Useful interpretability for real-world machine learning

Chandan Singh

May 8, 2022

Useful interpretability for real-world machine learning

by

Chandan Singh

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

EECS

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Bin Yu, Chair
Professor Srigokul Upadhyayula
Professor Trevor Darrell

Spring 2022

Useful interpretability for real-world machine learning

Abstract

Useful interpretability for real-world machine learning

by

Chandan Singh

Doctor of Philosophy in EECS

University of California, Berkeley

Professor Bin Yu, Chair

The recent surge in highly successful, but opaque, machine-learning models has given rise to a dire need for interpretability. This work addresses the problem of interpretability with novel definitions, methodology, and scientific investigations, ensuring that interpretations are *useful* by grounding them in the context of real-world problems and audiences. We begin by defining what we mean by interpretability and some desiderata surrounding it, emphasizing the underappreciated role of context. We then dive into novel methods for interpreting/improving neural network models, focusing on how to best score, use, and distill interactions. Next, we turn from neural networks to relatively simple rule-based models, where we investigate how to improve predictive performance while maintaining an extremely concise model. Finally, we conclude with work on open-source software and data for facilitating interpretable data science. In each case, we dive into a specific context which motivates the proposed methodology, ranging from cosmology to cell biology to medicine. Code for everything is available at github.com/csinva.

Contents

Acknowledgments

During my PhD studies at Berkeley, I have been exceptionally lucky to learn and grow with the help of (an unusually long list of) numerous talented researchers and professionals. I am greatly thankful to the mentors, colleagues, and friends (not mutually exclusive!) who helped me succeed in research and become who I am today.

First and foremost, I would like to thank my advisor, Bin Yu. She has been an *incredible* mentor, teacher, and role model for me. Spending time with her has been nothing short of a master class in tenacity, exploration, and diligence (not to mention a lot of fun!) Her extensive efforts, unrelenting curiosity, and broad knowledge have granted me the rarest of PhDs, spanning studies from the single cell to the cosmos to the emergency room to the theory of deep learning. I dare say there is not a data scientist in the world who is more helpful in more situations than Bin. On a personal level, she has gone far beyond the typical requirements expected of an advisor, helping me with matters extending well beyond the realm of technical questions. She has generously spent countless hours advising me, like in the first months of the lockdown when we spent each day closely and frantically working to do what we could to help forecast the pandemic progression at the county-level. Bin has caringly dragged out of me ideas and work that I never knew possible; I owe her a great debt of gratitude for all of her support as an advisor, both in research and in life. If I had a do-over to do a hundred PhDs, I would pick Bin as an advisor every time.

Next, I am extremely thankful for the mentoring of Gokul Upadhyayula, the most thoughtful scientist one could ever meet. Gokul taught me what it means to do something carefully; if he built a boat out of *toothpicks* and told me it was safe, I would gladly sail it across the Atlantic – I'd know it was done right. I am incredibly lucky he shared with me his breathtaking insights into the life of a cell and for patiently educating me about not just scientific specifics, but the whole scientific process. I'm also grateful to the folks in Gokul's lab (particularly Xiongtao Ruan) who embodied and passed along to me many secrets of good science.

I have also been very lucky to work with Dr. Aaron Kornblith. As we traded medical expertise for machine-learning knowledge, Aaron's pointed and direct questions often teased out in me assumptions and answers I never would have seen otherwise (not to mention a a whole lot about medical decision-making). I hope to emulate Aaron's amiability, which could charm a raccoon, and that of the many wonderful people I've had the chance to work with through him, particularly Gabriel Devlin and Newton Addo.

I am grateful also to the other legendary members of my thesis and qualifying examination committee: Trevor Darrell and Jennifer Listgarten. Their helpful feedback and comments have been a valuable source of improvement to this thesis.

Among a sea of wonderful company here at Berkeley, I was fortunate to land among some wonderful PhD student in the Yu Group. I worked closely with Jamie Murdoch my first years, who introduced me to the mysteries of interpretability, and also how to be annoying in a good way (i.e. direct). I wholeheartedly appreciate his knack for finding unique problems and ideas in an already very crowded space. Raaz Dwivedi quickly became a good friend as we spent years dreaming up what neural nets can do and why. His witty ideas always came bundled with a joke and a smile, making even frustrating times fun. In my later years, Xiao Li was a paragon of dependability and could not have been a better partner in the years we spent poking at cellular data.

I am thankful also to the wonderful postdocs I have had the opportunity to work with. Wooseok Ha has been an extraordinarily smart and kind friend (not to mention an extremely hardworking collaborator) as we set on the course to do some genuinely useful interpretability. His herculean efforts pulled the AWD paper over years of hurdles, resulting in the work I'm proudest of today. More recently, Yan Shuo has been an excellent partner / cultivator of tree ideas, and I'm grateful to have had the chance to work and grow together. I'm thankful also to the cosmology crew - Francois Lanusse, and also Vanessa Boehm + Jia Liu - for inviting us to explore mysteries beyond the world of machine-learning (and the world altogether).

I'm also deeply thankful to the Yu Group members who have been an invaluable fountain of support and ideas during my stay at Berkeley. Laura Reiger, who visited us from Denmark, was a wonderfully persistent collaborator, wholeheartedly rejecting the notion that interpretations could not be made useful. I also have had a wonderful time working with James Duncan, whose rare love for code and calm coolness made for some of the nicest collaborations I've ever had, even when things were virtual. I've quickly become friends with Omer Ronen who has been a wonderful teammate in research (and basketball) + Abhineet Agarwal, a great research collaborator. I was also lucky to work with and help advise various undergrads whose skills and perseverance were far beyond any reasonable standard: Keyan Nasseri (whose super-human work on imodels helped make it truly special), Rush Kapoor, Summer Devlin, Chris Lu, Sahil Saxena, and Saarim Rahman. I am grateful to all of the former and current Yu-Group members who have made my stay wonderful, particularly Karl

Kumbier, Yuansi Chen, Reza Abbasi-Asl, Yu Wang, Tiffany Tang, Robbie Netzorg, Nikhil Ghosh, Theo Saarinen, Nicholas Altieri, Rebecca Barter, Corrine Elliot, and Simon Walter.

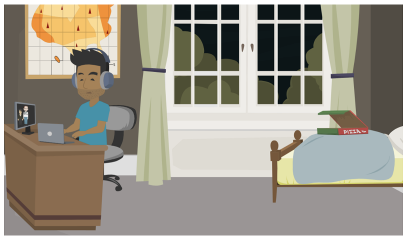
I am thankful also to many folks outside of Berkeley who have helped my research get to where it is. I was lucky to have learned from mentors during undergrad who spent countless hours with me (when I surely wasn't helpful at all) and whose advice and guidance propelled me to pursue research: Yanjun Qi (who I was lucky to reconnect with during my PhD!), William Levy, Srini Turaga, and Ben Arthur. I'm grateful also for the summer I spent under the supervision of Pietro Perona at Amazon, working with Guha Balakrishnan and Luis Goncalves. Additionally, for a wonderfully fun time at Pacmed AI in Amsterdam where I was fortunate to work with Giovanni Cina and Michele Tonutti. More recently, I've also been lucky to enjoy time with the team at Paige AI, especially Chris Kanan, Dilip Thiagarajan, Brandon Rothrock, and Patricia Raciti.

I'm also thankful to the staff members in EECS, Statistics, and BAIR who have always been more than generous with their help during the last 5 years, especially the incredible Shirley Salanio.

During the last 5 years, I have been fortunate to know, befriend, and learn from innumerous people at Berkeley, of which I'll only name a few: TMG (Alan Dong, Phong Nguyen, Gautam Gunjala, Alex Reinking, Stanley Smith, Kieran Peleaux, and Alain Anton), BAIR friends (Armin, Ashvin, Yeshwanth, Ashish, Zihao, Allan, ...), Canny lab folks (Roshan Rao, David Chan, and Forrest Huang), and many, many more.

I'd particularly like to highlight Jamie Simon, whose unrelenting curiosity is inspiring and will surely lead to great discoveries one day. Our nightly chats and weekend ventures are the parts of Berkeley that shine brightest in my memory. I'd also like to sincerely thank Saloni Singh, who has the largest heart of anyone I have ever known, and without whose support I would not have been able to do this PhD.

Finally and most importantly, I would like to thank my family: Amma, Mom, Dad, Roli, and all for their unconditional love and guidance; their (unimaginably large) support means the world to me and I am the luckiest of people to have them. I undoubtedly could not have achieved this without them, and hope I can make them proud with what I learned here.



Chapter 1

Overview

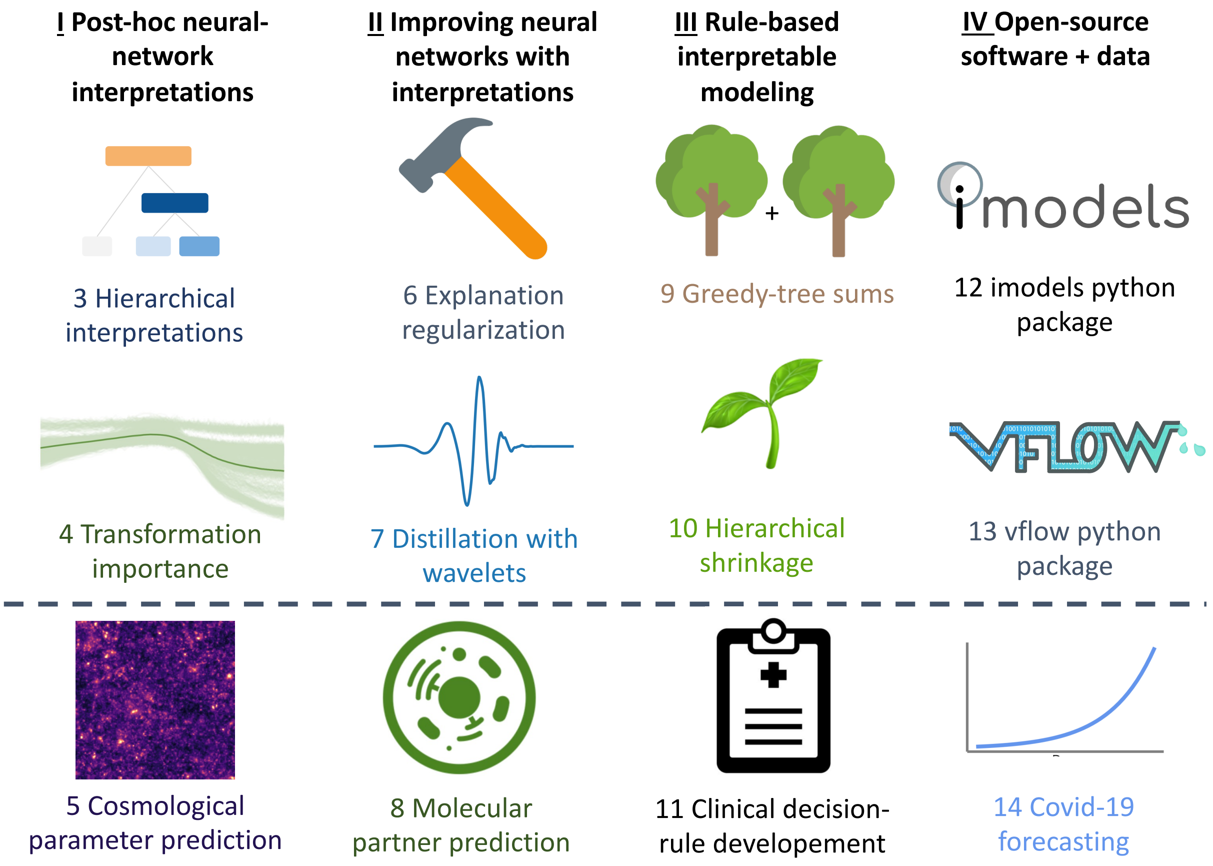

I Post-hoc neural-network interpretations
II Improving neural networks with interpretations
III Rule-based interpretable modeling
IV Open-source software + data
3 Hierarchical interpretations
6 Explanation regularization
9 Greedy-tree sums
imodels
12 imodels python package
4 Transformation importance
7 Distillation with wavelets
10 Hierarchical shrinkage
VFLOW
13 vflow python package
5 Cosmological parameter prediction
8 Molecular partner prediction
11 Clinical decision-rule developement
14 Covid-19 forecasting

Figure 1.1: Overview of this manuscript. Each column corresponds to a different part of thesis, and chapters below the dotted line correspond to real-world problems motivating the methodology in the chapters above. Code is available at github.com/csinva.

Machine-learning models have recently received considerable attention for their ability to accurately predict a wide variety of complex phenomena. However, there is a growing realization that, in addition to predictions, these models are capable of producing useful information (i.e. interpretations) about domain relationships contained in data. More precisely, interpretable machine learning can be defined as "the extraction of *relevant* knowledge from a machine-learning model concerning relationships either contained in data or learned by the model" [186] (see the next section for a much deeper discussion of this).

Interpretations have found uses both in their own right, e.g. medicine [153], science [13, 278], and policy-making [37] as well as in auditing predictions themselves in response to issues such as regulatory pressure [97] and fairness [74]. In these domains, interpretations have been shown to help with evaluating a learned model, providing information to repair a model (if needed), and building trust with domain experts [47]. However, this increasing role, along with the explosion in proposed interpretation techniques [186, 193, 291, 273, 90, 11, 300, 100] has raised considerable concerns about the use of interpretation methods in practice [4]. Furthermore, it is unclear how interpretation techniques should be evaluated in the real-world context to advance our understanding of a particular problem.

To do so, we first review some of the desiderata of interpretability, following our 2019 PNAS paper [186] (co-authored with Jamie Murdoch, Reza Abbasi-Asl, Karl Kumbier, and Bin Yu). We then discuss some methods for critically evaluating interpretations. We then expound on new methodology to address gaps in the interpretability of machine-learning models. Crucially, this methodology is developed and evaluated in the context of real-world problems in conjunction with domain experts. This work spans different levels, trying to extract insight from black-box models, as well as replace them whenever possible with simpler ones.

Fig. 1.1 shows an overview of this thesis, which aims to tackle the interpretability problem strictly grounded in real-world problems. Part I begins by explaining different methods for post-hoc interpretation of neural networks. These methods enable understanding interactions between different features in a neural network, and are grounded in the context of cosmological parameter prediction (Chapter 5). Part II then shows how these interpretation methods can be used to directly improve neural networks, either through regularization (Chapter 6) or through distillation (Chapter 7). This is showcased in the context of molecular partner prediction (Chapter 8). Next, Part III introduces improved methodology for building highly predictive rule-based models that are extremely concise, grounded in the problem of clinical decision-rule development. Finally, Part IV introduces new open-source software and data for inerpretable modeling.

1.1 Part I: Post-hoc neural-network interpretations

A vast line of prior work has focused on assigning importance to individual features, such as pixels in an image or words in a document. Several methods yield feature-level importance for different architectures. They can be categorized as gradient-based [255, 264, 240, 18], decomposition-based [185, 242, 17] and others [59, 83, 218, 303], with many similarities among the methods [10, 159]. While many methods have been developed to attribute importance to individual features of a model's input, relatively little work has been devoted to understanding interactions between key features. These interactions are a crucial part of interpreting modern deep-learning models, as they are what enable strong predictive performance on structured data.

Part I covers two recent methods developed to extract the *interactions* between features that an (already trained) DNN has learned. Chapter 3 covers agglomerative contextual decomposition (ACD), which generates hierarchical importances by greedily scoring and combining group-level importances. This enables simple and effective visualization of which features are important for an individual predicton. This work was published in ICLR 2019 [248], and was joint with Jamie Murdoch and Bin Yu.

ACD shows how to attribute importance to interactions between features. However, in many cases, raw features such as pixels in an image or words in a document may not be the most meaningful spaces to perform interpretation. When features are highly correlated or features in isolation are not semantically meaningful, the resulting attributions need to be improved. To remedy this problem, Chapter 4 introduces transformation importance (TRIM), which allows for computing scores for interactions on transformations of a model's input. This was published in a 2020 ICLR workshop [252], joint with Wooseok Ha, Francois Lanusse, Vanessa Boehm, and Bin Yu. Other methods have been recently developed for understanding model interactions with varying degrees of computational cost and faithfulness to the trained model [274, 273, 64].[1].

Both these methods allow for better understanding in the context of cosmological parameter prediction (Chapter 5), where interpretability allows one to trust a model's predictions when applied to real astronomical data.

1.2 Part II: Improving neural networks with interpretations

Given the introduced methods for interpreting interactions and transformations introduced in Part I, Part II covers two methods for using these attributions to directly improve models. This is an important and often overlooked step when introducing and evaluating interpretation methods, which helps to ground the utility of an interpretation with a direct use-case.

Chapter 6 introduces Contextual Decomposition Explanation Penalization (CDEP), which directly regularizes interaction-importance scores during training, allowing a practitioner to inject their domain knowledge into the training process. This can improve generalization performance in interesting ways, depending on the context of the problem. CDEP was published at ICML 2020 [223] and is joint with Laura Rieger, Jamie Murdoch, and Bin Yu.

[1]For a paper summarizing this part and related previous work, see [246]

Chapter 7 shows how TRIM scores can be used to distill a fully trained neural network into a simple data-driven wavelet model. Surprisingly, the distilled models actually *improve* their predictive performance despite an incredible reduction in their size (e.g. going from millions of parameters to ten). This work helps show how complex models along with domain-expert knowledge can be combined to yield a final, interpretable model. This work was published in Neurips 2021 [100] and is joint with Wooseok Ha, Francois Lanusse, Srigokul Upadhyayula, and Bin Yu. It is then shown how AWD works in the context of molecular partner prediction (Chapter 8), where it improve predictive performance in the limited-data regime and drastically decrease model size and inference time.

1.3 Part III: Rule-based interpretable modeling

This section turns away from deep learning completely and focuses instead on rule-based modeling. Whenever possible, fitting a simple model in the first place is preferable to fitting a complex model and then inspecting it with post-hoc interpretations. A sufficiently simple rule-based model can be completely understood, and is easy to simulate by hand, memorize, and reason about counterfactuals with.

Chapter 9 introduces FIGS, a new method which extends the traditional CART decision algorithm to the more flexible class of decision-tree sums. The algorithm is still greedy and fast, but can now better capture certain structures in data, such as additivity. As a result, FIGS can often achieve high predictive performance even with very few rules. This work is currently a preprint [269] joint with Yan Shuo Tan, Keyan Nasseri, Abhineet Agarwal, and Bin Yu.

Chapter 10 introduces Hierarchical Shrinkage (HS), a method for regularizing any decision-tree algorithm. HS regularizes the value at any node in the true towards its parents, often resulting in improved generalization accuracy. It can be applied posthoc and is extremely fast, making it easy to apply in many situations. This work is currently a preprint [5] joint with Abhineet Agarwal, Yan Shuo Tan, Omer Ronen, and Bin Yu.

Both FIGS and HS are then considered in the context of clinical decision-rule development (Fig. 9.3, [135]), where interpretability of rules is crucial to their effective use in the emergency room. This work is currently a preprint [135], co-led with Aaron Kornblith.

1.4 Part IV: Open-source software and data

In data-science and machine-learning, good open-source software and data repositories are as useful (if not more) than good ideas. This part covers two python packages grown out of the research above as well as a curated data repository for open-source modeling.

Chapter 12 covers `imodels`, a python package for fitting interpretable (mostly rule-based models). The package is fully scikit-learn compatible and makes it easy to use cutting-edge interpretable models such as FIGS and HS, as well as RuleFit [86] or Bayesian Rule

Lists [143]. As of April 2021, it has 715+ github stars and 34k+ downloads from pypi. `imodels` was published in JOSS [251] with Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu.

Chapter 13 covers `vflow`, an ambitious python package trying to make stability analysis simple. It provides users a simple interface for stability analysis, i.e. checking the robustness of results from a data-science pipeline to various judgement calls made during modeling. This ensures that arbitrary judgement calls made by data-practitioners (e.g. specifying a default imputation strategy) do not dramatically alter the final conclusions made in a modeling pipeline. This package was published in JOSS [71] with James Duncan, Rush Kapoor, Abhineet Agarwal, and Bin Yu.

Chapter 2

Interpretability: for what and for whom?

trust disentanglement simulatability fairness explainability

In the absence of a well-formed definition of interpretability, a broad range of methods with a correspondingly broad range of outputs (e.g. visualizations, natural language, mathematical equations) have been labeled as interpretation. This has led to considerable confusion about the notion of interpretability. In particular, it is unclear what it means to interpret something, what common threads exist among disparate methods, and how to select an interpretation method for a particular problem/audience.

In this chapter, we attempt to address these concerns. To do so, we first define interpretability in the context of machine learning and place it within a generic data science life cycle. This allows us to distinguish between two main classes of interpretation methods: model-based[1] and post hoc. We then introduce the Predictive, Descriptive, Relevant (PDR) framework, consisting of three desiderata for evaluating and constructing interpretations: predictive accuracy, descriptive accuracy, and relevancy, where relevancy is judged by a human audience. Using these terms, we categorize a broad range of existing methods, all grounded in real-world examples[2]. In doing so, we provide a common vocabulary for researchers and practitioners to use in evaluating and selecting interpretation methods. We then show how our work enables a clearer discussion of open problems for future research.

Defining interpretable machine learning

On its own, interpretability is a broad, poorly defined concept. Taken to its full generality, to interpret data means to extract information (of some form) from it. The set of methods falling under this umbrella spans everything from designing an initial experiment to visualizing final results. In this overly general form, interpretability is not substantially different from the established concepts of data science and applied statistics.

Instead of general interpretability, we focus on the use of interpretations to produce insight from ML models as part of the larger data-science life cycle. We define interpretable machine learning as the extraction of *relevant* knowledge from a machine-learning model concerning relationships either contained in data or learned by the model. Here, we view knowledge as being *relevant* if it provides insight for a particular audience into a chosen problem. These insights are often used to guide communication, actions, and discovery. They can be produced in formats such as visualizations, natural language or mathematical equations, depending on the context and audience. For instance, a doctor who must diagnose a single patient will want qualitatively different information than an engineer determining if an image classifier is discriminating by race. What we define as interpretable ML is sometimes referred to as explainable ML, intelligible ML or transparent ML. We include these headings under our definition.

Background on interpretable machine learning

Interpretability is a quickly growing field in machine learning, and there have been multiple works examining various aspects of interpretations (sometimes under the heading *explainable AI*). One line of work focuses on providing an overview of different interpretation methods with a strong emphasis on post hoc interpretations of deep learning models [48, 99], sometimes pointing out similarities between various methods [159, 10]. Other work has focused on the narrower problem of evaluating interpretations [68, 93] and what properties they should satisfy [152]. These previous works touch on different subsets of interpretability, but do not address interpretable machine learning as a whole, and give limited guidance on how interpretability can actually be used in data-science life cycles. We aim to do so by providing a

[1]For clarity, throughout the thesis we use the term *model* to refer to both machine-learning models and algorithms.

[2]Examples were selected through a non-exhaustive search of related work.

framework and vocabulary to fully capture interpretable machine learning, its benefits, and its applications to concrete data problems.

Interpretability also plays a role in other research areas. For example, interpretability is a major topic when considering bias and fairness in ML models [102, 30, 60]. In psychology, the general notions of interpretability and explanations have been studied at a more abstract level [126, 156], providing relevant conceptual perspectives. Additionally, we comment on two related areas that are distinct but closely related to interpretability: causal inference and stability.

Causal inference Causal inference [117] is a subject from statistics which is related, but distinct, from interpretable machine learning. According to a prevalent view, causal inference methods focus solely on extracting causal relationships from data, i.e. statements that altering one variable will cause a change in another. In contrast, interpretable ML, and most other statistical techniques, are used to describe general relationships. Whether or not these relationships are causal cannot be verified through interpretable ML techniques, as they are not designed to distinguish between causal and non-causal effects.

In some instances, researchers use both interpretable machine learning and causal inference in a single analysis [22]. One form of this is where the non-causal relationships extracted by interpretable ML are used to suggest potential causal relationships. These relationships can then be further analyzed using causal inference methods, and fully validated through experimental studies.

Stability Stability, as a generalization of robustness in statistics, is a concept that applies throughout the entire data-science life cycle, including interpretable ML. The stability principle requires that each step in the life cycle is stable with respect to appropriate perturbations, such as small changes in the model or data. Recently, stability has been shown to be important in applied statistical problems, for example when trying to make conclusions about a scientific problem [292] and in more general settings [101]. Stability can be helpful in evaluating interpretation methods and is a prerequisite for trustworthy interpretations. That is, one should not interpret parts of a model which are not stable to appropriate perturbations to the model and data. This is demonstrated through examples in the text [208, 2, 22].

2.1 Interpretation in the data science life cycle

Before discussing interpretation methods, we first place the process of interpretable ML within the broader data-science life cycle. Figure 2.1 presents a deliberately general description of this process, intended to capture most data-science problems. What is generally referred to as interpretation largely occurs in the modeling and post hoc analysis stages, with the problem, data and audience providing the context required to choose appropriate methods.

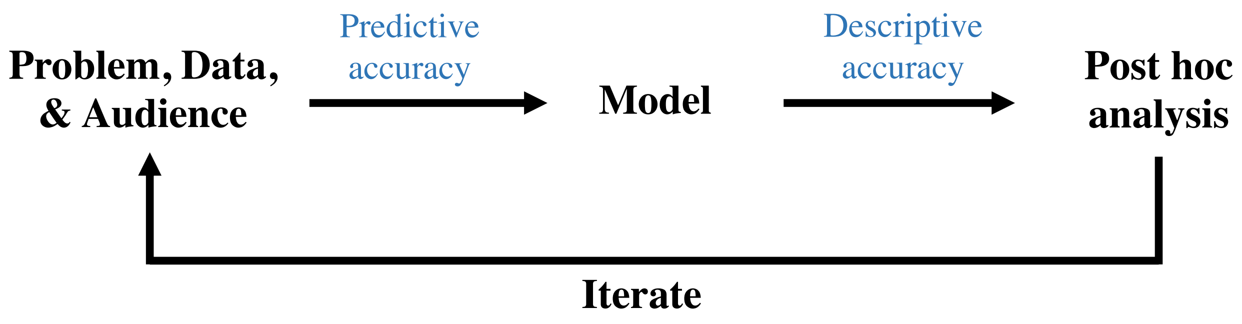


Figure 2.1: Overview of different stages (black text) in a data-science life cycle where interpretability is important. Main stages are discussed in Sec. 2.1 and accuracy (blue text) is described in Sec. 2.2.

Problem, data, and audience At the beginning of the cycle, a data-science practitioner defines a domain problem that they would like to understand using data. This problem can take many forms. In a scientific setting, the practitioner may be interested in relationships contained in the data, such as how brain cells in a particular area of the visual system relate to visual stimuli [225]. In industrial settings, the problem often concerns the predictive performance or other qualities of a model, such as how to assign credit scores with high accuracy [116], or do so fairly with respect to gender and race [60]. The nature of the problem plays a role in interpretability, as the relevant context and audience are essential in determining what methods to use.

After choosing a domain problem, the practitioner collects data to study it. Aspects of the data-collection process can affect the interpretation pipeline. Notably, biases in the data (i.e. mismatches between the collected data and the population of interest) will manifest themselves in the model, restricting one's ability to generalize interpretations generated from the data to the population of interest.

Model Based on the chosen problem and collected data, the practitioner then constructs a predictive model. At this stage, the practitioner processes, cleans, and visualizes data, extracts features, selects a model (or several models) and fits it. Interpretability considerations often come into play in this step related to the choice between simpler, easier to interpret models and more complex, black-box models, which may fit the data better. The model's ability to fit the data is measured through predictive accuracy.

Post hoc analysis Having fit a model (or models), the practitioner then analyzes it for answers to the original question. The process of analyzing the model often involves using interpretability methods to extract various (stable) forms of information from the model. The extracted information can then be analyzed and displayed using standard data analysis methods, such as scatter plots and histograms. The ability of the interpretations to properly describe what the model has learned is denoted by descriptive accuracy.

Iterate If sufficient answers are uncovered after the post hoc analysis stage, the practitioner finishes. Otherwise, they update something in the chain (problem, data, and/or model) and iterate [29]. Note that they can terminate the loop at any stage, depending on the context of the problem.

Interpretation methods within the PDR framework

In the framework described above, our definition of interpretable ML focuses on methods in either the modeling or post hoc analysis stages. We call interpretability in the modeling stage *model-based interpretability* (Sec. 2.3). This part of interpretability is focused upon the construction of models that readily provide insight into the relationships they have learned. In order to provide this insight, model-based interpretability techniques must generally use simpler models, which can result in lower predictive accuracy. Consequently, model-based interpretability is best used when the underlying relationship is sufficiently simple that model-based techniques can achieve reasonable predictive accuracy, or when predictive accuracy is not a concern.

We call interpretability in the post hoc analysis stage *post hoc interpretability* (Sec. 2.4). In contrast to model-based interpretability, which alters the model to allow for interpretation, post-hoc interpretation methods take a trained model as input, and extract information about what relationships the model has learned. They are most helpful when the data is especially complex, and practitioners need to train a black-box model in order to achieve reasonable predictive accuracy.

After discussing desiderata for interpretation methods, we investigate these two forms of interpretations in detail and discuss associated methods.

2.2 The PDR desiderata for interpretations

In general, it is unclear how to select and evaluate interpretation methods for a particular problem and audience. To help guide this process, we introduce the PDR framework, consisting of three desiderata that should be used to select interpretation methods for a particular problem: predictive accuracy, descriptive accuracy, and relevancy.

Accuracy

The information produced by an interpretation method should be faithful to the underlying process the practitioner is trying to understand. In the context of ML, there are two areas where errors can arise: when approximating the underlying data relationships with a model (predictive accuracy) and when approximating what the model has learned using an interpretation method (descriptive accuracy). For an interpretation to be trustworthy, one should try to maximize both of the accuracies. In cases where either accuracy is not very high, the resulting interpretations may still be useful. However, it is especially important to check their trustworthiness through external validation, such as running an additional experiment.

Predictive accuracy

The first source of error occurs during the model stage, when an ML model is constructed. If the model learns a poor approximation of the underlying relationships in the data, any information extracted from the model is unlikely to be accurate. Evaluating the quality of a model's fit has been well-studied in standard supervised ML frameworks, through measures such as test-set accuracy. In the context of interpretation, we describe this error as predictive accuracy.

Note that in problems involving interpretability, one must appropriately measure predictive accuracy. In particular, the data used to check for predictive accuracy must resemble the population of interest. For instance, evaluating on patients from one hospital may not generalize to others. Moreover, problems often require a notion of predictive accuracy that goes beyond just average accuracy. The distribution of predictions matters. For instance, it could be problematic if the prediction error is much higher for a particular class. Finally, the predictive accuracy should be stable with respect to reasonable data and model perturbations. One should not trust interpretations from a model which changes dramatically when trained on a slightly smaller subset of the data.

Descriptive accuracy

The second source of error occurs during the post hoc analysis stage, when interpretation methods are used to analyze a fitted model. Oftentimes, interpretations provide an imperfect representation of the relationships learned by a model. This is especially challenging for complex black-box models such as neural networks, which store nonlinear relationships between variables in non-obvious forms.

▷ **Definition** We define *descriptive accuracy*, in the context of interpretation, as the degree to which an interpretation method objectively captures the relationships learned by machine learning models.

A common conflict: predictive vs descriptive accuracy

In selecting what model to use, practitioners are sometimes faced with a trade-off between predictive and descriptive accuracy. On the one hand, the simplicity of model-based interpretation methods yields consistently high descriptive accuracy, but can sometimes result in lower predictive accuracy on complex datasets. On the other hand, in complex settings such as image analysis, complicated models can provide high predictive accuracy, but are harder to analyze, resulting in a lower descriptive accuracy.

Relevancy

When selecting an interpretation method, it is not enough for the method to have high accuracy - the extracted information must also be relevant. For example, in the context of genomics, a patient, doctor, biologist, and statistician may each want different (yet consistent) interpretations from the same model. The context provided by the problem and data stages in Figure 2.1 guides what kinds of relationships a practitioner is interested in learning about, and by extension the methods that should be used.

▷ **Definition** We define an interpretation to be *relevant* if it provides insight for a particular audience into a chosen domain problem.

Relevancy often plays a key role in determining the trade-off between predictive and descriptive accuracy. Depending on the context of the problem at hand, a practitioner may choose to focus on one over the other. For instance, when interpretability is used to audit a model's predictions, such as to enforce fairness, descriptive accuracy can be more important. In contrast, interpretability can also be used solely as a tool to increase the predictive accuracy of a model, for instance, through improved feature engineering.

Having outlined the main desiderata for interpretation methods, we now discuss how they link to interpretation in the modeling and post hoc analysis stages in the data-science life cycle. Figure 2.2 draws parallels between our desiderata for interpretation techniques introduced in Sec. 2.2 and our categorization of methods in Sec. 2.3 and Sec. 2.4. In particular, both post hoc and model-based methods aim to increase descriptive accuracy, but only model-based affects the predictive accuracy. Not shown is relevancy, which determines what type of output is helpful for a particular problem and audience.

2.3 Model-based interpretability

We now discuss how interpretability considerations come into play in the modeling stage of the data science life cycle (see Figure 2.1). At this stage, the practitioner constructs an ML model from the collected data. We define model-based interpretability as the construction of models that readily provide insight into the relationships they have learned. Different model-based interpretability methods provide different ways of increasing descriptive accuracy by constructing models which are easier to understand, sometimes resulting in lower predictive accuracy. The main challenge of model-based interpretability is to come up with models that are simple enough to be easily understood by the audience, while maintaining high predictive accuracy.

In selecting a model to solve a domain problem, the practitioner must consider the entirety of the PDR framework. The first desideratum to consider is predictive accuracy. If the constructed model does not accurately represent the underlying problem, any subsequent analysis will be suspect [34, 84]. Second, the main purpose of model-based interpretation methods is to increase descriptive accuracy. Finally, the relevancy of a model's output must

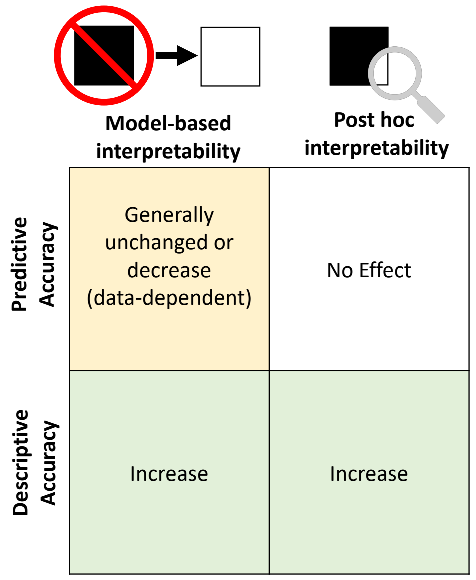

Model-based interpretability
Post hoc interpretability
Predictive Accuracy
Descriptive Accuracy
Generally unchanged or decrease (data-dependent)
No Effect
Increase
Increase

Figure 2.2: Impact of interpretability methods on descriptive and predictive accuracies. Model-based interpretability (Sec. 2.3) involves using a simpler model to fit the data which can negatively affect predictive accuracy, but yields higher descriptive accuracy. Post hoc interpretability (Sec. 2.4) involves using methods to extract information from a trained model (with no effect on predictive accuracy). These correspond to the model and post hoc stages in Figure 2.1.

be considered, and is determined by the context of the problem, data, and audience. We now discuss some common types of model-based interpretability methods.

Sparsity

When the practitioner believes that the underlying relationship in question is based upon a sparse set of signals, they can impose sparsity on their model by limiting the number of non-zero parameters. In this section, we focus on linear models, but sparsity can be helpful more generally. When the number of non-zero parameters is sufficiently small, a practitioner can interpret the variables corresponding to those parameters as being meaningfully related to the outcome in question, and can also interpret the magnitude and direction of the parameters. However, before one can interpret a sparse parameter set, one should check for stability of the parameters. For example, if the signs/magnitudes of parameters, or the predictions change due to small perturbations in the data set, the coefficients should not be interpreted [149] .

When the practitioner is able to correctly incorporate sparsity into their model, it can improve all three interpretation desiderata. By reducing the number of parameters to analyze, sparse models can be easier to understand, yielding higher descriptive accuracy. Moreover, incorporating prior information in the form sparsity into a sparse problem can help a model achieve higher predictive accuracy and yield more relevant insights. Note that incorporating sparsity can often be quite difficult, as it requires understanding the data-specific structure of the sparsity and how it can be modelled.

Methods for obtaining sparsity often utilize a penalty on a loss function, such as LASSO [271] and sparse coding [195], or on a model selection criteria such as AIC or BIC [7, 39]. Many search-based methods have been developed to find sparse solutions. These methods search through the space of non-zero coefficients using classical subset-selection methods (e.g. orthogonal matching pursuit [204]). Model sparsity is often useful for high-dimensional problems, where the goal is to identify key features for further analysis. For instance, sparsity penalties have been incorporated into random forests to identify a sparse subset of important features [9].

In the following example from genomics, sparsity is used to increase the relevancy of an interpretation by reducing the number of potential interactions to a manageable level.

▷ **Ex.** Identifying interactions among regulatory factors or biomolecules is an important question in genomics. Typical genomic datasets include thousands or even millions of features, many of which are active in specific cellular or developmental contexts. The massive scale of such datasets make interpretation a considerable challenge. Sparsity penalties are frequently used to make the data manageable for statisticians and their collaborating biologists to discuss and identify promising candidates for further experiments.

For instance, one recent study [208] uses a biclustering approach based on sparse canonical correlation analysis (SCCA) to identify interactions among genomic expression features in *Drosophila melanogaster* (fruit flies) and *Caenorhabditis elegans* (roundworms). Sparsity penalties enable key interactions among features to be summarized in heatmaps which contain few enough variables for a human to analyze. This study also performs stability analysis, finding their model to be robust to different initializations and perturbations to hyperparameters.

Simulatability

A model is said to be simulatable if a human (for whom the interpretation is intended) is able to internally simulate and reason about its entire decision-making process (i.e. how a trained model produces an output for an arbitrary input). This is a very strong constraint to place on a model, and can generally only be done when the number of features is low, and the underlying relationship is simple. Decision trees [32] are often cited as a simulatable model, due to their hierarchical decision-making process. Another example is lists of rules [86, 143], which can easily be simulated. However, it is important to note that these models cease to be simulatable when they become large. In particular, as the complexity of the model increases (number of nodes in a decision tree, or the number of rules in a list), it becomes increasingly difficult for a human to internally simulate.

Due to their simplicity, simulatable models have very high descriptive accuracy. When they can also provide reasonable predictive accuracy, they can be very effective. In the following example, a novel simulatable model is able to produce high predictive accuracy, while maintaining the high levels of descriptive accuracy and relevancy normally attained by small-scale rules-based models.

▷ **Ex.** In medical practice, when a patient has been diagnosed with atrial fibrillation, caregivers often want to predict the risk that the particular patient will have a stroke in the next year. Given the potential ramifications of medical decisions, it is important that these predictions are not only accurate, but interpretable to both the caregivers and patients.

To make the prediction, [143] uses data from 12,586 patients detailing their age, gender, history of drugs and conditions, and whether they had a stroke within a year of diagnosis. In order to construct a model that has high predictive and descriptive accuracy, [143] introduce a method for learning lists of if-then rules that are predictive of one year stroke risk. The resulting classifier, displayed in Fig S1, requires only seven if-then statements to achieve competitive accuracy, and is easy for even non-technical practitioners to quickly understand.

Although this model is able to achieve high predictive and descriptive accuracy, it is important to note that the lack of stability in these types of models can limit their uses. If the practitioner's intent is to simply understand a model that is ultimately used for predictions, these types of models can be very effective. However, if they want to produce knowledge about the underlying dataset, the fact that the learned rules can change significantly when the model is re-trained limits their generalizability.

Modularity

We define an ML model to be modular if a meaningful portion(s) of its prediction-making process can be interpreted independently. A wide array of models satisfy modularity to different degrees. Generalized additive models [104] force the relationship between variables in the model to be additive. In deep learning, specific methods such as attention [129] and modular network architectures [11] provide limited insight into a network's inner workings. Probabilistic models can enforce modularity by specifying a conditional independence structure which makes it easier to reason about different parts of a model independently [134].

The following example uses modularity to produce relevant interpretations for use in diagnosing biases in training data.

▷ **Ex.** When prioritizing patient care for pneumonia patients in a hospital, one possible method is to predict the likelihood of death within 60 days, and focus on the patients with a higher mortality risk. Given the potential life and death consequences, being able to explain the reasons for hospitalizing a patient or not is very important.

A recent study [47] uses a dataset of 14,199 pneumonia patients, with 46 features including from demographics (e.g. age and gender), simple physical measurements (e.g. heart rate, blood pressure) and lab tests (e.g. white blood cell count, blood urea nitrogen). To predict mortality risk, they use a generalized additive model with pairwise interactions, displayed below. The univariate and pairwise terms ($f_j(x_j)$ and $f_{ij}(x_i, x_j)$) can be individually interpreted in the form of curves and heatmaps respectively.

$$g(\mathbb{E}[y]) = \beta_0 + \sum_j f_j(x_j) + \sum_{i \neq j} f_{ij}(x_i, x_j) \tag{2.1}$$

By inspecting the individual modules, the researchers found a number of counterintuitive properties of their model. For instance, the fitted model learned that having asthma is associated with a lower risk of dying from pneumonia. In reality, the opposite is true - patients with asthma are known to have a higher risk of death from pneumonia. Because of this, in the collected data all patients with asthma received aggressive care, which was fortunately effective at reducing their risk of mortality relative to the general population.

In this instance, if the model were used without having been interpreted, pneumonia patients with asthma would have have been de-prioritized for hospitalization. Consequently, the use of ML would increase their likelihood of dying. Fortunately, the use of an interpretable model enabled the researchers to identify and correct errors like this one, better ensuring that the model could be trusted in the real world.

Domain-based feature engineering

While the type of model is important in producing a useful interpretation, so are the features that are used as inputs to the model. Having more informative features makes the relationship that needs to be learned by the model simpler, allowing one to use other model-based interpretability methods. Moreover, when the features have more meaning to a particular audience, they become easier to interpret.

In many individual domains, expert knowledge can be useful in constructing feature sets that are useful for building predictive models. The particular algorithms used to extract features are generally domain-specific, relying both on the practitioner's existing domain expertise and insights drawn from the data through exploratory data analysis. For example, in natural language processing, documents are embedded into vectors using tf-idf [211]. Moreover, using ratios, such as the Body Mass Index (BMI), instead of raw features can greatly simplify the relationship a model learns, resulting in improved interpretations. In the example below, domain knowledge about cloud coverage is exploited to design three simple features that increase predictive accuracy while maintaining the high descriptive accuracy of a simple predictive model.

▷ **Ex.** When modelling global climate patterns, an important quantity is the amount and location of arctic cloud coverage. Due to the complex, layered nature of climate models, it is beneficial to have simple, easily auditable, cloud coverage models for use by down-stream climate scientists.

In [241], the authors use an unlabeled dataset of arctic satellite imagery to build a model predicting whether each pixel in an image contains clouds or not. Given the qualitative similarity between ice and clouds, this is a challenging prediction problem. By conducting exploratory data analysis and utilizing domain knowledge through interactions with climate scientists, the authors identify three simple features that are sufficient to cluster whether or not images contain clouds. Using these three features as input to quadratic discriminant analysis, they achieve both high predictive accuracy and transparency when compared with expert labels (which were not used in developing the model).

Model-based feature engineering

There are a variety of automatic approaches for constructing interpretable features. Two examples are unsupervised learning and dimensionality reduction. Unsupervised methods, such as clustering, matrix factorization, and dictionary learning, aim to process unlabelled data and output a description of their structure. These structures often shed insight into relationships contained within the data and can be useful in building predictive models. Dimensionality reduction focuses on finding a representation of the data which is lower-dimensional than the original data. Methods such as principal components analysis [121], independent components analysis [24], and canonical correlation analysis [114] can often identify a few interpretable dimensions, which can then be used as input to a model or to provide insights in their own right. Using fewer inputs can not only improve descriptive accuracy, but can increase predictive accuracy by reducing the number of parameters to fit. In the following example, unsupervised learning is used to represent images in a low-dimensional, genetically meaningful, space.

▷ **Ex.** Heterogeneity is an important consideration in genomic problems and associated data. In many cases, regulatory factors or biomolecules can play a specific role in one context, such as a particular cell type or developmental stage, and have a very different role in other contexts. Thus, it is important to understand the "local" behavior of regulatory factors or biomolecules. A recent study [287], uses unsupervised learning to learn spatial patterns of gene expression in *Drosophila* (fruit fly) embryos. In particular, they use stability driven nonnegative matrix factorization to decompose images of complex spatial gene expression patterns into a library of 21 "principal patterns", which can be viewed as pre-organ regions. This decomposition, which is interpretable to biologists, allows the study of gene-gene interactions in pre-organ regions of the developing embryo.

2.4 Post hoc interpretability

We now discuss how interpretability considerations come into play in the post hoc analysis stage of the data-science life cycle. At this stage, the practitioner analyzes a trained model in order to provide insights into the learned relationships. This is particularly challenging when the model's parameters do not clearly show what relationships the model has learned. To aid in this process, a variety of post hoc interpretability methods have been developed to provide insight into what a trained model has learned, without changing the underlying model. These methods are particularly important for settings where the collected data is high-dimensional and complex, such as with image data. In these settings, interpretation methods must deal with the challenge that individual features are not semantically meaningful, making the problem more challenging than on datasets with more meaningful features. Once the information has been extracted from the fitted model, it can be analyzed using standard, exploratory data analysis techniques, such as scatter plots and histograms.

When conducting post hoc analysis, the model has already been trained, so its predictive accuracy is fixed. Thus, under the PDR framework, a researcher must only consider descriptive accuracy and relevancy (relative to a particular audience). Improving on each of these criteria are areas of active research.

Most widely useful post hoc interpretation methods fall into two main categories: prediction-level and dataset-level interpretations, which are sometimes referred to as local and global interpretations, respectively. Prediction-level interpretation methods focus on explaining individual predictions made by models, such as what features and/or interactions led to the particular prediction. Dataset-level approaches focus on the global relationships the model has learned, such as what visual patterns are associated with a predicted response. These two categories have much in common (in fact, dataset-level approaches often yield information at the prediction-level), but we discuss them separately, as methods at different levels are meaningfully different. Prediction-level insights can provide fine-grained information about individual predictions, but often fail to yield dataset-level insights when it is not feasible to examine a sufficient amount of prediction-level interpretations.

Dataset-level interpretation

When a practitioner is interested in more general relationships learned by a model, e.g. relationships that are relevant for a particular class of responses, they use dataset-level interpretations. For instance, this form of interpretation can be useful when it is not feasible for a practitioner to look at a large number of local predictions. In addition to the areas below, we note that there are other emerging techniques, such as model distillation [58, 90].

Interaction and feature importances

Feature importance scores, at the dataset-level, try to capture how much individual features contribute, across a dataset, to a prediction. These scores can provide insights into what features the model has identified as important for which outcomes, and their relative importance. Methods have been developed to score individual features in many models including neural networks [194], random forests, [33, 261], and generic classifiers [8].

In addition to feature importances, methods exist to extract important interactions between features. Interactions are important as ML models are often highly nonlinear and learn complex interactions between features. Methods exist to extract interactions from many ML models, including random forests [22, 136, 64] and neural networks [273, 1]. In the below example, the descriptive accuracy of random forests is increased by extracting Boolean interactions (a problem-relevant form of interpretation) from a trained model.

▷ **Ex.** High-order interactions among regulatory factors or genes play an important role in defining cell-type specific behavior in biological systems. Thus, extracting such interactions from genomic data is an important problem in biology.

A previous line of work considers the problem of searching for biological interactions associated with important biological processes [22, 136]. To identify candidate biological interactions, the authors train a series of iteratively re-weighted RFs and search for stable combinations of features that frequently co-occur along the predictive RF decision paths. This approach takes a step beyond evaluating the importance of individual features in an RF, providing a more complete description of how features influence predicted responses. By interpreting the interactions used in RFs, the researchers identified gene-gene interactions with 80% accuracy in the *Drosophila* embryo and identify candidate targets for higher-order interactions.

Statistical feature importances

In some instances, in addition to the raw value, we can compute statistical measures of confidence as feature importance scores, a standard technique taught in introductory statistics classes. By making assumptions about the underlying data generating process, models like linear and logistic regression can compute confidence intervals and hypothesis tests for the values, and linear combinations, of their coefficients. These statistics can be helpful in determining the degree to which the observed coefficients are statistically significant. It is important to note that the assumptions of the underlying probabilistic model must be fully verified before using this form of interpretation. Below we present a cautionary example where different assumptions lead to opposing conclusions being drawn from the same dataset.

▷ **Ex.** Here, we consider the lawsuit *Students for Fair Admissions, Inc. v. Harvard* regarding the use of race in undergraduate admissions to Harvard University. Initial reports by Harvard's Office of Institutional Research used logistic regression to model the probability of admission using different features of an applicant's profile, including their race [192]. This analysis found that the coefficient associated with being Asian (and not low income) had a coefficient of -0.418 with a significant p-value (<0.001). This negative coefficient suggested that being Asian had a significant negative association with admission probability.

Subsequent analysis from both sides in the lawsuit attempted to analyze the modeling and assumptions to decide on the significance of race in the model's decision. The plaintiff's expert report [14] suggested that race was being unfairly used by building on the original report from Harvard's Office of Institutional Research. It also incorporates analysis on more subjective factors such as "personal ratings" which seem to hurt Asian students' admission. In contrast, the expert report supporting Harvard University [43] finds that by accounting for certain other variables, the effect of race on Asian students acceptance is no longer significant. Significances derived from statistical tests in regression or logistic regression models at best establish association, but not causation. Hence the analyses from both sides are flawed.

This example demonstrates the practical and misleading consequences of statistical feature importances when used inappropriately.

Visualizations

When dealing with high-dimensional datasets, it can be challenging to quickly understand the complex relationships that a model has learned, making the presentation of the results particularly important. To help deal with this, researchers have developed a number of different visualizations which help to understand what a model has learned. For linear models with regularization, plots of regression coefficient paths show how varying a regularization parameter affects the fitted coefficients. When visualizing convolutional neural networks trained on image data, work has been done on visualizing filters [297, 193], maximally activating responses of individual neurons or classes [180], understanding intra-class variation [282], and grouping different neurons [300]. For Long Short Term Memory Networks (LSTMs), researchers have focused on analyzing the state vector, identifying individual dimensions that correspond to meaningful features (e.g. position in line, within quotes) [124], and building tools to track the model's decision process over the course of a sequence [260].

In the following example, relevant interpretations are produced by using maximal activation images for identifying patterns that drive the response of brain cells.

▷ **Ex.** A recent study visualizes learned information from deep neural networks to understand individual brain cells [2]. In this study, macaque monkeys were shown images while the responses of brain cells in their visual system (area V4) were recorded. Neural networks were trained to predict the responses of brain cells to the images. These neural networks produce accurate fits, but provide little insight into what patterns in the images increase the brain cells response without further analysis. To remedy this, the authors introduce DeepTune, a method which provides a visualization, accessible to neuroscientists and others, of the patterns which activate a brain cell. The main intuition behind the method is to optimize the input of a network to maximize the response of a neural network model (which represent a brain cell).

The authors go on to analyze the major problem of instability. When post hoc visualizations attempt to answer scientific questions, the visualizations must be stable to reasonable perturbations; if there are changes in the visualization due to the choice of a model, it is likely not meaningful. The authors address this explicitly by fitting eighteen different models to the data and using a stable optimization over all the models to produce a final consensus DeepTune visualization.

Analyzing trends and outliers in predictions

When interpreting the performance of an ML model, it can be helpful to look not just at the average accuracy, but also at the distribution of predictions and errors. For example, residual plots can identify heterogeneity in predictions, and suggest particular data points to analyze, such as outliers in the predictions, or examples which had the largest prediction errors. Moreover, these plots be used to analyze trends across the predictions. For instance, in the example below, influence functions are able to efficiently identify mislabelled data points.

Prediction-level interpretation

Prediction-level approaches are useful when a practitioner is interested in understanding how individual predictions are made by a model. Note that prediction-level approaches can sometimes be aggregated to yield dataset-level insights.

Feature importance scores

The most popular approach to prediction-level interpretation has involved assigning importance scores to individual features. Intuitively, a variable with a large positive (negative) score made a highly positive (negative) contribution to a particular prediction. In the deep learning literature, a number of different approaches have been proposed to address this problem [264, 240, 18, 242, 185, 59, 218, 303], with some methods for other models as well [158]. These are often displayed in the form of a heat map highlighting important features. Note that feature importance scores at the prediction-level can offer much more information than feature importance scores at the dataset-level. This is a result of heterogeneity in a nonlinear model: the importance of a feature can vary for different examples as a result of interactions with other features.

While this area has seen progress in recent years, concerns have been raised about the descriptive accuracy of these methods. In particular, [4] shows that many popular methods produce similar interpretations for a trained model versus a randomly-initialized one, and are qualitatively very similar to an edge detector. Moreover, it has been shown that some feature importance scores for CNNs are doing (partial) image recovery which is unrelated to the network decisions [190].

▷ **Ex.** When using ML models to predict sensitive outcomes, such as whether a person should receive a loan or a criminal sentence, it is important to verify that the algorithm is not discriminating against people based on protected attributes, such as race or gender. This problem is often described as ensuring ML models are "fair". In [60], the authors introduce a variable importance measure designed to isolate the contributions of individual variables, such as gender, among a set of correlated variables.

Based on these variable importance scores, the authors construct transparency reports, such as the one displayed in Fig S2. This figure displays the importance of features used to predict that "Mr. Z" is likely to be arrested in the future (an outcome which is often used in predictive policing), with each bar corresponding to a feature provided to the classifier, and the y axis displaying the importance score for that feature. In this instance, the race feature is the largest value, indicating that the classifier is indeed discriminating based on race. Thus, in this instance, prediction-level feature importance scores can identify that a model is unfairly discriminating based on race.

Alternatives to feature importances

While feature importance scores can provide useful insights, they also have a number of limitations [4, 112]. For instance, they are unable to capture when algorithms learn interactions between variables. There is currently an evolving body of work centered around uncovering and addressing these limitations. These methods focus on explicitly capturing and displaying the interactions learned by a neural network [184, 249], alternative forms of interpretations such as textual explanations [226], influential data points [133], and analyzing nearest neighbors [46, 201].

2.5 Future work

Having introduced the PDR framework for defining and discussing interpretable machine learning, we now leverage it to frame what we feel are the field's most important challenges moving forward. Below, we present open problems tied to each of the chapter's three main sections: interpretation desiderata (Sec. 2.2), model-based interpretability (Sec. 2.3), and post hoc interpretability (Sec. 2.4).

Measuring interpretation desiderata

Currently, there is no clear consensus in the community around how to evaluate interpretation methods, although some recent work has begun to address it [68, 152, 93]. As a result, the standard of evaluation varies considerably across different work, making it challenging both for researchers in the field to measure progress, and for prospective users to select suitable methods. Within the PDR framework, to constitute an improvement, a new interpretation method must improve at least one desideratum (predictive accuracy, descriptive accuracy, or relevancy) without unduly harming the others. While improvements in predictive accuracy are easy to measure, measuring improvements in descriptive accuracy and relevancy remains a challenge.

Measuring descriptive accuracy

One way to measure an improvement to an interpretation method is to demonstrate that its output better captures what the ML model has learned, i.e. its descriptive accuracy. However, unlike predictive accuracy, descriptive accuracy is generally very challenging to measure or quantify [112]. As a fall-back, researchers often show individual, cherry-picked, interpretations which seem "reasonable". These kinds of evaluations are limited and unfalsifiable. In particular, these results are limited to the few examples shown, and not generally applicable to the entire dataset.

While the community has not settled on a standard evaluation protocol, there are some promising directions. In particular, the use of simulation studies presents a partial solution. In this setting, a researcher defines a simple generative process, generates a large amount of data from that process, and trains their ML model on that data. Assuming a proper simulation setup, a sufficiently powerful model to recover the generative process, and sufficiently large training data, the trained model should achieve near-perfect generalization accuracy. To compute an evaluation metric, they can then check whether their interpretations recover aspects of the original generative process. For example, [273, 274] train neural networks on a suite of generative models with certain built-in interactions, and test whether their method successfully recovers them. Here, due to the ML model's near-perfect generalization accuracy, we know that the model is likely to have recovered some aspects of the generative process, thus providing a ground truth against which to evaluate interpretations. In a related approach, when an underlying scientific problem has been previously studied, prior experimental findings can serve as a partial ground truth to retrospectively validate interpretations [22].

Demonstrating relevancy to real-world problems

Another angle for developing improved interpretation methods is to improve the relevancy of interpretations for some audience or problem. This is normally done by introducing a novel form of output, such as feature heatmaps [264], rationales [142], feature hierarchies [249] or identifying important elements in the training set [133]. A common pitfall in the current literature is to focus on the novel output, ignoring what real-world problems it can actually solve. Given the abundance of possible interpretations, it is particularly easy for researchers to propose novel methods which do not actually solve any real-world problems.

There have been two dominant approaches for demonstrating improved relevancy. The first, and strongest, is to directly use the introduced method in solving a domain problem. For instance, in one example discussed above [22], the authors evaluated a new interpretation method (iterative random forests) by demonstrating that it could be used to identify meaningful biological Boolean interactions for use in experiments. In instances like this, where the interpretations are used directly to solve a domain problem, their relevancy is indisputable. A second, less direct, approach is the use of human studies, often through services like Amazon's Mechanical Turk. Here, humans are asked to perform certain tasks, such as evaluating how much they trust a model's predictions [249]. While challenging to properly construct and perform, these studies are vital to demonstrate that new interpretation methods are, in fact, relevant to any potential practitioners. However, one shortcoming of this approach is that it is only possible to use a general audience of AMT crowdsourced workers, rather than a more relevant, domain-specific audience.

Model-based

Now that we have discussed the general problem of evaluating interpretations, we highlight important challenges for the two main sub-fields of interpretable machine learning: model-based and post hoc interpretability. Whenever model-based interpretability can achieve reasonable predictive accuracy and relevancy, by virtue of its high descriptive accuracy it is preferable to fitting a more complex model, and relying upon post hoc interpretability. Thus, the main focus for model-based interpretability is increasing its range of possible use cases by increasing its predictive accuracy through more accurate models and transparent feature engineering. It is worth noting that sometimes a combination of model-based and post hoc interpretations is ideal.

Building accurate and interpretable models

In many instances, model-based interpretability methods fail to achieve a reasonable predictive accuracy. In these cases, practitioners are forced to abandon model-based interpretations in search of more accurate models. Thus, an effective way of increasing the potential uses for model-based interpretability is to devise new modeling methods which produce higher predictive accuracy while maintaining their high descriptive accuracy and relevance. Promising examples of this work include the previously discussed examples on estimating pneumonia risk from patient data [47] and Bayesian models for generating rule lists to estimate a patient's risk of stroke [143]. Detailed directions for this work are suggested in [230].

Tools for feature engineering

When we have more informative and meaningful features, we can use simpler modeling methods to achieve a comparable predictive accuracy. Thus, methods that can produce more useful features broaden the potential uses of model-based interpretations. The first main category of work lies in improved tools for exploratory data analysis. By better enabling researchers to interact with and understand their data, these tools (combined with domain knowledge) provide increased opportunities for them to identify helpful features. Examples include interactive environments [132, 207, 270], tools for visualization [21, 283, 281], and data exploration tools [168, 284, 251]. The second category falls under unsupervised learning, which is often used as a tool for automatically finding relevant structure in data. Improvements in unsupervised techniques such as clustering and matrix factorization could lead to more useful features.

Post hoc

In contrast to model-based interpretability, much of post hoc interpretability is relatively new, with many foundational concepts still unclear. In particular, we feel that two of the most important questions to be answered are what an interpretation of an ML model should look like, and how post hoc interpretations can be used to increase a model's predictive accuracy. It has also been emphasized that in high stakes decisions practitioners should be very careful when applying post hoc methods with unknown descriptive accuracy [230].

What should an interpretation of a black-box look like

Given a black-box predictor and real-world problem, it is generally unclear what format, or combination of formats, is best to fully capture a model's behavior. Researchers have proposed a variety of interpretation forms, including feature heatmaps [264], feature hierarchies [249] and identifying important elements in the training set [133]. However, in all instances there is a gap between the simple information provided by these interpretations and what the model has actually learned. Moreover, it is unclear if any of the current interpretation forms can fully capture a model's behaviour, or if a new format altogether is needed. How to close that gap, while producing outputs relevant to a particular audience/problem, is an open problem.

Using interpretations to improve predictive accuracy

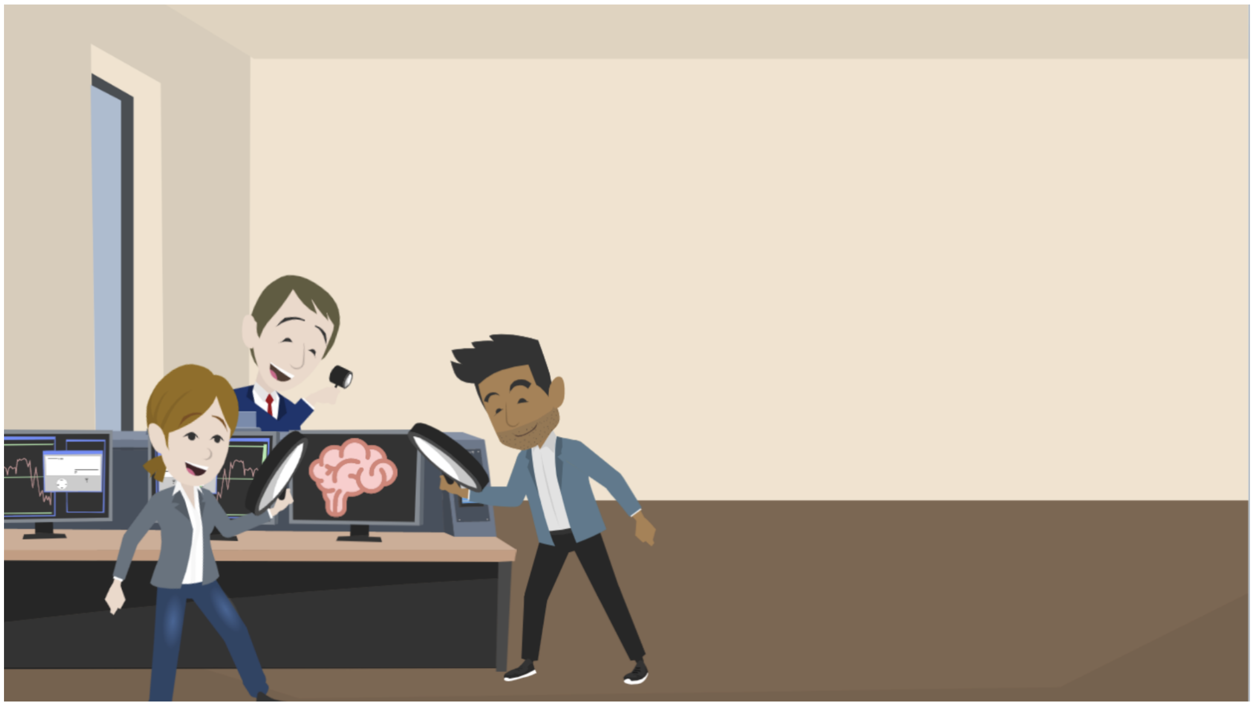
In some instances, post hoc interpretations uncover that a model has learned relationships a practitioner knows to be incorrect. For instance, prior interpretation work has shown that a binary husky vs. wolf classifier simply learns to identify whether there is snow in the image, ignoring the animals themselves [218]. A natural question to ask is whether it is possible for the practitioner to correct these relationships learned by the model, and consequently increase its predictive accuracy. Given the challenges surrounding simply generating post hoc interpretations, research on their uses has been limited [229, 296], particularly in modern deep learning models. However, as the field of post hoc interpretations continues to mature, this could be an exciting avenue for researchers to increase the predictive accuracy of their models by exploiting prior knowledge, independently of any other benefits of interpretations.

Part I

Post-hoc neural-network interpretations

Chapter 3

Hierarchical, disentangled interpretations (ACD)



3.1 Motivating the need for hierarchical interpretations

The success of deep neural networks (DNNs) can largely be attributed to their ability to learn complex, non-linear, relationships between variables. A major hurdle for interpretability is finding a way to to effectively visualize these relationships has led DNNs to be characterized as black boxes. In this chapter, we introduce the use of hierarchical interpretations to explain

DNN predictions. Our proposed method, agglomerative contextual decomposition (ACD)[1], is a general technique that can be applied to a wide range of DNN architectures and data types. Given a prediction from a trained DNN, ACD produces a hierarchical clustering of the input features, along with the contribution of each cluster to the final prediction. This hierarchy is optimized to identify clusters of features that the DNN learned are predictive (see Fig. 3.1).

The development of ACD consists of two novel contributions. First, importance scores for groups of features are obtained by generalizing contextual decomposition (CD), a previous method for obtaining importance scores for LSTMs [184]. This work extends CD to arbitrary DNN architectures, including convolutional neural networks (CNNs). Second, most importantly, we introduce the idea of hierarchical saliency, where a group-level importance measure, in this case CD, is used as a joining metric in an agglomerative clustering procedure. While we focus on DNNs and use CD as our importance measure, this concept is general, and could be readily applied to any model with a suitable measure for computing importances of groups of variables.

We demonstrate the utility of ACD on both long short term memory networks (LSTMs) [109] trained on the Stanford Sentiment Treebank (SST) [254] and CNNs trained on MNIST [139] and ImageNet [233]. Through human experiments, we show that ACD produces intuitive visualizations that enable users to better reason about and trust DNNs. In particular, given two DNN models, we show that users can use the output of ACD to select the model with higher predictive accuracy, and that overall they rank ACD as more trustworthy than prior interpretation methods. In addition, we demonstrate that ACD's hierarchy is robust to adversarial perturbations [265] in CNNs.

[1]Code and scripts for running ACD and experiments available at `https://github.com/csinva/hierarchical-dnn-interpretations`

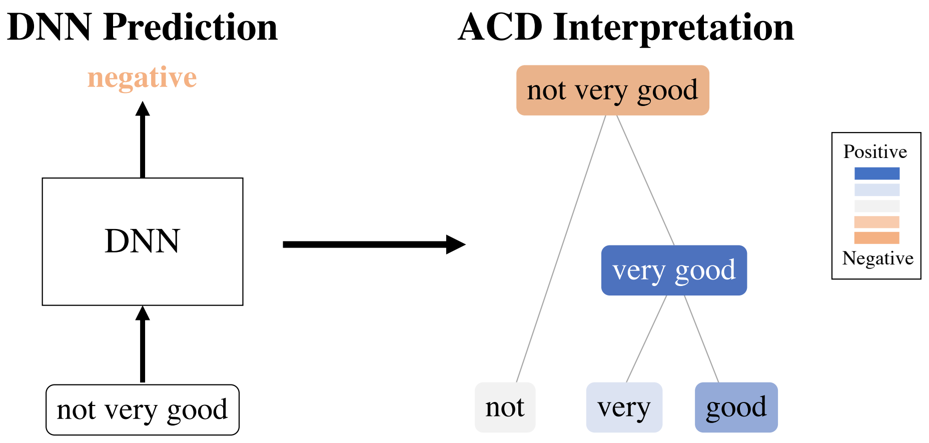


Figure 3.1: ACD illustrated through the toy example of predicting the phrase “not very good” as negative. Given the network and prediction, ACD constructs a hierarchy of meaningful phrases and provides importance scores for each identified phrase. In this example, ACD identifies that “very” modifies “good” to become the very positive phrase “very good”, which is subsequently negated by ”not” to produce the negative phrase “not very good”. Best viewed in color.

3.2 Background on feature importance

Interpreting DNNs is a growing field [186] spanning a range of techniques including feature visualization [193, 291], analyzing learned weights [273] and others [90, 11, 300]. Our work focuses on local interpretations, where the task is to interpret individual predictions made by a DNN.

Local interpretation Most prior work has focused on assigning importance to individual features, such as pixels in an image or words in a document. There are several methods that give feature-level importance for different architectures. They can be categorized as gradient-based [255, 264, 240, 18], decomposition-based [185, 242, 17] and others [59, 83, 218, 303], with many similarities among the methods [10, 159].

By contrast, there are relatively few methods that can extract the interactions between features that a DNN has learned. In the case of LSTMs, [184] demonstrated the limitations of prior work on interpretation using word-level scores, and introduced contextual decomposition (CD), an algorithm for producing phrase-level importance scores from LSTMs. Another simple baseline is occlusion, where a group of features is set to some reference value, such as zero, and the importance of the group is defined to be the resulting decrease in the prediction value [297, 145]. Given an importance score for groups of features, no existing work addresses how to search through the many possible groups of variables in order to find a small set to show to users. To address this problem, this work introduces hierarchical interpretations as a principled way to search for and display important groups.

Hierarchical importance Results from psychology and philosophy suggest that people prefer explanations that are simple but informative [103, 214] and include the appropriate amount of detail [126]. However, there is no existing work that is both powerful enough to capture interactions between features, and simple enough to not require a user to manually search through the large number of available feature groups. To remedy this, we propose a hierarchical clustering procedure to identify and visualize, out of the considerable number of feature groups, which ones contain meaningful interactions and should be displayed to the end user. In doing so, ACD aims to be informative enough to capture meaningful feature interactions while displaying a sufficiently small subset of all feature groups to maintain simplicity.

3.3 Methods for contextual decomposition and ACD

This section introduces ACD through two contributions: Sec. 3.3 proposes a generalization of CD from LSTMs to arbitrary DNNs, and Sec. 3.3 explains the main contribution: how to combine these CD scores with hierarchical clustering to produce ACD.

Extending contextual decomposition (CD) importance scores to general DNNs

In order to generalize CD to a wider range of DNNs, we first reformulate the original CD algorithm into a more generic setting than originally presented. For a given DNN $f(x)$, we can represent its output as a SoftMax operation applied to logits $g(x)$. These logits, in turn, are the composition of L layers g_i, such as convolutional operations or ReLU non-linearities.

$$f(x) = \text{SoftMax}(g(x)) = \text{SoftMax}(g_L(g_{L-1}(...(g_2(g_1(x)))))) \tag{3.1}$$

Given a group of features $\{x_j\}_{j \in S}$, our generalized CD algorithm, $g^{CD}(x)$, decomposes the logits $g(x)$ into a sum of two terms, $\beta(x)$ and $\gamma(x)$. $\beta(x)$ is the importance measure of the feature group $\{x_j\}_{j \in S}$, and $\gamma(x)$ captures contributions to $g(x)$ not included in $\beta(x)$.

$$g^{CD}(x) = (\beta(x), \gamma(x)) \tag{3.2}$$

$$\beta(x) + \gamma(x) = g(x) \tag{3.3}$$

To compute the CD decomposition for $g(x)$, we define layer-wise CD decompositions $g_i^{CD}(x) = (\beta_i, \gamma_i)$ for each layer $g_i(x)$. Here, β_i corresponds to the importance measure of $\{x_j\}_{j \in S}$ to layer i, and γ_i corresponds to the contribution of the rest of the input to layer i. To maintain the decomposition we require $\beta_i + \gamma_i = g_i(x)$ for each i. We then compute CD scores for the full network by composing these decompositions.

$$g^{CD}(x) = g_L^{CD}(g_{L-1}^{CD}(...(g_2^{CD}(g_1^{CD}(x))))) \tag{3.4}$$

Previous work [184] introduced decompositions g_i^{CD} for layers used in LSTMs. The generalized CD described here extends CD to other widely used DNNs, by introducing layer-wise CD decompositions for convolutional, max-pooling, ReLU non-linearity and dropout layers. Doing so generalizes CD scores from LSTMs to a wide range of neural architectures, including CNNs with residual and recurrent architectures.

At first, these decompositions were chosen through an extension of the CD rules detailed in [184], yielding a similar algorithm to that developed concurrently by [94]. However, we found that this algorithm did not perform well on deeper, ImageNet CNNs. We subsequently modified our CD algorithm by partitioning the biases in the convolutional layers between γ_i and β_i in Equation 3.5, and modifying the decomposition used for ReLUs in Equation 3.10.

When g_i is a convolutional or fully connected layer, the layer operation consists of a weight matrix W and a bias b. The weight matrix can be multiplied with β_{i-1} and γ_{i-1} individually, but the bias must be partitioned between the two. We partition the bias proportionally based on the absolute value of the layer activations. For the convolutional layer, this equation yields only one activation of the output; it must be repeated for each activation.

$$\beta_i = W\beta_{i-1} + \frac{|W\beta_{i-1}|}{|W\beta_{i-1}| + |W\gamma_{i-1}|} \cdot b \tag{3.5}$$

$$\gamma_i = W\gamma_{i-1} + \frac{|W\gamma_{i-1}|}{|W\beta_{i-1}| + |W\gamma_{i-1}|} \cdot b \tag{3.6}$$

When g_i is a max-pooling layer, we identify the indices, or channels, selected by max-pool when run by $g_i(x)$, denoted max_idxs below, and use the decompositions for the corresponding channels.

$$max\_idxs = \underset{idxs}{\text{argmax}} \; [\text{maxpool}(\beta_{i-1} + \gamma_{i-1}; idxs)] \tag{3.7}$$

$$\beta_i = \beta_{i-1}[max\_idxs] \tag{3.8}$$

$$\gamma_i = \gamma_{i-1}[max\_idxs] \tag{3.9}$$

Finally, for the ReLU, we update our importance score β_i by computing the activation of β_{i-1} alone and then update γ_i by subtracting this from the total activation.

$$\beta_i = \text{ReLU}(\beta_{i-1}) \tag{3.10}$$

$$\gamma_i = \text{ReLU}(\beta_{i-1} + \gamma_{i-1}) - \text{ReLU}(\beta_{i-1}) \tag{3.11}$$

For a dropout layer, we simply apply dropout to β_{i-1} and γ_{i-1} individually, or multiplying each by a scalar. Computationally, a CD call is comparable to a forward pass through the network f.

Agglomerative Contextual Decomposition (ACD)

Given the generalized CD scores introduced above, we now introduce the clustering procedure used to produce ACD interpretations. At a high-level, our method is equivalent to agglomerative hierarchical clustering, where the CD interaction is used as the joining metric to determine which clusters to join at each step. This procedure builds the hierarchy by starting with individual features and iteratively combining them based on the interaction scores provided by CD. The displayed ACD interpretation is the hierarchy, along with the CD importance score at each node.

More precisely, algorithm 1 describes the exact steps in the clustering procedure. After initializing by computing the CD scores of each feature individually, the algorithm iteratively selects all groups of features within k% of the highest-scoring group (where k is a hyperparameter, fixed at 95 for images and 90 for text) and adds them to the hierarchy.

Each time a new group is added to the hierarchy, a corresponding set of candidate groups is generated by adding individual contiguous features to the original group. For text, the candidate groups correspond to adding one adjacent word onto the current phrase, and for images adding any adjacent pixel onto the current image patch. Candidate groups are ranked according to the CD interaction score, which is the difference between the score of the candidate and original groups.

ACD terminates after an application-specific criterion is met. For sentiment classification, we stop once all words are selected. For images, we stop after some predefined number of iterations and then merge the remaining groups one by one using the same selection criteria described above.

Algorithm 1 is not specific to DNNs; it requires only a method to obtain importance scores for groups of input features. Here, we use CD scores to arrive at the ACD algorithm, which makes the method specific to DNNs, but given a feature group scoring function, Algorithm 1 can yield interpretations for any predictive model. CD is a natural score to use for DNNs as it aggregates saliency at different scales and converges to the final prediction once all the units have been selected.

3.4 ACD succeeds in providing useful qualitative and quantitative interpretation

We now present empirical validation of ACD on both LSTMs trained on SST and CNNs trained on MNIST and ImageNet. First, we introduce the reader to our visualization in Sec. 3.4, and how it can (anecdotally) be used to understand models in settings such as diagnosing incorrect predictions, identifying dataset bias, and identifying representative phrases of differing lengths. We then provide quantitative evidence of the benefits of ACD in Sec. 3.4 through human experiments and demonstrating the stability of ACD to adversarial perturbations.

Algorithm 1 Agglomeration algorithm.

```
ACD(Example x, model, hyperparameter k, function CD(x, blob; model))
  # initialize
  tree = Tree()                                                  # tree to output
  scoresQueue = PriorityQueue()                     # scores, sorted by importance
  for feature in x :
      scoresQueue.push(feature, priority=CD(x, feature; model))

  # iteratively build up tree
  while scoresQueue is not empty :
      selectedGroups = scoresQueue.popTopKPercentile(k)       # pop off top k elements
      tree.add(selectedGroups)                        # Add top k elements to the tree

      # generate new groups of features based on current groups and add them to the queue
      for selectedGroup in selectedGroups :
          candidateGroups = getCandidateGroups(selectedGroup)
          for candidateGroup  in candidateGroups :
              scoresQueue.add(candidateGroup,     priority=CD(x,     candidateGroup;model)-
  CD(x,selectedGroup; model))
  return tree
```

Experimental details

We first describe the process for training the models from which we produce interpretations. As the objective of this paper is to interpret the predictions of models, rather than increase their predictive accuracy, we use standard best practices to train our models. All models are implemented using PyTorch. For SST, we train a standard binary classification LSTM model[2], which achieves 86.2% accuracy. On MNIST, we use the standard PyTorch example[3], which attains accuracy of 97.7%. On ImageNet, we use a pre-trained VGG-16 DNN architecture [244] which attains top-1 accuracy of 42.8%. When using ACD on ImageNet, for computational reasons, we start the agglomeration process with 14-by-14 superpixels instead of individual pixels. We also smooth the computed image patches by adding pixels surrounded by the patch. The weakened models for the human experiments are constructed from the original models by randomly permuting a small percentage of their weights. For SST/MNIST/ImageNet, 25/25/0.8% of weights are randomized, reducing test accuracy from 85.8/97.7/42.8% to 79.8/79.6/32.3%.

[2]model and training code from https://github.com/clairett/pytorch-sentiment-classification

[3]model and training code from https://github.com/pytorch/examples/tree/master/mnist

Qualitative experiments

Before providing quantitative evidence of the benefits of ACD, we first introduce the visualization and demonstrate its utility in interpreting a predictive model's behavior.

Understanding predictive models using ACD

In the following examples, we demonstrate the use of ACD to diagnose incorrect predictions in SST and identify dataset bias in ImageNet. These examples are only a few of the potential uses of ACD.

Text example - diagnosing incorrect predictions In the first example, we show the result of running ACD for our SST LSTM model in Figure 3.2. We can use this ACD visualization to quickly diagnose why the LSTM made an incorrect prediction. In particular, note that the ACD summary of the LSTM correctly identifies two longer phrases and their corresponding sentiment *a great ensemble cast* (positive) and *n't lift this heartfelt enterprise out of the ordinary* (negative). It is only when these two phrases are joined that the LSTM inaccurately predicts a positive sentiment. This suggests that the LSTM has erroneously learned a positive interaction between these two phrases. Prior methods would not be capable of detecting this type of useful information.

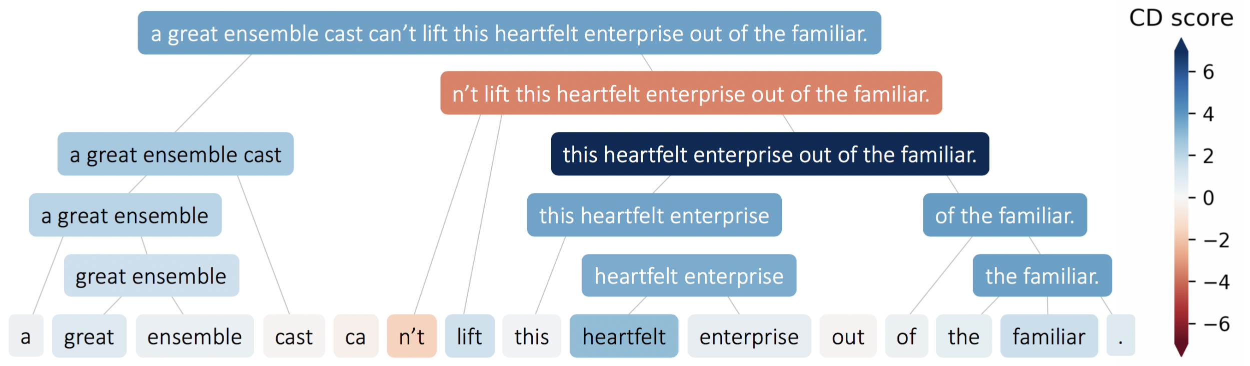

a great ensemble cast can't lift this heartfelt enterprise out of the familiar.
n't lift this heartfelt enterprise out of the familiar.
a great ensemble cast
this heartfelt enterprise out of the familiar.
a great ensemble
this heartfelt enterprise
of the familiar.
great ensemble
heartfelt enterprise
the familiar.
a
great
ensemble
cast
ca
n't
lift
this
heartfelt
enterprise
out
of
the
familiar
.
CD score
6
4
2
0
−2
−4
−6

Figure 3.2: ACD interpretation of an LSTM predicting sentiment. Blue is positive sentiment, white is neutral, red is negative. The bottom row displays CD scores for individual words in the sentence. Higher rows display important phrases identified by ACD, along with their CD scores, converging to the model's (incorrect) prediction in the top row. (Best viewed in color)

Vision example - identifying dataset bias Fig. 3.3 shows an example using ACD for an ImageNet VGG model. Using ACD, we can see that to predict "puck", the CNN is not just focusing on the puck in the image, but also on the hockey player's skates. Moreover, by comparing the fifth and sixth plots in the third row, we can see that the network is only able to distinguish between the class “puck” and the other top classes when the orange skate and green puck patches merge into a single orange patch. This suggests that the CNN has learned that skates are a strong corroborating features for pucks. While intuitively reasonable in the context of ImageNet, this may not be desirable behavior if the model were used in other domains.

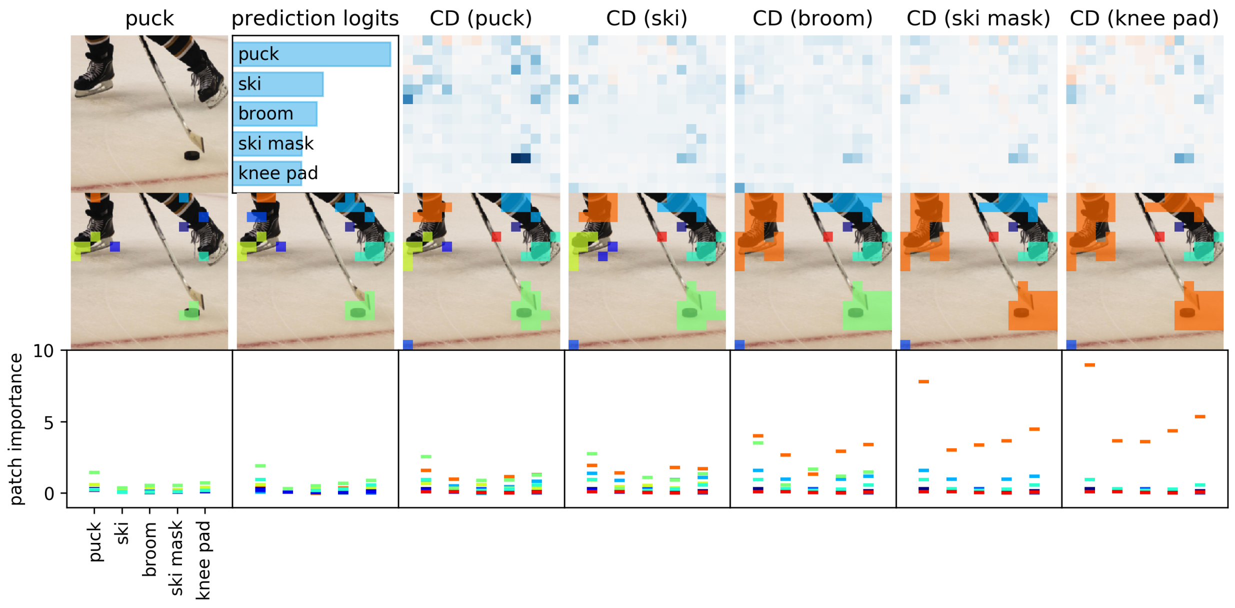


Figure 3.3: ACD interpretation for a VGG network prediction, described in 3.4. ACD shows that the CNN is focusing on skates to predict the class “puck”, indicating that the model has captured dataset bias. The top row shows the original image, logits for the five top-predicted classes, and the CD superpixel-level scores for those classes. The second row shows separate image patches ACD has identified as being independently predictive of the class “puck”. Starting from the left, each image shows a successive iteration in the agglomeration procedure. The third row shows the CD scores for each of these patches, where patch colors in the second row correspond to line colors in the third row. ACD successfully finds important regions for the target class (such as the puck), and this importance increases as more pixels are selected. Best viewed in color.

Identifying top-scoring phrases

When feasible, a common means of scrutinizing what a model has learned is to inspect its most important features, and interactions. In Table 3.1, we use ACD to show the top-scoring phrases of different lengths for our LSTM trained on SST. These phrases were extracted by running ACD separately on each sample in SST’s validation set. The score of each

Length	**Positive**	**Negative**
1	pleasurable, sexy, glorious	nowhere, grotesque, sleep
3	amazing accomplishment., great fun.	bleak and desperate, conspicuously lacks.
5	a pretty amazing accomplishment.	ultimately a pointless endeavour.
8	presents it with an unforgettable visual panache.	my reaction in a word: disappointment.

Table 3.1: Top-scoring phrases of different lengths extracted by ACD on SST's validation set. The positive/negative phrases identified by ACD are all indeed positive/negative.

phrase was then computed by averaging over the score it received in each occurrence in a ACD hierarchy. The extracted phrases are clearly reflective of the corresponding sentiment, providing additional evidence that ACD is able to capture meaningful positive and negative phrases.

Quantitative experiments

Having introduced our visualization and provided qualitative evidence of its uses, we now provide quantitative evidence of the benefits of ACD.

Human experiments

We now demonstrate through human experiments that ACD allows users to better trust and reason about the accuracy of DNNs. Human subjects consist of eleven graduate students at the author's institution, each of whom has taken a class in machine learning. Each subject was asked to fill out a survey with two types of questions: whether, using ACD, they could identify the more accurate of two models and whether they trusted a models output. In both cases, similar questions were asked on three datasets (SST, MNIST and ImageNet), and ACD was compared against three baselines: CD [184], Integrated Gradients (IG) [264], and occlusion [145, 297].

Identifying an accurate model The objective of this section was to determine if subjects could use a small number of interpretations produced by ACD in order to identify the more accurate of two models. For each question in this section, two example predictions were chosen. For each of these two predictions, subjects were given interpretations from two different models (four total), and asked to identify which of the two models had a higher predictive accuracy. Each subject was asked to make this comparison using three different sets of examples for each combination of dataset and interpretation method, for 36 total

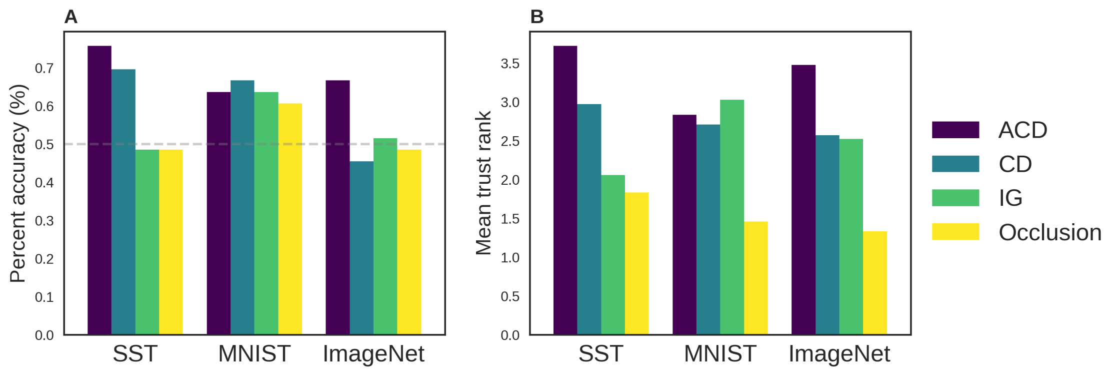


Figure 3.4: Results for human studies. **A.** Binary accuracy for whether a subject correctly selected the more accurate model using different interpretation techniques **B.** Average rank (from 1 to 4) of how much different interpretation techniques helped a subject to trust a model, higher ranks are better.

comparisons. To remove variance due to examples, the same three sets of examples were used across all four interpretation methods.

The predictions shown were chosen to maximize disagreement between models, with SST also being restricted to sentences between five and twenty words, for ease of visualization. To prevent subjects from simply picking the model that predicts more accurately for the given example, for each question a user is shown two examples: one where only the first model predicts correctly and one where only the second model predicts correctly. The two models considered were the accurate models of the previous section and a weakened version of that same model (details given in Sec. 3.4).

Fig 3.4A shows the results of the survey. For SST, humans were better able to identify the strongly predictive model using ACD compared to other baselines, with only ACD and CD outperforming random selection (50%). Based on a one-sided two-sample t-test, the gaps between ACD and IG/Occlusion are significant, but not the gap between ACD and CD. In the simple setting of MNIST, ACD performs similarly to other methods. When applied to ImageNet, a more complex dataset, ACD substantially outperforms prior, non-hierarchical methods, and is the only method to outperform random chance, although the gaps between ACD and other methods are only statistically suggestive (p-values fall between 0.15 and 0.07).

Evaluating trust in a model In this section, the goal is to gauge whether ACD helps a subject to better trust a model's predictions, relative to prior techniques. For each question, subjects were shown interpretations of the same prediction using four different interpretation methods, and were asked to rank the interpretations from one to four based on how much they instilled trust in trust the model. Subjects were asked to do this ranking for three different examples in each dataset, for nine total rankings. The interpretations were produced from the more accurate model from the previous section, and the examples were chosen using the same criteria as the previous section, except they were restricted to examples correctly predicted by the more accurate model.

Fig 3.4B shows the average ranking received by each method/dataset pair. ACD substantially outperforms other baselines, particularly for ImageNet, achieving an average rank of 3.5 out of 4, where higher ranks are better. As in the prior question, we found that the hierarchy only provided benefits in the more complicated ImageNet setting, with results on MNIST inconclusive. For both SST and ImageNet, the difference in mean ranks between ACD and all other methods is statistically significant (p-value less than 0.005) based on a permutation test, while on MNIST only the difference between ACD and occlusion is significant.

ACD hierarchy is robust to adversarial perturbations

While there has been a considerable amount of work on adversarial attacks, little effort has been devoted to qualitatively understanding this phenomenon. In this section, we provide evidence that, on MNIST, the hierarchical clustering produced by ACD is largely robust to adversarial perturbations. This suggests that ACD's hierarchy captures fundamental features of an image, and is largely immune to the spurious noise favored by adversarial examples.

To measure the robustness of ACD's hierarchy, we first qualitatively compare the interpretations produced by ACD on both an unaltered image and an adversarially perturbed version of that image. Empirically, we found that the extracted hierarchies are often very similar. To generalize these observations, we introduce a metric to quantify the similarity between two ACD hierarchies. This metric allows us to make quantitative, dataset-level statements about the stability of ACD feature hierarchies with respect to adversarial inputs. Given an ACD hierarchy, we compute a ranking of the input image's pixels according to the order in which they were added to the hierarchy. To measure the similarity between the ACD hierarchies for original and adversarial images, we compute the correlation between their corresponding rankings. As ACD hierarchies are class-specific, we average the correlations for the original and adversarially altered predictions.

We display the correlations for five different attacks (computed using the Foolbox package [213], each averaged over 100 randomly chosen predictions, in Table 3.2. As ACD is the first local interpretation technique to compute a hierarchy, there is little prior work available for comparison. As a baseline, we use our agglomeration algorithm with occlusion in place of CD. The resulting correlations are substantially lower, indicating that features detected by ACD are more stable to adversarial attacks than comparable methods. These results provide evidence that ACD's hierarchy captures fundamental features of an image, and is largely immune to the spurious noise favored by adversarial examples.

Attack Type	**ACD**	**Agglomerative Occlusion**
Saliency [202]	0.762	0.259
Gradient attack	0.662	0.196
FGSM [96]	0.590	0.131
Boundary [36]	0.684	0.155
DeepFool [179]	0.694	0.202

Table 3.2: Correlation between pixel ranks for different adversarial attacks. ACD achieves consistently high correlation across different attack types, indicating that ACD hierarchies are largely robust to adversarial attacks. Using occlusion in place of CD produces substantially less stable hierarchies.

Chapter 4

Transformation importance (TRIM)

4.1 The need for transformation importance

ACD allows one to attribute importance to interactions between features. However, in many cases, raw features such as pixels in an image or words in a document may not be the most meaningful spaces to perform interpretation. When features are highly correlated or features in isolation are not semantically meaningful, the resulting attributions need to be improved.

To meet this challenge, we propose TRIM (Transformation Importance), an approach for attributing importance to transformations of the input features (see Fig. 4.1). This is critical for making interpretations relevant to a particular audience/problem, as attributions in a domain-specific feature space (e.g. frequencies or principal components) can often be far more interpretable than attributions in the raw feature space (e.g. pixels or biological readings). Moreover, features after transformation can be more independent, semantically meaningful, and comparable across data points. This idea is related to existing works suggesting the use of a "simplified input-representation" [218, 159], but we generalize these works beyond transformations which map existing features into simplified binary features. The work here focuses on combining TRIM with contextual decomposition (CD), an existing attribution method [184], although TRIM can be combined with any local interpretation method.

We focus on cosmology example, where attributing importance to transformations helps understand cosmological models in a more interpretable feature space. Specifically, we consider weak gravitational lensing convergence maps, i.e. maps of the mass distribution in the Universe integrated up to a certain distance from the observer. In a cosmological experiment (e.g. a galaxy survey), these mass maps are obtained by measuring the distortion of distant galaxies caused by the deflection of light by the mass between the galaxy and the observer [20]. These maps contain a wealth of physical information of interest to cosmologists, such as the total matter density in the universe, Ω_m. Current research aims at identifying the most informative features in these maps for inferring the true cosmological parameters. The traditional summary statistic for lensing maps is the power spectrum which is known to be sub-optimal for parameter inference. Tighter parameter constraints can be obtained by including higher-order statistics, such as the bispectrum [57] and peak counts [148]. However, DNN-based inference methods claim to improve on constraints based on these traditional summaries [220, 219, 82].

On top of the accurate predictive power of a DNN, here it is also important to understand what the model learns. Knowing which features are important provides deeper understanding and can be used to design optimal experiments or analysis methods. Moreover, because these models are trained on numerical simulations (realizations of the Universe with different cosmological parameters), it is important to validate that the model uses physical features rather than latching on to numerical artifacts in the simulations. TRIM shows promise for understanding and validating that the DNN learns appropriate physical features by analyzing attributing importance in the spectral domain.

4.2 Calculating transformation importance

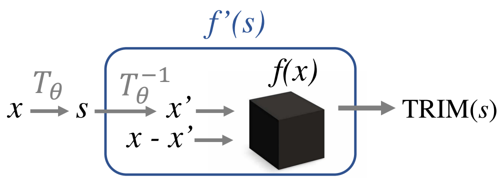


Figure 4.1: TRIM: Attributing importance to a transformation of an input $T_\theta(x)$ given a model $f(x)$.

We aim to interpret the prediction made by a model f given a single input x. The input x is in some domain $\mathcal{X}$, but we desire an explanation for its representation s in a different domain $\mathcal{S}$, defined by a mapping $T : \mathcal{X} \to \mathcal{S}$, such that $s = T(x)$. For example, if x is an image, s may be its Fourier representation, and T would be the Fourier transform. Notably, this process is entirely post-hoc: the model f is already fully trained on the domain $\mathcal{X}$. By reparametrizing our network as shown in Fig. 4.1, we can obtain attributions in the domain $\mathcal{S}$. If we require that the mapping T be invertible, so that $x = T^{-1}(s)$, we can represent each data point x with its counterpart s in the desired domain, and our function to interpret becomes $f' = f \circ T^{-1}$; the function f' can be interpreted with any existing local interpretation method *attr* (e.g. LIME [218], Integrated Gradients [264])).[1] Once we have the reparameterized function $f'(s)$, we need only specify which part of the input to interpret to define TRIM:

Definition 1. Given a model f, an input x, a mask M, a transformation T, and an attribution method *attr*,

[1]Note that if the transformation T is not perfectly invertible (i.e. $x \neq x'$), then the residuals $x - x'$ may also be required for local interpretation. For example, they are required for any gradient-based attribution method to aid in computing $\partial f' / \partial s$.

$$\begin{aligned} \text{TRIM}(s) &= attr\left(f'; s\right) \\ \text{where } f' &= f \circ T^{-1}, s = M \odot T(x) \end{aligned}$$

Here M is a mask used to specify which parts of the transformed space to interpret and $\odot$ denotes elementwise multiplication.

In the work here, the choice of attribution method *attr* is CD, as it can disentangle the importance of features and their interactions, and has been rigorously evaluated using real data [184], human experiments [249], and during model training [222]. In this case, $attr\left(f; x', x\right)$ represents the CD score for the features x' as part of the input x. Different from previous work, this formulation does not require that x' simply be a binary masked version of x. Rather, the selection of the mask M allows a human/domain scientist to decide which transformed features to score. In the case of image classification, rather than simply scoring a pixel, one may score the contribution of a frequency band to the prediction $f(x)$. In this case, T is the FFT and M is a mask which is zero for frequencies outside of the band and 1 for frequencies inside of the band, so that x' represents the bandpass-filtered image.

This general setup allows for attributing importance to a wide array of transformations. For example, T could be any invertible transform (e.g. a wavelet transform), or a linear projection (e.g. onto a sparse dictionary). Moreover, we can parameterize the transformation T_θ and learn the parameters θ to produce a desirable representation (e.g. sparse or disentangled).

4.3 Results for transformation importance

We investigate a text-classification setting using TRIM. We train a 3-layer fully connected DNN with ReLU activations on the Kaggle Fake News dataset[2], achieving a final test accuracy of 94.8%. The model is trained directly on a bag-of words representation, but TRIM can provide a more succinct space via a topic model transformation (learned via latent dirichlet allocation [27]). Fig. 4.2 shows the mean attributions for different topics when the model predicts *Fake*. Interestingly, the topic with the highest mean attribution contain recognizable words such as *clinton* and *emails*.

[2]`https://www.kaggle.com/c/fake-news/overview`

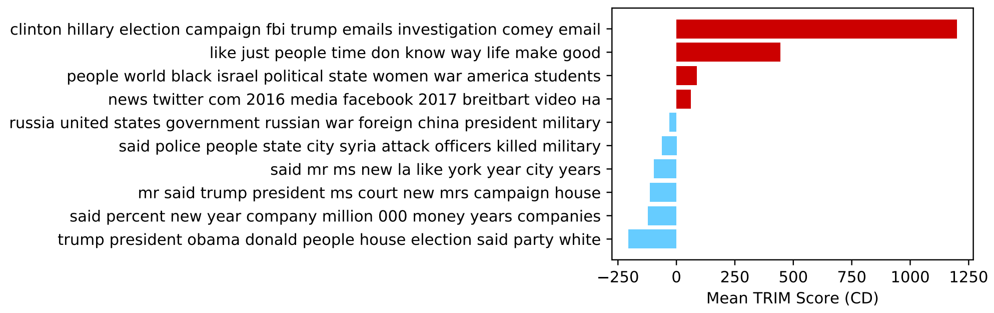


Figure 4.2: TRIM attributions for a fake-news classifier based on a topic model transformation. Each row shows one topic, labeled with the top ten words in that topic. Higher attributions correspond to higher contribution to the class *fake*. Calculated over all points which were accurately classified as *fake* in the test set (4,160 points).

In the case of a perfectly invertible transformation, such as the Fourier transform, TRIM simply measures the ability for the underlying attribution method (in this case CD) to correctly attribute importance in the transformed space. As such, we can rely on the careful evaluation of contextual decomposition in previous work, where it has been shown to (1) accurately recover known feature importances and feature interactions [184], (2) correctly inform human decision-making and be robust to adversarial perturbations [249], and (3) reliably alter a neural network's predictions when regularized appropriately [222].

On top of these evaluations, we add synthetic simulations showing the ability of CD to recover known groundtruth feature importances. Features are generated i.i.d. from a standard normal distribution. Then, a binary classification outcome is defined by selecting a random frequency and testing whether that frequency is greater than its median value. Finally, we train a 3-layer fully connected DNN with ReLU activations to learn this classification task and then test the ability of different methods to assign this frequency the highest importance. Table 4.1 shows the percentage of errors made by different methods in such a setup. CD has the lowest error on average, compared to popular baselines.

CD	DeepLift [242]	SHAP [159]	Integrated Gradients [264]
0.4 ± 0.282	3.6 ± 0.833	4.0 ± 0.897	4.2 ± 0.876

Table 4.1: Error (%) in recovering a groundtruth important frequency in simulated data using different attribution methods with TRIM, averaged over 500 simulated datasets.

Discussion The results here show promise for TRIM to enable deeper understanding in cosmology and suggest potential uses for TRIM across a variety of different domains. Moreover, the TRIM experiments here can be extended to a much broader class of transformations, which could be selected by a domain expert or optimized to exhibit different desirable properties. Ultimately, we hope TRIM can contribute to a new wave of scientific discovery using machine learning.

Chapter 5

Real-world problem: cosmological parameter prediction

We now turn to a cosmology example, where attributing importance to transformations helps understand cosmological models in a more meaningful feature space. Specifically, we consider weak gravitational lensing convergence maps, i.e. maps of the mass distribution in the Universe integrated up to a certain distance from the observer. In a cosmological experiment (e.g. a galaxy survey), these mass maps are obtained by measuring the distortion of distant galaxies caused by the deflection of light by the mass between the galaxy and the observer [20]. These maps contain a wealth of physical information of interest to cosmologists, such as the total matter density in the universe, Ω_m. Current research aims at identifying the most informative features in these maps for inferring the true cosmological parameters, with DNN-based inference methods often obtaining state-of-the-art results [220, 219, 82].

In this context, it is important to not only have a DNN that predicts well, but also understand what it learns. Knowing which features are important provides deeper understanding and can be used to design optimal experiments or analysis methods. Moreover, because this DNN is trained on numerical simulations (realizations of the Universe with different cosmological parameters), it is important to validate that it uses physical features rather than latching on to numerical artifacts in the simulations. TRIM can help understand and validate that the DNN learns appropriate physical features by analyzing attributing importance in the spectral domain.

A DNN is trained to accurately predict Ω_m from simulated weak gravitational lensing convergence maps (full details in [252]). To understand what features the model is using, we desire an interpretation in the space of the power spectrum. The images in Fig. 5.1 show how different information is contained within different frequency bands in the mass maps. The plot in Fig. 5.1 shows the TRIM attributions with CD (normalized by the predicted value) for different frequency bands when predicting the parameter Ω_m. Interestingly, the most important frequency band for the predictions seems to peak at scales around $\ell = 10^4$ and then decay for higher frequencies.[1] A physical interpretation of this result is that the DNN concentrates on the most discriminative part of the Power Spectrum, i.e. at scales large enough not to be dominated by sample variance, and smaller than the frequency cutoff at which the simulations lose power due to resolution effects.

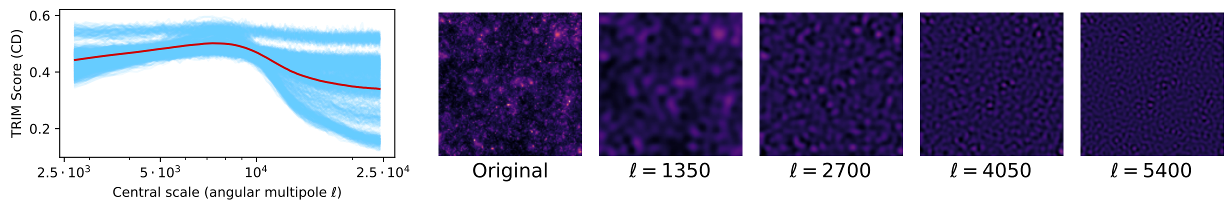


Figure 5.1: Different scales (i.e. frequency bands) contribute differently to the prediction of Ω_m. Each blue line corresponds to one testing image and the red line shows the mean. Images show the features present at different scales. The bandwidth is Δ_ℓ =2,700.

Fig. 5.2 shows some of the curves from Fig. 5.1 separated based on their cosmology, to show how the curves vary with the value of Ω_m. Increasing the value of Ω_m increases the contribution of scales close to $\ell = 10^4$, making other frequencies relatively unimportant. This seems to correspond to known cosmological knowledge, as these scales seem to correspond to galaxy clusters in the mass maps, which are structures very sensitive to the value of Ω_m.The fact that the importance of these features vary with Ω_m would seem to indicate that at lower Ω_m the model is using a different source of information, not located at any single scale, for making its prediction.

[1]Here the unit of frequency used is angular multipole ℓ.

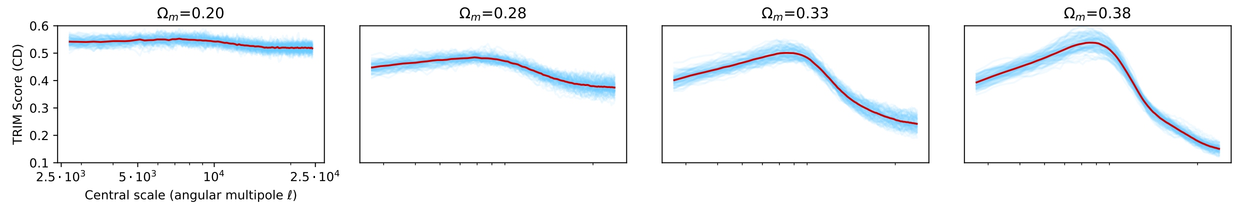


Figure 5.2: TRIM attributions vary with the value of Ω_m.

5.1 Cosmological experiment details

These simulations use the publicly available `MassiveNuS` simulation suite [155], composed of 101 different N-body simulations spanning a range of cosmologies varying three parameters: the total neutrino mass Σm_ν, the normalization of the primordial power spectrum A_s, and the total matter density Ω_m. These simulations are run at a single resolution of 1024^3 particles for a 512 Mpc/h box size, and then ray-traced to obtain lensing convergence maps at source redshifts ranging from z_s =1.0 to $z_s = 1100$. To build our dataset, we select 10 different cosmologies, listed in Table 5.1, each of which provides 10,000 mass maps at source redshift $z_s = 1$. We rebin these maps to size 256x256 with a pixel resolution of 0.8 arcmin.

m_ν	Ω_m	$10^9 A_s$
0.0	0.3	2.1
0.06271	0.3815	2.2004
0.06522	0.2821	1.8826
0.06773	0.4159	1.6231
0.07024	0.2023	2.3075
0.07275	0.3283	2.2883
0.07526	0.3355	1.5659
0.07778	0.2597	2.4333
0.0803	0.2783	2.3824
0.08282	0.2758	1.8292

Table 5.1: Parameter values used in cosmology simulations.

Peak counting algorithm

Here we describe the peak counting algorithm developed in [219] to compare the performance of various filters. In weak lensing, peaks are defined as local maxima on the lensing convergence maps. In the original peak counting algorithm, a histogram is made for each convergence map based on counting the raw pixel (height) values of the peaks on the maps (see Fig. 5.3). At training time, the mean histograms and the covariance matrices are then created for each setting of the cosmological parameters $\xi = (m_\nu, \Omega_m, 10^9 A_s)$; and at test time, individual histograms are compared to the mean histograms via the distance

$$d_{h,\xi} = (h - \mu_\xi)^\top \Sigma_\xi^{-1} (h - \mu_\xi),$$

and the parameters ξ with the lowest distance $d_{h,\xi}$ is selected as prediction values. Here h represents the histogram for a given map, and μ_ξ, Σ_ξ, respectively, represent the mean histogram and the covariance matrix of the histograms for a cosmology with parameters ξ.

In [219], the peak counting algorithm is generalized to exploit more information around the peaks compared with the height of the peaks. Inspired by the first layer of the trained CNN for parameter estimation, they propose to use peak steepness based on the isotropic Laplace filter,

$$L = -\frac{10}{3} \begin{pmatrix} -0.05 & -0.2 & -0.05 \\ -0.2 & 1 & -0.2 \\ -0.05 & -0.2 & -0.05 \end{pmatrix},$$

which computes the difference of the peaks and the surrounding pixel values, or the Roberts cross kernels,

$$R_x = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, R_y = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

which compute the gradient at the peaks. For the Laplace filter, the peak steepness values are calculated via convolving the filter with the input images at the position of the peaks. For the Roberts cross kernels, the two filters R_x and R_y are applied to the 4 adjacent 2×2 pixel blocks around the peaks and the magnitudes are calculated via $G_i = \sqrt{G_{x,i}^2 + G_{y,i}^2}, i = 1, \ldots, 4$, where $G_{x,i}$ and $G_{y,i}$ are the sub-images after convolve R_x and R_y with the i-th adjacent pixel blocks. Then the sum of the 4 magnitudes $\sum_{i=1}^4 G_i$ is used to get the peak steepness values.

Here we further use the wavelet filters distilled by AWD as peak-finding filters in the peak counting algorithm. To match the size of the distilled AWD filters with that of the Laplace filter or Roberts cross kernels, we extract 3×3 subfilters from the wavelet filters where a majority of the mass is concentrated on. This results in 4 different 3×3 filters, corresponding to three wavelet filters (LH,HL,HH) and one approximation filter (LL), which are then used as peak-finding filters to calculate the histograms of the peak steepness values. Fig. 5.3 shows the distributions of peak steepness values using various filters mentioned above.

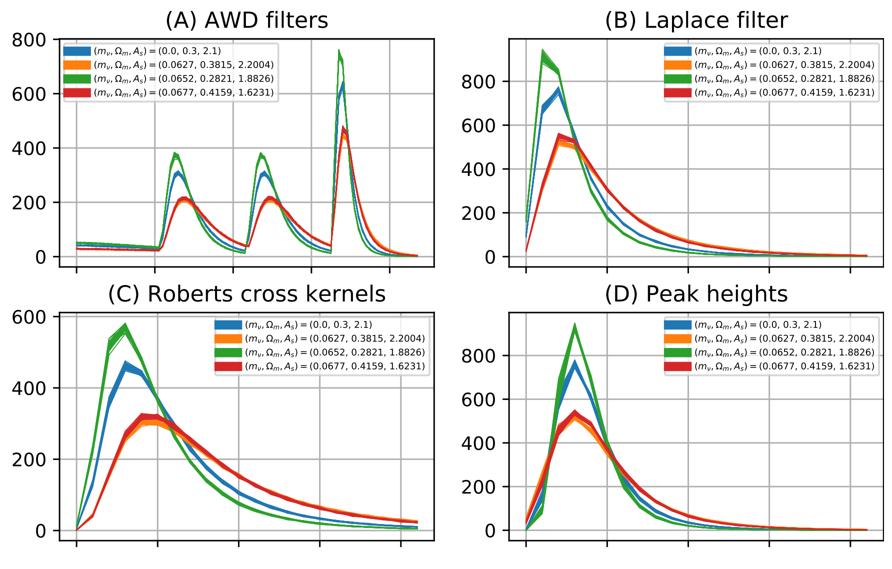


Figure 5.3: Peak steepness distributions using various filters.

To run the peak-counting algorithm with various filters, we need to select the number, width, and range of bins. For the Laplace filter and Roberts cross kernels, we use the same settings as [219] which runs bins from 0 to 0.22 in 0.01 wide. In the case of the wavelet filters, we keep the same number of bins while the range is chosen via the algorithm's performance on a held-out validation set. The resulting bin is then used to evaluate the prediction performance on the test set.

Wavelet activation maps

As part of our interpretability analysis, we now show images that highlight important features for predicting Ω_m (total fraction of matter in the universe) in Fig. 5.4. To create the images, for each map we calculate feature attributions on the wavelet domain extracted by AWD using TRIM (here we use IG [264] to get attributions). Then only the wavelet coefficients with top 600 attributions (out of $73,839$) are retained to transform back to the image domain using inverse wavelet transform. We can see that the activation maps highlight localized regions in the original maps that correspond to the high intensity peaks and voids. This is consistent with the known cosmology theory that these peaks contain high constraining power to predict cosmological parameters of the universe.

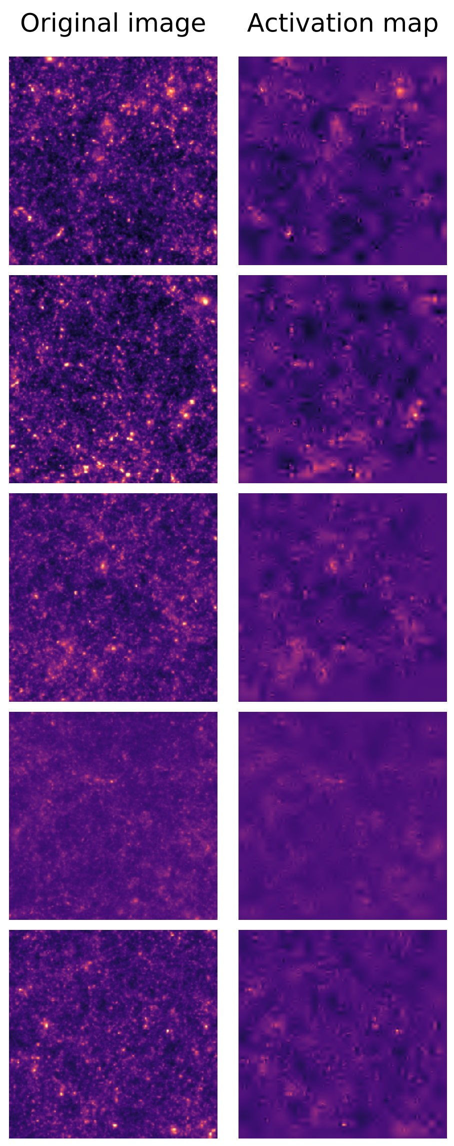


Figure 5.4: Wavelet activation maps for individual images made by the AWD model.

Part II

Leveraging neural-network interpretations to improve models

Chapter 6

Penalizing explanations to align neural networks with prior knowledge (CDEP)

6.1 Intro to directly improving models with explanations

While much work has been put into developing methods for explaining DNNs, relatively little work has explored the potential to use these explanations to help build a better model. Some recent work proposes forcing models to attend to certain regions [40, 176, 70], penalizing the gradients or expected gradients of a neural network [229, 19, 70, 228, 154, 76], or using layer-wise relevance propagation to prune/improve models [263, 290]. A newly emerging line of work investigates how domain experts can use explanations during the training loop to improve their models (e.g. [237]).

Here, we cover contextual decomposition explanation penalization (CDEP), a method which leverages CD to enable the insertion of domain knowledge into a model [223]. Given prior knowledge in the form of importance scores, CDEP works by allowing the user to directly penalize importances of certain features or feature interactions. This forces the DNN to not only produce the correct prediction, but also the correct explanation for that prediction. CDEP can be applied to arbitrary DNN architectures and is often orders of magnitude faster and more memory efficient than recent gradient-based methods [229, 76]; CDEP offers significant computational improvements, since, unlike gradient-based attributions, the CD score is computed along the forward pass, only first derivatives are required for optimization, early layers can be frozen, and all activations of a DNN do not need to be cached to perform backpropagation; furthermore, with gradient-based methods the training requires the storage of activations and gradients for all layers of the network as well as the gradient with respect to the input, whereas penalizing CD requires only a small constant amount of memory more than standard training.[1]

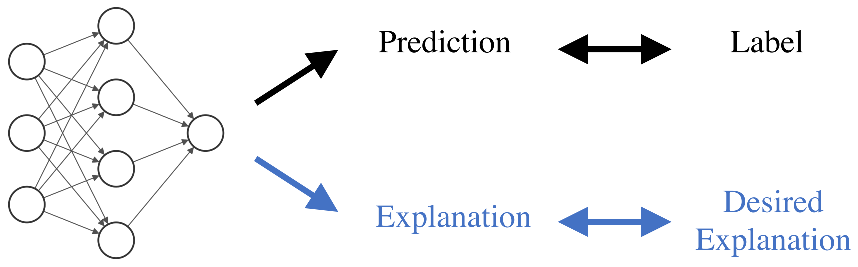


Figure 6.1: CDEP allows practitioners to penalize both a model's prediction and the corresponding explanation.

While we focus on the use of contextual decomposition, which allows the penalization of both feature importances and interactions [184, 249], CDEP can be readily adapted for existing interpretation techniques, as long as they are differentiable. Moreover, CDEP is a general technique, which can be applied to arbitrary neural network architectures, and is often orders of magnitude faster and more memory efficient than recent gradient-based methods, allowing its use on meaningful datasets.

We demonstrate the effectiveness of CDEP via experiments across a wide array of tasks. In the prediction of skin cancer from images, CDEP improves the prediction of a classifier by teaching it to ignore spurious confounders present in the training data.

In a variant of the MNIST digit-classification task where the digit's color is used as a misleading signal, CDEP regularizes a network to focus on a digit's shape rather than its color. Finally, simple examples show how CDEP can help mitigate fairness issues, both in text classification and risk prediction.

6.2 Background on using interpretations as regularization

Explanation methods Many methods have been developed to help explain the learned relationships contained in a DNN. For local or prediction-level explanation, most prior work has focused on assigning importance to individual features, such as pixels in an image or words in a document. There are several methods that give feature-level importance for different architectures. They can be categorized as gradient-based [255, 264, 240, 18, 221], decomposition-based [185, 242, 17] and others [59, 83, 218, 303], with many similarities

[1]Code, notebooks, scripts, documentation, and models for reproducing experiments here and using CDEP on new models available at `https://github.com/laura-rieger/deep-explanation-penalization`.

among the methods [10, 159]. However, many of these methods have been poorly evaluated so far [4, 190], casting doubt on their usefulness in practice. Another line of work, which we build upon, has focused on uncovering interactions between features [184], and using those interactions to create a hierarchy of features displaying the model's prediction process [249, 252].

Uses of explanation methods While much work has been put into developing methods for explaining DNNs, relatively little work has explored the potential to use these explanations to help build a better model. Some recent work proposes forcing models to attend to regions of the input which are known to be important [40, 176], although it is important to note that attention is often not the same as explanation [119].

An alternative line of work proposes penalizing the gradients of a neural network to match human-provided binary annotations and shows the possibility to improve performance [229, 19, 70] and adversarial robustness [228]. Two recent papers extend these ideas by penalizing gradient-based attributions for natural language models [154] and to produce smooth attributions [76]. [70] applies a similar idea to improve image segmentation by incorporating attention maps into the training process.

Predating deep learning, Zaidan, Eisner, and Piatko (2007) consider the use of "annotator rationales" in sentiment analysis to train support vector machines. This work on annotator rationales was recently extended to show improved explanations (not accuracy) in particular types of CNNs [262].

Other ways to constrain DNNs While we focus on the use of explanations to constrain the relationships learned by neural networks, other approaches for constraining neural networks have also been proposed. A computationally intensive alternative is to augment the dataset in order to prevent the model from learning undesirable relationships, through domain knowledge [28], projecting out superficial statistics [279] or dramatically altering training images [92]. However, these processes are often not feasible, either due to their computational cost or the difficulty of constructing such an augmented data set. Adversarial training has also been explored [301]. These techniques are generally limited, as they are often tied to particular datasets, and do not provide a clear link between learning about a model's learned relationships through explanations, and subsequently correcting them.

6.3 CDEP methodology

In the following, we will first establish the general form of the augmented loss function. We then describe Contextual Decomposition (CD), the explanation method proposed by [184]. Based on this, we introduce CDEP and point out its desirable computational properties for regularization. In Sec. 6.3 we describe how prior knowledge can be encoded into explanations and give examples of typical use cases. While we focus on CD scores, which allow the penalization of interactions between features in addition to features themselves, our approach readily generalizes to other interpretation techniques, as long as they are differentiable.

Augmenting the loss function

Given a particular classification task, we want to teach a model to not only produce the correct prediction but also to arrive at the prediction for the correct reasons. That is, we want the model to be right for the right reasons, where the right reasons are provided by the user and are dataset-dependent. Assuming a truthful explanation method, this implies that the explanation provided by the DNN for a particular decision should be aligned with a pre-supplied explanation encoding our knowledge of the underlying reasons.

To accomplish this, we augment the traditional objective function used to train a neural network, as displayed in Eq. (6.1) with an additional component. In addition to the standard prediction loss $\mathcal{L}$, which teaches the model to produce the correct predictions by penalizing wrong predictions, we add an explanation error $\mathcal{L}_{\text{expl}}$, which teaches the model to produce the correct explanations for its predictions by penalizing wrong explanations.

In place of the prediction and labels $f_\theta(X), y$, used in the prediction error $\mathcal{L}$, the explanation error $\mathcal{L}_{\text{expl}}$ uses the explanations produced by an interpretation method $\text{expl}_\theta(X)$, along with targets provided by the user expl_X. As is common with penalization, the two losses are weighted by a hyperparameter $\lambda \in \mathbb{R}$:

$$\begin{aligned}\hat{\theta} = \underset{\theta}{\text{argmin}}\ & \overbrace{\mathcal{L}\left(f_\theta(X), y\right)}^{\text{Prediction error}} \\ & + \lambda \underbrace{\mathcal{L}_{\text{expl}}\left(\text{expl}_\theta(X), \text{expl}_X\right)}_{\text{Explanation error}}\end{aligned} \tag{6.1}$$

The precise meaning of expl_X depend on the context. For example, in the skin cancer image classification task described in Sec. 6.4, many of the benign skin images contain band-aids, while none of the malignant images do. To force the model to ignore the band-aids in making their prediction, in each image $\text{expl}_\theta(X)$ denotes the importance score of the band-aid and expl_X would be zero. These and more examples are further explored in Sec. 6.4.

CDEP objective function

We substitute the (A)CD scores introduced in Chapter 3 into the generic equation in Eq. (6.1) to arrive at CDEP as it is used in this paper. While we use CD for the explanation method $\text{expl}_\theta(X)$, other explanation methods could be readily substituted at this stage. In order to convert CD scores to probabilities, we apply a SoftMax operation to $g^{CD}(x)$, allowing for easier comparison with the user-provided labels expl_X. We collect from the user, for each input x_i, a collection of feature groups $x_{i,S}$, $x_i \in \mathbb{R}^d, S \subseteq \{1, ..., d\}$, along with explanation target values $\text{expl}_{x_{i,S}}$, and use the $\| \cdot \|_1$ loss for $\mathcal{L}_{\text{expl}}$.

This yields a vector $\beta(x_j)$ for any subset of features in an input x_j which we would like to penalize. We can then collect ground-truth label explanations for this subset of features, expl_{x_j} and use it to regularize the explanation. Using this we arrive at the equation for the weight parameters with CDEP loss:

$$\hat{\theta} = \underset{\theta}{\text{argmin}} \overbrace{\sum_i \sum_c -y_{i,c} \log f_\theta(x_i)_c}^{\text{Prediction error}} + \underbrace{\lambda \sum_i \sum_S ||\beta(x_{i,S}) - \text{expl}_{x_{i,S}}||_1}_{\text{Explanation error}} \tag{6.2}$$

In the above, i indexes each individual example in the dataset, S indexes a subset of the features for which we penalize their explanations, and c sums over each class.

Updating the model parameters in accordance with this formulation ensures that the model not only predicts the right output but also does so for the right (aligned with prior knowledge) reasons. It is important to note that the evaluation of what the right reasons are depends entirely on the practitioner deploying the model. As with the class labels, using wrong or biased explanations will yield a wrong and biased model.

Encoding domain knowledge as explanations

The choice of ground-truth explanations expl_X is dependent on the application and the existing domain knowledge. CDEP allows for penalizing arbitrary interactions between features, allowing the incorporation of a very broad set of domain knowledge.

In the simplest setting, practitioners may precisely provide groundtruth human explanations for each data point. This may be useful in a medical image classifications setting, where data is limited and practitioners can endow the model with knowledge of how a diagnosis should be made. However, collecting such groundtruth explanations can be very expensive.

To avoid assigning human labels, one may utilize programmatic rules to identify and assign groundtruth importance to regions, which are then used to help the model identify important/unimportant regions. For example, Sec. 6.4 uses rules to identify spurious patches in images which should have zero importance and Sec. 6.4 uses rules to identify and assign zero importance to words involving gender.

In a more general case, one may specify importances of different feature interactions. For example in Sec. 6.4 we specify that the importance of pixels in isolation should be zero, so only interactions between pixels can be used to make predictions. This prevents a model from latching onto local cues such as color and texture when making its prediction.

Computational considerations

Previous work has proposed ideas similar to Eq. (6.1), where the choice of explanation method is based on gradients [229, 76]. However, using such methods leads to three main complications which are solved by our approach.

The first complication is the optimization process. When optimizing over gradient-based attributions via gradient descent, the optimizer requires the gradient of the gradient, requiring that all network components be twice differentiable. This process is computationally expensive and optimizing it exactly involves optimizing over a differential equation, often making it intractable. In contrast, CD attributions are calculated along the forward pass of the network, and as a result, can be optimized plainly with back-propagation using the standard single forward-pass and backward-pass per batch.

A second advantage from the use of CD in Eq. (6.2) is the ability to quickly finetune a pre-trained network. In many applications, particularly in transfer learning, it is common to finetune only the last few layers of a pre-trained neural network. Using CD, one can freeze early layers of the network and quickly finetune final layers, as the calculation of gradients of the frozen layers is not necessary.

Third, CDEP incurs much lower memory usage than competing gradient-based methods. With gradient-based methods the training requires the storage of activations and gradients for all layers of the network as well as the gradient with respect to the input (which can be omitted in normal training). Even for the simplest gradient-based methods, this more than doubles the required memory for a given batch and network size, sometimes becoming prohibitively large. In contrast, penalizing CD requires only a small constant amount of memory more than standard training.

6.4 CDEP improves predictive performance using domain knowledge

The results here demonstrate the efficacy of CDEP on a variety of datasets using diverse explanation types. Sec. 6.4 shows results on ignoring spurious patches in the ISIC skin cancer dataset [52], Sec. 6.4 details experiments on converting a DNN's preference for color to a preference for shape on a variant of the MNIST dataset [139], Sec. 6.4 showcases the use of CDEP to train a neural network that aligns better with a pre-defined fairness measure, and Sec. 6.4 shows experiments on text data from the Stanford Sentiment Treebank (SST) [254].[2]

Ignoring spurious signals in skin cancer diagnosis

In recent years, deep learning has achieved impressive results in diagnosing skin cancer, with predictive accuracy sometimes comparable to human doctors [77]. However, the datasets used to train these models often include spurious features which make it possible to attain high test accuracy without learning the underlying phenomena [285]. In particular, a popular dataset from ISIC (International Skin Imaging Collaboration) has colorful patches present in approximately 50% of the non-cancerous images but not in the cancerous images as can

[2]All models were trained in PyTorch [203].

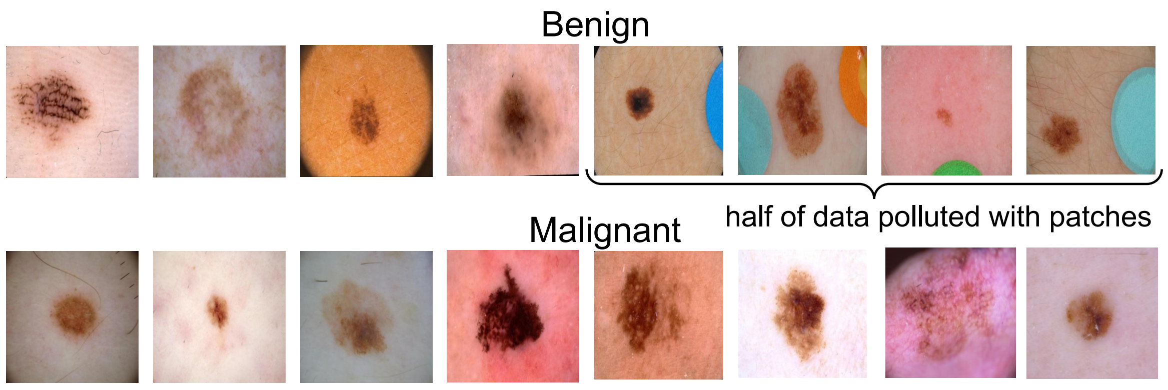


Figure 6.2: Example images from the ISIC dataset. Half of the benign lesion images include a patch in the image. Training on this data results in the neural network overly relying on the patches to classify images. We aim to avoid this with our method.

be seen in Fig. 6.2 [52]. An unpenalized DNN learns to look for these patches as an indicator for predicting that an image is benign as can be seen in Fig. 6.3. We use CDEP to remedy this problem by penalizing the DNN placing importance on the patches during training.

The task in this section is to classify whether an image of a skin lesion contains (1) benign melanoma or (2) malignant melanoma. In a real-life task, this would for example be done to determine whether a biopsy should be taken. The ISIC dataset consists of 21,654 images with a certain diagnosis (19,372 benign, 2,282 malignant), each diagnosed by histopathology or a consensus of experts. We excluded 2247 images since they had an unknown or not certain diagnosis.

To obtain the binary maps of the patches for the skin cancer task, we first segment the images using SLIC, a common image-segmentation algorithm [3]. Since the patches are a different color from the rest of the image, they are usually their own segment. Subsequently we take the mean RGB and HSV values for all segments and filter for segments in which the mean was substantially different from the typical caucasian skin tone. Since different images were different from the typical skin color in different attributes, we filtered for those images recursively. As an example, in the image shown in the appendix in Fig. S3, the patch has a much higher saturation than the rest of the image.

After the spurious patches were identified, we penalized them with CDEP to have zero importance. For classification, we use a VGG16 architecture [244] pre-trained on the ImageNet Classification task[63][3] and freeze the weights of early layers so that only the fully connected layers are trained. To account for the class imbalance present in the dataset, we weigh the classes to be equal in the loss function.

Table 6.1 shows results comparing the performance of a model trained with and without

[3]Pre-trained model retrieved from torchvision.

CDEP. We report results on two variants of the test set. The first, which we refer to as "no patches" only contains images of the test set that do not include patches. The second also includes images with those patches. Training with CDEP improves the AUC and F1-score for both test sets.

Table 6.1: Results from training a DNN on ISIC to recognize skin cancer (averaged over three runs). Results shown for the entire test set and for only the images the test set that do not include patches ("no patches"). The network trained with CDEP generalizes better, getting higher AUC and F1 on both. Std below 0.006 for all AUC and below 0.012 for all F1.

	AUC (no patches)	F1 (no patches)	AUC (all)	F1 (all)
Vanilla (No patches)	0.87	0.57	0.92	0.55
Vanilla	0.93	0.67	0.96	0.67
RRR	0.76	0.45	0.87	0.45
CDEP	**0.95**	**0.73**	**0.97**	**0.73**

In the first row of Table 6.1, the model is trained using only the data without the spurious patches, and the second row shows the model trained on the full dataset. The network trained using CDEP achieves the best F1 score, surpassing both unpenalized versions.

Interestingly, the model trained with CDEP also improves when we consider the entire (biased) dataset, indicating that the model does in fact generalize better to all examples. We also compared our method against the method introduced in 2017 by Ross, Hughes, and Doshi-Velez (RRR). For this, we restricted the batch size to 16 (and consequently use a learning rate of 10^{-5}) due to memory constraints.[4]

Using RRR did not improve on the base AUC, implying that penalizing gradients is not helpful in penalizing higher-order features.[5] In fact, using RRR severely decreased performance in all considered metrics, implying that penalizing gradients not only does not help but impedes the learning of relevant features.

Visualizing explanations To investigate how CDEP altered a DNN's explanations, we visualize GradCAM heatmaps [198, 240] on the ISIC test dataset with a regularized and unregularized network in Fig. 6.3. As expected, after penalizing with CDEP, the DNN attributes less importance to the spurious patches, regardless of their position in the image.

[4]A higher learning rate yields NaN loss and a higher batch size requires too much GPU RAM, necessitating these settings. Due to this a wider sweep of hyperparameters was not possible.

[5]We were not able to compare against the method recently proposed in [76] due to its prohibitively slow training and large memory requirements.

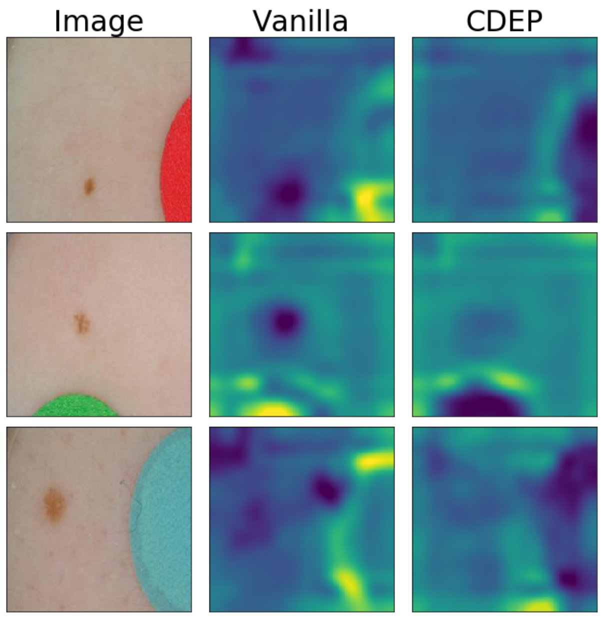


Figure 6.3: Visualizing heatmaps for correctly predicted exampes from the ISIC skin cancer test set. Lighter regions in the heatmap are attributed more importance. The DNN trained with CDEP correctly captures that the patch is not relevant for classification.

More examples are shown in the appendix. Anecdotally, patches receive less attribution when the patch color was far from a Caucasian human skin tone, perhaps because these patches are easier for the network to identify.

Combating inductive bias on variants of the MNIST dataset

In this section, we investigate CDEP's ability to alter which features a DNN uses to perform digit classification, using variants of the MNIST dataset [139] and a standard CNN architecture for this dataset retrieved from PyTorch [6].

ColorMNIST Similar to one previous study [147], we alter the MNIST dataset to include three color channels and assign each class a distinct color, as shown in Fig. 6.4. An unpenalized DNN trained on this biased data will completely misclassify a test set with inverted colors, dropping to 0% accuracy (see Table 6.2), suggesting that it learns to classify using the colors of the digits rather than their shape.

[6]Retrieved from github.com/pytorch/examples/blob/master/mnist.

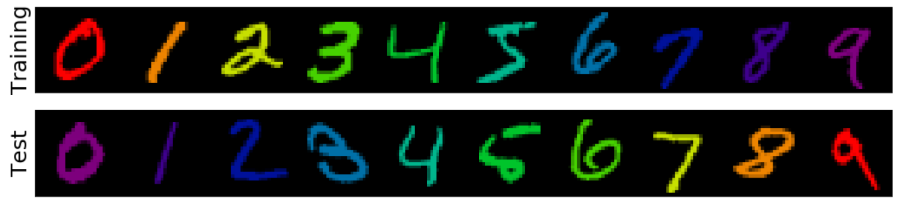


Figure 6.4: ColorMNIST: the shapes remain the same between the training set and the test set, but the colors are inverted.

Here, we want to see if we can alter the DNN to focus on the shape of the digits rather than their color. We stress that this is a toy example where we artificially induced a bias; while the task could be easily solved by preprocessing the input to only have one color channel, this artificial bias allows us to measure the DNN's reliance on the confounding variable *color* in end-to-end training. By design, the task is intuitive and the bias is easily recognized and ignored by humans. However, for a neural network trained in a standard manner, ignoring the confounding variable presents a much greater challenge.

Interestingly, this task can be approached by minimizing the contribution of pixels in isolation (which only represent color) while maximizing the importance of groups of pixels (which can represent shapes). To do this, we penalize the CD contribution of sampled single pixel values, following Eq. (6.2). By minimizing the contribution of single pixels we encourage the network to focus instead on groups of pixels. Since it would be computationally expensive and not necessary to apply this penalty to every pixel in every training input, we sample pixels to be penalized from the average distribution of nonzero pixels over the whole training set for each batch.

Table 6.2 shows that CDEP can partially divert the network's focus on color to also focus on digit shape. We compare CDEP to two previously introduced explanation penalization techniques: penalization of the squared gradients (RRR) [229] and Expected Gradients (EG) [76] on this task. For EG we additionally try penalizing the variance between attributions of the RGB channels (as recommended by the authors of EG in personal correspondence). None of the baselines are able to improve the test accuracy of the model on this task above the random baseline, while CDEP is able to significantly improve this accuracy to 31.0%. We show the increase of predictive accuracy with increasing penalization in the appendix. Increasing the regularizer rate for CDEP increases accuracy on the test set, implying that CDEP meaningfully captured and penalized the bias towards color.

DecoyMNIST For further comparison with previous work, we evaluate CDEP on an existing task: DecoyMNIST [76]. DecoyMNIST adds a class-indicative gray patch to a random corner of the image. This task is relatively simple, as the spurious features are

Table 6.2: Test Accuracy on ColorMNIST and DecoyMNIST. CDEP is the only method that captures and removes color bias. All values averaged over thirty runs. Predicting at random yields a test accuracy of 10%.

	Vanilla	CDEP	RRR	Expected Gradients
ColorMNIST	0.2 ± 0.2	**31.0 ± 2.3**	0.2 ± 0.1	10.0 ± 0.1
DecoyMNIST	60.1 ± 5.1	**97.2 ± 0.8**	**99.0 ± 1.0**	**97.8 ± 0.2**

not entangled with any other feature and are always at the same location (the corners). Table 6.2 shows that all methods perform roughly equally, recovering the base accuracy. Results are reported using the best penalization parameter λ, chosen via cross-validation on the validation set. We provide details on the computation time, and memory usage in Table S1, showing that CDEP is similar to existing approaches. However, when freezing early layers of a network and finetuning, CDEP very quickly becomes more efficient than other methods in both memory usage and training time.

Fixing bias in COMPAS

In all examples so far, the focus has been on improving generalization accuracy. Here, we turn to improving notions of fairness in models while preserving prediction accuracy instead.

We train and analyze DNNs on the COMPAS dataset [138], which contains data for predicting recidivism (i.e whether a person commits a crime / a violent crime within 2 years) from many attributes. Such models have been used for the purpose of informing whether defendants should be incarcerated and can have very serious implication. As a result, we examine and influence the model's treatment of race, restricting our analysis to the subset of people in the dataset whose race is identified as *black* or *white* (86% of the full dataset). All models were fully connected DNNs with two hidden layers of size 5, ReLU nonlinearity, and dropout rate of 0.1 (see appendix for details).

We analyze the effect of CDEP to alter models with respect to one particular notion of fairness: the wrongful conviction rate (defined as the fraction of defendants who are recommended for incarceration, but did not recommit a crime in the next two years). We aim to keep this rate low and relatively even across races, similar to the common "equalized odds" notion of fairness [67]; note that a full investigation of fairness and its most appropriate definition is beyond the scope of the work here.

Table 6.3 shows results for different models trained on the COMPAS dataset. The first row shows a model trained with standard procedures and the second row shows a model trained with the race of the defendants hidden. The unregularized model in the first row has a stark difference in the rates of false positives between *black* and *white* defendants. Black defendants are more than twice as likely to be misclassified as high-risk for future crime. This is in-line with previous analysis of the COMPAS dataset [138].

Obscuring the sensitive attribute from the model does not remove this discrepancy. This is due to the fact that black and white people come from different distributions (e.g. black defendants have a different age distribution).

The third row shows the results for CDEP, where the model is regularized to place more importance on the race feature and its interactions, encouraging it to learn the dependence between race and the distribution of other features. By doing so, the model achieves a lower wrongful conviction rate for both black and white defendants, as well as bringing these rates noticeably closer together by disproportionally lowering the wrongful conviction rate for black defendants. Notably, the test accuracy of the model stays relatively fixed despite the drop in wrongful conviction rates.

Table 6.3: Fairness measures on the COMPAS dataset. WCR stands for wrongful conviction rate the fraction of innocent defendants who are recommended for incarceration). All values averaged over five runs.

	Test acc	WCR(Black)	WCR(White)
Vanilla	67.8±1.0	0.47± 0.03	0.22±0.03
Race hidden	68.5±0.3	0.44±0.02	0.23±0.01
CDEP	**68.8±0.3**	**0.39±0.04**	**0.20± 0.01**

Fixing bias in text data

To demonstrate CDEP's effectiveness on text, we use the Stanford Sentiment Treebank (SST) dataset [254], an NLP benchmark dataset consisting of movie reviews with a binary sentiment (positive/negative). We inject spurious signals into the training set and train a standard LSTM [7] to classify sentiment from the review.

We create three variants of the SST dataset, each with different spurious signals which we aim to ignore (examples in the appendix). In the first variant, we add indicator words for each class (positive: 'text', negative: 'video') at a random location in each sentence. An unpenalized DNN will focus only on those words, dropping to nearly random performance on the unbiased test set. In the second variant, we use two semantically similar words ('the', 'a') to indicate the class by using one word only in the positive and one only in the negative class. In the third case, we use 'he' and 'she' to indicate class (example in Fig. 6.5). Since these gendered words are only present in a small proportion of the training dataset ($\sim 2\%$),

[7]Retrieved from github.com/clairett/pytorch-sentiment-classification.

Positive

pacino is the best **she**'s been in years and keener is marvelous
she showcases davies as a young woman of great charm, generosity and diplomacy

Negative

i'm sorry to say that this should seal the deal - arnold is not, nor will **he** be, back.
this is sandler running on empty, repeating what **he**'s already done way too often.

Figure 6.5: Example sentences from the SST dataset with artificially induced bias on gender.

for this variant, we report accuracy only on the sentences in the test set that do include the pronouns (performance on the test dataset not including the pronouns remains unchanged). Table 6.4 shows the test accuracy for all datasets with and without CDEP. In all scenarios, CDEP is successfully able to improve the test accuracy by ignoring the injected spurious signals.

Table 6.4: Results on SST. CDEP substantially improves predictive accuracy on the unbiased test set after training on biased data.

	Unpenalized	CDEP
Random words	56.6 ± 5.8	**75.4** ± **0.9**
Biased (articles)	57.8 ± 0.8	**68.2** ± **0.8**
Biased (gender)	64.2 ± 3.1	**78.0** ± **3.0**

6.5 Limitations and extensions of CDEP

In this work we introduce a novel method to penalize neural networks to align with prior knowledge. Compared to previous work, CDEP is the first of its kind that can penalize complex features and feature interactions. Furthermore, CDEP is more computationally efficient than previous work, enabling its use with more complex neural networks.

We show that CDEP can be used to remove bias and improve predictive accuracy on a variety of toy and real data. The experiments here demonstrate a variety of ways to use CDEP to improve models both on real and toy datasets. CDEP is quite versatile and can be used in many more areas to incorporate the structure of domain knowledge (e.g. biology or physics). The effectiveness of CDEP in these areas will depend upon the quality of the prior knowledge used to determine the explanation targets.

Future work includes extending CDEP to more complex settings and incorporating more fine-grained explanations and interaction penalizations. We hope the work here will help push the field towards a more rigorous way to use interpretability methods, a point which will become increasingly important as interpretable machine learning develops as a field [68, 186].

Chapter 7

Adaptive wavelet distillation from neural networks through interpretations

7.1 Intro to adaptive wavelet distillation

One promising approach to constructing interpretable models without sacrificing prediction performance is model distillation. Model distillation [108] transfers the knowledge in one model (i.e., the teacher), into another model (i.e., the student), where the student model often has desirable properties, such as being more interpretable than the teacher model. Recent works have considered distilling a DNN into inherently interpretable models such as a decision tree [90, 58, 144] or a global additive model [267], with some success. Here, we consider distilling a DNN into a learnable wavelet transform, which is a powerful tool to describe signals both in time (spatial) and frequency domains that has found numerous successful applications in physical and biomedical sciences.

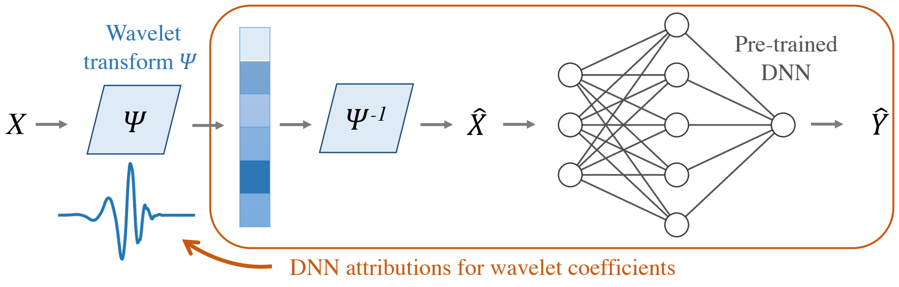


Figure 7.1: Adaptive wavelet distillation uses attributions from a trained DNN to improve its wavelet transform, while satisfying constraints for reconstruction error and wavelet constraints.

Wavelets have many properties amenable to interpretation: they can form an orthogonal basis, identify a sparse representation of a signal, and tile different frequencies and spatial locations (and sometimes rotations), allowing for multiresolution analysis. Most previous work has focused on hand-designed wavelets for different scenarios rather than wavelets which adapt to given data. Recent work has explored wavelets which adapt to an input data distribution, under the name optimized wavelets or adaptive wavelets [216, 234, 151, 31, 107, 215, 120, 266]. Moreover, some work has used wavelets as part of the underlying structure of a neural network, as in wavelet networks [299], wavelet neural networks [298, 65], or the scattering transform [165, 38]. However, none of them utilize wavelets for model interpretability.

Fig. 7.1 outlines Adaptive Wavelet Distillation (AWD), our approach for distilling a wavelet transform from a trained DNN. A key novelty of AWD is that it uses *attributions from a trained DNN* to improve the learned wavelets;[1] this incorporates information not just about the input signals, as is done in previous work, but also about the target variable and the inductive biases present in the DNN.[2]

This paper deviates significantly from a typical NeurIPS paper. While there has been an explosion of work in "interpretable machine learning" [177], there has been very limited development and grounding of these methods *in the context of a particular problem and audience*. This has led to much confusion about how to develop and evaluate interpretation methods [4, 68]; in fact, a major part of the issue is that interpretability cannot be properly defined without the context of a particular problem and audience [186]. As interpretability and scientific machine learning enter a new era, researchers must ground themselves in real-world problems and work closely with domain experts.

This paper focuses on scientific machine learning—providing insight for a particular scientific audience into a chosen scientific problem— and from its outset, was designed to solve a particularly challenging cosmology problem in close collaboration with cosmologists. We showcase how AWD can inform relevant features in a fundamental problem in cosmology: inferring cosmological parameters from weak gravitational lensing convergence maps.[3] In this case, AWD identifies high-intensity peaks in the convergence maps and yields an easily interpretable model which outperforms state-of-the-art neural networks in terms of prediction performance. We next find that AWD successfully provides prediction improvements in another scientific application (now in collaboration with cell-biology experts): molecular-partner prediction. In this case, AWD allows us to vet that the model's use of clathrin corresponds to our domain knowledge about how clathrin must build up slowly then fall in order to predict a successful event. In both cases, the wavelet models from AWD further extract compressed representations of the input in comparison to the standard wavelet model while concisely explaining model behavior. We hope that the depth and grounding of the scientific problems in this work can spur further interpretability research in real-world

[1]By attributions, we mean feature importance scores given input data and a pre-trained DNN.

[2]Though we focus on DNNs, AWD works for any black-box models for which we can attain attributions.

[3]For the purpose of this work, we work with simulated lensing maps.

problems, where interpretability can be evaluated by and enrich domain knowledge, beyond benchmark data contexts such as MNIST [139] where the need for interpretability is less cogent.

7.2 Background on the wavelet transform

Wavelet transform

Wavelets are a class of functions that are localized both in the time and frequency domains. In the classical setting, each wavelet is a variation of a single wavelet ψ, called the *mother wavelet.* A family of discrete wavelets can be created by scaling and translating the mother wavelet in discrete increments:

$$\left\{\psi_{j,n}(t) = \frac{1}{\sqrt{2^j}}\psi\left(\frac{t-2^j n}{2^j}\right)\right\}_{(j,n)\in\mathbb{Z}^2}, \tag{7.1}$$

where each wavelet in the family $\psi_{j,n}(t)$ represents a unique scale and translation of ψ. With a carefully constructed wavelet ψ (see []), the family of wavelets (7.1) forms an orthonormal basis of $L^2(\mathbb{R})$. Namely, any signal $x \in L_2(\mathbb{R})$ can be decomposed into

$$x = \sum_n \sum_j d_j[n]\psi_{j,n}, \tag{7.2}$$

where the wavelet (or detail) coefficients $d_j[n]$ at scale 2^j are computed by taking the inner product with the basis functions, $d_j[n] = \langle x, \psi_{j,n}\rangle = \int x(t)\psi_{j,n}(t)dt$. The decomposition (7.2) requires an infinite number of scalings to calculate the discrete wavelet transform. To make this decomposition computable, the *scaling function* ϕ is introduced so that

$$x = \sum_n a_J[n]\phi_{J,n} + \sum_n \sum_j^J d_j[n]\psi_{j,n}, \tag{7.3}$$

where $\phi_{J,n}(t) = 2^{-J/2}\phi(2^{-J}t - n)$ represent different translations of ϕ at scale 2^J and $a_J[n] = \langle x, \phi_{J,n}\rangle$ are the corresponding approximation coefficients. Conceptually, the $\phi_{J,n}$ form an orthogonal basis of functions that are smoother at the given scale 2^J, and therefore can be used to decompose the smooth residuals not captured by the wavelets [164].

A fundamental property of the discrete wavelet transform is that the approximation and detail coefficients at scale 2^{j+1} can be computed from the approximation coefficients of the previous scale at 2^j [163, 173]. To see this, let us define the two discrete filters, lowpass filter h and highpass filter g

$$h[n] = \langle\frac{1}{\sqrt{2}}\phi(t/2), \phi(t-n)\rangle \quad \text{and} \quad g[n] = \langle\frac{1}{\sqrt{2}}\psi(t/2), \phi(t-n)\rangle. \tag{7.4}$$

Then the following recursive relations hold between the approximation and detail coefficients at two consecutive resolutions:

$$\begin{cases} a_{j+1}[p] = \sum_n h[n-2p]a_j[n] = a_j \star \bar{h}[2p]; \\ d_{j+1}[p] = \sum_n g[n-2p]a_j[n] = a_j \star \bar{g}[2p], \end{cases} \tag{7.5}$$

where we denote $\bar{h}[n] = h[-n]$ and $\bar{g}[n] = g[-n]$. Conversely, the approximation coefficients at scale 2^j can be recovered from the coarser-scale approximation and detail coefficients using

$$a_j[p] = \sum_n h[p-2n]a_{j+1}[n] + \sum_n g[p-2n]d_{j+1}[n]. \tag{7.6}$$

Together, these recursive relations lead to the filter bank algorithm, the cascade of discrete convolution and downsampling, which can be efficiently implemented in time $\mathcal{O}$ (Signal length). The discrete wavelet transform can be extended to two dimensions, using a separable (row-column) implementation of 1D wavelet transform along each axis (see []).

Transformation Importance (TRIM)

The work here requires the ability to compute attributions which identify important features given input data and a trained DNN. For this, we rely on TRIM from Chapter 4, an approach which attributes importance to features in a transformed domain (here, the wavelet domain) via a straightforward model reparameterization.

7.3 Adaptive wavelet distillation through interpretations

Adaptive wavelet distillation (AWD) aims to learn a wavelet transform which effectively represents the input data as well as captures information about a model trained to predict a response using the input data. Depending on a problem's context, the resulting wavelet model may or not be sufficiently interpretable for use, or may or not provide similar or better prediction performance as the pre-trained model f. However, we provide two scientific data problems in section 7.4 where we can do both. In fact, the AWD method has been developed in the context of solving the cosmology problem.

We now detail how AWD wavelets can be built upon to form an extremely simple model in various contexts (see Sec. 7.4). We first require that the wavelet transform is invertible, allowing for reconstruction of the original data. This ensures that the transform does not lose any information in the input data. We next assure that the learned wavelet is a valid wavelet: the wavelet function ψ and the corresponding scaling function ϕ span a sequence of subspaces satisfying the multiresolution axioms [162]. Finally, we add the distillation part of AWD. We calculate the attribution scores of a given model f for each coefficient in the wavelet representation, and try to find a wavelet function ψ that makes these attributions sparse. Intuitively, this ensures that the learned wavelet should find a representation which can concisely explain a model's prediction. Writing the discrete wavelet transform using the discrete filters h and g (see Eq. (7.4)), we now give a final optimization problem for AWD:

$$\underset{h,g}{\text{minimize}}\ \mathcal{L}(h,g) = \underbrace{\frac{1}{m}\sum_i \|x_i - \widehat{x}_i\|_2^2}_{\text{Reconstruction loss}} + \underbrace{\frac{1}{m}\sum_i W(h,g,x_i;\lambda)}_{\text{Wavelet loss}} + \underbrace{\gamma\sum_i \|\text{TRIM}_{\Psi,f}(\Psi x_i)\|_1}_{\text{Interpretation loss}}, \tag{7.7}$$

where Ψ denotes a wavelet transform operator induced by ψ, and $\widehat{x}_i$ denotes the reconstruction of the data point x_i. Here $\lambda, \gamma > 0$ represent hyperparameters that are tuned by users. The only parameters optimized are the lowpass filter h and the highpass filter g. The corresponding scaling and wavelet functions can be obtained from (h,g) via the following mapping [164]: $\widehat{\phi}(w) = \prod_{p=1}^{\infty} \frac{\widehat{h}(2^{-p}w)}{\sqrt{2}}$ and $\widehat{\psi}(w) = \frac{1}{\sqrt{2}}\widehat{g}(w/2)\widehat{\phi}(w/2)$, where $\widehat{\phi}$ and $\widehat{\psi}$ represent the Fourier transforms of ϕ and ψ respectively.

Wavelet loss The wavelet loss ensures that the learned filters yield a valid wavelet transform. In contrast to the wavelet constraints used in [216], our formulation introduces additional terms that ensure almost sufficient and necessary conditions on the filters (h,g) to build an orthogonal wavelet basis. Specifically, [164, Theorem 7.2] states the following sufficient conditions on the lowpass filter: if h satisfies

$$\sum_n h[n] = \sqrt{2} \quad \text{and} \quad |\widehat{h}(w)|^2 + |\widehat{h}(w+\pi)|^2 = 2 \ \text{ for all } w, \tag{7.8}$$

as well as some mild conditions, it can generate a scaling function such that the scaled and translated family of the scaling function forms an orthonormal basis of the space of multiresolution approximations of $L^2(\mathbb{R})$. [41, Theorem 3] further shows that the orthogonality of translates of the scaling function implies that the lowpass filter is orthogonal after translates by 2, i.e.,

$$\sum_n h[n]h[n-2k] = \begin{cases} 1 & \text{if } k = 0 \\ 0 & \text{otherwise} \end{cases}, \quad \text{and as a result, } \|h\|_2 = 1. \tag{7.9}$$

Hence the conditions (7.8), (7.9) characterize the almost sufficient and necessary conditions on the lowpass filter. Moreoever, [164, Theorem 7.3] shows that the valid highpass filter can be constructed from the lowpass filter: in the time domain, it can be written as

$$g[n] = (-1)^n h[N-1-n], \tag{7.10}$$

where N is the support size of h. Together with (7.9), we can also deduce that the highpass filter has mean zero, i.e., $\sum_n g[n] = 0$ which is necessary for the filter g. See [] for further details.

Finally, we want the learned wavelet to provide sparse representations so we add the ℓ_1 norm penalty on the wavelet coefficients. Combining all these constraints via regularization terms, we define the wavelet loss at the data point x_i as

$$W(h, g, x_i; \lambda) = \lambda\|\Psi x_i\|_1 + (\sum_n h[n] - \sqrt{2})^2 + (\sum_n g[n])^2 + (\|h\|_2^2 - 1)^2 \\ + \sum_w (|\widehat{h}(w)|^2 + |\widehat{h}(w+\pi)|^2 - 2)^2 + \sum_k (\sum_n h[n]h[n-2k] - \mathbf{1}_{k=0})^2,$$

where g is set as in (7.10) and $\lambda > 0$ controls strength of the sparsity of the wavelet representations. We enforce the penalty $(|\widehat{h}(w)|^2 + |\widehat{h}(w+\pi)|^2 - 2)^2$, only at the discrete values of $w \in \{\frac{2\pi k}{N}, k = 1, \ldots, N\}$ through the discrete Fourier transform. Notice that the wavelet loss does not introduce any additional hyperparameters besides λ. In fact, we empirically observe that the sum of penalty terms, except the sparsity penalty, remains very close to zero as long as the filters (h, g) are initialized using known wavelet filters and the interpretation loss is not enforced too strongly.

Interpretation loss The interpretation loss enables the distillation of knowledge from the pre-trained model f into the wavelet model. It ensures that attributions in the space of wavelet coefficients Ψx_i are sparse, where the attributions of wavelet coefficients is calculated by TRIM [252], as described in Sec. 7.2. This forces the wavelet transform to produce representations that concisely explain the model's predictions at different scales and locations. The hyperparameter γ controls the overall contribution of the interpretation loss; large values of γ can result in large numerical differences from satisfying the conditions of the mathematical wavelet filters. To our knowledge, this is the first method which uses interpretations from a pre-trained model to improve a wavelet representation. This enables the wavelets to not only adapt to the distribution of the inputs, but also gain information about the predicted outputs through the lens of the model f.

7.4 AWD improves interpretability, prediction performance, and compression in two scientific problems and in simulations

Fig. 7.3 shows a visual schematic of the distillation and prediction setup for one synthetic and two scientific data problems in this section, whose details will be discussed in the following subsections.[4]

[4]In all experiments, the wavelet function is computed from the corresponding lowpass filter using the *PyWavelets* package [141] and building on the *Pytorch Wavelets* [56, Chapter 3] package.

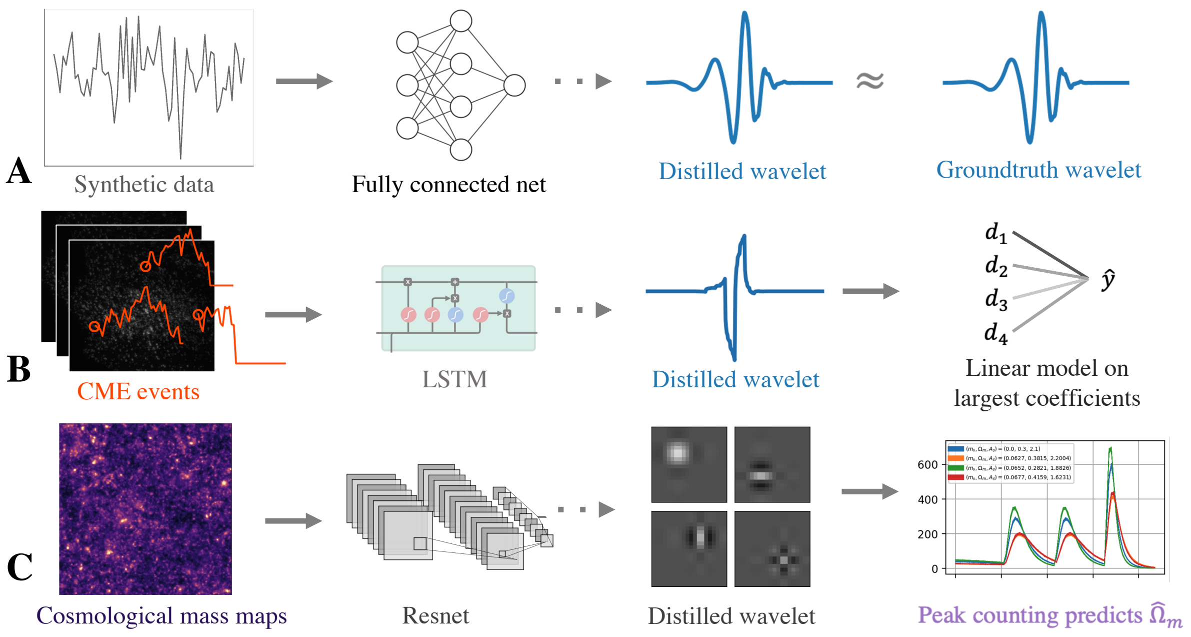


Figure 7.2: Distillation and prediction setup for the three scenarios in Sec. 7.4. (**A**) In synthetic simulations, AWD is able to recover groundtruth wavelet (DB5) that are linked to a response variable (Sec. 7.4). (**B**) Wavelets distilled by AWD from an LSTM trained to predict molecular partners capture biologically meaningful properties of a large build up in clathrin fluorescence followed by a sharp drop and enable prediction using only a few key coefficients (Chapter 8). (**C**) AWD finds wavelets that are efficient at capturing cosmological information in weak lensing convergence maps and can improve state-of-the-art performance of cosmological parameter inference using an AWD-based simple peak-counting algorithm (Sec. 7.4).

Synthetic data

We begin our evaluation using simulations to verify whether AWD can recover groundtruth wavelets from noisy data. In these simulations, the inputs x_i are generated i.i.d. from a standard Gaussian distribution $\mathcal{N}(0, 1)$. To generate the response variable, the inputs are transformed into the wavelet domain using Daubechies (DB) 5 wavelets [61], and the response is generated from a linear model $y_i = \langle \Psi x_i, \beta \rangle + \epsilon_i$, where the true regression coefficients are 2 for a few selected locations at a particular scale and 0 otherwise; the noise ϵ_i is generated i.i.d. from a Gaussian distribution $\mathcal{N}(0, 0.1^2)$. Then, a 3-layer fully connected neural network with ReLU activations is trained on the pairs of x_i, y_i to accurately predict this response. Note that for any non-singular matrix A, the mapping $x \mapsto \langle A^{-1}\Psi x, A^\top \beta \rangle$ predicts the response equally well, but the representations in the groundtruth wavelet explain the model's

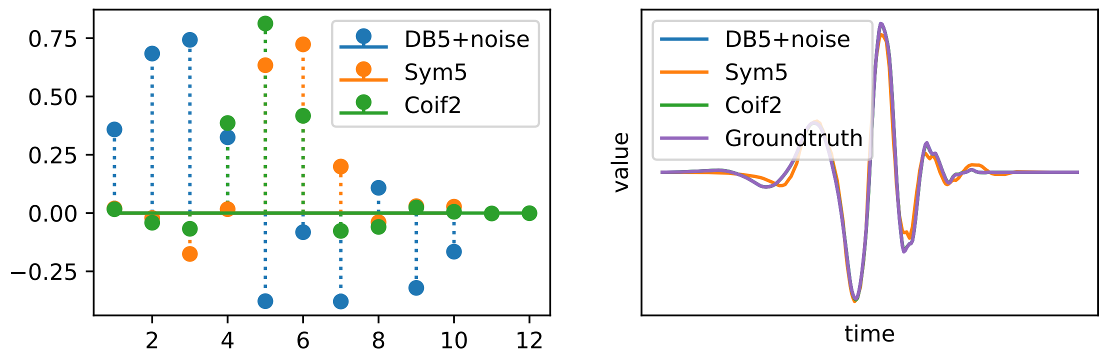


Figure 7.3: AWD accurately identifies the groundtruth important wavelet in simulated data. (**A**) Plots of the initial lowpass filters. (**B**) Final wavelets extracted by AWD.

prediction most concisely. The challenge is then to accurately distill the groundtruth wavelet (DB5) from this DNN. This task is fairly difficult: AWD must not only select which scale and locations are important, it must also precisely match the shape of h and g to the groundtruth wavelet.

Fig. 7.3 shows the performance of AWD on this task. We initialize the AWD lowpass filter to different known lowpass filters corresponding to DB5 (and add noise), Symlet 5, and Coiflet 2 (shown in Fig. 7.3(**A**)) and then optimize the objective given in Eq. (7.7). In order to recover the groundtruth, we select hyperparameters λ and γ that minimize the distance to the groundtruth wavelet $\psi^\star$. Distance is measured via $\mathrm{d}(\psi, \psi^\star) = \min\{\min_k \|\psi^k - \psi^\star\|_2, \min_k \|\widetilde{\psi}^k - \psi^\star\|_2\}$, where ψ^k is the wavelet ψ circular shifted by k and $\widetilde{\psi}$ is the wavelet ψ flipped in the left/right direction. That is, d calculates the minimum ℓ_2 distance between two wavelets under circular shifts and left/right flip. When the two wavelets have different size of support, the shorter wavelet is zero-padded to the length of the longer [216]. Fig. 7.3(**B**) shows that for each different initialization, we find that the distilled wavelet gets considerably closer to the groundtruth wavelet. In particular, the results for DB5+noise and Coiflet 2 are nearly identical to the groundtruth and cannot be distinguished in the plot. This is particularly impressive since the support size of Coiflet 2 differs from that of the groundtruth wavelet, making the task more difficult. Overall, these results demonstrate the ability of AWD to distill key information out of a pre-trained neural network.

Estimating a fundamental parameter surrounding the origin of the universe

We now focus on a cosmology problem, where AWD helps replace DNNs with a more interpretable alternative. Specifically, we consider weak gravitational lensing convergence maps, i.e., maps of the mass distribution in the universe integrated up to a certain distance from the observer. In a cosmological experiment (e.g. a galaxy survey), these mass maps are obtained by measuring the distortion of distant galaxies caused by the deflection of light by the mass between the galaxy and the observer [20]. These maps contain a wealth of physical information of interest, such as the total matter density in the universe, Ω_m. Current cosmology research aims to identify the most informative features in these maps for inferring the cosmological parameters such as Ω_m. The traditional summary statistic for lensing maps is the power spectrum which is known to be sub-optimal for parameter inference. Tighter parameter constraints can be obtained by including higher-order statistics, such as the bispectrum [57] and peak counts [148]. However, DNN-based inference methods claim to improve on constraints based on these traditional summaries [220, 219, 82].

Here, we aim to improve the predictive power of DNN-based methods while gaining interpretability by distilling a predictive AWD model. In this context, it is critically important to obtain interpretability, as it provides deeper understanding into what information is most important to infer Ω_m and can be used to design optimal experiments or analysis methods. Moreover, because these models are trained on numerical simulations (realizations of the Universe with different cosmological parameters), it is important to validate that the model uses reasonable features rather than latching on to numerical artifacts in the simulations. We start by training a model to predict Ω_m from simulated weak gravitational lensing convergence maps. We train a DNN[5] to predict Ω_m from 100,000 mass maps simulated with 10 different sets of cosmological parameter values at the universe origin from the `MassiveNuS` simulations [155] (full simulation details given in Chapter 5), achieving an R^2 value of 0.92 on the test set (10,000 mass maps); Fig. 7.2C shows an example mass map.

We again construct an interpretable model using the wavelets distilled by AWD from the trained DNN. To make predictions, we use the simple peak-counting algorithm developed in a previous work [219], which convolves a peak-finding filter with the input images. Then, these peaks are used to regress on the outcome. In contrast to the fixed filters such as Laplace or Roberts cross used in previous works [219], here we use the wavelets distilled by AWD, which result in three 2D wavelet filters (LL, LH, HL) and the 2D approximation filter (LL). The size of the distilled AWD filters is 12×12 and inspection of these filters shows a majority of the mass is concentrated on 3×3 subfilters (see Fig. 7.2C). Then we extract those subfilters to use for peak-finding filters—by doing so, the size of the filters match with those used in [219] (additional details given in Sec. 5.1). The hyperparameters for AWD are selected by evaluating the predictive model's performance on a held-out validation set.

Table 7.1 shows the results of predicting using the peak-finding algorithm with various filters. The evaluation metric is the RMSE (Root mean square error). Its performance again outperforms the fully trained neural network (Resnet) model and the standard non-adaptive wavelet (DB5) model, as well as other baseline methods using Laplace filter and Roberts cross filter (see Sec. 5.1 for details on how these filters are defined). Moreover, as can be seen in the compression rate, the AWD wavelet provides more efficient representations for the mass maps as well as concise explanation for the DNN's predictions compared to the DB

[5]The model's architecture is Resnet 18 [105], modified to take only one input channel.

5 wavelet.

Table 7.1: Performance comparisons for different models in cosmological parameter prediction. The lower RMSE and compression rate indicate better results. For RMSE, standard deviations are estimated from 10, 000 bootstrap samples.

	AWD (Ours)	Roberts-Cross	Laplace	DB5 Wavelet	Resnet
Regression (RMSE $\times 10^{-2}$)	**1.029 (0.033)**	1.259 (0.039)	1.369 (0.047)	1.569 (0.048)	1.156 (0.024)
Compression rate	**0.610**	N/A	N/A	0.620	N/A

Fig. 7.2C shows the learned AWD filters corresponding to the best distilled wavelet. The learned wavelet filters are symmetric and resemble the "matched filters" which have been used in the past to identify peaks on convergence maps in the cosmology literature [167, 236]. We expect from cosmology knowledge that much information is contained in the peaks of the convergence maps (their amplitude, shape, and numbers), so this indeed matches our expectations based on physics. The high predictive performance further demonstrates that the AWD filters are more efficient at capturing cosmological information and better adapted to the shape of the peaks, than standard wavelets could do.

Moreover, the adaptive wavelet distillation allows us to look at "wavelet activation maps" (see Fig. 5.4) to localize on locations in the convergence maps where important information is concentrated. In other words, we can indeed see that the AWD wavelet concentrates on identifying high intensity peaks, which is where most of the "localized" information is expected from theory.

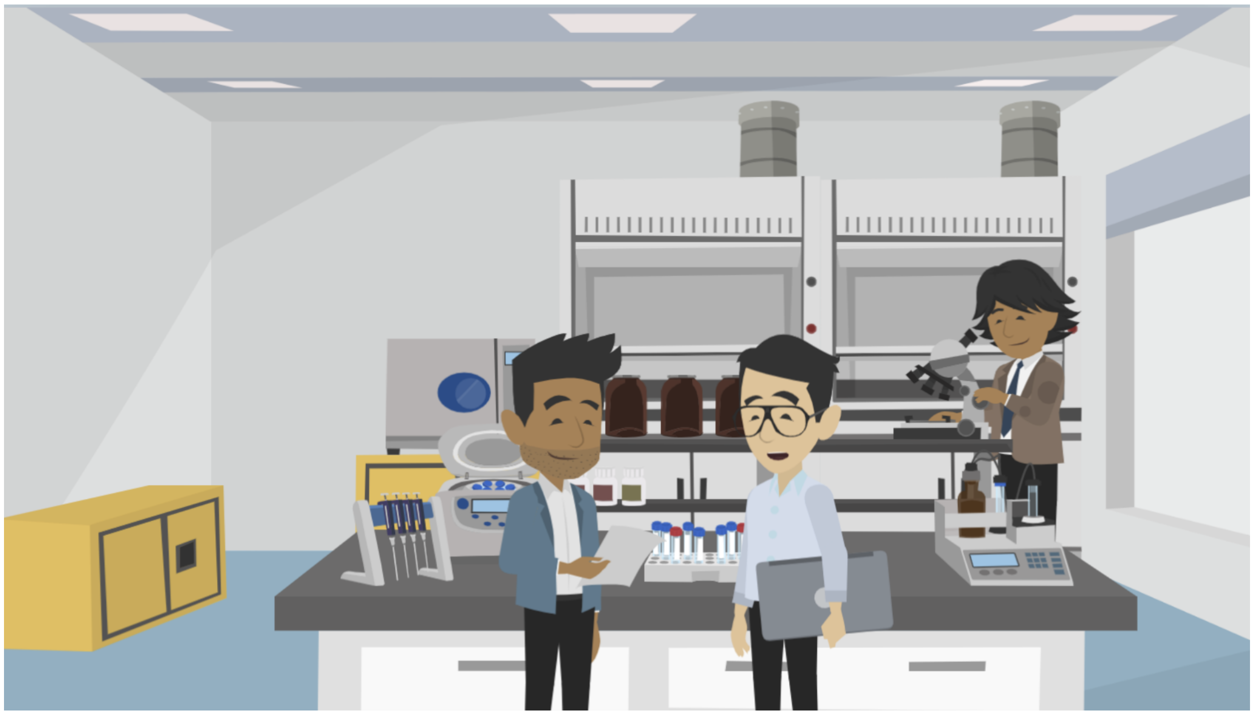
7.5 Discussion

In this work, we introduce AWD, a method to distill adapative wavelets from a pre-trained supervised model such as DNNs for interpretation. Doing so enables AWD to automatically detect and adapt to aspects of data that are important for prediction in an interpretable manner. The benefits of distilling relevant predictive information captured in a DNN are demonstrated through applications to synthetic and real data in two scientific settings. Overall, AWD allows us to interpret a DNN in terms of conventional wavelets, bringing interpretability with domain insights while simultaneously improving compression and computational costs, all while preserving or improving predictive power.

Future work Here, we test our method with the saliency attribution method; many advanced interpretation techniques have been developed in the past years and the comparison between different interpretation techniques must be carefully explored in the context of a particular problem and audience. When optimizing the objective Eq. (7.7) via gradient descent, it requires the gradient of the gradient, which is computationally expensive especially for large data and network sizes. A wavelet-based distillation approach that is computationally more amenable is an interesting direction for future research. The current work learns a single-layer wavelet transform, but the complex nature of modern datasets often require strong nonlinearities. Future work could extend AWD beyond a single-layer wavelet transform, e.g. by borrowing ideas from scattering transform [38] or to other interpretable models [231, 251]. This would allow for bridging closer to deep learning while keeping interpretability, which can be effectively applied to other areas, such as computer vision and natural-image classification. We hope to continue this line of research in order to improve the interpretability and computational efficiency of DNN models across many domains ranging from physical and biomedical sciences to computer vision and information technology.

Chapter 8

Real-world problem: molecular-partner prediction in cell biology



We now turn our attention to a crucial question in cell biology: understanding clathrin-mediated endocytosis (CME) [130, 106]. It is the primary pathway by which things are transported into the cell, making it essential functions of higher eukaryotic life [169]. Many questions about this process remain unanswered, prompting a line of studies aiming to better understand this process [123]. One major challenge with CME analysis is the ability to readily distinguish between abortive coats (ACs) and successful clathrin-coated pits (CCPs). Doing so enables an understanding of what mechanisms allow for successful endocytosis. This is a challenging problem where DNNs have recently been shown to outperform classical statistical and ML methods.

Fig. 8.1 shows the pipeline for this challenging problem. A tracking algorithm is run on videos of cells to identify a time-series trace for each endocytic event [6]. An LSTM model [109] is then trained to classify which endocytic events are successful from the extracted time-series traces and CD scores identify which parts of the traces the model uses. Using these CD scores, domain experts are able to validate that the model does, in fact, use reasonable features such as the max value of the time-series traces and the length of the trace.

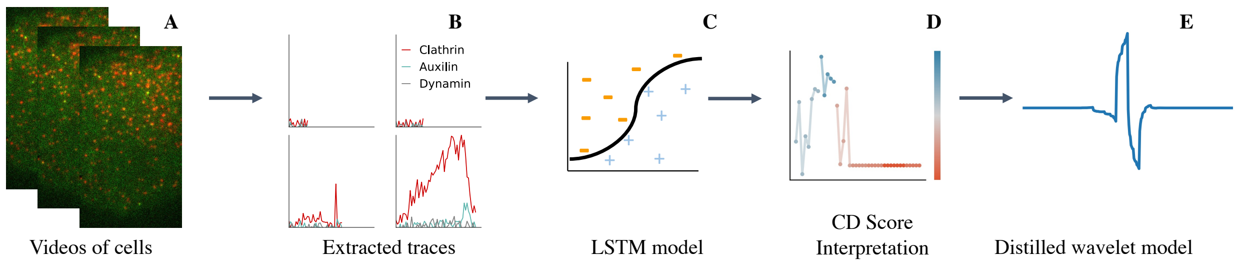


Figure 8.1: Molecular partner prediction pipeline. (**A**) Tracking algorithms run on videos of cells identify (**B**) time-series traces of endocytic events. (**C**) An LSTM model learns to classify which endocytic events are successful and (**D**) CD scores identify which parts of the traces the model uses. (**E**) AWD distills the LSTM model into a simple wavelet model which is able to obtain strong predictive performance.

However, the LSTM model is still relatively difficult to understand and computationally intensive. To create an extremely transparent and concise model, we distill the model into a relatively simple LSTM model. We use AWD to learn an adaptive wavelet and then fit a predictive model on this wavelet by extracting only the maximum 6 wavelet coefficients at each of 5 scales. By taking the maximum coefficients, these features are expected to be invariant to the specific location where a CME event occurs in the input data. This results in a final model with 30 coefficients (6 wavelet coefficients at 5 scales). These wavelet coefficients are used to train a linear model, and the best hyperparameters are selected via cross-validation on the training set.

Fig. 8.2 shows qualitatively how the learned wavelet function ψ changes as a function of the interpretation penalty γ (increasing to the right) and the sparsity penalty λ (increasing downwards). In the initial stage of training, we initialize the lowpass filter to correspond to the Daubechies (DB) 5 wavelet. Different combinations of the penalties lead to vastly different learned wavelets, though they all tend to reveal edge-detecting characteristics for a reasonable range of hyperparameter values.

The red box in Fig. 8.2 highlights the best learned wavelet (for one particular run) extracted by AWD corresponding to the setting of hyperparameters $\lambda = 0.005$ and $\gamma = 0.043$. Table 8.1 compares the results for AWD to the original LSTM and the initialized, non-

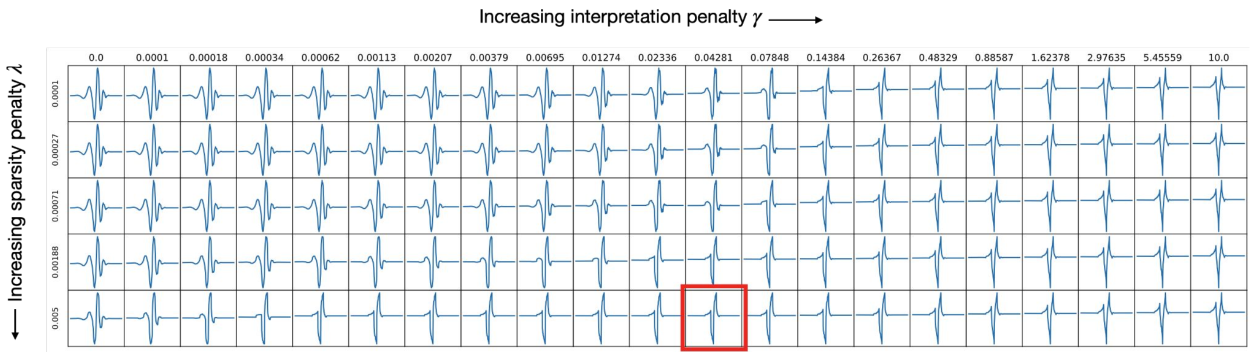


Figure 8.2: Varying sparsity and interpretation penalty yields different valid wavelets. Wavelet highlighted in red is selected by cross-validation and yields the best prediction performance.

adaptive DB5 wavelet model, where the performance is measured via a standard R^2 score, a proportion of variance in the response that is explained by the model. The AWD model not only closes the gap between the standard wavelet model (DB5) and the neural network, it considerably improves the LSTM's performance (a 10% increase in the R^2 score). Moreover, we calculate the compression rates of the AWD wavelet and DB5—these rates measure the proportion of wavelet coefficients in the test set, in which the magnitude and the attributions are both above 10^{-3}. The AWD wavelet exhibits much better compression than DB5 (an 18% reduction), showing the ability of AWD to simultaneously provide sparse representations and explain the LSTM's predictions concisely. The AWD model also dramatically decreases the computation time at test time, a more than 200-fold reduction when compared to LSTM.

In addition to improving prediction accuracy, AWD enables domain experts to vet their experimental pipelines by making them more transparent. By inspecting the learned wavelet, AWD allows for checking what clathrin signatures signal a successful CME event; it indicates that the distilled wavelet aims to identify a large buildup in clathrin fluorescence (corresponding to the building of a clathrin-coated pit) followed by a sharp drop in clathrin fluorescence (corresponding to the rapid deconstruction of the pit). This domain knowledge is extracted from the pre-trained LSTM model by AWD using only the saliency interpretations in the wavelet space.

Table 8.1: Performance comparisons for different models in molecular-partner prediction. AWD substantially improves predictive accuracy, compression rate, and computation time on the test set. A higher R^2 score, and lower compression factor, and lower computation time indicate better results. For AWD, values are averaged over 5 different random seeds.

	AWD (Ours)	Standard Wavelet (DB5)	LSTM
Regression (R^2 score)	**0.262 (0.001)**	0.197	0.237
Compression factor	**0.574 (0.010)**	0.704	N/A
Computation time	**0.0002s**	**0.0002s**	0.0449s

To see the effect of interpretation loss on learning the wavelet transforms and increased performance, we also learn the wavelet transform while setting the interpretation loss to be zero. In this case, the best regression R^2 score selected via cross-validation is 0.231, and the adaptive wavelets without the interpretation loss still outperforms the baseline wavelet but fail to outperform the neural network models.

Diving further into the model's interpretability, Fig. 8.3 shows how the final model makes a single prediction. The different wavelet coefficients are extracted before being linearly combined to make the final prediction.

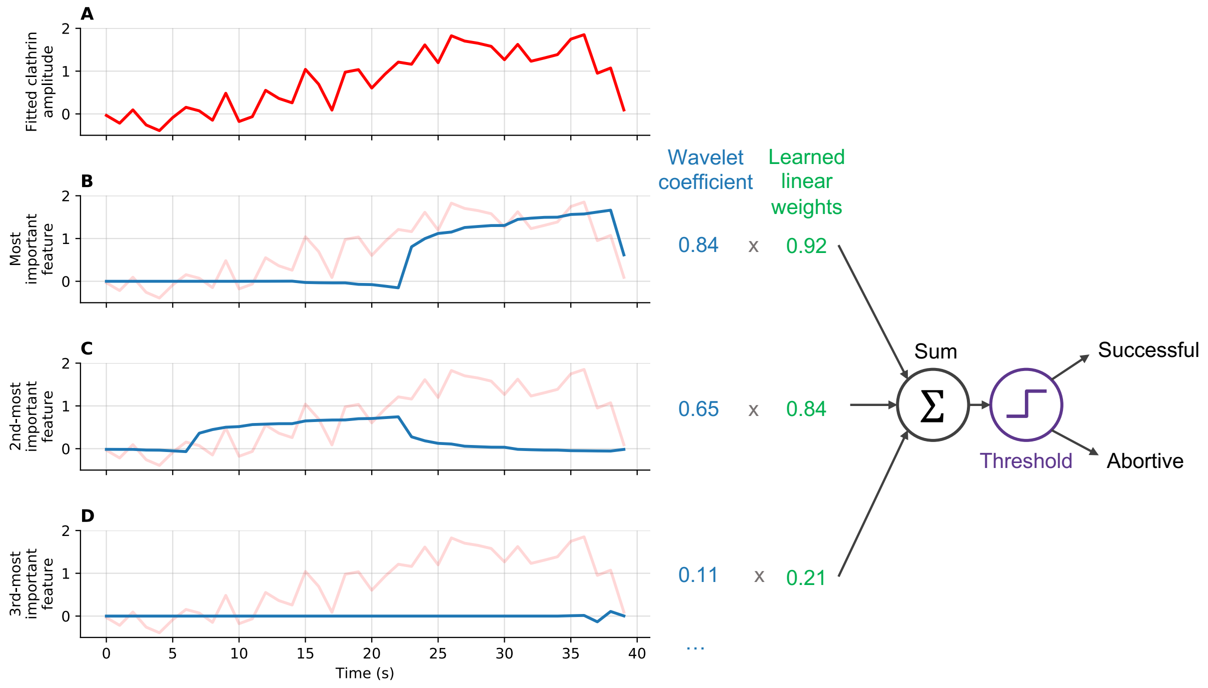
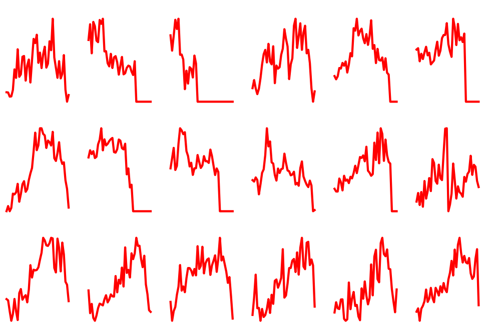


Figure 8.3: Interpreting a single prediction made by the wavelet model. The model takes the fitted clathrin amplitude shown in (**A**) and predicts that the event is successful. (**B, C, D**) show the three most important features for making this prediction. Each blue curve represents the input reconstruction for a single wavelet at a single scale. The curves in (**B**) and (**C**) seem to capture meaningful components of the clathrin signal, as they find a gradual rise in the signal, a large peak in the signal, and finally a steep drop in the signal at the end. The model is simply a linear combination of wavelet coefficients: each blue curve yields a coefficient which is then multiplied by a learned weight. The final prediction of successful or abortive is then made by thresholding the sum of these products. In this case, the first 2 coefficients dominate the prediction, and contributions for all remaining coefficients (some of which are omitted) are considerably less. For abortive predictions, the wavelet coefficients are usually much smaller (or negative).

8.1 Experimental details for molecular partner-prediction

This section gives an overview of the data collected with regards to clathrin-mediated endocytosis in Sec. 8.1. For a more detailed overview, see [106]. The cells are derived from a breast-cancer line and placed on 2D slips with the Shiga toxin. Two fluorescent markers were used: clathrin was tagged with RFP and auxilin with EGFP. The data was collected at a frequency of 1 Hertz.

In order to convert the raw fluorescence images to time-series traces, we use tracking code from previous work [6]. The tracking fits a Gaussian curve to the images (with standard deviation given by the imaging parameters). When the fit to the first channel (i.e. clathrin) is significant,[1] the track is recorded and a fit is forced to the second channel (i.e. auxilin). The amplitudes of each track over time are then extracted. Fig. 8.4 shows some examples of extracted clathrin traces.


(a) Successful events

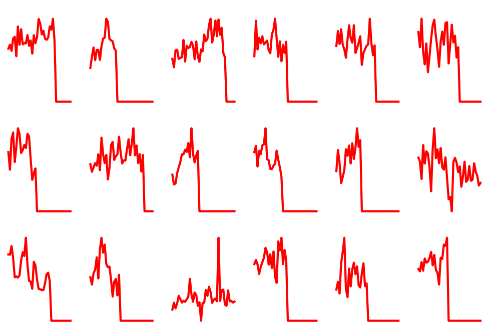

(b) Abortive events

Figure 8.4: Clathrin traces (model inputs) for randomly selected events.

The AWD filters were trained for 100 epochs with Adam optimizer with a learning rate of 0.001. The experiment was run multiple times with respect to the randomness of mini-batches in the training procedure. All experiments were run on an AWS instance of p3.16xlarge for a few days.

Distilled scaling functions and wavelets

Here we show the best wavelets selected by cross-validation and the corresponding scaling functions for 5 different runs of the experiments. The results are stable across multiple runs, all capturing information about how rapid changes in the clathrin trace is useful for predicting the auxilin response.

[1]Here, significant is defined to be p-value less than 0.05, but the results are not sensitive to this precise threshold.

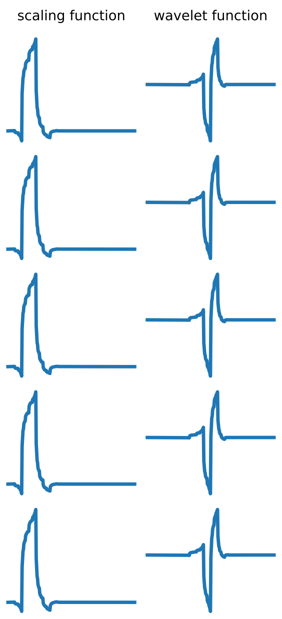


Figure 8.5: Optimal scaling and wavelet functions extracted by AWD across five random seeds.

Varying sparsity and interpretation penalty

Fig. 8.6 shows the learned wavelets distilled by AWD as the interpretation penalty γ and the sparsity penalty λ vary. Unlike Fig. 8.2 where the lowpass filter is initialized to the DB 5 wavelet in the initial stage of training, here the lowpass filter is initialized to that corresponding to the Sym 5 wavelet. For large values of γ, the learned wavelets captures qualitatively the same biological features as those shown in Fig. 8.2.

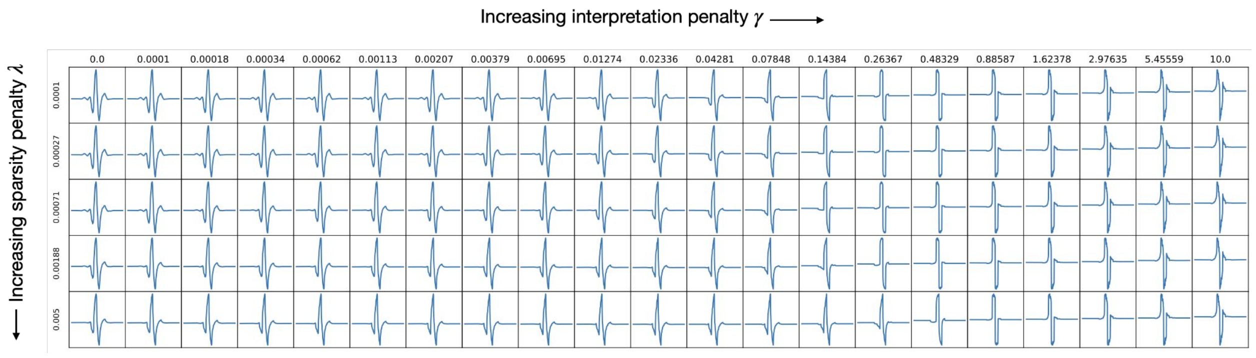


Figure 8.6: Varying sparsity and interpretation penalty yields different valid wavelets. In the initial stage of training, the lowpass filter is initialized to that corresponding to the Symlet 5 wavelet.

Part III

Rule-based models for interpretable modeling

Chapter 9

Fast interpretable greedy-tree sums

9.1 Introduction to FIGS

Modern machine-learning methods such as random forests [33], gradient boosting [87, 49], and deep learning [140] display impressive predictive performance, but are complex and opaque, leading many to call them "black-box" models. This is unfortunate, as model interpretability is critical in many applications [230, 186], particularly in high-stakes settings such as medicine, biology, and policy-making. Interpretability allows models to be audited for general validation, errors, biases, and therefore also more amenable to improvement by domain experts. It facilitates counterfactual reasoning, which is the bedrock of scientific insight, and it instills trust/distrust in a model when warranted. As an added benefit, interpretable models tend to be faster and more computationally efficient than black-box models.

Decision trees are a prime example of interpretable models [32, 87, 209, 231, 251]. They can be easily visualized and simulated even by non-experts, and thus fit naturally into the operating human-in-the-loop AI workflow of many organizations. While they are flexible, and thus have the potential to adapt to complex data, they often tend to be outperformed by black-box models in terms of prediction accuracy. However, this performance gap is not intrinsic to interpretable models, e.g. see examples in [231, 100, 175, 251]. Indeed, in this paper, we will show how this gap can be partially bridged by carefully examining how and why decision trees fall short, and then directly targeting these weaknesses.

Our starting point is the observation that *decision trees can be statistically inefficient at fitting regression functions with additive components* [268]. To illustrate this, consider the following toy example: $y = \mathbb{1}_{X_1>0} + \mathbb{1}_{X_2>0} \cdot \mathbb{1}_{X_3>0}$. [1] The two components of this function can be individually implemented by trees with 1 split and 2 splits respectively. However, implementing their sum with a single tree requires at least 5 splits, as we are forced to combine their tree structures in a fractal manner: a copy of the second tree has to be grown out of every leaf node of the first tree (see Fig. 9.1). Indeed, it is easy to see that a single tree f implementing the sum of independent tree functions $f_1, \dots, f_k$ satisfies

$$\#\text{leaves}(f) \geq \prod_{k=1}^{K} \#\text{leaves}(f_k),$$

and so is much more complicated than simply encoding the function in terms of the original trees in the summation.

This need to grow a deep tree implies two statistical weaknesses of decision trees when fitting them to additive generative models. First, growing a deep tree greatly increases the probability of splitting on noisy features. Second, leaves in a deep tree contain fewer samples, which means that the tree predictions have higher variance. These two weaknesses could be avoided if we could fit a separate decision tree to each additive component of the generative model and present their sum as our model estimate. Existing ensemble methods are unable to disentangle the separate additive components, because they fit each tree either individually (random forests), or sequentially (gradient boosting).

To address these weaknesses, we propose Fast Interpretable Greedy-Tree Sums (FIGS), a novel yet natural algorithm which is able to *grow a flexible number of trees simultaneously.* This procedure is based on a simple modification to Classification and Regression Trees (CART) [32], allowing it to adapt to additive structure if present by starting new trees, while still maintaining the ability of CART to adapt to higher-order interaction terms. Meanwhile, the running time of FIGS remains largely similar to CART due to the similarity of the two

[1]This toy model is an instance of a Local Spiky and Sparse (LSS) model [23], which is potentially grounded in real biological mechanisms whereby an outcome is related to interactions of inputs which display thresholding behavior.

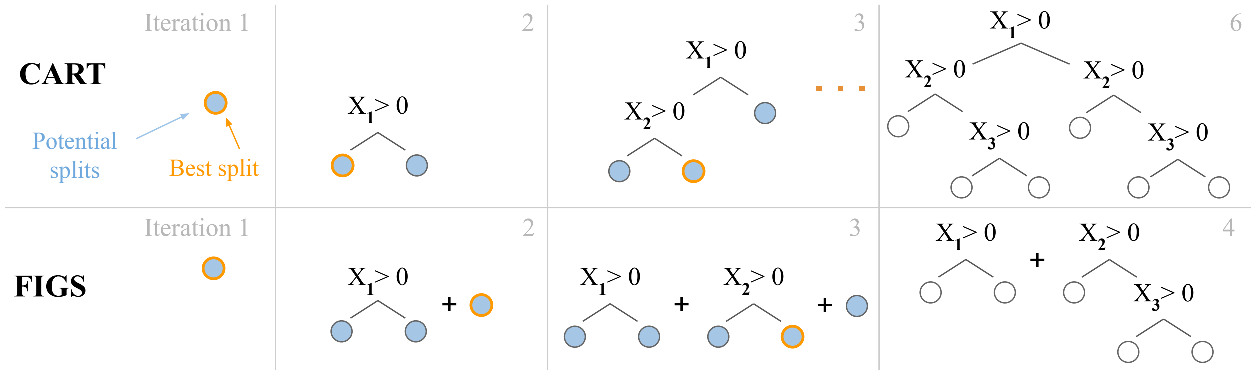


Figure 9.1: FIGS algorithm overview for learning the toy function $y = \mathbb{1}_{X_1>0} + \mathbb{1}_{X_2>0} \cdot \mathbb{1}_{X_3>0}$. FIGS greedily adds one node at a time, considering splits not just in an individual tree but within an ensemble of trees. This can lead to much more compact models, as it avoids repeated splits (e.g. in the final CART model shown in the top-right).

algorithms. FIGS also remains interpretable by keeping the total number of splits in the model limited, allowing for the model to be easily visualized and simulated by hand.

While CART cannot achieve the minimax rate for fitting (generalized) additive generative models with C^1 component functions even with oracle access to the optimal tree structure [268], we show that FIGS can do so under this setting (Theorem 1). In a population setting, we also show that FIGS is able to disentangle separate additive components (Theorem 2) without any constraints on the component functions. We verify both theorems in finite-sample simulations, showing situations where FIGS even outperforms random forests. Meanwhile, extensive experiments across a wide array of real-world datasets show that FIGS achieves state-of-the-art performance while maintaining a concise, interpretable model (e.g. having less than 20 total splits). In particular, they greatly improve upon the predictive performance of CART, and this improvement hints at the presence of approximate additive structure in many of these datasets.

In what follows, Sec. 9.2 introduces FIGS, Sec. 9.3 covers related work, Sec. 9.4 establishes two theoretical results underpinning its performance, Sec. 9.5 shows simulations supporting these two results, and Sec. 9.6 shows extensive experiments suggesting FIGS predicts well with very few splits on real-world datasets.

9.2 FIGS: Algorithm description and runtime

FIGS proposes a natural but powerful extension to CART which forms a sum of trees rather than a single tree. The total number of splits in the model is restricted by a threshold (chosen either by a user or cross-validation). Given this threshold, the greedy algorithm flexibly determines how to allocate these splits among a variable number of trees.

Formally, suppose we are given training data $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. When growing a tree, CART chooses for each node $\mathfrak{t}$ the split s that maximizes the (weighted) impurity decrease in the responses $\mathbf{y}$. This has the formula

$$\hat{\Delta}(s, \mathfrak{t}, \mathbf{y}) := \sum_{\mathbf{x}_i \in \mathfrak{t}} (y_i - \bar{y}_{\mathfrak{t}})^2 - \sum_{\mathbf{x}_i \in \mathfrak{t}_L} (y_i - \bar{y}_{\mathfrak{t}_L})^2 - \sum_{\mathbf{x}_i \in \mathfrak{t}_R} (y_i - \bar{y}_{\mathfrak{t}_R})^2,$$

where $\mathfrak{t}_L$ and $\mathfrak{t}_R$ denote the left and right child nodes of $\mathfrak{t}$ respectively. We call such a split s a *potential split*, and note that for each step in the algorithm, CART actualizes the potential split with the largest impurity decrease value.

FIGS extends CART to greedily grow a small *tree-sum* (see Algorithm 2). That is, at each iteration of FIGS, the algorithm chooses either to make a split on one of the current K trees $\hat{f}_1, \ldots, \hat{f}_K$ in the sum, or to add a new stump to the sum. To make this decision, it still applies the CART splitting rule detailed above to identify potential splits, but instead of using the original response vector, it makes use of the leave-$\hat{f}_k$-out residual vector $r_i^{(-k)} = y_i - \sum_{l \neq k} \hat{f}_l(\mathbf{x}_i)$ to compute the impurity decrease for each tree $\hat{f}_k$. FIGS makes only one split among the $K + 1$ potential splits: The one corresponding to the largest impurity decrease. The prediction over each of the new leaf nodes is defined to be the mean of the $r_i^{(-k)}$ values for samples it contains. At inference time, the prediction is made by summing the predictions of each tree.

Algorithm 2 FIGS fitting algorithm.

```
1:  **FIGS**($X$: features, $y$: outcomes, $stopping\_threshold$)
2:  $trees$ = []
3:  **while** count_total_splits($trees$) < $stopping\_threshold$: :
4:      $all\_trees$ = join($trees$, build_new_tree()) # add new tree
5:      $potential\_splits$ = []
6:      **for** $tree$ in $all\_trees$:
7:          $y\_residuals$ = $y$ − predict($all\_trees$ except $tree$)
8:          **for** $leaf$ in $tree$:
9:              $potential\_split$ = split($X$, $y\_residuals$, $leaf$)
10:             $potential\_splits$.append($potential\_split$)
11:     $best\_split$ = split_with_min_impurity($potential\_splits$)
12:     $trees$.insert($best\_split$)
```

FIGS is related to backfitting [35], but differs from it in important ways: FIGS neither assumes a fixed number of component predictors, nor updates them in a cyclic manner; in fact, FIGS coordinates a competition among the trees being fitted at each iteration, thus mitigating backfitting's potential to overfit to residuals.

Due to its similarity to CART, FIGS supports many natural modifications that are used in CART trees. For example, different impurity measures can be used; here we use Gini impurity for classification and mean-squared-error for regression. Additionally, FIGS could benefit from pruning or by being used as part of an ensemble model.

Run-time analysis The run-time complexity for FIGS to grow a model with m splits in total is $O(dm^2n^2)$, where d the number of features, and n the number of samples. In contrast, CART has a run-time of $O(dmn^2)$. Both of these worst-case run-times given above are quite fast, and the gap between them is relatively benign as we usually make a small number of splits for the sake of interpretability.

Selecting the model's stopping threshold. Choosing a threshold on the total number of splits can be done similar to CART: using a combination of the model's predictive performance and domain knowledge on how interpretable the model needs to be. Alternatively, the threshold can be selected using an impurity decrease threshold [32] rather than a hard threshold on the number of splits. We discuss potential data-driven choices of the threshold in the Discussion (Sec. 9.7).

9.3 Background on tree-sums

There is a long history of greedy methods for learning individual trees, e.g. C4.5 [209], CART [32], and ID3 [210]. Recent work has proposed global optimization problems rather than greedy algorithms for trees, which can incur a high computational cost but improve performance given a fixed rule budget [150, 115, 26]. However, due to the limitations of a single tree, all these methods suffer from the problem of having repeated splits or repeated subtrees [200], a failure we will quantify in the results section.

Besides trees, there are a variety of other interpretable methods such as rule lists [143, 12] or rule sets [53, 62]; for an overview and python implementation, see [251].

Also similar to the work here are methods that learn an additive model of rules, where a rule is defined to be an axis-aligned, rectangular region in the input space. RuleFit [86] is a popular method that learns a model by first extracting rules from multiple greedy decision trees fit to the data and then learning a linear model using those rules as features. FIGS is able to improve upon RuleFit by greedily selecting higher-order interactions when needed, rather than simply using all rules from some pre-specified tree depth. MARS [88] greedily learns an additive model of splines in a manner similar to FIGS, but loses interpretability as a result of using splines rather than rules.

Loosely related to this work are additive models of trees, such as Random Forest [33], gradient-boosted trees [85], BART [51] and AddTree [157], which use tree ensembles as a way to boost predictive accuracy without focusing on finding an interpretable model. Also loosely related are posthoc methods which aim to help understand a black-box model, but ultimately cannot be as interpretable as an individual interpretable model [160, 87, 64].

9.4 Theoretical evidence that FIGS adapts to additive structure

Tight generalization upper bounds have proved elusive for CART due to the complexity of analyzing the tree growing procedure, and are difficult for FIGS for the same reason. However, even if we knew the optimal tree structure for CART, having to use empirical averages instead of population means for the prediction over each leaf leads to an ℓ_2 generalization lower bound of $\Omega(n^{-2/(d+2)})$ when the data is generated from an additive model with C^1 component functions, which is much worse than the minimax rate of $\tilde{O}\big(dn^{-2/3}\big)$ for this problem [268]. In comparison, assuming that we know the optimal tree structures, but not the optimal tree predictions, we are able to derive a much faster rate for models comprising sums of trees.

To formalize our theorem, consider a collection of trees $\mathfrak{C} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_L\}$, We define a *tree-sum model* on $\mathfrak{C}$ to be a function $\tilde{f}$ that is a sum of component functions $\tilde{f}_1, \ldots, \tilde{f}_L$, with $\tilde{f}_l$ implementable by $\mathcal{T}_l$, for $l = 1, \ldots, L$. Now suppose we are given training data $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$. Define a tree-sum model on $\mathfrak{C}$ to be *best-fit* with respect to $\mathcal{D}_n$ if it is an empirical ℓ_2 risk minimizer in this class of models.For each query point $\mathbf{x}$, this must satisfy the best-fit property that

$$\tilde{f}_l(\mathbf{x}) = \frac{1}{N(\mathfrak{t}_l(\mathbf{x}))} \sum_{\mathbf{x}_i \in \mathfrak{t}_l(\mathbf{x})} \left(y_i - \sum_{k \neq l} \tilde{f}_k(\mathbf{x}_i) \right), \tag{9.1}$$

where $\mathfrak{t}_l(\mathbf{x})$ is the leaf in $\mathcal{T}_l$ containing $\mathbf{x}$, and that this property is satisfied approximately by FIGS because of the update formula.

Our generative model: Let $\mathbf{x}$ be a random variable with distribution π on $[0,1]^d$. Suppose that we have independent blocks of features $I_1, \ldots, I_K$, of sizes $d_1, \ldots, d_K$. For each k, let $P_k \colon [0,1]^d \to [0,1]^{I_k}$ denote the projection onto the coordinates in I_k. Let $y = f(\mathbf{x}) + \epsilon$ where $\mathbb{E}\{\epsilon \mid \mathbf{x}\} = 0$ and

$$f(\mathbf{x}) = \sum_{k=1}^{K} f_k(P_k(\mathbf{x})) + f_0. \tag{9.2}$$

Theorem 1 (Generalization upper bounds using oracle tree structure). *Given the generative model described above, further suppose the distribution π_k of each independent block $\mathbf{x}_{I_k}$ has a continuous density, each f_k in (9.2) is C^1, with $\|\nabla f_k\|_2 \leq \beta_k$, and that ϵ is homoskedastic with variance $\mathbb{E}\{\epsilon^2 \mid \mathbf{x}\} = \sigma^2$. Then there exists an oracle collection of K trees $\mathfrak{C} = \{\mathcal{T}_1, \ldots, \mathcal{T}_K\}$, with $\mathcal{T}_k$ splitting only on features in I_k for each k, and a best-fit tree-sum model on $\mathfrak{C}$ with respect to $\mathcal{D}_n$, $\tilde{f} = \sum_{k=1}^K \tilde{f}_k$, for which we have the following ℓ_2 upper bound on the complement of a vanishing event $\mathcal{E}$:*

$$\mathbb{E}_{\mathcal{D}_n, \mathbf{x}} \Big\{ (\tilde{f}(\mathbf{x}) - f(\mathbf{x}))^2 \mathbf{1}\{\mathcal{E}^c\} \Big\} \leq \sum_{k=1}^{K} c_k \left(\frac{\sigma^2}{n} \right)^{\frac{2}{d_k+2}}. \tag{9.3}$$

Here, $c_k := 8(d_k\beta_k^2\|\pi_k\|_\infty)^{\frac{d_k}{d_k+2}}$, while $\mathbb{P}\{\mathcal{E}\} = O(n^{-2/(d_{max}+2)})$ where $d_{max} = \max_k d_k$ is the size of the largest feature block in (9.2)*.*

It is instructive to consider two extreme cases: If $d_k = 1$ for each k, then we have an upper bound of $O\big(dn^{-2/3}\big)$. If on the other hand $K = 1$, we have an upper bound of $O\big(n^{-2/(d+2)}\big)$. Both (partially oracle) bounds match the well-known minimax rates for their respective inference problems [212], hinting that FIGS might be able adapt to both additive structure as well as higher-order interactions. We also believe that (9.3) is the minimax rate in general for any block structure.

We note that the error event $\mathcal{E}$ is due to the query point possibly landing in leaf nodes containing very few or even zero training samples, which can be thus be detected and avoided in practice by imputing a default value.

The proof builds on recent work [131] which shows how to interpret CART as a "local orthogonal greedy procedure": Growing a CART tree corresponds to greedily adding to a set of engineered linear predictors. This interpretation has a natural extension to FIGS, but at the cost of orthogonality.

Our next result shows that FIGS is able to disentangle the different additive components of f into distinct trees as intended, if the algorithm is run in the large sample limit.

Theorem 2 (**Oracle disentanglement**)**.** *Suppose we run Algorithm 2 with the following oracle modifications:*

1. *Split impurities are defined via:*

$$\begin{aligned}\Delta(s, \mathfrak{t}, r) := \pi(\mathfrak{t})\mathrm{Var}\{r \mid \mathbf{x} \in \mathfrak{t}\} \\ - \pi(\mathfrak{t}_L)\mathrm{Var}\{r \mid \mathbf{x} \in \mathfrak{t}_L\} - \pi(\mathfrak{t}_R)\mathrm{Var}\{r \mid \mathbf{x} \in \mathfrak{t}_R\}\end{aligned} \tag{9.4}$$

2. *The prediction over each new leaf node is defined to be the population mean of the residual function $r^{(-k)}$ over the leaf.*

At any number of iterations, let $\hat{f} = \sum_{k=1}^{K'} \hat{f}_k$ denote the working model. Then for each tree $\hat{f}_k$, the set of features split upon is contained within a single index set I_k for some k.

The number of terms K' in the fitted model need not be equal to K. Note that the two modifications are equivalent to running FIGS in the large sample limit, as for any function $h(\mathbf{x}, y)$, we have $n^{-1}\hat{\Delta}(s, \mathfrak{t}, h) \to \Delta(s, \mathfrak{t}, \mathbf{h})$ and $\bar{h}_{\mathfrak{t}} \to \mathbb{E}\{h \mid \mathfrak{t}\}$ as $n \to \infty$.

9.5 Simulations support theoretical results

Sec. 9.5 shows simulations supporting Theorem 1 and Sec. 9.5 shows simulations supporting Theorem 2.

FIGS achieves fast rates for ℓ_2 generalization error for additive models

Fig. 9.2 investigates the ℓ_2 generalization error for FIGS as a function of the number of training samples used. As predicted by Theorem 1, FIGS error decreases at a faster rate than that of either CART or Random Forest (RF). We simulated data via a sparse sum of squares model $y = \sum_{j=1}^{20} x_j^2 + \epsilon$ with $\mathbf{x} \sim \text{Unif}([0,1]^{50})$, and $\epsilon \sim N(0, 0.01)$.

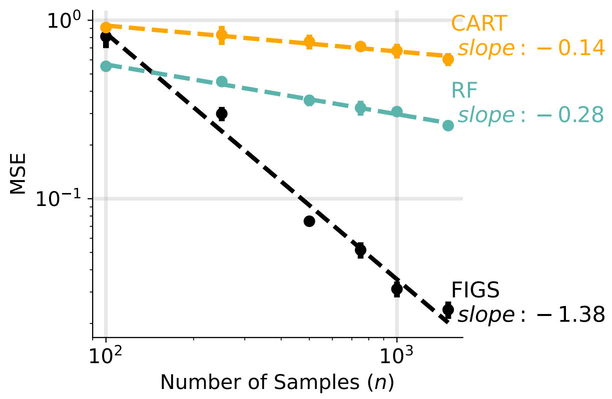


Figure 9.2: FIGS test error rate for additive data decreases faster than CART and random forest (RF), as predicted by Theorem 1. Averaged over 4 runs (errors bars are standard error of the mean and are often within the points).

FIGS disentangles additive components of additive models

To investigate disentanglement (Theorem 2), we add interactions into our generating model, and set

$$y = \sum_{i=0}^{4} x_{3i+1} x_{3i+2} x_{3i+3} + \epsilon,$$

while keeping the other parameters as before and using 2,500 training samples to fit FIGS. When training FIGS on this data, we hope that each tree learned by the algorithm will contain splits only from a single interaction. Fig. 9.3 shows that this is largely what happens. Let T_l be the number of trees learned by FIGS on dataset l, and set $T = \sum_{l=1}^{10} T_l$. Given the collection of all trees a single index, we construct the T by 15 matrix M, whose (i, j)-th entry is the number of splits in tree i on feature j. We then compute the pairwise cosine similarities between the columns of M, displaying the results in Fig. 9.3. Note that pairs of features that never get split upon on in the same tree have a similarity value of 0, while pairs of features that always have the same number of splits in each tree have a value of 1. Fig. 9.3 shows that the empirically observed similarity values are remarkably close to this ideal.

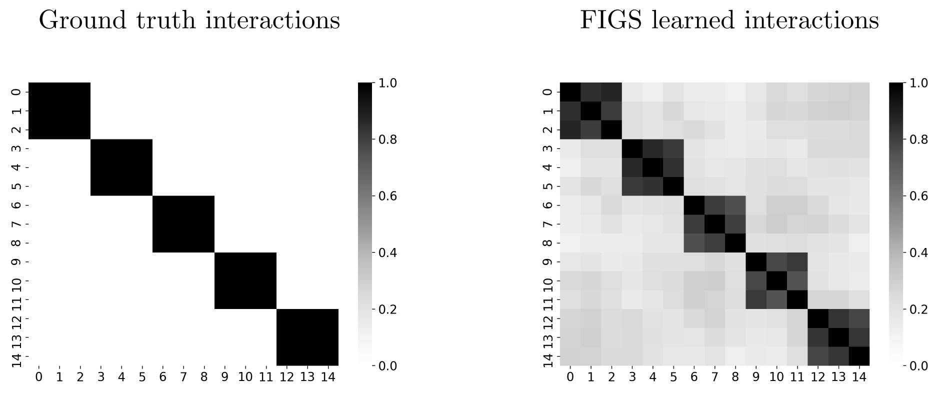


Figure 9.3: FIGS disentangles interactions into different additive components, as predicted by Theorem 2. When fitted to a sum of three-way interactions, FIGS correctly places interacting terms into the same tree (dark blocks).

9.6 FIGS results on real-world datasets

This section gives a brief overview of the datasets analyzed here before Sec. 9.6 shows FIGS's predictive performance and Sec. 9.6 shows its ability to identify additive structures in real-world data.

For classification, we study four large datasets previously used to evaluate rule-based models [280] along with the two largest UCI binary classification datasets used in the classic Random-Forest paper [33, 15] (overview in Table 9.1). For regression, we study all datasets used in the Random-Forest paper with at least 200 samples along with three of the largest non-redundant datasets from the PMLB benchmark [227]. 80% of the data is used for training/3-fold cross-validation and 20% of the data is used for testing.

FIGS predicts well with few splits on real-world datasets

Fig. 9.4 shows the models' performance results (on test data) as a function of the number of splits in the fitted model[2]. For both classification and regression, **FIGS** is compared to CART, RuleFit, and Boosted Stumps (CART stumps learned via gradient-boosting). For classification, we additionally compare against C4.5 and for regression we additionally compare against CART using the mean-absolute-error (MAE) splitting-criterion. We finally also add a Random Forest black-box baseline with 100 trees, which uses many more splits than all the other models. [3]

[2]For RuleFit, each term in the linear model is counted as one split

[3]We also compare against Gradient-boosting with decision trees of depth 2, but find that it is outperformed by CART in this limited-rule regime, so we omit these results for clarity. We also attempt to compare to optimal tree methods, such as GOSDT [150], but find that they are unable to fit the dataset sizes here.

	Name	Samples	Features
Classification	Readmission	101763	150
	Credit [288]	30000	33
	Recidivism	6172	20
	Juvenile [197]	3640	286
	German credit	1000	20
	Diabetes [253]	768	8
Regression	Breast tumor [227]	116640	9
	CA housing [199]	20640	8
	Echo months [227]	17496	9
	Satelite image [227]	6435	36
	Abalone [188]	4177	8
	Diabetes [75]	442	10
	Friedman1 [88]	200	10
	Friedman2 [88]	200	4
	Friedman3 [88]	200	4

Table 9.1: Real-world datasets analyzed here: classification (top panel), regression (bottom panel).

The top two rows of Fig. 9.4 show results for classification (measured using the ROC area under the curve, i.e. AUC), and the bottom three rows show results for regression (measured using R^2. On average, FIGS outperforms baseline models when the number of splits is very low. The performance gain from FIGS over other baselines is larger for the datasets with more samples (e.g. the top row of Fig. 9.4), matching the intuition that FIGS performs better because of its increased flexibility. For two of the large datasets (*Credit* and *Recidivism*), FIGS even outperforms the black-box Random Forest baseline, despite using less than 15 rules. For the smallest classification dataset (*Diabetes*), FIGS performs extremely well with very few (less than 10) rules, but starts to overfit as more rules are added.

FIGS diagnoses possible additive structures in real-world datasets

Fig. 9.5 shows an example comparing individual models learned by FIGS and CART on the Diabetes classification dataset [25, 253]. In this dataset, eight risk factors were collected and used to predict the onset of diabetes within five years. The dataset consists of 768 female subjects from the Pima Native American population near Phoenix, AZ, USA 268 of the subjects developed diabetes, which is treated as a binary label.

Fig. 9.5 shows two models, one learned by FIGS and one learned by CART. In both models, a higher prediction corresponds to a higher risk of developing diabetes. Both achieve roughly the same performance (FIGS yields an AUC of 0.820 whereas CART yields an AUC of 0.817), but the models have some key differences. The FIGS model includes fewer features and fewer total rules than the CART model, making it easier to understand in its entirety.

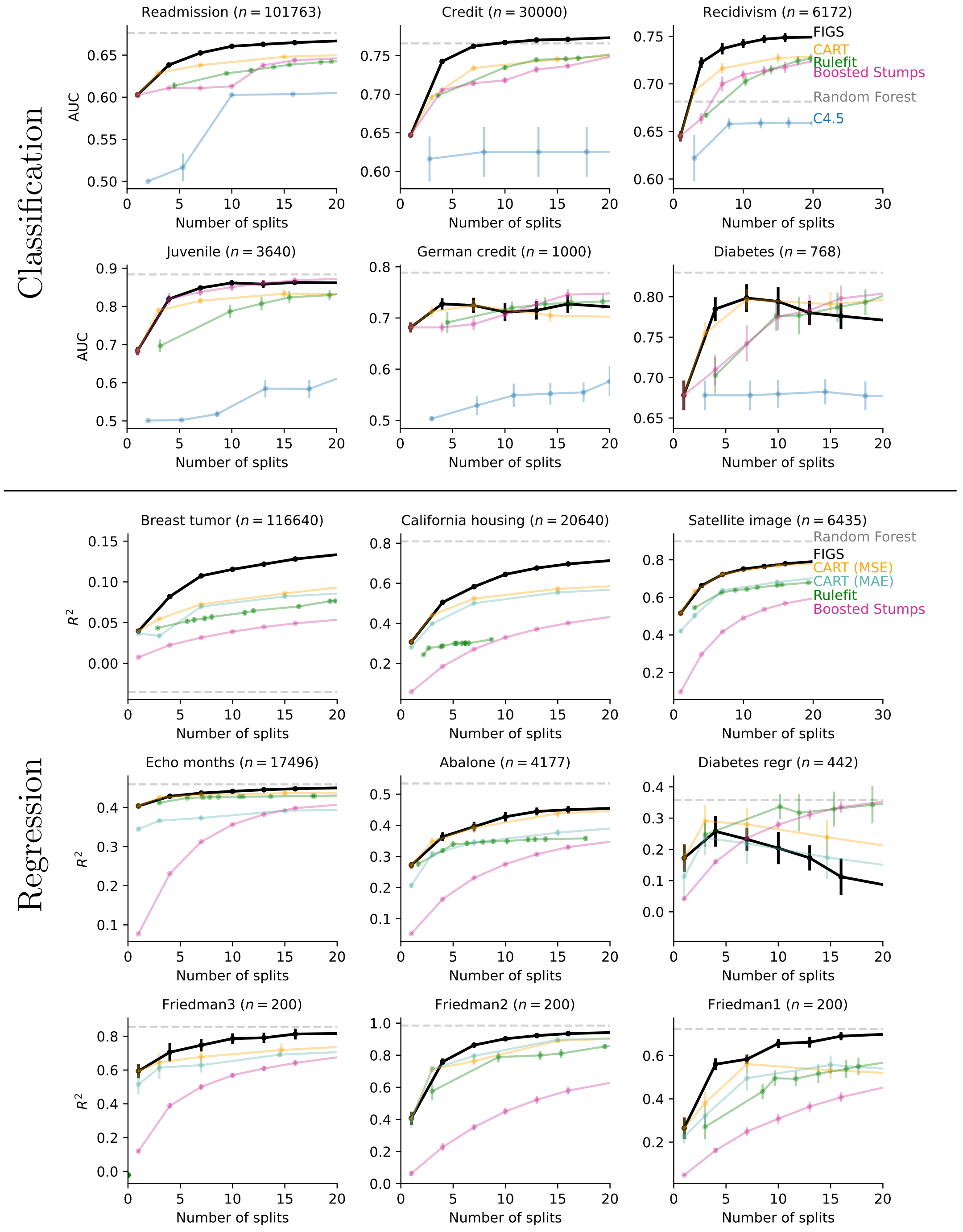


Figure 9.4: FIGS performs extremely well using very few splits, particularly when the dataset is large. Top two rows show results for classification datasets (measured by AUC of the ROC curve) and the bottom three rows show results for regression datasets (measured by R^2). Errors bars show standard error of the mean, computed over 6 random data splits.

Moreover, the FIGS model completely decouples interactions between features, making it clear that each of the features contributes independently of one another, something which any single-tree model is unable to do.

The FIGS model makes its prediction by summing the contribution for the leaf-node of each tree in the model (where some trees consist of only one split). For example, if a subject's plasma glucose is greater than 166, their BMI (body-mass index) is greater than 29, and their age is less than 29, then their final risk score is $0.55 + 0.26 + 0 = 0.81$. To make this prediction, the CART model must instead use an interaction between plasma glucose and BMI.

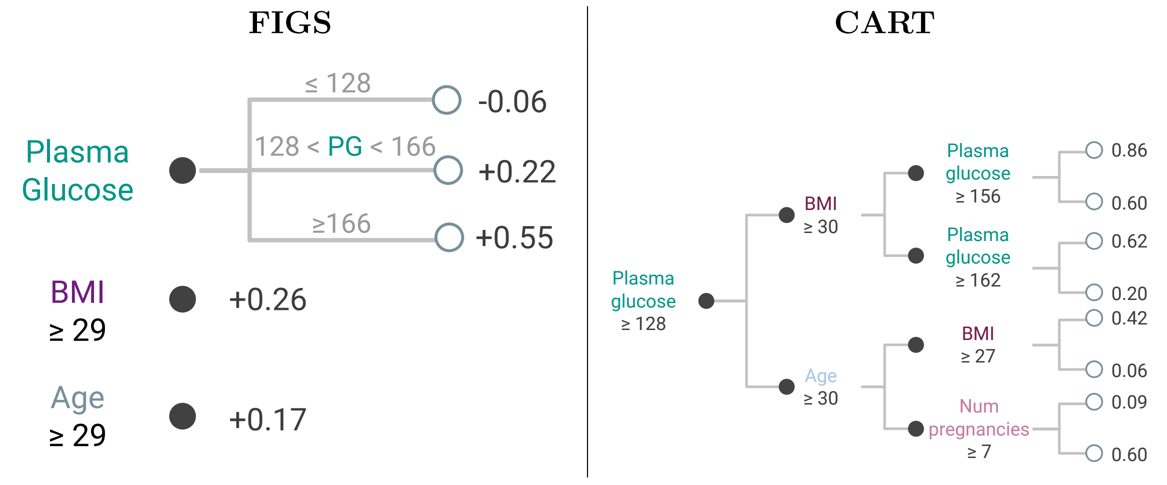


Figure 9.5: Comparison between FIGS and CART on the diabetes dataset. FIGS learns a simpler model, which disentangles interactions between features. Both models achieve the same generalization performance (FIGS yields an AUC of 0.820 whereas CART yields 0.817.)

Next, Fig. 9.6 investigates whether FIGS avoids the issue of repeated rules. It shows the fraction of rules which are repeated within a learned model as a function of the total number of rules in the model. We define a rule to be repeated if the model contains another rule using the same feature and a threshold whose value is within 0.01 of the original rule's threshold.[4] FIGS consistently learns fewer repeated rules than CART, one signal that it is avoiding learning redundant subtrees by separately modeling additive components. For clarity, Fig. 9.6 shows only the largest three datasets studied here, but other datasets demonstrate the same relationship.

[4]This result is stable to reasonable variation in the choice of this threshold.

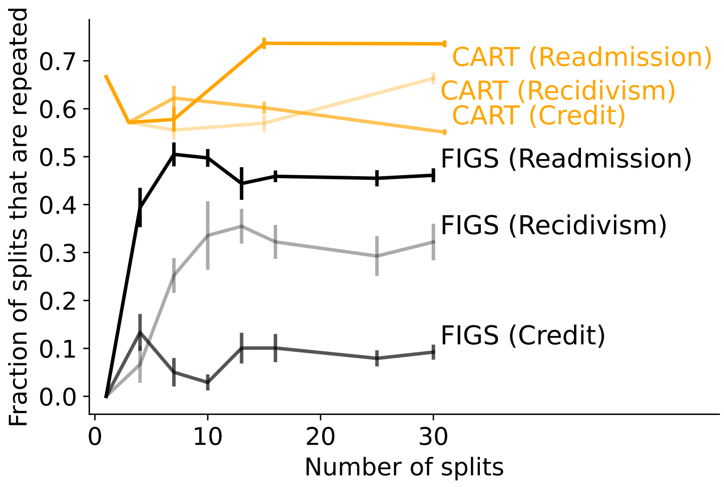


Figure 9.6: FIGS learns less redundant models than CART. As a function of the number of rules in the learned model, we plot the fraction of rules, repeated for three different datasets. Error bars show standard error of the mean, computed over 6 random splits.

9.7 FIGS Discussion

FIGS is a powerful and natural extension to CART which achieves improved predictive performance over popular baseline tree-based methods across a wide array of datasets while maintaining interpretability by using very few splits.

FIGS has many natural extensions. It is a greedy algorithm, but could be extended by using a global optimization algorithm over the class of tree-sum models. Alternatively, a FIGS model could be distilled into a simpler model (e.g. a single tree or rule-list). Additionally, the class of FIGS models could be further extended to include linear terms or allow for summations of trees to be present at split nodes, rather than just at the root. Future work could also explore using FIGS (or a randomized version of FIGS), for interaction detection, building off of Theorem 2 and Fig. 9.3.

In this work, we vary the total number of splits in the model and analyze the performance. As mentioned earlier, this regularization parameter in FIGS can be tuned as done in CART. In some situations, a data-driven choice of threshold may be desirable. As seen in Sec. 9.6, using cross-validation (CV) to select the threshold almost always leads to the largest allowed value for the total number of splits for the datasets and parameter ranges that we considered. This is not surprising as CV doesn't consider stability or interpretability when selecting a model. Future work can use criteria related to BIC [239] or stability in combination with CV [149] for selecting this threshold based on data. In future work, one could also vary the total number of splits and number of trees separately, helping to build prior knowledge into the fitting process.

FIGS as proposed has some potential limitations. It is more flexible than CART, and as such could potentially overfit to small data faster than CART. To mitigate overfitting, FIGS's flexibility could be penalized via novel regularization techniques, such as regularizing individual leaves or regularizing a linear model formed from the rules extracted by FIGS. Alternatively, FIGS might be distilled into an even simpler rule-based model to impose more regularization. We note however, that the potential for overfitting does not materialize in our experiments (e.g. Fig. 9.4), perhaps since starting a new tree helps combat the problem of estimating the mean value of a leaf node with very few points. We hope FIGS can pave the way towards more transparent and interpretable modeling that can improve machine-learning practice going forward, particularly in high-stakes domains such as medicine and policy making.

Chapter 10

Hierarchical shrinkage for trees

10.1 Introduction to HS

Decision tree models, used for supervised learning since the 1960s [182, 172, 210], have recently attained renewed prominence because they embody key elements of interpretability: shallow trees are easily described and visualized, and can even be implemented by hand. While the precise definition and utility of interpretability have been a subject of much debate [186, 69, 230, 231], all agree that it is an important notion in high-stakes decision-making, such as medical-risk assessment and criminal justice. For this reason, decision trees have been widely applied in both areas [256, 137, 143, 12].

By far the most popular decision tree algorithm is Classification and Regression Trees (CART) [32]. These can be ensembled to form a Random Forest (RF) [33] or used as

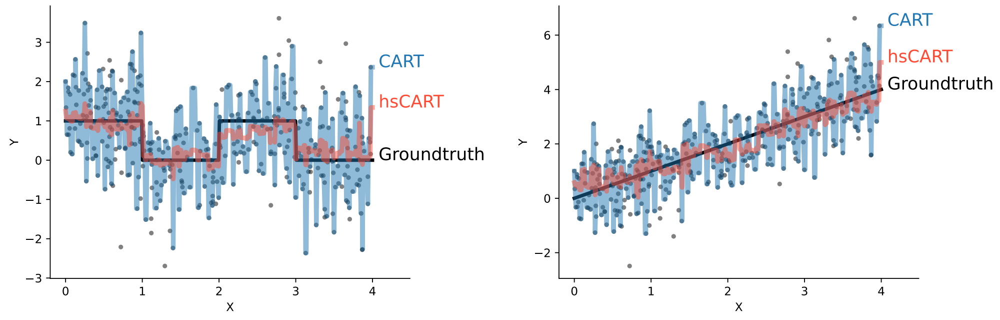


Figure 10.1: Example of HS for toy univariate regression problems. HS regularizes model predictions to improve estimates in noisy leaves that have few samples. CART is fit to the data in the blue dots and then HS is applied posthoc (hsCART).

weak learners in Gradient Boosting (GB) [87]; both algorithms have achieved state-of-the-art performance over a wide class of prediction problems [45, 44, 80, 196, 113], and are implemented in popular machine learning packages such as `ranger` [286], `scikit-learn` [205], and `imodels` [251]. Variants of these algorithms, such as iterative random forest for finding stable interactions [22], have found use in scientific applications.

In view of the widespread use of tree-based methods, we seek to provide a new lens on their regularization. On the one hand, decision trees often obey traditional statistical wisdom in that they need to be regularized to prevent overfitting. In practice, this is carried out by specifying an early stopping condition for tree growth, such as a maximum depth, or alternatively, pruning the tree after it is grown [89]. These procedures, however, only regularize tree models via their *tree structure*, and it is usually taken for granted that the prediction over each leaf should be the average response of the training samples it contains. We show that this can be very limiting: *shrinking these predictions in a hierarchical fashion can significantly reduce generalization error in both regression and classification settings* (e.g. see Fig. 10.1).

On the other hand, trees used in an RF are usually not explicitly regularized and interpolate the data by being grown to purity (e.g. see the default settings of `scikit-learn` and `ranger`). Instead, RF prevents overfitting by relying upon the randomness injected into the algorithm during tree growth, which acts as a form of implicit regularization [33, 171]. We show that apart from this implicit regularization, more regularization, in the form of hierarchical shrinkage, does improve generalization even while using a smaller ensemble for many data sets.

Equally important, regularizing RFs also improves the quality of their post-hoc interpretations. RFs are usually interpreted via their feature and interaction importances, which have been used to provide scientific insight in areas such as remote sensing and genomics [50, 22, 78]. The reproducibility and scientific meaning of such interpretations become questionable when the underlying RF model has poor predictive performance [186], or when they are highly sensitive to data perturbations [292]. We show that HS improves the interpretability of RF by both simplifying and stabilizing its decision boundaries and SHAP values [159] on a number of real-world data sets.

Our proposed method, which we call *Hierarchical Shrinkage* (HS), is an extremely fast and simple yet effective algorithm for the *post-hoc* regularization of *any* tree-based model. It does not alter the tree structure, and instead replaces the average response (or prediction) over a leaf in the tree with a *weighted average of the mean or average responses over the leaf and each of its ancestors*. The weights depend on the number of samples in each leaf, and are controlled by a single regularization parameter λ that can be tuned efficiently via generalized cross validation. HS is agnostic to the way the tree is constructed and can be applied post-hoc to trees constructed with greedy methods such as CART and C4.5 [209], as well optimal decision trees grown via dynamic programming or integer optimization techniques [150].

A more naive form of shrinkage, which we call *leaf-based shrinkage* (LBS), appears as part of XGBoost [49]: whenever a new tree is grown, the average response (prediction) over each leaf is shrunk directly towards the sample mean of the responses. LBS also occurs[1] in Bayesian Additive Regression Trees (BART) [51], which grows an ensemble of trees via a backfitting MCMC algorithm. Comparing LBS to HS on several real-world datasets shows that HS uniformly has better predictive performance than LBS.

We explain the advantages of HS by building on recent work which uses decision stumps associated to each interior node of a tree to construct a new (supervised) feature representation [131]. The original tree model is recovered as the linear model obtained by regressing the responses on the supervised features. We show that HS is exactly the *ridge regression* solution in this supervised feature space, while LBS can also be viewed as ridge regression, but with a different (supervised) feature space (of the same dimension) that relies only on the leaf nodes. This allows us to use ridge regression calculations heuristically to partially explain both the reasonableness of the shrinkage scaling in HS, as well as our empirical evidence that HS achieves consistently better predictive accuracy than LBS (see Sec. 10.5).

The rest of the paper is organized as follows. Sec. 10.2 gives a formal statement of HS, and discusses several computational considerations. Sec. 10.5 discuss the interpretation of HS as ridge regression on the supervised features. Sec. 10.3 presents the results of extensive numerical experiments on simulated and real world data sets that illustrate the gains in prediction accuracy from applying the method. Sec. 10.4 shows how HS improves the interpretability of RFs.

10.2 The Hierarchical Shrinkage (HS) algorithm

Throughout this paper, we work in the supervised learning setting where we are given a training set $\mathcal{D}_n = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, from which we learn a tree model $\hat{f}$ for the regression function.

[1]When conditioned on the structure of a given tree, as well as all other trees in the ensemble, the posterior distribution for the contribution of a leaf node is a product of Gaussian likelihood functions centered at the model residuals as well as a Gaussian prior. A simple calculation shows that the posterior mean can be obtained from the residual mean via LBS.

Given a query point $\mathbf{x}$, let $\mathfrak{t}_L \subset \mathfrak{t}_{L-1} \subset \cdots \subset \mathfrak{t}_0$ denote its leaf-to-root path, with $\mathfrak{t}_L$ and $\mathfrak{t}_0$ representing its leaf node and the root node respectively. For any node $\mathfrak{t}$, let $N(\mathfrak{t})$ denote the number of samples it contains, and $\hat{\mathbb{E}}_{\mathfrak{t}}\{y\}$ the average response. The tree model prediction can be written as the telescoping sum:

$$\hat{f}(\mathbf{x}) = \hat{\mathbb{E}}_{\mathfrak{t}_0}\{y\} + \sum_{l=1}^{L}\Big(\hat{\mathbb{E}}_{\mathfrak{t}_l}\{y\} - \hat{\mathbb{E}}_{\mathfrak{t}_{l-1}}\{y\}\Big).$$

HS transforms $\hat{f}$ into a shrunk model $\hat{f}_\lambda$ via the formula:

$$\hat{f}_\lambda(\mathbf{x}) := \hat{\mathbb{E}}_{\mathfrak{t}_0}\{y\} + \sum_{l=1}^{L}\frac{\hat{\mathbb{E}}_{\mathfrak{t}_l}\{y\} - \hat{\mathbb{E}}_{\mathfrak{t}_{l-1}}\{y\}}{1+\lambda/N(\mathfrak{t}_{l-1})}, \tag{10.1}$$

where λ is a hyperparameter chosen by the user, for example by cross validation. We emphasize that HS maintains the tree structure, and *only* modifies the prediction over each leaf node.

Since HS continues to make a constant prediction over each leaf node, our method thus comprises a one-off modification of these values. This can be computed in $O(m)$ time, where m is the total number of nodes in the tree. No other aspects of the underlying data structure are modified, with test time prediction occurring in exactly the same way as in the original tree. Moreover, our method HS does not even need to see the original training data, and only requires access to the fitted tree model. These features make it extremely lightweight and easy to implement, as we have done in the open-source package `imodels` [251]. By applying HS to each tree in an ensemble, it can be generalized to methods such as RF and gradient boosting.

While not typically done, it is possible to regularize RFs via other hyperparameters such as maximum tree depth. Tuning these hyperparameters, however, requires refitting the RF at every value in a grid. This quickly becomes computationally expensive, even for moderate dataset sizes[2], over multiple folds in a cross-validation (CV) set up. In contrast, since HS is applied post-hoc, *we only need to fit the RF once per CV fold*, leading to potentially enormous time savings. In addition, due to the connection between our method and ridge regression, it is even possible to get away with fitting the RF only *once* by using generalized cross-validation [95][3].

We also note the formula for LBS:

$$\hat{f}_\lambda^l(\mathbf{x}) := \hat{\mathbb{E}}_{\mathfrak{t}_0}\{y\} + \frac{\hat{\mathbb{E}}_{\mathfrak{t}_L}\{y\} - \hat{\mathbb{E}}_{\mathfrak{t}_0}\{y\}}{1+\lambda/N(\mathfrak{t}_L)}. \tag{10.2}$$

[2]Many popular tree-building algorithms such as CART have a run time of $O(pn^2)$ for constructing a binary tree.

[3]This allows for efficient computation of leave-one-out cross-validation error, which can be used to select λ, without refitting the RF.

Expanding this into a telescoping sum similar to (10.1), we see that the major difference between the two formulas is that whereas LBS shrinks each term by the same factor, HS shrinks each term by a different amount, with the amount of shrinkage controlled by the number of samples in the ancestor. This increased flexibility leads to better prediction performance for the final model, as evidenced by our results presented in the next section.

10.3 HS improves predictive performance on real-world datasets

Data overview

In this section, we study the performance of HS on a collection of classification and regression datasets selected as follows. For classification, we consider a number of datasets used in the classic Random Forest paper [33, 15], as well as two that are commonly used to evaluate rule-based models [280]. For regression, we consider all regression datasets used by [33] with at least 200 samples, as well as a variety of data-sets from the PMLB benchmark [227] ranging from small to large sample sizes. Table 10.1 displays the number of samples and features present in each dataset. In all cases, $2/3$ of the data is used for training (hyperparameters are selected via 3-fold cross-validation on this set) and $1/3$ of the data is used for testing.

	Name	Samples	Features
Classification	Heart	270	15
	Breast cancer	277	17
	Haberman	306	3
	Ionosphere [243]	351	34
	Diabetes [253]	768	8
	German credit	1000	20
	Juvenile [197]	3640	286
	Recidivism	6172	20
Regression	Friedman1 [88]	200	10
	Friedman3 [88]	200	4
	Diabetes [75]	442	10
	Geographical music	1059	117
	Red wine	1599	11
	Abalone [188]	4177	8
	Satellite image [227]	6435	36
	CA housing [199]	20640	8

Table 10.1: Real-world datasets analyzed here for classification (top panel) and regression (bottom panel).

HS improves prediction performance for commonly used tree methods

The prediction performance results for classification and regression are plotted in Fig. 10.3A and Fig. 10.3B respectively, with the number of leaves, as a proxy for the model complexity, plotted on the x-axis. We consider trees grown using four different techniques: CART, CART with cost-complexity pruning (CCP), C4.5, and GOSDT [150], a method that grows optimal trees in terms of the cost-complexity penalized misclassification loss.

For each of the four tree-growing methods, we grow a tree to a fixed number of leaves m,[4] for several different choices of $m \in \{2, 4, 8, 12, 15, 20, 24, 28, 30, 32\}$ (in practice, m would be pre-specified by a user or selected via cross-validation). For each tree, we compute its prediction performance before and after applying HS, where the regularization parameter for HS is selected from the set $\lambda \in \{0.1, 1.0, 10.0, 25.0, 50.0, 100.0\}$ via cross-validation. Results for each experiment are averaged over 10 random data splits. We observe that HS (solid lines in Fig. 10.3A,B) *does not hurt prediction in any of our data sets, and often leads to substantial performance gains.* For example, taking $m = 15$, we observe an average increase in relative predictive performance (measured by AUC) of $6.2\%, 6.5\%, 8\%$ for HS applied to CART and CART with CCP, and C4.5 respectively for the classification data sets. For the regression data sets with $m = 15$, we observe an average relative increase in R^2 performance of $9.8\%, 10.1\%$ for CART and CART with CCP respectively.

As expected, the improvements tend to be larger when m increases and for smaller datasets (e.g. the top row of Fig. 10.3A and Fig. 10.3B), although for larger datasets we see substantial improvements using HS once the number of leaves in the model is increased.

The fact that improvements hold for CART (CCP) shows that the effect of HS is not entirely replicated by tree structure regularization, and instead, *the two regularization methods can be used synergistically.* Indeed, applying HS can lead to the selection of a larger tree. Since tree models are sometimes used for subgroup search, larger trees from HS could allow for the discovery of otherwise undetected subgroups.

Fig. 10.2 shows a simulation result analyzing the bias-variance tradeoff for CART with and without HS. Here, data is generated from a linear model with Gaussian noise added during training. While predictive performance curves are often U-shaped because of the bias-variance tradeoff, those for HS are monotonic since HS is able to effectively reduce variance. The optimal regularization parameter λ decreases with the total number of leaves; this is corroborated by our calculations in Sec. 10.5.

HS outperforms LBS

We next compare the performance of HS to that of leaf-based shrinkage (LBS), which is used in XGBoost. Fig. 10.3C shows that hsCART tends to outperform CART (LBS), when repeating the same experiments as in Fig. 10.3A).

[4]For CART (CCP), we grow the tree to maximum depth, and tune the regularization parameter to yield m leaves.

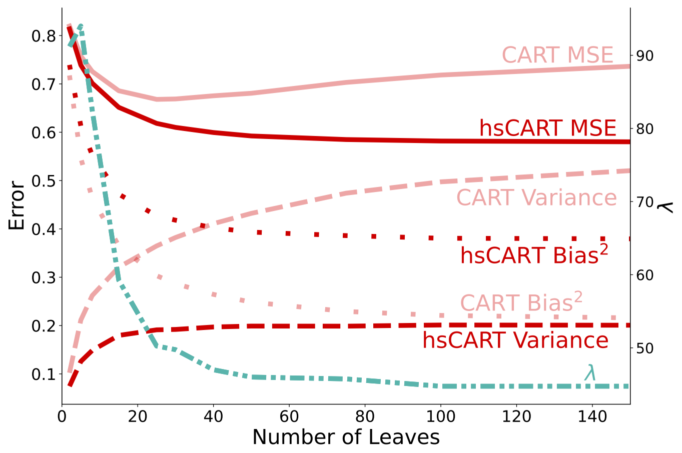


Figure 10.2: Test error for CART with HS (hsCART) stays low as the number of leaves increases, whereas CART test error increases due to overfitting. Data is simulated from a linear model with Gaussian noise.

HS improves prediction performance for RF

As mentioned earlier, trees in an RF are typically grown to purity without any constraints on depth or node size. Nonetheless, [171] argues that the `mtry` parameter[5], which controls the degree of feature sub-selection, "serves much the same purpose as the shrinkage penalty in explicitly regularized regression procedures like lasso and ridge regression." This parameter is typically set to a default value[6], and is not tuned. We compare the performance of regularizing RF via HS against maximum tree depth and `mtry`, tuning the hyperparameter for each method via cross validation. We repeat this for several different choices of B, the number of trees in the RF, and like before, average the results over 10 random data splits.

The results, displayed in Fig. 10.3D show that *HS significantly improves the prediction accuracy of RF* across the datasets we considered. Moreover, HS clearly outperforms the two RF-regularization methods (using `depth` and `mtry`) in all but one dataset (breast-cancer). This is especially promising because HS is also the fastest and easiest method to implement, as it does not require refitting the RF. Moreover, *hsRF tends to achieve its maximum performance with fewer trees* than RF without regularization; as a consequence, RF with HS is often able to achieve the same performance with an ensemble that is five times smaller, allowing us to achieve large savings in computational resources.

We also compare hsRF to the predictive performance of BART, and observe that hsRF and BART are comparable in terms of prediction performance. However, hsRF is much faster to fit than BART (typically 10-15 times faster) and we can also apply HS to BART.

[5]This parameter is denoted `mtry` in `ranger` and `max_features` in `scikit-learn`.

[6]Typically $\sqrt{p}$ for classification and $p/3$ for regression, where p is the number of features.

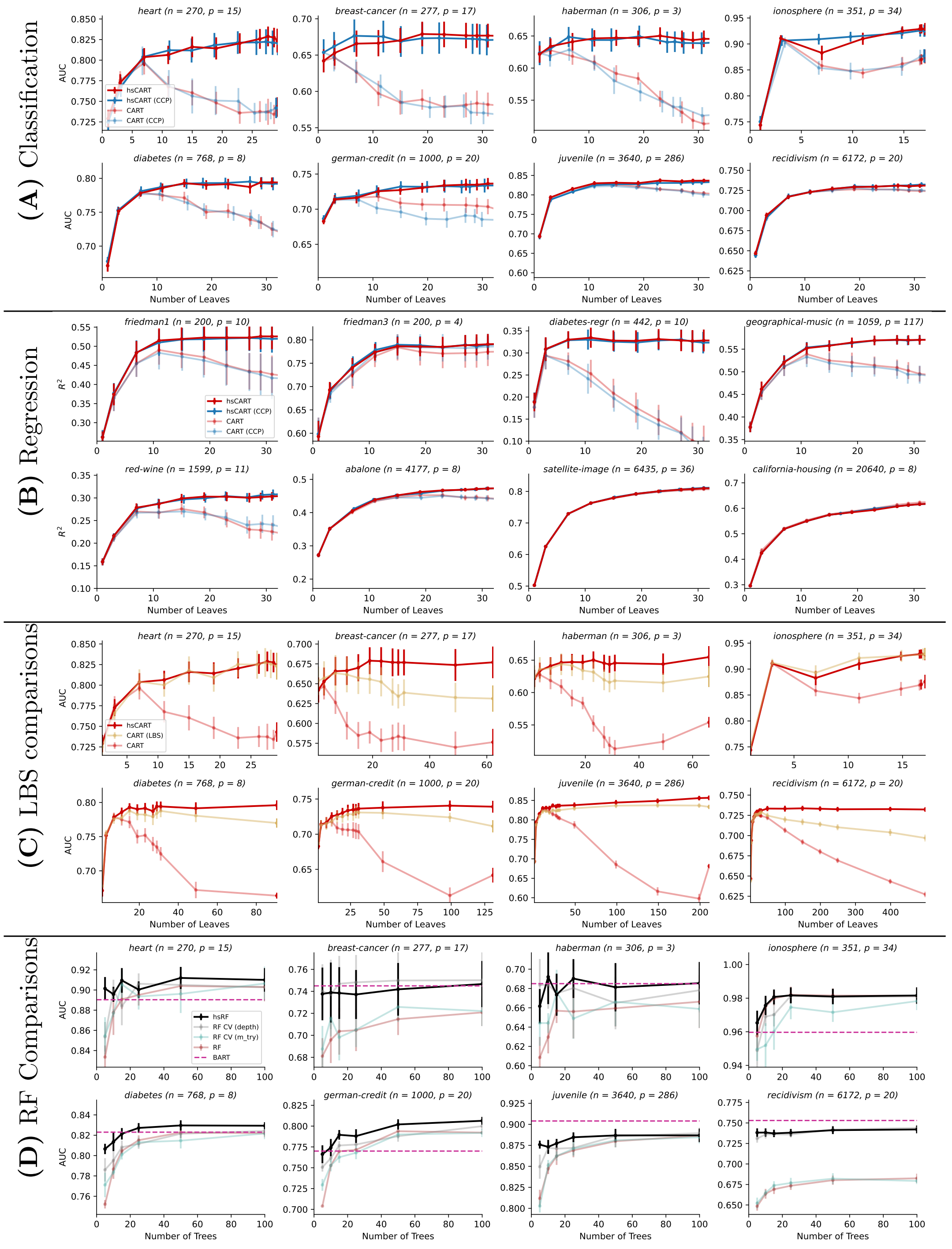


Figure 10.3: Hierarchical Shrinkage (solid lines) often improves predictive performance across various datasets, particularly for small datasets. **(A)** Top two rows show results for classification datasets (measured by AUC of the ROC curve) and **(B)** the next two rows show results for regression datasets (measured by R^2). HS often significantly improves the performance over CART, CART with CCP, and **(C)** leaf-based shrinkage. **(D)** HS even improves results for Random Forests as a function of the number of trees. Across all panels, errors bars show standard error of the mean computed over 10 random data splits. Note that the y-axis scales differ across plots.

10.4 HS improves RF interpretations by simplifying and stabilizing them

In addition to improving predictive performance, HS reduces variance and removes sampling artifacts, resulting in (i) simplified boundaries, (ii) stabilized feature importance scores, and (iii) making it easier to interpret interactions in the model.

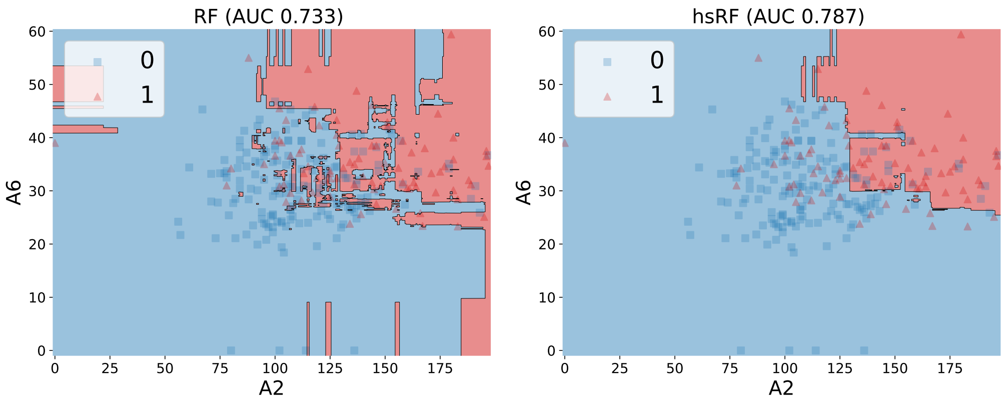


Figure 10.4: Comparison of decision boundary learned by RF vs hsRF on the Diabetes dataset, when fitted using only two features. HS prevents overfitting by creating a smoother, simpler decision boundary, resulting in improved performance and interpretability.

Fig. 10.4 shows an example of simplification: smoothing decision boundaries. On the diabetes dataset [253], RF can achieve strong performance (AUC 0.733) even when fitted to only two features. When HS is applied to this RF, the performance increases (to an AUC of 0.787), but the decision boundary also becomes considerably smoother and less fragmented. Since these two features are the only inputs to the model, these smooth boundaries enable a user to identify much clearer regions for high-risk predictions.

In models with many features, post-hoc interpretations, such as SHAP scores [159], can help a practitioner understand how a model makes its predictions. Fig. 10.5 shows that HS improves the stability of SHAP scores. Stability is measured using the variance of SHAP values when models are fit to 100 random train-test splits of the breast-cancer dataset. This makes the model interpretations less sensitive to minor data perturbations and thus more trustworthy. Moreover, these improvements in stability persist even for datasets such as Heart, Diabetes, and Ionosphere, for which HS does not greatly improve prediction performance. Hence, HS can improve the stability and interpretability of RF, even when it does not improve its predictive performance.

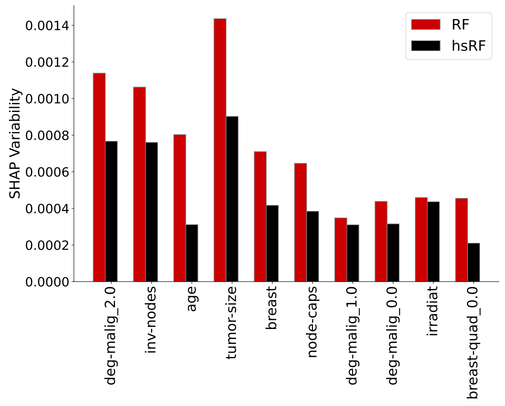


Figure 10.5: Comparison of SHAP plots learnt by Random Forests on the breast-cancer dataset before / after applying HS. HS displays lower variability across different data perturbations (without dropping in predictive performance), indicating enhanced stability.

Fig. 10.6 shows an example investigating the SHAP scores across the breast-cancer dataset for a trained RF. After applying HS, the SHAP values for each feature often cluster much more nicely. Each cluster corresponds to a group of samples for which a feature contributes a similar amount to the predicted response, regardless of the value of other features. As a result, each cluster can be interpreted without taking into account feature interactions, simplifying the user's interpretation. Since HS improves the model's predictive performance, the clustered SHAP scores suggest that HS improves performance by regularizing some unnecessary interactions in the model, and makes the fitted function closer to being additive, which allows for simpler interpretations.

10.5 HS as ridge regression on supervised features

Recent work by showed that decision trees are linear models on features obtained via supervised feature learning [131]. To see this, consider a tree model $\hat{f}$, with a fixed indexing of its interior nodes $\{\mathfrak{t}_0, \mathfrak{t}_1, \ldots, \mathfrak{t}_{m-1}\}$. We first associate to each node $\mathfrak{t}$ the decision stump

$$\psi_{\mathfrak{t}}(\mathbf{x}) = \frac{N(\mathfrak{t}_R)\mathbf{1}\{\mathbf{x} \in \mathfrak{t}_L\} - N(\mathfrak{t}_L)\mathbf{1}\{\mathbf{x} \in \mathfrak{t}_R\}}{\sqrt{N(\mathfrak{t}_L)N(\mathfrak{t}_R)}}, \tag{10.3}$$

where $\mathfrak{t}_L$ and $\mathfrak{t}_R$ denote the left and right children of $\mathfrak{t}$ respectively. This is a tri-valued function that is positive on the left child, negative on the right child, and zero everywhere else. Concatenating the decision stumps together yields a supervised feature map via $\Psi(\mathbf{x}) = \big(\psi_{\mathfrak{t}_0}(\mathbf{x}), \ldots, \psi_{\mathfrak{t}_{m-1}}(\mathbf{x})\big)$ and a transformed training set $\Psi(\mathcal{D}_n) \in \mathbb{R}^{n \times m}$. One can easily check that these feature vectors are orthogonal in $\mathbb{R}^n$, and furthermore that their squared ℓ_2 norms are the number of samples contained in their corresponding nodes: $\|\psi_{\mathfrak{t}_i}\|^2 = N(\mathfrak{t}_i)$.

More interestingly, on work showed (see Lemma 3.2 therein) that we have functional equality between the tree model and the kernel regression model with respect to the supervised feature map Ψ [131] was able to show , or in other words, $\hat{f}(\mathbf{x}) = \hat{\boldsymbol{\beta}}^T\Psi(\mathbf{x})$, where $\hat{\boldsymbol{\beta}} = \Psi(\mathcal{D}_n)^{\dagger}\mathbf{y}$. An easy extension of his proof yields the following result.

Theorem 3. *Let $\hat{\boldsymbol{\beta}}_\lambda$ be the solution to the ridge regression problem*

$$\min_{\boldsymbol{\beta}}\left\{\sum_{i=1}^{n}\left(\boldsymbol{\beta}^T\Psi(\mathbf{x}_i) - y_i\right)^2 + \lambda\|*\|\boldsymbol{\beta}^2\right\}. \tag{10.4}$$

We have the functional equality $\hat{f}_\lambda(\mathbf{x}) = \hat{\boldsymbol{\beta}}_\lambda^T\Psi(\mathbf{x})$.

Since the decision stumps (10.3) are orthogonal, we can decompose (10.4) into m independent univariate ridge regression problems, one with respect to each node $\mathbf{t}$:

$$\min_{\beta}\left\{\sum_{i=1}^{n}\left(\beta\psi_{\mathfrak{t}}(\mathbf{x}_i) - y_i\right)^2 + \lambda\beta^2\right\}. \tag{10.5}$$

Next, we use this connection of HS to ridge regression to argue heuristically that *the same λ works well for each regression subproblem* (10.5). This helps to justify our choice of denominator for each term in the HS formula (10.1) (a different choice would have led to a rescaling of the features $\psi_{\mathfrak{t}_i}$.)

Assume for the moment that the tree structure and hence the feature map is independent of the responses, which can be achieved via sample splitting. This is known in the literature

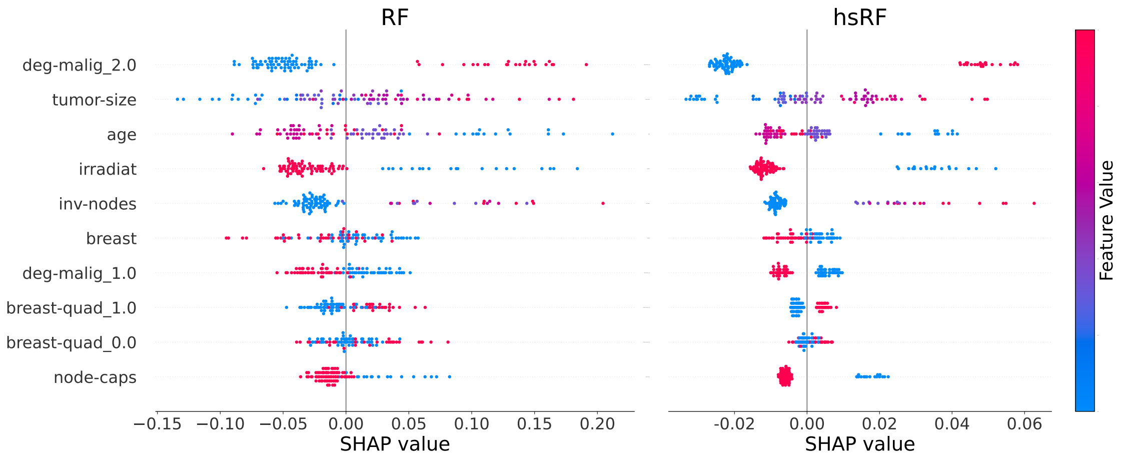


Figure 10.6: Comparison of SHAP values for RF on the breast-cancer dataset before/after applying HS with features arranged by importance. Each point represents a single sample in the dataset. HS leads to more clustered SHAP values for each feature, reflecting less heterogentity in the SHAP values of each feature.

as the "honesty condition", and has been widely used to simplify the analysis of tree-based methods [16]. Define $\boldsymbol{\beta}_* \in \mathbb{R}^m$ to have the value

$$\boldsymbol{\beta}_*(\mathfrak{t}) := \frac{\sqrt{N(\mathfrak{t}_L)N(\mathfrak{t}_R)}}{N(\mathfrak{t})}(\mathbb{E}\{y \mid \mathfrak{t}_L\} - \mathbb{E}\{y \mid \mathfrak{t}_R\}), \tag{10.6}$$

for the coordinate associated with each node $\mathfrak{t}$. For any query point $\mathbf{x}$, $\boldsymbol{\beta}_*^T\Psi(\mathbf{x})$ gives the mean response over the leaf $\mathfrak{t}(\mathbf{x})$ containing it. Furthermore, knowing $\Psi(\mathbf{x})$ is equivalent to knowing the leaf containing $\mathbf{x}$. Putting these two facts together show that the population residuals $r_i := y_i - \boldsymbol{\beta}_*^T\Psi(\mathbf{x})$ satisfy $\mathbb{E}\{r_i \mid \Psi(\mathbf{x}_i)\} = 0$, so that we have a generative linear model, in which we can calculate that the optimal regularization parameter for (10.5) is equal to $\lambda_{opt}(\mathfrak{t}) = \sigma^2(\mathfrak{t})/\boldsymbol{\beta}_*(\mathfrak{t})^2$, where $\sigma^2(\mathfrak{t})$ is roughly equal to the conditional variance[7] of the residual over $\mathfrak{t}$.

Given the connection between impurity and residual variance, if the tree model $\hat{f}$ considered in this section is grown using an impurity decrease stopping condition, we should expect the residual variance to be relatively similar over all leaves, so that $\sigma^2(\mathfrak{t})$ also does not vary too much over different nodes. Meanwhile

$$(\mathbb{E}\{y \mid \mathfrak{t}_L\} - \mathbb{E}\{y \mid \mathfrak{t}_R\})^2 \approx \mathrm{diam}(\mathfrak{t})^2 \approx 2^{-2\mathrm{depth}(\mathfrak{t})/p} \tag{10.7}$$

where p is the dimension of the original feature space. Since the maximum depth is typically $O(\log n)$, $\lambda_{opt}(\mathfrak{t})$ also does not vary too much across different nodes, and we do not lose too much by using a common value of λ across all the univariate subproblems (10.5) corresponding to different decision stump features.

A more naive (supervised) tree-based feature map is the one-hot encoding of an original feature vector obtained by treating the leaf index as a categorical variable. We denote this using Ξ. While Ξ can be obtained from Ψ via an invertible linear transformation, the two maps result in different kernels, and thus different ridge regression problems. Indeed, the ridge regression solution with respect to Ξ is equivalent to performing LBS on the tree model. The leaf indicator features are also orthogonal, so we may similarly decompose this ridge regression problem into independent univariate subproblems, one for each leaf. However, in this case, the population regression vector $\boldsymbol{\beta}_*^l$, for which $\boldsymbol{\beta}_*^{lT}\Xi(\mathbf{x})$ gives the expected response over each leaf, has coordinates equal to the population expectation over the leaves: $\boldsymbol{\beta}_*^l(\mathfrak{t}) = \mathbb{E}\{y \mid \mathfrak{t}\}$. As such, the optimal regularization parameters for different leaf nodes could be very different, and we lose more by having to use a common value of λ.

In practice, sample splitting is rarely done, and the feature map depends on the responses. Nonetheless, we believe that the heuristics detailed above continue to hold to a certain extent as shown in our experimental results.

[7]More precisely, it is equal to $\frac{N(\mathfrak{t}_L)}{N(\mathfrak{t})}\mathrm{Var}\{r \mid \mathfrak{t}_R\} + \frac{N(\mathfrak{t}_R)}{N(\mathfrak{t})}\mathrm{Var}\{r \mid \mathfrak{t}_L\}$.

10.6 HS Discussion

HS is a fast yet powerful regularization procedure that can be applied to any tree-based model without changing its structure. In our experiments, HS never hurts prediction performance, and often leads to substantial gains in both predictive performance and interpretability. HS is partly motivated by previous non-minimax-optimal generalization lower bounds for decision trees that predict using average responses over each leaf [268], pointing to a possible limitation of averaging.HS allows us to break this inferential barrier by pooling information from multiple leaves.

The work here naturally suggests many exciting future directions regarding the regularization of trees and RFs. First, replacing ridge regression with more sophisticated linear methods such as lasso or elastic net can result in other promising regularization methods. In addition, HS could improve other structured rule-based models, such as rule lists and tree sums [269].

Meanwhile, we have only scratched the surface of the relationships between regularization, robustness, and interpretability. Indeed, the connection between HS and ridge regression suggests the relevance of observations that ridge regression helps with robust generalization. Furthermore, we have seen that the simpler and smoother decision boundaries resulting from HS are more likely to generalize well. Hence, we conjecture that HS could improve the predictive performance of RF with respect to covariate shift. Moreover, the improved clustering and stability of SHAP scores after applying HS suggest that regularization via HS could improve the identification of important features, even when using alternative methods such as MDI.

Chapter 11

Real-world problem: clinical decision-rule development

11.1 Intro to clinical decision rules

Background Blunt intra-abdominal injury is a leading cause of preventable death and disability in children in the U.S. [128]. Computed tomography scans (CT) are the reference standard to diagnose intra-abdominal injury. In the last 30 years, CT use in children has increased without proportional improvements in clinical outcomes [170]. Indiscriminate use of CT is associated with an increased risk of radiation-induced malignancy [174]. Uncertainty and the lack of evidence in emergency department risk-stratification strategies lead to wide variation in CT use [166]. Furthermore, variability in practice increases cost and reduces effectiveness, efficiency, and quality of pediatric trauma care [276]. The Pediatric Emergency Care Applied Research Network (PECARN) prospectively developed a clinical decision instrument (CDI) to identify children after blunt torso trauma at very low risk for intra-abdominal injury undergoing acute intervention to decrease indiscriminate CT use [110].

Importance Emergency care requires rapid and accurate decisions across a diverse group of patients and practices. CDIs reduce variability for high-prevalence conditions by offering the potential for more accurate and reliable diagnostic strategies than clinician judgment alone [55]. However, before widespread use, CDIs require external validation. External validation is considered a more robust test of diagnostic performance than internal validation, and is critical to understanding the reliability of CDIs as they are generalized to new populations [224, 122]. If the CDI performs poorly during external validation, it may be refined, reconsidered, or even abandoned [98]. However, prospective external validation may be expensive and cumbersome. Therefore, introducing a step to assess a CDI before external validation can ensure that it is developed and modeled to be as predictive and stable as possible, to increase the chance of successful external validation.

Recent progress in data science has led to innovative frameworks to assess the prediction performance and stability of healthcare-related diagnostic models, such as CDIs. The Predictability-Computability-Stability (PCS) framework is a unified approach to data science that protects against instability induced by subjective decisions made during the data science lifecycle [292, 146]. PCS has improved drug-response prediction [146], gene-interaction search [22], and drug subgroup discovery in clinical trials [73]; these case-studies suggest that PCS may improve the CDI development and validation process before further investment into external validation. In addition to predictability as a reality check, two critical aspects of PCS are interpretability and stability analysis. To undergo PCS vetting, a CDI must be developed using interpretable methods, ensuring reproducibility [230]. Stability measures how much a CDI varies as choices made during the data science life cycle (including data cleaning and modeling), such as reasonable data alterations or different modeling techniques [293]. Here, multiple CDIs are developed by subsampling the original PECARN dataset. First, they are screened based on their test characteristics (predictability) and interpretability before assessing the variability of the importances of different predictor variables across high-performing CDIs (variable-level stability).

Goals of This Investigation The main objective of this study was to demonstrate the use of the PCS data science framework in vetting clinical decision instrument development (methods in Section 2.1 and results in Section 3.1). The secondary objective was to assess and externally validate the original PECARN clinical decision instrument for identifying children at very low risk of intra-abdominal injuries undergoing acute intervention after blunt torso trauma (methods in Section 2.2 and results in Section 3.2). Section 4 provides a discussion before Section 5 provides a conclusion.

11.2 Methods for clinical decision rule development with PCS

We analyzed two independent prospectively collected datasets from two large pediatric research networks, PECARN and the Pediatric Surgical Research Collaborative (PedSRC). This secondary analysis of anonymized data was deemed exempt from review by the University of California, San Francisco, and Medical University of South Carolina institutional review boards. There were two objectives of this study. The first (Section 2.1) was to demonstrate the PCS framework for improving CDI development. The second (Section 2.2) was to assess prediction performance and stability of the original PECARN CDI on external validation.

Objective 1: Demonstrate PCS Data Science Framework for Improving CDI Development

We followed the PCS framework, which goes beyond traditional reporting guidelines to assess the impact of reasonable human judgment calls by conducting reasonable data/model perturbations across the entire data science lifecycle [55, 293]. PCS offers a framework to assess a CDI for diagnostic performance based on predictive performance (i.e. sensitivity and specificity) and computational needs, putting weight on stability. During the development of a CDI, investigators make many “judgment calls”, i.e. subjective decisions which may lead to variability in the final developed CDI. PCS recommends that investigators ensure that study conclusions are stable to any such judgment calls. These judgment calls can be checked by measuring the stability of conclusions when alternative “reasonable” judgment calls are made. Reasonable judgment calls are those solicited through direct engagement between clinicians and data scientists (see the Discussion section for a more detailed look at PCS in the context of CDIs).

In this study, the PCS framework was applied to CDI development, including all CDI development and validation stages (it could also be applied to the data cleaning stage, but was not done here). First, the PCS framework (1) defines the clinical problem, then reviews all aspects of (2) collecting and preprocessing data, and (3) develops CDIs using interpretable and rule-based models. Next, these CDIs are vetted for their (4) predictive performance (predictability) and the importance of predictor variables. Last, PCS (5) supports the interpretation of results by identifying variability in all the PCS steps (stability), ensuring CDIs are developed to be supported by both data and domain knowledge (provider input). In addition, PCS guided all aspects of data documentation and analysis; code is available on Github[1] [98].

[1]`https://github.com/csinva/iai-clinical-decision-rule`

Development and Validation Dataset

The PECARN dataset is a prospective cohort of 12,044 children after blunt torso trauma between May 2007 and January 2010 in 20 emergency departments [110]. Predictor variables were collected prospectively using a standard data collection tool. We used the PECARN definition for the a priori outcome of interest of intra-abdominal injury undergoing acute intervention.

Following the original PECARN methods, we excluded any variable that was missing more than 5%, and used predictor variables with at least moderate inter-rater agreement, with the lower bound of the 95% confidence interval (CI) of the k measurements being at least 0.4 [289]. Missing values for a predictor variable were imputed via its median, and we manually combined predictors that conveyed redundant information based on their correlations.

Original PECARN CDI Development. Redevelopment of the PECARN CDI ensures the replicability of the original trial. We followed the original PECARN development and internal cross-validation process to redevelop the PECARN CDI to identify children at very low risk for intra-abdominal injuries undergoing acute intervention [110]. We used a Classification and Regression Trees (CART) rule list [32], which involves binary recursive partitioning using the Gini criterion [257].

PCS CDI Development. We developed several alternative CDIs (corresponding to different judgment calls during modeling) to compare the predictive performance and perform stability analysis of the PECARN CDI. The following models were used to develop CDIs: logistic regression, CART decision trees, rule lists [32], Bayesian Rule Lists [143], iterative Random Forests [22], RuleFit [86], Optimal sparse decision trees‘[150], Fast interpretable greedy-tree sums [269] and manual subgroup analysis. Each rule-based predictive model was chosen for its interpretability, taking the form of either a parsimonious list, tree, or set of binary rules. We used a stratified splitting technique to divide the PECARN dataset into a development set (i.e. a training set), 7,985 children (66%), and a validation set, 4,059 children (34%). Predictive models were fit using the imodels python package [251][2]. Hyperparameters were selected via manual tuning using only the development dataset.

CDI Predictive Performance. We calculated standard diagnostic statistics to report CDI performance. We used sensitivity and specificity curves to compare the diagnostic test characteristics of each CDI in the PECARN development and internal validation datasets. Furthermore, many more test characteristics were reported for each CDI, including their positive predictive value and Brier score (which helps evaluate the calibration of a CDI) [232]. The CDIs were ranked heuristically from the sensitivity-specificity curves by weighting (threshold-dependent) sensitivity five times more than specificity. CDIs with poor predictive performance (i.e., achieving a sensitivity below 90%) were eliminated before further analysis.

CDI Stability. We assessed CDI stability by performing side-by-side comparisons of the PECARN CDI and alternative CDIs. To assess predictor-variable stability, we report

[2]`https://github.com/csinva/imodels` *version 0.2.5*

the frequency and non-zero permutation-importance score of each predictor variable for each CDI [33]. The permutation importance measures the effect a predictor variable has on the overall prediction model's error. If a predictor variable is important, permuting or shuffling the value increases the model's error. The predictor variables with high permutation importance, especially across many different CDIs have greater stability.

We also compared the variability of diagnostic test characteristics between the PECARN development and internal validation datasets to assess the generalization of the model (i.e. stability of the predictive performance). Large changes in test characteristics suggest that the model is unstable in generalizing to new data. Moreover, a CDI can be unstable even when being re-developed to the same data. This is because many models contain some randomness in fitting, which can produce a different result when a model is re-developed. Therefore, we also measure randomness when each model is re-developed as a marker of stability. Prediction models were then ranked based on predictive performance (sensitivity and specificity), and then on variable-level stability.

Objective 2: Predictability and Stability of the Original PECARN CDI on External Validation

External Validation Dataset

The PedSRC dataset is based on a prospective cohort of 2,188 children with blunt trauma at 14 non-PECARN Level I pediatric trauma centers [259]. Predictor variables were collected prospectively using a standard data collection tool. The PedSRC study defined intra-abdominal injury as any injury to an intra-abdominal structure identified on abdominal CT or at laparotomy. We matched the a priori PedSRC outcome of intra-abdominal injury undergoing intervention to the PECARN outcome.

We matched predictor and outcome variables between the datasets through distribution assessment and expert review. To ensure consistent matching, all variable linkages between datasets were reviewed by domain experts, including PECARN and PedSRC study principal investigators, to ensure biologic plausibility and ensure original data definition was congruent between the respective datasets. Variables with subjectivity were further screened, original documentation reviewed, and expert authorship team consensus was used to match variables. The same missing data strategy was used on the PedSRC and PECARN datasets.

PCS External Validation. To externally validate each CDI, we calculated threshold-bound and threshold-free standard diagnostic statistics. We calculated sensitivity, specificity, negative and positive predictive values, positive and negative likelihood ratios. We also included false positives, false negatives, accuracy, and F1 score. The F1 score, an accuracy indicator, emphasizes the clinical relevance of sensitivity over specificity and ranges from 1 (best value) to 0 (worst value). We used sensitivity-specificity curves to compare the test characteristics of each candidate CDI on the external test dataset. We ranked predictor variable importance by assessing each variable's redundancy and weighted predictive power on the external validation dataset. Finally, we assessed overall CDI performance by evaluating the diagnostics test characteristics and variable importance.

We considered clinical context, predictive performance, computational speed, and stability to assign each CDI a rank. To compare the PCS framework to external validation, we first ranked predictive performance and stability. As the goal of the CDI is to limit unnecessary CT use in children after blunt torso trauma, we set a comparison threshold for predictive performance as a sensitivity five times more than specificity with a lower bound sensitivity of at least 95%. We calculated standard diagnostic statistics to report CDI performance, including sensitivity, specificity, negative and positive predictive values, positive, negative likelihood ratios, false positives, false negatives, accuracy, and F1 score. We also ranked predictor variable importance by assessing each variable's redundancy and weighted predictive power on the external validation dataset. Similarly, we measured overall stability as the proportion of the CDI's predictive performance assigned to predictor variables with the highest and lowest variable-level stability.

11.3 CDR Results

Results for Objective 1: Demonstrating the PCS Framework in CDI Development

Characteristics of Study Patients. The PECARN dataset included 12,044 children (Table 1). In PECARN, the mean (SD) age was 10.3 (5.4) years (1,167 patients <2 years), ranging from 0 to 18 years. The PedSRC external validation dataset included 2,188 children. The mean (SD) age was 7.8 (4.6) years (216 patients <2 years), ranging from 0 to 15 years. The PedSRC had a higher prevalence of motor vehicle collisions, compared to the PECARN development and validation datasets, 46.3% vs. 31.8% and 31.4%, and children with intra-abdominal injuries undergoing acute intervention, 2.8% vs. 1.7% and 1.7%, respectively (Table 1).

Clinical Decision Instrument Development. We replicated the original PECARN CDI development using the PECARN dataset and redeveloped the identical seven ordered decision predictor variables in the PECARN rule list. The randomness for all re-developed models had no effect on any of the final CDIs performances.

Clinical Decision Instrument Internal Validation. Each CDI had a decline in performance between the development and internal validation PECARN datasets (Fig. 11.1); however, the magnitude of the performance drop differed between different CDIs. The greater the magnitude in reduction suggests a less stable model. For example, the Iterative Random Forest CDI (red) and CART decision tree (orange) had the largest decline in performance between the development and validation datasets, suggesting that the prediction model was overfitting to the development dataset. In contrast, a fitted Bayesian rule list (blue), CART rule list (green), and Rule fit (purple) all retained similar predictive accuracy between de-

Table 11.1: Patient demographics and outcomes of the PECARN dataset split into development and validation (80:20), and the PedSRC external validation dataset.

	PECARN	**PedSRC**		
	Total (N=12,044)	**Development** (n=7,985)	**Internal Validation** (n=4,059)	**External Validation** (N=2,188)
Age <2 years (%)	1167 (9.7%)	761 (9.5%)	406 (10%)	216 (9.9%)
Sex Male (%)	7384 (61.3%)	4887 (61.2%)	2497 (61.5%)	N/A
MVC (%)	3832 (31.8%)	2505 (31.4%)	1327 (32.7%)	1014 (46.3%)
CT scan (%)	5,179 (43.0%)	3,393 (42.5%)	1,786 (44.0%)	967 (44.2%)
IAI (%)	761 (6.3%)	485 (6.1%)	276 (6.8%)	261 (11.9%)
IAI-I (%)	203 (1.7%)	133 (1.7%)	70 (1.7%)	62 (2.8%)

PECARN: Pediatric Emergency Care Applied Research Network; PedSRC: Pediatric Surgery Research Collaborative; MVC: motor vehicle collision; CT scan: computed tomography; IAI: intra-abdominal injury; IAI-I: intra-abdominal injury undergoing acute intervention

velopment and validation. Table 2 summarizes the results of threshold-specific weights in which the sensitivity is weighted five times more heavily as specificity.

Predictability The original PECARN, Rule Fit, and Bayesian Rule List had minimal changes in performance between the development and internal validation datasets, suggesting relatively high predictability for these CDIs (Fig. 11.1B). In contrast, CART Rule List, CART Decision Tree, and Iterative Random Forest had greater proportional declines in performance, suggesting lower predictability when heterogeneity in datasets was introduced.

Predictor-variable stability The most stable predictor variables were *abdominal trauma/seat belt sign*, *Glasgow Coma Scale Score < 14*, and *abdominal tenderness*. These three variables were the most frequent recurring predictor variables between CDIs. These three variables also had the highest non-zero permutation scores between the different CDIs. Therefore, it was recognized that the top three performing predictor variables were selected in the PECARN CDI and the four top-performing CDIs.

Results for Objective 2: External Validation of PECARN CDI

Distributions and Variable Matching for the external validation dataset. Predictor and outcome variables between the PECARN and PedSRC datasets were matched and evaluated for variable-level distributions (Fig. 11.2). Most variables had direct matches between datasets. The distribution of variables was well-matched except for the PECARN dataset inclusion of patients 15-17 years, and the lower frequency of children presenting after motor vehicle collisions (MVC).

External validation predictive performance. The original PECARN CDI successfully identified all but six children with intra-abdominal injuries undergoing acute interventions (sensitivity 97%, specificity 42.5%) on the PECARN dataset (Table 2b). On external validation using the PedSRC dataset, the original PECARN CDI maintained high prediction performance with an external validation sensitivity of 97.0% and specificity 44.0% (Table 2c). However, the original PECARN CDI missed two children with intra-abdominal injuries

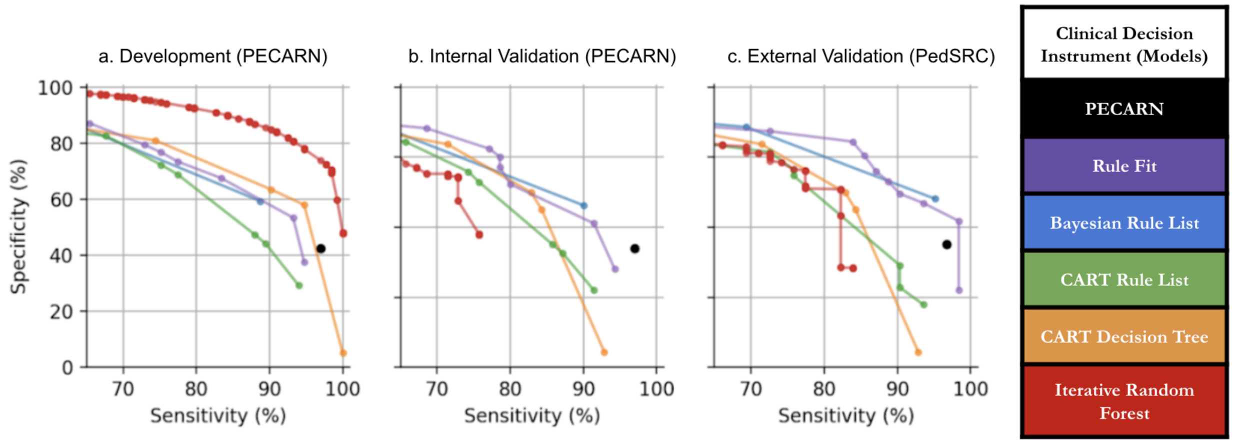


Figure 11.1: Sensitivity-specificity curves for clinical decision instruments to evaluate children after blunt torso trauma on the PECARN (a) development dataset (b) internal validation dataset, and (c) external validation on the PedSRC. The clinical decision instruments were then ranked by predictability from best to worst (top to bottom).

Tying together Objective 1 & 2: Comparing PCS Predictions to External Validation

The ranked overall performance of the CDI on external validation matched that of the PCS framework prediction rankings (Fig. 11.1C). This suggests that the results obtained from the PCS framework yielded useful information about the CDI's external validation performance, prior to collecting or analyzing the external validation dataset. In addition, the predictive performance was similar between internal validation and external validation (using the PedSRC dataset). However, most CDIs slightly improved their performances, suggesting that the CDIs are not overfitting to the PECARN dataset (Table 2c). The original PECARN CDI, Bayesian rule list, and Rule Fit had similar performances as in the PECARN datasets. In contrast, Iterative Random Forest, CART decision tree, and CART rule list had large declines in predictive performance (Fig. 11.1C).

11.4 Discussion on clinical decision rules

In the discussion, first we seek to describe PCS in the context of CDI development and vetting focusing on three key topics: predictability, stability, and interpretability. Next, we exemplify these three topics and their implications for the PECARN CDI.

Table 11.3: Predictive performance of the clinical decision instruments with sensitivity weighted five times more heavily as specificity. (a) PECARN Development dataset, (b) PECARN Internal Validation Dataset, (c) PedSRC External Validation Dataset.

(a) PECARN Development Dataset	**PECARN**	**Bayesian Rule List**	**CART Decision Tree**	**CART Rule List**	**Iterative Random Forest**	**Rule Fit**
Sensitivity	98%	89%	95%	94%	98%	95%
Specificity	43%	59%	58%	29%	70%	47%
F1 score	0.056	0.07	0.07	0.04	0.10	0.06
Brier score	0.016	0.02	0.58	0.02	0.01	0.08

(b) PECARN Internal Validation Dataset	**PECARN**	**Bayesian Rule List**	**CART Decision Tree**	**CART Rule List**	**Iterative Random Forest**	**Rule Fit**
Sensitivity	94%	90%	84%	91%	71%	97%
Specificity	41%	58%	56%	28%	68%	33%
F1 score	0.053	0.07	0.06	0.04	0.07	0.04
Brier score	0.016	0.02	0.59	0.02	0.02	0.08

(c) PedSRC External Validation Dataset	**PECARN**	**Bayesian Rule List**	**CART Decision Tree**	**CART Rule List**	**Iterative Random Forest**	**Rule Fit**
Sensitivity	96.8%	95%	94%	90%	81%	97%
Specificity	44.0%	60%	60%	39%	63%	55%
F1 score	0.091	0.12	0.12	0.07	0.11	0.11
Brier score	0.026	0.03	0.56	0.03	0.03	0.09

Contextualizing PCS in the context of CDI development

Predictability The predictive performance of a CDI serves as the benchmark in the clinical literature. The concept of diagnostic test characteristics, such as sensitivity and specificity, are well-described and clinically used metrics for predictability. For example, previous literature has found that the PECARN CDI has a higher sensitivity than clinical judgment alone [289]. This study sought to evaluate the predictability of a CDI using

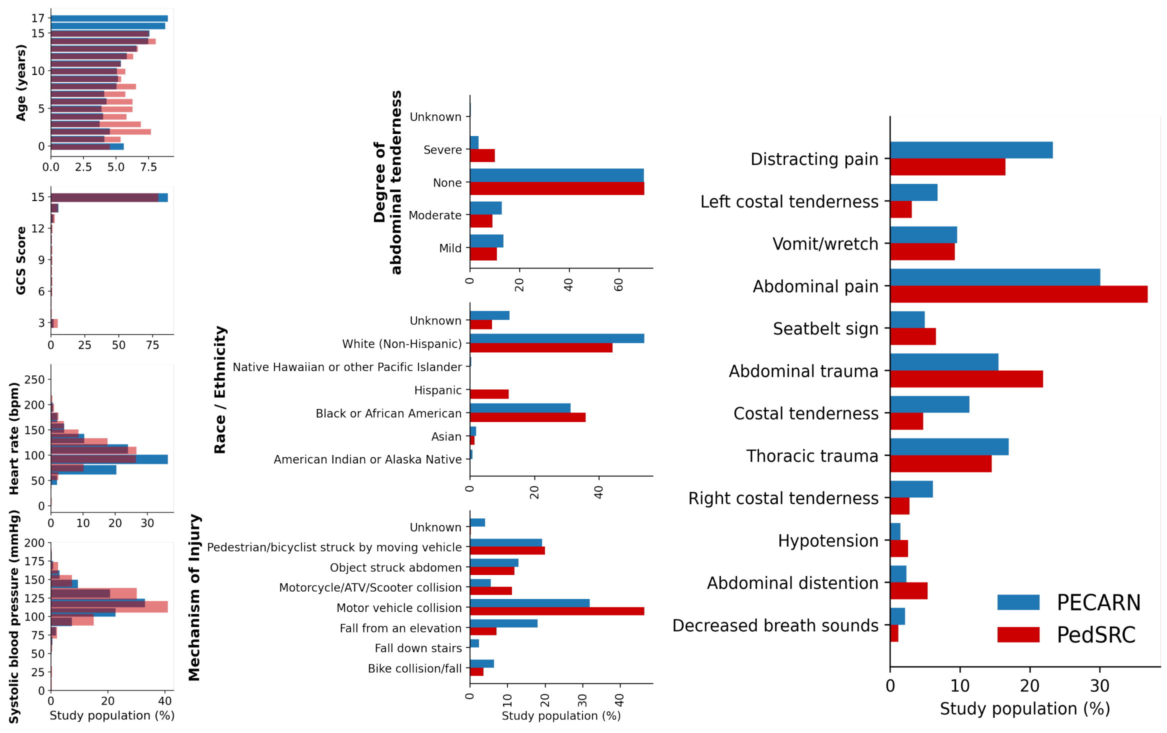


Figure 11.2: Matched demographic and predictor variables from PECARN and PedSRC datasets visually represented for overall distributions.

**GCS: Glasgow Coma Scale score; bpm: beats per minute; ATV: all-terrain vehicle; PECARN: Pediatric Emergency Care Applied Research Network; PedSRC: Pediatric Surgical Research Collaboration*

threshold-dependent discriminative metrics (i.e., sensitivity) and threshold-free metrics (i.e. sensitivity-specificity curves). We found that the PECARN, Bayesian, and Rule Fit CDIs were the most predictable on external validation (PedSRC). However, CDIs used in clinical practice are designed to make predictions on varying populations, over time, and within differing conditions. Therefore, before using a CDI in clinical practice, investigators should validate how well a CDI will perform under varying conditions.

Stability Stability should be checked for all aspects of the data science lifecycle. Here, we largely focus on predictor-level stability, estimating how the feature importance of each predictor variable changes as a result of different judgment calls made during modeling. We also examine the stability of both the predictive performance and individual predictors to different calls made during data preprocessing. For example, we tried using GCS as a continuous predictor variable compared to different binary thresholds. The effect of this and many other judgment calls were found to be minimal and are omitted here (but can be found on our github).

Interpretability Interpretability enables the integration of domain expertise for the development and implementation of a CDI [217, 191, 304]. In contrast, black-box machine-

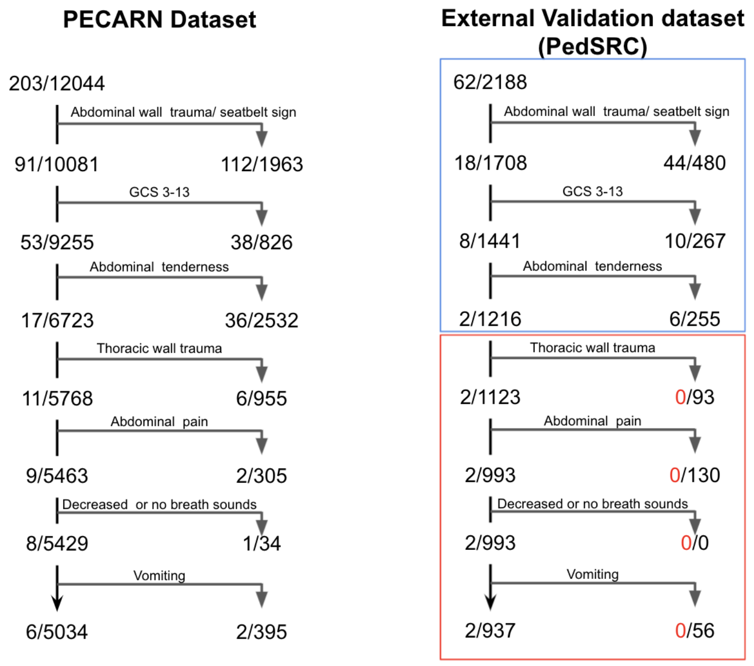


Figure 11.3: Prediction tree for the original PECARN clinical decision instrument on (a) PECARN internal validation dataset, and (b) PedSRC external validation dataset. The blue box shows that the top three predictor variables retained all the predictive power for the clinical decision instrument on external validation. The red box shows the predictor variables without prediction power on external validation. From the top of the rule to the bottom, risks for the identified subgroups monotonically decrease, although risks are systematically higher on the PedSRC data.

learning models lack interpretability and may fail for unknown reasons when externally validated [302]. Post-hoc interpretations, such as permutation importance used here, can offer some interpretability [159, 64, 5, 249], but are not a substitute for developing an interpretable model [230, 269, 251, 187]. Therefore, we only consider parsimonious rule-based models. Each CDI is represented as a straightforward set or list of logical rules (IF:THEN statements), which can then be visualized. We restrict each model to a reasonable number of logical steps (fewer than 10), so each CDI can be assessed in real-time. We additionally fit logistic regression and optimal decision tree models, but found that they had poor; we find that fast interpretable greedy-tree sums learn precisely the same rules as CART so we omit this model here. PCS offers clear documentation guidelines to ensure the process is replicable, reproducible, and interpretable [293].

As stated, black-box machine-learning models lack interpretability and may fail for unknown reasons when tested on new populations [191]. Examples of such complex models are neural networks, random forests, and support vector machines. However, even seemingly simple models such as logistic regression or decision trees can become uninterpretable if they are large enough and have too many steps [230]. Pennell (2020) utilized such models to re-evaluate the PECARN dataset [206]. The authors concluded that they had developed and validated a novel risk model using modern machine learning techniques. However, these complex machine-learning models lack the interpretability to integrate judgment, thus not allowing review nor the recognition of bias, which may build mistrust in the user [304]. Therefore, we use interpretable models with visual representation to allow stability analysis and ensure the integration of clinical judgment within the CDI [217].

Implications for the PECARN CDI

As the second aim of this paper, we assessed the prediction performance and the stability of the original PECARN CDI for identifying children at very low risk of intra-abdominal injuries undergoing acute intervention after blunt torso trauma on external validation. Clinically, there is no standard, generalizable, validated strategy to identify children after blunt torso trauma in whom CT scans can safely be avoided. Instead, providers use ad hoc strategies that are inaccurate, and may fail to identify life-threatening injuries, leading to over-reliance on diagnostic imaging [110, 42, 161, 127]. In 2013, PECARN sought to address the variability in accuracy and consistency by prospectively developing a CDI for children after blunt torso trauma [110].

We used two uniquely matched prospectively collected but independent datasets to assess the CDI predictions and stability on external validation. Through this process, we reexamined the original PECARN findings using alternative reasonable statistical models and found the original PECARN CDI to be high performing. The PECARN CDI was highly predictive across the development, internal validation, and external validation datasets. Therefore, PECARN has strong predictive performance, which measures how well a CDI predicts in heterogeneous cohorts. We also found that three predictor variables made up the entirety of the predictive power on external validation: abdominal wall trauma, Glasgow Coma Scale Score <14, and abdominal tenderness. This is not surprising, as these three variables were also the most stable based on the PCS framework and made up the majority of the predictive power on the PECARN dataset (identifying 94.4% of the correctly predicted IAI-I patients).

Through the PCS framework, we found that the predictability, and stability of the original PECARN CDI warrants further investment and investigation, including prospective external validation. In contrast, if we found that the model or predictor variables were unstable in the original study, we would recommend against further validation. Our study can serve as an example for how investigators may evaluate the predictability and stability of a CDI for inherent weakness, prior to investing in a prospective external validation.

We found that if PCS could be successfully integrated as a novel step into prediction and diagnostic model development before external validation, there is a potential to streamline and evaluate CDIs to improve performance or expose weaknesses and avoid further investment in CDIs with poor stability. This is important because many CDIs have reduced accuracy during external validation [272]. Introducing a PCS step between CDI development and external validation, or using PCS directly for CDI development before external validation, will allow researchers, funders, and clinicians to understand better how CDIs may perform on future populations before external validation, impact analysis, or implementation into clinical practice. However, PCS is not able to replace external validation.

There are limitations to this study. First, we sought to develop high performing but interpretable CDIs. Therefore, we chose only rule-based models, including simple regression-based and complex machine learning models with interpretable visual outputs. The inclusion of less interpretable models may have improved diagnostic accuracy but interfered with conducting stability analysis, introducing domain expertise, and more easily recognizing bias. Second, the PECARN and PedSRC datasets were collected from different research groups. There is a potential for partial verification bias on external validation because the PedSRC dataset was not based on consecutive patient enrollment, and follow-up was limited to medical record review. Third, three predictor variables did not match between datasets. Two variables could not be matched because they were present in only one of the datasets: *gender* (PECARN only) and *femur fracture* (PedSRC only). The third predictor variable was *distracting injury* (prospectively collected in PECARN but retrospectively aggregated in PedSRC). Given the limitations of this study, we believe prospective external validation is required before implementing the CDI.

Part IV

Open-source software and data

Chapter 12

imodels: a python library for interpretable modeling

imodels is a Python package for concise, transparent, and accurate predictive modeling. It provides users a simple interface for fitting and using state-of-the-art interpretable models, all compatible with scikit-learn [205] These models can often replace black-box models while improving interpretability and computational efficiency, all without sacrificing predictive accuracy. In addition, the package provides a framework for developing custom tools and rule-based models for interpretability.

Recent advancements in machine learning have led to increasingly complex predictive models, often at the cost of interpretability. There is often a need for models which are inherently interpretable [230, 186], particularly in high-stakes applications such as medicine, biology, and political science. In these cases, interpretability can ensure that models behave

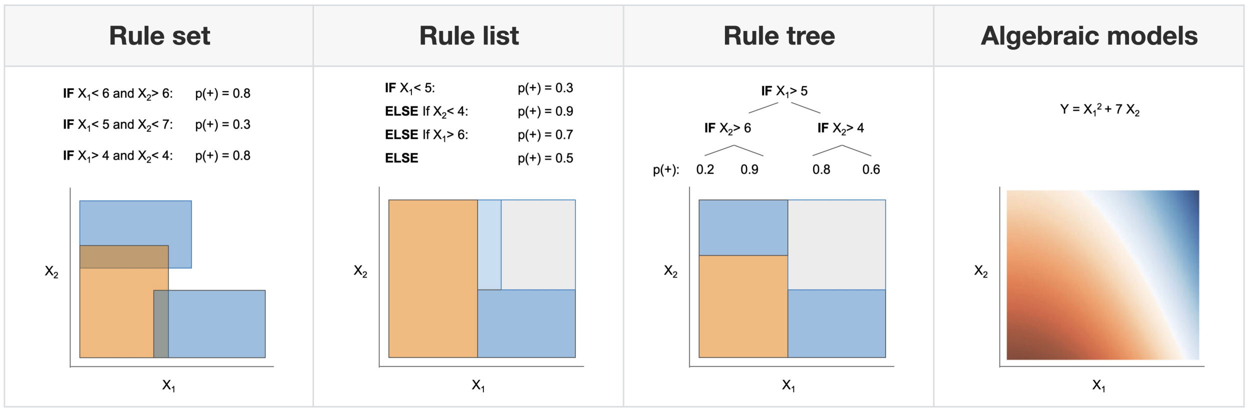

Rule set
IF X1< 6 and X2> 6: p(+) = 0.8
IF X1< 5 and X2< 7: p(+) = 0.3
IF X1> 4 and X2< 4: p(+) = 0.8
Rule list
IF X1< 5: p(+) = 0.3
ELSE If X2< 4: p(+) = 0.9
ELSE If X1> 6: p(+) = 0.7
ELSE p(+) = 0.5
Rule tree
IF X1> 5
IF X2> 6
IF X2> 4
p(+): 0.2 0.9 0.8 0.6
Algebraic models
Y = X1² + 7 X2
X1
X2

Figure 12.1: Examples of different supported model forms. The bottom of each box shows predictions of the corresponding model as a function of X_1 and X_2.

reasonably, identify when models will make errors, and make the models more trusted by domain experts. Moreover, interpretable models tend to be much more computationally efficient then larger black-box models.

Despite the development of many methods for fitting interpretable models [178], implementations for such models are often difficult to find, use, and compare to one another. *imodels* aims to fill this gap by providing a simple unified interface and implementation for many state-of-the-art interpretable modeling techniques.

12.1 Features

Interpretable models can take various forms. Fig. 12.1 shows four possible forms a model in the *imodels* package can take. Each form constrains the final model in order to make it interpretable, but there are different methods for fitting the model which differ in their biases and computational costs. The *imodels* package contains implementations of various such methods and also useful functions for recombining and extending them.

Rule sets consist of a set of rules which each act independently. There are different strategies for deriving a rule set, such as Skope-rules [54] or Rulefit [86]. Rule lists are composed of a set of rules which act in sequence, and include models such as Bayesian rule lists [143] or the oneR algorithm [111]. Rule trees are similar to rule lists, but allow branching after rules. This includes models such as CART decision trees [32]. Algebraic models take a final form of simple algebraic expressions, such as supersparse linear integer models [275].

Chapter 13

Veridical-flow: a python package for building trustworthy data-science pipelines with PCS

VeridicalFlow is a Python package for simplifying building reproducible and trustworthy data-science pipelines using the PCS framework [293]. It provides users a simple interface for stability analysis, i.e. checking the robustness of results from a data-science pipeline to various judgement calls made during modeling. This ensures that arbitrary judgement calls made by data-practitioners (e.g. specifying a default imputation strategy) do not dramatically alter the final conclusions made in a modeling pipeline. In addition to wrappers facilitating stability analysis, *VeridicalFlow* also automates many cumbersome coding aspects of python pipelines, including experiment tracking and saving, parallelization, and caching, all through integrations with existing python packages. Overall, the package helps to code using the PCS (predictability-computability-stability) framework, by screening models for predictive performance, helping automate computation, and facilitating stability analysis.

13.1 Statement of need

Predictability, computability, and stability are central concerns in modern statistical/machine-learning practice, as they are required to help vet that findings reflect reality, can be reasonably computed, and are robust as the many judgement calls during the data-science life cycle which often go unchecked [293].

The package focuses on stability, but also provides wrappers to help support and improve predictability and computability. Stability is a common-sense principle related to notions of scientific reproducibility [81, 118], sample variability, robust statistics, sensitivity analysis [235], and stability in numerical analysis and control theory. Moreover, stability serves as a prerequisite for understanding which parts of a model will generalize and can be interpreted [186].

Importantly, current software packages offer very little support to facilitate stability analyses. *VeridicalFlow* helps fill this gap by making stability analysis simple, reproducible, and computationally efficient. This enables a practitioner to represent a pipeline with many different perturbations in a simple-to-code way, while using prediction analysis as a reality check to screen out poor models.

13.2 Features

Using *VeridicalFlow*'s simple wrappers easily enables many best practices for data science, and makes writing pipelines easy.

Stability	Computability	Reproducibility
Replace a single function (e.g. preprocessing) with a set of functions representing different judgement calls and easily assess the stability of downstream results	Automatic parallelization and caching throughout the pipeline	Automatic experiment tracking and saving

Table 13.1: Three key aspects of *VeridicalFlow*.

The main features of *VeridicalFlow* center around stability analysis. The central concept is to replace given functions with a set of functions subject to different pipeline perturbations that are documented and argued for in PCS documentation [293]. Then, a set of useful analysis functions and computations enable easily assessing the stability to these perturbations on top of predictive screening for reality checks.

The package also helps users to improve the efficacy of their computational pipeline. Computation is (optionally) handled through Ray [183], which easily facilitates parallelization across different machines and along different perturbations of the pipeline. Caching is handled via *joblib*, so that individual parts of the pipeline do not need to be rerun.

Experiment-tracking and saving are (optionally) handled via integration with MLFlow [295], which enables automatic experiment tracking and saving.

Chapter 14

Covid-19: county-level data curation and death forecasting

In recent times, the COVID-19 pandemic has dramatically changed the shape of our global society and economy to an extent modern civilization has never experienced. Unfortunately, the vast majority of countries, the United States included, were thoroughly unprepared for the situation we now find ourselves in. There are currently many new efforts aimed at understanding and managing this evolving global pandemic. This article, together with the data we have collated (and continue to update), represents one such effort.

Our goals are to provide access to a large data repository combining data from a range of different sources and to forecast short-term (up to 2 weeks) COVID-19 mortality at the county level in the United States. We also provide uncertainty assessments of our forecasts in the form of prediction intervals based on conformal inference [277].

Predicting the short-term impact (e.g., 1 or 2 weeks in the future) of the virus in terms of the number of deaths is critical for many reasons. Not only can it help elucidate the overall fallout of the virus, but it can also help guide difficult policy decisions, such as whether or not to impose or ease lockdowns, and where to send much-needed personal protective equipment (PPE). While many other studies focus on predicting the long-term (several months or a year) trajectory of COVID-19, these approaches are currently difficult to verify due to a lack of long-term COVID-19 data.[1] On the other hand, predictions for immediate short-term trajectories are much easier to verify and are more likely to be accurate than long-term forecasts since there are comparatively fewer uncertainties involved, for example, due to policy change or behavioral changes in society. So far, the vast majority of predictive efforts have focused on modeling COVID-19 case counts or death counts at the national or state level [79], rather than the more fine-grained county level that we consider in this article. To the best of our knowledge, ours was the first work on county-level forecasts.[2]

[1]In the time since the first version of this article **in May 2020**, such longer-term predictions are likely now more verifiable.

[2]At the time of our first submission to arXiv on May 16, 2020, we were not aware of any concurrent work on county-level forecasts.

The predictions we produce in this article focus on recorded cumulative death counts, rather than recorded cases since recorded cases fail to accurately capture the true prevalence of the virus due to previously limited testing availability. Moreover, comparing different counties based on the *number* recorded cases is difficult since some counties conducted more tests than others: the number of positive tests does not equal the number of actual cases. While the *proportion* of tests that are positive is more comparable across different counties, our modeling approach focuses on recorded death counts rather than proportions, since these are not influenced by testing biases. It is worth noting, however, that the recorded death count is likely to be an undercount of the number of true number COVID-19 deaths, since evidence implies that many deaths that occurred outside of hospitals were often not counted.[3] Nonetheless, the recorded death count is generally believed to be more reliable than the recorded case count, and recent efforts have been made to ensure that COVID-19 death counts are more accurately recorded, for example, by including probable deaths and deaths occurring at home [189].

We first introduce our data repository and summarize the data sources contained within, as well as discussing any sources of bias in the data. This data repository is being updated continuously (as of October 2020) and includes a wide variety of COVID-19–related information in addition to the county-level case counts and death counts.

Next, we introduce our predictive approach, wherein we fit a range of different exponential and linear predictor models using our curated data. Each predictor captures a different aspect of the behaviors exhibited by COVID-19, both spatially and temporally, that is, across regions and time. The predictions generated by the different methods are combined using an ensembling technique by [238], and we refer to the ensemble model as the combined linear and exponential predictors (CLEP).

We also develop uncertainty estimates for our predictors in the form of prediction intervals, which we call maximum (absolute) error prediction intervals (MEPI). The ideas behind these intervals come from conformal inference [277] where the prediction interval coverage is well defined as the empirical proportion of days when the observed cumulative death counts fall inside the prediction intervals.

The results detail the evaluation of the predictors and the prediction intervals for the forecasts 3, 5, 7, and 14 days into the future. We use the data from January 23, 2020, the day after the first COVID-19 confirmed case (on January 22) in the United States [258], and report the prediction performance over the period March 22, 2020, to June 20, 2020. Overall, we find that CLEP predictions are adaptive to the exponential and subexponential nature of COVID-19 outbreak, with errors of around 15% for 7-day-ahead predictions, and errors of around 30% for 14-day-ahead predictions. We also provide detailed results for our prediction intervals MEPI from April 11, 2020, to June 20, 2020. And we observe that MEPIs are reasonably narrow and cover the recorded number of deaths for more than 90% of days for most of the counties in the United States.

Making both the data and the predictive algorithms used in this article accessible to oth-

[3]For the period up to June 21, 2020, considered in this article, for example, see [125]

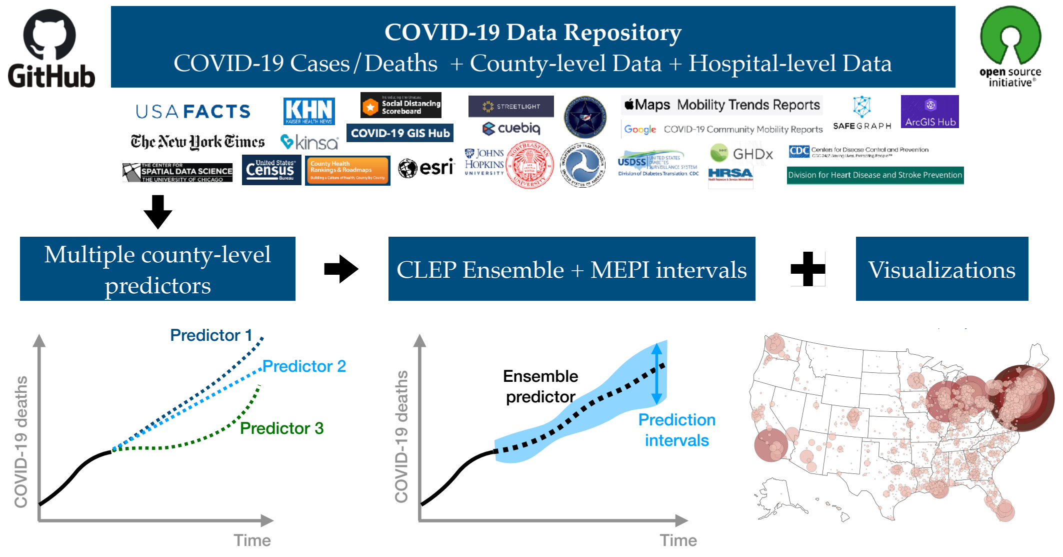


Figure 14.1: **An overview of the article.** We curate an extensive data repository combining data from multiple data sources. We then build several predictors for county-level predictions of cumulative COVID-19 death counts, and develop an ensembling procedure combined linear and exponential predictors (CLEP) and a prediction interval scheme maximum (absolute) error prediction intervals (MEPI) for these predictions. Both CLEP and MEPI are generic machine learning methods and can be of independent interest. All the data, and predictions are publicly available at GitHub repo (`https://github.com/Yu-Group/covid19-severity-prediction`). Visualizations are available at `https://covidseverity.com/` and `https://geodacenter.github.io/covid/map.html`, in collaboration with the Center for Spatial Data Science at the University of Chicago.

ers is key to ensuring their usefulness. Thus the data, code, and predictors we discuss in this article are open source on GitHub (`https://github.com/Yu-Group/covid19-severity-prediction`) and are also updated daily with several visualizations at `https://covidseverity.com`. While the results in this article contain case and death information at county level in the United States from January 23, 2020, to June 20, 2020; the data, forecasts, and visualizations in the GitHub repository and on our website continue to be updated daily. See Figure 14.1 for a high-level summary of the contributions made in this work.

We also reflect on the lessons learned from this process in a short perspective piece [294].

Appendix

This thesis omits some of the work done during my PhD. For example this includes work completed over summers, such as my work on using GANs for causal matching [245]. It also omits work that is still incomplete, including some deep-learning theory work on quantifying complexity using minimum-description length [72], interpretation work on tree ensembles [64], and using computer vision for hummingbird tracking[4]. I was also fortunate to be play a small part in some larger projects, such as NL-Augmenter [66]. It also does not mention my computational-neuroscience research, which was started during my undergraduate studies but mostly completed at the early start of my PhD [247, 181, 250, 91].

Funding I have been very fortunate to be supported financially throughout my PhD by The Center for Science of Information (CSoI), a US NSF Science and Technology Center under grant agreement CCF-0939370. I also gratefully acknowledge research support by grants ARO W911NF1710005, ONR N00014-16-1-2664, NSF DMS-1613002, and NSF IIS 1741340.

[4]`https://github.com/csinva/hummingbird-tracking`

Bibliography

[1] Reza Abbasi-Asl and Bin Yu. "Structural Compression of Convolutional Neural Networks Based on Greedy Filter Pruning". In: *arXiv preprint arXiv:1705.07356* (2017).

[2] Reza Abbasi-Asl et al. "The DeepTune framework for modeling and characterizing neurons in visual cortex area V4". In: *bioRxiv* (2018), p. 465534.

[3] Radhakrishna Achanta et al. "SLIC superpixels compared to state-of-the-art superpixel methods". In: *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012), pp. 2274–2282.

[4] Julius Adebayo et al. "Sanity checks for saliency maps". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9505–9515.

[5] Abhineet Agarwal et al. "Hierarchical Shrinkage: improving the accuracy and interpretability of tree-based methods". en. In: *arXiv:2202.00858 [cs, stat]* (Feb. 1, 2022). arXiv: 2202.00858. URL: `http://arxiv.org/abs/2202.00858`.

[6] François Aguet et al. "Advances in analysis of low signal-to-noise images link dynamin and AP2 to the functions of an endocytic checkpoint". In: *Developmental cell* 26.3 (2013), pp. 279–291.

[7] Hirotugu Akaike. "Factor analysis and AIC". In: *Selected Papers of Hirotugu Akaike*. Springer, 1987, pp. 371–386.

[8] André Altmann et al. "Permutation importance: a corrected feature importance measure". In: *Bioinformatics* 26.10 (2010), pp. 1340–1347.

[9] Dhammika Amaratunga, Javier Cabrera, and Yung-Seop Lee. "Enriched random forests". In: *Bioinformatics* 24.18 (2008), pp. 2010–2014.

[10] Marco Ancona et al. "Towards better understanding of gradient-based attribution methods for Deep Neural Networks". In: *6th International Conference on Learning Representations (ICLR 2018)*. 2018.

[11] Jacob Andreas et al. "Neural module networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 39–48.

[12] Elaine Angelino et al. "Learning certifiably optimal rule lists for categorical data". In: *arXiv preprint arXiv:1704.01701* (2017).

[13] Christof Angermueller et al. “Deep learning for computational biology”. In: *Molecular systems biology* 12.7 (2016), p. 878.

[14] PETER S. arcidiacono. “Exhibit A: EXPERT REPORT OF PETER S. ARCIDIACONO”. In: *http://samv91khoyt2i553a2t1s05i-wpengine.netdna-ssl.com/wp-content/uploads/2018/ 415-1-Arcidiacono-Expert-Report.pdf* (2018).

[15] Arthur Asuncion and David Newman. *UCI machine learning repository*. 2007.

[16] Susan Athey and Guido Imbens. “Recursive partitioning for heterogeneous causal effects”. In: *Proceedings of the National Academy of Sciences* 113.27 (2016), pp. 7353–7360.

[17] Sebastian Bach et al. “On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation”. In: *PloS one* 10.7 (2015), e0130140.

[18] David Baehrens et al. “How to explain individual classification decisions”. In: *Journal of Machine Learning Research* 11.Jun (2010), pp. 1803–1831.

[19] Yujia Bao et al. “Deriving machine attention from human rationales”. In: *arXiv preprint arXiv:1808.09367* (2018).

[20] Matthias Bartelmann and Peter Schneider. “Weak gravitational lensing”. In: *Physics Reports* 340.4-5 (2001), pp. 291–472. DOI: `10.1016/S0370-1573(00)00082-X`. arXiv: `astro-ph/9912508 [astro-ph]`.

[21] RL Barter and B Yu. “Superheat: Supervised heatmaps for visualizing complex data”. In: *arXiv preprint arXiv:1512.01524* (2015).

[22] Sumanta Basu et al. “Iterative random forests to discover predictive and stable high-order interactions”. en. In: *Proceedings of the National Academy of Sciences* 115.8 (Feb. 20, 2018). publisher: National Academy of Sciences section: Biological Sciences PMID: 29351989, pp. 1943–1948. ISSN: 0027-8424, 1091-6490. DOI: `10.1073/pnas.1711236115`.

[23] Merle Behr et al. “Provable Boolean Interaction Recovery from Tree Ensemble obtained via Random Forests”. In: *arXiv preprint arXiv:2102.11800* (2021).

[24] Anthony J Bell and Terrence J Sejnowski. “An information-maximization approach to blind separation and blind deconvolution”. In: *Neural computation* 7.6 (1995), pp. 1129–1159.

[25] PeterH Bennett, ThomasA Burch, and Max Miller. “Diabetes mellitus in American (Pima) indians”. In: *The Lancet* 298.7716 (1971), pp. 125–128.

[26] Dimitris Bertsimas and Jack Dunn. “Optimal classification trees”. In: *Machine Learning* 106.7 (2017), pp. 1039–1082.

[27] David M Blei, Andrew Y Ng, and Michael I Jordan. “Latent dirichlet allocation”. In: *Journal of machine Learning research* 3.Jan (2003), pp. 993–1022.

[28] Tolga Bolukbasi et al. “Man is to computer programmer as woman is to homemaker? debiasing word embeddings”. In: *Advances in neural information processing systems*. 2016, pp. 4349–4357.

[29] George EP Box. “Science and statistics”. In: *Journal of the American Statistical Association* 71.356 (1976), pp. 791–799.

[30] Danah Boyd and Kate Crawford. “Critical questions for big data: Provocations for a cultural, technological, and scholarly phenomenon”. In: *Information, communication & society* 15.5 (2012), pp. 662–679.

[31] Laurent Brechet et al. “Compression of biomedical signals with mother wavelet optimization and best-basis wavelet packet selection”. In: *IEEE Transactions on Biomedical Engineering* 54.12 (2007), pp. 2186–2192.

[32] L. Breiman et al. *Classification and Regression Trees*. Monterey, CA: Wadsworth and Brooks, 1984. URL: `https://www.routledge.com/Classification-and-Regression-Trees/Breiman-Friedman-Stone-Olshen/p/book/9780412048418`.

[33] Leo Breiman. “Random Forests”. en. In: *Machine Learning* 45.1 (Oct. 1, 2001), pp. 5–32. ISSN: 1573-0565. DOI: `10.1023/A:1010933404324`.

[34] Leo Breiman et al. “Statistical modeling: The two cultures (with comments and a rejoinder by the author)”. In: *Statistical science* 16.3 (2001), pp. 199–231.

[35] Leo Breiman and Jerome H Friedman. “Estimating optimal transformations for multiple regression and correlation”. In: *Journal of the American statistical Association* 80.391 (1985), pp. 580–598.

[36] Wieland Brendel, Jonas Rauber, and Matthias Bethge. “Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models”. In: *arXiv preprint arXiv:1712.04248* (2017).

[37] Tim Brennan and William L Oliver. “The emergence of machine learning techniques in criminology”. In: *Criminology & Public Policy* 12.3 (2013), pp. 551–562.

[38] Joan Bruna and Stéphane Mallat. “Invariant scattering convolution networks”. In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1872–1886.

[39] Kenneth P Burnham and David R Anderson. “Multimodel inference: understanding AIC and BIC in model selection”. In: *Sociological methods & research* 33.2 (2004), pp. 261–304.

[40] Kaylee Burns et al. “Women also snowboard: Overcoming bias in captioning models”. In: *arXiv preprint arXiv:1803.09797* (2018).

[41] Sidney Burrus, Ramesh Gopinath, and Haitao Guo. “Introduction to wavelets and wavelet transforms: a primer”. In: *Englewood Cliffs* (1997).

[42] Andrew J. Capraro, David Mooney, and Mark L. Waltzman. “The use of routine laboratory studies as screening tools in pediatric abdominal trauma”. eng. In: *Pediatric Emergency Care* 22.7 (July 2006). PMID: 16871106, pp. 480–484. ISSN: 1535-1815. DOI: `10.1097/01.pec.0000227381.61390.d7`.

[43] David Card. “Exhibit 33: Report of David Card”. In: *https://projects.iq.harvard.edu/files/diverse-education/files/legal_-_card_report_revised_filing.pdf* (2018).

[44] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. “An empirical evaluation of supervised learning in high dimensions”. In: *Proceedings of the 25th International Conference on Machine learning.* 2008, pp. 96–103.

[45] Rich Caruana and Alexandru Niculescu-Mizil. “An empirical comparison of supervised learning algorithms”. In: *Proceedings of the 23rd International Conference on Machine learning.* 2006, pp. 161–168.

[46] Rich Caruana et al. “Case-based explanation of non-case-based learning methods.” In: *Proceedings of the AMIA Symposium.* American Medical Informatics Association. 1999, p. 212.

[47] Rich Caruana et al. “Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission”. In: *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* ACM. 2015, pp. 1721–1730.

[48] Supriyo Chakraborty et al. “Interpretability of deep learning models: a survey of results”. In: *Interpretability of deep learning models: a survey of results.* 2017.

[49] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining.* 2016, pp. 785–794.

[50] Xi Chen and Hemant Ishwaran. “Random forests for genomic data analysis”. In: *Genomics* 99.6 (2012), pp. 323–329.

[51] Hugh A Chipman, Edward I George, and Robert E McCulloch. “BART: Bayesian additive regression trees”. In: *The Annals of Applied Statistics* 4.1 (2010), pp. 266–298.

[52] Noel Codella et al. “Skin Lesion Analysis Toward Melanoma Detection 2018: A Challenge Hosted by the International Skin Imaging Collaboration (ISIC)”. In: *arXiv preprint arXiv:1902.03368* (2019).

[53] William W Cohen and Yoram Singer. “A simple, fast, and effective rule learner”. In: *AAAI/IAAI* 99.335-342 (1999), p. 3.

[54] Skope Collaboration. *Skope-rules.* 2021. URL: `https://github.com/scikit-learn-contrib/skope-rules`.

[55] Gary S. Collins et al. "Transparent reporting of a multivariable prediction model for individual prognosis or diagnosis (TRIPOD): The tripod statement". In: *Journal of Clinical Epidemiology* 68.2 (2015). publisher: The Authors, pp. 112–121. DOI: `10.1016/j.jclinepi.2014.11.010`.

[56] Fergal Cotter. "Uses of Complex Wavelets in Deep Convolutional Neural Networks". PhD thesis. University of Cambridge, 2020.

[57] William R Coulton et al. "Constraining neutrino mass with the tomographic weak lensing bispectrum". In: *Journal of Cosmology and Astroparticle Physics* 2019.05, 043 (2019), p. 043. DOI: `10.1088/1475-7516/2019/05/043`. arXiv: `1810.02374 [astro-ph.CO]`.

[58] Mark Craven and Jude W Shavlik. "Extracting tree-structured representations of trained networks". In: *Advances in neural information processing systems*. 1996, pp. 24–30.

[59] Piotr Dabkowski and Yarin Gal. "Real Time Image Saliency for Black Box Classifiers". In: *arXiv preprint arXiv:1705.07857* (2017).

[60] Anupam Datta, Shayak Sen, and Yair Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems". In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 598–617.

[61] I DAUBECHIES. "Orthonormal bases of compactly supported wavelets". In: *Commun. Pure Appl. Math.* 41 (1988), pp. 909–996.

[62] Krzysztof Dembczyński, Wojciech Kotłowski, and Roman Słowiński. "Maximum likelihood rule ensembles". In: *Proceedings of the 25th international conference on Machine learning*. 2008, pp. 224–231.

[63] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[64] Summer Devlin et al. "Disentangled Attribution Curves for Interpreting Random Forests and Boosted Trees". In: *arXiv preprint arXiv:1905.07631* (2019).

[65] J. Dheeba, N. Albert Singh, and S. Tamil Selvi. "Computer-aided detection of breast cancer on mammograms: A swarm intelligence optimized wavelet neural network approach". In: *Journal of Biomedical Informatics* 49 (2014), pp. 45–52. ISSN: 1532-0464. DOI: `https://doi.org/10.1016/j.jbi.2014.01.010`. URL: `https://www.sciencedirect.com/science/article/pii/S1532046414000124`.

[66] Kaustubh D Dhole et al. "NL-Augmenter: A Framework for Task-Sensitive Natural Language Augmentation". In: *arXiv preprint arXiv:2112.02721* (2021).

[67] William Dieterich, Christina Mendoza, and Tim Brennan. "COMPAS risk scales: Demonstrating accuracy equity and predictive parity". In: *Northpointe Inc* (2016).

[68] Finale Doshi-Velez and Been Kim. "A roadmap for a rigorous science of interpretability". In: *arXiv preprint arXiv:1702.08608* (2017).

[69] Finale Doshi-Velez and Been Kim. “Towards a rigorous science of interpretable machine learning”. In: *arXiv preprint arXiv:1702.08608* (2017).

[70] Mengnan Du et al. “Learning credible deep neural networks with rationale regularization”. In: *arXiv preprint arXiv:1908.05601* (2019).

[71] James Duncan et al. “VeridicalFlow: a Python package for building trustworthy data science pipelines with PCS”. In: *Journal of Open Source Software* 7.69 (2022), p. 3895.

[72] Raaz Dwivedi et al. “Revisiting minimum description length complexity in overparameterized models”. In: *arXiv preprint arXiv:2006.10189* (2020).

[73] Raaz Dwivedi et al. “Stable Discovery of Interpretable Subgroups via Calibration in Causal Studies”. en. In: *International Statistical Review* 88.S1 (2020), S135–S178. ISSN: 1751-5823. DOI: `10.1111/insr.12427`.

[74] Cynthia Dwork et al. “Fairness through awareness”. In: *Proceedings of the 3rd innovations in theoretical computer science conference*. ACM. 2012, pp. 214–226.

[75] Bradley Efron et al. “Least angle regression”. In: *The Annals of statistics* 32.2 (2004), pp. 407–499.

[76] Gabriel Erion et al. “Learning Explainable Models Using Attribution Priors”. In: *arXiv preprint arXiv:1906.10670* (2019).

[77] Andre Esteva et al. “Dermatologist-level classification of skin cancer with deep neural networks”. In: *Nature* 542.7639 (2017), p. 115.

[78] Jeffrey S Evans et al. “Modeling species distribution and change using random forest”. In: *Predictive Species and Habitat Modeling in Landscape Ecology*. Springer, 2011, pp. 139–159.

[79] NM Ferguson et al. *Impact of non-pharmaceutical interventions (NPIs) to reduce COVID19 mortality and healthcare demand*. 2020. URL: `https://www.imperial.ac.uk/media/imperial-college/medicine/sph/ide/gida-fellowships/Imperial-College-COVID19-NPI-modelling-16-03-2020.pdf`.

[80] Manuel Fernández-Delgado et al. “Do we need hundreds of classifiers to solve real world classification problems?” In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 3133–3181.

[81] Ronald Aylmer Fisher et al. “The design of experiments.” In: *The design of experiments.* 2nd Ed (1937). DOI: `10.1038/137252a0`.

[82] Janis Fluri et al. “Cosmological constraints with deep learning from KiDS-450 weak lensing maps”. In: *Physical Review D* 100.6, 063514 (2019), p. 063514. DOI: `10.1103/PhysRevD.100.063514`. arXiv: `1906.03156 [astro-ph.CO]`.

[83] Ruth C Fong and Andrea Vedaldi. “Interpretable explanations of black boxes by meaningful perturbation”. In: *arXiv preprint arXiv:1704.03296* (2017).

[84] David A Freedman. “Statistical models and shoe leather”. In: *Sociological methodology* (1991), pp. 291–313.

[85] Yoav Freund, Robert E Schapire, et al. “Experiments with a new boosting algorithm”. In: *icml*. Vol. 96. Citeseer. 1996, pp. 148–156.

[86] J. H. Friedman and B. E. Popescu. “Predictive Learning Via Rule Ensembles”. In: *The Annals of Applied Statistics* 2.3 (2008), pp. 916–954. DOI: 10.1214/07-aoas148.

[87] Jerome H Friedman. “Greedy function approximation: a gradient boosting machine”. In: *Annals of statistics* (2001), pp. 1189–1232.

[88] Jerome H Friedman. “Multivariate adaptive regression splines”. In: *The annals of statistics* (1991), pp. 1–67.

[89] Jerome Friedman, Trevor Hastie, Robert Tibshirani, et al. *The Elements of Statistical Learning*. Vol. 1. 10. Springer Series in Statistics New York, 2001.

[90] Nicholas Frosst and Geoffrey Hinton. “Distilling a Neural Network Into a Soft Decision Tree”. In: *arXiv preprint arXiv:1711.09784* (2017).

[91] Jan Funke et al. “Large scale image segmentation with structured loss based deep learning for connectome reconstruction”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.7 (2018), pp. 1669–1680.

[92] Robert Geirhos et al. “ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness”. In: *arXiv preprint arXiv:1811.12231* (2018).

[93] Leilani H Gilpin et al. “Explaining Explanations: An Approach to Evaluating Interpretability of Machine Learning”. In: *arXiv preprint arXiv:1806.00069* (2018).

[94] Fréderic Godin et al. “Explaining Character-Aware Neural Networks for Word-Level Prediction: Do They Discover Linguistic Rules?” In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2018, pp. 3275–3284.

[95] Gene H. Golub, Michael Heath, and Grace Wahba. “Generalized Cross-Validation as a Method for Choosing a Good Ridge Parameter”. In: *Technometrics* 21.2 (1979), pp. 215–223. DOI: 10.1080/00401706.1979.10489751. eprint: https://www.tandfonline.com/doi/pdf/10.1080/00401706.1979.10489751. URL: https://www.tandfonline.com/doi/abs/10.1080/00401706.1979.10489751.

[96] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and harnessing adversarial examples”. In: *arXiv preprint arXiv:1412.6572* (2014).

[97] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a” right to explanation””. In: *arXiv preprint arXiv:1606.08813* (2016).

[98] Steven M. Green, David L. Schriger, and Donald M. Yealy. "Methodologic standards for interpreting clinical decision rules in emergency medicine: 2014 update". In: *Annals of Emergency Medicine* 64.3 (2014). publisher: American College of Emergency Physicians, pp. 286–291. DOI: `10.1016/j.annemergmed.2014.01.016`.

[99] Riccardo Guidotti et al. "A Survey Of Methods For Explaining Black Box Models". In: *arXiv preprint arXiv:1802.01933* (2018).

[100] Wooseok Ha et al. "Adaptive wavelet distillation from neural networks through interpretations". In: *Advances in Neural Information Processing Systems* 34 (2021).

[101] Frank R Hampel et al. *Robust statistics: the approach based on influence functions.* Vol. 196. John Wiley & Sons, 2011.

[102] Moritz Hardt, Eric Price, Nati Srebro, et al. "Equality of opportunity in supervised learning". In: *Advances in neural information processing systems.* 2016, pp. 3315–3323.

[103] Gilbert H Harman. "The inference to the best explanation". In: *The philosophical review* 74.1 (1965), pp. 88–95.

[104] Trevor Hastie and Robert Tibshirani. "Generalized Additive Models". In: *Statistical Science* 1.3 (1986), pp. 297–318.

[105] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition.* 2016, pp. 770–778.

[106] Kangmin He et al. "Dynamics of Auxilin 1 and GAK in clathrin-mediated traffic". In: *Journal of Cell Biology* 219.3 (2020).

[107] James M Hereford, David W Roach, and Ryan Pigford. "Image compression using parameterized wavelets with feedback". In: *Independent Component Analyses, Wavelets, and Neural Networks.* Vol. 5102. International Society for Optics and Photonics. 2003, pp. 267–277.

[108] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. "Distilling the knowledge in a neural network". In: *arXiv preprint arXiv:1503.02531* (2015).

[109] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[110] James F Holmes et al. "Identifying children at very low risk of clinically important blunt abdominal injuries." In: *Annals of emergency medicine* 62.2 (Aug. 2013). publisher: Elsevier Inc., 107–116.e2. DOI: `10.1016/j.annemergmed.2012.11.009`.

[111] Robert C Holte. "Very simple classification rules perform well on most commonly used datasets". In: *Machine learning* 11.1 (1993), pp. 63–90. DOI: `10.1023/A:1022631118932`.

[112] Giles Hooker. "Generalized functional anova diagnostics for high-dimensional functions of dependent variables". In: *Journal of Computational and Graphical Statistics* 16.3 (2007), pp. 709–732.

[113] Giles Hooker and Lucas Mentch. “Bridging Breiman’s Brook: From Algorithmic Modeling to Statistical Learning”. In: *Observational Studies* 7.1 (2021), pp. 107–125.

[114] Harold Hotelling. “Relations between two sets of variates”. In: *Biometrika* 28.3/4 (1936), pp. 321–377.

[115] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. “Optimal sparse decision trees”. In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[116] Cheng-Lung Huang, Mu-Chen Chen, and Chieh-Jen Wang. “Credit scoring with a data mining approach based on support vector machines”. In: *Expert systems with applications* 33.4 (2007), pp. 847–856.

[117] Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

[118] Peter Ivie and Douglas Thain. “Reproducibility in scientific computing”. In: *ACM Computing Surveys (CSUR)* 51.3 (2018), pp. 1–36. DOI: `10.1145/3186266`.

[119] Sarthak Jain and Byron C Wallace. “Attention is not Explanation”. In: *arXiv preprint arXiv:1902.10186* (2019).

[120] Dhruv Jawali, Abhishek Kumar, and Chandra Sekhar Seelamantula. “A Learning Approach for Wavelet Design”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2019, pp. 5018–5022.

[121] IT Jolliffe. “Principal component analysis”. In: (1986).

[122] A. C. Justice, K. E. Covinsky, and J. A. Berlin. “Assessing the generalizability of prognostic information”. eng. In: *Annals of Internal Medicine* 130.6 (Mar. 16, 1999). PMID: 10075620, pp. 515–524. ISSN: 0003-4819. DOI: `10.7326/0003-4819-130-6-199903160-00016`.

[123] Marko Kaksonen and Aurélien Roux. “Mechanisms of clathrin-mediated endocytosis”. In: *Nature Reviews Molecular Cell Biology* 19.5 (2018), p. 313.

[124] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. “Visualizing and understanding recurrent networks”. In: *arXiv preprint arXiv:1506.02078* (2015).

[125] Josh Katz, Denise Lu, and Margot Sanger-katz. “U.S. Coronavirus Death Toll Is Far Higher Than Reported, C.D.C. Data Suggests”. In: *The New York Times* (2020). URL: `https://www.nytimes.com/interactive/2020/04/28/us/coronavirus-death-toll-total.html`.

[126] Frank C Keil. “Explanation and understanding”. In: *Annu. Rev. Psychol.* 57 (2006), pp. 227–254.

[127] M. S. Keller et al. “The utility of routine trauma laboratories in pediatric trauma resuscitations”. In: *Am J Surg* 188.6 (Dec. 2004). edition: 2004/12/28, pp. 671–8. ISSN: 0002-9610 (Print) 0002-9610 (Linking). DOI: `10.1016/j.amjsurg.2004.08.056`.

[128] Mary Ella Kenefake, Matthew Swarm, and Jennifer Walthall. “Nuances in pediatric trauma”. eng. In: *Emergency Medicine Clinics of North America* 31.3 (Aug. 2013). PMID: 23915597, pp. 627–652. ISSN: 1558-0539. DOI: 10.1016/j.emc.2013.04.004.

[129] Jinkyu Kim and John F Canny. “Interpretable Learning for Self-Driving Cars by Visualizing Causal Attention.” In: *ICCV*. IEEE. 2017, pp. 2961–2969.

[130] Tom Kirchhausen, David Owen, and Stephen C Harrison. “Molecular structure, function, and dynamics of clathrin-mediated membrane traffic”. In: *Cold Spring Harbor perspectives in biology* 6.5 (2014), a016725.

[131] Jason M. Klusowski. “Universal Consistency of Decision Trees in High Dimensions”. In: *arXiv preprint arXiv:2104.13881* (2021).

[132] Thomas Kluyver et al. “Jupyter Notebooks-a publishing format for reproducible computational workflows.” In: *ELPUB*. 2016, pp. 87–90.

[133] Pang Wei Koh and Percy Liang. “Understanding black-box predictions via influence functions”. In: *arXiv preprint arXiv:1703.04730* (2017).

[134] Daphne Koller, Nir Friedman, and Francis Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[135] Aaron E Kornblith et al. “Predictability and Stability Testing to Assess Clinical Decision Instrument Performance for Children After Blunt Torso Trauma”. In: *medRxiv* (2022). DOI: 10.1101/2022.03.08.22270944. eprint: https://www.medrxiv.org/content/early/2022/03/08/2022.03.08.22270944.full.pdf. URL: https://www.medrxiv.org/content/early/2022/03/08/2022.03.08.22270944.

[136] Karl Kumbier et al. “Refining interaction search through signed iterative Random Forests”. In: *arXiv preprint arXiv:1810.07287* (2018).

[137] Nathan Kuppermann et al. “Identification of children at very low risk of clinically-important brain injuries after head trauma: a prospective cohort study”. In: *The Lancet* 374.9696 (2009), pp. 1160–1170.

[138] Jeff Larson et al. “How we analyzed the COMPAS recidivism algorithm”. In: *ProPublica (5 2016)* 9 (2016).

[139] Yann LeCun. “The MNIST database of handwritten digits”. In: *http://yann. lecun. com/exdb/mnist/* (1998).

[140] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep learning”. In: *nature* 521.7553 (2015), p. 436.

[141] Gregory Lee et al. “PyWavelets: A Python package for wavelet analysis”. In: *Journal of Open Source Software* 4.36 (2019), p. 1237.

[142] Tao Lei, Regina Barzilay, and Tommi Jaakkola. “Rationalizing neural predictions”. In: *arXiv preprint arXiv:1606.04155* (2016).

[143] Benjamin Letham et al. “Interpretable classifiers using rules and Bayesian analysis: Building a better stroke prediction mode”. In: *The Annals of Applied Statistics* 9.3 (2015), pp. 1350–1371. DOI: 10.1214/15-aoas848.

[144] Jiawei Li et al. “TNT: An Interpretable Tree-Network-Tree Learning Framework Using Knowledge Distillation”. In: *Entropy* 22.11 (2020), p. 1203.

[145] Jiwei Li, Will Monroe, and Dan Jurafsky. “Understanding neural networks through representation erasure”. In: *arXiv preprint arXiv:1612.08220* (2016).

[146] Xiao Li et al. “A stability-driven protocol for drug response interpretable prediction (staDRIP)”. In: *arXiv:2011.06593 [q-bio, stat]* (Nov. 16, 2020). arXiv: 2011.06593. URL: http://arxiv.org/abs/2011.06593.

[147] Yi Li and Nuno Vasconcelos. “REPAIR: Removing Representation Bias by Dataset Resampling”. In: *arXiv preprint arXiv:1904.07911* (2019).

[148] Zack Li et al. “Constraining neutrino mass with tomographic weak lensing peak counts”. In: *Physical Review D* 99.6, 063527 (2019), p. 063527. DOI: 10.1103/PhysRevD.99.063527. arXiv: 1810.01781 [astro-ph.CO].

[149] Chinghway Lim and Bin Yu. “Estimation stability with cross-validation (ESCV)”. In: *Journal of Computational and Graphical Statistics* 25.2 (2016), pp. 464–492.

[150] Jimmy Lin et al. “Generalized and scalable optimal sparse decision trees”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 6150–6160.

[151] Jing Lin and MJ Zuo. “Gearbox fault diagnosis using adaptive wavelet filter”. In: *Mechanical systems and signal processing* 17.6 (2003), pp. 1259–1269.

[152] Zachary C Lipton. “The mythos of model interpretability”. In: *arXiv preprint arXiv:1606.03490* (2016).

[153] Geert Litjens et al. “A survey on deep learning in medical image analysis”. In: *Medical image analysis* 42 (2017), pp. 60–88.

[154] Frederick Liu and Besim Avci. “Incorporating Priors with Feature Attribution on Text Classification”. In: *arXiv preprint arXiv:1906.08286* (2019).

[155] Jia Liu et al. “MassiveNuS: cosmological massive neutrino simulations”. In: *JCAP* 2018.3, 049 (2018), p. 049. DOI: 10.1088/1475-7516/2018/03/049. arXiv: 1711.10524 [astro-ph.CO].

[156] Tania Lombrozo. “The structure and function of explanations”. In: *Trends in cognitive sciences* 10.10 (2006), pp. 464–470.

[157] José Marcio Luna et al. “Building more accurate decision trees with the additive tree”. In: *Proceedings of the national academy of sciences* 116.40 (2019), pp. 19887–19893.

[158] Scott M Lundberg, Gabriel G Erion, and Su-In Lee. “Consistent Individualized Feature Attribution for Tree Ensembles”. In: *arXiv preprint arXiv:1802.03888* (2018).

[159] Scott M Lundberg and Su-In Lee. "A unified approach to interpreting model predictions". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4768–4777.

[160] Scott M Lundberg et al. "Explainable AI for trees: From local explanations to global understanding". In: *arXiv preprint arXiv:1905.04610* (2019).

[161] Prashant Mahajan et al. "Comparison of Clinician Suspicion Versus a Clinical Prediction Rule in Identifying Children at Risk for Intra-abdominal Injuries after Blunt Torso Trauma". In: *Academic Emergency Medicine* 22.9 (2015), pp. 1034–1041. DOI: `10.1111/acem.12739`.

[162] Stephane Mallat. "A theory for multiresolution signal decomposition: the wavelet representation". In: *IEEE transactions on pattern analysis and machine intelligence* 11.7 (1989), pp. 674–693.

[163] Stephane Mallat. "Multiresolution approximations and wavelet orthonormal bases of $L^2(R)$". In: *Transactions of the American mathematical society* 315.1 (1989), pp. 69–87.

[164] Stéphane Mallat. *A wavelet tour of signal processing, Third edition: The sparse way.* Academic Press, 2008.

[165] Stéphane Mallat. "Group invariant scattering". In: *Communications on Pure and Applied Mathematics* 65.10 (2012), pp. 1331–1398.

[166] J. R. Marin et al. "Variation in Computed Tomography Imaging for Pediatric Injury-Related Emergency Visits". In: *J Pediatr* 167.4 (Oct. 2015). edition: 2015/08/04, 897–904 e3. ISSN: 1097-6833 (Electronic) 0022-3476 (Linking). DOI: `10.1016/j.jpeds.2015.06.052`.

[167] Matteo Maturi et al. "An optimal filter for the detection of galaxy clusters through weak lensing". In: *Astronomy & Astrophysics* 442.3 (2005), pp. 851–860.

[168] Wes McKinney et al. "Data structures for statistical computing in python". In: *Proceedings of the 9th Python in Science Conference*. Vol. 445. Austin, TX. 2010, pp. 51–56.

[169] Harvey T McMahon and Emmanuel Boucrot. "Molecular mechanism and physiological functions of clathrin-mediated endocytosis". In: *Nature reviews Molecular cell biology* 12.8 (2011), p. 517.

[170] J. A. Meltzer et al. "Association of Whole-Body Computed Tomography With Mortality Risk in Children With Blunt Trauma". In: *JAMA Pediatr* 172.6 (June 1, 2018). edition: 2018/04/10, pp. 542–549. ISSN: 2168-6211 (Electronic) 2168-6203 (Linking). DOI: `10.1001/jamapediatrics.2018.0109`.

[171] Lucas Mentch and Siyu Zhou. "Randomization as regularization: a degrees of freedom explanation for random forest success". In: *arXiv preprint arXiv:1911.00190* (2019).

[172] Robert Messenger and Lewis Mandell. “A modal search technique for predictive nominal scale multivariate analysis”. In: *Journal of the American Statistical Association* 67.340 (1972), pp. 768–772.

[173] Yves Meyer. *Wavelets and Operators: Volume 1*. 37. Cambridge university press, 1992.

[174] Diana L. Miglioretti et al. “The use of computed tomography in pediatrics and the associated radiation exposure and estimated cancer risk”. eng. In: *JAMA pediatrics* 167.8 (Aug. 1, 2013). PMID: 23754213 PMCID: PMC3936795, pp. 700–707. ISSN: 2168-6211. DOI: `10.1001/jamapediatrics.2013.311`.

[175] Arnaud Mignan and Marco Broccardo. “One neuron versus deep learning in aftershock prediction”. In: *Nature* 574.7776 (2019), E1–E3.

[176] Masahiro Mitsuhara et al. “Embedding Human Knowledge in Deep Neural Network via Attention Map”. In: *arXiv preprint arXiv:1905.03540* (2019).

[177] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2019.

[178] Christoph Molnar. *Interpretable machine learning. A Guide for Making Black Box Models Explainable*. Lulu. com, 2020. URL: `https://christophm.github.io/interpretable-ml-book/`.

[179] Seyed Mohsen Moosavi Dezfooli, Alhussein Fawzi, and Pascal Frossard. “Deepfool: a simple and accurate method to fool deep neural networks”. In: *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. EPFL-CONF-218057. 2016.

[180] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. “Deepdream-a code example for visualizing neural networks”. In: *Google Research* 2 (2015), p. 5.

[181] Danielle Morel, Chandan Singh, and William B Levy. “Linearization of excitatory synaptic integration at no extra cost”. In: *Journal of Computational Neuroscience* 44.2 (2018), pp. 173–188.

[182] James N Morgan and John A Sonquist. “Problems in the analysis of survey data, and a proposal”. In: *Journal of the American Statistical Association* 58.302 (1963), pp. 415–434.

[183] Philipp Moritz et al. “Ray: A distributed framework for emerging AI applications”. In: *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 2018, pp. 561–577.

[184] W James Murdoch, Peter J Liu, and Bin Yu. “Beyond word importance: Contextual decomposition to extract interactions from LSTMs”. In: *arXiv preprint arXiv:1801.05453* (2018).

[185] W James Murdoch and Arthur Szlam. “Automatic rule extraction from long short term memory networks”. In: *arXiv preprint arXiv:1702.02540* (2017).

[186] W. James Murdoch et al. "Definitions, methods, and applications in interpretable machine learning". In: *Proceedings of the National Academy of Sciences of the United States of America* 116.44 (2019), pp. 22071–22080. DOI: `10.1073/pnas.1900654116`.

[187] W. James Murdoch et al. "Interpretable machine learning: definitions, methods, and applications". In: *arXiv preprint arXiv:1901.04592* (2019).

[188] Warwick J Nash et al. "The population biology of abalone (haliotis species) in tasmania. i. blacklip abalone (h. rubra) from the north coast and islands of bass strait". In: *Sea Fisheries Division, Technical Report* 48 (1994), p411.

[189] Quang P Nguyen and Kara W Schechtman. "Confirmed and Probable COVID-19 Deaths, Counted Two Ways". In: *The COVID Tracking Project* (2020). URL: `https://covidtracking.com/blog/confirmed-and-probable-covid-19-deaths-counted-two-ways`.

[190] Weili Nie, Yang Zhang, and Ankit Patel. "A theoretical explanation for perplexing behaviors of backpropagation-based visualizations". In: *arXiv preprint arXiv:1805.07039* (2018).

[191] Beau Norgeot et al. "Minimum information about clinical artificial intelligence modeling: the MI-CLAIM checklist". eng. In: *Nature Medicine* 26.9 (Sept. 2020). PMID: 32908275 PMCID: PMC7538196, pp. 1320–1324. ISSN: 1546-170X. DOI: `10.1038/s41591-020-1041-y`.

[192] Harvard University Office of Institutional Research. "Exhibit 157: Demographics of Harvard College Applicants". In: *http://samv91khoyt2i553a2t1s05i-wpengine.netdna-ssl.com/wp-content/uploads/2018/06/Doc-421-157-May-30-2013-Report.pdf* (2018), pp. 8–9.

[193] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. "Feature Visualization". In: *Distill* 2.11 (2017), e7.

[194] Julian D Olden, Michael K Joy, and Russell G Death. "An accurate comparison of methods for quantifying variable importance in artificial neural networks using simulated data". In: *Ecological Modelling* 178.3-4 (2004), pp. 389–397.

[195] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[196] Randal S Olson et al. "Data-driven advice for applying machine learning to bioinformatics problems". In: *Biocomputing 2018: Proceedings of the Pacific Symposium.* World Scientific. 2018, pp. 192–203.

[197] Joy D Osofsky. "The effects of exposure to violence on young children (1995)." In: *Carnegie Corporation of New York Task Force on the Needs of Young Children; An earlier version of this article was presented as a position paper for the aforementioned corporation.* (1997).

[198] Utku Ozbulak. *PyTorch CNN Visualizations*. `https://github.com/utkuozbulak/pytorch-cnn-visualizations`. 2019.

[199] R Kelley Pace and Ronald Barry. "Sparse spatial autoregressions". In: *Statistics & Probability Letters* 33.3 (1997), pp. 291–297.

[200] Giulia Pagallo and David Haussler. "Boolean feature discovery in empirical learning". In: *Machine learning* 5.1 (1990), pp. 71–99.

[201] Nicolas Papernot and Patrick McDaniel. "Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning". In: *arXiv preprint arXiv:1803.04765* (2018).

[202] Nicolas Papernot et al. "The limitations of deep learning in adversarial settings". In: *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*. IEEE. 2016, pp. 372–387.

[203] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[204] Yagyensh Chandra Pati, Ramin Rezaiifar, and Perinkulam Sambamurthy Krishnaprasad. "Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition". In: *Conf. Rec. 27th Asilomar Conf. Signals, Syst. Comput.* IEEE. 1993, pp. 40–44.

[205] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *the Journal of machine Learning research* 12.Oct (2011), pp. 2825–2830. URL: `http://jmlr.org/papers/v12/pedregosa11a.html`.

[206] Christopher Pennell et al. "Risk assessment for intraabdominal injury following blunt trauma in children". In: *Journal of Trauma and Acute Care Surgery* Publish Ah (2020). ISSN: 0000000000. DOI: `10.1097/ta.0000000000002717`.

[207] Fernando Pérez and Brian E Granger. "IPython: a system for interactive scientific computing". In: *Computing in Science & Engineering* 9.3 (2007).

[208] Harold Pimentel, Zhiyue Hu, and Haiyan Huang. "Biclustering by sparse canonical correlation analysis". In: *Quantitative Biology* 6.1 (2018), pp. 56–67.

[209] J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

[210] J. Ross Quinlan. "Induction of decision trees". In: *Machine learning* 1.1 (1986), pp. 81–106.

[211] Juan Ramos et al. "Using tf-idf to determine word relevance in document queries". In: *Proceedings of the first instructional conference on machine learning*. Vol. 242. 2003, pp. 133–142.

[212] Garvesh Raskutti, Martin J Wainwright, and Bin Yu. "Minimax-Optimal Rates For Sparse Additive Models Over Kernel Classes Via Convex Programming." In: *Journal of Machine Learning Research* 13.2 (2012).

[213] Jonas Rauber, Wieland Brendel, and Matthias Bethge. "Foolbox v0. 8.0: A python toolbox to benchmark the robustness of machine learning models". In: *arXiv preprint arXiv:1707.04131* (2017).

[214] Stephen J Read and Amy Marcus-Newhall. "Explanatory coherence in social explanations: A parallel distributed processing account." In: *Journal of Personality and Social Psychology* 65.3 (1993), p. 429.

[215] Daniel Recoskie. "Learning Sparse Orthogonal Wavelet Filters". In: (2018).

[216] Daniel Recoskie and Richard Mann. "Learning sparse wavelet representations". In: *arXiv preprint arXiv:1802.02961* (2018).

[217] Brendan M. Reilly and Arthur T. Evans. "Translating clinical research into clinical practice: impact of using prediction rules to make decisions". eng. In: *Annals of Internal Medicine* 144.3 (Feb. 7, 2006). PMID: 16461965, pp. 201–209. ISSN: 1539-3704. DOI: `10.7326/0003-4819-144-3-200602070-00009`.

[218] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should i trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 1135–1144.

[219] Dezső Ribli, Bálint Ármin Pataki, and István Csabai. "An improved cosmological parameter inference scheme motivated by deep learning". In: *Nature Astronomy* 3.1 (2019), p. 93.

[220] Dezső Ribli et al. "Weak lensing cosmology with convolutional neural networks on noisy data". In: *Monthly Notices of the Royal Astronomical Society* 490.2 (2019), pp. 1843–1860. DOI: `10.1093/mnras/stz2610`. arXiv: `1902.03663 [astro-ph.CO]`.

[221] Laura Rieger and Lars Kai Hansen. "Aggregating explainability methods for neural networks stabilizes explanations". In: *arXiv preprint arXiv:1903.00519* (2019).

[222] Laura Rieger et al. "Interpretations are useful: penalizing explanations to align neural networks with prior knowledge". In: *arXiv preprint arXiv:1909.13584* (2019).

[223] Laura Rieger et al. "Interpretations are useful: penalizing explanations to align neural networks with prior knowledge". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 8116–8126.

[224] Richard D. Riley et al. "External validation of clinical prediction models using big datasets from e-health records or IPD meta-analysis: opportunities and challenges". eng. In: *BMJ (Clinical research ed.)* 353 (June 22, 2016). PMID: 27334381 PMCID: PMC4916924, p. i3140. ISSN: 1756-1833. DOI: `10.1136/bmj.i3140`.

[225] Anna W Roe et al. "Toward a unified theory of visual area V4". In: *Neuron* 74.1 (2012), pp. 12–29.

[226] Anna Rohrbach et al. "Grounding of textual phrases in images by reconstruction". In: *European Conference on Computer Vision*. Springer. 2016, pp. 817–834.

[227] Joseph D Romano et al. "PMLB v1. 0: an open source dataset collection for benchmarking machine learning methods". In: *arXiv preprint arXiv:2012.00058* (2020).

[228] Andrew Slavin Ross and Finale Doshi-Velez. "Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients". In: *Thirty-second AAAI conference on artificial intelligence*. 2018.

[229] Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. "Right for the right reasons: Training differentiable models by constraining their explanations". In: *arXiv preprint arXiv:1703.03717* (2017).

[230] Cynthia Rudin. "Please Stop Explaining Black Box Models for High Stakes Decisions". In: *arXiv preprint arXiv:1811.10154* (2018).

[231] Cynthia Rudin et al. "Interpretable Machine Learning: Fundamental Principles and 10 Grand Challenges". In: *arXiv preprint arXiv:2103.11251* (2021).

[232] Kaspar Rufibach. "Use of Brier score to assess binary predictions". English. In: *Journal of Clinical Epidemiology* 63.8 (Aug. 1, 2010). publisher: Elsevier PMID: 20189763, pp. 938–939. ISSN: 0895-4356, 1878-5921. DOI: 10.1016/j.jclinepi.2009.11.009.

[233] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[234] Phil Sallee and Bruno Olshausen. "Learning sparse multiscale image representations". In: *Advances in neural information processing systems* 15 (2002), pp. 1351–1358.

[235] Andrea Saltelli. "Sensitivity analysis for importance assessment". In: *Risk analysis* 22.3 (2002), pp. 579–590. DOI: 10.1111/0272-4332.00040.

[236] Fabian Schmidt and Eduardo Rozo. "Weak-lensing Peak Finding: Estimators, Filters, and Biases". In: *The Astrophysical Journal* 735.2 (2011), p. 119.

[237] Patrick Schramowski et al. "Making deep neural networks right for the right scientific reasons by interacting with their explanations". In: *Nature Machine Intelligence* 2.8 (2020), pp. 476–486.

[238] Gerald DT Schuller et al. "Perceptual audio coding using adaptive pre-and post-filters and lossless compression". In: *IEEE Transactions on Speech and Audio Processing* 10.6 (2002), pp. 379–390.

[239] Gideon Schwarz. "Estimating the dimension of a model". In: *The annals of statistics* (1978), pp. 461–464.

[240] Ramprasaath R Selvaraju et al. "Grad-cam: Visual explanations from deep networks via gradient-based localization". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[241] Tao Shi et al. "Daytime arctic cloud detection based on multi-angle satellite data with case studies". In: *Journal of the American Statistical Association* 103.482 (2008), pp. 584–593.

[242] Avanti Shrikumar et al. "Not just a black box: Learning important features through propagating activation differences". In: *arXiv preprint arXiv:1605.01713* (2016).

[243] Vincent G Sigillito et al. "Classification of radar returns from the ionosphere using neural networks". In: *Johns Hopkins APL Technical Digest* 10.3 (1989), pp. 262–266.

[244] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[245] Chandan Singh, Guha Balakrishnan, and Pietro Perona. "Matched sample selection with GANs for mitigating attribute confounding". In: *arXiv preprint arXiv:2103.13455* (2021).

[246] Chandan Singh, Wooseok Ha, and Bin Yu. "Interpreting and improving deep-learning models with reality checks". In: *arXiv preprint arXiv:2108.06847* (2021).

[247] Chandan Singh and William B Levy. "A consensus layer V pyramidal neuron can sustain interpulse-interval coding". In: *PloS one* 12.7 (2017), e0180839.

[248] Chandan Singh, W James Murdoch, and Bin Yu. "Hierarchical Interpretations for Neural Network Predictions". en. In: *International Conference on Learning Representations* (2019), p. 26. URL: `https://openreview.net/forum?id=SkEqro0ctQ`.

[249] Chandan Singh, W. James Murdoch, and Bin Yu. "Hierarchical interpretations for neural network predictions". In: *International Conference on Learning Representations*. 2019. URL: `https://openreview.net/forum?id=SkEqro0ctQ`.

[250] Chandan Singh, Beilun Wang, and Yanjun Qi. "A constrained, weighted-l1 minimization approach for joint discovery of heterogeneous neural connectivity graphs". In: *arXiv preprint arXiv:1709.04090* (2017).

[251] Chandan Singh et al. "imodels: a python package for fitting interpretable models". In: *Journal of Open Source Software* 6.61 (2021), p. 3192. DOI: `10.21105/joss.03192`. URL: `https://doi.org/10.21105/joss.03192`.

[252] Chandan Singh et al. *Transformation Importance with Applications to Cosmology*. 2020. arXiv: `2003.01926 [stat.ML]`.

[253] Jack W Smith et al. "Using the ADAP learning algorithm to forecast the onset of diabetes mellitus". In: *Proceedings of the annual symposium on computer application in medical care*. American Medical Informatics Association. 1988, p. 261.

[254] Richard Socher et al. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on empirical methods in natural language processing*. 2013, pp. 1631–1642.

[255] Jost Tobias Springenberg et al. "Striving for simplicity: The all convolutional net". In: *arXiv preprint arXiv:1412.6806* (2014).

[256] Henry J Steadman et al. "A classification tree approach to the development of actuarial violence risk assessment tools". In: *Law and Human Behavior* 24.1 (2000), pp. 83–100.

[257] I. G. Stiell and G. A. Wells. “Methodologic standards for the development of clinical decision rules in emergency medicine”. eng. In: *Annals of Emergency Medicine* 33.4 (Apr. 1999). PMID: 10092723, pp. 437–447. ISSN: 0196-0644. DOI: 10.1016/s0196-0644(99)70309-4.

[258] Erin K Stokes et al. “Coronavirus disease 2019 case surveillance—United States, January 22–May 30, 2020”. In: *Morbidity and Mortality Weekly Report* 69.24 (2020), p. 759.

[259] Christian J. Streck et al. “Identifying Children at Very Low Risk for Blunt Intra-Abdominal Injury in Whom CT of the Abdomen Can Be Avoided Safely”. In: vol. 224. issue: 4. Journal of the American College of Surgeons. 2017, 449–458.e3. DOI: 10.1016/j.jamcollsurg.2016.12.041. URL: http://linkinghub.elsevier.com/retrieve/pii/S1072751517300376.

[260] Hendrik Strobelt et al. “Visual analysis of hidden state dynamics in recurrent neural networks”. In: *CoRR, abs/1606.07461* (2016).

[261] Carolin Strobl et al. “Conditional variable importance for random forests”. In: *BMC bioinformatics* 9.1 (2008), p. 307.

[262] Julia Strout, Ye Zhang, and Raymond J Mooney. “Do Human Rationales Improve Machine Explanations?” In: *arXiv preprint arXiv:1905.13714* (2019).

[263] Jiamei Sun et al. “Explain and improve: LRP-inference fine-tuning for image captioning models”. In: *Information Fusion* 77 (2022), pp. 233–246.

[264] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic attribution for deep networks”. In: *ICML* (2017).

[265] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *arXiv preprint arXiv:1312.6199* (2013).

[266] Cheng Tai and E Weinan. “Multiscale adaptive representation of signals: I. the basic framework”. In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 4875–4912.

[267] Sarah Tan et al. “Learning global additive explanations for neural nets using model distillation”. In: *arXiv preprint arXiv:1801.08640* (2018).

[268] Yan Shuo Tan, Abhineet Agarwal, and Bin Yu. “A cautionary tale on fitting decision trees to data from additive models: generalization lower bounds”. In: *arXiv preprint arXiv:2110.09626* (2021).

[269] Yan Shuo Tan et al. “Fast Interpretable Greedy-Tree Sums (FIGS)”. en. In: *arXiv:2201.11931 [cs, stat]* (Jan. 27, 2022). arXiv: 2201.11931. URL: http://arxiv.org/abs/2201.11931.

[270] RStudio Team. *RStudio: Integrated Development Environment for R*. RStudio, Inc. Boston, MA, 2016. URL: http://www.rstudio.com/.

[271] Robert Tibshirani. “Regression shrinkage and selection via the lasso”. In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), pp. 267–288.

[272] D.B. Toll et al. “Validation, updating and impact of clinical prediction rules: A review”. en. In: *Journal of Clinical Epidemiology* 61.11 (Nov. 2008), pp. 1085–1094. ISSN: 08954356. DOI: `10.1016/j.jclinepi.2008.04.008`.

[273] Michael Tsang, Dehua Cheng, and Yan Liu. “Detecting statistical interactions from neural network weights”. In: *arXiv preprint arXiv:1705.04977* (2017).

[274] Michael Tsang et al. “Can I trust you more? Model-Agnostic Hierarchical Explanations”. In: *arXiv preprint arXiv:1812.04801* (2018).

[275] Berk Ustun and Cynthia Rudin. “Supersparse linear integer models for optimized medical scoring systems”. In: *Machine Learning* 102.3 (2016), pp. 349–391. DOI: `10.1007/s10994-015-5528-6`.

[276] Adam M. Vogel et al. “Variability in the evalution of pediatric blunt abdominal trauma”. eng. In: *Pediatric Surgery International* 35.4 (Apr. 2019). PMID: 30426222, pp. 479–485. ISSN: 1437-9813. DOI: `10.1007/s00383-018-4417-z`.

[277] Vladimir Vovk, Alex Gammerman, and Glenn Shafer. *Algorithmic learning in a random world*. Springer Science & Business Media, 2005.

[278] Mai-Anh T Vu et al. “A shared vision for machine learning in neuroscience”. In: *Journal of Neuroscience* (2018), pp. 0508–17.

[279] Haohan Wang et al. “Learning Robust Representations by Projecting Superficial Statistics Out”. In: *arXiv preprint arXiv:1903.06256* (2019).

[280] Tong Wang. “Gaining Free or Low-Cost Interpretability with Interpretable Partial Substitute”. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by Kamalika Chaudhuri and Ruslan Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 6505–6514. URL: `http://proceedings.mlr.press/v97/wang19a.html`.

[281] Michael Waskom et al. “Seaborn: statistical data visualization”. In: *URL: https://seaborn. pydata. org/(visited on 2017-05-15)* (2014).

[282] Donglai Wei et al. “Understanding intra-class knowledge inside CNN”. In: *arXiv preprint arXiv:1507.02379* (2015).

[283] Hadley Wickham. *ggplot2: elegant graphics for data analysis*. Springer, 2016.

[284] Hadley Wickham. *tidyverse: Easily Install and Load the 'Tidyverse'*. 2017. URL: `https://CRAN.R-project.org/package=tidyverse`.

[285] Julia K. Winkler et al. "Association Between Surgical Skin Markings in Dermoscopic Images and Diagnostic Performance of a Deep Learning Convolutional Neural Network for Melanoma RecognitionSurgical Skin Markings in Dermoscopic Images and Deep Learning Convolutional Neural Network Recognition of MelanomaSurgical Skin Markings in Dermoscopic Images and Deep Learning Convolutional Neural Network Recognition of Melanoma". In: *JAMA Dermatology* (Aug. 2019). ISSN: 2168-6068. DOI: `10.1001/jamadermatol.2019.1735`. eprint: `https://jamanetwork.com/journals/jamadermatology/articlepdf/2740808/jamadermatology\_winkler\_2019\_oi\_190038.pdf`. URL: `https://doi.org/10.1001/jamadermatol.2019.1735`.

[286] Marvin N Wright, Andreas Ziegler, et al. "ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R". In: *Journal of Statistical Software* 77.i01 (2017).

[287] Siqi Wu et al. "Stability-driven nonnegative matrix factorization to interpret spatial gene expression and build local gene networks". In: *Proceedings of the National Academy of Sciences* 113.16 (2016), pp. 4290–4295.

[288] I-Cheng Yeh and Che-hui Lien. "The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients". In: *Expert Systems with Applications* 36.2 (2009), pp. 2473–2480.

[289] Kenneth Yen et al. "Interobserver agreement in the clinical assessment of children with blunt abdominal trauma". In: *Academic Emergency Medicine* 20.5 (2013), pp. 426–432. DOI: `10.1111/acem.12132`.

[290] Seul-Ki Yeom et al. "Pruning by explaining: A novel criterion for deep neural network pruning". In: *Pattern Recognition* 115 (2021), p. 107899.

[291] Jason Yosinski et al. "Understanding neural networks through deep visualization". In: *arXiv preprint arXiv:1506.06579* (2015).

[292] Bin Yu. "Stability". In: *Bernoulli* 19.4 (2013), pp. 1484–1500.

[293] Bin Yu and Karl Kumbier. "Veridical data science". In: *Proceedings of the National Academy of Sciences of the United States of America* 117.8 (2020), pp. 3920–3929. DOI: `10.1073/pnas.1901326117`.

[294] Bin Yu and Chandan Singh. "Seven Principles for Rapid-Response Data Science: Lessons Learned from Covid-19 Forecasting". In: *arXiv preprint arXiv:2108.08445* (2021).

[295] Matei Zaharia et al. "Accelerating the machine learning lifecycle with MLflow". In: *IEEE Data Eng. Bull.* 41.4 (2018), pp. 39–45. DOI: `10.1145/3399579.3399867`.

[296] Omar Zaidan, Jason Eisner, and Christine Piatko. "Using "annotator rationales" to improve machine learning for text categorization". In: *Proceedings of NAACL HLT*. 2007, pp. 260–267.

[297] Matthew D Zeiler and Rob Fergus. “Visualizing and understanding convolutional networks”. In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[298] Jun Zhang et al. “Wavelet neural networks for function learning”. In: *IEEE transactions on Signal Processing* 43.6 (1995), pp. 1485–1497.

[299] Qinghua Zhang and Albert Benveniste. “Wavelet networks”. In: *IEEE transactions on Neural Networks* 3.6 (1992), pp. 889–898.

[300] Quanshi Zhang et al. “Interpreting CNN knowledge via an Explanatory Graph”. In: *arXiv preprint arXiv:1708.01785* (2017).

[301] Tianyuan Zhang and Zhanxing Zhu. “Interpreting Adversarially Trained Convolutional Neural Networks”. In: *arXiv preprint arXiv:1905.09797* (2019).

[302] Esra Zihni et al. “Opening the black box of artificial intelligence for clinical decision support: A study predicting stroke outcome”. In: *PLoS ONE* 15.4 (2020), pp. 1–15. ISSN: 1111111111. DOI: `10.1371/journal.pone.0231166`.

[303] Luisa M Zintgraf et al. “Visualizing deep neural network decisions: Prediction difference analysis”. In: *arXiv preprint arXiv:1702.04595* (2017).

[304] Joseph J Zorc, James M Chamberlain, and Lalit Bajaj. “Machine Learning at the Clinical Bedside-The Ghost in the Machine.” In: *JAMA pediatrics* 162.1 (May 2019), W1–W73. DOI: `10.1001/jamapediatrics.2019.1075`.