Assessing the Interpretability of Large Language Models

Candidate Number: 1058888

The University of Oxford

A thesis presented for the degree of

Master of Science in Advanced Computer Science

Source Code: link

Word Count: 18511 (Computed by Texcount/Overleaf)

Trinity Term 2022

Acknowledgements

Abstract

Machine learning models, most notably deep neural networks, have grown rapidly in size and complexity in recent years. While this increase in capacity has brought significant gains in performance across a variety of tasks, it has also rendered the process by which these models make decisions opaque. In response to this issue, a significant amount of recent research (under various headings such as interpretable machine learning, explainable AI, etc.) has focused on explaining model decisions. Within natural language processing (NLP), this concern is particularly salient. Recently, NLP models have been significantly scaled up, the largest models having tens or hundreds of billions of parameters. These larger models have achieved impressive gains in performance across many linguistic domains and tasks, prompting extensive research into how they generate text and learn representations of language. Providing human comprehensible explanations of how these models generate their outputs, though, is a difficult problem. Furthermore, evaluating and comparing the quality of different explanation methods is itself a non-trivial problem. In this work, we use a variety of approaches to evaluate the quality of explanations, and particularly feature importance methods, across several different large language models (LLMs). The works that proposed these methods for evaluation generally applied them to a limited set of models, usually the base BERT model [14] or one of its variants. This leaves an open question: **Do feature importance explanations still work equally well across language models of differing architectures and sizes**? By applying these approaches to models with different structures and sizes, we can assess the robustness of explanation methods to different architectural choices. We additionally assess the robustness of several explanation methods to adversarial attacks, and propose a novel variant of a previously published method to generate adversarial examples that more closely resemble their original inputs. We find that varying the size of language models has little impact on the quality of explanations, but that the performance of explanation methods on different architectures of similar size varies more drastically. This variation across different model architectures highlights the difficulty of using these explanation methods in practice, as both the best method to use and the accuracy of its results are difficult to know a priori.

Table of Contents

List of Figures

List of Tables

1 | Introduction

1.1 Motivation and Contributions

As increases in computing power and hardware advances have enabled the training of increasingly complex machine learning models, the scope of real-world machine learning applications has broadened. Companies across various industries have begun using highly complex models for important applications such as facial recognition [3], machine translation [23], and financial services [18]. Generally, the validity of these approaches point to the performance of the models, e.g. accuracy on held-out evaluation data. High performance on evaluation data is, from this view, assumed to lead to high performance in the real-world context of the application, as it provides an unbiased estimate of future predictive performance (under the assumption that future data comes from the same distribution as the held-out data). However, looking purely at aggregate metrics of performance obscures important failure cases. Various forms of bias and discrimination [4, 11], decision-making based on spurious correlations [38], and many other problematic behaviors have all been well documented across machine learning models [40]. None of these problems are apparent from a model's performance on held-out testing data, but nevertheless raise deep concerns about the legitimacy of applying machine learning to any important decision-making processes.

Within natural language processing, these problems are particularly pressing. Recently trained language models are truly enormous, often with hundreds of billions of parameters [10, 27, 13]. In addition, these models are trained on billions of example sentences, generally scraped from the internet [16]. For any one of these large language models, we have poorly understood internal behavior governed by a massive number of parameters, combined with an underlying dataset that frequently contains information that is factually unreliable, prejudiced or offensive [20]. Add to that the numerous potential industrial applications of these LLMs, and recent investment to realize them [61], and suddenly all the issues described above are magnified.

One approach to helping understand and identify these kinds of issues is the subfield of machine learning interpretability, which focuses on explaining model decisions in a human-comprehensible way. Explanations of model behavior can be used to identify problems like bias and verify that algorithmic decisions only make use of factors that human decision-makers would consider 'valid' for a particular task (bias in this context being the use of 'invalid' factors, e.g. race or sex, for prediction). While there has been prior work applying general interpretability methods to LLMs, and proposing some language specific explanations, the evaluation of the quality of these explanations has generally been limited to models of similar architectures and scales. To understand how effective explanatory tools may be in practice, where architecture and scale may vary widely, we must evaluate explanations on a broader class of language models.

In this work, we undertake a systematic evaluation of the quality of different explanations for language models, focusing in particular on feature attribution methods. We examine how the quality of different explanation methods, in terms of their ability to accurately capture the behavior of the models they explain, changes across models of different scales and architectures. We also evaluate the extent to which we can generate adversarial examples for different model explanations, i.e. inputs that produce similar outputs using similar features, but have different explanations. In practice, these models may be applied to a variety of models, so measuring the quality of explanations in a diverse array of contexts is critical to assessing and understanding their practical utility.

1.2 Thesis Outline

Section 1 has outlined the motivation and goals of this work. The remaining sections are organized as follows:

- Section 2 provides a review of concepts in natural language processing and interpretable machine learning. Further, we provide a detailed overview of the large language models and interpretability tools considered in this thesis.
- Section 3 describes the experimental setups used to evaluate explanation quality for large language models.
- Section 4 presents the experimental results, and discusses some notable findings and relationships between the different explanation methods and classes of models.
- Section 5 is a discussion of those results and their implications for the application of interpretability methods to large language models.

2 | Background and Related Work

This section provides an overview of topics in natural language processing and machine learning interpretability that are relevant to the later experiments and results.

2.1 Natural Language Processing

Natural language processing (NLP) is a subfield that sits between artificial intelligence and linguistics, and focuses on how to use and analyze language data computationally. Problems in NLP have been approached in a variety of distinct ways, but for the purposes of this work, we consider machine learning approaches, that solve a task by learning a model based on some example training data. In particular, the experiments and results here focus on transformer architectures [57], a relatively recent kind of neural network language model. There are many other machine learning methods for NLP, such as recurrent models like LSTMs [26]; however, in recent years, transformers [57] have become the most prominent and widely used [63].

In this section, we first provide a brief overview of how input text is transformed into a numerical format (such that it can be handled computationally), and then a description of the transformer architecture. Following that, we describe the paradigm of pretraining and finetuning that has emerged in NLP, where large models are pretrained on a generic language modeling task and then specialized with task-specific data later.

2.1.1 Data Preprocessing and Word Embeddings

Given some input text, we have to transform the raw text into a form that a machine learning model can operate on. This is generally done in two steps: tokenization, and embedding. The tokenization step converts the text into a list of individual text tokens. For example, the text "they are happy" could be tokenized into the list ["they", "are", "happy"]. It is worth noting, though, that the tokenization does not have to be based on whitespace, and that different tokenization schemes may result in very different token lists. The models considered in this work, for instance, generally use 'subword tokenization' which will frequently split a word into multiple tokens (e.g. "happy" becomes ["hap", "py"]).

The embedding step maps the individual tokens from the tokenization to vectors $\mathbf{e}_i \in \mathbb{R}^d$, called **word embeddings**. The mapping from a token to an embedding is deterministic, and predefined as part of the model architecture. The set of tokens with unique embeddings is called the model **vocabulary**. The dimensionality d of each embedding is also a hyperparameter of the model. Any token not in the model vocabulary is mapped to a special embedding vector for 'unknown' tokens. The final input to the model for some text string s is then a list of embeddings $(\mathbf{e}_1, \mathbf{e}_2, ...\mathbf{e}_t)$, where t is the number of tokens representing s after tokenization. The particular values of the embedding vectors are often trained jointly with the model itself, so a token's embedding will change throughout training to optimize the training objective, along with the rest of the model parameters.

2.1.2 Transformer Models

With the ability to place data in a usable numeric form, we can now describe the transformer architecture. The first implementation of transformers in [57] was a sequence-to-sequence model; a model that takes some text as input, and produces different text as output. Specifically, the architecture performed machine translation, taking as input text in some source language, and producing the translation of that text in a target language. The transformer consists of two components, an encoder and a decoder. The encoder and decoder both are composed of a series of 'attention blocks', each of which applies a self-attention operation to the input embeddings, generating a new set of transformed embeddings as outputs. The encoder transforms the input embeddings into a new contextualized representation, where embeddings for individual tokens can include relevant information from other tokens. The decoder then uses the output of the encoder to autoregressively generate an output sequence.

More formally, an attention block takes as input a sequence of embeddings $\mathbf{e} = (\mathbf{e}_1, ..., \mathbf{e}_t)$, with $\mathbf{e}_i \in \mathbb{R}^d$ (for the first block in the transformer these are the word embeddings described above, while later blocks take in the output of the preceding blocks). It produces as output a new set of embeddings $(\mathbf{c}_1..., \mathbf{c}_t)$, often of the same dimensionality d as the inputs, although this is not required. The first step of the block is the self-attention computation, which is parameterized by three weight matrices, $\mathbf{W}^K \in \mathbb{R}^{d_k \times d}, \mathbf{W}^Q \in \mathbb{R}^{d_k \times d}, \mathbf{W}^V \in \mathbb{R}^{d_v \times d}$. d_k, d_v are hyperparameters specifying the dimensions of these intermediate outputs. These matrices are applied independently to each embedding, producing three new sets of matrices $\mathbf{K}$,$\mathbf{Q}$,$\mathbf{V}$, with

$$\mathbf{K}_i = \mathbf{W}^K \mathbf{e}_i$$

$$\mathbf{Q}_i = \mathbf{W}^Q \mathbf{e}_i$$

$$\mathbf{V}_i = \mathbf{W}^V \mathbf{e}_i.$$

These three matrices are called the **keys**, **queries**, and **values**, respectively. Each of these matrices also have an intuitive interpretation as performing a kind of search. The query vectors (or rows of the query matrix $\mathbf{Q}$) act roughly like search queries for information that is important to a particular token. The key vectors represent how relevant each token is to different queries. The dot product of a key vector $\mathbf{K}_i$ and a query vector $\mathbf{Q}_j$ then measures how relevant some token i is to another token j. The value vectors $\mathbf{V}$ are the actual transformed content of the tokens (like the content of a search result irrespective of its relevance), and are weighted by the search relevance/dot product so that 'more relevant' tokens have a larger impact on the resulting output. More formally, we can compute the overall **attention value** from a token i to another token j as the following:

$$\text{attention}(i, j) = \frac{exp(\mathbf{Q}_i \cdot \mathbf{K}_j)}{\sum_j exp(\mathbf{Q}_i \cdot \mathbf{K}_j)}$$

It is simply the dot product of the query vector for token i with the key vector for token j, and then normalized using the softmax function: $\text{softmax}(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)}$. The normalization step ensures that all the attention values for a single token sum up to one. The output of the self-attention operation for token i is then a sum of all the value vectors for the tokens, weighted by their attention values:

$$\mathbf{e}'_i = \sum_j \text{attention}(i, j)\mathbf{V}_j$$

In practice, the $\mathbf{e}'_i$ vectors are computed more efficiently all at once as matrices, so the complete equation is:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}^1$$

The additional $\sqrt{d_k}$ factor in the denominator is just a normalization factor in-

[1]Note that the softmax function here is applied independently along the rows of the resulting matrix, but is written as above for notational convenience.

troduced in [57] that helps stabilize the gradients during training. Finally, each of the outputs from the attention operation are passed through a multilayer perceptron (MLP), often a two layer network parameterized by matrices $\mathbf{W}_1 \in \mathbb{R}^{d_v \times d_{mlp}}, \mathbf{W}_2 \in \mathbb{R}^{d_{mlp} \times d}$. The MLP is applied independently to each $\mathbf{e}'_i$, so one component of the output $\mathbf{c}_i$ can be written as:

$$\mathbf{c}_i = MLP(\sum_j \text{attention}(i, j)\mathbf{V}_j)$$

Intuitively, self-attention is meant to allow to different components of the input to learn to use information from other relevant parts of the sequence. The attention values that weight the sum over $\mathbf{V}$ can be thought of as representing the importance of a particular token j to the current token i. The new representation $\mathbf{c}_i$ for i can then include information about any other components in the input. For this reason, embeddings output from attention blocks are often called **contextualized word embeddings**. This is as opposed to static word embeddings, like those described in Section 2.1.1, which depend only upon an individual token for their value and not the surrounding context.

Many transformer architectures also include layer normalization and residual connections after both the self-attention operation and the MLP. Additionally, they often use **multi-head attention**, which is a variant of self-attention which uses multiple independent attention computations. An attention head consists of the key, query and value matrices $\mathbf{W}^K, \mathbf{W}^Q, \mathbf{W}^V$ that define self-attention (so the simplest case described above is a single attention head). Multi-head attention uses multiple different attention heads, with k sets of matrices, and then concatenates the resulting output vectors together before a final linear projection:

$$\text{MultiHeadAttention}(\mathbf{e}) = Concat(\text{head}_1, ..., \text{head}_k)\mathbf{W}^O$$

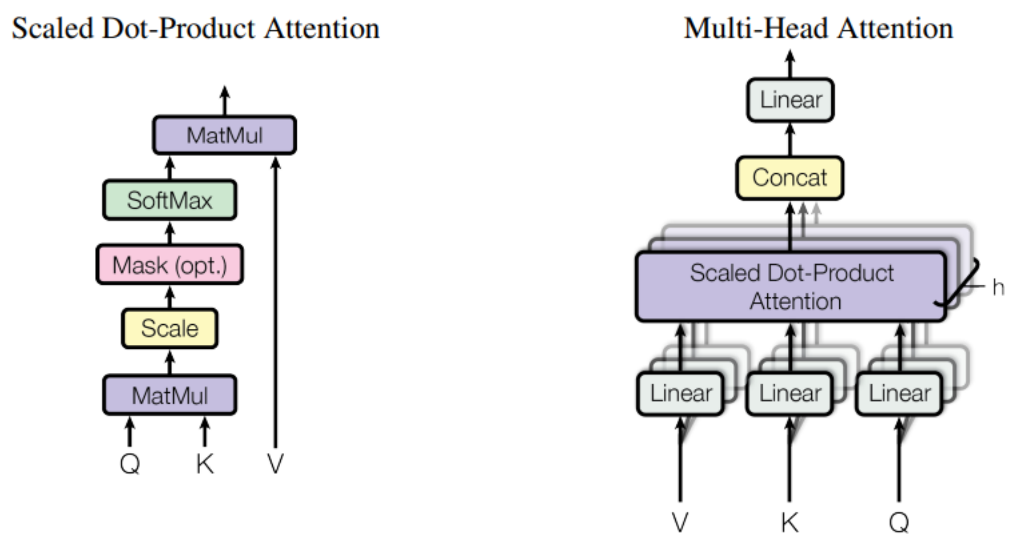


Figure 2.1: Scaled Dot Product Attention and Multi-Head Attention from [57]

where head$_i$ is a single attention head:

$$\text{head}_i = \text{Attention}(\mathbf{W}_i^Q\mathbf{e}, \mathbf{W}_i^K\mathbf{e}, \mathbf{W}_i^V\mathbf{e})$$

The three matrices $\mathbf{W}_i^Q, \mathbf{W}_i^K, \mathbf{W}_i^V$ are equivalent to those defined for single-head attention above, and $\mathbf{W}^O \in \mathbb{R}^{kd_v \times d_{model}}$ is a final linear projection of the concatenated outputs, where d_{model} is the input dimension of the following MLP. Figure 2.1 from the original paper [57] shows diagrams for both single and multi-head attention.

Lastly, it is important to note that the attention operation described above, as well as the application of the MLP, is invariant to the ordering of tokens in the sentence. This is clearly an issue for language tasks (e.g. "man bites dog" is certainly not the same as "dog bites man"). To solve this problem, a **positional embedding** is added to each input embedding before the encoder and decoder, ensuring that the

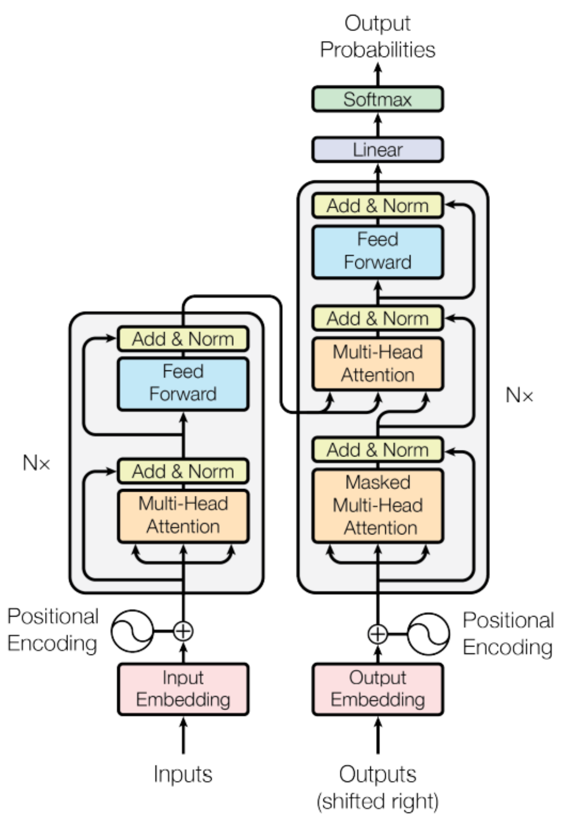


Figure 2.2: Transformer Architecture from [57]

model can approximate any sequence-to-sequence function [67]. This embedding is meant to provide information about the location of each token in the sentence, in addition to the information about the token itself already present in the original embedding. These positional embeddings can be learned, or simply a function of the position of a token that produces a unique vector.

The transformer encoder consists of a stack of these attention blocks, where each block takes in the output embeddings of the previous block. The last attention block in the encoder then outputs a set of contextualized word embeddings. The decoder also consists of a series of attention blocks, but adds an additional self-attention layer in each one, which computes the attention between the inputs to the decoder layer and the outputs of the encoder. This allows the decoder part of the model to use information from the encoder as it generates the output sequence. The bulk of the transformer architecture then consists of these few repeated operations. Specific tasks, like machine translation, are accomplished by adding classification layers that take in the contextualized embeddings from the decoder, and produce the task specific output, e.g. a distribution over a vocabulary. Figure 2.2 from [57] shows an illustration of the overall transformer architecture.

2.1.3 Language Model Pretraining and Transfer Learning

While the original transformer model consists of both an encoder and decoder, later works have used both parts independently to great effect [14, 41]. Encoder-only models, such as [14], have been used as feature extractors for downstream tasks, as their output contextualized embeddings contain more useful linguistic information than an embedding based solely on an input token. Decoder-only models like [41] are particularly effective for generating text, taking a prefix (usually called a prompt) as input and generating further text based on that. Additionally, the parallelizable structure of transformers (in contrast to a sequential model like an LSTM) enables the training of significantly larger models via distributed training schemes. As a result, a slew of enormous language models, essentially all either transformer encoders or decoders, have been trained and released as **pretrained models** [14, 35, 41, 7].

Pretrained models are usually trained using a **language modeling** objective, where the task is to predict some word given the other words in a sentence. For example, BERT [14], a commonly used pretrained model, was trained using masked language modeling. In this task, some of the words in an input sequence are replaced with a special mask token (e.g. "she walked the dog" may become "she walked the [MASK]"), and the model then predicts what word was originally present.

These kinds of pretraining objectives avoid the need to label data, since the label is already present in the original text, and open up the huge amount of text on the internet as readily available training data [16]. These models are often trained using billions of tokens and millions of gradient steps, in the hopes that the resulting network encodes broadly relevant features of language. Despite the simplicity of the objective, prior work has found evidence of fairly complex linguistic relationships encoded in these pretrained models [55]. Further, after the initial pretraining stage, these models can be adapted to more specific tasks, such as natural language inference or question answering [14], and frequently achieve state-of-the-art performance. This boost of performance from pretraining is often attributed to **transfer learning**, the generalization of useful and universal language features to specific tasks. However, while this approach is empirically effective for language models, the precise mechanism by which it works is debated [25, 33].

To apply a pretrained model to some more specific task, the model is often trained further on a task-specific dataset, a process called **finetuning**. In this approach, all the model parameters are jointly optimized to solve the finetuning task, and the pretrained model serves as a good initialization for the training process. Instead of optimizing the entire model, practitioners also frequently treat the pretrained model as a fixed feature extractor, feeding the contextualized output embeddings into a classification model, which is trained, but leaving the pretrained model frozen. Falling between these two are training schemes that finetune only a few layers of the pretrained model, e.g. the last two layers plus the classification model, rather than all of them [39].

2.2 Machine Learning Interpretability

With large pretrained language models serving as the starting points for so many different language tasks and applications, it is critical for us to be able to understand how they produce their outputs. This section outlines some of the reasons why interpretability in machine learning is important, and some of qualities we want good interpretations to have. Following that, it briefly summarizes the methods and evaluation frameworks used for the experiments in this work.

2.2.1 The Goals of Interpretability in Machine Learning

Interpretability is an inherently ill-defined concept in machine learning. What constitutes a valid interpretation or explanation of a model is, in the most general case, a subjective determination, dependent on context, the interpreter, and some equally difficult to define notion of 'comprehension' or 'understanding' of a given interpretation. Despite the broader vagueness of interpretability as an idea, we can still consider the different potential goals of interpreting models, and how those relate to necessary qualities of good explanations. Interpretability can serve many different roles, such as increasing user trust in a system or ensuring safe behavior. For difficult-to-formalize objectives like safety or ethics, interpretability can serve as a proxy [17]. If we can understand how a model makes decisions, then we can see when and why it makes unethical or unsafe ones.

With these uses in mind, one of the most important qualities of any interpretability method is faithfulness to the behavior of the underlying model. For feature importance explanations, this would mean that components of the input that are more important for prediction (in the sense that altering them would have the largest impact on the prediction) should be assigned greater importance by the explanation. Models that are designed to be easily interpretable, often called **inherently interpretable models**, have this behavior by default [44]. The explanation comes as part of the model architecture, and so perfectly reflects model behavior as well. Sparse linear models[2] are an example of this, where the relatively small set of non-zero coefficients can be taken as an explanation of the importance of different components of an input (although the extent to which sparsity provides interpretability is contestable [34]). In this work, though, we are concerned with **post-hoc explanations**, which provide explanations of uninterpretable black-box models, such as deep neural networks, after they have been trained [37, 43, 53, 45, 5, 49]. More specifically, we focus on **local** post-hoc explanations, which provide explanations for individual model predictions, rather than for the model as a whole [64]. To explain an entire model faithfully, the post-hoc explanation would have to be as complex as the model itself [44]. Local explanations, on the other hand, only need to explain model behavior around a particular input, which may be significantly simpler. In order for these explanations to be trustworthy and useful, we have to evaluate the extent to which they are faithful to the models they explain.

2.2.2 Feature Importance Methods

Generally, the inputs to a machine learning model are called features. The tokens that represent some text, for example, would be the input features to a natural language model. Feature importance explanations are a broad class of local, post-hoc explanation methods which all attempt to determine how relevant a particular input feature was to a model's prediction [64]. At a high level, all of these methods

[2]These are linear models of the form $f(x) = w^T x + b$, where x is the input features, and w, b are a set of coefficients for each feature and a bias term, respectively. Sparse linear models are distinct from linear models in the use of a variable selection mechanism, e.g. Lasso [56], which encourages solutions where elements of w are 0. This entirely eliminates the impact of certain features in x on the output.

can be viewed as functions $f(m, \mathbf{x}) = \mathbf{a}$, where m is the function defined by the ML model, $\mathbf{x}$ is an input to m, and $\mathbf{a}$ is the resulting attribution, a vector in $\mathbb{R}^{|\mathbf{x}|}$ that assigns an importance value to each element of $\mathbf{x}$. Different approaches compute these values in different ways, as discussed in more detail below.

2.2.2.1 Gradient Methods

The simplest approach to estimating the importance of different features is using the gradient of the model output with respect to the inputs [48]. Note that this requires the model be end-to-end differentiable. In the simplest case, often called **vanilla gradients**, the importance of some input feature $x_i \in \mathbf{x}$, relative to an output $m(x)$, is simply defined as:

$$\text{importance}_i(\mathbf{x}, m) = \frac{\partial m(\mathbf{x})}{\partial x_i}^3.$$

This can also be written more compactly as $\nabla_{\mathbf{x}} m(\mathbf{x})$ for all the elements of $\mathbf{x}$, where ∇ is the gradient operator. This gradient measures how sensitive the output $m(\mathbf{x})$ is to infinitesimal perturbations of the input features. Ideally, the sensitivity for more important input features would be higher, and so they would have a larger gradient/importance score. In practice, however, vanilla gradient explanations are often very noisy and misleading [2]. There have been several proposed explanations for this behavior, such as gradient saturation [53] or sharp local fluctuations of the gradient that do not impact prediction [49]. Many variants of this basic method have been proposed to alleviate these issues and improve the resulting explanations [49, 53, 5, 36, 45].

Aside from the vanilla gradients method described above, we evaluate two addi-

[3] Note that if $m(\mathbf{x})$ outputs a vector rather than a scalar (as in multi-class classification), the derivative is computed with respect to a particular output position, usually the highest predicted class in the classification case.

tional kinds of gradient explanations. The first is a straightforward variation of vanilla gradients where the gradient is multiplied by the original input (generally called gradient*input). In this case, the explanation is given by

$$\text{importance}_i(\mathbf{x}, m) = \frac{\partial m(\mathbf{x})}{\partial x_i} * x_i.$$

For explanations of word embeddings, where x_i is an embedding vector in $\mathbb{R}^n$, this corresponds to a dot product between the token embedding and its gradient. The second method is Integrated Gradients [53]. Integrated Gradients was designed to avoid some of the problems with vanilla gradient explanations, such as gradient saturation. It approximates a path integral over gradients between a reference input (called the baseline) and the input being explained. Feature importances for Integrated Gradients are given by:

$$\text{importance}_i(\mathbf{x}, m) = (x_i - x'_i) * \int_0^1 \frac{\partial m(\mathbf{x}' - \alpha * (\mathbf{x} - \mathbf{x}'))}{\partial x_i} d\alpha$$

where $\mathbf{x}'$ is the baseline. For experiments in this work, the baseline inputs are generated by replacing all the original tokens in the input with uninformative padding or masking tokens.

2.2.2.2 Perturbation/Surrogate Model Methods

A separate class of feature importance methods do not assume differentiability, instead treating the model as a black box and estimating importance values based purely on input and output values. Two approaches, LIME [43] and SHAP [37], are the most commonly used examples. Both these methods work by generating perturbed versions of the given input. An input image, for example, might be perturbed by having a portion of it masked out. The various perturbations are then input to the black box model, and the output values are used as labels. This dataset of perturbed inputs and black box outputs is then used to train a simpler, more interpretable model to approximate the behavior of the black box model. The interpretation of the approximate model is then taken as the explanation of the black box model as well. The inputs are often also combined into a smaller set of higher-level features, so that the number of possible perturbations and features in the interpretable model is tractable. For vision models, for example, an input image might be segmented into several superpixels, and each superpixel could be masked or not-masked, instead of each pixel.

More formally, LIME with a linear interpretable model (the most common case) minimizes the following objective:

$$\mathcal{L}(m, f, \Pi_{\mathbf{x}}) = \sum_{\mathbf{x}'} \Pi_{\mathbf{x}}(\mathbf{x}')(m(\mathbf{x}') - f(\mathbf{x}'))^2$$

where m is a black box model, f is the linear model, and $\Pi_{\mathbf{x}}$ is a proximity function, which measures the distance between a perturbed sample $\mathbf{x}'$ and the primary input $\mathbf{x}$. The proximity function is meant to more heavily weight the approximation of perturbations that are close to the original input. The coefficients of the linear model then represent the importance of each feature to the output.

SHAP minimizes the same objective, but substitutes in a specific choice of $\Pi_{\mathbf{x}}$ so that the solution estimates the Shapley values [46] for each feature. Shapley values are a concept from cooperative game theory that describes the average contribution of individual agents in a coalition to the overall utility of the entire group. In the context of explainability, the input features take the place of the agents, and their contributions are the feature importance values. Let n be the number of input features in $\mathbf{x}$, and let $|\mathbf{x}'|$ be the number of non-zero features in a perturbed input. By setting $\Pi_{\mathbf{x}}$ to

$$\Pi_{\mathbf{x}}(\mathbf{x}') = \frac{n-1}{\binom{n}{|\mathbf{x}'|}|\mathbf{x}'|(n-|\mathbf{x}'|)}$$

we recover (an approximation of) the Shapley values from LIME. This version of SHAP that builds on LIME is called KernelSHAP (after the specific 'Shapley Kernel' $\Pi_{\mathbf{x}}$ defined above).

These surrogate model methods have the benefit of not requiring differentiability, and can avoid issues like gradient saturation mentioned earlier. However, they assume, by the choice of interpretable model, a simple, often linear, structure to the decision surface of the black box model around the input being explained. If that decision surface is not simple, the resulting explanation may be very misleading. They are also computationally more expensive, as multiple forward passes are required to generate labels for the perturbations, and a separate model has to be trained on that new dataset.

2.2.2.3 Attention Values

Several prior works have also proposed using the attention values computed by transformers as a form of explanation [9]. This is an explanation method that is built-in to models like transformers that use self-attention. Because self-attention values represent the 'relevance' of one token to another at a particular layer, those values should be able to provide some kind of interpretation of model behavior. Important tokens, ideally, should have high attention values, while less relevant tokens should have lower values. However, the validity of using attention values as explanations is highly contested [30, 60], and it is not clear that they are a consistent and faithful form of explanation. Additionally, the fact that transformers stack multiple layers of self-attention on top of one another complicates this process. Because the embeddings output by an attention block are linear combinations of information from all the input embeddings, they no longer represent individual inputs. Attention values in a transformer for layers beyond the first one, then, do not necessarily correspond directly to input tokens in the same position. Consider a sequence of token embeddings $(\mathbf{e}_1, ..., \mathbf{e}_n)$ that are passed through a self-attention layer and transformed into $(\mathbf{c}_1, ..., \mathbf{c}_n)$. If $(\mathbf{c}_1, ..., \mathbf{c}_n)$ are passed through a second self-attention layer, which computes a high attention value for $\mathbf{c}_1$, this is not necessarily indicative of the importance of $\mathbf{e}_1$. Despite this, because these pretrained models are trained to generate representations for each token, we can still assume that $\mathbf{c}_i$ roughly represents $\mathbf{e}_i$, and use the attention values at each layer as a measure of the importance of $\mathbf{e}_i$. This can be done by averaging the attention value from each position to a position i at each layer, or through more complicated approaches, as in [1]. While attention is not a standard form of post-hoc explanation like the previously described approaches, it is useful to include in this work, where we can compare across a large number of models and methods, to see how well it correlates with other forms of explanation. We evaluate two forms of of attention explanation here. The first is the previously described average attention paid to each token across all other tokens and layers (denoted Average Attention in the results). For a particular input token i, we can write this as:

$$AverageAttention_i(\mathbf{A}, \mathbf{x}) = \frac{1}{N_L}\frac{1}{N_H}\frac{1}{|\mathbf{x}|}\sum_{l=0}^{N_L}\sum_{h=0}^{N_H}\sum_{j=0}^{|\mathbf{x}|}\mathbf{A}_{lhji}$$

where N_L, N_H are the number of layers and attention heads, $|\mathbf{x}|$ is the number of tokens in the input $\mathbf{x}$ and $\mathbf{A}$ is the $N_L \times N_H \times |\mathbf{x}| \times |\mathbf{x}|$ tensor of attention values produced by passing $\mathbf{x}$ through a transformer.

The second approach is one of two methods proposed in [1] called Attention Rollout, which instead multiplies the attention matrices of each layer together to generate a combined attention matrix. First we average the attention values in each layer across attention heads, and then matrix multiply the averaged values of each layer together. To convert the final $n \times n$ attention matrix to n per-token values, we take the average attention paid to each token across all other tokens. Formally, for an input token i, this is

$$AttentionRollout_i(\mathbf{A}, \mathbf{x}) = \frac{1}{|\mathbf{x}|} \sum_{j=0}^{|\mathbf{x}|} (\prod_{l=0}^{N_L} \frac{1}{N_H} \sum_{h=0}^{N_H} \mathbf{A}_{lh})_{ji}$$

2.2.3 Evaluating Explanations

While methods for generating explanations from otherwise uninterpretable models are important, designing frameworks for evaluating the quality of those explanations is equally important. If we have no way of ensuring that these methods accurately explain the models they are applied to, then using them in practice, with the risk of misleading and incorrect explanations, will be ineffective and potentially dangerous. This section describes three approaches to evaluating explanation faithfulness. In one, we construct datasets where the features relevant to prediction are known, and then measure the extent to which feature importance explanations highlight those ground truth features. The second involves intervening on the model inputs, based on their explanations, and measuring the impact these interventions have on the predictions of the original model. The third randomizes various parts of the model, and compares the similarity of explanations between the original and randomized models.

2.2.3.1 Datasets with Known Feature Importances

One of the primary difficulties in evaluating explanation methods is separating the behavior of the underlying model from those of the explanation. An explanation that looks incorrect or unusual to a human could be incorrect, or it could just be the result of unintuitive behavior in the model being explained. In this latter context, faithful explanations could still appear incorrect, solely because of the underlying model. To get around this issue, several works have proposed evaluation setups that construct synthetic or semi-synthetic datasets where the relevance of different features for prediction are known [6, 68, 65]. By training models on these datasets where only certain features are useful for prediction, we know that those models that perform well must use those features. Explanations can then be evaluated by the extent to which they rank the known relevant features as more important than irrelevant ones.

The most relevant example of this for this work is [68], which proposes a general framework for generating semi-synthetic datasets of this sort. This framework starts by randomizing the original labels of the dataset. This ensures (with a high probability) that the features in the data that were originally correlated with the labels (and would therefore be useful for prediction) no longer are. After this, an 'input manipulation' is applied to each datapoint, adding in a new feature whose value depends on the randomized label. These new features of the data are then the only features that are useful to predict the label. [68] gives the example of adding a watermark to an image. If we scramble the image labels so that the original image pixels no longer correlate with the labels, and then add watermarks to the image based on the new labels, then explanations for models trained on this data should highlight the watermark significantly more than any other part of the image. The amount of correlation between the added features and the labels (which represents how important the feature is to prediction), can also be controlled by only adding the new features to a certain percentage of inputs with a particular label, allowing for more fine-grained evaluations of features with varying importances. In any case, the ground truth feature importance can be represented by a binary mask, with 0s everywhere except for the added features. This mask can then be compared to explanations using various different metrics [6]. We use two metrics for our experiments: Ground Truth Overlap and Mean Rank.

Ground Truth Overlap measures how frequently the most highly ranked features in the explanation match those in the label. Generically, this can be written as

$$GTOverlap(A, L) = \frac{1}{|A|} \sum_{i \in \{1 \ldots |A|\}} \mathbb{I}[topk(A_i, \|L_i\|_0) = topk(L_i, \|L_i\|_0)]$$

where A, L are the explanations and ground truth labels for a set of example inputs, A_i is the attribution vector for the ith input, and L_i is the binary ground truth vector for the ith input. $topk(x, k)$ is a function that returns the indices of the k greatest values in x, and $\mathbb{I}[x]$ is the indicator function, which returns 1 if the statement x is true and 0 if it is false. This measure checks if the locations of the highest ranked features in the explanation match the locations of the nonzero values in the label vector, and measures the proportion of the time that that is true over all the examples. Note that the number of features compared can vary per example (the L0 norm $\|L_i\|$ counts the number of non-zero features in the label), but in the SST case each example only has one relevant feature, so $\|L_i\| = 1$ for all samples. Higher values of Ground Truth Overlap are better, as it means the explanations are correctly identifying the most important features.

The second metric is Mean Rank, which was proposed by [6]. Mean Rank measures how much of an explanation's induced ranking, from greatest to least, you must retain before recovering all of the ground truth features. Formally, this is written as

$$MeanRank(A, L) = \frac{1}{|A|} \sum_{i \in 1 \ldots |A|} \min r, \text{ s.t. } topk(L_i, \|L_i\|_0) \subseteq topk(A_i, r).$$

r represents the depth of the ranking that we want to minimize, and we take the smallest value of r that still includes all the ground truth features. Lower values for this metric are better, because small values for r mean the ground truth features tend to cluster at the top of the ranking. Because the length of inputs varies, we also normalize each r in the sum by the length of the input sentence:

$$MeanRankPercentage(A, L) = \frac{1}{|A|} \sum_{i \in 1 \ldots |A|} \min \frac{r}{|L_i|}, \text{ s.t. } topk(L_i, \|L_i\|_0) \subseteq topk(A_i, r)$$

so the resulting values are all between zero and one. This can be seen as the average percentage of an input that must be included (based on an explanation's ranking) to recover the ground truth features.

Both these metrics are useful in situations where all the ground truth feature attributions are known, but that requirement limits their use in many realistic scenarios. In the experiment outlined above, we explicitly construct a dataset where the feature attributions are known, but for natural datasets and more complex tasks, there may be multiple predictive features and feature interactions that complicate the determination of ground truth attributions. In this case, these metrics would not be useful. Take, for example, a dataset with two features x_1, x_2 that are both equally predictive of the label. A classification model trained on this dataset could use either feature to predict the label, and a faithful explanation of that model would assign a high value to either x_1 or x_2. If either feature were chosen as the ground truth attribution, an explanation could appear to fail simply because the underlying model happens to use the other feature. Because of these kinds of concerns, it is critical to know precisely how different features contribute to predictions and what the behavior of the explained model is before using these metrics. Additional, more specific details of this experimental setup are also discussed in Section 3.1

2.2.3.2 Relevance of Explanation Rankings to Prediction

A different approach to estimating the quality of explanations avoids the need for specialized datasets, instead estimating how well explanations identify relevant features by altering the original input based on its explanation, and measuring the difference in prediction between the original and altered datapoints. This is done using two metrics, Sufficiency and Comprehensiveness [24]. In both cases, features are first ranked based on the values assigned to them by a feature importance explanation. This ranking induced by the explanation is then used to determine how the original datapoint is altered. Sufficiency is the difference between the original prediction and the prediction of the datapoint with only a subset of the top ranked features. All other features are replaced with zero or some similarly uninformative value. Comprehensiveness is the same predictive difference, but with the opposite alteration, removing only some of the top-ranked features, and leaving the rest unchanged. A faithful explanation should have a low Sufficiency value and a high Comprehensiveness value. Sufficiency measures the extent to which an explanation highlights features needed for a correct prediction. If the prediction changes little when using only the top-ranked features, then the explanation correctly identifies features that ‘justify’ the model prediction. Comprehensiveness measures how well an explanation captures all the relevant predictive features, rather than just a sufficient set. If we remove the highest ranked features, and the prediction changes only a small amount, then there are additional important features that the explanation ranks as unimportant. High Comprehensiveness then suggests that the explanation captures all or most of the important features near the top of its ranking. Generally, these metrics are averaged over a set of different percentages, i.e. keeping/removing 5%, 10%, etc. of the most important features. Both these metrics are precisely defined as follows:

Definition 2.1 (Sufficiency) *Let* $\mathbf{x} \in \mathcal{X}, \mathbf{a}_\mathbf{x} \in \mathbb{R}^{|\mathbf{x}|}$ *be an input and its corresponding feature attribution explanation, respectively.* $|\mathbf{x}|$ *is the dimension of an input* $\mathbf{x}$*. Let* $m : \mathcal{X} \to \mathbb{R}$ *be the function defined by the black-box model being explained, and* $K \in [0,1]^n$ *be the set of averaging fractions. KeepTop*$_k$: $\mathcal{X} \times \mathbb{R}^{|\mathbf{x}|} \to \mathcal{X}$ *is a perturbation function that takes as input the pair* $(\mathbf{x}, \mathbf{a}_\mathbf{x})$*, and returns a perturbed datapoint,* $\mathbf{x}' =$ *KeepTop*$_k(\mathbf{x}, \mathbf{a}_\mathbf{x})$. $\mathbf{x}'$ *is equivalent to* $\mathbf{x}$ *for the fraction of features* k *with the greatest corresponding values in* $\mathbf{a}_\mathbf{x}$*, and to zero (or an equivalently uninformative value) elsewhere. Sufficiency is then defined as*

$$\textit{Sufficiency}(\mathbf{x}, \mathbf{a}_\mathbf{x}) = \frac{1}{|K|} \sum_{k \in K} m(\mathbf{x}) - m(\textit{KeepTop}_k(\mathbf{x}, \mathbf{a}_\mathbf{x}))$$

Definition 2.2 (Comprehensiveness) *Let* $\mathbf{x}, \mathbf{a}_\mathbf{x}, m$ *and* K *be defined as in 2.1.* $RemoveTop_k : \mathcal{X} \times \mathbb{R}^{|\mathbf{x}|} \to \mathcal{X}$ *is a perturbation function that returns an altered datapoint* $\mathbf{x}'$ *based on the input pair* $(\mathbf{x}, \mathbf{a}_\mathbf{x})$*.* $\mathbf{x}'$ *is equal to* $\mathbf{x}$ *for all features except the fraction* k *with the greatest attribution values, which are set to zero. Comprehensiveness is then defined as*

$$\textit{Comprehensiveness}(\mathbf{x}, \mathbf{a}_\mathbf{x}) = \frac{1}{|K|} \sum_{k \in K} m(\mathbf{x}) - m(\textit{RemoveTop}_k(\mathbf{x}, \mathbf{a}_\mathbf{x}))$$

As a simple example, let m be a sentiment prediction model, that takes in a sentence and outputs the probability that the sentiment of the sentence is positive. If an input sentence were "he is happy", we would expect m to output a high probability, since the sentence is clearly positive. A good feature importance explanation for this input could be something like $\mathbf{a} = [0.02, 0.01, 3.4]$, with a significantly higher value for the word ‘happy’. If we remove or mask the bottom two tokens to compute Sufficiency, setting k from above to 0.66, we would have

"[MASK] [MASK] happy."[4] We would expect m("[MASK] [MASK] happy") to be close to the original prediction, since 'happy' is the only word in the input that is clearly positive.[5] To compute Comprehensiveness, we remove the top word, setting k to 0.33, resulting in "he is [MASK]". We expect m("he is [MASK]") to be lower than the original prediction, as "he is" does not have a strong positive or negative connotation. By performing these input alterations, and averaging Sufficiency and Comprehensiveness values across many examples and values of k, we can measure the extent to which different explanation methods identify features that are actually relevant to model predictions. However, although these measures avoid the need for ground truth feature attributions, they are not always reliable. The alteration procedure for computing Sufficiency/Comprehensiveness can produce new inputs that are out-of-distribution (OOD) for the original model m. OOD inputs are those that come from a different distribution than the data m was trained on, and performance on those OOD inputs may not be reflective of behavior on more realistic in-distribution examples [28].

2.2.3.3 Model Invariances

A third approach to measuring explanation quality takes a different view compared to the previous two approaches. Rather than focusing on how well explanations correlate with changes in prediction or with ground truth labels across many inputs, we can instead focus on how sensitive they are to changes in the model itself. If we alter the parameters of the model, then we would expect our explanations to change as well. If an explanation method is invariant to the model

[4]There are many different ways to perform this alteration. Here, we just use a generic mask token which is common in the pretraining objectives of models like BERT. However, tokens could be simply removed, or replaced with other kinds of uninformative baseline values.

[5]Generally speaking, this is something of a simplification. Because language is so deeply compositional, assigning sentiments to individual words like this is not always justified. For example, the phrase 'not happy' is certainly not positive, despite containing a word frequently associated with positivity. See [19] for a more extended discussion on this issue.

parameters, then it is not faithful to the model, since those parameters determine the model behavior and outputs. By randomizing portions of the model parameters, generating new explanations for it, and then comparing those explanations to those generated for the original, we can measure this invariance. This approach was first proposed in [2] and applied to a limited set of language models in [32]. Specifically, [2] provides two different randomization approaches for measuring explanation invariance, both generally applied to deep neural networks with multiple layers. In one, called cascading randomization, each layer of the model is randomized in succession, starting from the last and moving backwards. After reaching the input layer, the whole model has been randomly initialized. At each layer, after it is randomized, a new explanation is generated. The rank correlation between these explanations and the explanation for the unrandomized model then measures how sensitive an explanation method is to the model parameters. As more of the model is randomized, we expect a good explanation method to produce less correlated explanations. The second approach, called independent randomization, is similar, but randomizes each layer by itself, rather than in cascading fashion. All the other model parameters except for the one layer under consideration remain the same, so we can isolate the impact of individual layers on explanations. As in the cascading case, a new explanation is generated for each layer, and the correlations between those and the original explanation measure the explanation's sensitivity to the model.

As an example, Figure 2.3, from [2], shows a visualization of the cascading randomization approach, and how the explanations for an image model change (or do not for some methods) as successive layers of the model are randomized. SmoothGrad, for example, appears more sensitive to the model parameters than Guided GradCam. The image of the bird in the saliency map gradually disappears for SmoothGrad, and is entirely gone after 10 layers of randomization. Guided Grad-

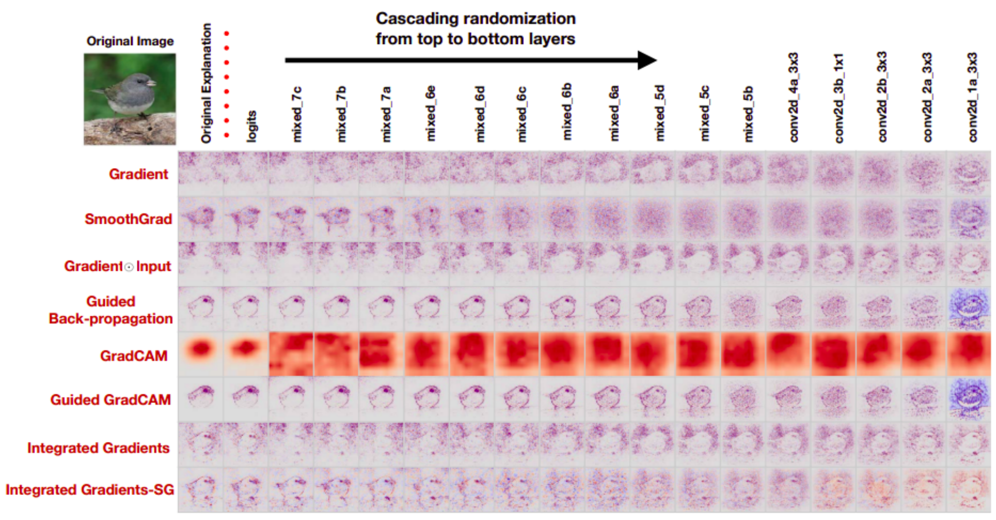


Figure 2.3: Explanations after Cascading Randomization for an ImageNet example on InceptionNet v3 from [2]

Cam, in contrast, includes the bird image clearly at every layer, showing that the explanations do not depend on the model parameters (and instead rely on artifacts in the input).

As mentioned above, the similarity between explanations is usually measured using the Spearman rank correlation, also called Spearman's ρ [51]. This measures the similarity between the rankings induced by two random variables, as opposed to measuring the similarity between their actual values like the standard Pearson correlation coefficient. More formally, let $R : \mathbb{R}^d \to \mathbb{N}^d$, be a function mapping feature attributions $\mathbf{a}_{\mathrm{x}i}, \mathbf{a}_{\mathrm{y}j} \in \mathbb{R}^d$ to their corresponding rankings from greatest to least (e.g. $R([.5, .3, .9]) = [2, 3, 1]$). For n samples $\mathbf{a}_{\mathrm{x}} = (\mathbf{a}_{\mathrm{x}1}...\mathbf{a}_{\mathrm{x}n})$ and $\mathbf{a}_{\mathrm{y}} = (\mathbf{a}_{\mathrm{y}1}...\mathbf{a}_{\mathrm{y}n})$, we can write the Spearman rank correlation as

$$\rho_{\tilde{R}(\mathbf{a}_{\mathrm{x}}),\tilde{R}(\mathbf{a}_{\mathrm{y}})} = \frac{cov(\tilde{R}(\mathbf{a}_{\mathrm{x}}), \tilde{R}(\mathbf{a}_{\mathrm{y}}))}{\sigma_{\tilde{R}(\mathbf{a}_{\mathrm{x}})}\sigma_{\tilde{R}(\mathbf{a}_{\mathrm{y}})}}$$

where $\tilde{R}(\mathbf{a}_{\mathrm{x}}) = (R(\mathbf{a}_{\mathrm{x}1})...R(\mathbf{a}_{\mathrm{x}n}))$ and $\tilde{R}(\mathbf{a}_{\mathrm{y}}) = (R(\mathbf{a}_{\mathrm{y}1})...R(\mathbf{a}_{\mathrm{y}n}))$ are the rankings for all samples, $cov(\tilde{R}(\mathbf{a_x}), \tilde{R}(\mathbf{a_y}))$ is the covariance between the rankings of $\mathbf{a_x}$ and $\mathbf{a_y}$, and $\sigma_{\tilde{R}(\mathbf{a_x})}, \sigma_{\tilde{R}(\mathbf{a_y})}$ are the standard deviations for both rank variables. This is a suitable choice for comparing feature importance explanations because it measures how the relative importance of the features, irrespective of their exact attribution values, changes as we randomize model parameters. If a ranking stays the same as parameters are randomized (meaning a high value of $\rho_{\tilde{R}(\mathbf{a_x}),\tilde{R}(\mathbf{a_y})}$), then the explanation method highlights the same features as important regardless of the underlying model behavior. Additional details on the experimental setup are given in Section 3.3.

2.3 Adversarial Examples for Explanations

Adversarial examples [22] were originally proposed as a way to demonstrate potentially unintuitive behaviors in machine learning models. Given an input and a trained model, a new input, the **adversarial example**, is generated via an optimization procedure. The objective of the optimization forces the adversarial example to be similar to the original input but to have a different prediction from the trained model. For example, [52] proposes a method for generating a single pixel perturbation of images that changes the prediction of an image classifier. Because the ground truth class of the image does not change with alteration of one pixel, the adversarial example is exploiting idiosyncrasies of the classifier, which would preferably not exist in a more robust model, to change the prediction.

Later work applies a similar framework to generating adversarial examples for explanations [21]. In this context, rather than attempting to change the output of a model with a minimal perturbation of the input, the goal is to change the explanation of the input, leaving the output unchanged. It is worth noting that this approach only applies to local explanations, as there is no datapoint to perturb or optimize with respect to for global explanation methods. [21] proposes an approach for generating adversarial examples for feature importance explanations, optimizing the following objective:

$$\arg\max_{\delta} \mathcal{D}(\mathbf{a}_{\mathbf{x}}, \mathbf{a}_{\mathbf{x}+\delta}), \text{ s.t. } ||\delta||_{\infty} \leq \epsilon, \ \arg\max(m(\mathbf{x})) = \arg\max(m(\mathbf{x}+\delta)).$$

δ is the perturbation of the original input $\mathbf{x}$. $\mathbf{a}_{\mathbf{x}}, \mathbf{a}_{\mathbf{x}+\delta}$ are the feature importance explanations for $\mathbf{x}$ and the adversarial example $\mathbf{x}+\delta$, respectively. $||\delta||_{\infty}$ is the max norm of the perturbation, which limits the largest component of the perturbation to be smaller than ϵ, a hyperparameter. $\mathcal{D} : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a distance function, which measures the similarity between two input explanations. Lastly, the constraint $\arg\max(m(\mathbf{x})) = \arg\max(m(\mathbf{x}+\delta))$ ensures that the prediction of the adversarial example, $m(\mathbf{x}+\delta)$ matches that of the original input $m(\mathbf{x})$. Maximizing $\mathcal{D}$ while satisfying the given constraints then produces a pair of datapoints $(\mathbf{x}, \mathbf{x}+\delta)$ that have the same predicted class, but different explanations (at least to the extent that $\mathcal{D}$ can be increased without violating any constraints).

[21] proposes three different choices (called 'attacks' in the paper) for the distance function $\mathcal{D}$. In this work we focus on the **top-k attack**, which defines $\mathcal{D}$ as

$$\mathcal{D}(\mathbf{a}_{\mathbf{x}}, \mathbf{a}_{\mathbf{x}+\delta}, k) = \sum_{i \in topk(\mathbf{a}_{\mathbf{x}}, k)} -\mathbf{a}_{\mathbf{x}+\delta, i}$$

where k is an integer hyperparameter between 1 and $|x|$, and $topk(\mathbf{x}, k)$ returns the indices of the greatest k elements in $\mathbf{x}$, ranked according to their attribution values $\mathbf{a}_{\mathbf{x}}$. This choice of $\mathcal{D}$ attempts to minimize the importance values assigned to the top k ranked elements in the original explanation. An ideal adversarial example would then assign very low importance values to the k elements that have the greatest values in the original explanation. This is an important choice of attack to assess, as explanation methods are frequently used in practice to determine the relative importance of features, relying on their rankings as opposed to the attribution values themselves. Adversarial examples that can change the rankings of explanations without altering the prediction would then raise questions as to the validity of ranking features by their attribution values.

The formulation proposed in [21] also allows for significant changes in model output, raising possible problems in interpreting the relationship between adversarial examples and the original inputs. Because the constraint between $m(\mathbf{x})$ and $m(\mathbf{x} + \delta)$ only requires that the predictions are equal, the probabilities that determine those predictions can still fluctuate. For example, an input with a predicted probability of 51% for a particular class could be an adversarial example with respect to an input with a 99% predicted probability for the same class. This is an undesirable property when we are generating adversarial examples for explanations. While two inputs may have the same predicted class, it is possible that different combinations of features were important in determining each prediction. In this case, treating one input as an adversarial example relative to the other would be misleading, as faithful explanations of the two inputs should give high attribution values to two different sets of features. Optimizing for adversarial examples that only match predictions then requires assuming that the same set of features determines all predictions within a particular class.

A more preferable adversarial example would be one that has a different explanation, but that also has a similar output probability, rather than just the same output prediction. This is a stronger constraint, and helps ensure that adversarial examples produce outputs that closely match the original datapoint. By strengthening this constraint, and forcing adversarial examples to be more similar to their original inputs, we make it more likely that the relevant features for the original and adversarial inputs are the same. When the important features for prediction are the same for adversarial and original inputs, then we can claim that any discrepancy between the explanations is the result of unexpected and potentially undesirable behavior of the explanation method. Otherwise, those discrepancies could be the result of changes in the relevant underlying features for the two different inputs, which are independent of the explanation method.

By augmenting the original objective from [21], we can additionally optimize for examples that match the output probabilities of the original inputs:

$$\arg\max_{\delta} \alpha\mathcal{D}(\mathbf{a}_{\mathbf{x}}, \mathbf{a}_{\mathbf{x}+\delta}) - \beta(m(\mathbf{x}) - m(\mathbf{x}+\delta))^2 \text{ s.t. } ||\delta||_\infty \leq \epsilon.$$

This new objective optimizes the same distance function $\mathcal{D}$, subject to the same perturbation constraint ϵ, but the prediction constraint is replaced by $(m(\mathbf{x}) - m(\mathbf{x}+\delta))^2$, the squared error between the outputs of the original input and the adversarial example.[6] $\alpha, \beta \in \mathbb{R}^+$ are hyperparameters determining the tradeoff between maximizing the difference between explanations and minimizing the difference in output predictions. Optimizing this objective forces adversarial examples to be close in probability, not only in prediction, by minimizing the squared error between the outputs. In the experiments, we compare both methods, and compare the impact of each one on both the explanations and predicted probabilities.

To assess the quality of the adversarial examples, we use three measures. The first two relate to the impact of the adversarial examples on explanations. The first, denoted "Rank Change" in the results, measures the percentage of the top k indices, sorted by their attribution values, that change between the original and

[6]Note that this assumes the output of m is a scalar for simplicity. In the multiclass case, you could either choose the probability of the predicted class or optimize for the sum of squared errors across all classes. In the experiments, we choose the former and optimize for similarity with the predicted classes probability.

adversarial explanation. For example, if the top three indices for the original explanation were $[12, 3, 7]$ and the top three indices for the adversarial explanation were $[3, 15, 7]$, then the Rank Change value would be 0.66, as the first and second rank values are different. Formally, we can write this as

$$RankChange(\mathbf{a_x}, \mathbf{a_{x+\delta}}, k) = \frac{1}{k}\sum_{i=0}^{k} \mathbb{I}[topk(\mathbf{a_x}, k)_i = topk(\mathbf{a_{x+\delta}}, k)_i]$$

where $topk(x, k)$ returns the indices of the top k elements in x, and $\mathbb{I}[x]$ is the indicator function, which is one if x is true and zero otherwise.

The second measure uses the attribution values directly, rather than their induced ranking, and is labeled "Top-k Sum Change" in the results. First, we sum the results of the top k attribution values for the original explanations, and then sum the attribution values in the same positions for the adversarial explanation. The difference between the two sums is then the Top-k Sum Change. Mathematically, this is:

$$TopkSumChange(\mathbf{a_x}, \mathbf{a_{x+\delta}}, k) = \sum_{i \in topk(\mathbf{a_x}, k)} \mathbf{a}_{\mathbf{x},i} - \mathbf{a}_{\mathbf{x}+\delta,i}$$

These two measures allow us to assess how effectively the adversarial examples change the explanations of the original inputs. RankChange measures the impact on the rankings of different features, while Top-k Sum Change measures the extent to which the attribution values themselves are reduced.

The remaining measure focuses on the impact of the adversarial example on the model output, rather than on the explanation. Labeled "Probability Change" in the results, this measure is the difference between the output probability of the predicted class for the original input and the adversarial input:

$$ProbabilityChange(\mathbf{x}, \mathbf{x} + \delta) = m(\mathbf{x}) - m(\mathbf{x} + \delta).[7]$$

As discussed earlier, ideal adversarial examples in this context should not change the output probability significantly while still changing the accompanying explanation. Therefore, the best adversarial examples should have high values for Rank Change and Top-k Sum Change, and low values for Probability Change. Conversely, explanation methods that are robust to adversarial perturbations should have low values for Rank Change and Top-k Sum Change. The Probability Change value only depends on the adversarial datapoint, not the explanation method, so its value is unimportant for assessing adversarial robustness, except insofar as it indicates that an adversarial example is valid (i.e. close in probability to the original input) or not.

[7]As mentioned earlier, this assumes a scalar output. The multiclass version is identical after choosing the greatest probability from the output vector m(x)

3 | Experimental Setups

With the general evaluation frameworks and explanation methods defined, we can now move on to the specific experiments conducted in this work. This section outlines the five sets of experiments we use to evaluate post-hoc explanations for language models. The first alters a well-known sentiment analysis dataset so that it has known feature attributions, enabling us to compare the fidelity of various explanation methods against ground truth feature importances. The second evaluates Sufficiency and Comprehensiveness on natural language inference tasks, using two different datasets. The third set of experiments uses layer randomization to measure how sensitive different kinds of explanations are to model parameters. The fourth section describes two methods for ensembling explanations together that we evaluate, to assess any potential benefits of combining multiple explanations together. The fifth section describes the experimental setup for evaluating the adversarial robustness of explanations. Lastly, the sixth section describes the two sets of transformer models to which we apply each experiment.

3.1 Adding Known Feature Attributions to the Stanford Sentiment Treebank

The goal of this experiment is to measure the faithfulness of an explanation with a simple, synthetic setup (Section 2.2.3.1). Applying the framework proposed in [68] to a sentiment analysis task, we design a dataset in which a single word is clearly predictive of the label. In this context, faithful explanations should always assign the predictive word the greatest value. [68] apply their framework to evaluate attention-based explanations produced by a bidirectional LSTM + Attention model, but we extend their results by evaluating transformer models across a greater number of explanation methods.

The Stanford Sentiment Treebank (SST) [50] dataset is designed for sentiment classification. The dataset consists of a set of movie reviews, and accompanying binary labels, with 0 indicating a negative review and 1 a positive review. The task is to predict from the review whether it is positive or negative in tone. Using the framework from Section 2.2.3.1, we augment this dataset with an additional feature that determines the labels. To this end, we first randomize the labels of the original dataset, to decorrelate them from the original features. Then, each review has either the word "positive or the word "negative" appended to the end of the sentence, depending on whether the label (post-randomization) is one or zero. The feature importances for the added "positive"'s and "negative"'s should be significantly higher than all other tokens for faithful explanations. An example augmentation for one (sentence, label) pair could work as follows:

Unmodified Datapoint: "the film was good", 1

Label Modification: "the film was good", 0

Feature Modification: "the film was good negative", 0

In this experiment, we use the version of SST included in the GLUE benchmark [59], which breaks up some of the original reviews into shorter examples than the original SST dataset. The task and labels are identical to the original SST. We use this version because training and explaining the shortened reviews is less computationally expensive, allowing us to run it on the available resources. Moreover, lengthier examples would provide little additional insight beyond making the task harder (and several explanation methods already perform poorly on the shortened reviews), as we are interested in isolating a single added token. The results of this experiment are given in Sections 4.1.1 and 4.2.1.

3.2 Evaluations on Natural Language Inference Tasks

The augmented SST dataset allows us to precisely isolate the relevant features for prediction, but it is also a synthetic design. This makes it difficult to reason about how explanation methods may perform on more realistic or complex tasks. To evaluate that case, we measure Sufficiency and Comprehensiveness (Section 2.2.3.2), which do not require ground truth feature attributions, on the task of natural language inference (NLI).

Natural language inference is a three-class classification task. A model takes as input two pieces of text, a premise and a hypothesis. It then predicts whether the premise entails the hypothesis, contradicts it, or is unrelated to it. For example, the premise "he walked the dog" and the hypothesis "the dog was walked" would be labeled as entailed. Replace the hypothesis with "they cooked dinner" or "the dog stayed at home" and the label would change to unrelated or contradiction, respectively. We evaluate explanations on two NLI datasets, MultiNLI [62] and e-SNLI [12].

A small implementation detail for NLI tasks is worth noting here. While the premises and hypotheses are given as two separate inputs, they are usually treated as one string of text for transformer models. Each premise and hypothesis is concatenated together (generally with a model-specific end-of-sentence token in between) and passed to the model as a single input.

The following two sections provide brief overviews of MultiNLI and e-SNLI datasets.

Premise: An adult dressed in black holds a stick.
Hypothesis: An adult is walking away, empty-handed.
Label: contradiction
Explanation: Holds a stick implies using hands so it is not empty-handed.

Premise: A child in a yellow plastic safety swing is laughing as a dark-haired woman in pink and coral pants stands behind her.
Hypothesis: A young mother is playing with her daughter in a swing.
Label: neutral
Explanation: Child does not imply daughter and woman does not imply mother.

Premise: A man in an orange vest leans over a pickup truck.
Hypothesis: A man is touching a truck.
Label: entailment
Explanation: Man leans over a pickup truck implies that he is touching it.

Figure 3.1: Example Rationale Annotations from e-SNLI

3.2.1 MutliNLI

MultiNLI is an NLI dataset designed to cover a broad range of different kinds of text. Examples in MultiNLI come from 10 different genres of text such as transcriptions of telephone conversations, fictional works, and face-to-face conversations. It has approximately four hundred and thirty-three thousand examples in total, with 392,702 in the training set, and 20,000 each in the validation and test sets.

3.2.2 e-SNLI

e-SNLI is based on the Stanford Natural Language Inference (SNLI) corpus [8]. SNLI was generated by having annotators write alternate captions for captioned images from the Flickr30K corpus [66]. Annotators were instructed to generate new captions that were definitely true, potentially true, or definitely false based on the original caption. The pairs of original and alternate captions were then used as premises and hypotheses in the SNLI dataset. e-SNLI adds human annotated rationales to the original SNLI examples. Rationales in this case are just the snippets of the premise or hypothesis necessary to justify the label. Annotators highlighted these relevant snippets for each example, and provided a natural language explanation for their highlights. Figure 3.1, from [12], shows a few examples with the rationale snippets highlighted and the annotator explanations. There are 570,000 examples total, with 550,000 in the training set and 10,000 in the validation and test sets. We use the version of e-SNLI provided by the ERASER benchmark [15], which only includes the highlighted rationales. This is because the additional natural language explanations are not needed to compare to feature importance explanations. The results for this experiment are given in Sections 4.1.2, 4.1.3, 4.2.2 and 4.2.3.

3.3 Layer Randomizations and Explanation Sensitivity

For any of the models trained on the datasets in Sections 3.1 and 3.2, we can additionally randomize their parameters and measure the sensitivity of different explanation methods as described in Section 2.2.3.3. Rather than randomizing individual layers, we randomize attention blocks as single units (consisting of a self-attention layer and the following MLP). The results for this experiment are given in Sections 4.1.4 and 4.2.4.

3.4 Ensembles of Explanations

In addition to evaluating the faithful of individual explanation methods, we also combine multiple explanations into ensembles. An ensemble of methods has the potential to be more consistent than any single method, if the individual explanations are not all failing on the same examples. Let $\mathbf{x}$ be an example datapoint,

$\mathbf{x} \in \mathbb{R}^n$. Let $\mathbf{A} = (\mathbf{a}_0...\mathbf{a}_k), \mathbf{a}_i \in \mathbb{R}^{|x|}$ be a set of feature attributions for $\mathbf{x}$ from k different explanation methods. We can combine these explanations by computing the average importance assigned to each feature $x_j \in \mathbf{x}$. The importance for the j-th feature is then

$$EnsembleImportance(A)_j = \frac{1}{|A|}\sum_i A_{ij}{}^1$$

We evaluate two different groups of ensembles using the same models and metrics as the individual methods, to see if combining them improves performance. The first group, denoted "Ensemble (All Methods)", averages all seven of the evaluated methods together. The second, "Ensemble (Top 3)", combines only LIME, SHAP and Integrated Gradients, which consistently outperform the other four methods (see 4). Gradients, Gradients*Input, Average Attention and Attention Rollout frequently perform approximately as well the random baseline in several cases, and so may just be adding noise to an ensemble. This second group helps isolate the effect of only combining methods that are also consistently effective in isolation. The results for the ensemble methods are given in Sections 4.1.5 and 4.2.5.

3.5 Adversarial Examples

To evaluate the robustness of different explanation methods to adversarial inputs, we generate two sets of adversarial examples for each model and dataset. Because we optimize the adversarial examples using gradient descent, the explanation methods we consider must be differentiable. Because of that, we only generate examples for Gradients, Gradients*Input and Integrated Gradients. The

[1]Note that all explanations ensembled here should be normalized to sum to 1, so that contributions to the average are independent of the relative magnitudes of the values produced by different methods.

adversarial examples are generated using the two objectives given in Section 2.3. The first objective, which only requires the predictions of the original and adversarial example to match, is labeled the "Unconstrained" method in the results. The second objective, which attempts to make the output probabilities of the original and adversarial examples as similar as possible, is labeled as "Constrained" in the results. The two hyperparameters α, β for the second objective are both set to one. For both objectives, ϵ, which determines the maximum size of the adversarial perturbation, is set to 2. For each example, the value of k, which determines the number of attribution positions to try and minimize in the distance function $\mathcal{D}$, is set to 15% of the length of the sentence, rounding down. Each example is optimized using gradient descent for 500 steps, with a learning rate of 0.1. The learning rate is halved every 200 steps. If the difference between the perturbation δ_t at timestep t and δ_{t-1} from the previous step is smaller than 1e-5 for 20 steps, then the optimization is terminated early. The results for this experiment are given in Sections 4.1.6 and 4.2.6.

3.6 Model Architectures

For both of the evaluation datasets, and the layer randomization, ensemble, and adversarial experiments, we evaluate language models along two different dimensions. One set of models has the same architectures and pretraining regimens, but differ in size. The other has models of similar sizes, but different architectures and pretraining objectives. Overviews of the different models and training parameters are provided in the rest of this section.

3.6.1 Scaling Comparisons

To compare explanations generated for models of different scales, we use four pretrained T5 [42] models of different sizes. The T5 architecture is roughly similar to the original transformer paper [57], with a few minor changes. T5 uses an altered form of layer normalization (without the usual bias term) and replaces the fixed positional embeddings in the original Transformer with relative positional embeddings [47]. The four pretrained models used in our experiments come from [54], which trained and released a large number of different T5 models of varying configurations. Specifically, we use the Tiny, Mini, Small and Base models from [54], which are all structurally similar but increasing in size. The four models have approximately 16 million, 31 million, 60 million and 220 million parameters respectively. All four models were pretrained for 2^{19} steps on the Colossal Cleaned Common Crawl Corpus [16]. For these experiments, because we only consider classification tasks and not sequence generation, only the encoder portion of these models is needed. The final parameter counts without the decoders are approximately 11, 20, 35 and 110 million (following the same order as above).[2]

Name	Number of Parameters (Millions)
T5 Tiny	11
T5 Mini	20
T5 Small	35
T5 Base	110

Table 3.1: Number of Parameters in the Different Neural Architectures used in the Scaling Experiments

[2]The parameter counts are not strictly half the original values due to shared parameters (like word embeddings) between the encoder and decoder

3.6.2 Architecture Comparisons

To compare language models with different architectures, pretraining objectives and pretraining data, we finetune four commonly used pretrained models of similar sizes: BERT [14], T5 [42], GPT2 [41], and RoBERTa [35]. Each of these models has around 110 million to 120 million parameters. Additional details about the structure and pretraining of each model follows.

3.6.2.1 BERT

BERT [14] is an encoder-only transformer pretrained on the BooksCorpus dataset [69] and text passages scraped from English wikipedia. The attention blocks in the BERT model are identical to those in the original transformer encoder. The pretraining tasks are masked language modeling, as described in section 2.1.3, and a next sentence prediction task. For this second task, a pretraining sentence is input to the model along with the correct next sentence 50% of the time and a random sentence from the dataset the other 50%. Sentences are separated by a '[SEP]' token added between them. The embedding of a special '[CLS]' token appended to the beginning of each input is then used to predict if the next sentence is the correct one. [14] include this additional pretraining task to improve performance on question-answering and natural language inference tasks. In these experiments, we use the BERT-base model, which has twelve layers, 768-dimensional embeddings, and a 30,000 token vocabulary. It has approximately 109 million parameters.

When finetuning BERT for classification tasks, the standard approach is to use the embedding for the '[CLS]' token as a representation of the whole sentence. To make a prediction, only the embedding for '[CLS]' is taken from the output and passed through the final classification layer. This means that the embedding for that token adjusts during finetuning to represent the task-relevant information for the whole sentence.

3.6.2.2 T5

T5 [42] is an encoder-decoder model. It is similar to the original transformer architecture, with the modifications already mentioned in section 3.6.1. It is worth noting that we use the base T5 model released from [42] for these experiments, not the one from [54] used in the scaling experiments. This version was pretrained on the Colossal Clean Crawled Corpus as well, but was additionally trained on a mixture of supervised language tasks, totaling approximately 1 trillion tokens of pretraining data. This model has 12 layers each for the encoder and decoder, and an embedding dimension of 768. Its vocabulary size is 32,128. This amounts to approximately 220 million parameters in total, which becomes around 110 million in this case since the decoder is unneeded. Unlike BERT, T5 does not add a special classification token to the start of the input, so we instead use the average of the output embeddings as input to the classification layer.

3.6.2.3 GPT2

GPT2 [41] is a decoder-only transformer. It is generally similar to the decoder of the original transformer, but uses a gaussian error linear unit (GELU) as its activation function, instead of ReLU, and learned positional embeddings, instead of static ones. It also does not include the cross-attention layers in the original transformer decoder (as there is no encoder output to cross-attend to in this context). The model is pretrained on the WebText dataset, which consists of ~40GB of text scraped from various websites.

Because it is a decoder model, elements in the input sequence only have non-zero attention values for tokens to their left. For finetuning on classification tasks, then, we take the embedding of the last token, which can attend to the entire input, as a representation of the sentence and use it as input to the classification head. The complete GPT2 model has approximately 1.5 billion parameters, which is clearly significantly larger than the other models described here. We use the smaller version released in the same paper (see Table 2 in [41]), which was pretrained in the same way. This model has 12 layers, 768 dimensional embeddings and 50,257 token vocabulary. The total number of parameters is around 124 million.

3.6.2.4 RoBERTa

RoBERTa [35] is an encoder-only model, nearly identical to BERT. RoBERTa was published as a replication of BERT that thoroughly examined the impact of training and architectural parameters on performance. The encoder itself is the same in both cases, but RoBERTa has a larger vocabulary than BERT (50,000 vs. 30,000), is pretrained on significantly more data and with larger batches, and does not use the next-sentence prediction task from BERT during pretraining. Classification finetuning is done in the same way as BERT, taking the '[CLS]' token as a representation of the sentence. The base RoBERTa model that we use has 12 layers and 768 dimensional embeddings, for a total of ~124 million parameters.

3.6.3 Training

The finetuning process for all eight models and both datasets described above are the same. All models are finetuned for 10 epochs, with a batch size of 16 and a learning rate of 5e-5. The learning rate follows a linear warmup for the first 6% of training steps, and then decays linearly back to zero after that. Finetuning stops early if the validation accuracy fails to increase for 5 epochs. The scheduling and parameters used here are similar to those used across the finetuning literature [14, 54, 35, 41], and our validation performance on these tasks across models after finetuning is roughly on par with prior published results.

Experiment and Model Summary

We use five different experimental setups/datasets to evaluate the quality of different explanation methods:

1. Section 3.1: An augmented version of the SST sentiment analysis dataset where the ground truth feature attributions are known.
2. Section 3.2: Two natural language inference datasets (MultiNLI and e-SNLI), where we evaluate explanations using Sufficiency and Comprehensiveness (Section 2.2.3.2)
3. Section 3.3: The parameter randomization tests proposed in [2], where explanation sensitivity is measured using the Spearman rank correlation between explanations of the original model and the randomized ones.
4. Section 3.4: Two different ensembles of explanation methods, to assess if combining methods together can improve performance.
5. Section 3.5: Two different objectives for generating adversarial examples, to evaluate the adversarial robustness of different explanation methods.

These five evaluations are applied to eight different language models. Four of them are T5 models with identical pretraining data, and similar architectures, which differ only in their number of parameters. The other four are different pretrained models, but of similar size: T5 Base, BERT Base, RoBERTa Base and GPT2 Small. Each of these eight models are trained on the two datasets described above, and layer randomization and ensemble results are generated for each one. This results in a total of twenty-four models for comparison.

4 | Results

This section shows the results for each of the previously described experiments. We finetune each model (four in the scaling setting, and four in the architectural setting) on each dataset, for a total of 24 models. Next, we generate explanations for each model using seven different explanation methods. The results here compare those explanations using the evaluation metrics relevant for each dataset (described in Section 2.2.3). There is also a random explanation included as a baseline. A random explanation is generated by taking a random permutation of $(1...|\mathbf{x}|)$, where $|\mathbf{x}|$ is the number of tokens in an input sequence $\mathbf{x}$. The permutation then serves as a random ranking of the features.

The layer randomization results in each section only include five of the seven explanation methods (excluding LIME and SHAP). LIME and SHAP train surrogate models to approximate the input-output behavior of the underlying model (and do not use the parameters directly as the other five methods do), therefore the layer randomization setup does not apply to them. The adversarial example results include only the three gradient-based methods, Gradients, Gradients*Inputs and Integrated Gradients, as they are fully differentiable.

4.1 Scaling

The following figures show comparisons between the number of parameters in four T5 encoders (ranging from approximately 10 million to 110 million) and the quality of explanations (as measured by several different metrics). Each line in each graph represents a different explanation method, and each of the four data points determining a line are determined by the explanations for that method of outputs from one of the four differently-sized T5 models. All shaded regions represent 95$\%$ confidence intervals[1], and all values are averaged over explanations for 500 samples, with the exception of the layer randomization results, which are averaged over 50 samples.

The layer randomization experiment requires generating $2L$ explanations per model (L explanations each for both the independent and cascading cases), where L is the number of layers in the model. Given the number of models evaluated, their depth, and the complexity of the explanatory methods[2], we use 50 samples as it is computationally feasible yet provides a sufficient number of examples for a good quality explanation.

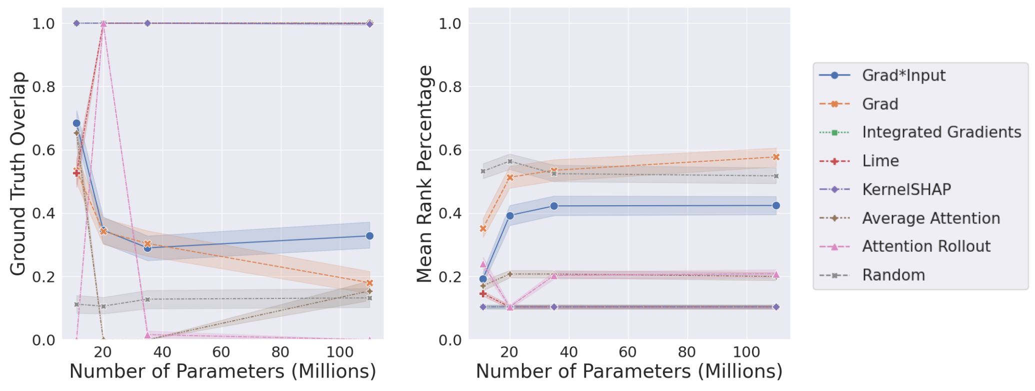
The adversarial example experiment is more computationally intensive than the layer randomizations, as generating an adversarial example requires a gradient-descent optimization procedure. At each step of gradient descent, forward passes of the underlying language model and backward passes to generate explanations are required. This can require 500 forward and backward passes per example (or more in the case of Integrated Gradients, which uses multiple backward passes). Because of the compute-intensive nature of this experiment, we generate adversarial examples for 16 datapoints for each model-dataset combination.

4.1.1 SST

For the SST experiments, the ground truth feature importances are known because of how we construct the dataset (described in Section 3.1). Consequentially, the generated explanations can be compared to the ground truth to measure their faith-

[1]Confidence Intervals are computed via the bootstrapping procedure built into the Seaborn plotting library (https://seaborn.pydata.org/)

[2]For example, LIME and SHAP both have approximate complexities of at least $\mathcal{O}(n * m * k^2)$, with n, m, k being the number of datapoints to explain, the number of perturbed samples, and the number of features, respectively. This is ignoring the cost of generating the perturbed samples, which adds an additional $O(n)$ forward passes of the underlying model.

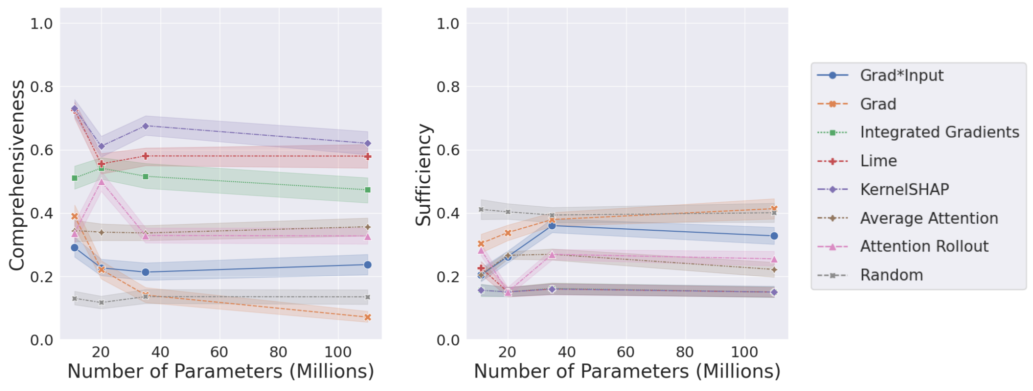


(a) Ground Truth Overlap (b) Mean Rank Percentage

Figure 4.1: Ground truth metrics for explanations of four T5 models finetuned on the augmented SST dataset. The x axis is the number of parameters in each of the four T5 models. The y axis is either the Ground Truth Overlap or Mean Rank value (Section 2.2.3.1). Faithful explanations will have high values for Ground Truth Overlap and low values for Mean Rank. The values are averaged over 500 examples and shown with 95% confidence intervals. Note that the lines for LIME (red), KernelSHAP (purple) and Integrated Gradients (green) overlap for significant portions of both graphs (at 1.0 for figure (a) and 0.1 for figure (b)).

fulness. We do this using two measures: Ground Truth Overlap and Mean Rank (see Section 2.2.3.1).

Figure 4.1 shows the performance of all seven explanation methods across four T5 models of different scales on the two metrics. The best performing methods across all model sizes are Integrated Gradients, LIME and KernelSHAP, which generally always rank the added feature most highly (although LIME performs somewhat poorly for the smallest model). Gradients and Gradients*Input perform significantly worse, only slightly outperforming the baseline random explanations at all but the smallest scale. The two attention explanations have the greatest amount of variation. Focusing on the ground truth overlap results, Figure 4.1(a), Attention Rollout fails to rank the added token as most important at any scale except the 2nd smallest, where it always does so. Average Attention, on the other hand, has better



(a) Comprehensiveness (b) Sufficiency

Figure 4.2: Comprehensiveness and Sufficiency for explanations of four T5 models of different sizes finetuned on the augmented SST dataset. The x axis is the number of parameters in each model. The y axis is the either Comprehensiveness or Sufficiency (Section 2.2.3.2). Faithful explanations should have high Comprehensiveness values and low Sufficiency values. Values are averaged over 500 examples and shown with 95% confidence intervals. Note that in (b), LIME (red), KernelSHAP (purple) and Integrated Gradients (green) all mostly overlap along $y = 0.1$.

performance at the smallest scale, and then fails to prioritize the added token after that. While these differences seem extreme, Figure 4.1(b) provides some explanation. The mean-rank for both Attention Rollout and Average Attention is quite low, fluctuating between 0.2 and 0.1. This means that even in the cases where they fail in terms of ground truth overlap, the correct token is still very highly ranked.

Previous work has noted that gradient-based explanations can be noisy and misleading, due to issues like gradient saturation [53] or sharp fluctuations of the gradient around the datapoint [49]. Factors like these could explain the poor performance of Gradients and Gradients*Input. Attention explanations have also been critiqued in prior literature as potentially unfaithful, due to a lack of correlation with other explanation methods and the ability to construct multiple attention sets of attention values that give equivalent predictions [30]. In this context, those criticisms of attention explanations may hold, and could be compounded by the stacking of multiple attention computations in transformer models, which mixes information across tokens, making the attention values for embeddings in later layers less directly representative of the corresponding input tokens than earlier layers (see Section 2.2.2.3).

Figure 4.2 shows similar results for Sufficiency and Comprehensiveness, the two perturbation based metrics described in Section 2.2.3.2. As mentioned in that section, higher values of Comprehensiveness and lower values of Sufficiency are better. With that in mind, the trends across all model scales match those in the ground truth metrics. LIME, Integrated Gradients and KernelSHAP all have the highest values for Comprehensiveness and the lowest values for Sufficiency. The Comprehensiveness values for all three methods are around 0.5, meaning that when the top ranked tokens are masked out, the prediction tends to decrease by 0.5. This makes sense in this context, as the added 'positive' or 'negative' token will always be the most highly ranked (based on Figure 4.1). When that token is masked out, the sentence becomes neutral (for this synthetic task), and a prediction of 0.5 is expected. A decrease in the predicted probability from 1.0 [3], the prediction with the added token, to 0.5 is then the greatest change we would expect to see.

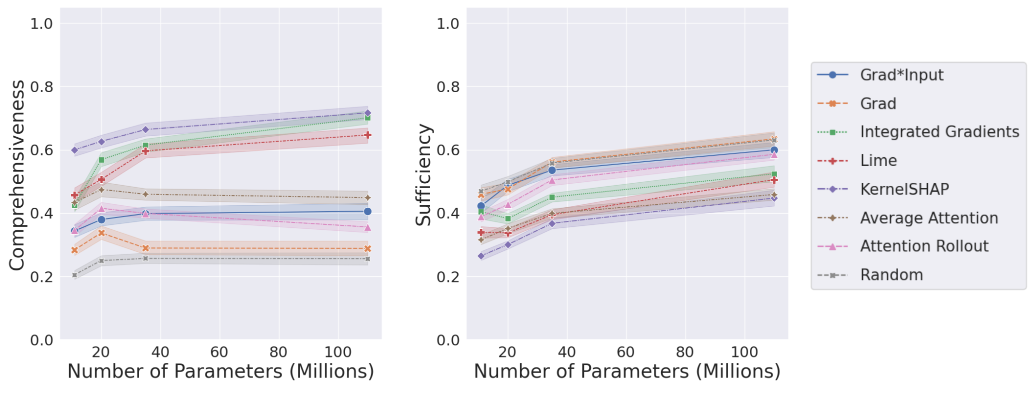
The two attention methods have lower Comprehensiveness values and higher Sufficiency values than the top three methods, making them less faithful (for this task) than those top performing approaches. As shown in the results for the ground truth metrics, the attention methods frequently ranked the predictive token as the second most important token, rather than the most important. Knowing this, these results are sensible. When computing Comprehensiveness, if only the top token is masked out (which would correspond to a low value of k for most sentences,

[3] We know the prediction is 1.0 in this case (or extremely close) because the underlying classification model solves this task perfectly and achieves 0 loss.

see Section 2.2.3.2), but the predictive token is left unmasked, we would see no change in the output prediction. Conversely, for Sufficiency, the predictive token would be more frequently removed if it is ranked lower than first, leading to a greater change in ouput. On average, then, it makes sense that the Comprehensiveness and Sufficiency values for these two methods fall slightly below LIME, SHAP and Integrated Gradients.

The Gradients and Gradients*Input methods have the lowest values for Comprehensiveness and the highest values for Sufficiency. This matches the trends in the ground truth metrics for these two methods, which often did not rank the predictive token the most highly, or place it near the top of the rank (like the attention methods). This is likely due to the more general issues with gradient methods mentioned earlier in relation to the ground truth results [49, 53].

4.1.2 MultiNLI



(a) Comprehensiveness (b) Sufficiency

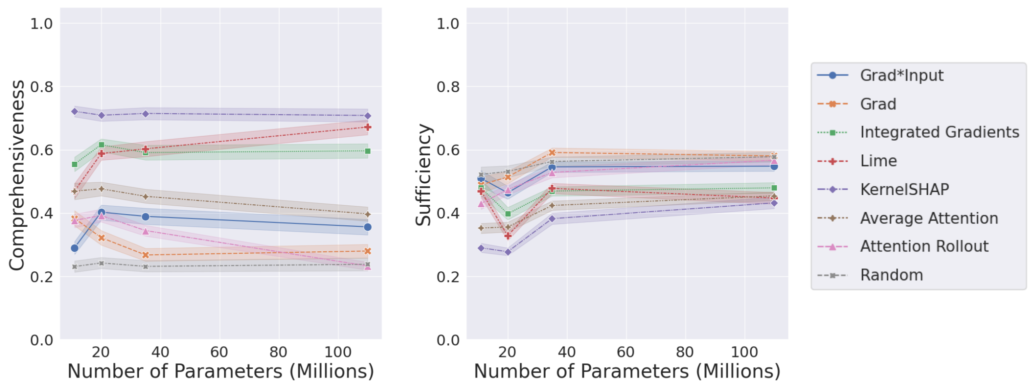
Figure 4.3: Comprehensiveness and Sufficiency for explanations of four T5 models of different sizes finetuned on MultiNLI. The x axis is the number of parameters in each T5 Model. The y axis is the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness values and low Sufficiency values. The values are averaged over 500 examples and shown with 95% intervals.

The augmented SST dataset is a synthetic setup. It allows us to isolate a particular feature that we know is important, but that construction also makes it difficult to know how different explanation methods will transfer to more complex tasks and datasets. Figure 4.3 shows Comprehensiveness and Sufficiency results for the four T5 models trained on MultiNLI, which is a significantly more challenging dataset than the previous SST case. MultiNLI requires inferring the truth of some hypthothetical statement given some premise, which requires more complex linguistic reasoning than the SST experiment, in which the labels are determined by the presence of a single word. The overall trends are consistent with those of the previous results, with LIME, KernelSHAP and Integrated Gradients having the highest Comprehensiveness values and lowest Sufficiency values. However, all explanation methods tend to be less effective for this more complicated task. The highest Comprehensiveness value (for KernelSHAP) is around 10% lower than in the SST case, and the lowest Sufficiency value (also for KernelSHAP) is around 10% higher.

4.1.3 e-SNLI

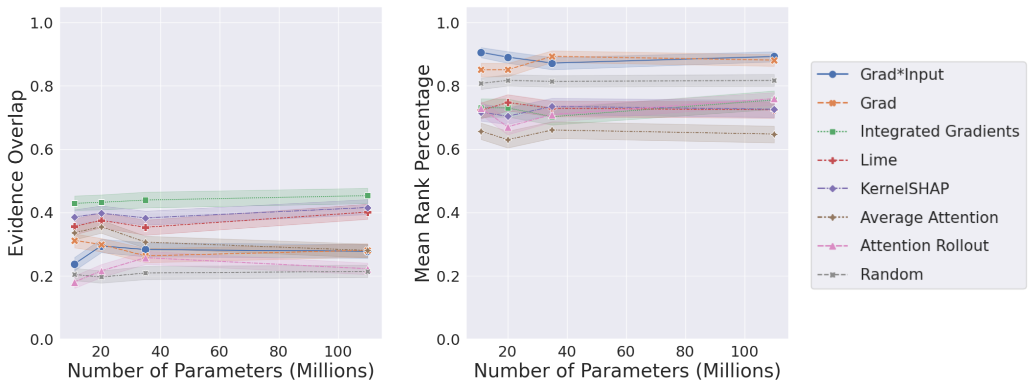
Figure 4.4 shows the Comprehensiveness and Sufficiency results for the models trained on e-SNLI. The overall trends are consistent with the MultiNLI results, in that the same three methods tend to perform best (LIME, Integrated Gradients and KernelSHAP). Certain individual rankings shift (e.g. Integrated Gradients has slightly lower Comprehensiveness compared to LIME for e-SNLI, but is higher for MultiNLI), but the general divide between the top three methods and the rest persists.

e-SNLI also has labels for what words in each input human annotators thought explained each classification. These annotations (generally called rationales) can be used to see how well the explanations from each method correspond to those gen-



(a) Comprehensiveness (b) Sufficiency

Figure 4.4: Comprehensiveness and Sufficiency for explanations of four T5 models of different sizes finetuned on e-SNLI. The x axis is the number of parameters in each T5 model. The y axis is the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness values and low Sufficiency values. The values are averaged over 500 examples and shown with 95% confidence intervals.



(a) Ground Truth Overlap (b) Mean Rank Percentage

Figure 4.5: Ground Truth Overlap and Mean Rank between human rationale annotations and explanations of four T5 models of different sizes finetuned on e-SNLI. The x axis is the number of parameters in each T5 model. The y axis is the Ground Truth Overlap or Mean Rank value, using human rationale annotations as the ground truth for comparison. Plausible explanations should have high Ground Truth Overlap and low Mean Rank values. The values are averaged over 500 examples and shown with 95% confidence intervals.

erated by humans. This has no bearing on faithfulness, as an explanation method could generate something that matches a human's explanation but fails to reflect the underlying model behavior. Despite that, it is useful to consider this metric (often called plausibility), as in practice people will be more likely to accept and trust plausible explanations. Ideal explanations would then be both faithful and plausible. By converting the rationales to binary vectors (1 for highlighted words and 0 otherwise), we can compare explanations to the rationales in the same way as the ground truth attributions in the SST experiment. Figure 4.5 shows results for both of the ground truth metrics using a modified version of those in section 4.1.1. The 'ground truth' in this case is just the annotated rationales, rather than known feature attributions. None of the methods produce particularly plausible explanations, with at most 40% overlap for Integrated Gradients (a small improvement considering the random baseline overlaps 20% of the time). The mean rank values are similarly poor, with most methods requiring between 60 and 80 percent of the complete ranking before recovering the complete rationales.

4.1.4 Layer Randomizations

Figures 4.6 and 4.7 shows the results of the layer randomization experiments across all four sizes of T5 model. The results shown are for the augmented SST models. The analogous figures for models trained on MultiNLI and e-SNLI are in the appendix, but show the same trends as the figures here.

The gradient-based methods are all generally sensitive to randomizations in the parameters at any layer in the models and at any size. For both independent and cascading randomizations, the rank correlation between the original explanation and the partially randomized ones drops nearly to zero at all layers. The two attention methods are less sensitive to parameter changes. Average Attention shows gradually decreasing correlations from the last layer to the first across all cases. Attention Rollout shows a similar declining trend for the smaller models, but changes in larger models, with perfect correlation in deeper layers and then a rapid decrease to zero correlation in the final few layers.

The rapid decrease in correlation for gradient methods is understandable, as the gradient value, by definition, is tied to value of the parameters. For example, randomizing any layer could cause the sign of the gradient to flip, which in turn would cause a large change in correlation. The gradual decrease of Average Attention can also be explained. Because the Average Attention explanation is just an average of all the attention values generated by the model for an input, if the weights are randomized at the last layer, only the last attention values are affected. This means that only one term out of the total average is changed. Randomizing an earlier layer will also affect the attention values of all the following layers, and so the overall average will shift by a larger amount. This sequential relationship explains the mostly monotonic decrease in correlation for Average Attention. The behavior for Attention Rollout is more complex. For smaller models, such as T5 Tiny or T5 Mini, the change in correlation roughly matches that of Average Attention, and can be explained via similar reasoning. For larger models, though, Attention Rollout tends to be invariant to changes in the parameters of later layers. The multiplication of attention matrices that determines the Attention Rollout explanation seems to reach a fixed point after the first few layers, after which the additional multiplications reproduce the same values.

The gradient-based methods here are all clearly dependent on the model parameters, although this does not hold for all explanation methods that use gradients [2]. Average Attention shows a clear linear dependence on the parameters, and Attention Rollout is invariant to parameter changes in deeper layers of larger transformers. Both of these methods may be useful for explaining smaller transformer models, where there would be a stronger relationship between the model parameters and the produced attention values. However, for deeper transformer models, the explanations would be untrustworthy, as the later layers could be very relevant to determining the overall output, but would have little to no impact on the resulting attention-based explanations.

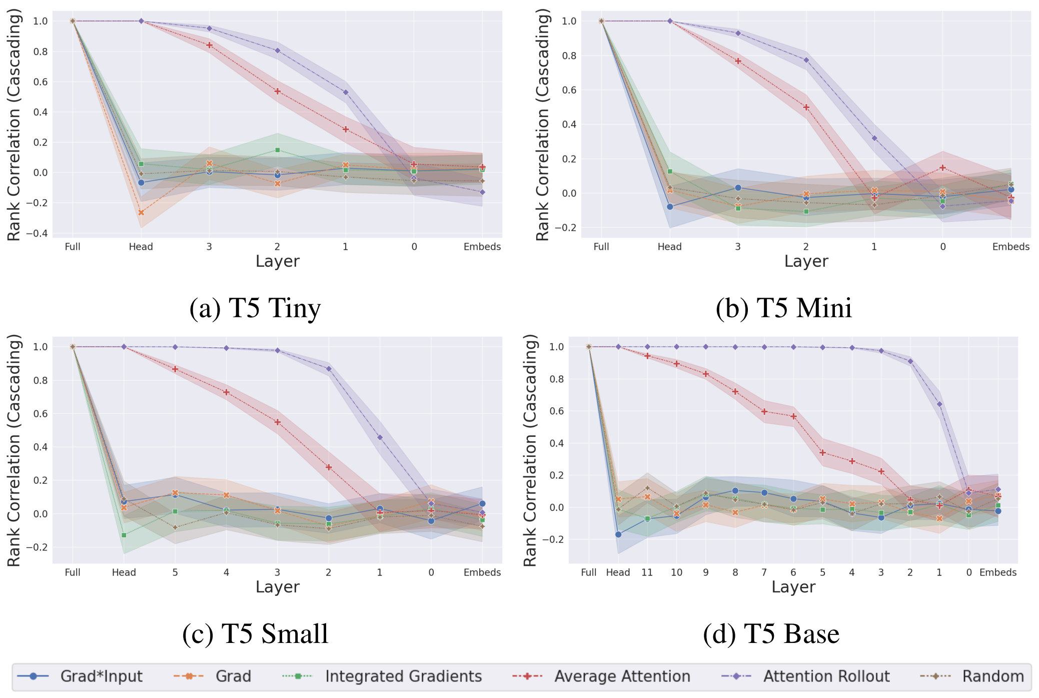


Figure 4.6: Spearman Correlations between explanations generated via cascading layer randomization (Section 3.3). Layer randomization was applied to four T5 models of differing sizes finetuned on the augmented SST dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model has been randomized up to the corresponding layer on the x axis. Randomization starts from the last layer and goes backward (from left to right in the graph). Faithful explanations should have decreasing correlations (or decrease all the way to zero) as layers of the model are randomized. High correlations indicate a lack of sensitivity to model parameters (i.e. unfaithful explanations).

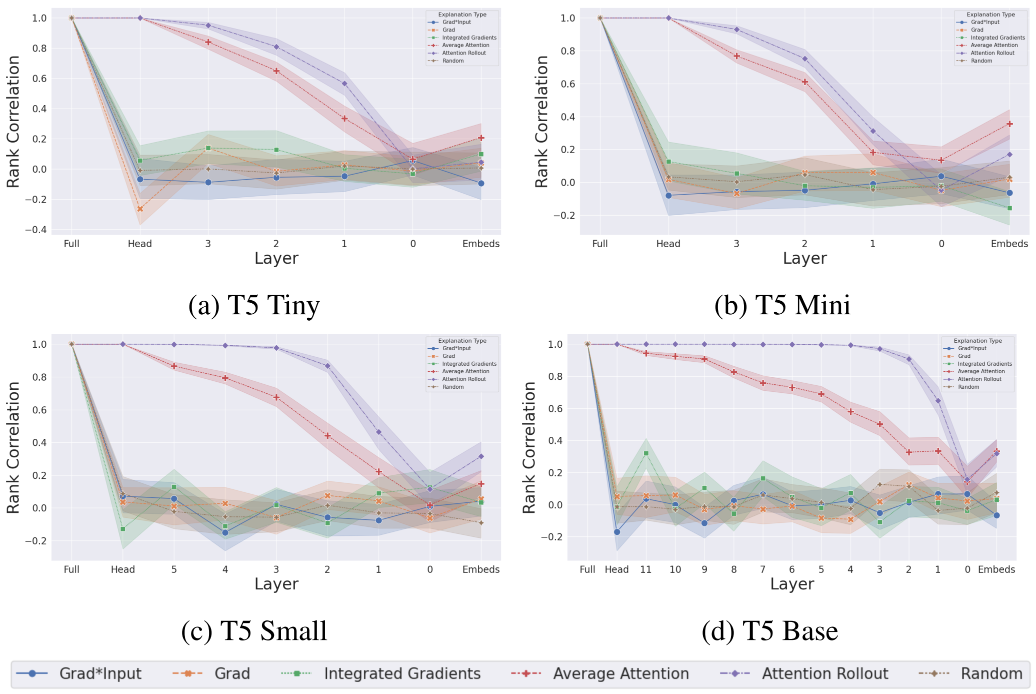
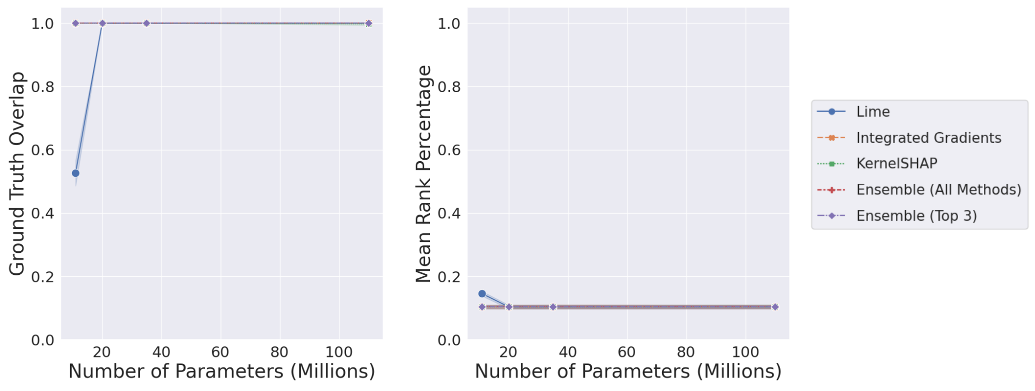


Figure 4.7: Spearman Correlations between explanations generated via independent layer randomization (Section 3.3). Independent layer randomization was applied to four different T5 models of increasing size finetuned on the augmented SST dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model parameters at the corresponding layer on the x axis have been randomized. Faithful explanations should have low correlations for each layer of the model except "Full", the original explanation on the far left.

4.1.5 Ensembles

Figure 4.8 shows the ground truth overlap and mean rank results for the two different ensemble methods applied to the models trained on the augmented SST dataset. LIME, SHAP and Integrated Gradients are included for comparison. Both ensembles perform as well as the best individual methods, so there is no drop in performance when averaging the explanations together. Figure 4.9 shows the Comprehensiveness and Sufficiency results. Ensembling does not appear to improve either metric, and both ensembles perform slightly worse than the best



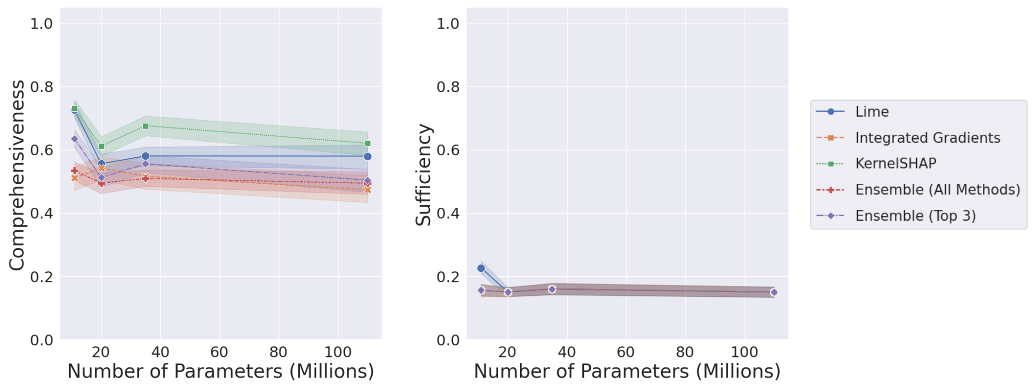
(a) Ground Truth Overlap (b) Mean Rank Percentage

Figure 4.8: Ground truth metrics for ensemble explanations of four T5 models of differing size (finetuned on the augmented SST dataset). LIME, SHAP and Integrated Gradients are included for comparison. The x axis is the number of parameters in each T5 model. The y axis is the Ground Truth Overlap or Mean Rank value. Faithful explanations should have high Ground Truth Overlap and low Mean Rank values. The values are averaged over 500 examples and shown with 95% confidence intervals. Note that the lines for all methods overlap almost completely at 1.0 in 4.8a and 0.1 in 4.8b.

individual methods. Analogous Comprehensiveness and Sufficiency figures for the MultiNLI and e-SNLI models are given in the appendix, and show similar trends. This roughly equal performance of the ensembles is expected as we are averaging a group of equally performing methods, but it does suggest that the methods are generating correct and incorrect explanations for similar examples. If different methods were generating significantly different explanations for the same example, we would expect to see more variation in the performance of the ensembles.

4.1.6 Adversarial Examples

Figures 4.10 and 4.11 show the results for the adversarial example experiments across all four T5 models of different sizes. The Rank Change and Top-k Sum



(a) Comprehensiveness (b) Sufficiency

Figure 4.9: Comprehensiveness and Sufficiency for ensemble explanations of four T5 models of different sizes (finetuned on the augmented SST dataset). LIME, SHAP, and Integrated Gradients are included for comparison. The x axis is the number of parameters in each model. The y axis is either the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness and low Sufficiency values. The values are averaged over 500 examples and shown with 95% confidence intervals. Note that in (b), all methods mostly overlap along $y = 0.1$.

Change metrics measures the impact of the adversarial examples on explanations, while Probability Change measures how much different the adversarial example's output probability is from the original input (see Section 2.3). A robust explanation method should have low values for Rank Change and Top-k Sum Change, and an ideal adversarial example should have high values for Rank Change and Top-k Sum Change as well as a low value for Probability Change. While the smaller sample size and large confidence intervals makes strong conclusions difficult, Integrated Gradients appears to be slightly more robust than Gradients and Gradients*Input, which both have higher Rank Change values. However, Integrated Gradients has slightly higher values for Top-k Sum Change, indicating that Gradients and Gradients*Input may be susceptible to large rank changes via small alterations of the attribution values. In all cases, a substantial portion of the ranking can be changed (approximately 40%) even if the Top-k Sum Change values

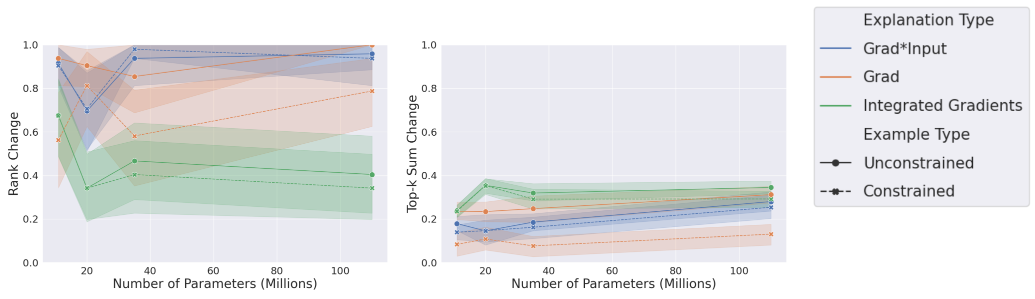


Figure 4.10: Rank Change and Top-k Sum Change values for adversarial examples generated for explanations of four T5 models of different sizes (finetuned on the augmented SST dataset). The x axis is the number of parameters in each model. The y axis is the Rank Change or Top-k Sum Change value. Adversarially robust methods should have low Rank Change and Top-k Sum Change values. The solid lines are values for examples generated using the unconstrained objective, and the dotted lines are for examples generated using the constrained objective. The values are averaged over 16 examples and shown with 95% confidence intervals.

appear low.

A separate point of interest is the relatively similar performance of the Constrained and Unconstrained objectives. The two objectives perform similarly for Gradients*Input and Integrated Gradients, despite the more difficult objective of the Constrained case. Gradients shows a larger gap between the two objectives, as the Unconstrained objective appears to find adversarial examples with higher Rank Change values and Top-k Sum Change values than the Constrained one. The similarity of these results across the two objectives suggests that there may be adversarial examples that are more similar to the original inputs while still changing the explanation as much as more dissimilar examples. These examples may be missed by the objective that only constrains the predictions to be equivalent, but are found when trying to maintain equal probabilities instead.

Figure 4.11 provides some evidence for this. The Unconstrained objective results

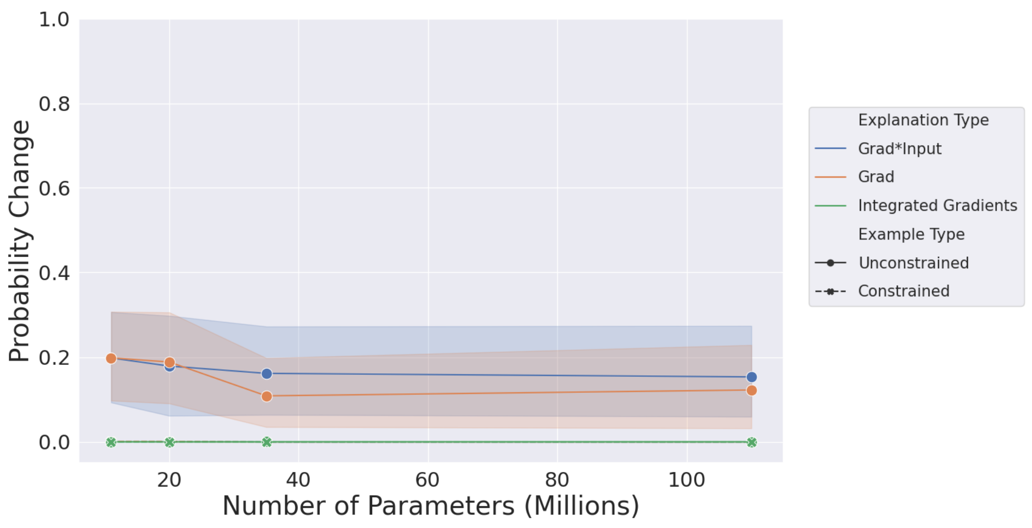


Figure 4.11: Probability Change values for adversarial examples generated for explanations of four T5 models of different sizes (finetuned on the augmented SST dataset). The x axis is the number of parameters in each model. The y axis is the Probability Change value. Lower values are preferable. The solid lines are values for examples generated using the unconstrained objective, and the dotted lines are for examples generated using the constrained objective. The values are averaged over 16 examples and shown with 95% confidence intervals. Note that all three constrained lines and the solid line for Integrated Gradients overlap at 0.0.

in a change in output probability for two of the three methods, Gradients*Input and Gradients. The Constrained objective, in contrast, finds adversarial examples that match the original output probability perfectly and so all methods have a value of 0 across all model scales. The augmented objective then appears to find adversarial examples that are nearly as effective as the unconstrained case, and that are significantly more similar in terms of output probability, a desirable quality when attacking explanations instead of predictions. Additional figures with analogous results for MultiNLI and e-SNLI are given in the appendix.

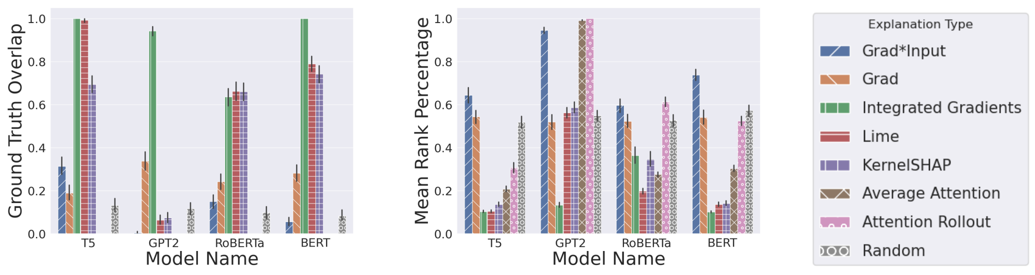
Scaling Experiments Summary

Increasing the scale of transformer models appears to have relatively little impact on the faithfulness of explanations. The same three methods (LIME, SHAP, and Integrated Gradients) were consistently the best performing across multiple scales and datasets, and the faithfulness of each method individually tended to vary by small amounts as the number of parameters increased. Gradients, Gradients*Input and both attention methods did vary more widely in Figure 4.1a, but the corresponding rank values in Figure 4.1b are still mostly flat, indicating that the shifts may be caused by small changes in the rankings that explanations produce (e.g. a feature moving from the second highest position to the highest). Overall, most methods were consistent in their behavior across all model scales. Additionally, the adversarial results showed that Integrated Gradients may be slightly more robust to adversarial perturbations than Gradients and Gradients*Inputs, and also provided some evidence that the augmented adversarial objective we propose finds effective adversarial examples while matching the original output probability more closely than the Unconstrained objective.

4.2 Architectures

The results in this section show the breakdown of the different experiments across different architectures. The figures in this section compare the four different pretrained models described in Section 3.6.2. The black lines on each bar represent a 95% confidence interval [4], and values for each method are averaged over 500 examples. The layer randomization experiments are performed in the same way as Section 4.1, as we have one figure per model in both sets of experiments, and the correlation values are averaged over 50 examples. The adversarial experiments are also averaged over 16 examples, as in the scaling experiments.

4.2.1 SST

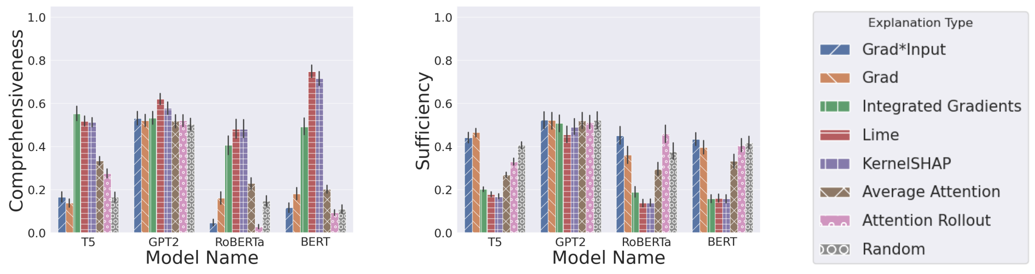


(a) Ground Truth Overlap (b) Mean Rank Percentage

Figure 4.12: Ground Truth Overlap and Mean Rank for explanations of four different pretrained models trained on the augmented SST dataset. The x axis is the name of each of the pretrained models. The y axis is either the Ground Truth Overlap or Mean Rank value. Faithful explanations should have high Ground Truth Overlap values and low Mean Rank values. The values are averaged over 500 examples, and the black bars indicate 95% confidence intervals.

In this experiment, we train models on an augmented version of the SST dataset, where an added special token is the only feature that determines the label. The cor-

[4] Confidence intervals are computed using the bootstrapping procedure in the Seaborn plotting library (https://seaborn.pydata.org/).



(a) Comprehensiveness (b) Sufficiency

Figure 4.13: Comprehensiveness and Sufficiency for explanations of four different pretrained models trained on the augmented SST dataset. The x axis is the name of each pretrained model. The y axis is either the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness values and low Sufficiency values. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.

rect feature attribution is then known, since the special token is the only predictive feature. Figure 4.12 shows the results for the two ground truth measures across all four models, comparing the known feature attributions with the generated explanations. Each bar in the bar graph represents a different explanation method. In general, there is a great deal of variance across all models for both measures. Integrated Gradients tends to be the best performing, working particularly well for T5, GPT2 and BERT. However, it is significantly less effective for RoBERTa. LIME and KernelSHAP, the other two highest performing methods in the scaling experiments, vary widely in terms of performance. LIME works extremely well for T5, but is middling for RoBERTa and BERT and poor for GPT2. KernelSHAP is similar, although its performance on T5 more closely matches its performance on RoBERTa and BERT. Gradients tends to perform approximately as well as the random baseline, exceeding it for T5 and RoBERTa Base, but underperforming it for GPT2 and BERT. Gradients*Input always outpeforms the random baseline, although not by a significant margin. Lastly, the two attention methods fail to assign the highest attribution to the added token across all models, but diverge in terms of their Mean Rank values. Average Attention achieves a relatively low Mean Rank value compared to the top three performing methods (excepting GPT2), while Attention Rollout only outperforms the random baseline's Mean Rank for T5.

One possible explanation for the stark difference in performance for LIME and KernelSHAP on GPT2 is that GPT2 is an autoregressive language model, while the other three models are bidirectional. A crucial component of an autoregressive language model is that the attention matrix is masked so that a token can only have non-zero attention values for tokens to its left. Interventions made by LIME or KernelSHAP that alter earlier tokens in the input may then have an outsized impact on the output, as those earlier tokens affect the attention values of all the tokens that come after them.

To similarly understand the behavior of the attention explanations, we need to separately consider their application to bidirectional and unidirectional models. For the bidirectional models (T5, RoBERTa, BERT), the low Ground Truth Overlap and high Mean Rank values of the attention explanations, as compared to the earlier scaling experiment results (Section 4.1.1) where the Mean Rank values matched the top three methods, is likely a combination of two factors. The first is one previously discussed in the scaling section, where the spurious token is frequently near the top of the ranking, despite never being the top ranked feature (the relatively low mean rank for T5 Base provides some evidence for this). The second problem is that the greater depth of these models means attention becomes an increasingly poor proxy for input feature importance. At each layer, the input features become more entangled via self-attention, and the resulting contextualized embeddings are less representative of the original input token in that position. As all of these models are relatively deep, the attention values assigned to embeddings in deeper layers may not reflect the importance of the feature originally at that position. This is in addition to other previously noted concerns regarding the use of attention values as forms of explanation [30].

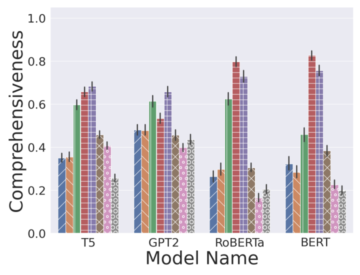
For unidirectional models, like GPT2, all these issues hold, but with one additional concern. The left-to-right masking proposed as an explanation for LIME and SHAP's poor performance also affects the attention methods, by biasing the attention values towards earlier tokens. Because of this bias, the earlier tokens in the sequence will, on average, be given greater attributions. The Mean Rank value of 1.0 for both attention methods, when applied to GPT2, supports this hypothesis, as the predictive token is added to the end of the sentence. If the explanation ranking is determined by the order of the tokens, then the entire ranking would be needed to recover the predictive/last token, i.e. the Mean Rank of the explanation would be 1.0.

We now consider the metrics that do not require ground truth attributions. Figure 4.13 shows the Sufficiency and Comprehensiveness results. For T5, RoBERTa and BERT, the trends match those in the scaling experiments (Section 4.1.1. Integrated Gradients, LIME and KernelSHAP have the highest Comprehensiveness values and lowest Sufficiency values. Gradients and Gradients*Input perform the worst, with values frequently around the random baseline performance. The two attention methods fall in between, occasionally underperforming the random baseline, as in Attention Rollout for RoBERTa, but also outperforming Gradients and Gradients*Input in some cases, such as Average Attention and Attention Rollout for T5 Base. The Comprehensiveness and Sufficiency results for GPT2 are quite distinct from the other three methods. All the methods, including the random baseline, perform roughly equally, with little of the variation seen across the other models.

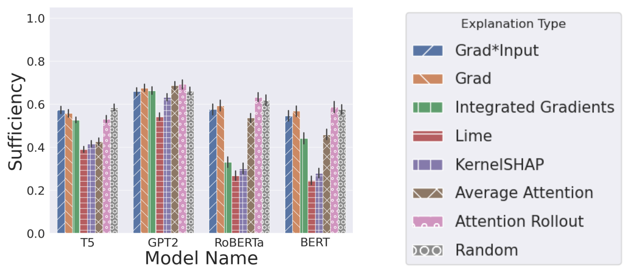
Understanding why this different behavior arises between GPT2 and the other models requires considering how these models were originally trained. When perturbing inputs to measure Comprehensiveness and Sufficiency, certain tokens in the input are replaced with special padding or masking tokens (depending on which special tokens are defined for the model). The bidirectional models were pretrained with a masked language modeling objective, where random tokens are masked out and then predicted by the model, a process similar to the input perturbations used for Comprehensiveness and Sufficiency. GPT2, in contrast, was trained with an autoregressive language modeling objective, where it only had to predict the next word that followed after some input text. Because of this difference in pretraining objective, inputs with masked features may be significantly more out-of-distribution (OOD) for GPT2 than they are for the other three models, and the resulting prediction on perturbed inputs may be less reflective of predictions made on in-distribution data. Large changes in prediction could then be the result of the OOD nature of the input rather than the relevance of any actual masked features, which would explain how even a random explanation, where the features likely have no particular relevance, can still induce a large change in the prediction (i.e. can still achieve relatively high Comprehensiveness).

This would also explain why GPT2 has roughly similar values for Comprehensiveness and Sufficiency. Compared to the other models, which have fairly significant gaps between Comprehensiveness and Sufficiency (particularly for methods like LIME that perform well), the values for GPT2 are fairly close together for all methods. If the change in prediction is due more to the existence of masked tokens than to the actual tokens that were masked, we would expect Comprehensiveness and Sufficiency to produce approximately similar changes in prediction, since they only differ in how they use the attributions to determine what features should be masked.

4.2.2 MultiNLI



(a) Comprehensiveness



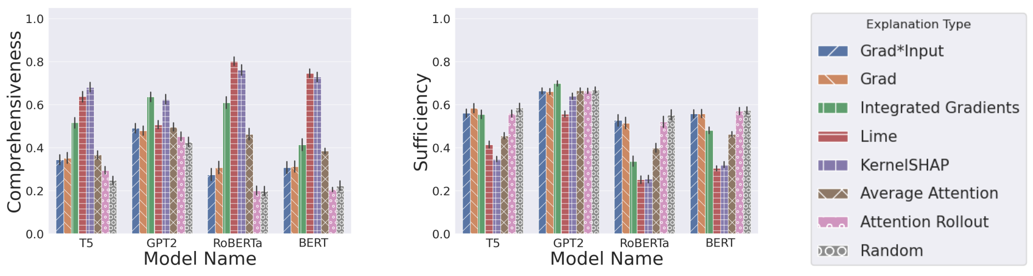
(b) Sufficiency

Figure 4.14: Comprehensiveness and Sufficiency for explanations of four different pretrained models finetuned on MultiNLI. The x axis is the name of each pretrained model. The y axis is either the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness values and low Sufficiency values. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.

In the MultiNLI experiment, we train models on the MultiNLI dataset [62], and then use Sufficiency and Comprehensiveness (see Section 2.2.3.2) to measure explanation faithfulness. Figure 4.14 shows the Comprehensiveness and Sufficiency results for the models trained on MultiNLI. This supports the previous results, in that Integrated Gradients, LIME and KernelSHAP are consistently the best performing, with the highest Comprehensiveness values and the lowest Sufficiency values. We see similarly wide variation between models, such as the 20% difference in Comprehensiveness between RoBERTa and BERT, despite the fact they are almost identical architectures.

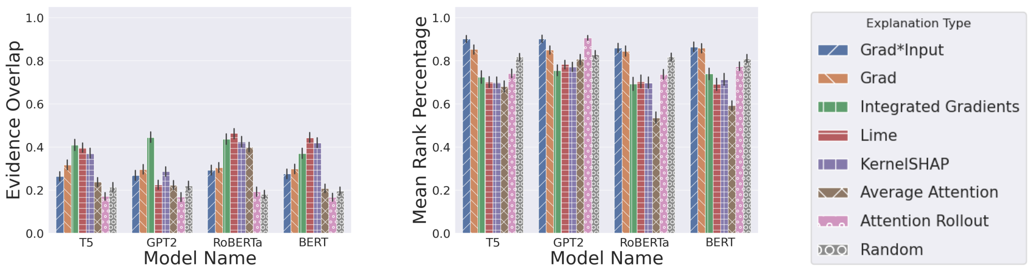
As in Section 4.2.1, GPT2 shows significantly less variation across methods, and a high performing random baseline. This is likely for the same reasons described there, although there is more variation between Integrated Gradients and KernelSHAP and the other methods, suggesting that maybe the impact of the OOD perturbed inputs is lessened for more complex tasks (i.e. the differences between methods are more severe, and so persist despite the OOD problem).

4.2.3 e-SNLI



(a) Comprehensiveness (b) Sufficiency

Figure 4.15: Comprehensiveness and Sufficiency for explanations of four different pretrained models finetuned on e-SNLI. The x axis is the name of each of the pretrained models. The y axis is either the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness values and low Sufficiency values. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.



(a) Ground Truth Overlap (b) Mean Rank Percentage

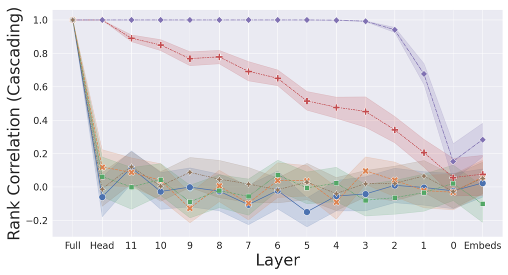
Figure 4.16: Ground Truth Overlap and Mean Rank between human rationale annotations and explanations of four different pretrained models finetuned on e-SNLI. The x axis is the name of each of the pretrained models. The y axis is the Ground Truth Overlap or Mean Rank value, using human rationale annotations as the ground truth for comparison. Plausible explanations should have high Ground Truth Overlap and low Mean Rank values. The values are averaged over 500 examples and shown black bars indicating 95% confidence intervals.

In the e-SNLI experiment, we train each model on the e-SNLI dataset, which is a natural language inference dataset with additional annotations explaining the labels, and use Comprehensiveness and Sufficiency to measure explanation faithfulness. Additionally, we use the annotations in e-SNLI to evaluate explanation plausibility, the similarity of automated explanations to those given by humans. Figure 4.15 shows the Sufficiency and Comprehensiveness results for models trained on e-SNLI. Overall, the results are very similar to those in Section 4.2.2. Integrated Gradients, LIME and KernelSHAP are the best performing on the bidirectional models, and GPT2 shows small variations across methods. LIME and Integrated Gradients do slightly outperform other methods for GPT2, similar to the results in the MultiNLI case.

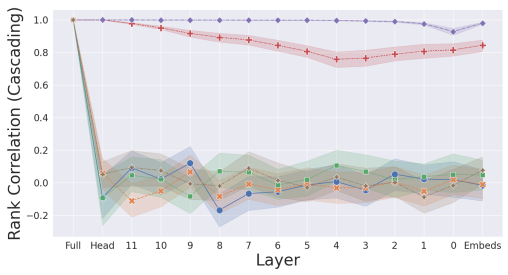
The plausibility results for the e-SNLI models are shown in Figure 4.16. As in the scaling case, all methods generate fairly implausible explanations. LIME, KernelSHAP and Integrated Gradients do slightly better than the others for the three bidirectional models, but no method exceeds an overlap in the low 40% range. For GPT2, Integrated Gradients generates slightly more plausible explanations than other methods, although the margin is still quite small. Overall, the explanations produced by these methods, despite varying in faithfulness, produce generally implausible explanations. Even those explanations generated by the highest performing methods when measured by faithfulness metrics only match human explanations a small fraction of the time (approximately 40% in the best cases).

4.2.4 Layer Randomizations

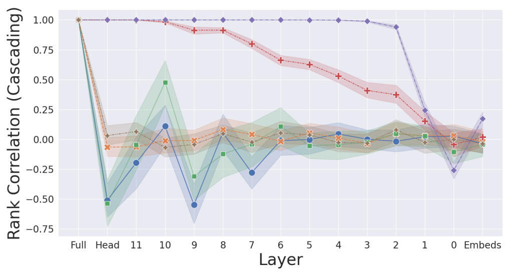
In the layer randomization experiment, we randomize the parameters of each model and measure the similarity (via rank correlation) between explanations generated for the randomized model and for the original. A high degree of similarity indicates that the explanation is independent of the model parameters, and therefore unfaithful to model behavior. Figures 4.17 and 4.18 show the layer random-



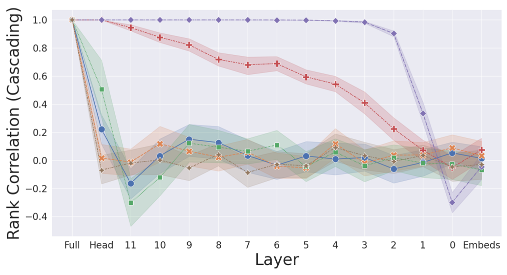
(a) T5 Base



(b) GPT2 Small



(c) RoBERTa Base



(d) BERT Base

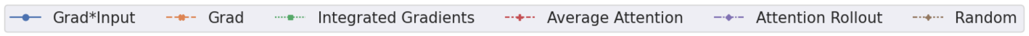
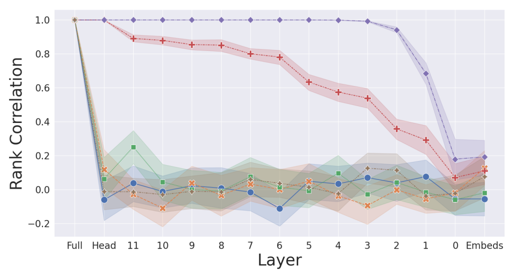
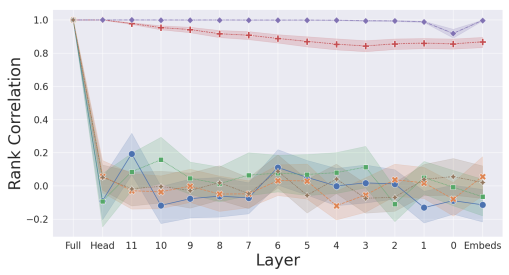


Figure 4.17: Spearman Correlations between explanations generated via cascading layer randomization (Section 3.3). Layer randomization was applied to four different pretrained models finetuned on the augmented SST dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model has been randomized up to the corresponding layer on the x axis. Randomization starts from the last layer and goes backward (from left to right in the graph)

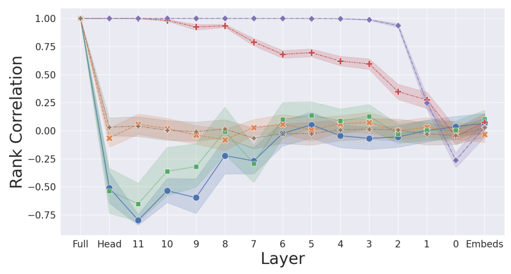
ization results for all four pretrained models. For all four models, every method except those based on attention seems sensitive to parameter randomization, with correlations between the original and randomized explanations dropping to zero (or in some cases to a negative correlation, e.g. 4.18c) after just one or two layers of randomization. For the bidirectional models, the attention methods for these models show the same relationships seen in the scaling experiments, which are discussed in 4.1.4. For GPT2, however, both attention methods seem entirely invariant to the parameters. Because the attention computation is masked in GPT2, the resulting explanation values are skewed towards earlier inputs. Due to this,



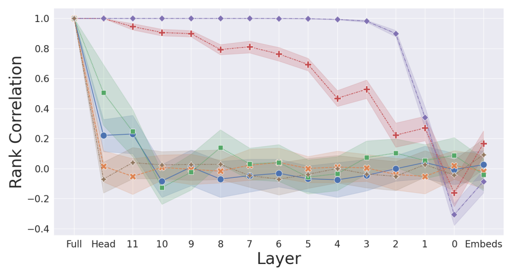
(a) T5 Base



(b) GPT2 Small



(c) RoBERTa Base



(d) BERT Base

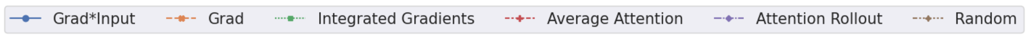
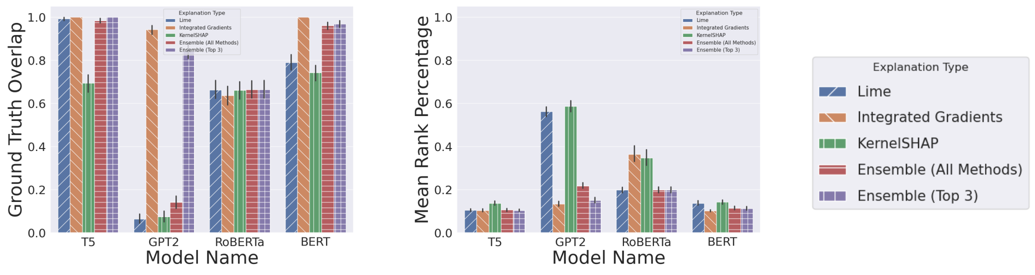


Figure 4.18: Spearman Correlations between explanations generated via independent layer randomization (Section 3.3). Independent layer randomization was applied to four different pretrained models finetuned on the augmented SST dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model parameters at the corresponding layer on the x axis have been randomized.

the explanation ranking likely matches the ordering of the input tokens a significant portion of the time, and the ordering of the input tokens is independent of the parameters. In that case, it would make sense that randomizing the model parameters had little effect on the explanation rankings.

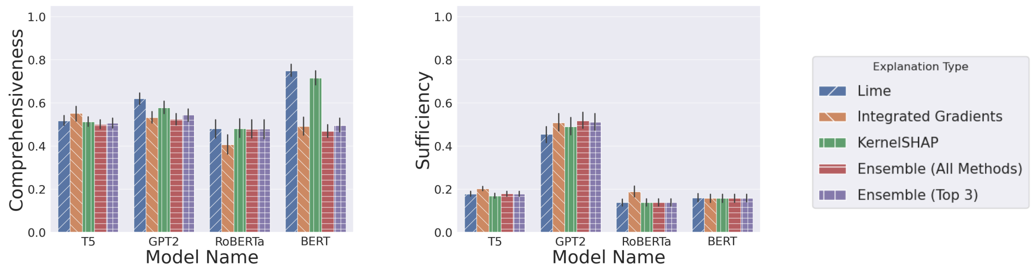
4.2.5 Ensembles

In this section we evaluate two different ensembles of explanations applied to four pretrained models. One ensemble averages the attributions of all seven explanation methods considered here together, while the other averages the attributions



(a) Ground Truth Overlap (b) Mean Rank Percentage

Figure 4.19: Ground truth metrics for ensemble explanations of four different pretrained models (finetuned on the augmented SST dataset). LIME, SHAP and Integrated Gradients are included for comparison. The x axis is the number name of each of the pretrained models. The y axis is the Ground Truth Overlap or Mean Rank value. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.



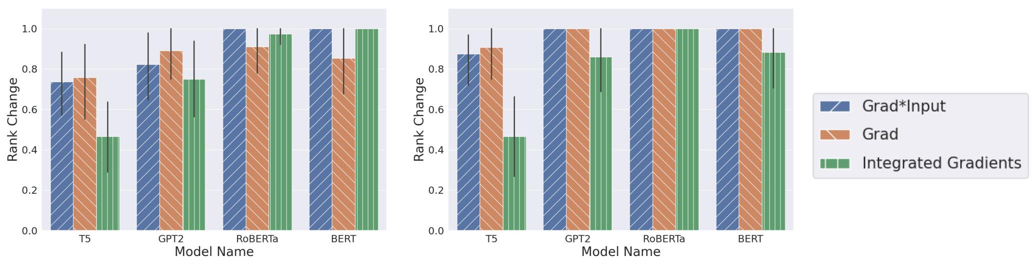
(a) Comprehensiveness (b) Sufficiency

Figure 4.20: Comprehensiveness and Sufficiency for ensemble explanations of four different pretrained models (finetuned on the augmented SST dataset). LIME, SHAP, and Integrated Gradients are included for comparison. The x axis is the name of each of the pretrained models. The y axis is either the Comprehensiveness or Sufficiency value. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.

of the three best performing methods, LIME, SHAP and Integrated Gradients. Figure 4.19 shows the ground truth results for the two ensembles across the four different pretrained models. Ensembling the top three methods together works well, approximately matching the best performing method in terms of Ground Truth Overlap across all models. The three-method ensemble also matches the best method in terms of Mean Rank across all four models. The full ensemble is equally effective for all models except GPT2, where it has a significantly lower value for Ground Truth Overlap. The less faithful explanation methods may produce explanations with more consistent structure for GPT2 (e.g. the bias towards earlier inputs in GPT2 attention explanations), which may skew the ensemble more systematically. However, the complete ensemble still has a Mean Rank value for GPT2 that is close to the best performing methods, suggesting that the ensemble still improves the explanation, even if it does not give the added token the greatest attribution.

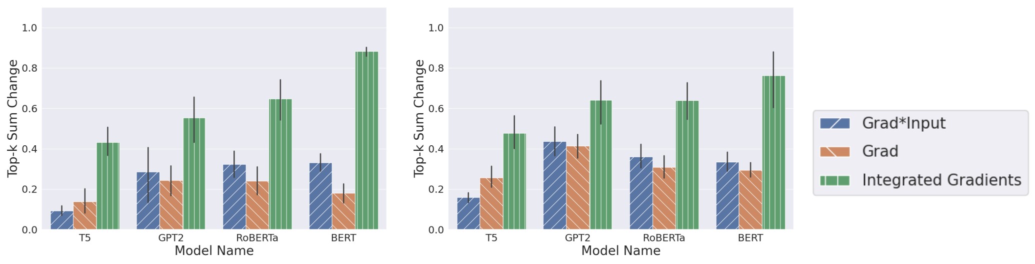
Figure 4.20 shows the Comprehensiveness and Sufficiency results for the two ensembles. The ensembling provides little benefit for these two measures, performing on par with the best methods or worse (in the case of Comprehensiveness for BERT base). The analogous results for MultiNLI and e-SNLI are given in the appendix, and show similar trends to the SST results shown here.

4.2.6 Adversarial Examples



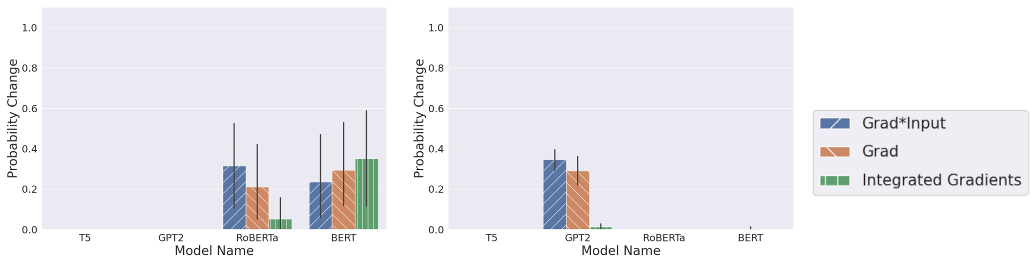
(a) Constrained Rank Change (b) Unconstrained Rank Change

Figure 4.21: Rank Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the augmented SST dataset. The x axis is the name of each of the pretrained models. The y axis is the Rank Change value. Robust explanations should have low values for Rank Change. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.



(a) Constrained Top-k Sum Change (b) Unconstrained Top-k Sum Change

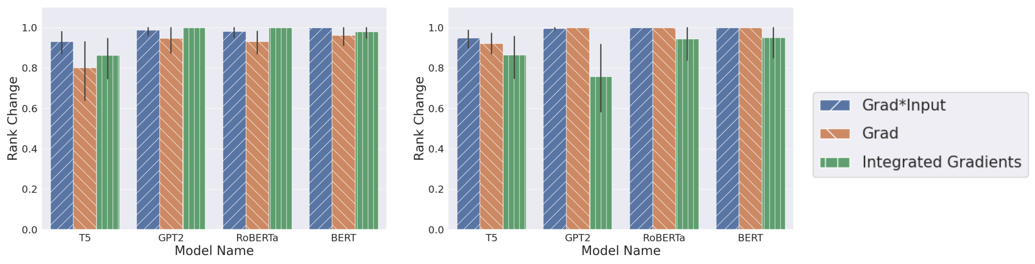
Figure 4.22: Top-k Sum Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the augmented SST dataset. The x axis is the name of each of the pretrained models. The y axis is the Top-k Sum Change value. Robust explanations should have low values for Top-k Sum Change. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.



(a) Constrained Probability Change (b) Unconstrained Probability Change

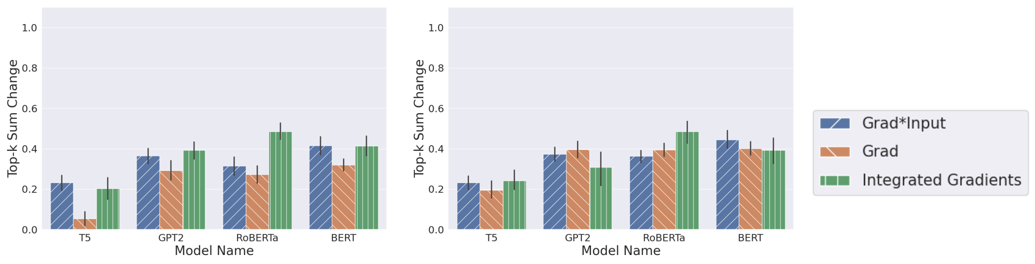
Figure 4.23: Probability Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the augmented SST dataset. The x axis is the name of each of the pretrained models. The y axis is the Probability Change value. Lower values are preferable. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.

Figures 4.21, 4.22 and 4.23 show the results of the adversarial example experiments across all four pretrained models finetuned on the augmented SST dataset. The values for the Constrained and Unconstrained objectives are presented side by side for comparison. The figures showing Rank Change and Top-k Sum Change for each of the models broadly agree with the trends in the scaling adversarial re-



(a) Constrained Rank Change (b) Unconstrained Rank Change

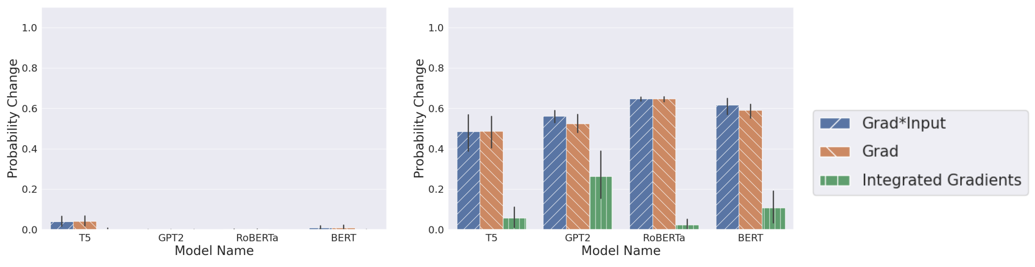
Figure 4.24: Rank Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the MultiNLI dataset. The x axis is the name of each of the pretrained models. The y axis is the Rank Change value. Robust explanations should have low values for Rank Change. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.



(a) Constrained Top-k Sum Change (b) Unconstrained Top-k Sum Change

Figure 4.25: Top-k Sum Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the MultiNLI dataset. The x axis is the name of each of the pretrained models. The y axis is the Top-k Sum Change value. Robust explanations should have low values for Top-k Sum Change. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.

sults (see Section 4.1.6). Integrated Gradients has slightly lower Rank Values than Gradients and Gradients*Inputs for two of the four models, although the margins are small, and the confidence intervals overlap for several models. Also similar to the scaling results, Integrated Gradients has slightly higher Top-k Sum Change values compared to Gradients and Gradients*Input. This may be due to the possi-



(a) Constrained Probability Change (b) Unconstrained Probability Change

Figure 4.26: Probability Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the MultiNLI dataset. The x axis is the name of each of the pretrained models. The y axis is the Probability Change value. Lower values are preferable. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.

bility previously mentioned that Gradients and Gradients*Input could have their rankings changed more significantly by small attribution alterations, leading to small Top-k Sum Change values but larger Rank Change values. Overall, all four models and three explanation methods seem vulnerable to adversarial examples. The lowest Rank Change value is over 40%, with many values from 80-100%, indicating that almost all of the top k ranked features are shifted to lower rankings.

We can additionally compare the two objective functions used to generate adversarial examples for these models, and see if the new augmented objective, which attempts to match the output probability of the original and adversarial examples, provides any benefit. In this instance, based on Figure 4.23, the original objective outperforms the augmented one. The Constrained objective finds matching adversarial examples for T5 and GPT2, resulting in values of 0 in the figure, but changes the output probability for RoBERTa and BERT. The original Unconstrained objective changes the output probability for GPT2, but not for the other three models, which all have values of zero. However, the low change in probability for the unconstrained objective may be due to the simplicity of the augmented SST task.

Figures 4.24, 4.25 and 4.26 show the analogous results for models finetuned on the MultiNLI dataset. The Rank Change values and Top-k Sum Change values shown in 4.24 and 4.25 show roughly equal performance, with the Unconstrained objective outperforming the Constrained objective by narrow margins. However, the probability changes in Figure 4.26 are much greater for the Unconstrained objective than the Constrained one across all models. This suggests that, for more complicated tasks like Natural Language Inference, it may be important to optimize directly for matching probabilities, rather than only constraining predictions. The analogous results for models finetuned on e-SNLI are given in the appendix, and show similar trends to those finetuned on MultiNLI.

Architecture Experiments Summary

As in the scaling experiments, the same methods (LIME, SHAP, and Integrated Gradients) were the highest performing across all models and datasets. The precise top performer was dependent on the context (e.g. Integrated Gradients significantly outperforms the other methods in identifying the most predictive feature for the SST-GPT2 combination), but one of those three methods was always the best. Across models, though, there is significant variation in the quality of explanations. While the relative rankings of methods were stable, different models produced explanations of widely varying faithfulness. GPT2 as compared to the other bidirectional models is one clear example, but even between related models like BERT and RoBERTa there is significant variation, such as the difference in performance of Integrated Gradients in Figure 4.12a. Additionally, the adversarial example results showed the susceptibility of these pretrained models and explanation methods to adversarial perturbations of their inputs. The comparison of two objective functions for generating adversarial examples further showed that our novel augmented objective can generate adversarial perturbations roughly on par with the original objective, and with significantly differences in output probability between the original and adversarial datapoints.

Results Summary

Of the seven methods evaluated, Integrated Gradients, KernelSHAP and LIME were consistently the best performing. Which method was the best choice varied depending on the specific setup and model, but there was a consistent gap between those three and the other four methods (with the exception of LIME and KernelSHAP's poor performance on the GPT2-SST combination). The two attention methods, Average Attention and Attention Rollout, slightly outperformed the standard gradient methods, Gradients and Gradients*Input, in several cases, but failed completely in other cases (e.g. when applied to GPT2) showing that certain architectures can have major impacts on the validity of attention as an explanation. Finally, Gradients and Gradients*Input performed poorly across all combinations, only slightly exceeding a baseline of randomly generated explanations. All three gradient-based methods and all models were susceptible to adversarial perturbations in varying degrees, with Integrated Gradients showing slightly more robustness than Gradients and Gradients*Input. Additionally, the adversarial experiments provide evidence that our novel adversarial objective produces adversarial examples as effective as the original objective [21] but with output probabilities significantly closer to the original datapoints.

5 | Discussion

5.1 Summary

Large language models are increasingly popular in both real-world applications [23, 61] and as a topic of research [41, 10, 33, 55]. However, we lack the ability to explain, at a human-comprehensible level, how these models determine their outputs given input text, a disconnect which can allow a number of other concerns in areas like machine learning fairness [40] and safety [20] to go unnoticed. Machine learning explainability is a subfield focused on solving this problem, and a variety of different methods to generate explanations have been proposed in prior literature [49, 53, 45, 36].

In this work we focus in particular on post-hoc feature importance explanations. These methods are applied to a trained machine learning model and a single input to that model, and assign an importance value to every feature in the input based on that feature's relevance to the model's output. Evaluating the quality and effectiveness of these explanation methods is itself a difficult problem, and many different frameworks have been proposed to benchmark them [65, 68, 2]. One important criterion to evaluate is explanation faithfulness, which is the extent to which an explanation accurately reflects the behavior of the model that it explains. The experiments in this work measure explanation faithfulness across a variety of different language model architectures, model sizes, and explanation methods. Previous work primarily used the base BERT model when proposing frameworks for measuring faithfulness, but a more thorough evaluation is necessary to understand how well those explanation methods generalize to other kinds of language models. Methods that work well for BERT will not necessarily be effective for other architectures, or at other model scales, and so this broader evaluation is needed to understand the contexts in which different explanation methods will be appropriate. We provide a systematic evaluation of several feature importance explanation methods across a variety of different language models, applying several different approaches to measuring faithfulness to build a broader and more robust view of the applicability and efficacy of feature importance explanations to large language models.

Overall, these experiments show a clear divide between three reasonably faithful explanation methods, LIME, SHAP and Integrated Gradients, and four less faithful explanation methods, Gradients, Gradients*Input, Average Attention and Attention Rollout. Across models of different scales and architectures, the same three methods were nearly always the best performing. However, these three are also the most computationally intensive to use. Integrated Gradients requires approximating a path integral, and so computes many gradients via back-propagation at different points between an input x and baseline value x' ($\mathcal{O}(nm)$ for n examples and m samples). LIME and SHAP both require generating a dataset of perturbed inputs and then training a linear model to generate an explanation ($\mathcal{O}(nm + nmk^2)$ for n examples of length k and m samples). Repeating this process for multiple examples is quite slow. In comparison, Gradients and Gradients*Input require only a backward pass for each example ($\mathcal{O}(n)$), and attention methods require no additional computation beyond what is needed for model inference ($\mathcal{O}(1)$).

Additionally, while the ranking of the methods was fairly consistent, there was still a great deal of individual variation in performance across different kinds of models. For example, different explanation methods had drastically different performances on bidirectional models (T5, BERT, RoBERTa) compared to unidirectional ones (GPT2). Differences in design choices such as pretraining objectives, vocabularies, tokenization schemes, attention mechanisms and more can have significant impacts on explanation quality, and it is unclear how to decide a priori which method will work best for a particular model. LIME and SHAP, for example, were frequently the best performing methods across various scales and kinds of architectures, but almost completely failed to prioritize the added positive/negative token for GPT2. Integrated Gradients, on the other hand, performed quite well across all models and then particularly well for the same GPT2 case.

In addition to the dataset and randomization focused evaluations, we also evaluated the adversarial robustness of several explanation methods, and proposed a novel adversarial objective based on one proposed in [21]. The novel objective is designed to encourage adversarial examples that are similar in probability, not just in prediction, as those adversarial examples are more likely to rely on the same features as the original datapoint. Our robustness experiments confirm prior evidence about the susceptibility of explanation methods to adversarial attacks and provide some evidence that our proposed objective can generate effective adversarial examples while closely matching the output probability of the original input datapoint.

Understanding exactly what aspects of model architecture and training determine these behaviors is an important step towards understanding what explanations to use in particular contexts. In terms of minimizing required compute while still generating faithful explanations, Integrated Gradients appears to work best for the models assessed here. Computing gradients for several samples to approximate an integral is still faster than generating a dataset of perturbations and training a linear model, and the resulting explanations were generally on par with those from LIME and SHAP. Clearly, though, there is enough variation, even within the related variants of transformers models, to justify including and assessing multiple methods within the context of a particular domain or application.

While this work covers a variety of different language models and explanation methods, there are still significant limitations to these results. For example, there are many more complex tasks in natural language processing that could require explanation, and that are difficult to evaluate using the methods outlined here, such as language generation or free-text question answering. There are also classes of language models that are not evaluated here and would require an analogous evaluation, such as recurrent models [26]. Lastly, there are many feature importance explanations not evaluated in this work [49, 5, 36, 45] and other forms of explanation [58, 31, 29] that will require new approaches to evaluating faithfulness and general explanation validity.

5.2 Further Work

The experiments presented here raise a lot of interesting questions, and there are several additional experiments that could help clarify and confirm different aspects of the results. One straightforward improvement would be to rerun the same experiments presented here, but averaged over multiple finetuning runs for each model. The results here are for one finetuned model for each case, which means that the resulting explanations may be affected by the particular solution found during finetuning, rather than aspects of the architecture or data that we are interested in. Averaging over results from multiple models would help eliminate this problem. Unfortunately, this approach is also quite computationally intensive, which is why we used only one model per case. There are a total of twenty-four models in these experiments, so n runs would require $24 \times n$ total models, a prohibitively large number when considering the time and compute required to finetune the larger language models used here.

Further, isolating the impact of different components of language models on explanations would be useful as well. Training multiple identical transformers from scratch (eliminating the pretraining objective) but changing one component at a time, e.g. using different tokenization schemes or vocabularies for each model, would help pinpoint the effect of those kinds of design decisions on explanations. Considering the large disparities in explanation quality between the different model architectures evaluated here, those decisions may have a large impact on how faithful the resulting explanations are. The pretraining objectives themselves also may have an impact on explanations. By comparing different pretraining objectives for the same model (e.g. using masked vs. left-to-right language modeling for a newly initialized BERT model), and then finetuning for specific tasks, you could evaluate how the pretraining objective affects explanations. This would be particularly useful for evaluating, for example, how the presence of a masking objective in pretraining impacts methods and metrics like LIME and SHAP or Sufficiency and Comprehensiveness, which rely on masking out input features.

5.3 Conclusion

Feature importance explanations are broadly applicable to many different kinds of machine learning models. In this work, we have specifically assessed their utility for transformer language models, and particularly how the validity of different methods changes across transformers of varying sizes and architectures. Increasing the size of a model tends to have relatively little impact on the quality of the resulting explanations, and methods that work well for small models appear to work well for large models as well. Across different architectures of a similar size, though, performance varies widely. Explanations for bidirectional and unidirectional models were drastically different for both input perturbation methods and attention methods, and the simplest gradient methods, Gradients and Gradients*Input were ineffective across nearly all models. Integrated Gradients was the most consistent of the methods, but still had significant variations in performance across models and datasets. Considering how unpredictable the validity of different methods seems to be across different language models, applications using these methods to justify or interpret LLMs need to extensively validate and compare different explanation methods to ensure that the final method used produces explanations that truly reflect the behavior of the underlying model.

Bibliography

[1] Samira Abnar and Willem Zuidema. “Quantifying attention flow in transformers”. In: *arXiv preprint arXiv:2005.00928* (2020).

[2] Julius Adebayo et al. “Sanity Checks for Saliency Maps”. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf.

[3] ClearView AI. *ClearView AI*. 2022. URL: https://www.clearview.ai/ (visited on 06/26/2022).

[4] Julia Angwin et al. *Machine Bias*. 2016. URL: https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing (visited on 06/26/2022).

[5] Sebastian Bach et al. “On Pixel-Wise Explanations for Non-Linear Classifier Decisions by Layer-Wise Relevance Propagation”. In: *PLOS ONE* 10.7 (July 2015), pp. 1–46. DOI: 10.1371/journal.pone.0130140. URL: https://doi.org/10.1371/journal.pone.0130140.

[6] Jasmijn Bastings et al. “"Will You Find These Shortcuts?" A Protocol for Evaluating the Faithfulness of Input Salience Methods for Text Classification”. In: *ArXiv* abs/2111.07367 (2021).

[7] Sidney Black et al. “GPT-NeoX-20B: An Open-Source Autoregressive Language Model”. In: *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*. virtual+Dublin: Association for Computational Linguistics, May 2022, pp. 95–136. DOI: 10.18653/v1/2022.bigscience-1.9. URL: https://aclanthology.org/2022.bigscience-1.9.

[8] Samuel R. Bowman et al. "A large annotated corpus for learning natural language inference". In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, Sept. 2015, pp. 632–642. DOI: 10.18653/v1/D15-1075. URL: https://aclanthology.org/D15-1075.

[9] Adrian MP Brasoveanu and Ruazvan Andonie. "Visualizing transformers for nlp: a brief survey". In: *2020 24th International Conference Information Visualisation (IV)*. IEEE. 2020, pp. 270–279.

[10] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[11] Joy Buolamwini and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification". In: *FAT*. 2018.

[12] Oana-Maria Camburu et al. "e-SNLI: Natural Language Inference with Natural Language Explanations". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc., 2018, pp. 9539–9549. URL: http://papers.nips.cc/paper/8163-e-snli-natural-language-inference-with-natural-language-explanations.pdf.

[13] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. DOI: 10.48550/ARXIV.2204.02311. URL: https://arxiv.org/abs/2204.02311.

[14] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Lin-*

guistics: Human Language Technologies, Volume 1 (Long and Short Papers). Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[15] Jay DeYoung et al. *ERASER: A Benchmark to Evaluate Rationalized NLP Models*. 2019. arXiv: 1911.03429 [cs.CL].

[16] Jesse Dodge et al. "Documenting the English Colossal Clean Crawled Corpus". In: *ArXiv* abs/2104.08758 (2021).

[17] Finale Doshi-Velez and Been Kim. "Towards A Rigorous Science of Interpretable Machine Learning". In: *arXiv: Machine Learning* (2017).

[18] Lauren Kirchner Emmanuel Martinez. *The Secret Bias Hidden in Mortage-Approval Algorithms*. 2021. URL: https://themarkup.org/denied/2021/08/25/the-secret-bias-hidden-in-mortgage-approval-algorithms (visited on 06/26/2022).

[19] Matt Gardner et al. "Competency Problems: On Finding and Removing Artifacts in Language Data". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 1801–1813. DOI: 10.18653/v1/2021.emnlp-main.135. URL: https://aclanthology.org/2021.emnlp-main.135.

[20] Samuel Gehman et al. "RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models". In: *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 3356–3369. DOI: 10.18653/v1/2020.findings-emnlp.301. URL: https://aclanthology.org/2020.findings-emnlp.301.

[21] Amirata Ghorbani, Abubakar Abid, and James Zou. *Interpretation of Neural Networks is Fragile*. 2017. DOI: 10.48550/ARXIV.1710.10547. URL: https://arxiv.org/abs/1710.10547.

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

[23] Google. *Recent Advances In Google Translate*. 2020. URL: https://ai.googleblog.com/2020/06/recent-advances-in-google-translate.html (visited on 06/26/2022).

[24] Peter Hase, Harry Xie, and Mohit Bansal. "The Out-of-Distribution Problem in Explainability and Search Methods for Feature Importance Explanations". In: *NeurIPS*. 2021.

[25] John Hewitt and Christopher D. Manning. "A Structural Probe for Finding Syntax in Word Representations". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4129–4138. DOI: 10.18653/v1/N19-1419. URL: https://aclanthology.org/N19-1419.

[26] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural Comput.* 9.8 (1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[27] Jordan Hoffmann et al. *Training Compute-Optimal Large Language Models*. 2022. DOI: 10.48550/ARXIV.2203.15556. URL: https://arxiv.org/abs/2203.15556.

[28] Sara Hooker et al. “Evaluating Feature Importance Estimates”. In: *arXiv* (2018). URL: https://arxiv.org/pdf/1806.10758.pdf.

[29] Alon Jacovi et al. *Contrastive Explanations for Model Interpretability*. 2021. DOI: 10.48550/ARXIV.2103.01378. URL: https://arxiv.org/abs/2103.01378.

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

[31] Been Kim et al. “Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV)”. In: (2017). DOI: 10.48550/ARXIV.1711.11279. URL: https://arxiv.org/abs/1711.11279.

[32] Narine Kokhlikyan et al. “Investigating sanity checks for saliency maps with image and text classification”. In: *arXiv preprint arXiv:2106.07475* (2021).

[33] Kundan Krishna, Jeffrey Bigham, and Zachary C. Lipton. “Does Pretraining for Summarization Require Knowledge Transfer?” In: *Findings of the Association for Computational Linguistics: EMNLP 2021*. Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 3178–3189. DOI: 10.18653/v1/2021.findings-emnlp.273. URL: https://aclanthology.org/2021.findings-emnlp.273.

[34] Zachary Chase Lipton. “The Mythos of Model Interpretability”. In: *CoRR* abs/1606.03490 (2016). arXiv: 1606.03490. URL: http://arxiv.org/abs/1606.03490.

[35] Yinhan Liu et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *ArXiv* abs/1907.11692 (2019).

[36] Yang Young Lu et al. “DANCE: Enhancing saliency maps using decoys”. In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 7124–7133. URL: https://proceedings.mlr.press/v139/lu21b.html.

[37] Scott M. Lundberg and Su-In Lee. “A Unified Approach to Interpreting Model Predictions”. In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 4768–4777. ISBN: 9781510860964.

[38] Tom McCoy, Ellie Pavlick, and Tal Linzen. “Right for the Wrong Reasons: Diagnosing Syntactic Heuristics in Natural Language Inference”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 3428–3448. DOI: 10.18653/v1/P19-1334. URL: https://aclanthology.org/P19-1334.

[39] Bonan Min et al. “Recent advances in natural language processing via large pre-trained language models: A survey”. In: *arXiv preprint arXiv:2111.01243* (2021).

[40] Eirini Ntoutsi et al. “Bias in data-driven artificial intelligence systems—An introductory survey”. In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 10.3 (2020), e1356.

[41] Alec Radford et al. “Language Models are Unsupervised Multitask Learners”. In: 2019.

[42] Colin Raffel et al. “Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer”. In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[43] 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*. KDD 16. San Francisco, California, USA: Association for Computing Machinery, 2016, pp. 1135–1144. ISBN: 9781450342322. DOI: 10.1145/2939672.2939778. URL: https://doi.org/10.1145/2939672.2939778.

[44] Cynthia Rudin. “Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead”. In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[45] Ramprasaath R Selvaraju et al. “Grad-cam: Visual explanations from deep networks via gradient-based localization”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 618–626.

[46] Lloyd S. Shapley. *A Value for N-Person Games*. Santa Monica, CA: RAND Corporation, 1952. DOI: 10.7249/P0295.

[47] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. *Self-Attention with Relative Position Representations*. 2018. DOI: 10.48550/ARXIV.1803.02155. URL: https://arxiv.org/abs/1803.02155.

[48] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. “Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps”. In: *CoRR* abs/1312.6034 (2014).

[49] Daniel Smilkov et al. “SmoothGrad: removing noise by adding noise”. In: *CoRR* abs/1706.03825 (2017). arXiv: 1706.03825. URL: http://arxiv.org/abs/1706.03825.

[50] Richard Socher et al. “Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank”. In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Seattle, Washing-

ton, USA: Association for Computational Linguistics, Oct. 2013, pp. 1631–1642. URL: https://aclanthology.org/D13-1170.

[51] Spearmans Rank Correlation Coefficient. *Spearmans Rank Correlation Coefficient*. [Online; accessed 01-August-2022]. 2022. URL: https://en.wikipedia.org/wiki/Spearman%5C%27s_rank_correlation_coefficient.

[52] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. “One pixel attack for fooling deep neural networks”. In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 828–841.

[53] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. “Axiomatic Attribution for Deep Networks”. In: *Proceedings of the 34th International Conference on Machine Learning - Volume 70*. ICML17. Sydney, NSW, Australia: JMLR.org, 2017, pp. 3319–3328.

[54] Yi Tay et al. *Scale Efficiently: Insights from Pre-training and Fine-tuning Transformers*. 2021. DOI: 10.48550/ARXIV.2109.10686. URL: https://arxiv.org/abs/2109.10686.

[55] Ian Tenney, Dipanjan Das, and Ellie Pavlick. “BERT Rediscovers the Classical NLP Pipeline”. In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 4593–4601. DOI: 10.18653/v1/P19-1452. URL: https://aclanthology.org/P19-1452.

[56] Robert Tibshirani. “Regression Shrinkage and Selection Via the Lasso”. In: *Journal of the Royal Statistical Society: Series B (Methodological)* 58.1 (1996), pp. 267–288. DOI: https://doi.org/10.1111/j.2517-6161.1996.tb02080.x. eprint: https://rss.onlinelibrary.wiley.com/doi/pdf/10.1111/j.2517-6161.1996.tb02080.

x. URL: https://rss.onlinelibrary.wiley.com/doi/abs/10.1111/j.2517-6161.1996.tb02080.x.

[57] Ashish Vaswani et al. "Attention is All you Need". In: *ArXiv* abs/1706.03762 (2017).

[58] Sandra Wachter, Brent Mittelstadt, and Chris Russell. "Counterfactual Explanations without Opening the Black Box: Automated Decisions and the GDPR". In: (2017). DOI: 10.48550/ARXIV.1711.00399. URL: https://arxiv.org/abs/1711.00399.

[59] Alex Wang et al. "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding". In: In the Proceedings of ICLR. 2019.

[60] Sarah Wiegreffe and Yuval Pinter. "Attention is not not explanation". In: *arXiv preprint arXiv:1908.04626* (2019).

[61] Kyle Wigger. *Open source NLP is fueling a new wave of startups*. 2021. URL: https://venturebeat.com/2021/12/23/open-source-nlp-is-fueling-a-new-wave-of-startups/ (visited on 07/03/2022).

[62] Adina Williams, Nikita Nangia, and Samuel Bowman. "A Broad-Coverage Challenge Corpus for Sentence Understanding through Inference". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1112–1122. DOI: 10.18653/v1/N18-1101. URL: https://aclanthology.org/N18-1101.

[63] Thomas Wolf et al. "Transformers: State-of-the-art natural language processing". In: *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*. 2020, pp. 38–45.

[64] Feiyu Xu et al. “Explainable AI: A brief survey on history, research areas, approaches and challenges”. In: *CCF international conference on natural language processing and Chinese computing*. Springer. 2019, pp. 563–574.

[65] Mengjiao Yang and Been Kim. “Benchmarking Attribution Methods with Relative Feature Importance”. In: *arXiv: Learning* (2019).

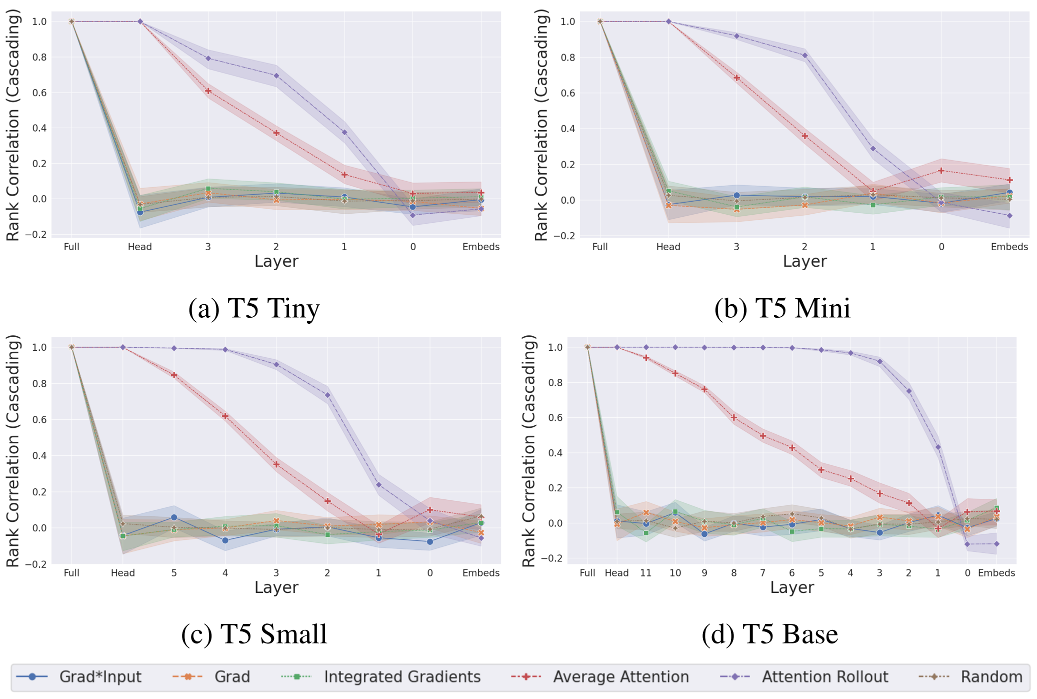
[66] Peter Young et al. “From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions”. In: *Transactions of the Association for Computational Linguistics* 2 (2014), pp. 67–78. DOI: 10.1162/tacl_a_00166. URL: https://aclanthology.org/Q14-1006.

[67] Chulhee Yun et al. “Are Transformers universal approximators of sequence-to-sequence functions?” In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=ByxRM0Ntvr.

[68] Yilun Zhou et al. “Do Feature Attribution Methods Correctly Attribute Features?” In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36 (June 2022), pp. 9623–9633. DOI: 10.1609/aaai.v36i9.21196.

[69] Yukun Zhu et al. “Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books”. In: *CoRR* abs/1506.06724 (2015). arXiv: 1506.06724. URL: http://arxiv.org/abs/1506.06724.

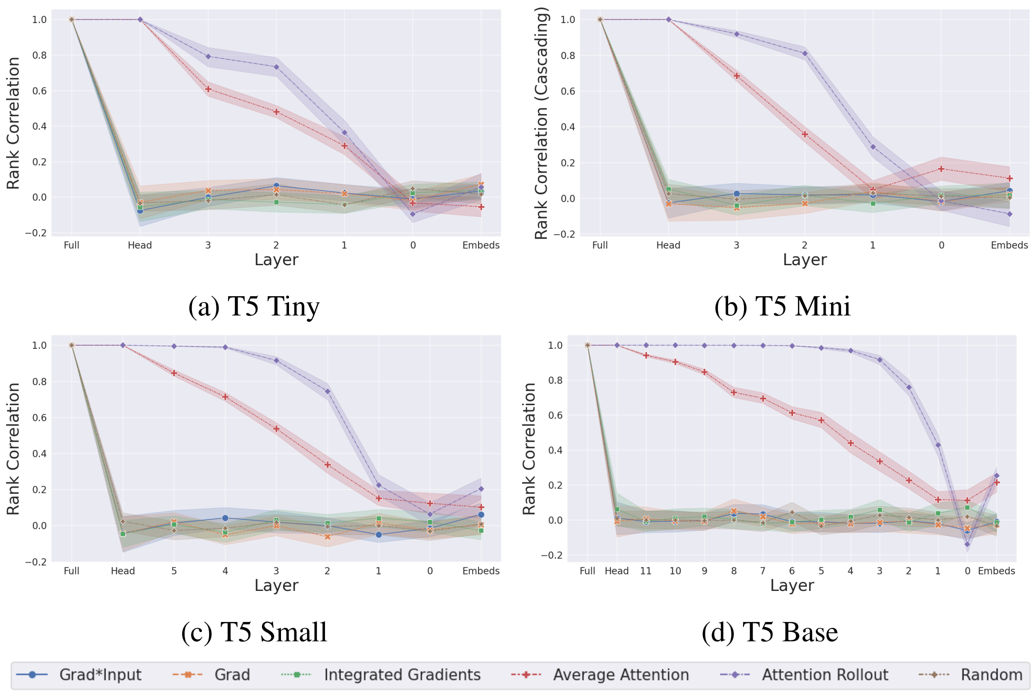
A | Additional Layer Randomization Figures



(a) T5 Tiny (b) T5 Mini

(c) T5 Small (d) T5 Base

Figure A.1: Spearman Correlations between explanations generated via cascading layer randomization (Section 3.3). Layer randomization was applied to four T5 models of differing sizes finetuned on the MultiNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model has been randomized up to the corresponding layer on the x axis. Randomization starts from the last layer and goes backward (from left to right in the graph). Faithful explanations should have decreasing correlations (or decrease all the way to zero) as layers of the model are randomized. High correlations indicate a lack of sensitivity to model parameters (i.e. unfaithful explanations).



(a) T5 Tiny

(b) T5 Mini

(c) T5 Small

(d) T5 Base

Figure A.2: Spearman Correlations between explanations generated via independent layer randomization (Section 3.3). Independent layer randomization was applied to four different T5 models of increasing size finetuned on the MultiNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model parameters at the corresponding layer on the x axis have been randomized. Faithful explanations should have low correlations for each layer of the model except "Full", the original explanation on the far left.

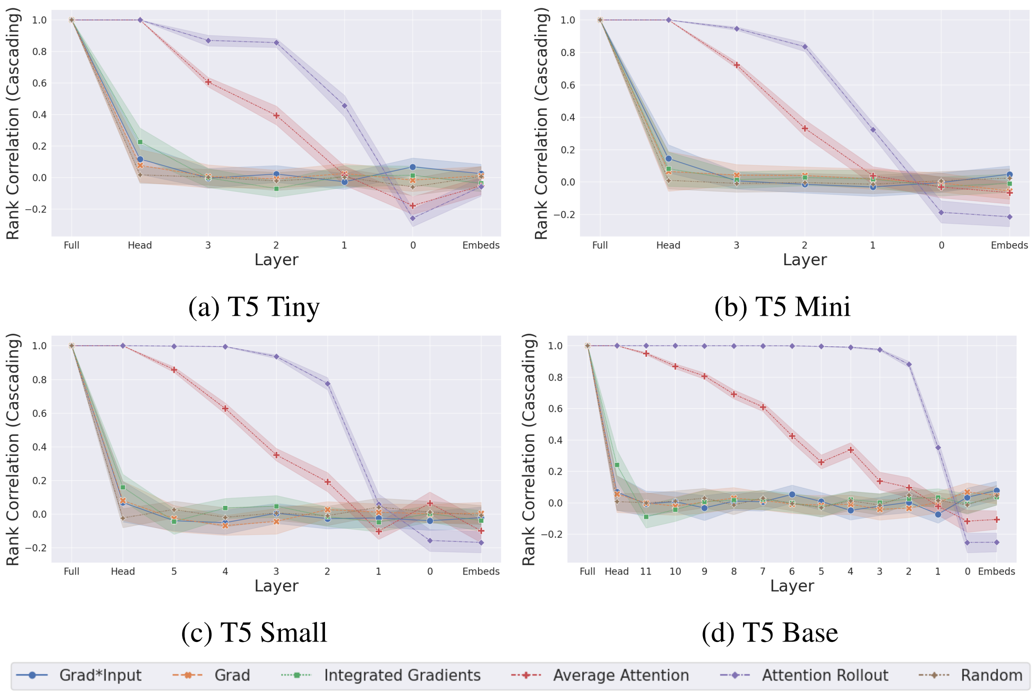
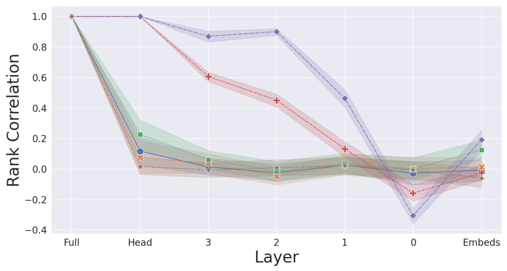
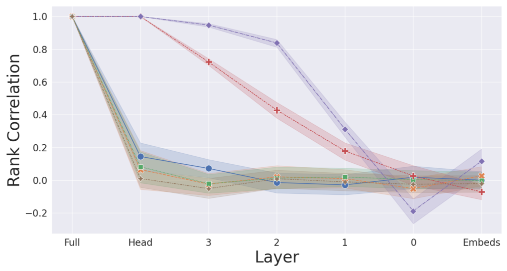


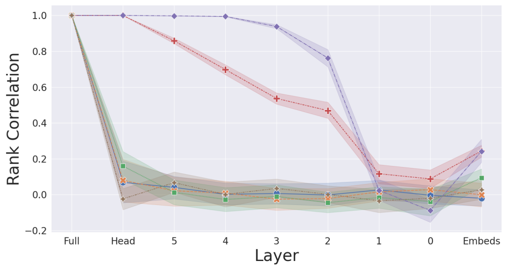
Figure A.3: Spearman Correlations between explanations generated via cascading layer randomization (Section 3.3). Layer randomization was applied to four T5 models of differing sizes finetuned on the e-SNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model has been randomized up to the corresponding layer on the x axis. Randomization starts from the last layer and goes backward (from left to right in the graph). Faithful explanations should have decreasing correlations (or decrease all the way to zero) as layers of the model are randomized. High correlations indicate a lack of sensitivity to model parameters (i.e. unfaithful explanations).



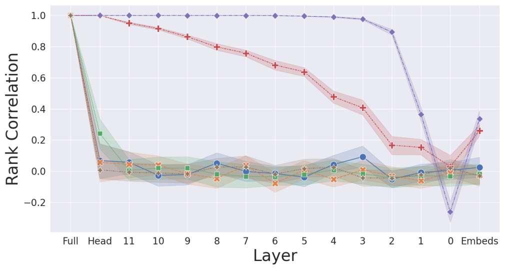
(a) T5 Tiny



(b) T5 Mini



(c) T5 Small



(d) T5 Base

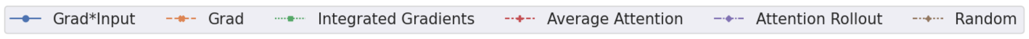
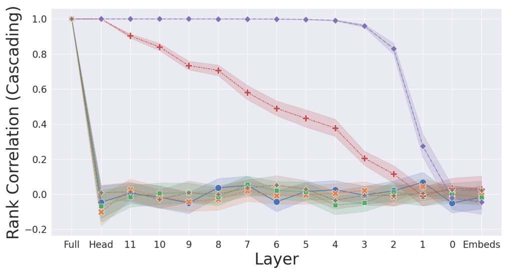
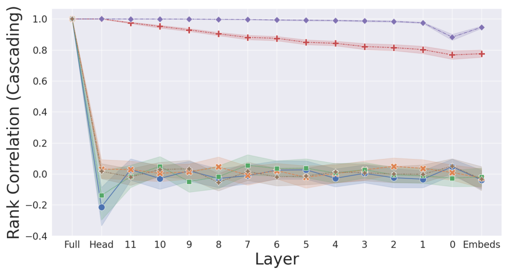


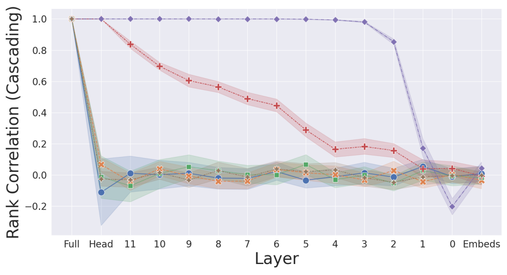
Figure A.4: Spearman Correlations between explanations generated via independent layer randomization (Section 3.3). Independent layer randomization was applied to four different T5 models of increasing size finetuned on the e-SNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model parameters at the corresponding layer on the x axis have been randomized. Faithful explanations should have low correlations for each layer of the model except "Full", the original explanation on the far left.



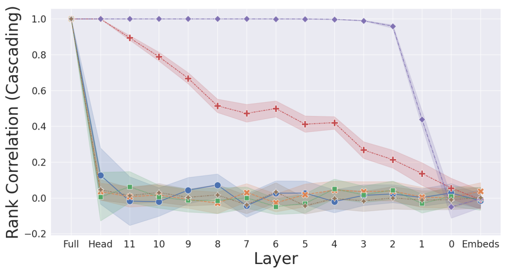
(a) T5 Base



(b) GPT2 Small



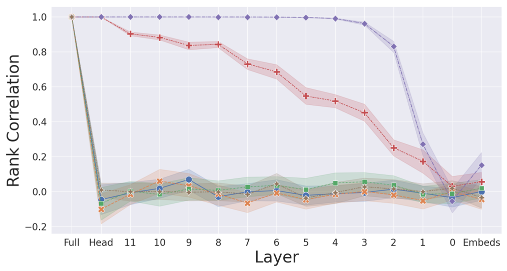
(c) RoBERTa Base



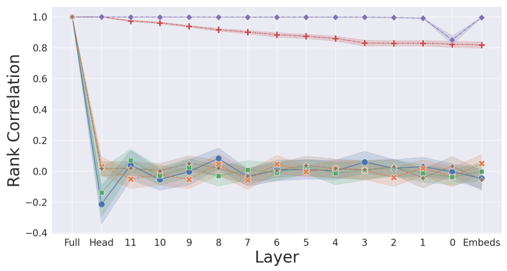
(d) BERT Base

Grad*Input
Grad
Integrated Gradients
Average Attention
Attention Rollout
Random

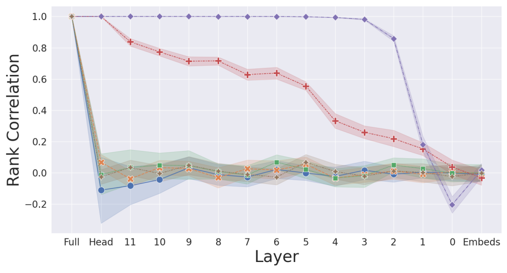
Figure A.5: Spearman Correlations between explanations generated via cascading layer randomization (Section 3.3). Layer randomization was applied to four different pretrained models finetuned on the MultiNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model has been randomized up to the corresponding layer on the x axis. Randomization starts from the last layer and goes backward (from left to right in the graph)



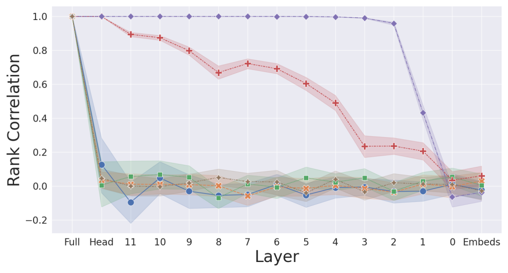
(a) T5 Base



(b) GPT2 Small

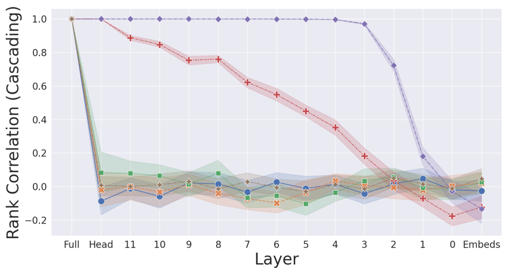


(c) RoBERTa Base

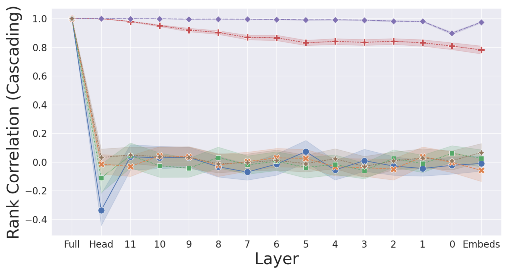


(d) BERT Base

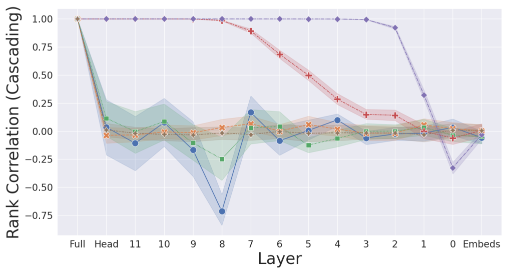
Figure A.6: Spearman Correlations between explanations generated via independent layer randomization (Section 3.3). Independent layer randomization was applied to four different pretrained models finetuned on the MultiNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model parameters at the corresponding layer on the x axis have been randomized.



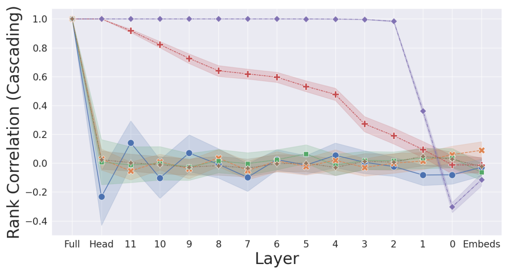
(a) T5 Base



(b) GPT2 Small



(c) RoBERTa Base



(d) BERT Base

Figure A.7: Spearman Correlations between explanations generated via cascading layer randomization (Section 3.3). Layer randomization was applied to four different pretrained models finetuned on the e-SNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model has been randomized up to the corresponding layer on the x axis. Randomization starts from the last layer and goes backward (from left to right in the graph)

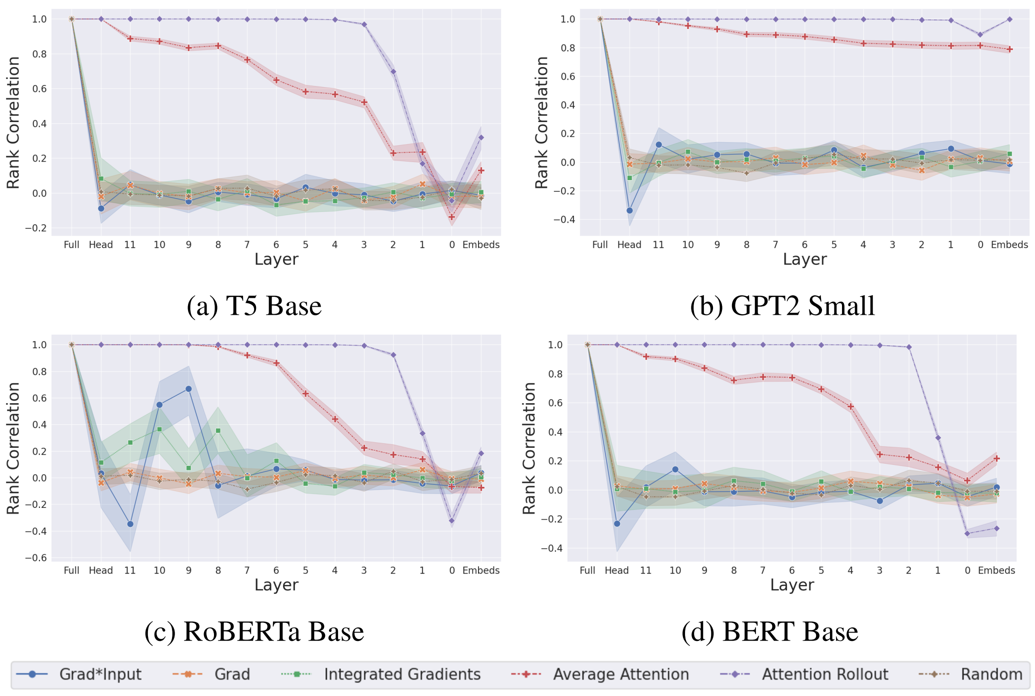
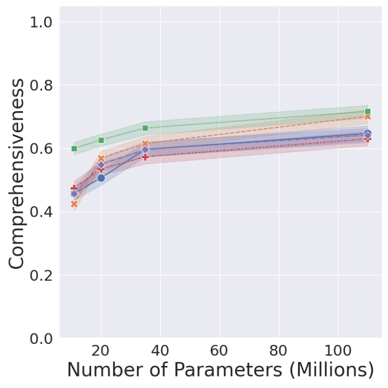
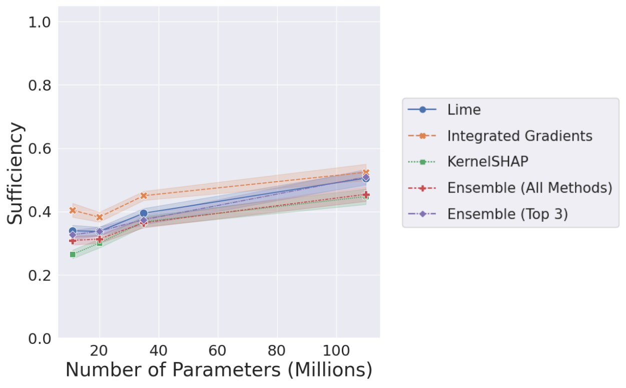


Figure A.8: Spearman Correlations between explanations generated via independent layer randomization (Section 3.3). Independent layer randomization was applied to four different pretrained models finetuned on the e-SNLI dataset. The x axis is the layers of each model. The y axis is the rank correlation between the original explanation and the explanation produced by the same method when the model parameters at the corresponding layer on the x axis have been randomized.

B | Additional Ensemble Figures

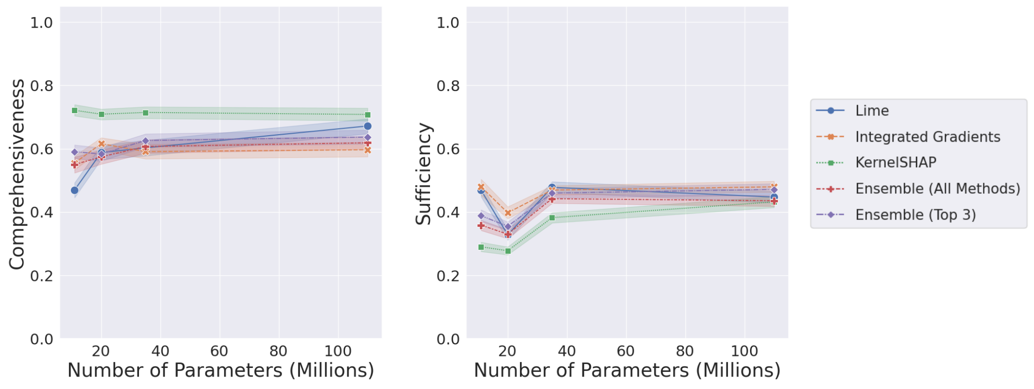


(a) Comprehensiveness



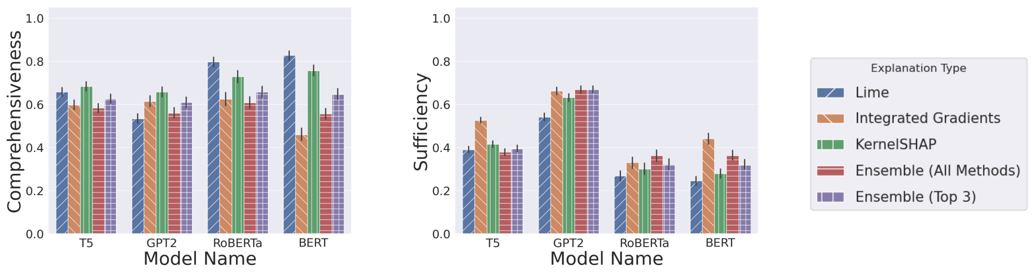
(b) Sufficiency

Figure B.1: Comprehensiveness and Sufficiency for ensemble explanations of four T5 models of different sizes (finetuned on the MultiNLI dataset). LIME, SHAP, and Integrated Gradients are included for comparison. The x axis is the number of parameters in each model. The y axis is either the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness and low Sufficiency values. The values are averaged over 500 examples and shown with 95% confidence intervals.



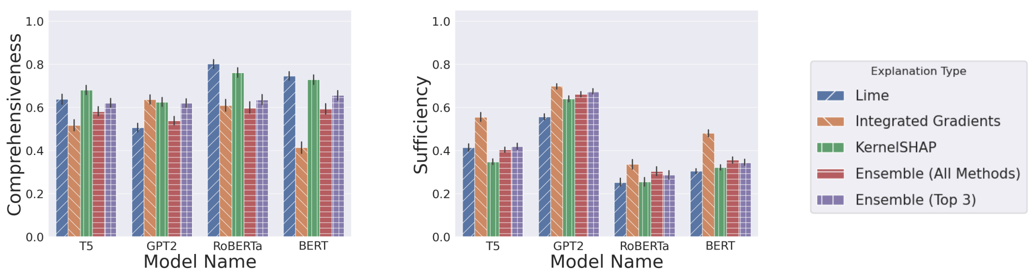
(a) Comprehensiveness (b) Sufficiency

Figure B.2: Comprehensiveness and Sufficiency for ensemble explanations of four T5 models of different sizes (finetuned on the e-SNLI dataset). LIME, SHAP, and Integrated Gradients are included for comparison. The x axis is the number of parameters in each model. The y axis is either the Comprehensiveness or Sufficiency value. Faithful explanations should have high Comprehensiveness and low Sufficiency values. The values are averaged over 500 examples and shown with 95% confidence intervals.



(a) Comprehensiveness (b) Sufficiency

Figure B.3: Comprehensiveness and Sufficiency for ensemble explanations of four different pretrained models (finetuned on the MultiNLI dataset). LIME, SHAP, and Integrated Gradients are included for comparison. The x axis is the name of each of the pretrained models. The y axis is either the Comprehensiveness or Sufficiency value. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.



(a) Comprehensiveness (b) Sufficiency

Figure B.4: Comprehensiveness and Sufficiency for ensemble explanations of four different pretrained models (finetuned on the e-SNLI dataset). LIME, SHAP, and Integrated Gradients are included for comparison. The x axis is the name of each of the pretrained models. The y axis is either the Comprehensiveness or Sufficiency value. The values are averaged over 500 examples and shown with black bars indicating 95% confidence intervals.

C | Additional Adversarial Example Figures

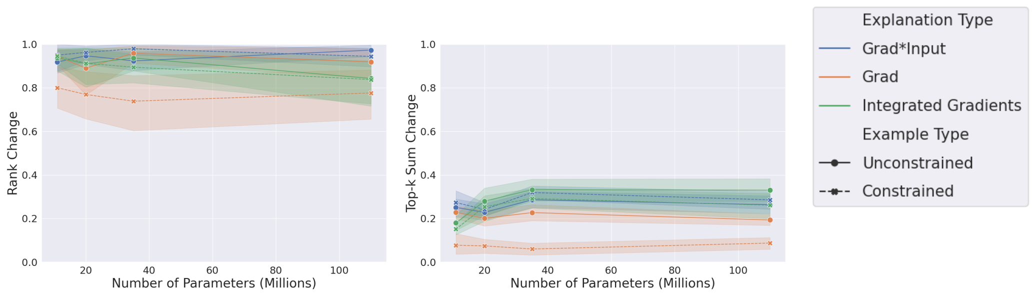


Figure C.1: Rank Change and Top-k Sum Change values for adversarial examples generated for explanations of four T5 models of different sizes (finetuned on the MultiNLI dataset). The x axis is the number of parameters in each model. The y axis is the Rank Change or Top-k Sum Change value. Adversarially robust methods should have low Rank Change and Top-k Sum Change values. The solid lines are values for examples generated using the unconstrained objective, and the dotted lines are for examples generated using the constrained objective. The values are averaged over 16 examples and shown with 95% confidence intervals.

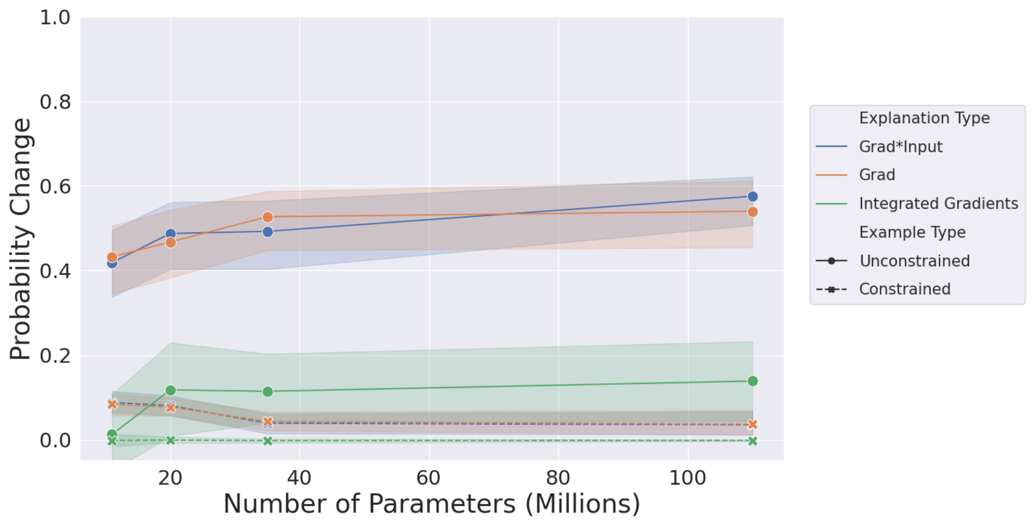


Figure C.2: Probability Change values for adversarial examples generated for explanations of four T5 models of different sizes (finetuned on the MultiNLI dataset). The x axis is the number of parameters in each model. The y axis is the Probability Change value. Lower values are preferable. The solid lines are values for examples generated using the unconstrained objective, and the dotted lines are for examples generated using the constrained objective. The values are averaged over 16 examples and shown with 95% confidence intervals.

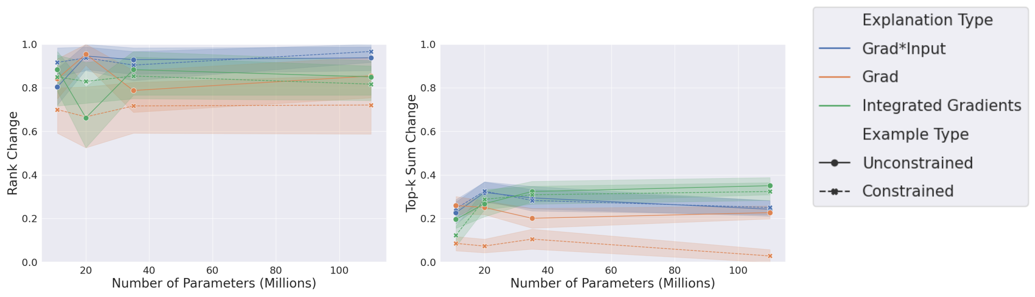


Figure C.3: Rank Change and Top-k Sum Change values for adversarial examples generated for explanations of four T5 models of different sizes (finetuned on the e-SNLI dataset). The x axis is the number of parameters in each model. The y axis is the Rank Change or Top-k Sum Change value. Adversarially robust methods should have low Rank Change and Top-k Sum Change values. The solid lines are values for examples generated using the unconstrained objective, and the dotted lines are for examples generated using the constrained objective. The values are averaged over 16 examples and shown with 95% confidence intervals.

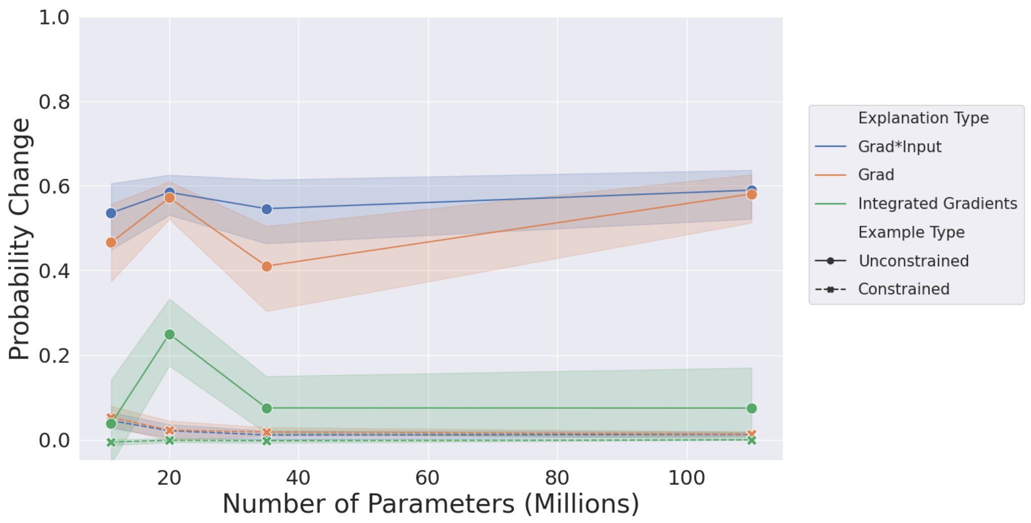
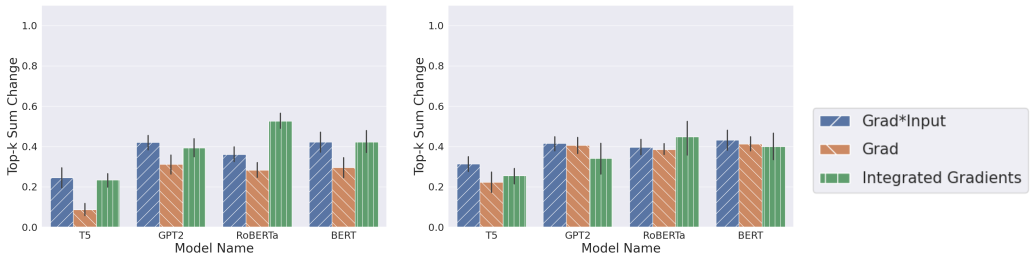
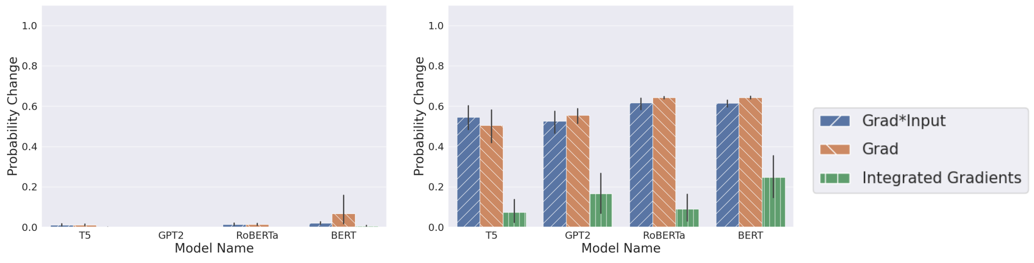


Figure C.4: Probability Change values for adversarial examples generated for explanations of four T5 models of different sizes (finetuned on the e-SNLI dataset). The x axis is the number of parameters in each model. The y axis is the Probability Change value. Lower values are preferable. The solid lines are values for examples generated using the unconstrained objective, and the dotted lines are for examples generated using the constrained objective. The values are averaged over 16 examples and shown with 95% confidence intervals.



(a) Constrained Top-k Sum Change (b) Unconstrained Top-k Sum Change

Figure C.5: Top-k Sum Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the e-SNLI dataset. The x axis is the name of each of the pretrained models. The y axis is the Top-k Sum Change value. Robust explanations should have low values for Top-k Sum Change. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.



(a) Constrained Probability Change (b) Unconstrained Probability Change

Figure C.6: Probability Change values for adversarial examples generated for explanations of four different pretrained models finetuned on the e-SNLI dataset. The x axis is the name of each of the pretrained models. The y axis is the Probability Change value. Lower values are preferable. The values are averaged over 16 examples and shown black bars indicating 95% confidence intervals.