Interpreting and improving deep-learning models with reality checks *

Chandan Singh[1,*], Wooseok Ha[1,*], and Bin Yu[1]

[1]University of California, Berkeley, Berkeley CA, USA
{cs1, haywse, binyu}@berkeley.edu
* Equal contribution

Abstract. Recent deep-learning models have achieved impressive predictive performance by learning complex functions of many variables, often at the cost of interpretability. This chapter covers recent work aiming to interpret models by attributing importance to features and feature groups for a single prediction. Importantly, the proposed attributions assign importance to interactions between features, in addition to features in isolation. These attributions are shown to yield insights across real-world domains, including bio-imaging, cosmology image and natural-language processing. We then show how these attributions can be used to directly improve the generalization of a neural network or to distill it into a simple model. Throughout the chapter, we emphasize the use of reality checks to scrutinize the proposed interpretation techniques.[1]

Keywords: Interpretability · Interactions · Feature importance · Neural network · Distillation

1 Interpretability: for what and for whom?

Deep neural networks (DNNs) 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, DNNs 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-

* We gratefully acknowledge partial support from NSF TRIPODS Grant 1740855, DMS-1613002, 1953191, 2015341, IIS 1741340, ONR grant N00014-17-1-2176, the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370, NSF grant 2023505 on Collaborative Research: Foundations of Data Science Institute (FODSI), the NSF and the Simons Foundation for the Collaboration on the Theoretical Foundations of Deep Learning through awards DMS-2031883 and 814639, and a grant from the Weill Neurohub.

[1] Code for all methods mentioned in this chapter is available at github.com/csinva and github.com/Yu-Group. All methods are implemented in PyTorch [1].

learning model concerning relationships either contained in data or learned by the model" [2].[2]

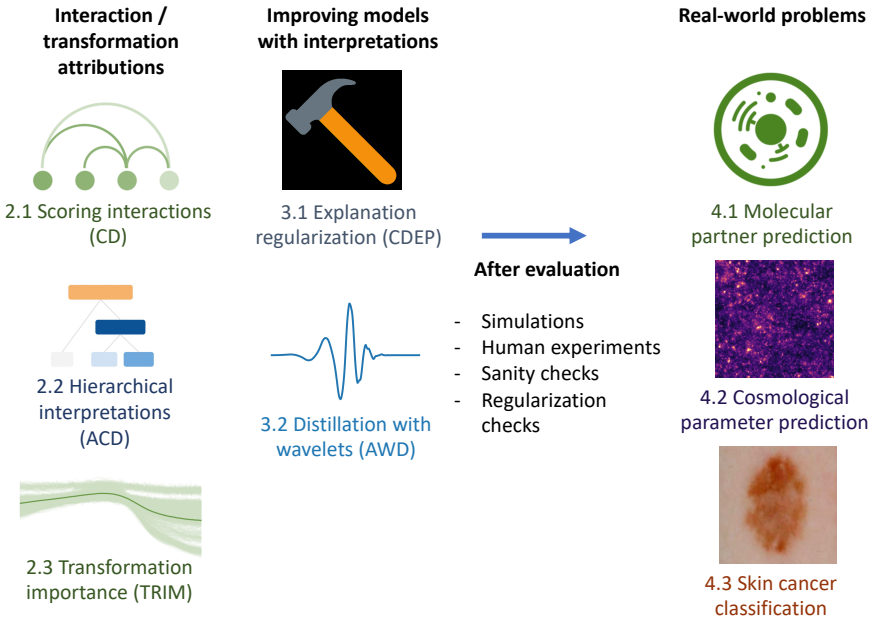

Fig. 1: Chapter overview. We begin by defining interpretability and some of its desiderata, following [2] (Sec 1). We proceed to overview different methods for computing interpretations for interactions/transformations (Sec 2), including for scoring interactions [3], generating hierarchical interpretations [4], and calculating importances for transformations of features [5]. Next, we show how these interpretations can be used to improve models (Sec 3), including by directly regularizing interpretations [6] and distilling a model through interpretations [7]. Finally, we show how these interpretations can be adapted to real-world applications (Sec 4), including molecular partner prediction, cosmological parameter prediction, and skin-cancer classification.

Here, we view knowledge as being *relevant* if it provides insight for a particular audience into a chosen problem. This definition highlights that interpretability is poorly specified without the context of a particular audience and problem, and should be evaluated with the context in mind. This definition also implies that interpretable ML provides correct information (i.e. *knowledge*), and we use the term interpretation, assuming that the interpretation technique at

[2] We include different headings such as explainable AI (XAI), intelligible ML and transparent ML under this definition.

hand has passed some form of *reality check* (i.e. it faithfully captures some notion of reality).

Interpretations have found uses both in their own right, e.g. medicine [8], policy-making [9], and science [10,11], as well as in auditing predictions themselves in response to issues such as regulatory pressure [12] and fairness [13]. 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 [14]. However, this increasing role, along with the explosion in proposed interpretation techniques [2,7,15–20] has raised considerable concerns about the use of interpretation methods in practice [21,22]. 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 [2] among many definitions [23–25], then discuss some methods for critically evaluating interpretations.

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 cover the PDR framework [2], consisting of three desiderata that should be used to select interpretation methods for a particular problem: predictive accuracy, descriptive accuracy, and relevancy. *Predictive accuracy* measures the ability of a model to capture underlying relationships in the data (and generally includes different measures of a model's quality of fit)—this can be seen as the most common form of reality check. In contrast, *descriptive accuracy* measures how well one can approximate what the model has learned using an interpretation method. Descriptive accuracy measures errors during the post-hoc analysis stage of modeling, when interpretations methods are used to analyze a fitted model. 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. *Relevancy* guides which interpretation to select based on the context of the problem, often playing a key role in determining the trade-off between predictive and descriptive accuracy; however, predictive accuracy and relevancy are not always a trade-off and the examples are shown in Sec 4.

Evaluating interpretations and additional reality checks Techniques striving for interpretations can provide a large amount of fine-grained information, often not just for individual features but also for feature groups [3,4]. As such, it is important to ensure that this added information correctly reflects a model (i.e. has high descriptive accuracy), and can be useful in practice. This is challenging in general, but there are some promising directions. One direction, often used in statistical research including causal inference, uses simulation studies to evaluate interpretations. In this setting, a researcher defines a simple generative process, generates a large amount of data from that process, and trains their statistical

or ML model on that data. Assuming a proper simulation setup, a sufficiently relevant and powerful model to recover the generative process, and sufficiently large training data, the trained model should achieve near-perfect generalization accuracy. The practitioner then measures whether their interpretations recover aspects of the original generative process. If the simulation captures the reality well, then it can be viewed as a weaker form of reality check.

Going a step further, interpretations can be tested by gathering new data in followup experiments or observations for retrospective validation. Another direction, which this chapter also focuses on, is to demonstrate the interpretations through domain knowledge which is relevant to a particular domain/audience. To do so, we closely collaborate with domain experts and showcase how interpretations can inform relevant knowledge in fundamental problems in cosmology and molecular-partner prediction. We highlight the use of reality checks to evaluate each proposed method in the chapter.

Chapter overview 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 [26–29], decomposition-based [30–32] and others [33–36], with many similarities among the methods [37, 38]. 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.

Here, we cover a line of work that aims to identify, attribute importance, and utilize interactions in neural networks for interpretation. We then explore how these attributions can be used to help improve the performance of DNNs. Despite their strong predictive performance, DNNs sometimes latch onto spurious correlations caused by dataset bias or overfitting [39]. As a result, DNNs often exploit bias regarding gender, race, and other sensitive attributes present in training datasets [40–42]. Moreover, DNNs are extremely computationally intensive and difficult to audit.

Fig 1 shows an overview of this chapter. We first overview different methods for computing interpretations (Sec 2), including for scoring interactions [3], generating hierarchical interpretations [4], and calculating importances for transformations of features [5]. Next, we show how these interpretations can be used to improve models (Sec 3), including by directly regularizing interpretations [6] and distilling a model through interpretations [7]. Finally, we show how these interpretations can be adapted to real-world problems (Sec 4), including molecular partner prediction, cosmological parameter prediction, and skin-cancer classification.

2 Computing interpretations for feature interactions and transformations

This section reviews three recent methods developed to extract the interactions between features that an (already trained) DNN has learned. First, Sec 2.1 shows how to compute importance scores for groups of features via contextual decomposition (CD), a method which works with LSTMs [3] and arbitrary DNNs, such as CNNs [4]. Next, Sec 2.2 covers agglomerative contextual decomposition (ACD), where a group-level importance measure, in this case CD, is used as a joining metric in an agglomerative clustering procedure. Finally, Sec 2.3 covers transformation importance (TRIM), which allows for computing scores for interactions on transformations of a model's input. Other methods have been recently developed for understanding model interactions with varying degrees of computational cost and faithfulness to the trained model [17, 43–47].

2.1 Contextual Decomposition (CD) importance scores for general DNNs

Contextual decomposition breaks up the forward pass of a neural network in order to find an importance score of some subset of the inputs for a particular prediction. For a given DNN $f(x)$, its output is represented as a SoftMax operation applied to logits $g(x)$. These logits, in turn, are the composition of L layers g_i, $i = 1, \ldots, L$, such as convolutional operations or ReLU non-linearities:

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

Given a group of features $\{x_j\}_{j \in S}$, the 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 $\beta(x)$.

$$g^{CD}(x) = (\beta(x), \gamma(x)), \tag{2}$$

$$\beta(x) + \gamma(x) = g(x). \tag{3}$$

Computing the CD decomposition for $g(x)$, requires layer-wise CD decompositions $g_i^{CD}(x) = (\beta_i, \gamma_i)$ for each layer $g_i(x)$, where $g_i(x)$ represents the vector of neural activations at the i-th layer. 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. Maintaining the decomposition requires $\beta_i + \gamma_i = g_i(x)$ for each i, the CD scores for the full network are computed by composing these decompositions.

$$g^{CD}(x) = g_L^{CD}(g_{L-1}^{CD}(...(g_2^{CD}(g_1^{CD}(x)))))). \tag{4}$$

Note that the above equation shows the CD algorithm g^{CD} takes as input a vector x and for each layer it outputs the pair of vector scores $g_i^{CD}(x) = (\beta_i, \gamma_i)$;

and the final output is given by a pair of numbers $g^{CD}(x) = (\beta(x), \gamma(x))$ such that the sum $\beta(x) + \gamma(x)$ equals the logits $g(x)$.

The initial CD work [3] introduced decompositions g_i^{CD} for layers used in LSTMs and the followup work [4] for layers used in CNNs and more generic deep architectures. Below, we give example decompositions for some commonly used layers, such as convolutional layer, linear layer, or ReLU activation.

When g_i is a convolutional or fully connected layer, the layer operation consists of a weight matrix W and a bias vector 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. The bias is partitioned 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.

$$\beta_i = W\beta_{i-1} + \frac{|W\beta_{i-1}|}{|W\beta_{i-1}| + |W\gamma_{i-1}|} \cdot b; \tag{5}$$

$$\gamma_i = W\gamma_{i-1} + \frac{|W\gamma_{i-1}|}{|W\beta_{i-1}| + |W\gamma_{i-1}|} \cdot b. \tag{6}$$

Next, for the ReLU activation function,[3] importance score β_i is computed as the activation of β_{i-1} alone and then update γ_i by subtracting this from the total activation.

$$\beta_i = \mathrm{ReLU}(\beta_{i-1}); \tag{7}$$

$$\gamma_i = \mathrm{ReLU}(\beta_{i-1} + \gamma_{i-1}) - \mathrm{ReLU}(\beta_{i-1}). \tag{8}$$

For a dropout layer, dropout is simply applied to β_{i-1} and γ_{i-1} individually. Computationally, a CD call is comparable to a forward pass through the network f.

Reality check: 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. Table 1 shows the ACD-top-scoring phrases of different lengths for an LSTM trained on SST (here the phrases are considered from all sentences in the SST's validation set). These phrases were extracted by running ACD separately on each sample in validation set. The score of each phrase was then computed by averaging over the score it received in each occurrence in an 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. The paper [3] also shows that CD properly captures negation interactions for phrases.

[3] See [3, Sec. 3.2.2] for other activation functions such as sigmoid or hyperbolic tangent.

Table 1: Top-scoring phrases of different lengths extracted by CD on SST's validation set. The positive/negative phrases identified by CD are all indeed positive/negative.

Length	Positive	Negative
1	pleasurable, glorious	nowhere, grotesque, sleep
3	amazing accomplishment., great fun.	bleak and desperate, conspicuously lacks.
5	a pretty amazing accomplishment.	ultimately a pointless endeavour.

2.2 Agglomerative Contextual Decomposition (ACD)

Next, we cover agglomerative contextual decomposition (ACD), 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 designed to identify clusters of features that the DNN learned are predictive. Throughout this subsection, we use the term CD interaction score between two groups of features to mean the difference between the scores of the combined group and the original groups.

Given the generalized CD scores introduced above, we now introduce the clustering procedure used to produce ACD interpretations. At a high level, this method is equivalent to agglomerative hierarchical clustering, where the CD interaction score 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 highest interaction scores provided by CD. The displayed ACD interpretation is the hierarchy, along with the CD importance score at each node.

The clustering procedure proceeds as follows. 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) 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 the original groups.

Reality check: human experiment Human experiments show that ACD allows users to better reason about the accuracy of DNNs. Each subject was asked to fill out a survey asking whether, using ACD, they could identify the more accurate of two models across three datasets (SST [48], MNIST [49] and ImageNet [50]), and ACD was compared against three baselines: CD [3], Integrated

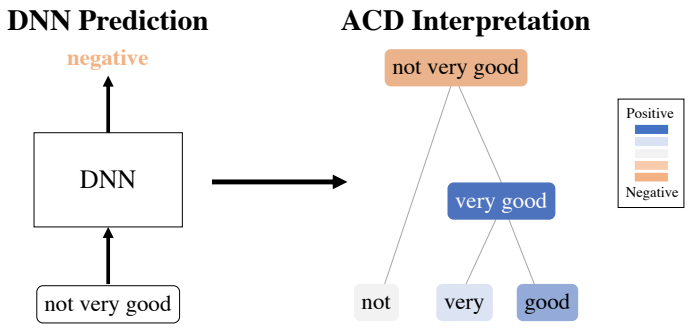

Fig. 2: ACD illustrated through the toy example of predicting the phrase "not very good" as negative. Given the network and prediction, ACD constructs a hierarchy of meaningful phrases and provides importance scores for each identified phrase. In this example, ACD identifies that "very" modifies "good" to become the very positive phrase "very good", which is subsequently negated by "not" to produce the negative phrase "not very good".

Gradients (IG) [27], and occlusion [51, 52]. Each model uses a standard architecture that achieves high classification accuracy, and has an analogous model with substantially poorer performance obtained by randomizing some fraction of its weights while keeping the same predicted label. The objective of this experiment was to determine if subjects could use a small number of interpretations produced by ACD to identify the more accurate of the two models.

For each question, 11 subjects were given interpretations from two different models (one high-performing and one with randomized weights), and asked to identify which of the two models had a higher generalization accuracy. To prevent subjects from simply selecting the model that predicts more accurately for the given example, for each question a subject is shown two sets of examples: one where only the first model predicts correctly and one where only the second model predicts correctly (although one model generalizes to *new* examples much better).

Fig 3 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. The paper [4] also contains results showing that the ACD hierarchy is robust to adversarial perturbations.

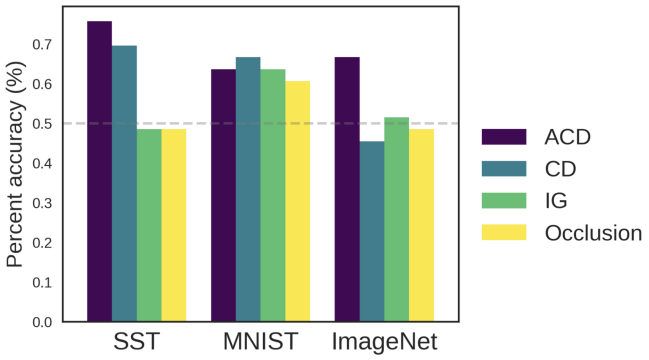

Fig. 3: Results for human studies. Binary accuracy for whether a subject correctly selected the more accurate model using different interpretation techniques.

2.3 Transformation importance with applications to cosmology (TRIM)

Both CD and ACD show 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 meet this challenge, TRIM (Transformation Importance) attributes importance to transformations of the input features (see Fig 4). 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. The work here focuses on combining TRIM with CD, although TRIM can be combined with any local interpretation method.

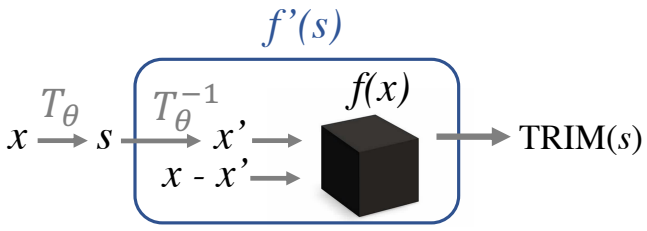

Fig. 4: TRIM: Attributing importance to a transformation of an input $T_\theta(x)$ given a model $f(x)$.

TRIM aims 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 the network as shown in Fig 4, 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 the 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 [35] or CD [3, 4])). 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 $\partial f'/\partial s$.[4] 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,*

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

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, and $attr\,(f; x', x)$ represents the CD score for the features x' as part of the input x. 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 $f(x)$. 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).

As a simple example, 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,[5] achieving a 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 [53]). Fig 5 shows the mean attributions for different topics when the model

[4] If the residual is not added, the gradient of $f' = f \circ T^{-1}$ requires $\partial f/\partial x|_{x'}$, which can potentially cause evaluation of f at the out-of-distribution examples $x' \neq x$.

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

predicts *Fake*. Interestingly, the topic with the highest mean attribution contains recognizable words such as *clinton* and *emails*.

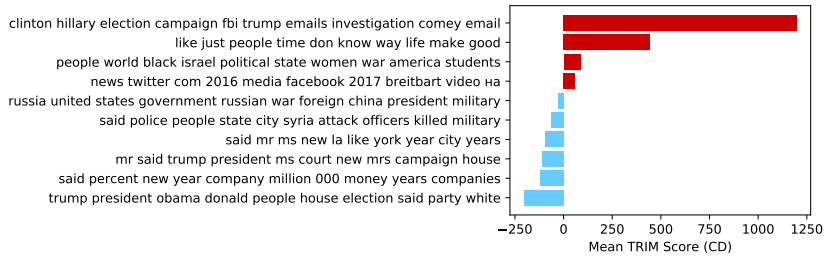

Fig. 5: TRIM attributions for a fake-news classifier based on a topic model transformation. Each row shows one topic, labeled with the top ten words in that topic. Higher attributions correspond to higher contribution to the class *fake*. Calculated over all points which were accurately classified as *fake* in the test set (4,160 points).

Simulation [6]

In the case of a perfectly invertible transformation, such as the Fourier transform, TRIM simply measures the ability of the underlying attribution method (in this case CD) to correctly attribute importance in the transformed space. We run synthetic simulations showing the ability of TRIM with 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 on this task and then test the ability of different methods to assign this frequency the highest importance. Table 2 shows the percentage of errors made by different methods in such a setup. CD has the lowest error on average, compared to popular baselines.

Table 2: Error (%) in recovering a groundtruth important frequency in simulated data using different attribution methods with TRIM, averaged over 500 simulated datasets.

CD	DeepLift [31]	SHAP [38]	Integrated Gradients [27]
0.4 ± 0.282	3.6 ± 0.833	4.0 ± 0.897	4.2 ± 0.876

[6] While simulation study in general is not reality check, as we mentioned in Sec 1, it can be seen as a weaker form of reality check as long as it captures the reality.

3 Using attributions to improve models

This section shows two methods for using the attributions introduced in Sec 2 to directly improve DNNs. Sec 3.1 shows how CD scores can be penalized during training to improve generalization in interesting ways and Sec 3.2 shows how attribution scores can be used to distill a DNN into a simple data-driven wavelet model.

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

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 [54–56] or penalizing the gradients or expected gradients of a neural network [56–61].

Here, we cover contextual decomposition explanation penalization (CDEP), a method which leverages CD to enable the insertion of domain knowledge into a model [6]. 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 [57, 61]; 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.

CDEP works by augmenting the traditional objective function used to train a neural network, as displayed in Eq. (9) 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. The two losses are weighted by a hyperparameter $\lambda \in \mathbb{R}$:

$$\hat{\theta} = \underset{\theta}{\text{argmin}} \overbrace{\mathcal{L}\left(f_\theta(X), y\right)}^{\text{Prediction error}} + \lambda \overbrace{\mathcal{L}_{\text{expl}}\left(\text{expl}_\theta(X), \text{expl}_X\right)}^{\text{Explanation error}} \tag{9}$$

CDEP uses CD as the explanation function used to compute $\text{expl}_\theta(X)$, allowing the penalization of interactions between features. We now substitute the above CD scores into the generic equation in Eq. (9) to arrive at CDEP as it is used in this chapter. 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 prior knowledge label explanations for this subset of features, expl_{x_j} and use it to regularize the explanation:

$$\hat{\theta} = \underset{\theta}{\text{argmin}} \overbrace{\sum_i \sum_c -y_{i,c} \log f_\theta(x_i)_c}^{\text{Prediction error}} + \lambda \overbrace{\sum_i \sum_S ||\beta(x_{i,S}) - \text{expl}_{x_{i,S}}||_1}^{\text{Explanation error}} \quad (10)$$

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.

The choice of prior knowledge 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 prior knowledge human explanations for each data point. To avoid assigning human labels, one may utilize programmatic rules to identify and assign prior knowledge importance to regions, which are then used to help the model identify important/unimportant regions. In a more general case, one may specify importances of different feature interactions.

Towards reality check: ColorMNIST task Here, we highlight CDEP's ability to alter which features a DNN uses to perform digit classification. Similar to one previous study [62], we alter the MNIST dataset to include three color channels and assign each class a distinct color, as shown in Fig 6. An unpenalized DNN trained on this biased data will completely misclassify a test set with inverted colors, dropping to 0% accuracy (see Table 3), suggesting that it learns to classify using the colors of the digits rather than their shape.

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, CDEP penalizes the CD contribution of sampled single-pixel values, following Eq. (10). Minimizing the contribution of single pixels encourages the DNN to focus instead on groups of pixels. Table 3 shows that CDEP can partially divert the network's focus on color to also focus on digit shape. The table includes 2 baselines: penalization of the squared gradients (RRR) [57] and Expected Gradients (EG) [61]. The baselines do not improve the test accuracy of the model on this task above the random baseline, while CDEP significantly improves the accuracy to 31.0%.

Fig. 6: ColorMNIST: the shapes remain the same between the training set and the test set, but the colors are inverted.

Table 3: Test Accuracy on ColorMNIST. 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

The paper [6] further shows how CDEP can be applied to diverse applications, such as notions of fairness in the COMPAS dataset [63] and in natural-language processing.

3.2 Distilling adaptive wavelets from neural networks with interpretations

One promising approach to acquiring highly predictive interpretable models is model distillation. Model distillation is a technique which distills the knowledge in one model into another model. Here, we focus on the case where we distill a DNN into a simple, wavelet model. Wavelets have many useful properties, including fast computation, an orthonormal basis, and interpretation in both spatial and frequency domains [64]. Here, we cover adaptive wavelet distillation (AWD), a method to learn a valid wavelet by distilling information from a trained DNN [7].

Eq. (11) shows the three terms in the formulation of the method. x_i represents the i-th input signal, $\widehat{x}_i$ represents the reconstruction of x_i, h and g represent the lowpass and highpass wavelet filters, and Ψx_i denotes the wavelet coefficients of x_i. λ is a hyperparameter penalizing the sparsity of the wavelet coefficients, which can help to learn a compact representation of the input signal and γ is a

hyperparameter controlling the strength of the interpretation loss, which controls how much to use the information coming from a trained model f:

$$\underset{h,g}{\text{minimize}}\ \mathcal{L}(h,g) = \underbrace{\frac{1}{m}\sum_i \|x_i - \widehat{x}_i\|_2^2}_{\text{Reconstruction loss}} + \underbrace{\frac{1}{m}\sum_i W(h,g,x_i;\lambda)}_{\text{Wavelet loss}} + \underbrace{\gamma \sum_i \|\text{TRIM}_f(\Psi x_i)\|_1}_{\text{Interpretation loss}}, \tag{11}$$

Here the reconstruction loss ensures that the wavelet transform is invertible, allowing for reconstruction of the original data. Hence the transform does not lose any information in the input data.

The wavelet loss ensures that the learned filters yield a valid wavelet transform. Specifically, [65,66] characterize the sufficient and necessary conditions on h and g to build an orthogonal wavelet basis. Roughly speaking, these conditions state that in the frequency domain the mass of the lowpass filer h is concentrated on the range of low frequencies while the highpass filter g contains more mass in the high frequencies. We also desire the learned wavelet to provide sparse representations so we add the ℓ_1 norm penalty on the wavelet coefficients. Combining all these conditions via regularization terms, we define the wavelet loss at the data point x_i as

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

where g is set as $g[n] = (-1)^n h[N-1-n]$ and where N is the support size of h (see [7] for further details on the formulations of wavelet loss).

Finally, 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, as described in Sec 2.3. This forces the wavelet transform to produce representations that concisely explain the model's predictions at different scales and locations.

A key difference between AWD and existing wavelet techniques (e.g. [67,68] is that they use *interpretations from a trained model* to learn the wavelets; this incorporates information not just about the signal but also an outcome of interest and the inductive biases learned by a DNN. This can help learn an interpretable representation that is well-suited to efficient computation and effective prediction.

Reality check: molecular partner prediction For evaluation, see Sec 4.1, which shows an example of how a distilled AWD model can provide a simpler, more interpretable model while improving prediction accuracy.

4 Real-data problems showcasing interpretations

In this section, we focus on three real-data problems where the methods introduced in Sec 2 and Sec 3 are able to provide useful interpretations in context. Sec 4.1 describes how AWD can distill DNNs used in cell biology, Sec 4.2 describes how TRIM + CD yield insights in a cosmological context, and Sec 4.3 describes how CDEP can be used to ignore spurious correlations in a medical imaging task.

4.1 Molecular partner prediction

We now turn our attention to a crucial question in cell biology: understanding clathrin-mediated endocytosis (CME) [69, 70]. It is the primary pathway by which things are transported into the cell, making it essential functions of higher eukaryotic life [71]. Many questions about this process remain unanswered, prompting a line of studies aiming to better understand this process [72]. One major challenge with analysis of CME, 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 7 shows the pipeline for this challenging problem. Tracking algorithms run on videos of cells identify time-series traces of endocytic events. An LSTM model learns to classify which endocytic events are successful 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.

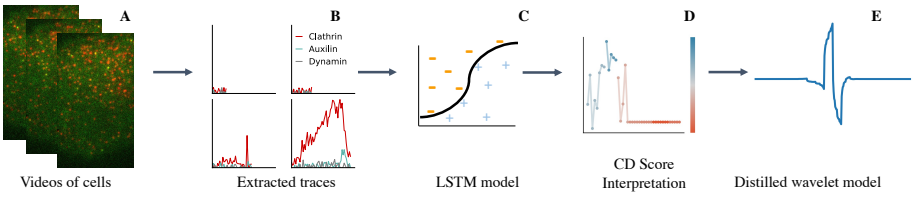

Fig. 7: Molecular partner prediction pipeline. (**A**) Tracking algorithms run on videos of cells identify (**B**) time-series traces of endocytic events. (**C**) An LSTM model learns to classify which endocytic events are successful and (**D**) CD scores identify which parts of the traces the model 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 model, we extract

only the maximum 6 wavelet coefficients at each scale. By taking the maximum coefficients, these features are expected to be invariant to the specific locations 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 7 shows 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 4 compares the results for AWD to the original LSTM and the initialized, non-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 of 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 4: 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 interpreration 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.

4.2 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 [73]. 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 [74–76].

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 [5]). To understand what features the model is using, we desire an interpretation in the space of the power spectrum. The images in Fig 8 show how different information is contained within different frequency bands in the mass maps. The plot in Fig 8 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.[7] 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.

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

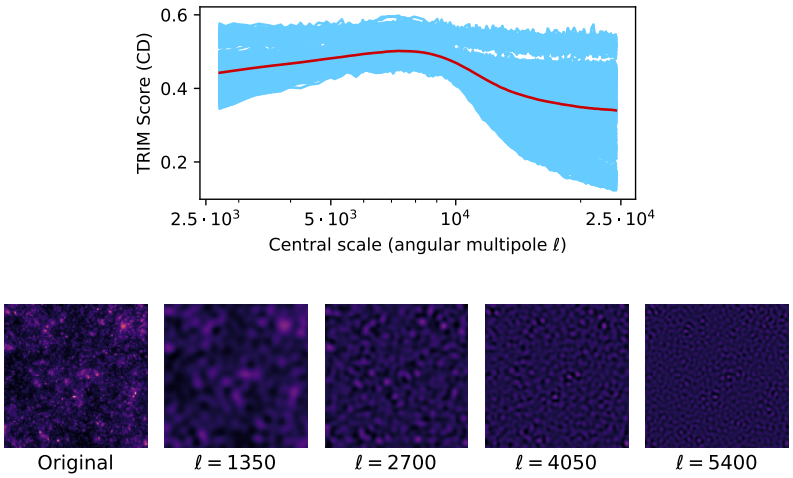

Fig. 8: Different scales (i.e. frequency bands) contribute differently to the prediction of Ω_m. Each blue line corresponds to one testing image and the red line shows the mean. Images show the features present at different scales. The bandwidth is Δ_ℓ =2,700.

Fig 9 shows some of the curves from Fig 8 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 varies 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.

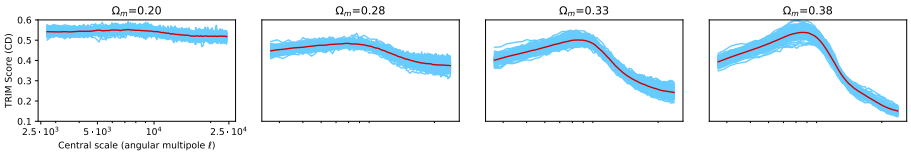

Fig. 9: TRIM attributions vary with the value of Ω_m.

4.3 Improving skin cancer classification via CDEP

In recent years, deep learning has achieved impressive results in diagnosing skin cancer [77]. However, the datasets used to train these models often include spu-

rious features which make it possible to attain high test accuracy without learning the underlying phenomena [39]. 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 be seen in Fig 10 [78]. We use CDEP to remedy this problem by penalizing the DNN placing importance on the patches during training.

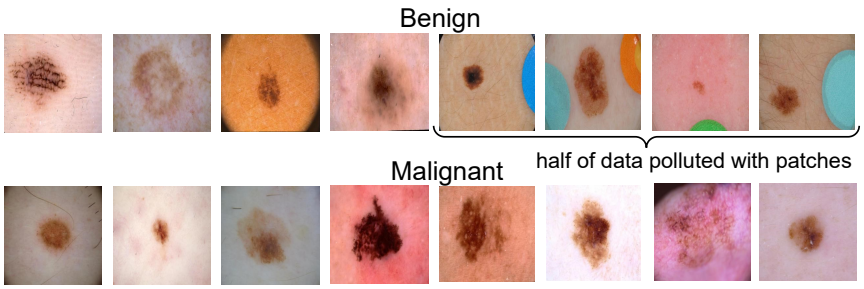

Fig. 10: Example images from the ISIC dataset. Half of the benign lesion images include a patch in the image. Training on this data results in the neural network overly relying on the patches to classify images; CDEP avoids this.

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. In order to identify the spurious patches, binary maps of the patches for the skin cancer task are segmented using SLIC, a common image-segmentation algorithm [79]. After the spurious patches were identified, they are penalized using to have zero importance.

Table 5 shows results comparing the performance of a DNN trained with and without 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, compared to both a Vanilla DNN and using the RRR method introduced in [57]. Further visual inspection shows that the DNN attributes low importance to regions in the images with patches.

Table 5: 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 test-set images that do not include patches ("no patches"). The network trained with CDEP generalizes better, getting higher AUC and F1 on both.

	AUC (no patches)	F1 (no patches)	AUC (all)	F1 (all)
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**

5 Discussion

Overall, the interpretation methods here are shown to (1) accurately recover known importances for features / feature interactions [3], (2) correctly inform human decision-making and be robust to adversarial perturbations [80], and (3) reliably alter a neural network's predictions when regularized appropriately [6]. For each case, we demonstrated the use of reality checks through predictive accuracy (the most common form of reality check) or through domain knowledge which is relevant to a particular domain/audience.

There is considerable future work to do in developing and evaluating attributions, particularly in distilling/building interpretable models for real-world domains and understanding how to better make useful interpretation methods. Below we discuss them in turn.

5.1 Building/distilling accurate and interpretable models

In the ideal case, a practitioner can develop a simple model to make their predictions, ensuring interpretability by obviating the need for post-hoc interpretation. Interpretable models tend to be faster, more computationally efficient, and smaller than their DNN counterparts. Moreover, interpretable models allow for easier inspection of knowledge extracted from the learned models and make reality checks more transparent. AWD [7] represents one effort to use attributions to distill DNNs into an interpretable wavelet model, but the general idea can go much further. There are a variety of interpretable models, such as rule-based models [81, 82] or additive models [14] whose fitting process could benefit from accurate attributions. Moreover, AWD and related techniques could be extended beyond the current setting to unsupervised/reinforcement learning settings or to incorporate multiple layers. Alternatively, attributions can be used as feature engineering tools, to help build simpler, more interpretable models. More useful features can help enable better exploratory data analysis, unsupervised learning, or reality checks.

5.2 Making interpretations useful

Furthermore, there is much work remaining to improve the relevancy of interpretations for a particular audience/problem. Given the abundance of possible interpretations, it is particularly easy for researchers to propose novel methods which do not truly solve any real-world problems or fail to faithfully capture some aspects of reality. A strong technique to avoid this is to directly test newly introduced methods in solving a domain problem. Here, we discussed several real-data problems that have benefited from improved interpretations [4], spanning from cosmology to cell biology. In instances like this, where interpretations are used directly to solve a domain problem, their relevancy is indisputable and reality checks can be validated through domain knowledge. A second, less direct, approach is the use of human studies where humans are asked to perform tasks, such as evaluating how much they trust a model's predictions [80]. While challenging to properly construct and perform, these studies are vital to demonstrating that new interpretation methods are, in fact, relevant to any potential practitioners. We hope the plethora of open problems in various domains such as science, medicine, and public policy can help guide and benefit from improved interpretability going forward.

References

1. Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017.
2. W James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences*, 116(44):22071–22080, 2019.
3. W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *ICLR*, 2018.
4. Chandan Singh, W. James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*, 2019.
5. Chandan Singh, Wooseok Ha, Francois Lanusse, Vanessa Boehm, Jia Liu, and Bin Yu. Transformation importance with applications to cosmology. *arXiv preprint arXiv:2003.01926*, 2020.
6. Laura Rieger, Chandan Singh, William Murdoch, and Bin Yu. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International Conference on Machine Learning*, pages 8116–8126. PMLR, 2020.
7. Wooseok Ha, Chandan Singh, Francois Lanusse, Eli Song, Song Dang, Kangmin He, Srigokul Upadhyayula, and Bin Yu. Adaptive wavelet distillation from neural networks through interpretations. *arXiv preprint arXiv:2107.09145*, 2021.
8. Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen AWM van der Laak, Bram van Ginneken, and Clara I Sánchez. A survey on deep learning in medical image analysis. *Medical image analysis*, 42:60–88, 2017.
9. Tim Brennan and William L Oliver. The emergence of machine learning techniques in criminology. *Criminology & Public Policy*, 12(3):551–562, 2013.

10. Christof Angermueller, Tanel Pärnamaa, Leopold Parts, and Oliver Stegle. Deep learning for computational biology. *Molecular systems biology*, 12(7):878, 2016.
11. Mai-Anh T Vu, Tulay Adali, Demba Ba, Gyorgy Buzsaki, David Carlson, Katherine Heller, Conor Liston, Cynthia Rudin, Vikaas Sohal, Alik S Widge, et al. A shared vision for machine learning in neuroscience. *Journal of Neuroscience*, pages 0508–17, 2018.
12. Bryce Goodman and Seth Flaxman. European union regulations on algorithmic decision-making and a" right to explanation". *arXiv preprint arXiv:1606.08813*, 2016.
13. Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226. ACM, 2012.
14. Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. 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*, pages 1721–1730. ACM, 2015.
15. Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.
16. Jason Yosinski, Jeff Clune, Anh Nguyen, Thomas Fuchs, and Hod Lipson. Understanding neural networks through deep visualization. *arXiv preprint arXiv:1506.06579*, 2015.
17. Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.
18. Nicholas Frosst and Geoffrey Hinton. Distilling a neural network into a soft decision tree. *arXiv preprint arXiv:1711.09784*, 2017.
19. Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 39–48, 2016.
20. Quanshi Zhang, Ruiming Cao, Feng Shi, Ying Nian Wu, and Song-Chun Zhu. Interpreting cnn knowledge via an explanatory graph. *arXiv preprint arXiv:1708.01785*, 2017.
21. Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. Sanity checks for saliency maps. In *Advances in Neural Information Processing Systems*, pages 9505–9515, 2018.
22. Arushi Gupta and Sanjeev Arora. A simple saliency method that passes the sanity checks. *arXiv preprint arXiv:1905.12152*, 2019.
23. Finale Doshi-Velez and Been Kim. A roadmap for a rigorous science of interpretability. *arXiv preprint arXiv:1702.08608*, 2017.
24. Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.
25. Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154*, 2018.
26. Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for simplicity: The all convolutional net. *arXiv preprint arXiv:1412.6806*, 2014.
27. Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. *ICML*, 2017.
28. Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep

networks via gradient-based localization. *See https://arxiv. org/abs/1610.02391 v3*, 7(8), 2016.
29. David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert MÃžller. How to explain individual classification decisions. *Journal of Machine Learning Research*, 11(Jun):1803–1831, 2010.
30. W James Murdoch and Arthur Szlam. Automatic rule extraction from long short term memory networks. 2017.
31. Avanti Shrikumar, Peyton Greenside, Anna Shcherbina, and Anshul Kundaje. Not just a black box: Learning important features through propagating activation differences. *arXiv preprint arXiv:1605.01713*, 2016.
32. Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140, 2015.
33. Piotr Dabkowski and Yarin Gal. Real time image saliency for black box classifiers. *arXiv preprint arXiv:1705.07857*, 2017.
34. Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. *arXiv preprint arXiv:1704.03296*, 2017.
35. 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*, pages 1135–1144. ACM, 2016.
36. Luisa M Zintgraf, Taco S Cohen, Tameem Adel, and Max Welling. Visualizing deep neural network decisions: Prediction difference analysis. *arXiv preprint arXiv:1702.04595*, 2017.
37. Marco Ancona, Enea Ceolini, Cengiz Oztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. In *6th International Conference on Learning Representations (ICLR 2018)*, 2018.
38. Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*, pages 4768–4777, 2017.
39. Julia K. Winkler, Christine Fink, Ferdinand Toberer, Alexander Enk, Teresa Deinlein, Rainer Hofmann-Wellenhof, Luc Thomas, Aimilios Lallas, Andreas Blum, Wilhelm Stolz, and Holger A. Haenssle. 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. *JAMA Dermatology*, 08 2019.
40. Nikhil Garg, Londa Schiebinger, Dan Jurafsky, and James Zou. Word embeddings quantify 100 years of gender and ethnic stereotypes. *Proceedings of the National Academy of Sciences*, 115(16):E3635–E3644, 2018.
41. Ziad Obermeyer, Brian Powers, Christine Vogeli, and Sendhil Mullainathan. Dissecting racial bias in an algorithm used to manage the health of populations. *Science*, 366(6464):447–453, 2019.
42. Julia Dressel and Hany Farid. The accuracy, fairness, and limits of predicting recidivism. *Science advances*, 4(1):eaao5580, 2018.
43. Michael Tsang, Youbang Sun, Dongxu Ren, and Yan Liu. Can i trust you more? model-agnostic hierarchical explanations. *arXiv preprint arXiv:1812.04801*, 2018.

44. Kedar Dhamdhere, Ashish Agarwal, and Mukund Sundararajan. The shapley taylor interaction index. *arXiv preprint arXiv:1902.05622*, 2019.
45. Hao Zhang, Xu Cheng, Yiting Chen, and Quanshi Zhang. Game-theoretic interactions of different orders. *arXiv preprint arXiv:2010.14978*, 2020.
46. Rui Wang, Xiaoqian Wang, and David I Inouye. Shapley explanation networks. *arXiv preprint arXiv:2104.02297*, 2021.
47. Summer Devlin, Chandan Singh, W James Murdoch, and Bin Yu. Disentangled attribution curves for interpreting random forests and boosted trees. *arXiv preprint arXiv:1905.07631*, 2019.
48. Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.
49. Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.
50. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
51. Jiwei Li, Will Monroe, and Dan Jurafsky. Understanding neural networks through representation erasure. *arXiv preprint arXiv:1612.08220*, 2016.
52. Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European conference on computer vision*, pages 818–833. Springer, 2014.
53. David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.
54. Kaylee Burns, Lisa Anne Hendricks, Kate Saenko, Trevor Darrell, and Anna Rohrbach. Women also snowboard: Overcoming bias in captioning models. *arXiv preprint arXiv:1803.09797*, 2018.
55. Masahiro Mitsuhara, Hiroshi Fukui, Yusuke Sakashita, Takanori Ogata, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. Embedding human knowledge in deep neural network via attention map. *arXiv preprint arXiv:1905.03540*, 2019.
56. Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. Learning credible deep neural networks with rationale regularization. *arXiv preprint arXiv:1908.05601*, 2019.
57. Andrew Slavin Ross, Michael C Hughes, and Finale Doshi-Velez. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*, 2017.
58. Yujia Bao, Shiyu Chang, Mo Yu, and Regina Barzilay. Deriving machine attention from human rationales. *arXiv preprint arXiv:1808.09367*, 2018.
59. 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.
60. Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. *arXiv preprint arXiv:1906.08286*, 2019.
61. Gabriel Erion, Joseph D Janizek, Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*, 2019.
62. Yi Li and Nuno Vasconcelos. Repair: Removing representation bias by dataset resampling. *arXiv preprint arXiv:1904.07911*, 2019.
63. Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. How we analyzed the compas recidivism algorithm. *ProPublica (5 2016)*, 9, 2016.

64. Stéphane Mallat. *A wavelet tour of signal processing, Third edition: The sparse way*. Academic Press, 2008.
65. Stephane G Mallat. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE transactions on pattern analysis and machine intelligence*, 11(7):674–693, 1989.
66. Yves Meyer. *Wavelets and Operators: Volume 1*. Number 37. Cambridge university press, 1992.
67. Daniel Recoskie and Richard Mann. Learning sparse wavelet representations. *arXiv preprint arXiv:1802.02961*, 2018.
68. Daniel Recoskie. Learning sparse orthogonal wavelet filters. 2018.
69. Tom Kirchhausen, David Owen, and Stephen C Harrison. Molecular structure, function, and dynamics of clathrin-mediated membrane traffic. *Cold Spring Harbor perspectives in biology*, 6(5):a016725, 2014.
70. Kangmin He, Eli Song, Srigokul Upadhyayula, Song Dang, Raphael Gaudin, Wesley Skillern, Kevin Bu, Benjamin R Capraro, Iris Rapoport, Ilja Kusters, et al. Dynamics of auxilin 1 and gak in clathrin-mediated traffic. *Journal of Cell Biology*, 219(3), 2020.
71. Harvey T McMahon and Emmanuel Boucrot. Molecular mechanism and physiological functions of clathrin-mediated endocytosis. *Nature reviews Molecular cell biology*, 12(8):517, 2011.
72. Marko Kaksonen and Aurélien Roux. Mechanisms of clathrin-mediated endocytosis. *Nature Reviews Molecular Cell Biology*, 19(5):313, 2018.
73. Matthias Bartelmann and Peter Schneider. Weak gravitational lensing. *Physics Reports*, 340(4-5):291–472, 2001.
74. Dezső Ribli, Bálint Ármin Pataki, José Manuel Zorrilla Matilla, Daniel Hsu, Zoltán Haiman, and István Csabai. Weak lensing cosmology with convolutional neural networks on noisy data. *Monthly Notices of the Royal Astronomical Society*, 490(2):1843–1860, 2019.
75. Dezső Ribli, Bálint Ármin Pataki, and István Csabai. An improved cosmological parameter inference scheme motivated by deep learning. *Nature Astronomy*, 3(1):93, 2019.
76. Janis Fluri, Tomasz Kacprzak, Aurelien Lucchi, Alexandre Refregier, Adam Amara, Thomas Hofmann, and Aurel Schneider. Cosmological constraints with deep learning from kids-450 weak lensing maps. *Physical Review D*, 100(6):063514, 2019.
77. Andre Esteva, Brett Kuprel, Roberto A Novoa, Justin Ko, Susan M Swetter, Helen M Blau, and Sebastian Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115, 2017.
78. Noel Codella, Veronica Rotemberg, Philipp Tschandl, M Emre Celebi, Stephen Dusza, David Gutman, Brian Helba, Aadi Kalloo, Konstantinos Liopyris, Michael Marchetti, et al. Skin lesion analysis toward melanoma detection 2018: A challenge hosted by the international skin imaging collaboration (isic). *arXiv preprint arXiv:1902.03368*, 2019.
79. Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.
80. Chandan Singh, W. James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. In *International Conference on Learning Representations*, 2019.

81. Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, et al. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371, 2015.
82. Chandan Singh, Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu. imodels: a python package for fitting interpretable models. *Journal of Open Source Software*, 6(61):3192, 2021.