Journal of Machine Learning Research 22 (2021) 1-54 Submitted 10/20; Revised 3/21; Published 5/21

Explaining Explanations: Axiomatic Feature Interactions for Deep Networks

Joseph D. Janizek* JJANIZEK@CS.WASHINGTON.EDU
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA 98195-4322, USA

Pascal Sturmfels* PSTURM@CS.WASHINGTON.EDU
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA 98195-4322, USA

Su-In Lee SUINLEE@CS.WASHINGTON.EDU
Paul G. Allen School of Computer Science & Engineering
University of Washington
Seattle, WA 98195-4322, USA

Editor: Ruslan Salakhutdinov

Abstract

Recent work has shown great promise in explaining neural network behavior. In particular, feature attribution methods explain the features that are important to a model's prediction on a given input. However, for many tasks, simply identifying significant features may be insufficient for understanding model behavior. The *interactions* between features within the model may better explain not only the model, but why certain features outrank others in importance. In this work, we present Integrated Hessians, an extension of Integrated Gradients (Sundararajan et al., 2017) that explains pairwise feature interactions in neural networks. Integrated Hessians overcomes several theoretical limitations of previous methods, and unlike them, is not limited to a specific architecture or class of neural network. Additionally, we find that our method is faster than existing methods when the number of features is large, and outperforms previous methods on existing quantitative benchmarks.

Keywords: Feature attribution, feature interaction, Aumann-Shapley value, interpretability, neural networks

1. Introduction and Prior Work

Deep neural networks are one of the most popular classes of machine learning (ML) model. They can achieve state-of-the-art performance in problem domains ranging from natural language processing to image recognition (Devlin et al., 2018; He et al., 2016). They have even outperformed other non-linear model types on structured tabular data (Shavitt and Segal, 2018). Because neural networks have traditionally been more difficult to interpret than simpler model classes, gaining a better understanding of their predictions is desirable for many reasons. Where these algorithms are used in automated decisions that affect

∗. These authors contributed equally to this work and are listed alphabetically.

humans, explanations may be legally required (Selbst and Powles, 2017). When used in high stakes applications, it is essential to ensure that models are making safe decisions for the right reasons (Geis et al., 2019). During model development, interpretability methods can help debug undesirable model behavior (Sundararajan et al., 2017).

1.1 Feature Attribution Methods

Many recent approaches focus on interpreting deep neural networks, ranging from those aiming to distill complex models into simpler ones (Tan et al., 2018; Wu et al., 2018; Puri et al., 2017) to those seeking to identify key concepts learned by a network (Kim et al., 2017; Olah et al., 2018, 2017; Fong and Vedaldi, 2018; Erhan et al., 2009; Mahendran and Vedaldi, 2015). Other approaches aim to build interpretable deep models, such as probabilistic interpretable deep models and deep networks for causal effects (Zhang et al., 2020; Zhou et al., 2016; Louizos et al., 2017). Among the best-studied sets of approaches for interpreting deep neural networks is a class known as *feature attribution methods* (Binder et al., 2016; Shrikumar et al., 2017; Lundberg and Lee, 2017; Ribeiro et al., 2016). These approaches explain a model's prediction by assigning credit to each input feature based on how much it influences the prediction. Although these approaches help practitioners identify salient features, they do not explain *why* the features are important or address features interactions in a model. To enrich our understanding of model behavior, we must develop methods to explain both. For example, in Figure 1, we show that word-level interactions can help us distinguish why deeper, more expressive neural networks outperform simpler ones on language tasks.

1.2 Feature Interaction Methods

Several existing methods explain feature interactions in neural networks. Cui et al. (2019) explain global interactions in Bayesian Neural Networks (BNN) by examining pairs of features that have large second-order derivatives at the input. The Neural Interaction Detection method detects statistical interactions between features by examining the weight matrices of feed-forward neural networks (Tsang et al., 2017). Further, several authors have proposed domain-specific methods to find interactions in the area of deep learning for genomics (Koo et al., 2018; Greenside et al., 2018). For example, Deep Feature Interaction Maps detect interactions between two features by calculating the change in the attribution of one feature that is incurred by changing the value of the second (Greenside et al., 2018). Singh et al. (2018) generalize Contextual Decomposition (Murdoch et al., 2018) to explain interactions for feed-forward and convolutional architectures. In game theory literature, Grabisch and Roubens (1999) propose the Shapley Interaction Index, which allocates credit to interactions between players in a coalitional game by considering all possible subsets of players. Recently, Dhamdhere et al. (2019) suggested a modified version of the Shapley Interaction Index that weights certain subsets of players differently in order to achieve a different set of desired axioms.

A recent paper by Tsang et al. (2020) proposes a method for "interaction attribution," which they compare to our method. Their approach has three steps: (1) detect pairwise interactions between features using a method called ArchDetect, (2) use these pairwise interactions to cluster features into groups so that interactions occur only between features

within the same group, and (3) attribute importance to those groups of features using a method called ArchAttribute. In Appendix B.1, we show that the quantity described by ArchAttribute is equivalent to a *Group Shapley value* rather than an interaction value, a fundamentally different quantity than we aim to measure in our paper, and hence is not comparable. Our method detects the extent of non-additivity between feature pairs, while their approach quantifies the total contribution of a group of interacting features. Both quantities may be of interest, depending on the particular application. Therefore, we instead compare our interaction detection method to ArchDetect.

1.3 Limitations of Prior Approaches

Previous approaches have substantially advanced our understanding of feature interaction in neural networks. However, all suffer from practical limitations, including being limited to specific types of architectures. Neural Interaction Detection applies only to feed-forward neural network architectures and cannot be used on networks with convolutions, recurrent units, or self-attention. Contextual Decomposition has been applied to LSTMs, feed-forward neural networks and convolutional networks, but to our knowledge is not straightforward to apply to more recent innovations in deep learning, such as self-attention layers. The approach suggested by Cui et al. (2019) is limited by its required use of Bayesian Neural Networks; it is unclear how to apply the method to standard neural networks. Deep Feature Interaction Maps work when a model's input features have only a few discrete values (such as genomic sequence data, which can be *in silico* mutated), since it marginalizes over all possible values of paired input features. The Shapley Interaction Index and Shapley Taylor Interaction Index, like the Shapley value, are NP-hard to compute exactly (Elkind et al., 2009).

Furthermore, most current methods to detect interactions fail to satisfy the common-sense axioms proposed for feature attribution methods (Sundararajan et al., 2017; Lundberg and Lee, 2017). As a result, these approaches are provably unable to find learned interactions or more generally find counter-intuitive interactions (see section 4). Current methods that do satisfy such axioms, such as those based on the Shapley Interaction Index (Grabisch and Roubens, 1999; Dhamdhere et al., 2019), are computationally inefficient to compute or even approximate.

1.4 Our contributions

First, we propose an approach, Integrated Hessians, to quantify pairwise feature interactions that can be applied to any neural network architecture. We also identify several common-sense axioms that feature-level interactions should satisfy and show that our proposed method satisfies them. Third, we provide a principled way to compute interactions in ReLU-based networks, which are piece-wise linear and have zero second derivatives. Further, we evaluate our method against existing methods and show that it more accurately identifies interactions in simulated data. Finally, we demonstrate the utility of Integrated Hessians in a variety of applications where identifying feature interactions in neural networks is useful.

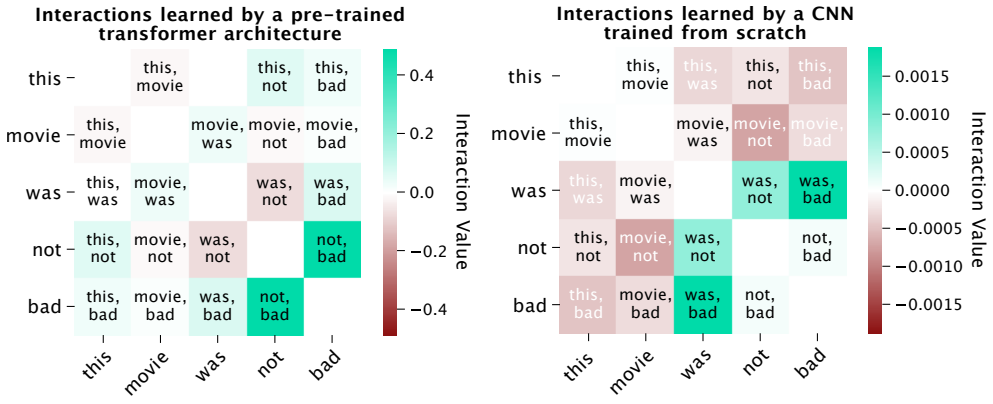

Figure 1: Interactions explain why certain models outperform others. Here, we examine word interactions in the sentence "this movie was not bad." We compare two models trained to perform sentiment analysis on the Stanford Sentiment data set: a pre-trained transformer, DistilBERT (left), which predicts the sentence has a positive sentiment with 98.2% confidence, and a convolutional neural network trained from scratch (right), which predicts a negative sentiment with 97.6% confidence. The transformer picks up negation patterns: "not bad" has a positive interaction, despite the word "bad" being negative. The CNN mostly picks up negative interactions, like "movie not" and "movie bad."

2. Explaining Explanations with Integrated Hessians

To derive our feature interaction values, we first consider Integrated Gradients (IG), a feature attribution method proposed by Sundararajan et al. (2017) based on the Aumann-Shapley value; the Aumann-Shapley value is a variant of the Shapley value for cooperative games with continuous rather than discrete players (Aumann and Shapley, 2015). We represent our model as a function $f : \mathbb{R}^d \mapsto \mathbb{R}$.[1] For a function $f(x)$, the IG attribution for the ith feature is defined as:

$$\phi_i(x) = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha, \tag{1}$$

where x is the sample to be explained and x' is a baseline value. For notation concision, we suppress the dependence on the choice of baseline x' when we write the IG attribution $\phi_i(x)$, but we discuss the importance of baselines in Section 2.1 and Appendix B.4.2. Although f is often a neural network, the sole requirement for computing attribution values is that f be differentiable along the path from x' to x. Our key insight is that the IG value for a differentiable model $f : \mathbb{R}^d \mapsto \mathbb{R}$ is *itself* a differentiable function $\phi_i : \mathbb{R}^d \mapsto \mathbb{R}$. This means

1. For multi-output models, such as multi-class classification problems, we assume the function is indexed into the correct output class.

that we can apply IG to itself in order to explain the degree to which feature j impacted the importance of feature i:

$$\Gamma_{i,j}(x) = \phi_j(\phi_i(x)). \tag{2}$$

For $i \neq j$, we derive that:

$$\Gamma_{i,j}(x) = (x_i - x'_i)(x_j - x'_j) \times \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta. \tag{3}$$

For $i = j$, the formula $\Gamma_{i,i}(x)$ has an additional first-order term:

$$\begin{aligned} \Gamma_{i,i}(x) = (x_i - x'_i) \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha\beta(x - x'))}{\partial x_i} d\alpha d\beta + \\ (x_i - x'_i)^2 \times \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta. \end{aligned} \tag{4}$$

We interpret $\Gamma_{i,j}(x)$ as the explanation of the importance of feature i in terms of the input value of feature j. For a full derivation of Integrated Hessians, see Appendix A.

2.1 Baselines and Expected Hessians

Several feature attribution methods have identified the necessity of a baseline value representing a lack of information when generating explanations (Sundararajan et al., 2017; Shrikumar et al., 2017; Binder et al., 2016; Lundberg and Lee, 2017). However, more recent work notes that choosing a single baseline value to represent a lack of information can be challenging in certain domains (Kindermans et al., 2019; Kapishnikov et al., 2019; Sundararajan and Taly, 2018; Ancona et al., 2017; Fong and Vedaldi, 2017; Sturmfels et al., 2020). As an alternative, Erion et al. (2019) proposed an extension of IG called Expected Gradients (EG), which samples many baseline inputs from the training set. We can therefore apply EG to itself to get Expected Hessians:

$$\Gamma^{EG}_{i,j}(x) = \mathbb{E}_{\alpha\beta \sim U(0,1) \times U(0,1), x' \sim D} \left[(x_i - x'_i)(x_j - x'_j) \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} \right]. \tag{5}$$

$$\begin{aligned} \Gamma^{EG}_{i,i}(x) = \mathbb{E}_{\alpha\beta \sim U(0,1) \times U(0,1), x' \sim D} \Bigg[(x_i - x'_i) \frac{\partial f(x' + \alpha\beta(x - x'))}{\partial x_i} \\ + (x_i - x'_i)^2 \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} \Bigg], \end{aligned} \tag{6}$$

where the expectation is over $x' \sim D$ for an underlying data distribution D, $\alpha \sim U(0,1)$ and $\beta \sim U(0,1)$. This formulation is useful when there is no single, natural baseline. Deriving Expected Hessians follows the same steps as deriving Integrated Hessians, but the integrals can be viewed as integrating over the product of two uniform distributions $\alpha\beta \sim U(0,1) \times U(0,1)$. We use Integrated Hessians for all main text examples; however, some of the Appendix examples use the Expected Hessians formulation.

Almost every attribution and interaction method to date requires a choice of baseline. Some approaches explicitly discuss their reliance on baselines (Sundararajan et al., 2017; Sundararajan and Najmi, 2019; Tsang et al., 2020), while others rely on baselines without explicitly mentioning them in their work. For example, Contextual Decomposition turns features "off" by setting them to a single reference baseline of 0 (Murdoch et al., 2018). By implementing a solution for domains where it is difficult to choose a single baseline, we seek to alleviate issues common to many interaction methods. Additionally, we note that the formulation as an expectation strictly generalizes the Integrated Hessians formulation; while it is possible to sample from many different baselines, integrating from a single one can also be conceptualized as sampling from a distribution with all of its density on that baseline. We do not always anticipate that sampling a larger distribution of baselines would be advantageous, especially in application settings where a single natural baseline can be easily defined.

2.2 Fundamental Axioms for Interaction Values

We now describe common-sense axioms that every interaction method should satisfy, and we show that Integrated Hessians satisfies them all.

2.2.1 Self and Interaction Completeness Axiom

Sundararajan et al. (2017) showed that, among other theoretical properties, IG satisfies the completeness axiom, which states: $\sum_i \phi_i(x) = f(x) - f(x')$. We show the following two equalities, which are immediate consequences of completeness:

$$\sum_i \sum_j \Gamma_{i,j}(x) = f(x) - f(x'). \tag{7}$$

$$\Gamma_{i,i}(x) = \phi_i(x) - \sum_{j \neq i} \Gamma_{i,j}(x). \tag{8}$$

We call equation (7) the *interaction completeness* axiom: the sum of the $\Gamma_{i,j}(x)$ terms adds up to the difference between the output of f at x and at the baseline x'. This axiom lends itself to another natural interpretation of $\Gamma_{i,j}(x)$: as the interaction between features i and j. That is, it represents the contribution that the pair of features i and j together add to the output $f(x) - f(x')$. It is vital to satisfy interaction completeness because it demonstrates a relationship between model output and interaction values. Without this axiom, it would not be clear how to interpret the scale of interactions.

Equation (8) shows how to interpret the self-interaction term $\Gamma_{i,i}(x)$ as the *main effect* of feature i after interactions with all other features have been subtracted. We note that equation (8) also implies the following, intuitive property about the main effect: if $\Gamma_{i,j} = 0$ for all $j \neq i$, or in the degenerate case where i is the only feature, we have $\Gamma_{i,i} = \phi_i(x)$. We call this the *self-completeness* axiom. Satisfying self-completeness provides the vital guarantee that the main effect of feature i equals its feature attribution value if that feature interacts with no other features.

The proof of these two equations is straightforward. First, we note that $\phi_i(x') = 0$ for any i because $x'_i - x'_i = 0$. Then, by completeness of Integrated Gradients, we have that:

$$\sum_j \Gamma_{i,j}(x) = \phi_i(x) - \phi_i(x') = \phi_i(x). \tag{9}$$

Re-arrangement provides the *self-completeness* axioms:

$$\Gamma_{i,i}(x) = \phi_i(x) \text{ if } \Gamma_{i,j}(x) = 0, \forall j \neq i. \tag{10}$$

Since Integrated Gradients satisfies completeness:

$$\sum_i \phi_i(x) = f(x) - f(x'). \tag{11}$$

Making the appropriate substitution from equation 9 shows the *interaction completeness* axiom:

$$\sum_i \sum_j \Gamma_{i,j}(x) = f(x) - f(x'). \tag{12}$$

2.2.2 Sensitivity Axiom

Integrated Gradients satisfies an axiom called *sensitivity*, which states that given an input x and a baseline x', if $x_i = x'_i$ for all i except j where $x_j \neq x'_j$ and if $f(x) \neq f(x')$, then $\phi_j(x) \neq 0$. Specifically, by completeness we know that $\phi_j(x) = f(x) - f(x')$. Intuitively, this means that if only one feature differs between the baseline and the input and changing that feature would change the output, then the amount the output changes should be equal to the importance of that feature.

We can extend this axiom to interactions by considering the case where two features differ from the baseline. We call this axiom *interaction sensitivity* and describe it as follows. If an input x and a baseline x' are equal everywhere except $x_i \neq x'_i$ and $x_j \neq x'_j$, and if $f(x) \neq f(x')$, then: $\Gamma_{i,i}(x) + \Gamma_{j,j}(x) + 2\Gamma_{i,j}(x) = f(x) - f(x') \neq 0$ and $\Gamma_{\ell,k} = 0$ for all $\ell, k \neq i, j$. Intuitively, this states that if the only features that differ from the baseline are i and j, then the difference in the output $f(x) - f(x')$ must be solely attributable to the main effects of i and j plus the interaction between them. This axiom holds simply by applying *interaction completeness* and observing that $\Gamma_{\ell,k}(x) = 0$ if $x_\ell = x'_\ell$ or $x_k = x'_k$.

2.2.3 Implementation Invariance Axiom

The implementation invariance axiom, described in Sundararajan et al. (2017), states that for two models f and g such that $f = g$, then $\phi_i(x; f) = \phi_i(x; g)$ for all features i and all points x regardless of how f and g are implemented. Although seemingly trivial, this axiom does not necessarily hold for attribution methods that use the network's implementation or structure to generate attributions. Critically, this axiom also does not hold for the interaction method proposed by Tsang et al. (2017), which looks at the first layer of a feed-forward neural network. Two networks may represent exactly the same function but differ greatly in their first layer.

This axiom is trivially seen to hold for Integrated Hessians since it holds for Integrated Gradients. However, without this key axiom, attributions/interactions could encode information about unimportant aspects of model structure rather than the actual decision surface of the model.

2.2.4 Linearity Axiom

Integrated Gradients also satisfies an axiom called *linearity*. Given two networks f and g, consider the output of the weighted ensemble of both networks to be $af(x) + bg(x)$. Then, the attribution $\phi_i(x; af + bg)$ of the weighted ensemble equals the weighted sum of attributions $a\phi_i(x; f) + b\phi_i(x; g)$ for all features i and samples x. This important axiom preserves network linearity and facilitates easy computation of attributions for network ensembles.

We can generalize linearity to interactions using the *interaction linearity* axiom:

$$\Gamma_{i,j}(x; af + bg) = a\Gamma_{i,j}(x; f) + b\Gamma_{i,j}(x; g), \tag{13}$$

for any i, j and all points x. Given that $\Gamma_{i,j}$ is a composition of linear functions ϕ_i, ϕ_j in terms of the parameterized networks f and g, it is itself a linear function of the networks; therefore, Integrated Hessians satisfies *interaction linearity*.

2.2.5 Symmetry-Preserving Axiom

We say that two features x_i and x_j are *symmetric* with respect to f if swapping them does not change the output of f anywhere. That is, $f(\cdots, x_i, \cdots, x_j, \cdots) = f(\cdots, x_j, \cdots, x_i, \cdots)$. Sundararajan et al. (2017) shows that Integrated Gradients is *symmetry preserving*, that is, if x_i and x_j are symmetric with respect to f, and if $x_i = x_j$ and $x'_i = x'_j$ for some input x and baseline x', then $\phi_i(x) = \phi_j(x)$. To generalize to interaction values, if the same conditions as above hold, then $\Gamma_{k,i}(x) = \Gamma_{k,j}(x)$ for any feature x_k. This axiom, which holds since $\Gamma_{k,i}(x) = \phi_i(\phi_k(x))$ and ϕ_i, ϕ_j are symmetry-preserving, is significant because it indicates that if two features are functionally equivalent to a model, then they must interact in the same way with respect to that model.

2.2.6 Interaction Symmetry Axiom

Another form of symmetry is important to note: *interaction symmetry*. This axiom states that, for any i, j, we have $\Gamma_{i,j}(x) = \Gamma_{j,i}(x)$. Although simple, it guarantees that the interaction function itself is symmetric with respect to the features it explains. It is straightforward to show that existing neural networks and their activation functions have continuous second partial derivatives, which implies that Integrated Hessians satisfies interaction symmetry.[2]

2.3 Approximating Integrated Hessians in Practice

Our interaction values include a double integral, which is intractable to compute analytically in the general case. To compute Integrated Gradients in practice, Sundararajan et al. (2017) introduced the following discrete sum approximation:

2. Section 3 describes the special case of the ReLU activation function.

$$\hat{\phi}_i(x) = (x_i - x'_i) \times \sum_{\ell=1}^{k} \frac{\partial f(x' + \frac{\ell}{k}(x - x'))}{\partial x_i} \times \frac{1}{k}, \tag{14}$$

where k is the number of points used to approximate the integral. To compute Integrated Hessians, we introduce a similar discrete sum approximation:

$$\hat{\Gamma}_{i,j}(x) = (x_i - x'_i)(x_j - x'_j) \times \sum_{\ell=l}^{k} \sum_{p=1}^{m} \frac{\ell}{k} \times \frac{p}{m} \times \frac{\partial f(x' + (\frac{\ell}{k} \times \frac{p}{m})(x - x'))}{\partial x_i \partial x_j} \times \frac{1}{km}. \tag{15}$$

Typically, it is easiest to compute this quantity when $k = m$ and the number of samples drawn is thus a perfect square. However, when a non-square number of samples is preferred we can generate sample points from the product distribution of two uniform distributions so that the number is the largest perfect square above the desired number of samples; we can index the sorted samples appropriately to get the desired number. The preceding formula omits the first-order term in $\Gamma_{i,i}(x)$, but it can be computed using the same principle.

Expected Hessians has a similar, if slightly easier, form:

$$\hat{\Gamma}^{EG}_{i,j}(x) = (x_i - x'_i)(x_j - x'_j) \sum_{\ell}^{k} \zeta_\ell \times \frac{\partial f(x' + \zeta_\ell(x - x'))}{\partial x_i \partial x_j} \times \frac{1}{k}, \tag{16}$$

where ζ_ℓ is the ℓth sample from the product distribution of two uniform distributions. We find in general that less than 300 samples are required for any given problem to approximately satisfy interaction completeness. For most problems, far less than 300 suffices (e.g., around 50), but the number is model and data dependent: larger models and higher-dimensional data generally require more samples than smaller models and lower-dimensional data.

3. Smoothing ReLU Networks

One major limitation not discussed in previous approaches to interaction detection in neural networks relates to use of the ReLU activation function, $\text{ReLU}(x) = \max(0, x)$, in many popular neural network architectures. Neural networks using ReLU are piecewise linear and have second partial derivatives equal to zero in all places. Previous second-order approaches (based on the Hessian) fail to detect any interaction in ReLU-based networks.

Fortunately, the ReLU activation function has a smooth approximation – the SoftPlus function: $\text{SoftPlus}_\beta(x) = \frac{1}{\beta}\log(1 + e^{-\beta x})$. SoftPlus more closely approximates ReLU as β increases and has well-defined higher-order derivatives. Furthermore, Dombrowski et al. (2019) have proved that a model's outputs *and* first-order feature attributions are minimally perturbed when ReLU activations are replaced by SoftPlus activations in a trained network. Therefore, we can apply Integrated Hessians on a network with ReLU activations by first replacing ReLU with SoftPlus. We note that no re-training is necessary for this approach.

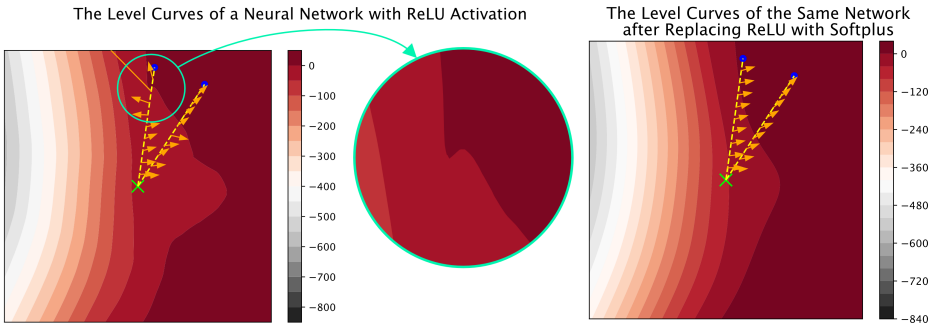

Figure 2: Replacing ReLU activations (left) with SoftPlus$_\beta$ activations (right) where $\beta = 10$ smooths the decision surface of a neural network: gradients tend to be more homogeneous along the integration path. Orange arrows show the gradient vectors at each point along the path from the reference (green x) to the input (blue dots). ReLUs can cause small bends in the output space with aberrant gradients.

In addition to being twice differentiable and letting us calculate interaction values in ReLU networks, replacing ReLU with SoftPlus offers other benefits when calculating interaction values. We show that smoothing a neural network (i.e., decreasing the value of β in the SoftPlus activation function) lets us accurately approximate the Integrated Hessians value with fewer gradient calls.

Theorem 1 *For a one-layer neural network with softplus$_\beta$ non-linearity, $f_\beta(x) = softplus_\beta(w^T x)$, and d input features, we can bound the number of interpolation points k needed to approximate the Integrated Hessians to a given error tolerance ϵ by $k \leq \mathcal{O}(\frac{d\beta^2}{\epsilon})$.*

The proof of Theorem 1 is shown in Appendix C. In addition to the proof for the single-layer case, Appendix C also presents empirical results to show that many-layered neural networks display the same property.

The intuition behind these results is that as we replace ReLU with SoftPlus, the decision surface of the network is smoothed (see Figure 2). We observe that the gradients tend to all have more similar direction along the path from reference to foreground sample once the network has been smoothed with SoftPlus replacement.

4. Explanation of XOR function

To explain why feature interactions can be more informative than feature attributions, we use the case of two binary features and a neural network representing an XOR function. This network has a large output when either feature is on alone, but it has a low magnitude output when both features are either on or off (Figure 3, left).

When we explain the network using Integrated Gradients with the zeros baseline, we see that samples where both features are on and those where both features are off get identical attributions (see Figure 3, middle). Integrated Hessians, however, differentiates these two samples by identifying the negative interaction that occurs between the two features when

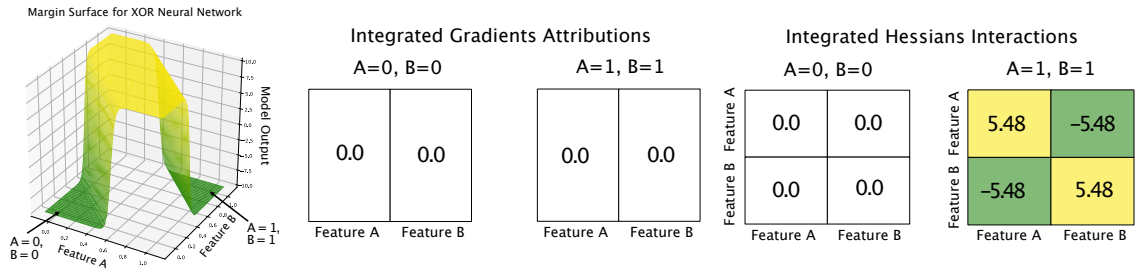

Figure 3: (Left) The margin surface for a neural network representing an XOR function. (Middle) Integrated Gradients feature attributions for two samples, the first where both features are turned off, and the second where both features are turned on. Both get identical Integrated Gradients attributions. (Right) Integrated Hessians feature interactions for the same two samples. We see that the Integrated Hessians values, but not the Integrated Gradients values, differentiate the two samples.

they are both on (Figure 3, right). Therefore, the interactions can usefully distinguish between (0,0), which has an output of 0 because it is identical to the baseline, and (1,1), which has an output of 0 because both features are on. When considered independently, each feature would increase the model's output, but when considered in interaction with one another, they cancel out the positive effects and drive the model's output back to the baseline.

This example also illustrates a problem with methods like Cui et al. (2019), which use the input Hessian without integrating over a path. In Figure 3, we see that the function is saturated at all points on the data manifold, meaning all elements of the Hessian will be 0 for all samples. In contrast, by integrating between the baseline and the samples, Integrated Hessians can correctly detect the negative interaction between the two features.

5. Empirical Evaluation

We empirically evaluated our method against other methods using benchmarks inspired by recent literature on quantitatively evaluating feature attribution methods (Adebayo et al., 2018; Kindermans et al., 2019; Hooker et al., 2019; Yeh et al., 2019; Lin et al., 2019). We compare Integrated Hessians to six existing methods: the Shapley Interaction Index (Grabisch and Roubens, 1999), the Shapley Taylor Interaction Index (Dhamdhere et al., 2019), ArchDetect (Tsang et al., 2020), Generalized Contextual Decomposition (Singh et al., 2018), Neural Interaction Detection (Tsang et al., 2017), and using the Hessian at the input sample (Cui et al., 2019).

5.1 Computation Time

First, we compared the computation time of each method. We explained interactions in a 5-layer neural network with SoftPlus activation and 128 units per layer. We ran each method on models with 5, 50 and 500 input features, and evaluated each method on 1000

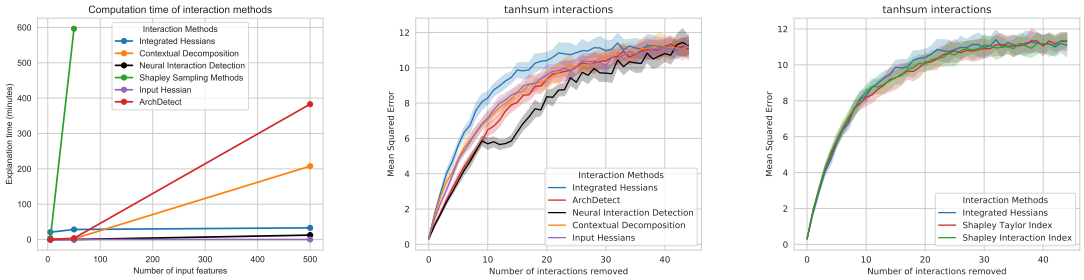

Figure 4: *Left*: The time each method takes to compute all pairwise interactions on 1000 samples with d features as a function of d. Existing methods scale poorly compared to our proposed method. We also benchmarked our methods against others (*Center*, heuristic methods; *Right*, Shapley value-based methods) using a modified version of Remove and Retrain (Hooker et al., 2019) on simulated interactions. Our method more accurately identifies the most important interactions than all existing methods other than the Shapley Interaction Index and Shapley Taylor Index, which are much more computationally expensive.

samples. Because computing the Shapley Interaction Index and Shapley Taylor Interaction Index is NP-hard in the general case (Elkind et al., 2009), we instead compared against a Monte Carlo estimation of the Shapley Interaction Index with 200 samples, analogous to how the Shapley Value is estimated in Kononenko et al. (2010). Even using a small number of samples, this comparison could not run to completion for the 500 feature case and would have taken an estimated *1000 hours* to complete. For each method, we computed all pairwise interactions: d^2 interactions for d features. Figure 4 (left) shows results.

We observe that our method is more tractable than the two other axiomatic approaches to interaction based on Shapley values. Further, we note that as the number of features grows, our method is more tractable than several of the heuristic methods, as well: other methods require at least $O(d^2)$ separate forward passes of the model to compute the interactions, which is non-trivial to parallelize. However, back-propagating the Hessian through the model is easily done in parallel on a GPU since this functionality already exists in modern deep learning frameworks (see Appendix B.3) (Paszke et al., 2019; Abadi et al., 2016).

5.2 Quantitative Comparison of Interaction Detection with Other Methods

To compare each method, we used the Remove and Retrain benchmark introduced in Hooker et al. (2019). The benchmark compares feature attributions by progressively ablating the most important features in each sample —ranked according to each attribution method —and then retraining the model on the ablated data and measuring the performance drop. The attribution method that incurred the fastest performance decline was the quickest to identifythe most predictive features in the data.

We generated a simulated regression task with 10 features where each feature was drawn independently from $\mathcal{N}(0, 1)$. The label was an additive sum of 20 interactions with random coefficients normalized to sum to 1, drawn without replacement from all possible pairs of features. A 3-layer neural network's predictions achieved over 0.99 correlation with the true label. We leave further details about experimental setup, as well as how we "remove" an interaction between two features, to Appendix B.4.1.

We compared each of the five interaction methods using Remove and Retrain on five different interaction types: $g_{\text{tanhsum}}(x_i, x_j) = \tanh(x_i + x_j)$, $g_{\text{cossum}}(x_i, x_j) = \cos(x_i + x_j)$, $g_{\text{multiply}}(x_i, x_j) = x_i * x_j$, $g_{\max}(x_i, x_j) = \max(x_i, x_j)$ and $g_{\min}(x_i, x_j) = \min(x_i, x_j)$. The results for g_{tanhsum} are displayed in Figure 4 (right). Our method most quickly identified the highest-magnitude interactions in the data, as demonstrated by the fastest increase in error. We include the remaining results in Appendix B.4 but note here that our method outperformed all existing heuristic methods on all interaction types and performed equivalently to both the Shapley Interaction Index and the Shapley Taylor Interaction Index.

In addition to the Remove and Retrain benchmark, we also tested our approach using the "sanity checks" proposed in Adebayo et al. (2018) to ensure that our interaction attributions were sensitive to network and data randomization. We found that our method passed both sanity checks (see Appendix B.4.3).

6. Applications of Integrated Hessians

6.1 NLP

Over the past decade, neural networks have been the go-to model for language tasks, from convolutional (Kim, 2014) to recurrent (Sundermeyer et al., 2012). More recently, large, pre-trained transformer architectures (Peters et al., 2018; Devlin et al., 2018) have achieved state-of-the-art performance on a wide variety of tasks. Previous work suggested investigating the internal weights of the attention mechanisms in attention-based models (Ghaeini et al., 2018; Lee et al., 2017; Lin et al., 2017; Wang et al., 2016). However, more recent work suggests that examining attention weights may not be a reliable way to interpret models with attention layers (Serrano and Smith, 2019; Jain and Wallace, 2019; Brunner et al., 2019). To resolve this issue, feature attributions have been applied to text classification models to understand which words most affect classification (Liu and Avci, 2019; Lai et al., 2019). However, these methods do not explain how words interact with their surrounding context.

We downloaded pre-trained weights for DistilBERT (Sanh et al., 2019) from the HuggingFace Transformers library (Wolf et al., 2019). We fine-tuned the model on the Stanford Sentiment Treebank data set (Socher et al., 2013), where the task was to predict whether a movie review had positive or negative sentiment. After 3 epochs of fine-tuning, DistilBERT achieved a validation accuracy of 0.9071 (0.9054 TPR / 0.9089 TNR).[3] We present further fine-tuning details in Appendix D.

In Figure 5, we show interactions generated by Integrated Hessians and attributions generated by Integrated Gradients on an example drawn from the validation set. The figure

3. This performance does not represent state-of-the-art, nor is sentiment analysis representative of the full complexity of existing language tasks. However, this paper focuses on explanation, and sentiment analysis is easily fine-tuned without extensive hyperparameter search.

demonstrates that DistilBERT learned intuitive interactions that would not be revealed from feature attributions alone. For example, a word like "painfully," which might have a negative connotation on its own, has a large positive interaction with the word "funny" in the phrase "painfully funny." In Figure 1, we demonstrate how interactions can help us understand why a fine-tuned DistilBERT model outperforms a simpler model: a convolutional neural network (CNN) that achieves an accuracy of 0.82 on the validation set. DistilBERT picks up positive interactions between negation words ("not") and negative adjectives ("bad") that a CNN fails to fully capture. Finally, in Figure 5, we use interaction values to reveal saturation effects: many negative adjectives describing the same noun interact positively. Although this may initially seem counter-intuitive, it reflects the structure of language. If a phrase has only one negative adjective, it stands prominently as the word that makes the phrase negative. At some point, however, describing a noun with an increasing number of negative adjectives makes any individual negative adjective less important towards classifying that phrase as negative.

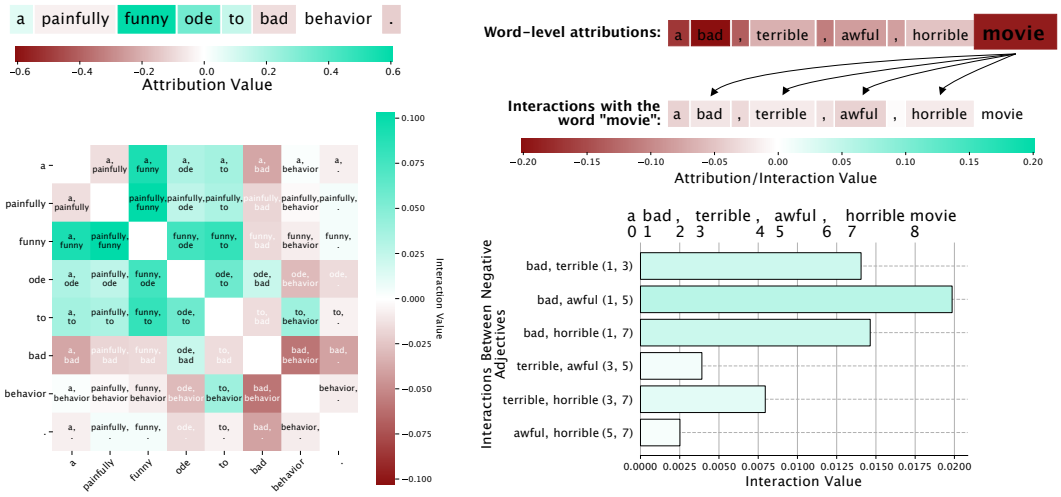

Figure 5: *Left*: Interactions in text reveal learned patterns such as the phrase "painfully funny" having a positive interaction despite the word "painfully" having a negative attribution. These interactions are not evident from attributions alone. *Right*: Interactions help reveal an unintuitive pattern in language models: saturation. Although the word "movie" interacts negatively with all negative modifying adjectives, those negative adjectives themselves all interact positively. The greater the number of negative adjectives in a sentence, the less each individual negative adjective contributes to the overall classification of the sentence.

6.2 Drug Combination Response Prediction

In the domain of anti-cancer drug combination response prediction, plotting Integrated Hessians can help to glean biological insights into the process we are modeling. We considered

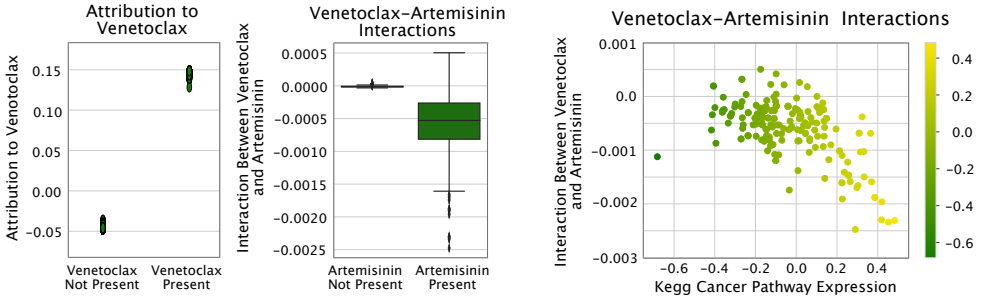

Figure 6: Left: Integrated Gradients values for Venetoclax. Middle: Venetoclax interactions with Artemisinin across all samples. Right: Venetoclax and Artemisinin interaction is driven by expression of genes in cancer samples.

one of the largest publicly available data sets measuring drug combination response in acute myeloid leukemia (Tyner et al., 2018). Each of the 12,362 samples consists of the measured response of a 2-drug pair tested on a patient's cancer cells. The 1,235 input features are split between features describing the drug combinations and those describing the cancerous cells, which we modeled using the neural architectures described in Hao et al. (2018) and Preuer et al. (2017).

According to the first-order explanations, the presence or absence of the drug Venetoclax in the drug combination was the most important feature. We also easily see that first-order explanations were inadequate in this case: while the presence of Venetoclax was generally predictive of a more responsive drug combination, the amount of positive response to this drug was predicted to vary across samples (see Figure 6, top left).

Integrated Hessians reveals that some of this variability is attributable to the drug with which Venetoclax was combined. The model learned a strong negative interaction between Venetoclax and Artemisinin (see Figure 6, middle), which we confirmed matched the ground truth ascertained from additional external data ($p = 2.31 \times 10^{-4}$, see Appendix F). Finally, we gained insight into the variability in the *interaction* values between Venetoclax and Artemisinin by plotting them against the expression level of a pathway containing cancer genes (see Figure 6, right). We found that patients with higher expression of this pathway tended to have a more negative interaction (sub-additive response) than those with a lower expression of it. Integrated Hessians enriches both our understanding of the interactions between drugs in our model and the genetic factors that influence this interaction.

7. Conclusion

We proposed a novel method called Integrated Hessians to explain feature interactions in neural networks. The interaction values we proposed have two natural interpretations: (1) as the effect of combining two features on a model's output, and (2) as the explanation of one feature's importance in terms of another. Our method provably satisfied common-sense axioms that previous methods did not and outperforms previous methods in practice at identifying known interactions on simulated data. Additionally, we demonstrated how

to glean interactions from neural networks trained with a ReLU activation function that has no second derivative. In accordance with recent work, we showed why replacing the ReLU activation function with the SoftPlus at explanation time was both intuitive and efficient. Finally, we performed several experiments to reveal the utility of our method, from understanding performance gaps between model classes to discovering patterns a model has learned on high-dimensional data.

We conclude that although feature attribution methods provide valuable insight into model behavior, such methods by no means end the discussion on interpretability. Rather, they encourage further work in deeper understanding model behavior. For example, our approach does not currently support the quantification of higher-order interactions between features. Future research to characterize such interactions would be valuable. For example, these methods may prove useful for image models, where individual pixels lack semantic meaning. Extending the efficiency or theoretical guarantees of existing methods that can detect these higher-order interactions, or developing new methods to provide these desiderata, may be of particular interest. Finally, while interactions add a degree of expressivity to feature attribution methods, finding approaches to explain models from outside the paradigm of feature attribution entirely would be creative and significant future research.

Acknowledgments

The results in Figure 6 are based upon data generated by the Cancer Target Discovery and Development (CTD2) Network (https://ocg.cancer.gov/programs/ctd2/data-portal) established by the National Cancer Institute's Office of Cancer Genomics. This work was funded by the National Science Foundation (DBI-1552309, DBI-1759487), American Cancer Society (RSG-14-257-01-TBG), and National Institutes of Health (R01 NIA AG 061132).

Appendix A. Deriving Interaction Values

Here, we derive the formula for Integrated Hessians from its definition: $\Gamma_{i,j}(x) = \phi_j(\phi_i(x))$. We start by expanding ϕ_j using the definition of Integrated Gradients:

$$\Gamma_{i,j}(x) := (x_j - x'_j) \times \int_{\beta=0}^{1} \frac{\partial \phi_i(x' + \beta(x - x'))}{\partial x_j} d\beta. \tag{17}$$

We consider the function $\frac{\partial \phi_i}{\partial x_j}(x)$, and we first assume that $i \neq j$

$$\frac{\partial \phi_i}{\partial x_j}(x) = \tag{18}$$

$$(x_i - x_i') \times \frac{\partial}{\partial x_j}\left(\int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha \right) = \tag{19}$$

$$(x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial}{\partial x_j}\left(\frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} \right) d\alpha = \tag{20}$$

$$(x_i - x_i') \times \int_{\alpha=0}^{1} \alpha \frac{\partial^2 f(x' + \alpha(x - x'))}{\partial x_i \partial x_j} d\alpha, \tag{21}$$

where we have assumed that the function f satisfies the conditions for the Leibniz Integral Rule (i.e., that integration and differentiation are interchangeable). These conditions require that the derivative of f, $\frac{\partial f}{\partial x_i}$ and its second derivative function $\frac{\partial^2 f}{\partial x_i \partial x_j}$ are continuous over x in the integration region, and that the bounds of integration are constant with respect to x. It is easy to see that the bounds of integration are constant with respect to x. It is also straightforward to see that common neural network activation functions — for example, $\text{sigmoid}(x) = \frac{1}{1+e^{-x}}$, $\text{softplus}_\beta(x) = \frac{1}{\beta}\log(1 + e^{-\beta x})$, or $\text{gelu}(x) = x\Phi(x)$, where $\Phi(x)$ is the cumulative distribution function of the normal distribution — have continuous first and second partial derivatives; this implies that compositions of these functions have continuous first and second partial derivatives, as well. Although this is not the case with the ReLU activation function, we discuss replacing it with SoftPlus in the main text.

We can proceed by plugging equation 21 into the original definition of $\Gamma_{i,j}(x)$:

$$\Gamma_{i,j}(x) := (x_j - x_j') \times \int_{\beta=0}^{1} \frac{\partial \phi_i(x' + \beta(x - x'))}{\partial x_j} d\beta = \tag{22}$$

$$(x_j - x_j') \times \int_{\beta=0}^{1} (x_i' - \beta(x_i - x_i') - x_i') \int_{\alpha=0}^{1} \alpha \frac{\partial^2 f(x' + \alpha(x' - \beta(x - x') - x'))}{\partial x_i \partial x_j} d\alpha d\beta = \tag{23}$$

$$(x_j - x_j')(x_i - x_i') \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta, \tag{24}$$

where all we've done is re-arrange terms.

Deriving $\Gamma_{i,i}(x)$ proceeds similarly:

$$\frac{\partial \phi_i}{\partial x_i}(x) = \tag{25}$$

$$\frac{\partial}{\partial x_i}\left((x_i - x_i')\right) \times \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha + \tag{26}$$

$$(x_i - x_i') \times \frac{\partial}{\partial x_i}\left(\int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha\right) =$$

$$\int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha + (x_i - x_i') \times \int_{\alpha=0}^{1} \alpha \frac{\partial^2 f(x' + \alpha(x - x'))}{\partial x_i \partial x_j} d\alpha, \tag{27}$$

using the chain rule. After similar re-arrangement, we arrive at

$$\Gamma_{i,i}(x) = (x_i - x_i') \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha\beta(x - x'))}{\partial x_i} d\alpha d\beta + \tag{28}$$

$$(x_i - x_i')^2 \times \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \alpha\beta \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta.$$

Appendix B. Comparing Against Existing Methods

We now elaborate on the relationship between our method and six existing methods:

- Integrated Hessians: Our proposed method.
- Input Hessian: Uses the Hessian at the input instance. Using the input Hessian at a particular instance to measure interaction values is the natural generalization of using the gradient to measure the importance of individual features, as done by Simonyan et al. (2013).
- Contextual Decomposition (CD): Introduced by Murdoch et al. (2018) for LSTMs and extended to feed-forward and convolutional architectures by Singh et al. (2018). We focus on the generalized version introduced by the latter.
- Neural Interaction Detection (NID): Introduced by Tsang et al. (2017), this method generated interactions by inspecting the weights of the first layer of a feed-forward neural network.
- Shapley Interaction Index (SII): Introduced by Grabisch and Roubens (1999) to allocate credit in coalitional game theory. It is used to explain interactions in neural networks similarly to how the Shapley value is used to explain attributions in Lundberg and Lee (2017).
- Shapley Taylor Interaction Index (STI): Introduced by Dhamdhere et al. (2019), this method is similar to the Shapley Interaction Index, but it changes the weighting of certain coalitions in order to naturally fulfill the axiom that interactions should sum to the output.

- ArchDetect (AD): Introduced by Tsang et al. (2020), this method detects interactions between pairs of features using a quantity similar to a discrete second-order derivative. See Appendix B.1 for more detail.
- Group Expected Hessian (GEH): Introduced by Cui et al. (2019), this method aggregates the input Hessian over many samples with respect to a Bayesian neural network.
- Deep Feature Interaction Maps (DFIM): Introduced by Greenside et al. (2018), this method determines interactions by seeing how much attributions change when features are perturbed.

We first discuss practical considerations regarding each method and then evaluate the degree to which each satisfies the axioms we identified in Section 2.2.

B.1 Comparison to Tsang et al. 2020

In Tables 1 and 2 of their recent paper, "How does this interaction affect me?: Interpretable attribution for feature interactions," Tsang et al. (2020) compare their "interaction attribution" method ArchAttribute to Integrated Hessians. We believe this is a faulty comparison. Their overall approach consists of three steps: (1) detect pairwise interactions between features using a method called ArchDetect, (2) use these pairwise interactions to cluster features into groups, such that interactions occur only between features within the same group, and (3) attribute importance to those groups of features using a method called ArchAttribute.

For a group $\mathcal{I}$ formed using the first two steps of this procedure, the ArchAttribute value is:

$$\phi(\mathcal{I}) = f(x^*_{\mathcal{I}} + x^{'}_{\setminus\mathcal{I}}) - f(x'), \tag{29}$$

where for p features in a data set, we let $\mathcal{I}$ be a subset of feature indices, $\mathcal{I} \subseteq \{1, 2, \cdots, p\}$. For a vector $x \in \mathbb{R}^p$, let $x_{\mathcal{I}} \in \mathbb{R}^p$ be defined as

$$(x_{\mathcal{I}})_i = \begin{cases} x_i, & \text{if } i \in \mathcal{I} \\ 0 & \text{otherwise.} \end{cases} \tag{30}$$

We let x^* represent the sample we want to explain, x' represent a neutral reference sample, and f be a black box model that takes a vector input and has a scalar output.

We prove why we believe this is a flawed comparison by showing that from a game-theoretic perspective, the value calculated by ArchAttribute corresponds to a Group Shapley Value rather than any sort of interaction value.

Theorem 2 *For a set of features $\mathcal{I}$ found using the first two steps of the procedure in Tsang et al. (2020), the ArchAttribute value $\Phi(\mathcal{I})$ is exactly the Group Shapley Value.*

Proof Define a value function $v : 2^N \mapsto \mathbb{R}$ as $v(S) = f(x^*_S + x^{'}_{\setminus S})$, where x^* is the sample we want to explain and $x^{'}$ is the all zeros vector. Therefore, we can write:

$$\Phi(\mathcal{I}) = v(\mathcal{I}) - v(\emptyset). \tag{31}$$

Because the Archipelago method defines $\mathcal{I}$ as a set of features that do not interact with features outside of the set $\mathcal{I}$, we have that:

$$v(\mathcal{I}) - v(\emptyset) = v(\mathcal{I} \cup S) - v(\emptyset \cup S), \forall S \subseteq N \setminus \mathcal{I}. \tag{32}$$

For *any* family of coefficients $\{p_S^{\mathcal{I}}(N)\}_{S \subseteq N \setminus \mathcal{I}}$ that define a probability distribution on $2^{N \setminus \mathcal{I}}$, since these coefficients will sum to 1, we can say:

$$v(\mathcal{I} \cup S) - v(\emptyset \cup S) = \sum_{S \subseteq N \setminus \mathcal{I}} p_S^{\mathcal{I}}(N)[v(\mathcal{I} \cup S) - v(S)]. \tag{33}$$

The R.H.S. of the last equation defines a "probabilistic generalized value," the most well-known of which is the Group Shapley Value (Marichal et al., 2007; Flores et al., 2019). ■

If *values* measure the individual power of a player in a cooperative game, *generalized values* extend this notion to coalitions of players (Marichal et al., 2007). Since the ArchAttribute method appears to be a method for calculating *group attributions* rather than *feature interactions*, we instead compare to the interaction detection method ArchDetect, also described in Tsang et al. (2020). The ArchDetect value for a pair of features i and j is given as

$$\bar{\omega}_{i,j} = \frac{1}{2}(\omega_{i,j}(x^*) + \omega_{i,j}(x^{'})), \tag{34}$$

where the quantities ω are defined:

$$\begin{aligned} \omega_{i,j}(x) = \big(\frac{1}{h_i h_j}\big(f(x^*_{\{i,j\}} + x_{\setminus\{i,j\}}) - f(x^{'}_{\{i\}} + x^*_{\{j\}} + x_{\setminus\{i,j\}}) \\ - f(x^*_{\{i\}} + x^{'}_{\{j\}} + x_{\setminus\{i,j\}}) + f(\boldsymbol{x}^{'}_{\{i,j\}} + x_{\setminus\{i,j\}})\big)\big)^2. \end{aligned} \tag{35}$$

If we substitute the more familiar set function notation, $v(S) = f(x^*_S + x^{'}_{\setminus S})$, we see that this value very closely resembles the Shapley Interaction Index (Grabisch and Roubens, 1999), where only two contexts are considered and the terms within the summation are squared. Hence, it is a significantly better comparison to make with our method.

B.2 Practical Considerations

This section describes four desirable properties that interaction methods may or may not satisfy:

1. Local: A method is local if it operates at the level of individual predictions. It is global if it operates over the entire data set. Which is better is task dependent, but local methods are often more flexible because they can be aggregated globally (Lundberg et al., 2020).

2. Architecture Agnostic: A method is architecture agnostic if it can be applied to any neural network architecture.

Interaction Method	Local	Architecture Agnostic	Data Agnostic	Higher-Order Interactions
Input Hessian	✓	✓	✓	
Integrated Hessians (ours)	✓	✓	✓	
CD (Singh et al., 2018)	✓	∼*	✓	✓
NID (Tsang et al., 2017)		✓	✓	✓
SII (Grabisch and Roubens, 1999)	✓	✓	✓	✓
STI (Dhamdhere et al., 2019)	✓	✓	✓	✓
AD (Tsang et al., 2020)	✓	✓	✓	
GEH (Cui et al., 2019)	✓		✓	
DFIM (Greenside et al., 2018)	✓	✓		

Table 1: Comparing the practical properties of existing interaction methods. Because GEH and DFIM are neither architecture nor data agnostic, respectively, we omit them from empirical comparisons. *Contextual Decomposition was originally introduced for LSTMs in Murdoch et al. (2018), and was generalized to feed-forward and convolutional architectures in Singh et al. (2018). However, the method has yet to be adapted to other architectures (e.g., transformers Devlin et al. (2018)).

3. Data Agnostic: A method is data agnostic if it can be applied to any type of data, no matter the structure.

4. Higher-Order Interactions: This paper primarily discusses interactions between pairs of features. However, some methods can generate interactions between groups of features larger than 2, and these are said to generate higher-order interactions.

In Table 1, we depict methods and the properties they satisfy. We note that GEH and DFIM are neither architecture nor data agnostic, respectively. Therefore, we do not include them in empirical comparisons since they cannot be run on feed-forward neural networks with arbitrary data. We also note that our method does not generate higher order interactions. Although in principle one could generate kth order interactions by recursively applying integrated gradients to itself k times, we do not discuss doing so in this paper.

B.3 Theoretical Considerations

We now evaluate which methods satisfy the axioms we presented in Section 2.2. Results are presented in Table 2. We note that Integrated Hessians and the Shapley Interaction Index share many theoretical properties, except that the Shapley Interaction Index trades completeness for the recursive axioms, which state that higher order interactions should be recursively defined from lower order ones. Whether this is preferable to satisfying completeness seems subjective.

The Shapley-Taylor Interaction Index satisfies completeness, although Dhamdhere et al. (2019) refer to it as the "efficiency" axiom. They also introduce a new "interaction distribution" axiom, which guarantees that interaction effects between a set of featuers are not improperly credited towards any proper subset of the original set. However, their inter-

Interaction Method	Completeness	Interaction Sensitivity	Implementation Invariant
Input Hessian			✓
Integrated Hessians (ours)	✓	✓	✓
CD (Singh et al., 2018)		∼**	
NID (Tsang et al., 2017)			
AD (Tsang et al., 2020)		✓	✓
SII (Grabisch and Roubens, 1999)		✓	✓
STI (Dhamdhere et al., 2019)	✓	✓	✓
GEH (Cui et al., 2019)			✓
DFIM (Greenside et al., 2018)			∼***

Interaction Method	Symmetry Preserving	Interaction Linearity	Recursive Axioms*
Input Hessian	✓	✓	
Integrated Hessians (ours)	✓	✓	
CD (Singh et al., 2018)	✓		
NID (Tsang et al., 2017)			
AD (Tsang et al., 2020)	✓		
SII (Grabisch and Roubens, 1999)	✓	✓	✓
STI (Dhamdhere et al., 2019)	✓	✓	
GEH (Cui et al., 2019)		✓	
DFIM (Greenside et al., 2018)	∼***	∼***	

Table 2: Comparing the theoretical guarantees of interaction methods. We define each axiom in Section 2.2. *The recursive axioms are discussed in Grabisch and Roubens (1999) and guarantee that higher-order interactions satisfy a specific recurrence relation. **Contextual Decomposition does not satisfy the exact sensitivity axiom we present, but it does guarantee that features equal to 0 will have zero interaction with any other feature. ***Whether or not Deep Feature Interaction Maps (DFIM) satisfies these properties relies on the underlying feature attribution method used: if the underlying method satisfies the properties, so does DFIM.

action index does not provide a relationship between attribution values (Shapley values) and interaction values, while our method does. Which proprety is more desirable is, again, subjective.

The Hessian satisfies implementation invariance for the same reason that our method does: it relies only on the value of the network function and its derivatives. The hessian also satisfies (1) linearity since the derivative is a linear operator, and (2) symmetry preservation due to Schwarz's theorem. Contextual decomposition is symmetry preserving by definition but fails on implementation invariance due to the way it splits the bias.

In terms of computational efficiency, we can better appreciate the benefit of our approach compared to the Shapley Interaction Index and the Shapley Taylor interaction index by examining the asymptotic complexity of each algorithm. For a given neural network, forward and backward passes have the same asymptotic time complexity, which is a function of the

number of nodes and layers (matrix multiplications) as well as the activation functions used (Rojas, 2013). Since the network itself does not change with the interaction method, we can therefore analyze each algorithm on the basis of the number of either forward or backward passes required to calculate all interactions. For each interaction, both the Shapley Interaction Index and the Shapley Taylor interaction index require considering an exponential number of subsets and require $O(2^d)$ forward passes. To calculate *all* interactions, these methods hence take $O(2^d d^2)$ forward passes. Integrated Hessians reduces the number of passes to be polynomial in the number of features, $O(kd^2)$, requiring k interpolation steps. For further discussion of the dependence of k on the smoothness of the function being explained, see Appendix C. Finally, heuristic methods like ArchDetect or Contextual Decomposition require $O(d^2)$ forward passes.

Although the Integrated Hessians method occupies an intermediate position in terms of asymptotic analysis, in practice we can calculate all possible pairs of interactions more quickly using this method due to the parallelization built into popular deep learning frameworks, back-propagating the Hessian through the model in parallel using GPUs (Paszke et al., 2019; Abadi et al., 2016).

B.4 Quantitative Evaluations

We now elaborate on the quantitative comparisons presented in the main text. Note that we do not compare with GEH and DFIM because they are not applicable to feed-forward neural networks with continuous data.

B.4.1 Remove and Retrain

The Remove and Retrain benchmark starts with a trained network on a given data set. It ranks the features most important for prediction on every sample in the data set according to a given feature attribution method. Using the ranking, it iteratively ablates the most important features in each sample and then re-trains the model on the ablated data. To run this method, it is necessary to ablate a feature, which the original paper does by mean or zero imputation (Hooker et al., 2019). However, doing so presents a problem when trying to compare methods that explain *interactions* between features rather than *attributions* to features.

Unlike ablating features, it is not straightforward to ablate an interaction between two features. Ablating both features in the interaction does not work, because it mixes main effects with interactions. Consider the function $f(x_1, x_2, x_3, x_4) = x_1 + x_2 + 0.1 * x_3 * x_4$. For $f(1.0, 1.0, 1.0, 1.0) = 2.1$, the largest and only interaction is between x_3 and x_4. However, ablating the pair x_1, x_2 would incur a larger performance hit because the features x_1 and x_2 have larger main effects than x_3 and x_4.

Instead, we opt to generate simulated data where the interactions are *known* and then ablate the interactions in the labels directly. We generate a simulated regression task with 10 features, where each feature is drawn independently from $\mathcal{N}(0, 1)$. The label is an additive sum of 20 interactions with random coefficients, drawn without replacement from all possible pairs of features. That is, for some specified interaction function g, we generate

the label y as

$$\sum_{i=1}^{20} \alpha_i g(x_{i,1}, x_{i,2}),$$

where $x_{i,1}, x_{i,2}$ is the pair of features chosen to be part of the ith interaction and α_i are random coefficients drawn from a uniform distribution and normalized to sum to 1:

$$\sum_{i=1}^{20} \alpha_i = 1$$

.

To ablate an interaction, we multiply the interaction in the label with gaussian noise, which ensures ablating the largest interactions adds the most amount of noise to the label. For example, let $f(x_1, x_2, x_3) = 0.5 * g(x_1, x_2) + 0.3g(x_1, x_3)$. To ablate the interaction between x_1 and x_2, we compute the ablated label $\hat{f}(x_1, x_2, x_3) = \epsilon * 0.5 * g(x_1, x_2) + 0.3g(x_1, x_3)$, where $\epsilon \sim \mathcal{N}(0, 1)$. The interaction method that identifies the largest interactions in the data adds the largest amount of random noise into the label, thus increasing error the fastest.

For the simulated data set described in Section 5.2, we trained a 3 layer neural network with 64 hidden units each and ReLU activation. It achieved near-perfect performance on the simulated regression task, explaining over 99% of variance in the label. For each progressive ablation of an interaction, we retrained the network 5 times and re-evaluated performance. We then plotted the mean and standard deviation of performance on a held-out set, as recommended by the original paper (Hooker et al., 2019). In Section 5.5, we showed the results for $g_{\text{tanhsum}}(x_i, x_j) = \tanh(x_i + x_j)$. We show results for these four additional interaction types:

- $g_{\text{cossum}}(x_i, x_j) = \cos(x_i + x_j)$
- $g_{\text{multiply}}(x_i, x_j) = x_i * x_j$
- $g_{\text{maximum}}(x_i, x_j) = \max(x_i, x_j)$
- $g_{\text{minimum}}(x_i, x_j) = \min(x_i, x_j)$

The results, in Figure 7 and Figure 8, show that our method consistently outperforms all other methods except the Monte Carlo estimation of the Shapley Interaction Index, which performs as well as our method. As discussed in Section 5.1, our method is much more computationally tractable than the Shapley Interaction Index, even using Monte Carlo estimation. We also observe that Neural Interaction Detection (Tsang et al., 2017) is not a local interaction method; rather, it detects interactions globally. To compare with it, we simply ablate the top-ranked interaction globally in all samples.

B.4.2 Impact of baseline

To demonstrate the impact of baseline selection on interaction detection, we re-ran our benchmark for interaction detection using four different choices of baseline. The first single baseline, called "min," sets each element in the baseline vector to its minimum observed

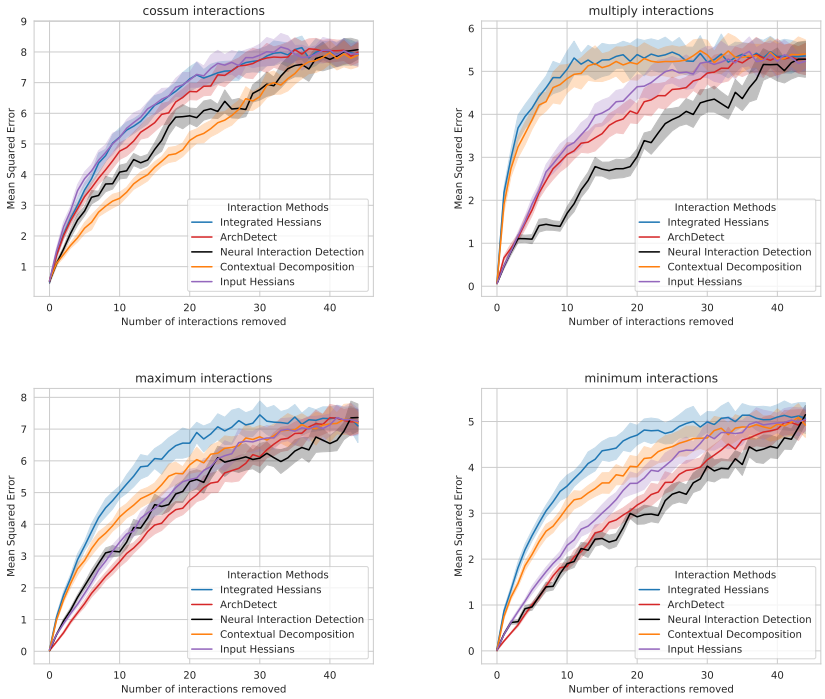

Figure 7: Additional comparisons with heuristic methods on known simulated interactions. The charts plot the mean and standard deviation across 5 re-trainings of the model as a function of ablated interactions. Our method (in blue) outperforms all existing heuristic methods on all interaction types.

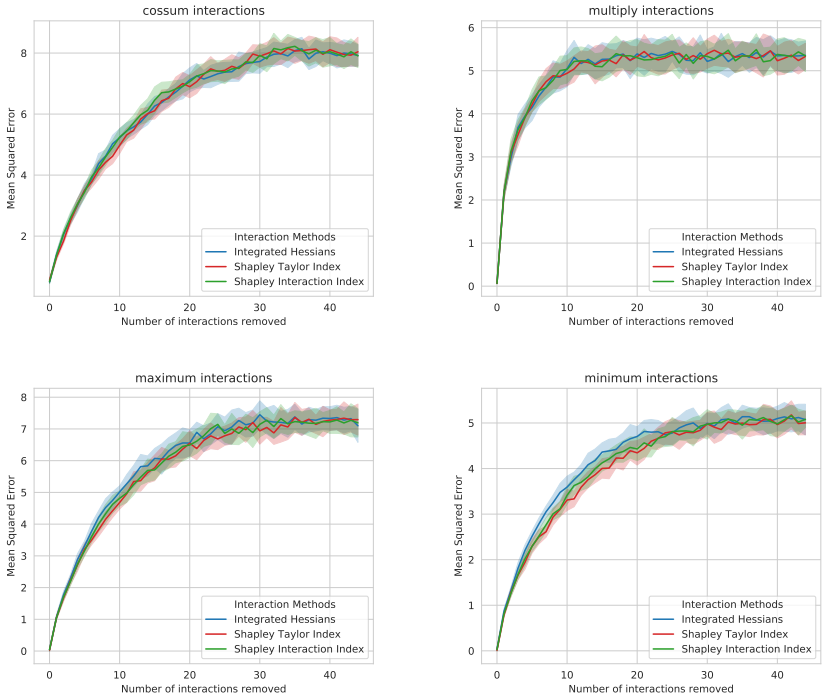

Figure 8: Additional comparisons with Shapley value-based methods on known simulated interactions (Top Left: cossum, Top Right: multiplicative, Bottom Left: maximum, Bottom Right: minimum). The charts plot the mean and standard deviation across 5 re-trainings of the model as a function of ablated interactions. Our method (in blue) performs comparably to the Shapley value-based methods on all interaction types but is significantly more computationally efficient.

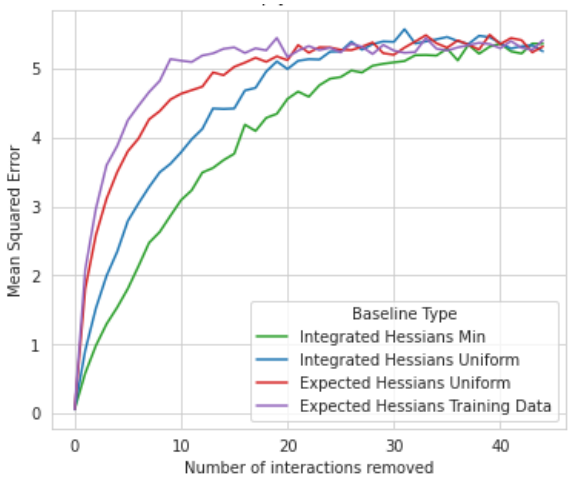

Figure 9: Using one of our synthetic data sets (with pairwise multiplicative interaction label), we demonstrate how for certain tasks, the choice of baseline can impact the quality of interaction detection. Both methods that take an expectation over distributions of baselines outperform single baseline methods, while using the training set as the distribution of baselines outperforms sampling baselines from the product of uniform distributions over the range of each individual feature.

value from the training data. It is similar to using the black image in image attribution. The second, "uniform," is a single randomly selected baseline from the product of uniform distributions ranging from the minimum to maximum observed values of each feature for the training data. The third baseline is a random sample (expected hessians) over baselines drawn from the product of uniform distributions mentioned above. The fourth is a random sample over baselines drawn from the training distribution (the approach we defined in Section 2.1). In Figure 9 we note that (1) both approaches that took an expectation over multiple baselines outperformed the single baseline options, and (2) that sampling baselines from the training data outperformed sampling baselines from the product of uniform distributions covering the feature ranges.

B.4.3 Sanity Checks

To ensure that our interaction attributions were sensitive to network and data randomization, we tested our approach using the "sanity checks" proposed in Adebayo et al. (2018). For this experiment, we used a data set of synthetic features, which consisted of five 0-mean unit-variance independent Gaussian random variables. The synthetic label was the sum of each of the ten possible pairwise multiplicative interactions between features with coefficients ranging in magnitude from 10 to 1. For our model, we used a neural network with two hidden layers and Tanh non-linearities until convergence (validation set $> R^2 : 0.99$). We next fit a network and found the interactions using Integrated Hessians. We then compared the rank correlation of these interactions with the interactions attained from explaining (1) a network with randomly initialized weights and (2) a network trained to convergence

on the training set on data where labels were shuffled at random (but the features were the same). In *both* settings, we found that the Spearman correlation between the true and randomized interactions was 0. This indicates that our method passed the sanity checks in Adebayo et al. (2018).

Appendix C. Effects of Smoothing ReLU Networks

C.1 Proof of Theorem 1

Theorem 1 For a one-layer neural network with softplus_β non-linearity, $f_\beta(x) = \text{softplus}_\beta(w^T x)$, and d input features, we can bound the number of interpolation points k needed to approximate the Integrated Hessians to a given error tolerance ϵ by $k \leq \mathcal{O}(\frac{d\beta^2}{\epsilon})$.

Proof As pointed out in Sundararajan et al. (2017) and Sturmfels et al. (2020), completeness can be used to assess the convergence of the approximation. We first show that decreasing β improves convergence for Integrated Gradients. To accurately calculate the Integrated Gradients value Φ for a feature i, we need to bound the error between the approximate computation and exact value. The exact value is given as:

$$\Phi_i(\theta, x, x') = (x_i - x_i') \times \int_{\alpha=0}^{1} \frac{\partial f(x'(1-\alpha) + x\alpha)}{\partial x_i} d\alpha. \tag{36}$$

To simplify notation, we can define the partial derivative that we want to integrate over in the ith coordinate as $g_i(x) = \frac{\partial F(x)}{\partial x_i}$:

$$\Phi_i(\theta, x, x') = (x_i - x_i') \times \int_{\alpha=0}^{1} g_i(x'\alpha + x(1-\alpha)) d\alpha. \tag{37}$$

Since the single-layer neural network with SoftPlus activation is monotonic along the path, the error in the approximate integral can be lower bounded by the left Riemann sum L_k:

$$L_k = \frac{\|x - x'\|}{k} \sum_{i=0}^{k-1} g_i(x' + \frac{i}{k}(x - x')) \leq \int_{\alpha=0}^{1} g_i(x'\alpha + x(1-\alpha)) d\alpha \tag{38}$$

and can likewise be upper-bounded by the right Riemann sum R_k:

$$\int_{\alpha=0}^{1} g_i(x'\alpha + x(1-\alpha)) d\alpha \leq R_k = \frac{\|x - x'\|}{k} \sum_{i=1}^{k} g_i(x' + \frac{i}{k}(x - x')). \tag{39}$$

We can then bound the magnitude of the error between the Riemann sum and the true integral by the difference between the right and left sums:

$$\epsilon \leq |R_k - L_k| = \frac{\|x - x'\|_2}{k} |g_i(x) - g_i(x')|. \tag{40}$$

By the mean value theorem, we know that for some $\eta \in [0, 1]$ and $z = x' + \eta(x - x')$, $g_i(x) - g_i(x') = \nabla_x g_i(z)^\top (x - x')$. Therefore:

$$\epsilon \leq \frac{\|x - x'\|}{k} \nabla_x g_i(z)^\top (x - x'). \tag{41}$$

Rewriting in terms of the original function, we have:

$$\epsilon \leq \frac{\|x - x'\|}{k} \sum_{j=0}^{d} \left(\frac{\partial^2 f(z)}{\partial x_i x_j} (x_j - x'_j) \right). \tag{42}$$

We can then consider the gradient vector of $g_i(x)$:

$$\nabla_x g_i(x) = \left[\frac{\beta w_0 e^{\beta w^T x}}{(e^{\beta w^T x} + 1)^2}, \frac{\beta w_1 e^{\beta w^T x}}{(e^{\beta w^T x} + 1)^2}, \cdots \right] \tag{43}$$

where each coordinate is maximized at the zeros input vector and takes a maximum value of $\beta w_i/4$. We can therefore bound the error in convergence as:

$$\epsilon \leq \frac{\|x - x'\|}{k} \sum_{j=0}^{d} \left(\frac{\beta \|w\|_\infty}{4} (x_j - x'_j) \right). \tag{44}$$

Ignoring the dependency on path length and the magnitude of the weights of the neural network, we see that:

$$k \leq \mathcal{O}\left(\frac{d\beta}{\epsilon} \right). \tag{45}$$

This demonstrates that the number of interpolation points k necessary to achieve a set error rate ϵ decreases as the activation function is smoothed (the value of β decreases). While this proof bounds the error in the approximation of the integral for a single feature, we get the error in completeness by multiplying by an additional factor of d features.

We can extend the same proof to Integrated Hessians values. We first consider the error for estimating off-diagonal terms $\Gamma_{i,j}, i \neq j$. The true value we are trying to approximate is given as:

$$\Gamma_{ij} = (x_i - x'_i)(x_j - x'_j) \times \int_{\alpha\beta} \frac{\partial^2 f(x' + \alpha\beta(x - x'))}{\partial x_i \partial x_j} d\alpha d\beta. \tag{46}$$

For simplicity of notation, we can say $h_{ij}(x) = \frac{\partial^2 F(x)}{\partial x_i \partial x_j}$. Assuming that we are integrating from the all-zeros baseline (as suggested in Sundararajan et al. (2017)), since $h_{ij}(x) =$ is monotonic on either interval from the 0 baseline, we can again bound the error in the double integral by the magnitude of the difference in the left and right Riemann sums:

$$\epsilon \leq \left| \frac{\|x - x'\|_2^2}{k} \sum_{j=1}^{k} \sum_{i=1}^{k} h_{ij}(x' + \frac{ij}{k}(x - x')) - \frac{\|x - x'\|_2^2}{k^2} \sum_{j=0}^{k-1} \sum_{i=0}^{k-1} h_{ij}(x' + \frac{ij}{k}(x - x')) \right|, \tag{47}$$

$$\epsilon \leq \frac{\|x - x'\|_2^2}{k} \left| \left(h_{ij}(x) + 2 \sum_{i=1}^{k-1} h_{ij}(x' + \frac{i}{\sqrt{k}}(x - x')) \right) - \left(h_{ij}(x') + 2 \sum_{i=1}^{k-1} h_{ij}(x') \right) \right|. \tag{48}$$

We can then use monotonicity over the interval to say that $h_{ij}(x' + \frac{i}{\sqrt{k}}(x - x')) < h_{ij}(x)$, which gives us:

$$\epsilon \leq \frac{(2k-1)\|x - x'\|_2^2}{k}\Big|h_{ij}(x) - h_{ij}(x')\Big|. \tag{49}$$

By the mean value theorem, we know that for some $\beta \in [0, 1]$, $h_{ij}(x) - h_{ij}(x') = \nabla_x h_{ij}(\beta)^\top (x - x')$. Substituting gives us:

$$\epsilon \leq \frac{(2k-1)\|x - x'\|_2^2}{k}\nabla_x h_{ij}(\beta)^\top (x - x'). \tag{50}$$

We can then consider the elements of the gradient vector:

$$\nabla_x h_{ij}(x) = \Bigg[- \frac{\beta^2 w_i w_j w_1 e^{\beta w^T x}\big(e^{\beta w^T x} - 1\big)}{(e^{\beta w^T x} + 1)^3}, - \frac{\beta^2 w_i w_j w_2 e^{\beta w^T x}\big(e^{\beta w^T x} - 1\big)}{(e^{\beta w^T x} + 1)^3}, \cdots \Bigg]. \tag{51}$$

For the univariate version of each coordinate, we can maximize the function by taking the derivative with respect to x and setting it equal to 0:

$$\frac{d}{dx}\Bigg(- \frac{\beta^2 e^{\beta x}\big(e^{\beta x} - 1\big)}{(e^{\beta x} + 1)^3} \Bigg) = \frac{\beta^3 e^{\beta x}(-4e^{\beta x} + e^{2\beta x} + 1)}{(e^{\beta x} + 1)^4} = 0. \tag{52}$$

We can see that this equation holds only when $(-4e^{\beta x} + e^{2\beta x} + 1) = 0$, and we can solve it by finding the roots of this quadratic equation, which occur when $x = \frac{1}{\beta}\log(2 \pm \sqrt{3})$. When we plug that back in, we find the absolute value of the function in that coordinate takes a maximum value of $\frac{\beta^2}{6\sqrt{3}}$. Therefore, for a given set of fixed network weights, we observe that the coordinate-wise maximum magnitude of $\nabla_x h_{ij}, \propto \beta^2$ and that the number of interpolation points necessary to reach a desired level of error in approximating the double integral decreases as β is decreased. Again ignoring the fixed weights and path length, the number of interpolation points necessary is bounded by

$$k \leq \mathcal{O}\Bigg(\frac{d\beta^2}{k}\Bigg). \tag{53}$$

For the $i = j$ terms (main effect terms), the error will have another additive factor of β, since main effect has an additional term equal to:

$$(x_i - x_i') \int_{\beta=0}^{1} \int_{\alpha=0}^{1} \frac{\partial f(x' + \alpha\beta(x - x'))}{\partial x_i} d\alpha d\beta. \tag{54}$$

When we bound the error in this approximate integral by the difference between the double left sum and double right sum, we find that:

$$\epsilon \leq \frac{(2k-1)\|x - x'\|_2^2}{k}\big|g_i(x) - g_i(x')\big|. \tag{55}$$

Following the exact same steps as in Equation 40 through Equation 44, we can then show the bound on the error of the on-diagonal terms will have an additional term that is $\propto \beta$. Due to the axiom of interaction completeness, the error bound of the entire convergence can be obtained by summing all individual terms, incurring another factor of d^2 in the bound. ■

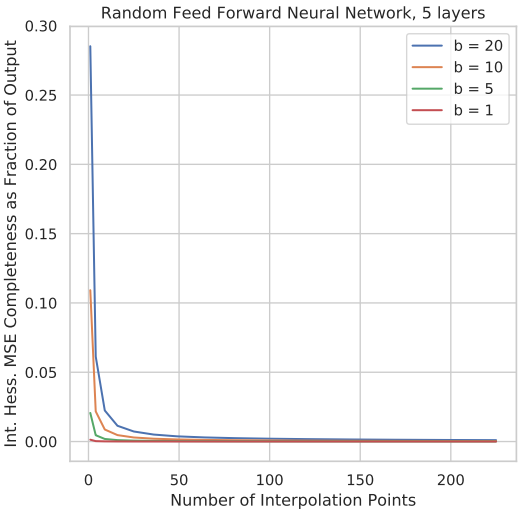

Figure 10: 5-Layer Network Results. Interaction completeness error (difference between model output and sum of Integrated Hessians values) decreases more quickly with the number of interpolation points as the β parameter for the SoftPlus activation function is decreased (as the function is smoothed). Results are averaged over 10 samples with 100 features for a neural network with 5 hidden layers of 50 nodes each.

C.2 SoftPlus Activation Empirically Improves Convergence

In addition to theoretically analyzing the effects of smoothing the activation functions of a single-layer neural network on the convergence of the approximate values of Integrated Gradients and Integrated Hessians, we wanted to empirically analyze the same phenomenon in deeper networks. We first created two networks: one with 5 hidden layers of 50 nodes, and a second with 10 hidden layers of 50 nodes. We then randomly initialized these networks using the Xavier Uniform intialization scheme (Glorot and Bengio, 2010). We created 10 samples to explain, each with 100 features drawn at random from the standard normal distribution. To evaluate the convergence of our approximate Integrated Hessians values, we plotted the interaction completeness error (the difference between model output and the sum of Integrated Hessians values) as a fraction of the magnitude of the function output. As we decreased the value of β, we smoothed the activations. We observed that the number of interpolations required to converge decreased (see Figure 10 and Figure 11). Note that the randomly initialized weights of each network were held constant, and only the value of β in the activation function changed.

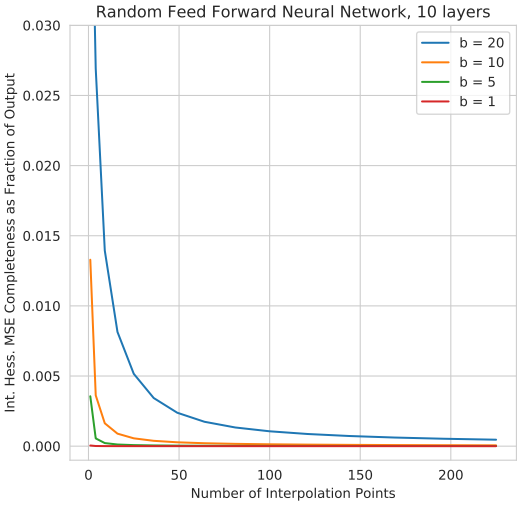

Figure 11: 10-Layer Network Results. Decreasing β has an even more dramatic effect on convergence when the network is deeper than in the 5-layer case. Again, this result shows that the interaction completeness error decreases more quickly with the number of interpolation points as the activation function is smoothed. Results are averaged over 10 samples with 100 features for a neural network with 10 hidden layers of 50 nodes each.

Appendix D. Details on the Sentiment Analysis Task

D.1 Fine-Tuning DistilBERT

As mentioned in Section 6.1, we downloaded pre-trained weights for DistilBERT, a pre-trained language model introduced in Sanh et al. (2019), from the HuggingFace Transformers library (Wolf et al., 2019). We fine-tuned the model on the Stanford Sentiment Treebank data set introduced by Socher et al. (2013). We fine-tuned for 3 epochs using a batch size of 32 and a learning rate of 0.00003. We used a a max sequence length of 128 tokens, and the Adam algorithm for optimization (Kingma and Ba, 2014). We tokenize using the HuggingFace uncased tokenizer. We did not search for these hyper-parameters; rather, they were the defaults presented for fine-tuning in the HuggingFace repository. We found that they worked adequately for our purposes, so we did not attempt to search through additional hyperparameters.

D.2 Training a CNN

The convolutional neural network we used for comparison in Section 6.1 was trained from scratch on the same data set. We randomly initialized 32-dimensional embeddings and used a max sequence length of 52. First, we applied dropout to the embeddings, with a dropout rate of 0.5. The network itself was composed of 1D convolutions with 32 filters of size 3 and 32 filters of size 8. Each filter size was applied separately to the embedding layer, after which max pooling with a stride of 2 was applied; the output of both convolutions was then concatenated and fed through a dropout layer, with a dropout rate of 0.5 during training. A hidden layer of size 50 followed the dropout, finally followed by a linear layer generating a scalar prediction to which the sigmoid function was applied.

We trained with a batch size of 128 for 2 epochs and used a learning rate of 0.001. We optimized using the Adam algorithm with the default hyper-parameters (Kingma and Ba, 2014). Since this model was not pre-trained on a large language corpus and lacks the expressive power of a deep transformer, it cannot capture patterns like negation that a fine-tuned DistilBERT can.

D.3 Generating Attributions and Interactions

To generate attributions and interactions, we used Integrated Gradients and Integrated Hessians with the zero-embedding baselinne, i.e., the embedding produced by the all zeros vector, which normally encodes the padding token. Because embedding layers are not differentiable, we generated attributions and interactions to the word embeddings and then summed over the embedding dimension to get word-level attributions and interactions, as done in Sundararajan et al. (2017). When computing attributions and interactions, we used 256 background samples. Because DistilBERT uses the GeLU activation function (Ramachandran et al., 2017), which has continuous first and second partial derivatives, there was no need to use the SoftPlus replacement. When we plotted interactions, we avoided plotting the main-effect terms in order to better visualize the interactions between words.

D.4 Additional Examples of Interactions

Here, we include additional examples of interactions learned on the sentiment analysis task. First, we expand upon the idea of saturation in natural language, displayed in Figure 12. We display interactions learned by a fine-tuned DistilBERT on the following phrases: "a bad movie" (negative with 0.9981 confidence), "a bad, terrible movie" (negative with 0.9983 confidence), "a bad, terrible, awful movie" (negative with 0.9984 confidence), and "a bad, terrible, awful, horrible movie" (negative with 0.9984 confidence). The confidence of the network saturates: a network output can get only so negative before it begins to flatten. However, the number of negative adjectives in the sentence increases. This means a sensible network would spread the same amount of credit (because the attributions sum to the saturated output) across a larger number of negative words, which is exactly what DistilBERT does. However, each word then gets less negative attribution than it would if it were alone. Thus, negative words have positive interaction effects, which is exactly what we see from the figure.

In Figure 13, we show another example of the full interaction matrix on a sentence from the validation set. In Figure 14, we present an example of how explaining the importance of a particular word can indicate whether that word is important because of its main effect or because of its surrounding context. We show additional examples from the validation set in Figures 15, 16, 17, 18, 19. Note that while some interactions make intuitive sense to humans ("better suited" being negative or "good script" being positive), many others are less intuitive. These interactions could indicate that the Stanford Sentiment Treebank data set does not fully capture the expressive power of language (e.g., it does not have enough samples to fully represent all possible interactions in language), or they could indicate that the model has learned higher order effects that cannot be explained by pairwise interactions alone.

Appendix E. Additional Experiments

E.1 Heart Disease Prediction

We aggregated interactions learned from many samples in a clinical data set and used them to reveal global patterns. We examined the Cleveland heart disease data set (Detrano et al., 1989; Das et al., 2009). After preprocessing, the data set contained 298 patients with 13 associated features, including demographic information like age and gender and clinical measurements such as systolic blood pressure and serum cholesterol. The task was to predict whether a patient had coronary artery disease. The list of features, which we reproduce here, is from Detrano et al. (1989), the original paper introducing the data set:

1. Age of patient (mean: 54.5 years $\pm$ standard deviation: 9.0)
2. Gender (202 male, 96 female)
3. Resting systolic blood pressure (131.6 mm Hg $\pm$ 17.7)
4. Cholesterol (246.9 mg/dl $\pm$ 51.9)
5. Whether or not a patient's fasting blood sugar was above 120 mg/dl (44 yes)

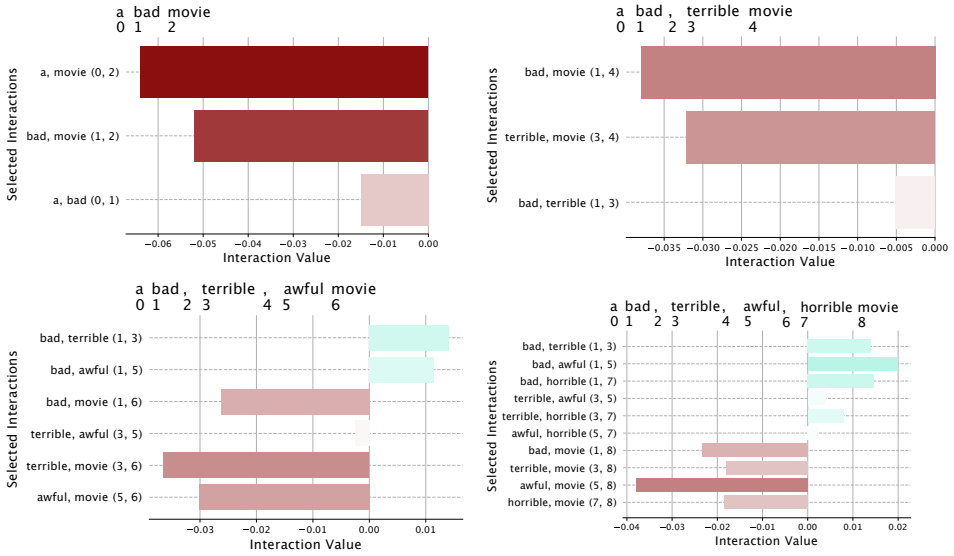

Figure 12: The effects of increasing saturation. As we add more negative adjectives to describe the word "movie," they interact increasingly positively even though they interact negatively with the word they describe. This is because each individual negative adjective has less impact on the overall negativity of the sentence the more negative adjectives there are.

6. Maximum heart rate achieved by exercise (149.5 bpm ± 23.0)

7. Whether or not a patient has exercise-induced angina (98 yes)

8. Excercise-induced ST-segment depression (1.05 mm ± 1.16)

9. Number of major vessels appearing to contain calcium as revealed by cinefluoroscopy (175 patients with 0, 65 with 1, 38 with 2, 20 with 3)

10. Type of pain a patient experienced if any (49 experienced typical anginal pain, 84 experienced atypical anginal pain, 23 experienced non-anginal pain and 142 patients experienced no chest pain)

11. Slope of peak exercise ST segment (21 patients had upsloping segments, 138 had flat segments, 139 had downsloping segments)

12. Whether or not a patient had thallium defects as revealed by scintigraphy (2 patients with no information available, 18 with fixed defects, 115 with reversible defects and 163 with no defects)

13. Classification of resting electrocardiogram (146 with normal resting ecg, 148 with an ST-T wave abnormality, and 4 with probable or definite left centricular hypertrophy)

Figure 13: An example from the Stanford Sentiment Analysis Treebank validation set. Interactions highlight intuitive patterns in text, such as phrases like "beautifully observed" and "miraculously unsentimental" being strongly positive interactions.

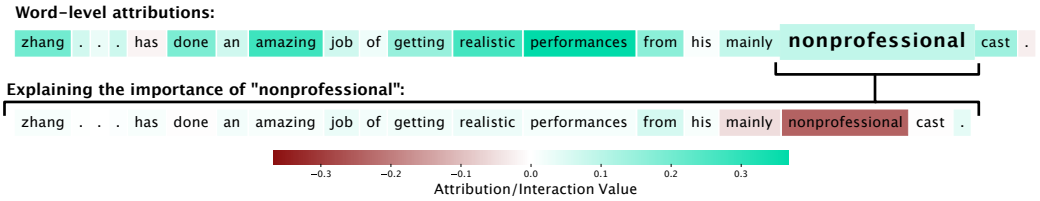

Figure 14: An example from the Stanford Sentiment Analysis Treebank validation set. This example shows that the word "nonprofessional" has a main effect that is negative, but the surrounding context outweighs the main effect and makes the overall attribution positive.

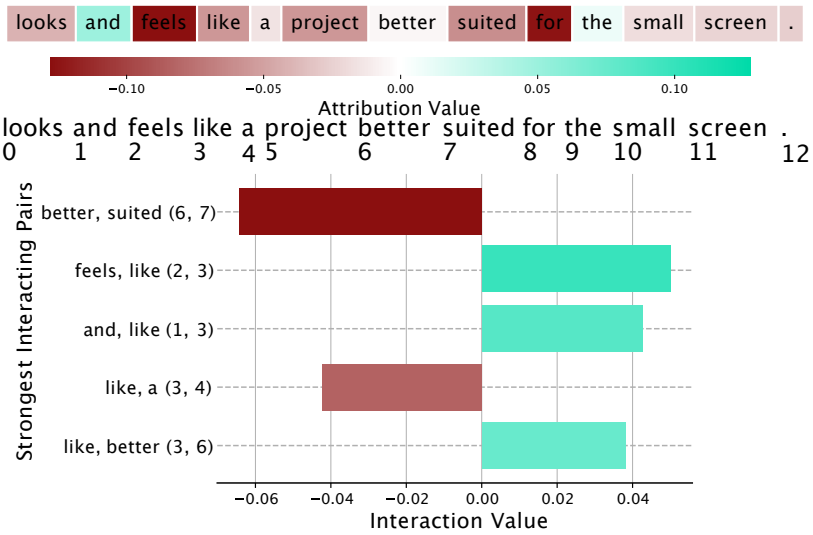

Figure 15: An example from the Stanford Sentiment Analysis Treebank validation set. This example highlights the phrase "better suited" being strongly negative, which is very intuitive. However, some of the other interactions are slightly less intuitive and may indicate lack of training data or higher order interactions beyond word pairs.

We split the data into 238 patients for training (of which 109 had coronary artery disease) and 60 for testing (of which 28 have coronary artery disease). We used a two-layer neural network with 128 and 64 hidden units, respectively, with SoftPlus activation after each layer. We optimized using gradient descent (processing the entire training set in a single batch) with an initial learning rate of 0.1 that decays exponentially with a rate 0.99 after each epoch. We used nesterov momentum with $\beta = 0.9$ (Sutskever et al., 2013). After training for 200 epochs, the network achieved a held-out accuracy of 0.8667, with a 0.8214 true positive rate and a 0.9062 true negative rate. Note that the hyper-parameters chosen here were not carefully tuned on a validation set - they simply seemed to converge to a reasonable performance on the training set. Our focus is not making state-of-the-art predictions or comparing model performance, but rather interpreting the patterns a reasonable model learns.

To generate attributions and interactions for this data set, we used Expected Gradients and Expected Hessians, with the training set forming the background distribution. We used 200 samples to compute both attributions and interactions, although we note this number is probably larger than necessary but was easy to compute due to the data set's small size.

Figure 20 shows which features were most important for predicting heart disease aggregated over the entire data set, as well as the trend of importance values. Interestingly, the model learns some strangely unintuitive trends: if a patient did not experience chest pain, they were more likely to have heart disease than if they experienced anginal chest pain. This could indicate problems with the way certain features were encoded, or perhaps data set bias. Figure 21 demonstrates an interaction learned by the network between

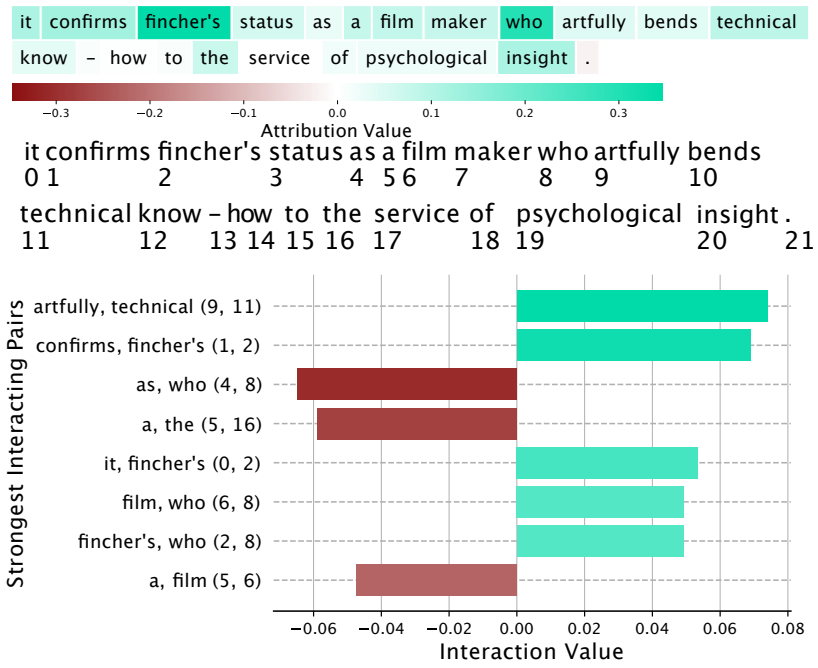

Figure 16: Another example from the Stanford Sentiment Analysis Treebank validation set. Notice how the strongest interact pairs may not necessarily be adjacent words, e.g., "artfully" and "technical."

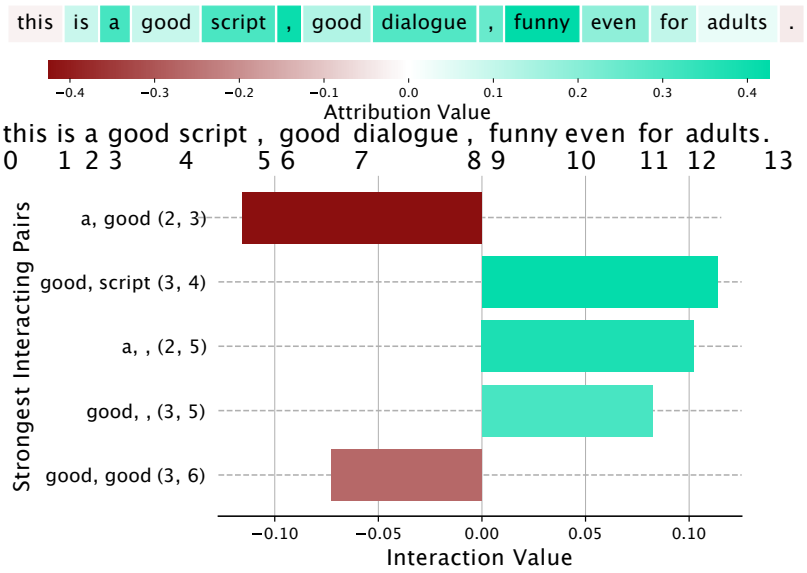

Figure 17: An example from the Stanford Sentiment Analysis Treebank validation set. Interestingly, the phrase "a good" has a negative interaction. This may indicate saturation effects, higher order effects, or that the model has simply learned an unintuitive pattern.

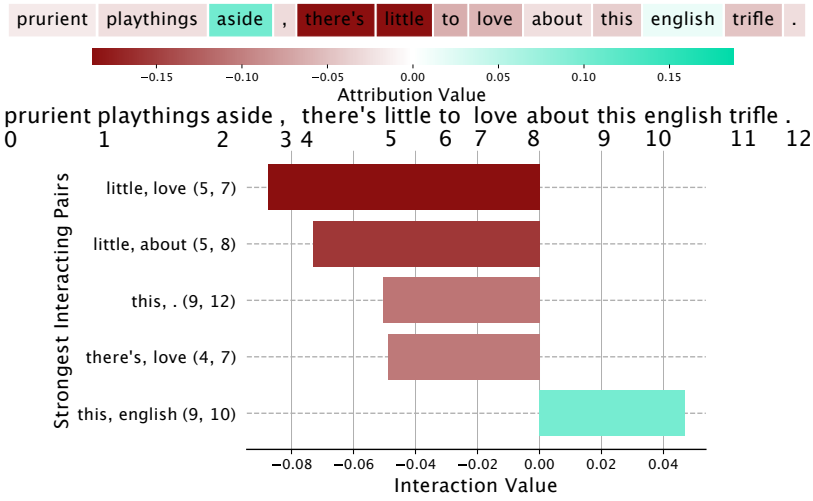

Figure 18: An example from the Stanford Sentiment Analysis Treebank validation set. This example shows some very intuitive negative interactions among the phrase "there's little to love". Interestingly, "this english" has a positive interaction: perhaps the data set has a bias for English movies?

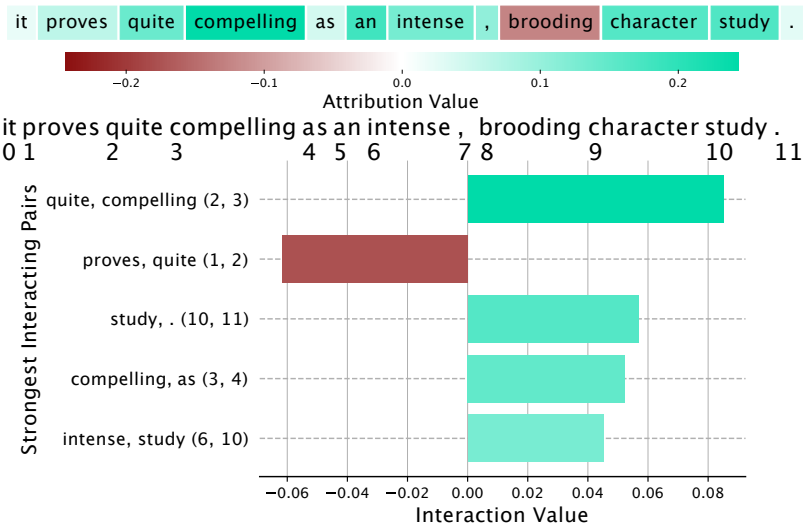

Figure 19: An example from the Stanford Sentiment Analysis Treebank validation set. This also shows intuitive patterns, e.g., "quite compelling" being strongly positive.

maximum heart rate achieved and gender, and Figure 22 demonstrates an interaction between exercise-induced ST-segment depression and the number of major vessels appearing to contain calcium.

In Figure 23, we examine interactions with a feature describing the number of major coronary arteries with calcium accumulation (0 to 3), as determined by cardiac cinefluoroscopy (Detrano et al., 1986). Previous research has shown that this technique is a reliable way to gauge calcium build-up in major blood vessels, and it serves as a strong predictor of coronary artery disease (Detrano et al., 1986; Bartel et al., 1974; Liu et al., 2015). Our model correctly learned that more coronary arteries with evidence of calcification indicate increased risk of disease. Additionally, Integrated Hessians reveals that our model learns a negative interaction between the number of coronary arteries with calcium accumulation and female gender. This supports the well-known phenomenon of under-recognition of heart disease in women – at the same levels of cardiac risk factors, women are less likely to have clinically manifest coronary artery disease (Maas and Appelman, 2010).

E.2 Pulsar Star Prediction

We used a physics data set to confirm that a model learned a global pattern that was visible in the training data. We utilized the HRTU2 data set, curated by Lyon et al. (2016) and originally gathered by Keith et al. (2010). The task was to predict whether or not a particular signal measured from a radio telescope was a pulsar star or generated from radio frequency interference (e.g. background noise). The features included statistical descriptors of measurements made from the radio telescope. The data set contained 16,259 examples generated through radio frequency interference and 1,639 examples that were pulsars. The data set had 4 statistical descriptors — mean, standard deviation, skewness

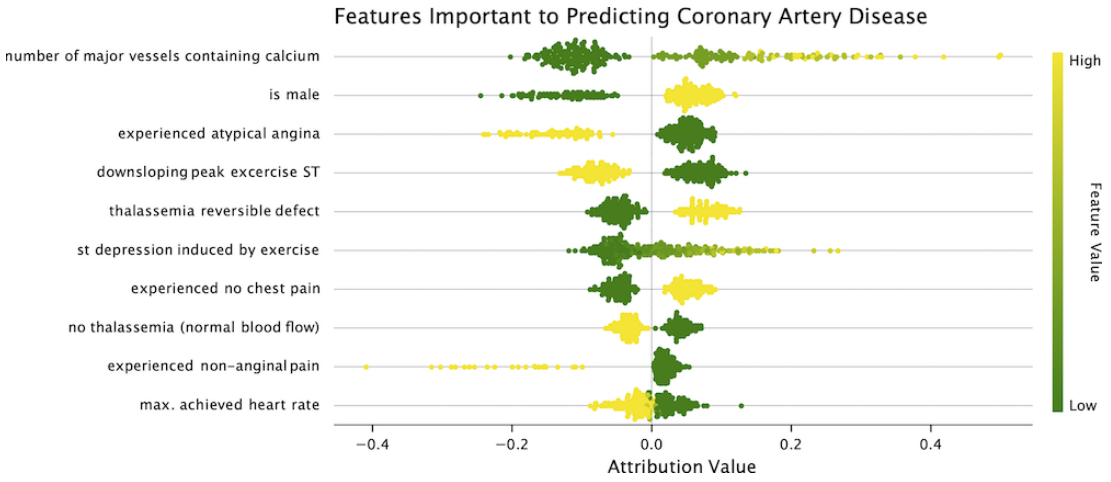

Figure 20: A summary of the most important features for predicting heart disease. A positive attribution indicates increased risk of heart disease (negative value indicates decreased risk of heart disease). The features are ordered by largest mean absolute magnitude over the data set. For binary features, high (yellow) indicates true while low (green) indicates false. For example, for the feature "experienced atypical angina," yellow means the patient did experience atypical angina, and green means the patient did not.

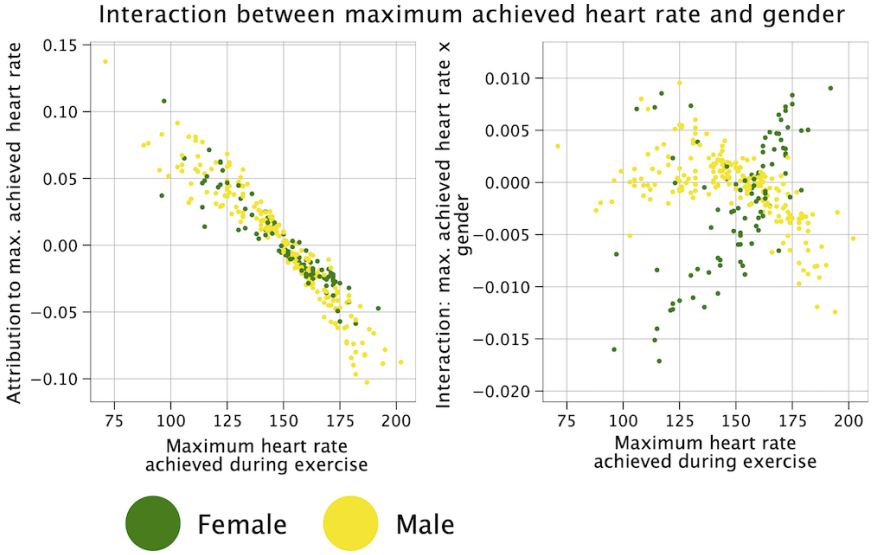

Figure 21: An interaction learned by the model between maximum achieved heart rate during exercise and the gender of the patient. In general, achieving a higher heart rate during exercise indicated a lower risk of heart disease, but the model learns that this pattern is stronger for men than for women.

Figure 22: An interaction learned between ST-segment depression and the number of major vessels appearing to contain calcium. The interaction seems to indicate that if a patient has many vessels appearing to contain calcium, then st-segment depression is less important toward driving risk, probably because the number of major vessels containing calcium becomes the main risk driver.

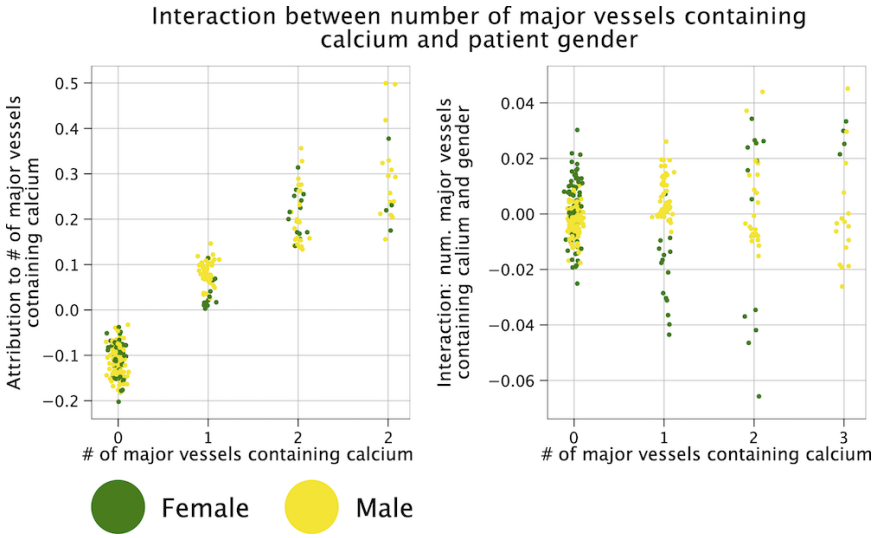

Figure 23: Left: Expected Gradients feature importance of the number of major vessels with accumulation of calcium as indicated by cardiac cinefluoroscopy. More vessels with calcium build-up indicated increased risk. Right: Expected Hessians feature interactions between patient gender and the number of major vessels containing calcium. When the Expected Hessians interactions are aggregated across the data set, they reveal that our model has learned that women with calcium deposition in one coronary artery are less likely than men to be diagnosed with coronary artery disease.

and kurtosis — of two measurements relating to pulsar stars: the *integrated pulse profile* (IP) and the *dispersion-measure signal-to-noise ratio curve* (DM-SNR), for a total of 8 features. The integrated pulse profile measures how much signal the supposed pulsar star emits as a function of the phase of the pulsar: as pulsars rotate, they emit radiation from their magnetic poles, which periodically sweeps over the earth. We can measure the radiation over time using a radio telescope and aggregating measurements over the phase to get the IP. Signals that are pulsar stars should in theory have stronger, more peaked integrated pulse profiles than those generated from radio frequency interference. The DM-SNR curve measures the degree to which phase correction changes the signal-to-noise ratio in the measured signal. Since pulsars are far away, their radio emissions get dispersed as they travel from the star to earth: low frequencies get dispersed more than high frequencies (e.g., they arrive later). Phase correction attempts to re-sync the frequencies; however, no amount of phase correction should help peak a signal if the signal was generated from radio frequency interference rather than a legitimate pulsar.

On this task, we used a two-layer neural network with 32 hidden units in both layers and the SoftPlus activation function after each layer. We optimized using stochastic gradient descent with a batch size of 256. We used an initial learning rate of 0.1, which decays with a rate of 0.96 every batch, and nesterov momentum with $\beta = 0.9$ (Sutskever et al., 2013). We trained for 10 epochs and used a class-weight ratio of 1:3 negative to positive to combat the imbalance in the training data set. Again, we note that these hyper-parameters are not necessarily optimal but were simply chosen because they produced reasonable convergence on the training set. We split the data into 14,318 training examples (1,365 are pulsars) and 3,580 testing examples (274 are pulsars) and achieved a held out test accuracy of 0.98 (0.86 TPR and 0.99 TNR).

To generate attributions and interactions for this data set, we used Expected Gradients and Expected Hessians, with the training set forming the background distribution. We used 200 samples to compute both attributions and interactions, although, as noted in Appendix E.1, 200 samples was probably larger than necessary. In Figure 24, we examine the interaction between two key features in the data set: kurtosis of the integrated profile, which we abbreviate as kurtosis (IP), and standard deviation of the dispersion-measure signal-to-noise ratio curve, which we abbreviate as standard deviation (DM-SNR). The bottom of Figure 24 shows that kurtosis (IP) is a highly predictive feature, while standard deviation (DM-SNR) is less predictive. However, in the range where kurtosis (IP) is roughly between 0 and 2, standard deviation (DM-SNR) helps distinguish between a concentration of negative samples at standard deviation (DM-SNR) < 40. We can verify that the model we trained correctly learns this interaction. By plotting the interaction values learned by the model against the value of kurtosis (IP), we see a peak positive interaction for points in the indicated range and with high standard deviation (DM-SNR). Interaction values show us that the model has successfully learned the expected pattern: that standard deviation (DM-SNR) has the highest discriminative power when kurtosis (IP) is in the indicated range.

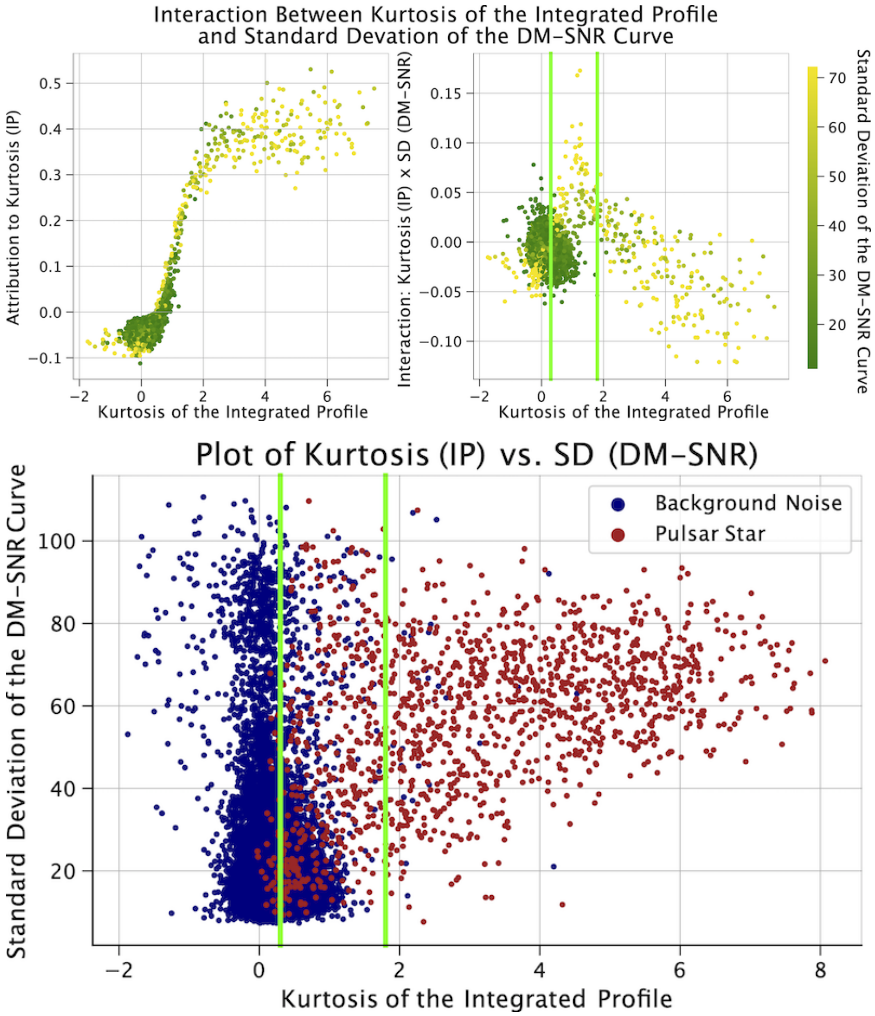

Figure 24: Top Left: Attributions to kurtosis (IP) generated by expected gradients. Top Right: The model learns a peak positive interaction when kurtosis (IP) is in the range [0, 2]. Bottom: A plot of the training data along the axes of the two aforementioned features, colored by class label. Although kurtosis (IP) seems to be the more predictive feature, in the highlighted band the standard deviation (DM-SNR) provides useful additional information: a larger standard deviation (DM-SNR) implies a higher likelihood of being a pulsar star.

Appendix F. Details for Anti-Cancer Drug Combination Response Prediction

F.1 Data Description

As mentioned in Section 6.2, our data set consisted of 12,362 samples (available from the CTD2 data portal). Each sample contained the measured response of a 2-drug pair tested on the cancer cells of a patient (Tyner et al., 2018). The 2-drug combination was described by both a *drug identity indicator* and a *drug target indicator*. For each sample, the drug identity indicator was a vector $x_{\text{id}} \in \mathbb{R}^{46}$ where each element represented one of the 46 anti-cancer drugs present in the data; each element took a value of 0 if the corresponding drug was not present in the combination and a value of 1 if the corresponding drug was present in the combination. Therefore, for each sample, x_{id} had 44 elements equal to 0 and 2 elements equal to 1, the most compact possible representation for the 2-drug combinations. The drug target indicator was a vector $x_{\text{target}} \in \mathbb{R}^{112}$, where each element represented one of the 112 unique molecular targets of the anti-cancer drugs in the data set. Each entry in this vector equalled 0 if neither drug targeted the given molecule, equalled 1 if one of the drugs in the combination targeted the given molecule, and equalled 2 if both drugs targeted the molecule. The targets were compiled using the information available on DrugBank (Wishart et al., 2018). The *ex vivo* samples of each patient's cancer was described using gene expression levels for each gene in the transcriptome, as measured by RNA-seq, $x_{\text{RNA}} \in \mathbb{R}^{15377}$. Before training, the data was split into two parts – 80% of the samples were used for model training, and an additional 20% were used as a held-out validation set to determine when the model had been trained for a sufficient number of epochs.

F.2 RNA-seq Preprocessing

The cancerous cells in each sample were described using RNA-seq data, i.e., measurements of the expression level of each gene in the sample. We describe here the preprocessing steps used to remove batch effects while preserving biological signals. We first converted raw transcript counts to fragments per kilobase of exon model per million mapped reads (FPKM), a measure known to better reflect the molar amount of each transcript in the original sample than raw counts. FPKM accounts for this by normalizing the counts for different genes according to the length of transcript, as well as for the total number of reads included in the sample (Mortazavi et al., 2008). The equation for FPKM is

$$\text{FPKM} = \frac{X_i \times 10^9}{N l_i}, \tag{56}$$

where X_i is the vector containing the number of raw counts for a particular transcript i across all samples, l_i is the effective length of that transcript, and N is the total number of counts. After converting raw counts to FPKM, we opted to consider only the protein-coding part of the transcriptome by removing all non-protein-coding transcripts from the data set. Protein-coding transcripts were determined according to the list provided by the HUGO Gene Nomenclature Committee (https://www.genenames.org/download/statistics-and-files/). In addition to non-protein-coding transcripts, we also removed any transcript that was not observed in $> 70\%$ of the samples. Transcripts were then $\log_2$ transformed and made 0-mean, unit variance. Finally, the ComBat tool (a robust empirical Bayes regression

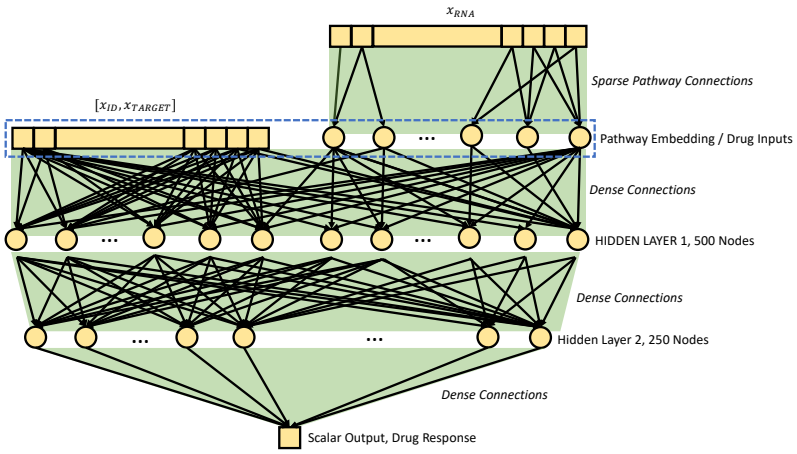

Figure 25: Neural network architecture for anti-cancer drug combination response prediction. We learn an embedding from all RNA-seq gene expression features (x_{RNA}) to KEGG pathways by sparsely connecting the inputs only to nodes corresponding to the pathways of which they are members. When we calculate feature attributions and interactions, we attribute to the layer that contains the raw drug inputs and the learned pathway embeddings (layer boxed with dashed blue line).

implemented as part of the sva R package) was used to correct for batch effects (Leek and Storey, 2007).

F.3 Model and Training Description

To model the data, we combined the successful approaches of Preuer et al. (2017) and Hao et al. (2018). Our network architecture was a simple feed-forward network (Figure 25), as in Preuer et al. (2017), where there were two hidden layers of 500 and 250 nodes, respectively, both with Tanh activation. To improve performance and interpretability, we followed Hao et al. (2018) in learning a *pathway-level* embedding of the gene expression data. The RNA-seq data, $x_{\text{RNA}} \in \mathbb{R}^{15377}$, was sparsely connected to a layer of 1077 nodes, where each node corresponded to a single pathway from KEGG, BioCarta, or Reactome (Kanehisa et al., 2002; Nishimura, 2001; Croft et al., 2014). We made this embedding non-linear by following the sparse connections with a Tanh activation function. The non-linear pathway embeddings were then concatenated to the drug identity indicators and the drug target indicators, and these served as inputs to the densely connected layers.We trained the network to optimize a mean squared error loss function and used the Adam optimizer in PyTorch with default hyperparameters and a learning rate equal to 10^{-5} (Kingma and Ba, 2014). We stopped the training when mean squared error on the held-out validation set failed to improve over 10 epochs, and found that the network reached an optimum at 200 epochs. For easier calculation and more intuitive attribution, we attributed the model's output to the layer

with the pathway embedding and drug inputs rather than to the raw RNA-seq features and drug inputs (see Figure 25).

For this experiment, we calculated all explanations and interactions using the Integrated Gradients and Integrated Hessians approach, using the all zeros vector as reference and $k > 256$ interpolation points.

F.4 Biological Interaction Calculation

To evaluate how well the interactions detected by Integrated Hessians matched with the ground truth for biological drug-drug interactions in this data set, we can use additional single drug response data (that our model was not given access to) in order to calculate *biological synergy*. Drug synergy is the degree of extra-additive or sub-additive response observed when two drugs are combined as compared to the additive response that would be expected if there were no interaction between the two compounds. The drug response for a single drug is measured as $\text{IC50}^{\text{single}}$, or the dose of that single drug necessary to kill half of the cells in an *ex vivo* sample. The drug response for a drug combination is measured as $\text{IC50}^{\text{combination}}$, or the dose of an equimolar combination of two drugs necessary to kill half of the cells in an *ex vivo* sample. The drug synergy between two drugs a and b can be calculated using the CI, or combination index:

$$CI_{a,b} = \frac{\text{IC50}_a^{\text{combination}}}{\text{IC50}_a^{\text{single}}} + \frac{\text{IC50}_b^{\text{combination}}}{\text{IC50}_b^{\text{single}}}. \tag{57}$$

For CI, a value greater than 1 indicates anti-synergy (negative interaction), while a value less than 1 indicates synergy (positive interaction). While our model was trained solely to predict $\text{IC50}^{\text{combination}}$, we determined how well the model learned true biological interactions by using the additional single drug response data to calculate synergy for particular samples. As described in Section 6.2, when we calculated CI values for all samples in which the combination of Venetoclax and Artemisinin was tested, then binarized the samples into synergistic and anti-synergistic, we saw that the Integrated Hessians values were higher in the truly synergistic group than in the truly anti-synergistic group ($p = 2.31 \times 10^{-4}$). This is remarkable given that the model had no access to single drug data whatsoever.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)*, pages 265–283, 2016.

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.

Marco Ancona, Enea Ceolini, Cengiz Öztireli, and Markus Gross. Towards better understanding of gradient-based attribution methods for deep neural networks. *arXiv preprint arXiv:1711.06104*, 2017.

Robert J Aumann and Lloyd S Shapley. *Values of non-atomic games*. Princeton University Press, 2015.

Alan G Bartel, James T Chen, Robert H Peter, Victor S Behar, Yihong Kong, and Richard G Lester. The significance of coronary calcification detected by fluoroscopy: a report of 360 patients. *Circulation*, 49(6):1247–1253, 1974.

Alexander Binder, Grégoire Montavon, Sebastian Lapuschkin, Klaus-Robert Müller, and Wojciech Samek. Layer-wise relevance propagation for neural networks with local renormalization layers. In *International Conference on Artificial Neural Networks*, pages 63–71. Springer, 2016.

Gino Brunner, Yang Liu, Damián Pascual, Oliver Richter, Massimiliano Ciaramita, and Roger Wattenhofer. On identifiability in transformers. *arXiv preprint*, 2019.

David Croft, Antonio Fabregat Mundo, Robin Haw, Marija Milacic, Joel Weiser, Guanming Wu, Michael Caudy, Phani Garapati, Marc Gillespie, Maulik R Kamdar, et al. The reactome pathway knowledgebase. *Nucleic acids research*, 42(D1):D472–D477, 2014.

Tianyu Cui, Pekka Marttinen, and Samuel Kaski. Recovering pairwise interactions using neural networks. *arXiv preprint arXiv:1901.08361*, 2019.

Resul Das, Ibrahim Turkoglu, and Abdulkadir Sengur. Effective diagnosis of heart disease through neural networks ensembles. *Expert systems with applications*, 36(4):7675–7680, 2009.

Robert Detrano, Ernesto E. Salcedo, Robert E. Hobbs, and John Yiannikas. Cardiac cinefluoroscopy as an inexpensive aid in the diagnosis of coronary artery disease. *The American Journal of Cardiology*, 57(13):1041 – 1046, 1986. ISSN 0002-9149. doi: https://doi.org/10.1016/0002-9149(86)90671-5. URL `http://www.sciencedirect.com/science/article/pii/0002914986906715`.

Robert Detrano, Andras Janosi, Walter Steinbrunn, Matthias Pfisterer, Johann-Jakob Schmid, Sarbjit Sandhu, Kern H Guppy, Stella Lee, and Victor Froelicher. International application of a new probability algorithm for the diagnosis of coronary artery disease. *The American journal of cardiology*, 64(5):304–310, 1989.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Kedar Dhamdhere, Ashish Agarwal, and Mukund Sundararajan. The shapley taylor interaction index. *arXiv preprint arXiv:1902.05622*, 2019.

Ann-Kathrin Dombrowski, Maximillian Alber, Christopher Anders, Marcel Ackermann, Klaus-Robert Müller, and Pan Kessel. Explanations can be manipulated and geometry is to blame. In *Advances in Neural Information Processing Systems*, pages 13567–13578, 2019.

Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009.

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.

Ramón Flores, Elisenda Molina, and Juan Tejada. Evaluating groups with the generalized shapley value. *4OR*, 17(2):141–172, 2019.

Ruth Fong and Andrea Vedaldi. Net2vec: Quantifying and explaining how concepts are encoded by filters in deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8730–8738, 2018.

Ruth C Fong and Andrea Vedaldi. Interpretable explanations of black boxes by meaningful perturbation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3429–3437, 2017.

J Raymond Geis, Adrian P Brady, Carol C Wu, Jack Spencer, Erik Ranschaert, Jacob L Jaremko, Steve G Langer, Andrea Borondy Kitts, Judy Birch, William F Shields, et al. Ethics of artificial intelligence in radiology: summary of the joint European and North American multisociety statement. *Radiology*, 293(2):436–440, 2019.

Reza Ghaeini, Xiaoli Z Fern, and Prasad Tadepalli. Interpreting recurrent and attention-based neural models: a case study on natural language inference. *arXiv preprint arXiv:1808.03894*, 2018.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.

Michel Grabisch and Marc Roubens. An axiomatic approach to the concept of interaction among players in cooperative games. *International Journal of game theory*, 28(4):547–565, 1999.

Peyton Greenside, Tyler Shimko, Polly Fordyce, and Anshul Kundaje. Discovering epistatic feature interactions from neural network models of regulatory dna sequences. *Bioinformatics*, 34(17):i629–i637, 2018.

Jie Hao, Youngsoon Kim, Tae-Kyung Kim, and Mingon Kang. Pasnet: pathway-associated sparse deep neural network for prognosis prediction from high-throughput data. *BMC bioinformatics*, 19(1):1–13, 2018.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. In *Advances in Neural Information Processing Systems*, pages 9734–9745, 2019.

Sarthak Jain and Byron C Wallace. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

Minoru Kanehisa et al. The kegg database. In *Novartis Foundation Symposium*, pages 91–100. Wiley Online Library, 2002.

Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viégas, and Michael Terry. Segment integrated gradients: Better attributions through regions. *arXiv preprint arXiv:1906.02825*, 2019.

MJ Keith, A Jameson, W Van Straten, M Bailes, S Johnston, M Kramer, A Possenti, SD Bates, NDR Bhat, M Burgay, et al. The high time resolution universe pulsar survey–i. system configuration and initial discoveries. *Monthly Notices of the Royal Astronomical Society*, 409(2):619–627, 2010.

Been Kim, Martin Wattenberg, Justin Gilmer, Carrie Cai, James Wexler, Fernanda Viegas, and Rory Sayres. Interpretability beyond feature attribution: Quantitative testing with concept activation vectors (tcav). *arXiv preprint arXiv:1711.11279*, 2017.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Igor Kononenko et al. An efficient explanation of individual classifications using game theory. *Journal of Machine Learning Research*, 11(Jan):1–18, 2010.

Peter K Koo, Praveen Anand, Steffan B Paul, and Sean R Eddy. Inferring sequence-structure preferences of rna-binding proteins with convolutional residual networks. *bioRxiv*, page 418459, 2018.

Vivian Lai, Jon Z Cai, and Chenhao Tan. Many faces of feature importance: Comparing built-in and post-hoc feature importance in text classification. *arXiv preprint arXiv:1910.08534*, 2019.

Jaesong Lee, Joong-Hwi Shin, and Jun-Seok Kim. Interactive visualization and manipulation of attention-based neural machine translation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 121–126, 2017.

Jeffrey T Leek and John D Storey. Capturing heterogeneity in gene expression studies by surrogate variable analysis. *PLOS Genetics*, 3(9):1–12, 09 2007. doi: 10.1371/journal.pgen.0030161. URL https://doi.org/10.1371/journal.pgen.0030161.

Zhong Qiu Lin, Mohammad Javad Shafiee, Stanislav Bochkarev, Michael St Jules, Xiao Yu Wang, and Alexander Wong. Explaining with impact: A machine-centric strategy to quantify the performance of explainability algorithms. *arXiv preprint arXiv:1910.07387*, 2019.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. A structured self-attentive sentence embedding. *arXiv preprint arXiv:1703.03130*, 2017.

Frederick Liu and Besim Avci. Incorporating priors with feature attribution on text classification. *arXiv preprint arXiv:1906.08286*, 2019.

Wei Liu, Yue Zhang, Cheuk-Man Yu, Qing-Wei Ji, Meng Cai, Ying-Xin Zhao, and Yu-Jie Zhou. Current understanding of coronary artery calcification. *Journal of geriatric cardiology: JGC*, 12(6):668, 2015.

Christos Louizos, Uri Shalit, Joris Mooij, David Sontag, Richard Zemel, and Max Welling. Causal effect inference with deep latent-variable models. *arXiv preprint arXiv:1705.08821*, 2017.

Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774, 2017.

Scott M. Lundberg, Gabriel Erion, Hugh Chen, Alex DeGrave, Jordan M. Prutkin, Bala Nair, Ronit Katz, Jonathan Himmelfarb, Nisha Bansal, and Su-In Lee. From local explanations to global understanding with explainable ai for trees. *Nature Machine Intelligence*, 2(1):56–67, 2020. doi: 10.1038/s42256-019-0138-9. URL https://doi.org/10.1038/s42256-019-0138-9.

Robert J Lyon, BW Stappers, Sally Cooper, JM Brooke, and JD Knowles. Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. *Monthly Notices of the Royal Astronomical Society*, 459(1):1104–1123, 2016.

Angela HEM Maas and Yolande EA Appelman. Gender differences in coronary heart disease. *Netherlands Heart Journal*, 18(12):598–603, 2010.

Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5188–5196, 2015.

Jean-Luc Marichal, Ivan Kojadinovic, and Katsushige Fujimoto. Axiomatic characterizations of generalized values. *Discrete Applied Mathematics*, 155(1):26–43, 2007.

Ali Mortazavi, Brian A Williams, Kenneth McCue, Lorian Schaeffer, and Barbara Wold. Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628, 2008.

W James Murdoch, Peter J Liu, and Bin Yu. Beyond word importance: Contextual decomposition to extract interactions from lstms. *arXiv preprint arXiv:1801.05453*, 2018.

Darryl Nishimura. Biocarta. *Biotech Software & Internet Report: The Computer Software Journal for Scient*, 2(3):117–120, 2001.

Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2(11):e7, 2017.

Chris Olah, Arvind Satyanarayan, Ian Johnson, Shan Carter, Ludwig Schubert, Katherine Ye, and Alexander Mordvintsev. The building blocks of interpretability. *Distill*, 3(3):e10, 2018.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

Kristina Preuer, Richard P I Lewis, Sepp Hochreiter, Andreas Bender, Krishna C Bulusu, and Günter Klambauer. DeepSynergy: predicting anti-cancer drug synergy with Deep Learning. *Bioinformatics*, 34(9):1538–1546, 12 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx806. URL `https://doi.org/10.1093/bioinformatics/btx806`.

Nikaash Puri, Piyush Gupta, Pratiksha Agarwal, Sukriti Verma, and Balaji Krishnamurthy. Magix: Model agnostic globally interpretable explanations. *arXiv preprint arXiv:1706.07160*, 2017.

Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

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, 2016.

Raúl Rojas. *Neural networks: a systematic introduction*. Springer Science & Business Media, 2013.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Andrew D Selbst and Julia Powles. Meaningful information and the right to explanation. *International Data Privacy Law*, 7(4):233–242, 12 2017. ISSN 2044-3994. doi: 10.1093/idpl/ipx022. URL `https://doi.org/10.1093/idpl/ipx022`.

Sofia Serrano and Noah A Smith. Is attention interpretable? *arXiv preprint arXiv:1906.03731*, 2019.

Ira Shavitt and Eran Segal. Regularization learning networks: deep learning for tabular datasets. In *Advances in Neural Information Processing Systems*, pages 1379–1389, 2018.

Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. Learning important features through propagating activation differences. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3145–3153. JMLR. org, 2017.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Chandan Singh, W James Murdoch, and Bin Yu. Hierarchical interpretations for neural network predictions. *arXiv preprint arXiv:1806.05337*, 2018.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y 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.

Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 5(1):e22, 2020.

Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. *arXiv preprint arXiv:1908.08474*, 2019.

Mukund Sundararajan and Ankur Taly. A note about: Local explanation methods for deep neural networks lack sensitivity to parameter values. *arXiv preprint arXiv:1806.04205*, 2018.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org, 2017.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*, 2012.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013.

Sarah Tan, Rich Caruana, Giles Hooker, Paul Koch, and Albert Gordo. Learning global additive explanations for neural nets using model distillation. *arXiv preprint arXiv:1801.08640*, 2018.

Michael Tsang, Dehua Cheng, and Yan Liu. Detecting statistical interactions from neural network weights. *arXiv preprint arXiv:1705.04977*, 2017.

Michael Tsang, Sirisha Rambhatla, and Yan Liu. How does this interaction affect me? interpretable attribution for feature interactions. *arXiv preprint arXiv:2006.10965*, 2020.

Jeffrey W Tyner, Cristina E Tognon, Daniel Bottomly, Beth Wilmot, Stephen E Kurtz, Samantha L Savage, Nicola Long, Anna Reister Schultz, Elie Traer, Melissa Abel, et al. Functional genomic landscape of acute myeloid leukaemia. *Nature*, 562(7728):526–531, 2018.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615, 2016.

David S Wishart, Yannick D Feunang, An C Guo, Elvis J Lo, Ana Marcu, Jason R Grant, Tanvir Sajed, Daniel Johnson, Carin Li, Zinat Sayeeda, et al. Drugbank 5.0: a major update to the drugbank database for 2018. *Nucleic acids research*, 46(D1):D1074–D1082, 2018.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771, 2019.

Mike Wu, Michael C Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Suggala, David I Inouye, and Pradeep K Ravikumar. On the (in) fidelity and sensitivity of explanations. In *Advances in Neural Information Processing Systems*, pages 10965–10976, 2019.

Hao Zhang, Bo Chen, Yulai Cong, Dandan Guo, Hongwei Liu, and Mingyuan Zhou. Deep autoencoding topic model with scalable hybrid bayesian inference. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

Mingyuan Zhou, Yulai Cong, and Bo Chen. Augmentable gamma belief networks. *The Journal of Machine Learning Research*, 17(1):5656–5699, 2016.