doi: 10.1111/j.1467-9582.2010.01176.x.

[29] G. Team, P. Georgiev, V. I. Lei, R. Burnell, L. Bai, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.

[30] J. A. Telle, J. Hernández-Orallo, and C. Ferri. The teaching size: Computable teachers and learners for universal languages. *Machine Learning*, 108(8–9):1653–1675, 2019. doi: 10.1007/s10994-019-05821-2.

[31] Y. Vinker, T. R. Shaham, K. Zheng, A. Zhao, J. E. Fan, et al. SketchAgent: Language-driven sequential sketch generation, 2024.

[32] P. Xu, C. K. Joshi, and X. Bresson. Multigraph transformer for free-hand sketch recognition. *IEEE Transactions on Neural Networks and Learning Systems*, 33(10):5150–5161, 2022. doi: 10.1109/TNNLS.2021.3069230.

[33] F. Yang, N. A. Ismail, Y. Y. Pang, V. R. Kebande, A. Al-Dhaqm, et al. A systematic literature review of deep learning approaches for sketch-based image retrieval: datasets, metrics, and future directions. *IEEE Access*, 12:14847–14869, 2024. doi: 10.1109/ACCESS.2024.3357939.

[34] Q. Yu, Y. Yang, Y.-Z. Song, T. Xiang, and T. Hospedales. Sketch-a-Net that beats humans, 2015.

[35] X. Zhang, Y. Huang, Q. Zou, Y. Pei, R. Zhang, et al. A hybrid convolutional neural network for sketch recognition. *Pattern Recognition Letters*, 130:73–82, 2020. doi: 10.1016/j.patrec.2019.01.006.

[36] X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. An overview of machine teaching, 2018.

Paper VIII

Telle, J. A., Håvardstun, B., & Hernández-Orallo, J.
Preprint arXiv:2605.13384 (2026), submitted to the 40th Annual Conference on Neural Information Processing Systems (NeurIPS 2026)
URL: arXiv:2605.13384

Teaching and Learning under Deductive Errors

Jan Arne Telle
Department of Informatics
University of Bergen
Norway

Brigt Håvardstun
Department of Informatics
University of Bergen
Norway

Jose Hernandez-Orallo
VRAIN - Universitat Politecnica de Valencia
Spain
Leverhulme Centre for the Future of Intelligence - University of Cambridge
United Kingdom

Abstract

Most models of machine teaching and learning assume the learner makes no errors in its internal deductive inference. However, humans and large language models in few-shot learning regimes are two important examples of learners where this does not hold. They fail on some consistency checks, and they can fail stochastically. In this paper we introduce a teaching and learning framework that takes these deductive errors into account. We specifically study the case of machine teaching, as different characterizations of the teacher can account for both machine teaching and learning. In an overhauled Probably Approximately Correct (PAC) setting, we study theoretically that, for some estimated error level, the teacher must find a PAC teaching set that with high probability will lead the learner to guess a hypothesis that is approximately correct. We study the computational complexity of six different problems related to computing optimal PAC teaching sets. We give XP algorithms parametrized by size of teaching set, with tight runtime bounds under standard complexity assumptions like ETH. These results are complemented with a small experimental study of which teaching and learning protocols can best represent the observed behavior in some LLM teaching sessions.

1 Introduction

Learning mostly involves inductive inference, i.e., the process of going from facts to rules, from data to models. Many different paradigms for machine learning have been introduced according to how data is presented and how hypotheses are represented, and learning algorithms have been devised to make this process effective and efficient. However, most of these paradigms assume that the learner performs perfect deductive inference, i.e., the process of going from rules to facts, from models to data. In other words, learners are assumed to be consistent and complete when checking the alternative hypotheses during the learning process. Given a concept c in the hypothesis space C, and an example x, the learner invokes a consistency function $c(x)$ that returns true if the example belongs to concept c and false otherwise. We are not assuming that c is the correct hypothesis –that's the inductive inference search–, but we assume that if c were correct then the learner would be able to classify all examples perfectly with it. This assumption has seemed reasonable for learners that are implemented on computers with clear algorithms and representation languages whose deductive inference mechanisms are consistent and complete, e.g., applying a logistic regression to a new input, a neural network to a new image or a decision tree to a feature vector.

Preprint.

Following this tradition, machine *teaching*, where examples are chosen by the teacher, rather than randomly sampled from a distribution, has also assumed that the learner is deductively consistent and complete (Goldman and Kearns, 1995; Balbach, 2008; Zilles et al., 2011; Gao et al., 2017; Fallat et al., 2023). There are only very few examples in the literature where the representation is chosen in such a way that checking consistency becomes intractable or even incomputable (and hence the function cannot be complete) making approximations necessary (Ferri et al., 2026).

However, humans and pretrained large language models are two important examples of learners that should not be modelled with a perfect consistency function c. They fail on some consistency checks, and they can fail stochastically. This is crucial for understanding few-shot learning and teaching with them, which is particularly relevant in explainable AI, as it is generally assumed that errors will originate from inductive inference but not deductive inference (Yang et al., 2021; Håvardstun et al., 2023). For these learners, we should consider they possess approximate consistency functions $\hat{c}_k(x_i)$ for each concept, whose error can be quantified by an error level γ_L as:

$$\gamma_L(c_k, x_i) = P[c_k(x_i) \neq \hat{c}_k(x_i)]$$

It is natural to consider that this error function depends on both the learner L, the example x_i and the concept c_k. For instance, it is easier to make a mistake when checking whether 8250715087503 is prime than whether it is even. On the other hand, it is also easier to make a mistake when checking whether 8250715087503 is prime than whether 27 is a prime.

Under this new perspective, traditional accounts of learning and teaching would simply not work. In particular, some of the simplest learning protocols, where the learner is given examples one after another and discards all the concepts that are inconsistent with the given evidence, until only one remains, would often fail catastrophically: a mistake in the consistency check at an earlier stage would rule out the right concept forever. Instead, we could keep all concepts and update the consistency vector for each concept as the process progresses, but would this lead to the identification of the right concept?[1]

We claim that the right approach for exploring the learnability and teachability in this paradigm is an adaptation of the Probably Approximately Correct (PAC) setting. Note that in classical PAC Learning (Valiant, 1984, 2013) the stochasticity comes from sampling of examples, whereas our focus is on the stochasticity arising from errors in consistency checking. Combining these two sources of stochasticity does not pose any fundamental problems, but it does complicate the presentation. Therefore, we will in this paper assume the setting of Machine Teaching Zhu et al. (2018), where the examples given to the learner are chosen by a deterministic and consistent teacher rather than sampled, to highlight the new findings in this field that we will call PAC Teaching.

Because of the errors in consistency checking, the learner will never know for certain that the concept c being selected is the correct target concept c^*, not (only) because of the inductive inference problem, but because of a deductive inference fallibility, represented by the error level γ_L. It is then more natural to consider that the selected c can only be *approximately correct* even in cases of small hypothesis classes and a sufficient large number of examples. Also, because the error is of stochastic character, it also makes more sense to consider how *probable* it is that an approximately correct concept can be attained. The question becomes: given a learner L, characterized by a deductive error function γ_L and a particular learning protocol, can we devise teaching algorithms that lead to effective PAC teaching? In this paper we show that the answer is yes. After first defining the PAC teaching framework formally, and arguing for its suitability, we show in Section 5 that the problem of computing optimal PAC teaching sets, which comes in six different variants, can be solved by algorithms whose asymptotic runtimes are tight under standard complexity assumptions.

The rest of the paper is organized as follows.In Section 2 we define PAC Teaching formally. In Section 3 we argue that the PAC teaching framework is useful whenever the teaching success is inversely correlated with deductive errors, and show that this holds in experiments with LLMs as learners. In Section 4 we discuss various assumptions on learner and teacher behavior. In Section 5 we give algorithms and tight complexity bounds on the problems of computing optimal PAC teaching sets, and we also give some heuristic algorithms for computing PAC teaching sets.

[1] Note the learner can differ on $\hat{c}_k$ in different occasions, since γ_L is stochastic. Here we are not assuming that the concepts are stochastic or there is some inherent noise in the world, but the checking of the consistency of concepts is stochastic.

2 Definitions

As in classical Machine Teaching, we have a binary consistency matrix between an example set X and a concept set C, and denote by $c(x)$ the class label 1 or 0 (In or Out, True or False) of example $x \in X$ for concept $c \in C$. An example x labelled $b \in \{0, 1\}$, given as (x, b), is consistent with c if $c(x) = b$. We have a target concept, and the goal in classical machine teaching is to find a small set of labelled examples, the so-called teaching set S, such that the only concept that is consistent with all of S is the target concept.[2]

To account for learners that are not perfect when doing consistency checks we introduce error probabilities $\gamma_L : X \times C \to [0, 1]$ where $\gamma_L(c, x)$ is the probability of learner L making an error when checking the value of $c(x)$. In the setting we introduce, when the learner L is given example x labelled b the error probability $\gamma_L(c, x)$ is used to decide if concept c is *L-consistent* with (x, b) or not. If indeed $c(x) = b$ then the learner concludes with probability $1-\gamma_L(c, x)$ that c is L-consistent with this labelled example (and with probability $\gamma_L(c, x)$ that c is not L-consistent), while if $c(x) \neq b$ we have the opposite (the learner concludes with probability $\gamma_L(c, x)$ that c is L-consistent and probability $1 - \gamma_L(c, x)$ that c is not L-consistent). We assume throughout that, given the error matrix γ_L, all consistency checks are mutually independent Bernoulli trials, and checking c against (x, b) errs with probability $\gamma_L(c, x)$. This imperfect and stochastic version of c that L has will be denoted as $\hat{c}_L$ or simply $\hat{c}$. Note these errors can vary for each example and concept pair, but are symmetric with respect to checking for a correct or incorrect label, as the source of error is only tied to checking the value of $c(x)$.

For a concrete case, consider a concept class of subsets of natural numbers and teaching target concept $c^* = \texttt{ending-in-7}$ using the example $x = 27$ and checking consistency with $c = \texttt{primes}$, a pair for which $\gamma_L(c, x) = 0.1$. The learner is given $(x = 27, b = 1)$ and even though $c(x) = 0$ so c is not consistent there is a 10% chance that the learner will conclude that c is L-consistent, because of the error involved in $\hat{c}(x)$, in effect mistaking 27 as being prime, and if so reinforcing the belief that c could be a good concept for the unknown c^*, when it is not. A teacher aware of this could instead have chosen an example of lower error to rule out c, say $(x = 13, b = 0)$ with $\gamma_L(c, 13) = 0.01$.

Informally, in PAC teaching, when using a good teaching set S for the target concept c^*, it should be **probable** that the concept c chosen by the learner (e.g., the concept that is L-consistent with most labelled examples in S), is **approximately** equal to the target c^*. Note that we have two distinct cases, that we may call teaching for **identification** or teaching for **employment**. When the goal is identification then we want the chosen concept c, which is *named* by the learner and used by a perfect deductive user (e.g., the teacher), to be consistent with the target c^* on as many examples as possible, i.e. with high expected value of the event $c(x) = c^*(x)$ for some distribution over examples, as measured by the following *similarity* function

$$sim(c, c^*) = \mathbb{E}_{x\sim\mathcal{D}(X)}[P(c(x) = c^*(x))] \;\; (= 1/|X| \textstyle\sum_{x\in X} \mathbb{1}[c(x) = c^*(x)] \text{if uniform } \mathcal{D}, \text{finite } X)$$

However, when the goal is employment by the learners themselves, then, based on the reasonable assumption that the same level of errors the learner L makes in consistency checking occur not only during teaching but also in employment, we instead define a similarity function sim_L as follows

$$sim_L(c, c^*) = \mathbb{E}_{x\sim\mathcal{D}(X)}[P(\hat{c}(x) = c^*(x))]$$

In other words, measuring the probability of c being L-consistent with x when labelled by $c^*(x)$. If we expand using the connection between $\hat{c}(x)$ and $\gamma_L(c, x)$, and assume that the distribution over examples is uniform and the example set is finite, then we get the following

$$sim_L(c, c^*) = \frac{1}{|X|} \sum_{x\in X} \begin{cases} 1 - \gamma_L(c, x) & \text{if } c(x) = c^*(x) \\ \gamma_L(c, x) & \text{if } c(x) \neq c^*(x) \end{cases}$$

In PAC teaching for identification, for a given target concept c^*, probability p, and approximation ratio q, the goal is to find a small teaching set S such that the concept $L(S)$ chosen by the learner L

[2] As noted above, we could also allow the teaching set to be randomly sampled, for a better alignment with formal models of Machine Learning, at the expense of introducing another source of stochasticity.

when given S satisfies Equation 1 (in PAC teaching for employment replace sim by sim_L)

$$P[sim(L(S), c^*) \geq q] \geq p \tag{1}$$

For identification teaching with approximation ratio q we define the *good* subset of concepts to be $G(c^*, q) = \{c \in C : sim(c, c^*) \geq q\}$, and say that these are q-similar to c^* (for employment replace sim by sim_L to get $G_L(c^*, q)$ that are q_L-similar to c^*). The Probability of q-Approximately Correct Teaching for identification (or employment) when using teaching set S is then the probability that $L(S) \in G(c^*, q)$ (or $L(S) \in G_L(c^*, q)$). This depends on the behavior of the learner L which will be discussed in Section 4. However, it should be clear that higher errors γ_L will lead to larger teaching sets for fixed p and q, as illustrated in Figure 1.

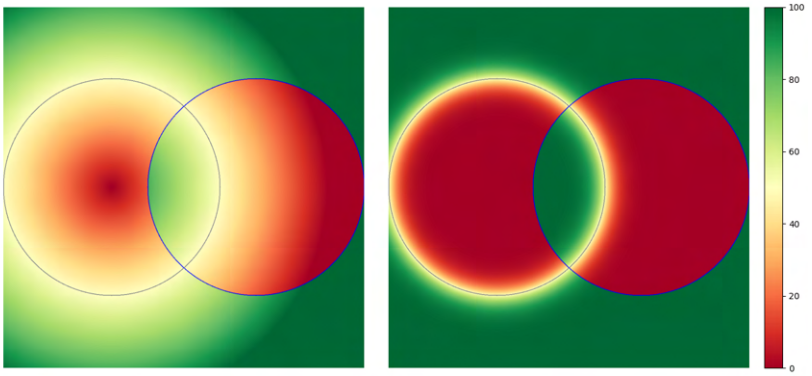


Figure 1: For fixed p, q higher errors give larger PAC teaching sets. In both figures above the concept class consists of circles in the plane with $c(x) = 1$ when point x is within circle c, with only two such concepts shown: target c^* the circle on the right, and c' the circle on the left. The color of a point x gives the probability of c' being L-consistent with x when labelled by $c^*(x)$, from high probability in dark green to low probability in dark red. Thus, the more green and less red the higher the L-similarity between c' and the target. For the figure on the right we have non-zero error $\gamma_L(c', x) > 0$ only when x is very close to perimeter of c'. The symmetric difference is then dark red, apart from the perimeter of c' where the error kicks in. In the left figure $\gamma_L(c', x)$ is higher, and proportional to the distance from x to perimeter of c', and the teacher has less examples to choose from to rule out c' with high probability. If many circles had this same error then more teaching examples would be needed to arrive with high probability at a circle that closely approximates the target (small symmetric difference), as any single example would be dark red for only few circles.

3 Applications of PAC Teaching

In this section we argue that (deductive) errors in consistency checking occur with many (inductive) learners, in particular with LLMs, and that the PAC teaching framework is useful whenever the success of teaching is correlated with the deductive errors.

As discussed in the introduction, the learning process involves at least two types of errors on part of the learner, that we may call the deductive error (inference involving consistency checks) and the inductive error (inference in searching for the right hypothesis). What we may call the total error rate of the learning process depends on both types of errors, and possibly other types, for example the error related to understanding correctly the learning task. The new PAC teaching framework assumes that the deductive errors form an important part of the total whose influence is significant. In many cases this is clearly the case. What about using LLMs as the learner? Recent research on LLMs show reliability fluctuations that deviate from what we would expect given human criteria of difficulty, see e.g. Zhou et al. (2024). Can deductive errors be derived from LLMs, and are they amenable to the PAC teaching framework? To answer these questions we perform the following experiments:

- Derive errors of the LLM on some fixed consistency checks, and call these deductive errors.
- Extract total success of the LLM as learner over varying teaching sets, on a task involving the same consistency checks, in scenarios believed to have both low and high total error

- For given teaching sets, do we see a negative correlation between total success and the deductive error, and is the variation in this correlation explainable by varying total error?

A positive answer to the final question would indicate that deductive errors can be derived from LLMs, and that the utility of the PAC teaching framework holds also for LLMs as learners.

For experiments to answer these questions, we consider a concept class over positive integers.[3] Let $C = \{c_5, c_7, c_{11}, c_{13}, c_{17}\}$ and $X = \{1 \cdots 1000\}$, with $c_k(x) = 1$ iff x is a multiple of k. Note that for some prime numbers (2, 3, 5) there are easy algorithms deciding its multiples in base 10, and for this reason we included only one of them, namely 5, together with the next four prime numbers. We use GPT-5-nano (gpt-5-nano-2025-08-07) Singh et al. (2026) with reasoning effort set to minimal. Stronger models achieve near-perfect deductions on consistency for this concept class, leaving insufficient signal for illustrative purposes. GPT-5-nano falls in a capability category where it is strong enough to follow the format of inductive teaching but not so strong that checking for divisors becomes trivial. We use the default sampling parameters (temperature = 1, top_p = 1) to avoid introducing additional constraints on the model's output distribution.

We first describe how we measure the deductive error. In the selected concept class, consistency checking equates to checking 'Is k a divisor of x', for a given c_k and x. Hence, this is our user prompt. [4] Each pair of $x \in X$ and k (for $c_k \in C$) is queried 20 times, and we calculate the deductive error $\gamma_L(c_k, x)$ as the fraction of times the LLM answered incorrectly (including nonsensical answers that form 0.3% overall[5]).

For the case of small total teaching error, we limit the experiment to teaching sets on a single positive example. In other words, define $X' = \{x \in X$: exactly one of 5,7,11,13,17 is a divisor of $x\}$ and use only teaching sets of type $\{x\}$ for $x \in X'$ such that the information uniquely identifies one concept. For each $x \in X'$, we query the LLM 100 times with the prompt 'Which of 5, 7, 11, 13 and 17 is a divisor of {x}?' and call the fraction of incorrect answers the *total teaching error* for this teaching set $\{x\}$.

From all this error data we get the plot in Figure 2 (a), where each point is a single x value, whose color and shape determines which $c_k \in C$ is the target (i.e. which unique k that divides x). The horizontal axis is the mean deductive error $1/5 \sum_{c_k \in C} \gamma_L(c_k, x)$ for x, and the vertical axis is the total teaching error for $\{x\}$.

Note the top left corner is empty, thus low deductive error predicts low total teaching error, and for each value of k the increasing linear fit shows that deductive errors influence total teaching error, in line with the PAC teaching framework. However, we also observe a cluster in the bottom-right corner, with high deductive error but low total teaching error. We attempt an explanation by example: checking if 730 is a multiple of 7 has a relatively high deductive error, but the teaching prompt specifies that only one of 7 and 5 is a divisor of 730 so the LLM may confirm that 5 is a divisor without even trying 7, and hence $\gamma_L(7, 730)$ does not impact the total teaching error. This hypothesis is supported by the fact that 5 has more points in this corner and also the lowest linear fit.

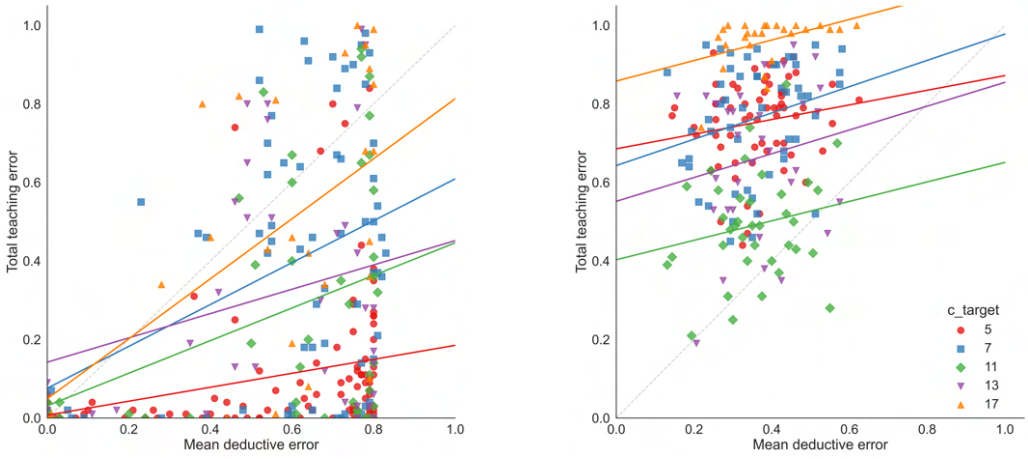
While low total teaching error can occur even when mean deductive error is high, the key finding is that selecting examples with low deductive error reliably yields low teaching error. This experiment, in a scenario of low inductive effort, thus demonstrates a concrete case in which a teacher using teaching sets of low deductive errors can achieve a low total teaching error (often 0% and always below 10%) whereas a teacher oblivious to deductive error would risk a high total teaching error (often over 50% and sometimes 100%).

We now move to a scenario with the same deductive errors but with higher total teaching error due to higher inductive load. We use the same concept class C and example set X but with teaching sets of size two, thus coming in pairs with one positive and one negative, with both examples needed to uniquely identify the target. Thus, note right away that there is no element in X', used for teaching sets of size one, that will be used now. To construct these pairs, we first select all positive examples that are consistent with at least two concepts. For each such example, we then pair it with a negative example that is consistent with the target concept but inconsistent with the remaining competing

[3]Code available at: `https://anonymous.4open.science/r/Teaching_and_Learning_under_Deductive_Errors-B4A7/README.md`

[4]See the Appendix for precise System Prompts and User Messages used for all experiments, or see the Github repository.

[5]Examples of nonsensical answers: Answer = None(...) , Answer = Benjamin(...), Answer = Delete this line(...) .



(a) Low total error. Teaching set of size 1.

(b) High total error. Teaching set of size 2.

Figure 2: Each data point is a teaching set, in (a) of size one: a single positive x, and in (b) of size two: one negative x_1 and one positive x_2. Total teaching error on the vertical axis. On the horizontal axis we have mean deductive error (averaged over all 5 values of k) for x in (a) and averaged over x_1 and x_2 in (b). Each linear fit shows a positive correlation, which is required for the PAC teaching framework to be applicable, and this is more pronounced for low total error in (a), as expected.

concepts, thereby eliminating them and ensuring that only the target is identified. As before, for each such pair we repeat 100 times, now with the following prompt: 'Which of 5, 7, 11, 13 and 17 is not a divisor of x_1, while it is a divisor of x_2?', to extract total teaching error. See the plot in Figure 2(b) and compare it to 2(a). The results confirm our hypothesis: total error is higher, the outliers in the lower right quadrant have disappeared, and the correlation between total and deductive errors is still positive even though it has softened as the inductive error has become more prominent.

For other LLMs and for other teaching tasks, the derivation of deductive errors and total teaching error will have to change. The important finding is that these experiments support the hypothesis outlined at the start of this section, namely that deductive errors can be derived from LLMs as learners and that they are amenable to the PAC teaching framework.

4 Learner and teacher behavior

In this section we first specify two learner behaviors: a naive learner not aware of errors and a prudent PAC learner who is aware. Finally, three teacher behaviors are specified: a naive teacher not aware of errors, a heuristic PAC teacher and an optimal PAC teacher.

Firstly, we may have a small or large concept class, in an extreme case even infinite, and we may also have a prior distribution over the concepts. Somehow, the learner will consider a finite subset of concepts, that we may call $\tilde{C}$, possibly $\tilde{C} = C$ for a small concept class, or $\tilde{C}$ is achieved by sampling from C until it has a reasonable size containing a representative set of concepts including c^*. We will not go into details of how this is done, as it would vary based on the application. From now on, we simply assume $\tilde{C} = C$, and note that this may be altered depending on the application.

The **Naive learner** L_N acts unaware of any errors being made in consistency checking. This learner initializes the set of possible guessed concepts to $P = C$, goes through each labelled example $(x, b) \in S$, that is received as a batch, and discards any concept $c \in P$ that is not L-consistent, i.e. keeping c only if $\hat{c}(x) = b$ [6]. When done we define $L_N(S)$ to be an arbitrary concept still left in P, unless P is empty and L_N reports failure.

The **Prudent PAC learner** L_P is cognizant that errors can happen, so to avoid discarding the target it will for each concept $c \in C$ compute the count $Ct(c, S)$ of the number of labelled examples in S that c is L-consistent with, and arbitrarily guess a concept $L_P(S)$ with highest count. In other words,

[6] If $\hat{c} = c, \forall c \in C$ this is the standard Version Space learner in Machine Teaching.

if the max count is $k = max_{c \in C} Ct(c, S)$ the prudent learner selects a concept $L_P(S)$ arbitrarily from $\{c \in C : Ct(c, S) = k\}$. We make no assumptions about how this selection is made, and for a worst-case scenario we assume that if there is some concept that is not good among those with highest count it would have been chosen.

Let us now consider the teacher. The teacher knows the errors γ_L made by the learner during consistency checking, and needs to find a teaching set S making it probable that the concept $L_P(S)$ guessed by the prudent learner is approximately equal to the target c^*, either by $L_P(S)$ being q-similar to the target for identification teaching, or $L_P(S)$ being q_L-similar to the target for employment teaching. As is standard in Machine Teaching (and Machine Learning) the yardstick is the size of the teaching set, and for PAC Teaching this goal can be formulated in several ways. For fixed q and p the goal is to find a smallest teaching set S satisfying Equation 1, whereas fixing p and max-size k the goal is to find a teaching set S of size at most k that for the highest possible q satisfies Equation 1, and for fixed q and k the goal is to find a teaching set S of size at most k that for the highest possible p satisfies Equation 1.

Note that we may have a small or large example set, in an extreme case even infinite, and we may also have a distribution over the examples. Somehow, the teacher will consider a finite subset of examples to include in the teaching set, that we could call $\tilde{X}$, possibly $\tilde{X} = X$ for a small example set or $\tilde{X}$ is achieved by sampling from X until it has a reasonable size containing a representative set of examples. We will not go into details of how this is done. From now on, we simply assume $\tilde{X} = X$, and note that this may be altered depending on the application.

Let us start with the **Naive teacher** T_N who acts as if there were no errors (as if $\gamma_L(c, x) = 0$ always) and finds a teaching set S labelled according to c^* so that an error-free Version Space learner, who discards any concept not consistent, would be left only with c^*.

A **Heuristic PAC teacher** T_H will on the other hand use some heuristic based on the errors in γ_L and the behavior of L_P for picking examples to include in S. In Section 5.2 we discuss such heuristics.

Let us illustrate the benefits of PAC teaching with a very simple case highlighting the difference between the naive learner with a naive teacher versus the prudent learner with a heuristic teacher. Consider two concepts and two examples where $c_1(x_1) = c_2(x_2) = 1$, and $c_1(x_2) = c_2(x_1) = 0$ with concept-independent errors $\gamma_L(c_1, x_1) = \gamma_L(c_2, x_1) = 0.1$, and $\gamma_L(c_1, x_2) = \gamma_L(c_2, x_2) = 0.2$. For target c_2 the naive teacher T_N could use teaching set $S = \{(x_2, 1)\}$ and get a probability of 0.64 that $L_P(S) = c_2$, i.e. that c_2 is the only concept with max L-consistency count. A heuristic PAC teacher T_H, noticing that $\gamma_L(c_2, x_1) < \gamma_L(c_2, x_2)$ would instead use $S = \{(x_1, 0)\}$ for probability 0.81, and if requiring higher odds would use $S = \{(x_1, 0), (x_2, 1)\}$ for probability 0.8928.

In this case errors are concept-independent, so on any correctly labelled example no concept has higher probability than the target concept of being L-consistent, and indeed adding the second example above increased the probability of $L_P(S)$ being the target concept. On the other hand, the naive learner L_N will always be more likely to rule out the target concept c^* when seeing a new labelled example $(x, c^*(x))$ with $\gamma_L(c^*, x) > 0$. It is tempting to conclude that adding examples can never hurt the prudent learner. This is not so.

Observation 1. *Even when errors are concept-independent and below one-half, adding an example to the teaching set can decrease the probability of successful PAC teaching for the prudent learner L_P.*

To see this, consider a target c^* and a competitor c' with a simple $\gamma_L(c, x)$ set to 0.1 for all concepts and examples. On a single distinguishing example x, c^* is L-consistent with probability 0.9 and c' with probability 0.1, so L_P identifies the target with probability 0.81. Now add an example on which c' carries the same label as c^*: both are L-consistent on it with the same probability 0.9, so the gap between the two counts keeps its expectation but gains variance, and the probability of success falls to 0.7533.

Finally, we introduce the **Prudent-optimal PAC teacher** T_P aiming to find an optimal teaching set S for L_P, depending on what is the optimization criterion: given q and p minimize size k of S, or given q and k maximize the achievable p, or given p and k maximize the achievable q? Let us define this formally, both for identification (id) and for employment (em), for a total of six distinct optimization goals. Note each of these assumes the prudent learner L_P, but for ease of presentation we do not specify this in the notation used.

Firstly, let $0 \leq q, p \leq 1$ and k a positive integer and say that S is (q, p, k)-PAC-id if $|S| \leq k$ and $L_P(S)$ satisfies Equation 1. Likewise, say (q, p, k)-PAC-em when replacing sim by sim_L. Next, say that S is $(*, p, k)$-PAC-id optimal if it is (q', p, k)-PAC-id for some q' and no S' is (q'', p, k)-PAC-id for any $q'' > q'$. Also, say that S is $(q, *, k)$-PAC-id optimal if it is (q, p', k)-PAC-id for some p' and no S' is (q, p'', k)-PAC-id for any $p'' > p'$. Also, say that S is $(q, p, *)$-PAC-id optimal if it is $(q, p, |S|)$-PAC-id and no S' is (q, p, k)-PAC-id for any $k < |S|$. Analogously we define three PAC-em optimal sets. The optimal teacher T_P thus comes in six versions, three for identification as defined below, and three analogous ones (probable-em-, approx-em-, and size-em-) for employment.

- the probable-id-optimizer takes as input q, k and finds S that is $(q, *, k)$-PAC-id optimal
- the size-id-optimizer takes as input q, p and finds S that is $(q, p, *)$-PAC-id optimal
- the approx-id-optimizer takes as input p, k and finds S that is $(*, p, k)$-PAC-id optimal

5 Computing optimal and heuristic PAC Teaching sets

In this section we consider various algorithms computing teaching sets, of both the prudent-optimal teacher and of the heuristic teacher.

5.1 Algorithms for computing optimal teaching sets

We start by investigating the computational complexity of the task faced by the prudent-optimal teacher T_P. We achieve tight runtime bounds for all six versions of this task: teaching for identification or employment while optimizing for size or approximation or probability. We give algorithms whose runtimes provide upper bounds matching the lower bounds given by standard complexity assumptions.

The algorithms can be adapted for identification teaching or employment teaching simply by switching between similarity functions sim and sim_L. Firstly, note that in time $O(|C||X|)$, for a fixed approx-factor q and target c^*, we compute the set of Good concepts q-similar (or q_L-similar) to c^*. The following Lemma for a given teaching set S is very useful. Its proof is based on a dynamic programming algorithm that fills a size $|S|(|S|+1)$ DP table where $DP[i][j]$ holds the probability that a fixed concept is L-consistent j times on the $i+1$ first examples[7].

Lemma 1. *Given consistency matrix $C \times X$ and γ_L, teaching set S, target concept c^* and approx factor q, and Good concepts q-similar (or q_L-similar) to c^*, compute the probability that $L_P(S)$ is Good in time $O(|C| \cdot |S|^2)$.*

This lemma is used as a subroutine to give XP algorithms, parametrized by max-size k, for the probable-id-optimizer and the probable-em-optimizer problems. We also show that the runtime of these algorithms cannot be improved to the point of removing k from the exponent, unless the W-hierarchy collapses.

Theorem 1. *The problem facing the Prudent-optimal PAC Teacher T_P for probable-optimizing (id or em), given consistency matrix $C \times X$ and γ_L, approx factor q and max-size k, for a given target, can be solved in time $O(k^3|C||X|^k)$. Moreover, when parametrized by k this problem is $W[2]$-hard.*

By similar techniques, we are also able to give XP algorithms for the approx-id-optimizer and the approx-em-optimizer problems, parametrized by max-size k. Again, we show that the runtime cannot be improved to the point of removing k from the exponent, unless the W-hierarchy collapses.

Theorem 2. *The problem facing the Prudent-optimal PAC Teacher T_P for approx-optimizing (id or em) within a fixed precision (say d decimal digits), given consistency matrix $C \times X$ and γ_L, prob factor p and max-size k, for a given target, can be solved in time $O(k^3|C||X|^k)$. Moreover, when parametrized by k this problem is $W[2]$-hard.*

We also give algorithms for the size-id-optimizer and size-em-optimizer problems. These are again shown to have runtimes that are asymptotically optimal under standard complexity assumptions, in this case the Exponential Time Hypothesis and the Strong Exponential Time Hypothesis.

[7] Proofs of the lemma and all theorems can be found in the Appendix.

Theorem 3. *The problem facing the Prudent-optimal PAC Teacher T_P for size-optimizing (id or em), given consistency matrix $C \times X$ with $|X| = m$ and γ_L, approx factor q and prob factor p, for a given target, can be solved in time $O(m^2|C|2^m)$. Moreover, under ETH no algorithm with runtime $2^{o(m)}$ exists, and under Strong ETH no algorithm with runtime $(2-\epsilon)^m$ exists for any $\epsilon > 0$.*

5.2 Heuristic algorithms computing teaching sets

We discuss algorithms for T_H, i.e. heuristics computing good teaching sets for an error-aware learner like L_P. One option is to focus on the **Probably** Correct aspect of PAC Teaching. Intuitively, we want to teach with examples that *uniquely identify* the target, at least with few concepts sharing the label of the target on this example. For a target concept c^* first go over each example $x \in X$ and calculate, for some distribution over concepts, the expected value of the event that c is L-consistent with $(x, c^*(x))$. We call this the Uniqueness of x.

$$Uniqueness(x) = \mathbb{E}_{c \sim \mathcal{D}(C)}[P(\hat{c}(x) = c^*(x))]$$

Then sort the examples by the Uniqueness value and construct a teaching set greedily adding examples of lowest value first.

Another option is to focus on the **Approximately** Correct aspect of PAC Teaching. Intuitively, we want to teach with an example x' such that for the set of concepts $C_{x'}$ that share the label of the target on this example, it holds that these concepts are similar to the target. For a target concept c^* first go over each example $x' \in X$ and calculate the set of concepts $C_{x'} = \{c \in C : c(x') = c^*(x')\}$, and then for some distribution over examples, calculate the expected value of the event that c drawn from $C_{x'}$, by some distribution, is L-consistent with $(x, c^*(x))$. We call this the Homogeneity-value of x'. Then sort the examples according to Homogeneity-value and construct a teaching set greedily adding examples of highest value first.

We implemented and ran these two heuristics on the concept class over multiples-of-k given in the plots of Figure 2. In the below table we see the results for the case of a teaching set of size 1. Both heuristics select a teaching set x of very low total teaching error of at most 0.02, for all 5 values of k, showing their usefulness in this case. When deductive error is less influential, see Table 2 for teaching sets of size two in the Appendix, these heuristics should be less useful.

Table 1: Selected teaching example 'x' (teaching size 1) by heuristic algorithm. The selected teaching examples have consistently low teaching error across all target concepts.

c_{target}	Criterion	x	Heuristic Score	Mean Deductive Error	Teaching Error
5	Uniqueness	10	0.0	0.00	0.02
	Homogeneity	10	0.124	0.00	0.02
7	Uniqueness	14	0.0	0.00	0.00
	Homogeneity	14	0.089	0.00	0.00
11	Uniqueness	11	0.0	0.00	0.02
	Homogeneity	305	0.086	0.79	0.00
13	Uniqueness	13	0.0	0.00	0.00
	Homogeneity	132	0.073	0.76	0.02
17	Uniqueness	34	0.0	0.00	0.00
	Homogeneity	34	0.077	0.00	0.00

We can also combine to give a heuristic for the **Probably and Approximately** Correct aspect of PAC Teaching. Calculate a combined value for each example, based on some weighting between having low Uniqueness and simultaneously having high Homogeneity, say $Combined(x) = (1 - Uniqueness(x)) + \alpha \times Homogeneity(x)$ for a chosen factor α. Then sort by these values and construct a teaching set greedily adding examples of highest $Combined$ value first.

6 Conclusion

We have introduced the PAC teaching framework, that is useful whenever we have a learner who does not perform perfect deductive inference. We have defined different behaviors of both the teacher and the learner in situations where these errors are stochastic and measurable. We have

given algorithms finding the optimal teaching sets in these situations, and shown that the runtime of these algorithms cannot be improved unless standard complexity assumptions fail.

The PAC teaching framework is useful even if we do not know the algorithms used by the learner. This is clear from Table 1, where the LLM is not applying a learning algorithm known to us, but after deriving its deductive errors we are able to use a heuristic algorithm to compute teaching sets that increase the chance of successful PAC teaching.

Limitations. This work is primarily theoretical, establishing a basic framework having many applications, with experiments on a single LLM; more extensive empirical evaluation will surely follow.

References

F. J. Balbach. Measuring teachability using variants of the teaching dimension. *Theoretical Computer Science*, 397(1-3):94–113, 2008.

C. Calabro, R. Impagliazzo, and R. Paturi. The complexity of satisfiability of small depth circuits. In *International Workshop on Parameterized and Exact Computation*, pages 75–85. Springer, 2009.

M. Cygan, H. Dell, D. Lokshtanov, D. Marx, J. Nederlof, Y. Okamoto, R. Paturi, S. Saurabh, and M. Wahlström. On problems as hard as CNF-SAT. *ACM Trans. Algorithms*, 12(3):41:1–41:24, 2016. doi: 10.1145/2925416. URL `https://doi.org/10.1145/2925416`.

R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Texts in Computer Science. Springer, 2013.

S. Fallat, D. Kirkpatrick, H. U. Simon, A. Soltani, and S. Zilles. On batch teaching without collusion. *Journal of Machine Learning Research*, 24:1–33, 2023.

C. Ferri, D. Garigliotti, J. Hernandez-Orallo, B. Håvardstun, and J. A. Telle. When redundancy matters: Machine teaching of representations. *Machine Learning*, 115(1):20, 2026.

Z. Gao, C. Ries, H. U. Simon, and S. Zilles. Preference-based teaching. *Journal of Machine Learning Research*, 18:31:1–31:32, 2017. URL `http://jmlr.org/papers/v18/16-460.html`.

S. A. Goldman and M. J. Kearns. On the complexity of teaching. *Journal of Computer and System Sciences*, 50(1):20–31, 1995.

B. A. T. Håvardstun, C. Ferri, J. Hernández-Orallo, P. Parviainen, and J. A. Telle. XAI with machine teaching when humans are (not) informed about the irrelevant features. In *ECML/PKDD*, 2023. URL `https://api.semanticscholar.org/CorpusID:262089041`.

A. Shinohara and S. Miyano. Teachability in computational learning. *New Generation Computing*, 8:337–348, 1991.

A. Singh, A. Fry, A. Perelman, et al. Openai gpt-5 system card, 2026. URL `https://arxiv.org/abs/2601.03267`.

L. Valiant. *Probably approximately correct: nature's algorithms for learning and prospering in a complex world*. Basic Books, 2013.

L. G. Valiant. A theory of the learnable. *Comms. of the ACM*, 27(11):1134–1142, 1984.

S. C.-H. Yang, W. K. Vong, R. B. Sojitra, T. Folke, and P. Shafto. Mitigating belief projection in explainable artificial intelligence via bayesian teaching. *Scientific reports*, 11(1):1–17, 2021.

L. Zhou, W. Schellaert, F. Martínez-Plumed, et al. Larger and more instructable language models become less reliable. *Nature*, 634:61–68, 2024. doi: 10.1038/s41586-024-07930-y.

X. Zhu, A. Singla, S. Zilles, and A. N. Rafferty. An overview of machine teaching. *arXiv preprint arXiv:1801.05927*, 2018.

S. Zilles, S. Lange, R. Holte, and M. Zinkevich. Models of cooperative teaching and learning. *Journal of Machine Learning Research*, 12(Feb):349–384, 2011.

7 Appendix

7.1 LLM prompts

Here are the prompts used in our LLM experiments. See also the Github repository mentioned in the footnote.

Deductive Error - System prompt

```
Answer format:  'Answer = Yes' or 'Answer = No'
```

Deductive Error - User message

```
Is {k} a divisor of {x}?
```

Low Inductive Load - Teaching System prompt

```
Answer format:  Answer = 5 or Answer = 7 or Answer = 11 or Answer = 13 or Answer
= 17
```

Low Inductive Load - Teaching User message

```
Which of 5, 7, 11, 13 and 17 is a divisor of {x}?
```

High Inductive load - Teaching System prompt

```
Answer format:  Answer = 5 or Answer = 7 or Answer = 11 or Answer = 13 or Answer
= 17
```

High Inductive load - Teaching User message

```
Which of 5, 7, 11, 13 and 17 is not a divisor of {x1}, while it is a divisor of
{x2}?
```

7.2 Proofs of Lemma and Theorems

Here is the proof of Lemma 1.

Proof. In Algorithm 1 we give the pseudo code. Note that there are five outer for loops.

Let us first argue for correctness. We apply a top-down approach, arguing from the fifth and last outer for loop to the first one, and hence start with restating the goal. With G the set of Good concepts and $B = C \setminus G$, we want to know the probability that a concept in G is L-consistent more times than all concepts in B. We go over all the possible scenarios where at least one concept in G is L-consistent at least once more than any concept in B, computed by the following formula

$$P[max_{g\in G}X_g > max_{b\in B}X_b|S] = \\ = \sum_{i=1}^{|S|} P[max_{g\in G}X_g = i|S] \cdot P[max_{b\in B}X_b \leq i-1|S]$$

In the above X_c is simply shorthand for the count $Ct(c, S)$, and the reader can check that this formula corresponds to the fifth and last outer for loop of the pseudo-code.

For sake of simplicity we omit the repeated S from the rest of the equations. The fourth outer for loop computes the probability that the highest count over all concepts in C equals i, and the formula for this is given below. In the below equation we use a general set of concepts S, which in the pseudo-code is switched out with G or B depending on the calculation.

$$P[max_{c\in S}X_c = i] = P[max_{c\in S}X_c \leq i] - P[max_{c\in S}X_c \leq i-1]$$

Algorithm 1 Calculate probability of Approximately Correct teaching for a given teaching set. Runtime $O(|C| \cdot |S|^2)$

Input: S, C, γ_L, c^* and q, yielding a partition of concepts into G good and B bad that we also assume as input.

```
k ← |S|
for c ∈ C do                                                    ▷ O(|C| · |S|^2)
    PMF_c = DP_sub(c, S)                                        ▷ O(|S|^2)
    for i ∈ {0 ··· k} do                                        ▷ O(|S|)
        P[X_c = i] = PMF_c[i]
    end for
end for
for c ∈ C do                                                    ▷ O(|C| · |S|)
    P[X_c ≤ 0] = P[X_c = 0]
    for i ∈ {1 ··· k} do                                        ▷ O(|S|)
        P[X_c ≤ i] = P[X_c = i] + P[X_c ≤ i − 1]
    end for
end for
for i ∈ { 0 ··· k} do                                           ▷ O(|G| · |S| + |B| · |S|) = O(|C| · |S|)
    P[max_{g∈G} X_g ≤ i] = 1
    P[max_{b∈B} X_b ≤ i] = 1
    for g ∈ G do                                                ▷ O(|G|)
        P[max_{g∈G} X_g ≤ i] ∗= P[X_g ≤ i]
    end for
    for b ∈ B do                                                ▷ O(|B|)
        P[max_{b∈B} X_b ≤ i] ∗= P[X_b ≤ i]
    end for
end for
P[max_{g∈G} X_g = 0] = P[max_{g∈G} X_g ≤ 0]
for i ∈ {1 ··· k} do                                            ▷ O(|S|)
    P[max_{g∈G} X_g = i] = P[max_{g∈G} X_g ≤ i] − P[max_{g∈G} X_g ≤ i − 1]
end for
Probability = 0
for i ∈ {1 ··· k} do                                            ▷ O(|S|)
    Probability += P[max_{g∈G} X_g = i] · P[max_{b∈B} X_b ≤ i − 1]
end for
return Probability
```

with the base case

$$P[max_{c \in S} X_c = 0] = P[max_{c \in S} X_c \leq 0]$$

The third outer for loop computes the probability that the highest count is *at most* i and the formula is given below

$$P[max_{c \in S} X_c \leq i] = \prod_{c \in S} P[X_c \leq i]$$

The second outer for loop calculates the probability of a concept c having count at most i, and for this we use the Cumulative Distribution Function CDF

$$P[X_c \leq i] = P[X_c \leq i - 1] + P[X_c = i]$$

with the base case

$$P[X_c \leq 0] = P[X_c = 0]$$

Algorithm 2 $DP_{sub}(c, S)$. Find $DP[i][j]$, probability that c is L-consistent j times on the $i+1$ first examples of S, for some $0 \leq i \leq |S|-1, 0 \leq j \leq |S|$.

Input: c, S, γ_L
  $keep(x, c) = (1 - \gamma_L(c, x))$ if label of x is same as $c(x)$
  $keep(x, c) = \gamma_L(c, x)$ if label of x is different from $c(x)$
  $k \leftarrow |S|$
  $DP \leftarrow [k][k+1]$ ▷ Initialization
  $DP[0][0] \leftarrow 1 - keep(S[0], c)$
  $DP[0][1] \leftarrow keep(S[0], c)$
  for $j \in \{2 \cdots k\}$ **do**
    $DP[0][j] \leftarrow 0$
  end for
  for $i \in \{1 \cdots k-1\}$ **do**
    $DP[i][0] \leftarrow DP[i-1][0] \cdot (1 - keep(S[i], c))$
  end for
  for $i \in \{1 \cdots k-1\}$ **do**
    for $j \in \{1 \cdots k\}$ **do**
      $DP[i][j] \leftarrow DP[i-1][j-1] \cdot keep(S[i], c) + DP[i-1][j] \cdot (1 - keep(S[i], c))$
    end for
  end for
  for $j \in \{0 \cdots k\}$ **do**
    $P[X_c = j] \leftarrow DP[k-1][j]$
  end for

Finally the first outer for loop calculates the probability that a single concept c has count exactly i, namely

$$P[X_c = i] \quad \forall_{i \in \{0 \cdots |S|\}}$$

In our pseudo-code this is done by a straightforward Dynamic programming called $DP_{sub}(c, S)$ given as a separate Algorithm 2. This DP fills a $|S|(|S|+1)$ size table where $DP[i][j]$, for some $0 \leq i \leq |S|-1, 0 \leq j \leq |S|$, holds the probability that c is L-consistent j times on the $i+1$ first examples.

As we have argued along the way, these formulas compute what we need and they correspond to the pseudo-code. We are therefore done with the correctness proof.

We now argue for the runtime. In the first outer for loop, the Dynamic Programming called $DP_{sub}(c, S)$ has runtime $O(|S|^2)$, and since this is computed for each concept the first outer loop is $O(|C||S|^2)$. The second and third outer for loops are $O(|C||S|)$, while the fourth and fifth are $O(|S|)$. We conclude that the runtime is $O(|C||S|^2)$ □

Here is the proof of Theorem 1.

Proof. For given q, k we search for the best probable-optimizing teaching set by going over all subsets of size at most k, and for each one run the algorithm of Lemma 1 to find the largest value p such that this set is (q, p, k)-PAC-id (alt. (q, p, k)-PAC-em). The set S associated with the highest value p achieved this way is a $(q, *, k)$-PAC-id (alt. $(q, *, k)$-PAC-em) optimal set. The total runtime is $O(|C| \sum_{i=1}^{k} \binom{|X|}{i} \cdot i^2)$ which can be simplified to $O(k^3 |C| |X|^k)$.

Improving this runtime to an FPT algorithm, i.e. to remove k from the exponent, would imply a collapse of the W-hierarchy in Parameterized Complexity, as we now show. By a reduction similar to one given in Shinohara and Miyano (1991) we show that the probable-optimizing problem is W-hard. In the k-Hitting Set problem we are given an integer k and a family F of subsets of a universe U and asked if there are k elements of the universe that hit every subset in F, and this problem is W[2]-hard Downey and Fellows (2013). The parameterized reduction from Hitting Set to the probable-optimizing problem constructs an instance with example set $X = U$, and for each $A \in F$ a concept $c_A \in C$ such that for $x \in X$ we set $c_A(x) = 1$ if and only if $x \in A$. We add

a target concept c^* with $c^*(x) = 0$ for all x, set $\gamma_L(c, x) = 0$ for all x, c, set $q = 1$, and crucially (for the parameterized aspect) use the value of k as the size bound. Assuming there was an FPT algorithm for the probable-optimizer parameterized by k, then given an instance of k-Hitting Set we follow the above construction and run this algorithm on the constructed instance and check if the optimal set returned was $(q, 1, k)$-PAC-id (or $(q, 1, k)$-PAC-em) optimal, i.e. with $L(S)$ guessing a 1-similar concept with probability 1, in effect guessing the target always. If that is the case we have a yes-instance of the Hitting set problem, and otherwise we have a no-instance. This would constitute an FPT algorithm for k-Hitting Set and concludes the proof of the Theorem. □

Here is the proof of Theorem 2.

Proof. For given p, k we search for the best approx-optimizing teaching set by using an outer loop that performs a binary search for the best value q within d decimal digits. This amounts to at most $log_2(10^d)$ iterations of the outer loop, which is $O(1)$ for constant d. For each such value of q we go over subsets of size at most k, and run the algorithm of Lemma 1, stopping as soon as the output probability p' is at least p, meaning this set is (q, p, k)-PAC-id (alt. (q, p, k)-PAC-em). When stopping this way we continue the binary search in the larger range, whereas if no set of size at most k gives the successful stop then we continue in the lower range. For the highest successful value of q the associated set S that forced the stop is a $(*, p, k)$-PAC-id (alt. $(*, p, k)$-PAC-em) optimal set. The total runtime is $O(|C| \sum_{i=1}^{k} \binom{|X|}{i} \cdot i^2)$ which can be simplified to $O(k^3|C||X|^k)$.

Improving this runtime to an FPT algorithm, i.e. to remove k from the exponent, would imply a collapse of the W-hierarchy in Parameterized Complexity, by a similar reduction as the one given above in the proof of Theorem 1. The difference to that reduction is only that now we set $p = 1$ and do the reduction to the case of fixed precision on a single decimal digit. Then, we have a yes-instance of k-Hitting Set if and only if the highest successful value returned is $q = 1.0$. Assuming there was an FPT algorithm for the approx-optimizer parameterized by k, there would also be one for k-Hitting Set. This concludes the proof of the Theorem. □

Here is the proof of Theorem 3.

Proof. For given q, p we search for the size-optimizing teaching set by going over all subsets and for each such S run the algorithm of Lemma 1 to check if S is $(q, p, |S|)$-PAC-id (alt. $(q, p, |S|)$-PAC-em). The smallest value k with a set S with $|S| = k$ giving a positive answer means a $(q, p, *)$-PAC-id (or $(q, p, *)$-PAC-em) optimal set of size k. The total runtime is $O(m^2|C|2^m)$.

It is known that the Hitting Set problem on a universe of size m cannot be solved in time $2^{o(m)}$ unless ETH fails Calabro et al. (2009), and under Strong ETH no algorithm with runtime $(2 - \epsilon)^m$ for any $\epsilon > 0$ Cygan et al. (2016). We reduce Hitting Set to size-optimising teacher by giving a similar reduction as the one given above in the proof of Theorem 1, to show the lower bound stated in the Theorem. The difference to that reduction is only that now we fix $q = p = 1$ and the size-constraint varies with the size of S. Moreover, the answer is checked against the size of the smallest S returned and the Hitting Set is a yes-instance if and only if $|S| \leq k$, for the value of k given as part of the Hitting Set instance. Assuming there was an algorithm for the size-optimizer with runtime $O(2^{o(|X|)})$ then this would translate, through this reduction, to an algorithm solving Hitting Set in time $2^{o(m)}$, refuting ETH. Likewise, an algorithm for the size-optimizer with runtime $O^*((2-\epsilon)^{|X|})$ for $\epsilon > 0$ would translate, through this reduction, to an algorithm solving Hitting Set in time $O^*((2 - \epsilon)^m)$, refuting Strong ETH. This concludes the proof of the Theorem. □

7.3 Table of results for the Heuristic algorithm

Table 2: Selected teaching set 'x_1, x_2' by heuristic algorithm in the scenario with high total teaching error. As deductive errors are less correlated with teaching error in this setting, the non-optimal teaching error resulting from these PAC teaching sets is expected.

c_{target}	Criterion	x_1, x_2	Heuristic Score	Mean Deductive Error	Teaching Error
5	Uniqueness	{35, 119}	0.13	0.23	0.77
	Homogeneity	(35, 119)	0.13	0.23	0.77
7	Uniqueness	{35, 660}	0.2	0.30	0.64
	Homogeneity	{455, 325}	0.2	0.43	0.94
11	Uniqueness	{55, 190}	0.3	0.18	0.39
	Homogeneity	{55, 190}	0.3	0.18	0.39
13	Uniqueness	{429, 374}	0.38	0.44	0.78
	Homogeneity	{845, 750}	0.38	0.38	0.65
17	Uniqueness	{357, 672}	0.28	0.50	1.00
	Homogeneity	{595, 385}	0.28	0.48	0.98