DEEP NEURAL NETWORKS EXPLAINABILITY: ALGORITHMS AND APPLICATIONS

A Dissertation

by

MENGNAN DU

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Xia (Ben) Hu
Committee Members,	James Caverlee
	Shuiwang Ji
	Xiaoning Qian
Head of Department,	Scott Schaefer

December 2021

Major Subject: Computer Science

ABSTRACT

Deep neural networks (DNNs) are progressing at an astounding rate, and these models have a wide range of real-world applications. Despite the superior performance, DNN models are often regarded as black-boxes and criticized by the lack of interpretability, since these models cannot provide meaningful explanations on how a certain prediction is made. Without the explanations to enhance the transparency of DNN models, it would become difficult to build up trust and credibility among end-users. In this dissertation, we investigate the following three research questions: *How can we provide explanations for pre-trained DNN models so as to provide insights into their decision making process? How can we make use of explanations to enhance the generalization ability of DNN models? And how can we employ explanations to promote the fairness of DNN models?*

First, we explore the explainability of two standard DNN architectures, including convolutional neural networks (CNNs) and recurrent neural networks (RNNs). Experimental results over a series of image and text classification benchmarks demonstrate the faithfulness and interpretability of the proposed two explanation methods. Second, we make use of explainability as a debugging tool to examine the vulnerability and failure reasons of DNNs, which further lead to insights that can be used to enhance the generalization ability of DNN models. We propose CREX and LTGR two frameworks, which encourage DNN models to focus more on evidence that actually matters for the task at hand, and to avoid overfitting to data-dependent bias and artifacts. Experimental analysis over several text benchmark datasets validate that our CREX and LTGR frameworks could effectively increase the generalization ability of DNN models. Third, explainability based analysis indicates that DNN models trained with standard cross entropy loss tend to capture the spurious correlation between fairness sensitive information in encoder representations with specific class labels. We propose a new mitigation technique, namely RNF, that achieves fairness by debiasing only the task-specific classification head of DNN models. Experimental results over several benchmark datasets demonstrate our RNF framework to effectively reduce discrimination of DNN models with minimal degradation in task-specific performance.

ACKNOWLEDGMENTS

First and foremost I am extremely grateful to my advisor Dr. Xia Hu for his invaluable advice and continuous support for my PhD study. He set a very high standard for my research, helped me cultivate good research taste, and provided guidance about how to conduct research that can lead to high impact. Beyond research, he also helped me set long-term career goals, and offered assistance patiently at every stage towards achieving this career goal.

Second, I like would to offer my special thanks to my dissertation committee, Dr. James Caverlee, Dr. Shuiwang Ji, and Dr. Xiaoning Qian. They gave me a lot of kind supports and helpful suggestions for my research projects and dissertation.

Additionally, I would like to thank all members of the DATA lab at Texas A&M for their kind support and collaboration, which has made my study and life at A&M a wonderful time. I would like to express my sincere gratitude to my mentors at Adobe research and Microsoft research, who gave me insightful suggestions for a couple of my projects during the internship time.

Last but not least, I would like to extend my sincere thanks to my parents. Without their encouragement and support, this amazing scientific journey would not have been possible.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Dr. Xia Hu [advisor], and Dr. James Caverlee, Dr. Shuiwang Ji of the Department of Computer Science and Engineering and Dr. Xiaoning Qian of the Department of Electrical and Computer Engineering.

All other work conducted for the dissertation was completed by the student independently.

Funding Sources

The work is in part supported by NSF IIS-1657196, IIS-1718840, CNS-1816497, IIS-1900990 and DARPA N66001-17-2-4031. The views and conclusions contained in this dissertation are those of the authors and should not be interpreted as representing any funding agencies.

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

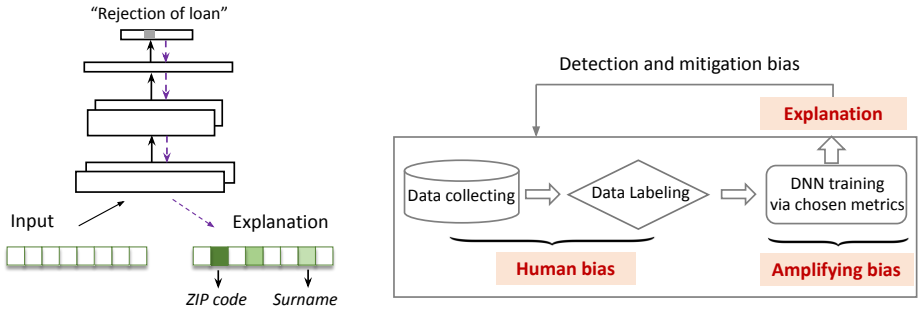
TABLE Page

1. Introduction

Deep neural networks (DNNs) are progressing at an astounding rate in the past decade. These models have a wide range of real-world applications, such as movie recommendations of Netflix, neural machine translation of Google, speech recognition of Amazon Alexa. Despite the successes, DNN have its own limitations and drawbacks. The most significant one is the lack of transparency behind their behaviors, which leaves users with little understanding of how particular decisions are made by these models. Consider, for instance, an advanced self-driving car equipped with various DNN algorithms doesn't brake or decelerate when confronting a stopped firetruck. This unexpected behavior may frustrate and confuse users, making them wonder why. Even worse, the wrong decisions could cause severe consequences if the car is driving at highway speeds and might finally crash the firetruck. The concerns about the black-box nature of complex DNN models have hampered their further applications in our society, especially in those critical decision-making domains like self-driving cars.

Explainability would be an effective tool to mitigate these problems. It gives DNN models the ability to explain or to present their behaviors in understandable terms to humans [3], which is named interpretability or explainability and we use them interchangeably in this dissertation[1]. Explainability would be an indispensable part for DNN models in order to better serve human beings and bring benefits to society. We investigate DNN explainability from two perspectives: *algorithms and applications*. Firstly, we design post-hoc explanation algorithms for pre-trained DNN models (see Figure 1.1 (a)). The underlying audience for explanations include any end-users and stake-holders. Our explanation could provide insights into the working mechanisms of DNNs and will increase their trust and encourage them to adopt DNN systems. Secondly, we investigate the application of explainability, by treating it as a debugging tool to boost the performance of DNN models (see Figure 1.1 (b)). The underlying audience for this are DNN system developers and

[1]Note that some literature, e.g., [4], explicitly differentiates between interpretability and explainability (or interpretation and explanation). Different work might have their own definitions for these two terms, and there is still no consensus about their subtle difference. In this dissertation, for ease of understandability, we use them interchangeably.



(a) Explanation algorithms (b) Explanation applications

Figure 1.1: (a) We propose explanation algorithms for pre-trained DNN models. (b) We employ explainability as a debugging tool to improve the performance of DNN models.

researchers. On one hand, this helps them better understand the problem, the data and why a model might fail, and eventually increase the model generalization ability and robustness. On the other hand, explainability also could help diagnose the reason that causes the discrimination of DNN models and eventually help improve algorithmic fairness.

Nowadays, DNNs are increasingly being used in decisions and processes that are critical for individuals, businesses, and society, their resulting implications are far-reaching. With explainability as the backbone, the model can have improved *transparency, robustness, and fairness*. Putting them together, our work could help achieve *Responsible AI* (or *Trustworthy AI*). Our work could enable DNNs to be more trustworthy in applications areas such as hiring, lending, criminal justice, healthcare, and education. Ultimately, it enables DNN models to better serve us human beings.

1.1 Dissertation Outline

This dissertation starts from providing explainability to pre-trained DNN models, including two most widely used DNN architectures: CNN and RNN (Chapter 2 - Chapter 3). Then this dissertation introduces how to make use of explainability as a debugging tool to improve the generalization ability and robustness of DNN models (Chapter 4 - Chapter 5). Finally, this dissertation presents how to make use of explainability to improve the fairness of models and reduce algorithmic discrimination (Chapter 6). Specifically this dissertation is organized in following chapters:

Chapter 2 describes providing post-hoc explanation for pre-trained CNN models [5]. We propose to investigate a guided feature inversion framework for taking advantage of the deep architectures towards effective interpretation. The proposed framework not only determines the contribution of each feature in the input but also provides insights into the decision-making process of CNN models. We apply the proposed interpretation model to different CNN architectures to provide explanations for image data. The interpretation results demonstrate the effectiveness of our proposed framework in providing class-discriminative interpretation for CNN-based prediction.

Chapter 3 presents providing post-hoc explanation for pre-trained RNN models [6]. We propose a novel attribution method, called REAT, to provide interpretations to RNN predictions. REAT decomposes the final prediction of a RNN into additive contribution of each word in the input text. This additive decomposition enables REAT to further obtain phrase-level attribution scores. In addition, REAT is generally applicable to various RNN architectures, including GRU, LSTM and their bidirectional versions. Experimental results demonstrate the faithfulness and interpretability of the proposed attribution method.

Chapter 4 develops a framework to improve the generalization ability of DNN models [7]. We propose CREX, which encourages DNN models to focus more on evidences that actually matter for the task at hand, and to avoid overfitting to data-dependent bias. Specifically, CREX regularizes the training process of DNNs with rationales, i.e., a subset of features highlighted by domain experts as justifications for predictions, to enforce DNNs to generate local explanations that conform with expert rationales. Experimental results on two text classification datasets demonstrate the increased credibility of DNNs trained with CREX. Comprehensive analysis further shows that while CREX significantly increases DNN accuracy on new and previously unseen data beyond test set.

Chapter 5 describes how to make use of explainability to interpret and mitigate the shortcut learning behavior of BERT-based NLU models [8]. Recent studies indicate that BERT-based models are prone to rely on shortcut features for prediction, without achieving true language understanding. As a result, these models fail to generalize to real-world out-of-distribution data. We propose to investigate how to diagnose the reasons that lead to the shortcut learning behavior by making use of

explainability, and also propose to mitigate the shortcut learning behavior of NLU models.

Chapter 6 presents an explainability based fairness mitigation framework [9]. Recent studies indicate that DNNs models are prone to show discriminations towards certain demographic groups. We observe that algorithmic discrimination can be explained by the high reliance of the models on fairness sensitive features. We propose a new mitigation technique, namely RNF, that achieves fairness by debiasing only the task-specific classification head of DNN models. The key idea of RNF is to discourage the classification head from capturing spurious correlation between fairness sensitive information in encoder representations with specific class labels. Experimental results over several benchmark datasets demonstrate our RNF framework to effectively reduce discrimination of DNN models with minimal degradation in task-specific performance.

Chapter 7 concludes this dissertation and offers discussion of the future work.

2. Towards Explanation of CNN-based Prediction with Guided Feature Inversion

Convolutional Neural Network (CNN) is one representative family of DNN models. Given raw image pixels as input, CNN models can extract high level image representations automatically, which is primarily achieved through stacking convolutional layers. CNN models have achieved extremely high prediction accuracy in a wide range of computer vision tasks, such as image classification, object detection, and semantic segmentation [10, 11, 12]. In this chapter, we introduce how to provide post-hoc explanation for pre-trained CNN models.[1]

2.1 Introduction

Despite the superior performance, CNN models are often regarded as black-boxes, since these models cannot provide meaningful explanations on how a certain prediction is made. Without the explanations to enhance the transparency of CNN models, it would become difficult to build up trust and credibility among end-users.

We focus on instance-level interpretation, which tries to answer what features of an input lead it to activate the CNN neurons to make a specific prediction. Conventional instance-level interpretations usually follow the philosophy of *local interpretation* [13]. Let $\mathbf{x}$ be an input for a CNN, the prediction of the CNN is denoted as a function $f(\mathbf{x})$. Through monitoring the prediction response provided by the function f around the neighborhood of a given point $\mathbf{x}$, the features in $\mathbf{x}$ which cause a larger change of f will be treated as more relevant to the final prediction. This either can be achieved by perturbing the input and observing the prediction differences [1, 13, 14, 15] (bottom-up manner) or calculating the gradient of output f with respect to input $\mathbf{x}$ [16, 17, 18, 19] (top-down manner). Although these approaches locally interpret CNN predictions to some extent, they usually ignore the intermediate layers of CNN, thus leave out vast informative intermediate information [20, 21]. In addition, these methods have the risk of triggering the artifacts of CNN

[1] Reprinted with permission from "Towards explanation of dnn-based prediction with guided feature inversion." by Mengnan Du, Ninghao Liu, Qingquan Song, and Xia Hu. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1358-1367. 2018. Copyright 2018 by ACM.

models [22, 23]. It has been demonstrated that some generated inputs can fool CNN and lead CNN to make unexpected outputs, which can not be counted as meaningful interpretations. By taking advantage of the intermediate layers information, it is more likely to characterize the behaviors of CNN under normal operating conditions. It motivates us to explore the utilization of intermediate information to derive more accurate interpretations.

Feature inversion has been initially studied for visualizing and understanding intermediate feature representations of CNN [24, 25]. It has been shown that the CNN representation could be inverted to an image which sheds light on the information extracted by each convolutional layer. The inversion results indicate that as the information propagates from the input layer to the output layer, the CNN classifier gradually compresses the input information, and discard information irrelevant to the prediction task. Besides, the inversion result from a specific layer also reveals the amount of information contained in that layer. However, these inversion results are relatively rough and obscure for delicate interpretations. It remains challenging to automatically extract the contributing factors for prediction in the input utilizing the feature inversion.

In this chapter, we propose an instance-level CNN interpretation model by performing guided image feature inversion. Leveraging the observations found in our preliminary experiments that the higher layers of CNN do capture the high-level content of the input as well as its spatial arrangement, we present guided feature reconstructions to explicitly preserve the object localization information in a "mask", so as to provide insights of what information is actually employed by the CNN for the prediction. In order to induce class-discriminative power upon interpretations, we further establish connections between the input and the target object by fine-tuning the interpretation result obtained from guided feature inversions with class-dependent constraints. In addition, we show that the intermediate activation values at higher convolutional layers of CNN are able to behave as a stronger regularizer, leading to more smooth, and continuous saliency maps. This regularization dramatically decreases the possibility to produce artifacts, thus providing more exquisite interpretations.

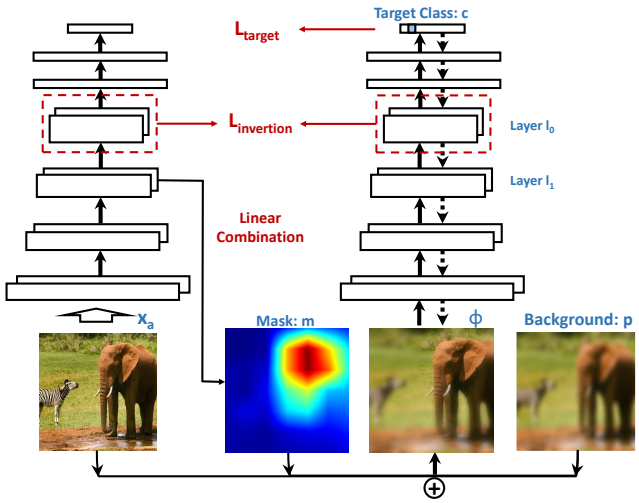


Figure 2.1: An illustration of the proposed interpretation framework. First, the original input $\mathbf{x}_a$ is sent to the CNN (on the left), and the representation at each layer of CNN is calculated and saved. Second, class-discriminative interpretation result is obtained by interacting with the CNN (on the right). The guided feature inversion Φ extracts the location for all the foreground objects (Sec. 2.2.2). Then we fine-tune the inversion result using the activation of the neuron for the target class in the last layer of CNN (Sec. 2.2.3). Besides, we impose a strong regularizer by using the integration of the intermediate layer activations of the original input as the mask (Sec. 2.2.4).

2.2 Interpretation of CNN-based Prediction

In this section, we introduce the proposed interpretation framework for interpreting CNN-based predictions. The pipeline of the proposed framework is illustrated in Fig. 2.1.

2.2.1 Problem Statement

Considering a multiclass classification task, a pre-trained CNN model can be treated as a function $\mathbf{f}(\mathbf{x})$ of the input $\mathbf{x} \in \mathbb{R}^d$. When feeding an input $\mathbf{x}$ to the CNN model, $\mathbf{f}_c(\mathbf{x}) \in [0, 1], c \in \{1, ..., C\}$ represents the corresponding classification probability score for class c. We focus on post-hoc interpretations [4] through explaining the CNN prediction result for a given data instance $\mathbf{x}$ to ensure the generality of the proposed method. We aim to find out the contributing factors in the input $\mathbf{x}$ that lead the CNN to make the prediction. Specifically, let c be the target object class that we want to interpret, and $\mathbf{x}_i$ corresponds to the i^{th} feature, then the interpretation for $\mathbf{x}$ is encoded by a score vector $\mathbf{s} \in \mathbb{R}^d$ where each score element $\mathbf{s}_i \in [0, 1]$ represents how relevant of that feature is for

explaining $\mathbf{f}_c(\mathbf{x})$. We use image classification as an example in this paper. In this case, the input vector $\mathbf{x}_a$ corresponds to the pixels of an image, and the score vector $\mathbf{s}$ will be the saliency map, where the pixels with higher scores represent higher relevance for the classification task.

2.2.2 Interpretation through Feature Inversion

In this section, we derive the initial interpretation for CNN-based interpretation using guided feature inversion. It has been studied that the deep image representation extracted from a layer in CNN could be inverted to a reconstructed image which captures the property and invariance encoded in that layer [24, 25]. An observation is that feature inversion can reveal how much information is preserved in the feature at a specific layer. Specifically, the reconstruction results from features of the first few layers preserve almost all the detailed image information, while the inversions from the last few layers merely contain the rough shape of the original image. This observation shows that CNN gradually filters out unrelated information for the classification task as the layer goes deeper. It thus motivates us to explore the feature inversion of higher layers of CNN to provide explanation for classification result of each instance.

Given a pre-trained L-layers CNN model, the intermediate feature representation at layer $l \in \{1, 2, ..., L\}$ could be denoted as a function $\mathbf{f}^l(\mathbf{x}_a)$ of the input image $\mathbf{x}_a$. The process of inverting the feature representation at layer l_0 can be regarded as computing the approximated inversion $\mathbf{f}^{-1}$ of the representation $\mathbf{f}^{l_0}(\mathbf{x}_a)$. The feature inversion tries to find the image $\mathbf{x}$ that minimizes the following objective function:

$$\mathbf{x}^* = \underset{\mathbf{x}}{\operatorname{argmin}} \quad \|\mathbf{f}^{l_0}(\mathbf{x}) - \mathbf{f}^{l_0}(\mathbf{x}_a)\|^2 + \mathcal{R}(\mathbf{x}), \tag{2.1}$$

where the squared error term forces the representation $\mathbf{f}^{l_0}(\mathbf{x}^*)$ of inversion result $\mathbf{x}^*$ and the original input representation $\mathbf{f}^{l_0}(\mathbf{x}_a)$ to be as similar as possible, while the regularization term $\mathcal{R}$ imposes a natural image prior. As the layer goes deeper for inversion, it is with higher confidence about which part of the input information is ultimately preserved for final prediction. Since the spatial configuration information of the target object is discarded at *fully connected layers*, the feature

inversion cannot recover the accurate object localization information from these layers [24, 25]. Therefore, we fix the inversion layer l_0 to be the last pooling layer before the first fully connected layer. Take the 8-layer AlexNet [11] as an example. We use the *pool5 layer* for feature inversion, where the spatial size is (6×6) and the total number of channels of this layer is 256. $\mathbf{x}$ is initialized randomly, and the optimal inversion result $\mathbf{x}^*$ could be obtained using gradient descent.

We propose the *guided feature inversion method*, where the expected inversion image representation is reformulated as the weighted sum of the original image $\mathbf{x}_a$ and another noise background image $\mathbf{p}$. We replace the optimization target $\mathbf{x}$ in Eq. (2.1) with the guided inversion $\Phi(\mathbf{x}_a, \mathbf{m})$:

$$\Phi(\mathbf{x}_a, \mathbf{m}) = \mathbf{x}_a \odot \mathbf{m} + \mathbf{p} \odot (1 - \mathbf{m}). \tag{2.2}$$

Note that the inversion image representation $\Phi(\mathbf{x}_a)$ should locate in the natural image space manifold. To this end, we derive $\mathbf{p}$ using a blurred image which is obtained by applying a Gaussian blur filter to the original image $\mathbf{x}_a$ [1]. The weight vector $\mathbf{m} \in [0, 1]^d$ denotes the significance of each pixel contributing to the feature representation $\mathbf{f}^{l}(\mathbf{x}_a)$. Therefore, we can recover the object location information from the weight vector $\mathbf{m}$. Instead of directly finding the inversion image representation, we optimize the weight vector $\mathbf{m}$, which is formulated as follows:

$$L_{\text{inversion}}(\mathbf{x}_a, \mathbf{m}) = \|\mathbf{f}^{l_0}(\Phi(\mathbf{x}_a, \mathbf{m})) - \mathbf{f}^{l_0}(\mathbf{x}_a)\|^2 + \alpha \cdot \frac{1}{d} \sum_{i=1}^{d} \mathbf{m}_i. \tag{2.3}$$

The first term corresponds to the inversion error. The error will be zero if all entries within $\mathbf{m}$ equal to 1. In the second term, we limit the area of $\mathbf{m}$ to be as small as possible in order to find out the most contributing regions in input $\mathbf{x}_a$. The parameter α is utilized to balance the inversion error and the area of $\mathbf{m}$. This formulation not only creates image $\Phi(\mathbf{x}_a, m)$ which matches the inner feature representation at layer l_0, but also preserves the object localization information in $\mathbf{m}$.

2.2.3 Class-Discriminative Interpretation

In this section, we derive class-discriminative interpretation to distinguish different categories of objects from the generated mask $\mathbf{m}$. Up to now, we only use the information from the former convolutional layers of CNN to generate mask $\mathbf{m}$. Although it has extracted all the foreground object information which are crucial for subsequent prediction, the connection between the input $\mathbf{x}_a$ and the target label c , which is largely encoded in the rest layers, has not been established yet. The mask $\mathbf{m}$ derived from the aforementioned formulation Eq.(2.3) will highlight all these objects.

We would like the interpretation result to highlight the target class and suppress the irrelevant classes through further leveraging the non-utilized layers. To this end, we render the guided feature reconstruction result $\Phi(\mathbf{x}_a)$ to strongly activate the softmax probability value $\mathbf{f}_c^L$ at the last hidden layer L of CNN for a given target label c, and reduce the activation for other classes $\{1, ..., C\} \setminus c$. In the meantime, we formulate the complementary counterpart of the mask $\mathbf{m}$ as $\mathbf{m}_{bg} = 1 - \mathbf{m}$. It contains irrelevant information with respect to target class c, including image background and other classes of foreground objects. Using the heatmap $\mathbf{m}_{bg}$ as weight, the background part of the image can be calculated as the weighted sum of the original images $\mathbf{x}_a$ and $\mathbf{p}$:

$$\Phi_{bg}(\mathbf{x}_a, \mathbf{m}_{bg}) = \mathbf{x}_a \odot \mathbf{m}_{bg} + \mathbf{p} \odot (1 - \mathbf{m}_{bg}). \tag{2.4}$$

We expect the object information for the target class c to be removed from $\Phi_{bg}(\mathbf{x}_a, \mathbf{m})$ to the maximization degree. When feeding $\Phi_{bg}(\mathbf{x}_a, \mathbf{m})$ to the CNN classifier, the prediction probability is supposed to be small. The class-discriminative interpretation formulation is defined as:

$$L_{\text{target}}(\mathbf{x}_a, \mathbf{m}) = -\mathbf{f}_c^L(\Phi(\mathbf{x}_a, \mathbf{m})) + \lambda \mathbf{f}_c^L(\Phi_{bg}(\mathbf{x}_a, \mathbf{m})) + \beta \cdot \frac{1}{d} \sum_{i=1}^{d} \mathbf{m}_i, \tag{2.5}$$

where λ and β control the importance of the highlighting term, suppressing term and the area of $\mathbf{m}$.

2.2.4 Regularization by Utilizing Intermediate Layers

The aforementioned formulation still has the weakness of generating undesirable artifacts without regularizations imposed to the optimization process. To generate more meaningful interpretation, we impose a stronger natural image prior by utilizing the intermediate activation features of CNN. This is motivated by the fact that higher convolutional layers of CNN are responsive to specific and semantically meaningful natural part (*e.g.*, face, building, or lamp) [20, 21]. These feature layers, when projected down to the pixel space, could correspond to the rough location of these semantic parts. Similar to decomposing an object into the combinations of its high-level parts, we assume the mask $\mathbf{m}$ could be decomposed as the combination of channels at a high-level layer of the targeted CNN model. Specially, let $\mathbf{f}_i^{l_1}(\mathbf{x}_a)$ represent the i^{th} channel of the ${l_1}^{th}$ layer of the CNN. We build the weight mask $\mathbf{m}$ as the weighted sum of the channels at a specific layer l_1:

$$\mathbf{m} = \sum_i \omega_i \mathbf{f}_i^{l_1}(\mathbf{x}_a). \tag{2.6}$$

Parameter vector ω captures the relevance of each channel map for the final prediction. Since the activation values $\mathbf{f}^{l_1}(\mathbf{x}_a)$ could locate at various ranges, the generated mask $\mathbf{m}$ may take a wide range of values if no constraint is imposed to the parameter ω. It is essential to limit $\mathbf{m} \in [0, 1]^n$, so as to guarantee the guided reconstruction in Eq.(2.2) is constrained at the expected input domain range. Therefore, the mask is further normalized through Min-Max normalization: $\mathbf{m} \leftarrow \frac{\mathbf{m} - \min(\mathbf{m})}{\max(\mathbf{m}) - \min(\mathbf{m})}$.

Before applying to the objective function, we still need to enlarge the mask $\mathbf{m}$ from the small resolution to an identical resolution with the original input. To guarantee the smoothness of the enlarged representation $\mathbf{m}$, we apply upsampling using bilinear interpolation. After replacing the original mask at Eq.(2.2) and Eq.(2.4) with the newly derived mask, we expect the guided inversion representation $\Phi(\mathbf{x}_a, \mathbf{m})$ and the background representation $\Phi_{bg}(\mathbf{x}_a, m)$ to become less likely to be affected by artifacts. The interpretation objective Eq.(2.3) and Eq.(2.5) could be reformulated as:

$$L_{\text{inversion}}(\mathbf{x}_a, \omega) = \|\mathbf{f}^{l_0}(\Phi(\mathbf{x}_a, \omega)) - \mathbf{f}^{l_0}(\mathbf{x}_a)\|^2 + \gamma \cdot \|\omega\|_1, \tag{2.7}$$

$$L_{\text{target}}(\mathbf{x}_a, \omega) = -\mathbf{f}_c^L(\Phi(\mathbf{x}_a, \omega)) + \lambda \mathbf{f}_c^L(\Phi_{bg}(\mathbf{x}_a, \omega)) + \delta \cdot \|\omega\|_1, \tag{2.8}$$

where $\omega_i \geq 0, i \in \{1, ..., n\}$, since we only focus on the channel maps which have a positive influence for making a prediction. Parameters γ and δ control the importance of the regularization term. We utilize the ℓ_1-norm regularization to ensure that only very few entries in the parameter vector ω is non-zero. This is motivated from the observation that objects could be depicted using only one or few object parts. This natural image prior brings two benefits. On one hand, it guarantees that less artifacts will be produced in the optimized mask. On the other hand, it dramatically reduce the numbers of the parameters to be optimized, leading to increased efficiency of the optimization.

We apply a two-stage optimization to derive class-discriminative interpretation for CNN-based prediction. In the first stage, we perform interpretation using guided feature inversion to find out the salient foreground part. After initializing the parameter $\omega_i = 0.1, i \in \{1, ..., n\}$, we perform gradient descent optimization to find out the optimal parameter vector ω as well as the mask $\mathbf{m}$. In the second stage, we obtain the class-discriminative interpretation by further fine-tune the parameter vector ω. After initializing ω, we reduce the learning rate every 10 iterations and further fine-tune the parameter ω in order to let the mask $\mathbf{m}$ more relevant to the target label. Note that to tackle the constraint in objective function (2.7) and (2.8) that parameter $\omega_i \geq 0, i \in \{1, ..., n\}$, we simply clip the ω to the valid range after each gradient descent iteration. At last, we generate the interpretation mask $\mathbf{m}$ for target class c.

2.3 Experiments

2.3.1 Visualization of Interpretation Results

In the following experiments, unless stated otherwise, the interpretation results are provided based on VGG-19 [12]. Specifically, we utilize the pre-trained VGG-19 model from torchvision[2]. Its Top-1 prediction error and Top-5 prediction error on ImageNet dataset are 27.62% and 9.12%, respectively. For inversion layer l_0, we use the *pool5* layer (the 5th pooling layer), and the size of representation at this layer is $(7 \times 7 \times 512)$. As for the base channel layer l_1 in Eq. (2.6), we utilize

[2]http://pytorch.org/docs/master/torchvision/models.html

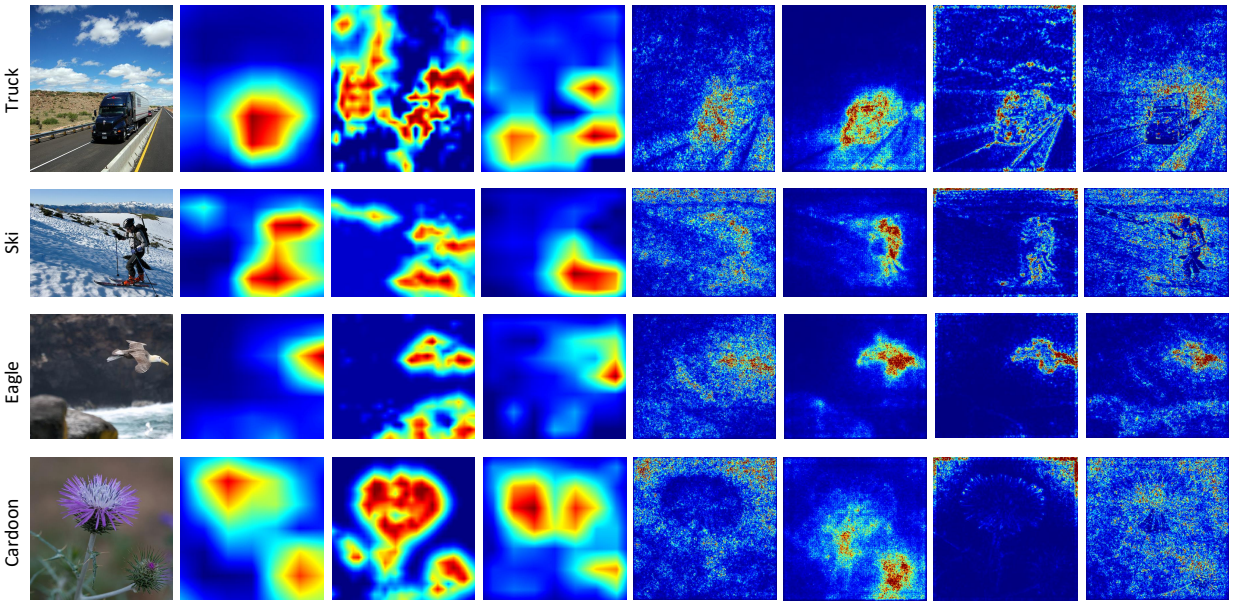
conv5_4, which is the layer prior to *pool5*, and has 512 channels of size (14×14). Therefore the length of parameter vector ω is also 512. The entries of ω are all initialized as 0.1. The λ is fixed to 1. The ℓ_1-norm regularizer parameter γ and δ are set to 10 and 1, respectively. All these parameters are tuned based on the quantitative and qualitative performance of the interpretation on a subset of the ILSVRC2014 [26] training set.

For input images, we resize them to the shape $(224 \times 224 \times 3)$, and transform them to the range $[0, 1]$, and then normalize them using mean vector $[0.485, 0.456, 0.406]$ and standard deviation vector $[0.229, 0.224, 0.225]$. No further pre-processing is performed. The background image $\mathbf{p}$ is obtained by applying a single Gaussian blur of radius 11 to the original input.

We employ Adam [27] optimizer to perform gradient descent, which achieves faster convergence rate than stochastic gradient descent (SGD). The number of iteration steps is 10, and 70 for the first stage and the second stage respectively. The learning rate is initialized to 10^{-2} for Adam optimizer, chosen by line search. At the second stage, we apply step decay, and reduce the learning rate by half every ten epochs.

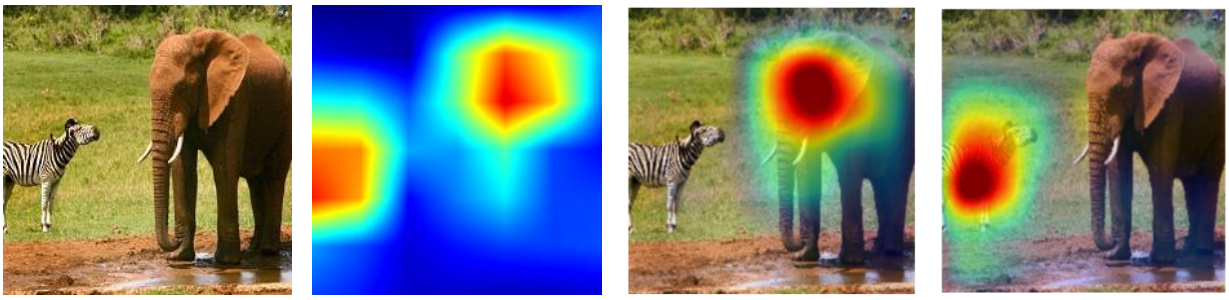
We qualitatively compare the saliency maps produced using the proposed method with those produced by six state-of-the-art methods, including Grad [16], GuidedBP [18], SmoothGrad [19], Integrated [17], Mask [1], Grad-CAM [28], see Fig. 2.2. Comparing to the other methods, it shows that our method generates more visually interpretable saliency maps. Take the second row for example. The DNN predicts the *ski* category with 98.2% confidence, and our method accurately highlights the helmet, skis, as well as ski poles. At the fourth row, our method highlights both the dominant cardoon at the center and the smaller cardoon at the lower right corner.

We also demonstrate that the proposed interpretation method could distinguish different classes as shown in Fig. 2.3. The VGG-19 model classifies the input as *African elephant* with 95.3% confidence, and *zebra* with 0.2% confidence, our model correctly gives the interpretation locations for both of two labels, even though the prediction probability of the latter is much lower than the probability of the former. An interesting discovery obtained from the saliency maps is that the head, ear, and nose parts are most discriminative to distinguish elephant, while the body part is most



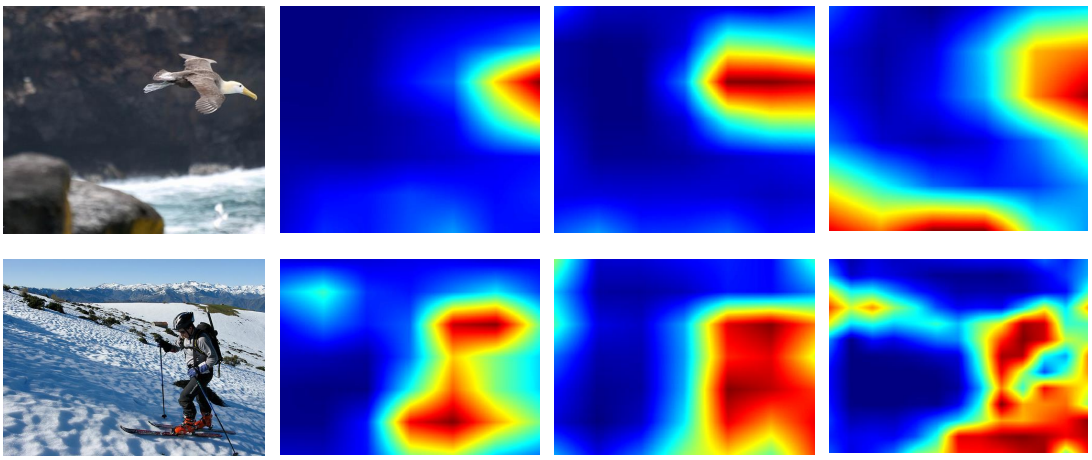
(a) Input (b) Ours (c) Mask (d) Grad-CAM (e) Grad (f) SmoothGrad (g) GuidedBP (h) Integrated

Figure 2.2: Visualization saliency maps comparing with 6 state-of-the-art methods.



(a) Input (b) Inversion map (c) "*elephant*" (d) "*zebra*"

Figure 2.3: Class discriminability of our algorithm. The inversion result (b) highlights all the foreground objects, while the final interpretation (c) and (d) highlight only the target object.



(a) Input (b) VGG-19 (c) ResNet-18 (d) AlexNet

Figure 2.4: Interpretation results for three CNN architectures.

crucial to classify zebra. It is consistent with our human cognition since we also rely on the head shape of the elephant and the black-and-white striped coats of zebra to classify them. Thus this interpretation is able to build trust with end users [29].

Interpretation Results Under Different CNN Architectures. Besides VGG-19 [12], we also provide explanation for two different network architectures, including AlexNet [11] and ResNet-18 [10]. For AlexNet, the *pool5* layer is utilized for both the inversion layer l_0 as well as the base channel layer l_1, the representation size of which is $(6 \times 6 \times 256)$. As for ResNet-18, the inversion layer l_0 utilizes *pool5*, and the base channel layer l_1 utilizes the next to last *conv layer*, both have size of $(7 \times 7 \times 512)$. For the rest of parameters, we utilize the same configuration as VGG-19.

The interpretation visualizations of three CNN architectures are shown in Fig. 2.4. For both the two inputs, the saliency maps generated by VGG-19 and ResNet-18 give the accurate location, while the ones yielded by AlexNet also highlight part of the background. One possible reason is that the AlexNet has only half number of channels at layer l_1 compared to the other two architectures. Besides, the smaller kernel filters as well as the increasing number of layers enable VGG-19 and ResNet-18 to learn more complex features and also lead to higher localization accuracy than AlexNet. It demonstrates that our model can be applied to a wide range of network architectures, including the neural network with skip-layer connections, and without fully connected layers.

	Grad	GuidedBP	LRP	CAM	Mask	Real	Ours
α	5.0	4.5	1.0	1.0	0.5	-	1.1
Error(%)	41.7	42.0	57.8	48.1	43.2	36.9	38.2

Table 2.1: Localization errors, and the optimal α values on ImageNet validation set of comparing methods. Error rate of comparing methods are taken from [1].

2.3.2 Quantitative Evaluation via Weakly Supervised Object Localization

In this section, we evaluate the localization performance of our interpretation method by applying the generated saliency maps to weakly supervised object localization tasks. The experiments are performed on the ImageNet validation set, which contains 50,000 images with bounding box annotations. Similar to [1, 30] as well as ILSVRC2014 [26] setting, 1762 images are excluded from the evaluation task because of their pool quality of annotations.

The saliency maps are binarized using mean thresholding by $\alpha \cdot m_I$, where m_I is the mean intensity of the saliency map and $\alpha \in [0.0 : 0.5 : 10.0]$, using the same setting with [1]. The tightest rectangle enclosing the whole segmented saliency map is counted as the final bounding box. The IOU (intersection over union) metric is utilized to measure the localization performance of each input instance. The localization is considered to be successful if the IOU score for an instance exceeds 0.5, otherwise it is treated as an error. The weakly supervised error is judged by the average localization error over the ImageNet validation set. For each comparing method, the α value is tuned using 1000 images selected from the ILSVRC2014 training dataset.

The object localization performance of the proposed method is compared with those of six state-of-the-art methods, including Grad [16], GuidedBP [18], LRP [31], Mask [1], CAM [32], and Real-time saliency [15]. The localization error values, as well as the optimal α values on Imagenet validation set are listed on Tab. 2.1. It shows our error is slightly higher than Real-time saliency [15] and outperforms all other 5 methods. Note that Real-time saliency[15] utilizes a U-Net [33] architecture which contains encoder and decoder network as mask, and parameters of the masking model is trained over a dataset. The large model size and using a whole dataset for training enables it to achieve relatively higher localization performance.

	Center	Grad	Grad-CAM	Ours
Accuracy (%)	0.483	0.531	0.550	0.547

Table 2.2: Pointing game accuracy on PASCAL VOC07 [2].

2.3.3 Pointing Game

In this section, we evaluate the class discriminability of our method by conducting the *Pointing game* experiment [1, 30]. The maximum point is first extracted from each generated saliency map, then according to whether the maximum point falls in one of the ground truth bounding boxes or not, a hit or a miss is counted. The pointing game localization accuracy for each object category is defined as: $\text{Acc} = \frac{\#\text{Hits}}{\#\text{Hits}+\#\text{Misses}}$. This process is repeated for all categories and the results are averaged as the final accuracy. The accuracy is evaluated over PASCAL VOC07 [2] test set, which contains 4952 images with multi-label bounding box ground truth. To obtain the classifier for this multi-label classification task, we replace the last fully connected layer of VGG-19 with a new layer which contains 20 output neurons, and then fine-tune the pre-trained VGG-19 model using the training set of VOC07. Following the standard multi-label classification setting, we utilize the Binary Cross Entropy between the target ground truth vector $\mathbf{l}$ and the Sigmoid soft label vector $\mathbf{y}$ as loss function: $l(\mathbf{l}, \mathbf{y}) = -[\mathbf{l} \cdot \log(\mathbf{y}) + (1 - \mathbf{l}) \cdot \log(1 - \mathbf{y})]$. Adam optimizer [27] is utilized to fine-tune the model, with a learning rate of 0.0001. The batch size is set to 64. Only the parameter of the last layer is tuned, and all the other parameters are left unchanged.

We compare the Pointing game performance with Grad [16], Grad-CAM [28], and a baseline method Center which utilizes the center of the image as the maximum point. To obtain the interpretation result, we employ the same empirical setting as in previous multi-classification setting, except that the parameter δ in Eq.(2.8) is set to 10, which works well on this multi-label dataset. Tab. 2.2 shows that our approach outperforms the center baseline and Grad, and achieves comparable performance with Grad-CAM. Taking into account that most real-life images contain more than one dominative class object in the foreground, the high class-discriminability of the proposed interpretation algorithm is thus an advantage to gain user trust.

3. Towards Explanation of RNN-based Prediction with Additive Decomposition

We have introduced how to provide post-hoc explanation for pre-trained CNN models in previous chapter. Different from CNNs, the RNNs (Recurrent neural networks), such as LSTM [34] and GRU [35], process the inputs in an recurrent way. RNN models have became increasingly deployed in different text classification applications, including sentiment classification [36], named entities recognition [37], textual entailment [38], etc. In this chapter, we introduce how to provide post-hoc explanation for pre-trained RNN models.[1]

3.1 Introduction

Despite the superior performance, RNNs are often criticized by their lack of interpretability, and are often treated as black-boxes [39]. It is highly desirable to explore the interpretability of RNNs, so as to provide insight of how they process text inputs and make inferences therefrom.

Explanation for RNN predictions is still a technically challenging problem. First, one challenge lies in how to guarantee that the interpretations are indeed faithful to the original model. Many previous attribution work, including back-propagation based methods [40, 41], perturbation based methods [42, 43], and local approximation based methods [13, 44], all follow the philosophy of *local interpretation*. That is, they generate interpretable approximation of the original model around the neighborhood of a given prediction to be explained. However, it is not guaranteed that the generated interpretations accurately reflect the decision making process of the original model [45, 46]. Second, it is challenging to develop a flexible attribution method which could generate attribution scores to text segments (e.g., phrases) of varying lengths. Prior work for RNNs attribution mainly focuses on identifying *word-level contribution scores* [13, 17, 41], which assigns a real-value score for each of the words, indicating the extent to which it contributes to a particular prediction. However, word-level attributions fail to explain why RNNs are successful to process sequences where the

[1] Reprinted with permission from "On attribution of recurrent neural network predictions via additive decomposition." by Mengnan Du, Ninghao Liu, Fan Yang, Shuiwang Ji, and Xia Hu. In The World Wide Web Conference, pp. 383-393. 2019. , Copyright 2019 by ACM.

order of the data entries matters. Consider an example of sentence sentiment analysis, "I do not dislike cabin cruisers." expresses a neutral opinion. Word-level attribution methods may accurately identify that the word "dislike" has a negative contribution for this prediction, but they fail to capture that the negation word "not" has shifted its polarity and the word combinations "not dislike" have a positive impact for model prediction.

In this chapter, we propose a decomposition based attribution method, called REAT (REcurrent ATtribution Method), to provide interpretation for given predictions made by a RNN in a faithful and flexible manner. Through modeling the information flowing process of the hidden state vectors in RNN models, REAT could decompose the final prediction of a RNN into additive contribution of each word in the input text. Since REAT is constructed by directly leveraging the information propagation process from hidden state vectors to the output layer, it enjoys the benefit of high faithfulness to the original RNN model. This method not only can quantify the contribution of each individual word to a prediction, but also could naturally be applied to identify the contribution of word sequences. It thus enables the illumination of how RNNs make use of sequential information, as well as how they capture long-term dependencies. In addition, REAT is widely applicable to different recurrent architectures, including LSTMs and GRUs, and their bidirectional versions.

3.2 Preliminaries

3.2.1 Post-hoc Attribution

Notations: Consider a typical multi-class text classification task, a RNN-based classification model can be denoted as $f : X \to Y$, where X is the text space, and $Y = \{1, ..., C\}$ denotes the set of output classes. The RNN model accepts an instance $\mathbf{x} \in X$ as input, and maps it to an output class: $f(\mathbf{x}) = c \in Y$. Assume the input is composed of a sequence of T words: $\mathbf{x} = \{x_1, ..., x_T\}$ and each word $x_t \in \mathbb{R}^d$ denotes the embedding representation of the t-th word. The high level idea of post-hoc attribution is to attribute the prediction $f(\mathbf{x})$ of a RNN model to its input features $\mathbf{x}$ and output a heatmap indicating the contribution of each feature $x_t \in \mathbf{x}$ to a particular class of interest c. Specifically, we target to generate attributions for a RNN prediction which is specified as follows:

Phrase-level Attribution: We first partition the input text into meaningful phrases and then attach an attribution score to each individual phrase. Given one index phrase as query. The index set of a phrase that we want to calculate its attribution score is denoted as: $A = \{q, ...r\}$ where $1 \leq q \leq r \leq T$. We indicate the attribution score for the targeting phrase as $S(\mathbf{x}_A)$.

3.2.2 RNN Architectures

RNNs come in many variants with different architectures, which results in different mapping functions f. We discuss three representative RNN architectures that are fundamental and have been widely used in many applications. The common formulations in different RNN architectures motivate the design of our interpretation approach.

LSTM: In a Vanilla RNN, at any time step in a sequence, the hidden state h_t is calculated based on its previous hidden state h_{t-1} and the current input vector x_t: $h_t = \text{tanh}(W_{hx}x_t + W_{hh}h_{t-1} + b_h)$. Comparing to the Vanilla RNN, LSTM makes two major changes. Firstly, LSTM introduces a cell state c_t that serves as an explicit memory. Secondly, instead of simply updating the hidden state h_t, LSTM uses three gates: input gate i_t, forget gate f_t, and output gate o_t to update the hidden state h_t.

$$
\begin{aligned}
i_t &= \sigma(W_{ix}x_t + W_{ih}h_{t-1} + b_i) \\
f_t &= \sigma(W_{fx}x_t + W_{fh}h_{t-1} + b_f) \\
o_t &= \sigma(W_{ox}x_t + W_{oh}h_{t-1} + b_o) \\
g_t &= \text{tanh}(W_{gx}x_t + W_{gh}h_{t-1} + b_g) \\
c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\
h_t &= o_t \odot \text{tanh}(c_t).
\end{aligned}
\tag{3.1}
$$

where each W and b represent weight matrix and bias vector respectively, and $\odot$ denotes element-wise multiplication.

GRU: GRU makes some slight modifications on the basis of LSTM. It only has two gates, i.e., reset gate r_t and update gate u_t. Besides, it merges the cell state and hidden state into a single hidden

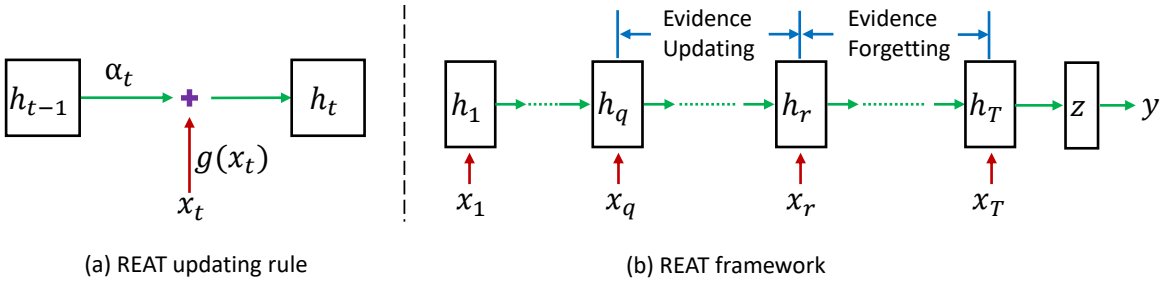


Figure 3.1: (a) The REAT updating rule. (b) An illustration of the proposed REAT framework.

state h_t. The updating rule of hidden state is denoted as follows.

$$
\begin{aligned}
r_t &= \sigma(W_{rx}x_t + W_{rh}h_{t-1} + b_r) \\
u_t &= \sigma(W_{ux}x_t + W_{uh}h_{t-1} + b_u) \\
g_t &= \tanh(W_{gx}x_t + r_t \odot W_{gh}h_{t-1} + b_g) \\
h_t &= u_t \odot h_{t-1} + (1 - u_t) \odot g_t.
\end{aligned}
\tag{3.2}
$$

Bidirectional GRU: This model is constructed by putting two independent GRUs together. The word sequences are fed into one GRU in normal time order, and in reverse time order into another. For each network, the hidden state is updated using the same rule as Eq. (3.2). For the sake of brevity, we use subscripts n and r to represent the $normal$ and $reverse$ network respectively. For classification tasks, the final hidden vector fed into the output layer is constructed by concatenating the hidden vector at time step T for the normal GRU and the hidden vector at time step 1 for the reverse one: $h = h_{T,n} \oplus h_{1,r}$, where symbol $\oplus$ denotes concatenation operation of two vectors. In the remaining part of this paper, we use BiGRU to denote Bidirectional GRU.

3.2.3 RNN Output Layer

To serve the purpose of multi-class text classification, a discriminative layer is added after the activation vector h_T of the last hidden layer at time step T. This layer takes the hidden state h_T as input and turns it into a logit vector z using a weight matrix W: $z = Wh_T$, and then produces the class probabilities using a softmax layer which converts the logit z_c for a class c into a probability

y_c, by comparing it with other logit values: $y_c = \text{softmax}(z)_c = \frac{\exp(z_c)}{\sum_{k=1}^{C} \exp(z_k)}$.

3.3 Methodology

In this section, we introduce the proposed attribution method for explaining RNN predictions. We first present a general method for phrase-level RNN prediction attribution, and then apply it to three widely used RNN architectures.

3.3.1 A Naive Attribution Approach

From the last section, we know that RNNs possess a series of hidden state vectors $\{h_t\}_{t=1,\ldots,T}$, where each vector h_t stores information about the past input blocks, ranging from time step 1 to t. A crucial property of the hidden state vector is that it is updated from time step to time step. Knowing how much information is accumulated at each time step enables us to derive the contribution of that time step towards the final prediction. Intuitively, we denote the response of the RNN model to word $\mathbf{x}_t$, and thereby the information gained at step t is: $\tilde{h}_t = g(\mathbf{x}_t) = h_t - h_{t-1}$. In this way, we can consider the final hidden state vector h_T to contain information accumulated at T time steps, denoted as: $h_T = \sum_{t=1}^{T} \tilde{h}_t = \sum_{t=1}^{T} (h_t - h_{t-1})$. Then, we can decompose the logit z_c for target class c using a sequence of factors:

$$z_c = W_c h_T = \sum_{t=1}^{T} W_c (h_t - h_{t-1}). \tag{3.3}$$

It can be treated as the additive contribution of each word in the input $\mathbf{x}$ to the output logit (unnormalized output probability) of class c. Therefore, the contribution of word $\mathbf{x}_t$ towards the logit z_c can be calculated as:

$$S(\mathbf{x}_t) = W_c (h_t - h_{t-1}). \tag{3.4}$$

However, this decomposition has a severe shortcoming. The underlying assumption of this formulation is that all evidence accumulated up to time step $t-1$ has been transferred to time t. This violates the updating rules of all the three architectures listed in Sec. 3.2.2. Actually, it fails to take into consideration the forgetting mechanism of RNNs. For example, both LSTM and GRU have

explicit gates to model the forgetting and remembering mechanism, which serves the purpose of controlling information flow and calculating how much proportion of information derived from previous time steps should be kept.

3.3.2 The Proposed Recurrent Attribution Method (REAT)

Many variants of RNNs share a similar form of hidden state updates. We summarize a common rule which can be applied to different recurrent network architectures, as illustrated in Fig. 3.1 (a). The rule maintains a hidden state h_t which summarizes information for past sequences. Also, at each time step t, the hidden state is updated using the equation:

$$h_t = \alpha_t \odot h_{t-1} + \tilde{h}_t, \tag{3.5}$$

where $\alpha_t \in [0,1]^{d'}$ so that only partial evidence obtained by RNN from previous $t-1$ steps is brought to the time step t. Here d' is the dimension of hidden state vectors. A higher value of α means that the RNN model preserves more important information from previous time steps. Here $\tilde{h}_t = g(\mathbf{x}_t)$ denotes the evidence that a RNN obtains at time step t. Some RNN architectures obey this rule exactly, like GRU, while some other architectures follow this rule approximately, such as LSTM. Based on this hidden state updating rule, we can iteratively trace back the generation of h_T and decompose the logit value z_c into the following form:

$$z_c = W_c h_T = \sum_{t=1}^{T} W_c(\tilde{h}_t \odot \prod_{k=t+1}^{T} \alpha_k). \tag{3.6}$$

Replacing each $\tilde{h}_t$ with $h_t - \alpha_t \odot h_{t-1}$, we can reformulate the additive decomposition as:

$$z_c = \sum_{t=1}^{T} W_c[(h_t - \alpha_t \odot h_{t-1}) \odot \prod_{k=t+1}^{T} \alpha_k]. \tag{3.7}$$

The main benefit of Eq. (3.7) comparing to Eq. (3.6) is that we do not need to know the exact form of $\tilde{h}_t$. Merely knowing the hidden state vector h_t and the updating parameter vector α_t will be

sufficient to derive the decomposition. Eq. (3.7) can be considered as the additive contribution of each word $\mathbf{x}_t$ towards the logit z_c, which is the unnormalized probability for target class c. By taking out the term relevant to time step t, we can derive the contribution value for a single word $\mathbf{x}_t$ at the current time step:

$$S(\mathbf{x}_t) = W_c[\underbrace{(h_t - \alpha_t \odot h_{t-1})}_{Updating} \odot \underbrace{\prod_{k=t+1}^{T} \alpha_k}_{Forgetting}]. \tag{3.8}$$

The above formulation within the square brackets is the element-wise multiplication of two terms. The left term denotes the updating evidence from time $t-1$ to t, i.e., the contribution to class c by the input word $\mathbf{x}_t$. The right term represents the forgetting mechanism of RNN. The evidence that a RNN has gathered at time step t gradually diminishes as the time increases from $t+1$ to the final time T. Only part of the updating evidence will have impacts on the classification task at time T.

Based on the word-level additive attribution formulation in Eq. (3.8), we can conveniently derive the phrase-level attribution. For a phrase $\mathbf{x}_A$, where $A = \{q, ...r\}$, $1 \leq q \leq r \leq T$, its attribution score $S(\mathbf{x}_A)$ can be denoted as:

$$S(\mathbf{x}_A) = W_c[\underbrace{(h_r - \prod_{j=q}^{r} \alpha_j \odot h_{q-1})}_{Updating} \underbrace{\prod_{k=r+1}^{T} \alpha_k}_{Forgetting}]. \tag{3.9}$$

Similar to word-level attribution, phrase-level attribution $S(\mathbf{x}_A)$ also contains two terms. The left term within the square brackets in Eq. (3.9) represents the updating evidence from time step $q-1$ to time r, while the right term denotes how much percentage of the evidence has been forgotten from time $r+1$ to T. We illustrate this process in as illustrated in Fig. 3.1 (b). It is worth noting that the key component here to derive phrase-level attribution score for a RNN classifier is to obtain the hidden state vectors and the updating vectors. Usually, only one feed forward operation is needed to derive phrase-level attribution score, which can be implemented efficiently.

3.3.3 Applications to Specific Architectures

GRU Attribution: The hidden state vector updating rule for GRU is written as: $h_t = u_t \odot h_{t-1} + (1 - u_t) \odot g_t$, which conforms with the paradigm in Eq. (3.5). Therefore, for a phrase $\mathbf{x}_A$, where $A = \{q, ...r\}$, $1 \leq q \leq r \leq T$, we can directly replace α_t in Eq. (3.9) with the updating gate vector u_t of the GRU model, and obtain the phrase-level attribution score $S(\mathbf{x}_A)$:

$$S(\mathbf{x}_A) = W_c[(h_r - \prod_{j=q}^{r} u_j \odot h_{q-1}) \odot \prod_{k=r+1}^{T} u_k]. \tag{3.10}$$

LSTM Attribution: Although it is difficult to directly match the LSTM updating rule of hidden state h_t in Eq. (3.1) to the paradigm in Eq. (3.5), the update of the cell state c_t adheres to the expected updating format in Eq. (3.5). It is denoted as: $c_t = f_t \odot c_{t-1} + i_t \odot g_t$, where f_t is the forgetting gate of LSTM. Based on the updating rule from the cell state c_t to hidden state: $h_t = o_t \odot tanh(c_t)$, approximately we can obtain: $c_t \sim \frac{h_t}{o_t}$, where the right term is a element-wise division. Thus we approximate the updating rule of the hidden state vector for LSTM as below:

$$h_t = \frac{f_t \odot o_t}{o_{t-1}} \odot h_{t-1} + \tilde{h}_t. \tag{3.11}$$

Then, we can further decompose the final hidden state vector h_T into the following formulation:

$$h_T = \sum_{t=1}^{T} (h_t - \frac{f_t \odot o_t}{o_{t-1}} \odot h_{t-1})) \odot \prod_{k=t+1}^{T} \frac{f_k \odot o_k}{o_{k-1}}. \tag{3.12}$$

For a phrase $\mathbf{x}_A$, where $A = \{q, ...r\}$, $1 \leq q \leq r \leq T$, we can obtain the attribution score $S(\mathbf{x}_A)$:

$$S(\mathbf{x}_A) = W_c[(h_r - \prod_{j=q}^{r} \frac{f_j \odot o_j}{o_{j-1}} \odot h_{q-1}) \odot \prod_{k=r+1}^{T} \frac{f_k \odot o_k}{o_{k-1}}]. \tag{3.13}$$

BiGRU Attribution: The last hidden state $h_{T,n}$ at time step T of the normal GRU uses the identical decomposition as in GRU. As for the reverse GRU, we use the hidden state vector at time step 1 in

order to capture the information from 1 to T, which can be decomposed as follows:

$$h_{1,r} = \sum_{t=1}^{T} (h_{t,r} - u_{t,r} \odot h_{t+1,r}) \odot \prod_{k=1}^{t-1} u_{k,r}. \tag{3.14}$$

Recall that the final hidden vector fed into the classification layer is the concatenation of the hidden vector at time step T for the normal GRU and the hidden vector at time step 1 for the reverse GRU, i.e., $h = h_{T,n} \oplus h_{1,r}$. As such, the logit value is computed as $z_c = W_c(h_{T,n} \oplus h_{1,r})$. To obtain the attribution score for a phrase $\mathbf{x}_A$, where $A = \{q, ...r\}$, we first calculate the updated hidden evidence for the normal network and the reverse network respectively. We then concatenate these two updated hidden evidence, and multiply it with W_c to produce the contribution scores.

$$\begin{aligned} S(\mathbf{x}_A) =& W_c[((h_{r,n} - \prod_{j=q}^{r} u_{j,n} \odot h_{q-1,n}) \odot \prod_{k=r+1}^{T} u_{k,n}) \oplus \\ & ((h_{q,r} - \prod_{j=q}^{r} u_{j,r} \odot h_{r+1,r}) \odot \prod_{k=1}^{q-1} u_{k,r})]. \end{aligned} \tag{3.15}$$

3.4 Experiments

We conduct experiments to evaluate the effectiveness of the proposed RNN interpretation method. Specifically, we have applied REAT to various RNN architectures, including GRU, LSTM and their bidirectional versions.

3.4.1 Datasets

We conduct our experiments on two publicly available sentiment analysis datasets.

Stanford Sentiment Treebank 2 (SST2) [47] - It contains 2 classes (negative and positive). The numbers of instances for training, development and test set are 6920, 872, and 1821 respectively.

Yelp Polarity (Yelp) [48] - It consists of reviews originally extracted from the Yelp Dataset Challenge 2015 data. Zhang et al. [48] constructed the Yelp reviews polarity dataset by considering stars 1 and 2 as negative, and considering 4 and 5 as positive. The numbers of instances for training set and test set are 560,000 and 38,000 respectively. We use a subset of this dataset by filtering out texts

whose length is larger than 40, and then randomly select part of the samples from the training set as development set. Ultimately, the dataset contains 80000, 1999, and 5492 instances for training, development, and test set respectively, with each polarity occupying around half of the instances.

3.4.2 Experimental Setup

For each classification model used in our experiments, it contains a word embedding layer to transform words to fixed length representation vectors, a recurrent network layer to transform word embeddings to hidden state vectors, and a classification layer for output. Specifically, the pre-trained 300-dimensional word2vec word embedding [49] is utilized to initialize the embedding layer. For those words that do not exist in word2vec, we initialize their embedding vectors with some random values. The dimension of hidden state vectors is 150 for both LSTM and GRU, and 300 for BiGRU. The classification layer is composed of a dense layer and a softmax nonlinear transformation. The Adam optimizer [27] is utilized to optimize these models and the learning rate is fixed to 10^{-3}. We train each model for 20 epoches and select the one with the best performance on the development set. Note that we freeze the embedding layer when training all models on SST2, while fine-tune the embedding parameters when training on Yelp. Empirical results show that this can lead to better prediction performance for models on both datasets.

3.4.3 Baseline Methods

We evaluate the proposed REAT method by comparing it with five baseline approaches.

- **Vanilla gradient (VanillaGrad)** [40]: Compute gradients of the output prediction with respect to individual entries in word embedding vectors, and use the L2 norm to reduce each vector of the gradients to a single attribution value, representing the contribution of each single word.

- **Integrated gradient (InteGrad)** [17]: Integrate all Vanilla gradients using a linear interpolation between a baseline input and the original input. Here the baseline input are sentences whose word embedding values are all set to zeros.

- **Gradient times input (GradInput)** [41]: First calculate the gradient of the output with respect

to word embedding, and then use dot product of the gradient vector and word embedding vector as the contribution score for a word.

- **LIME** [13]: A model-agnostic interpretation method. It approximates the behavior of a RNN in the neighborhood of a given input using an interpretable white-box model. Here, the interpretable model is a sparse linear model.

- **NaiveREAT**: A simplified variant of the proposed method, as introduced in Sec. 3.3.1. The attribution score for a single word is calculated using Eq. (3.4).

It is worth noting that except NaiveREAT, the other four baseline methods could only derive word-level contribution scores. To get phrase-level or sentence-level contribution scores, we sum up the scores of all word within a phrase or a sentence.

3.4.4 Attribution Faithfulness Evaluation

We evaluate the faithfulness of the attribution methods with respect to the target RNN models. We want to assess whether the attribution results correctly reflect the prediction behavior of RNNs. In general, the faithfulness of an attribution method is evaluated by deleting the sentence of the highest contribution score and observing the prediction changes of the target RNNs [50]. Specifically, the attribution method first produces contribution scores for sentences in the text. Then, it is expected that once the most important sentence is deleted, it will cause the probability value to significantly drop for the target class. Here we define faithfulness score as the metric:

$$S_{\text{faithfulness}} = \frac{1}{N}\sum_{i=1}^{N}(y(\mathbf{x}^{(i)}) - y(\mathbf{x}^{(i)}_{\setminus A})), \tag{3.16}$$

where A denotes the sentence identified as the most predictive for a prediction, and N is the total number of texts in the dataset. An advantage of this metric is that no knowledge is required of ground truth labels. Theoretically this metric can also be utilized to evaluate word-level and phrase-level attributions. However, empirically, we find some irrelevant words or phrases could also lead to big probability drop, because they cause grammar or syntactic errors instead of really changing the

Models	GRU	LSTM	BiGRU
VanillaGrad	0.272	0.243	0.068
InteGrad	0.255	0.253	0.113
GradInput	0.301	0.199	0.178
Lime	0.209	0.188	0.092
NaiveREAT	0.213	0.207	0.114
REAT	**0.311**	**0.318**	**0.196**

Table 3.1: Comparison about attribution faithfulness between our method and the baseline methods.

semantics [51, 52]. Therefore, we only use this metric in sentence-level attribution scenarios.

We search in SST2 dataset for texts that contain two sentences and use the word "but" as conjunction, and obtain a set with 142 texts. The faithfulness scores for different attribution methods on SST2 dataset are reported in Tab. 3.1. The proposed method consistently outperforms the baseline methods for all the three architectures. This result demonstrates two advantages of REAT: (1) the generated interpretations are highly faithful to original RNN, (2) the generated phrase-level interpretations are accurate. Since the NaiveREAT method does not consider the forgetting mechanism, it may assign false positive contribution scores for the first sentence in the testing texts. As a result, it is not faithful to the target model and achieves relatively low faithfulness score comparing to REAT. Besides, for the other baseline methods, they can only output word-level attribution scores. Since the word-level scores are not sufficiently faithful to the target model, the sentence-level scores calculated by summing up the word-level scores will further deviate from the prediction of the RNN model. As a result, these methods have only limited faithfulness performance.

3.4.5 Attribution Interpretability Evaluation

We evaluate the interpretability of the proposed method to show whether the generated interpretations are reasonable from some fundamental perspectives of human comprehension. We only search for texts that contain both positive adjective words and negative adjective words with obvious sentiment bias. The searching criterion is whether a word belongs to the human annotated lists containing both positive words[2] and negative words[3] [53]. We obtain a testing set with 78 and 99

[2]https://gist.github.com/mkulakowski2/4289437
[3]https://gist.github.com/mkulakowski2/4289441

	SST2			Yelp		
Models	GRU	LSTM	BiGRU	GRU	LSTM	BiGRU
VanillaGrad	41.0	42.3	26.9	57.6	44.4	38.4
InteGrad	44.9	52.6	42.3	58.6	42.4	33.3
GradInput	84.6	80.8	85.9	84.8	82.8	90.9
Lime	73.1	80.8	73.1	74.7	77.8	80.8
NaiveREAT	79.5	87.2	**92.3**	55.6	83.8	86.9
REAT	**85.9**	**89.7**	88.5	**87.9**	**83.8**	**92.9**

Table 3.2: Interpretability statistical comparison (in percent) of our method with baseline RNN attribution methods.

samples from SST2 and Yelp respectively. Then we generate a attribution score after feeding each text sample to a RNN, and evaluate the consistency of attribution with human annotations. Here, we focus on analyzing the attribution scores of these words for the positive sentiment side for all testing texts. The interpretation is considered to be a match if the attribution scores for positive words are lager than negative words, otherwise it is treated as a mismatch. Take text "It is ridiculous, of course but it is also refreshing" for example, if the attribution method assigns a higher contribution score to positive word "refreshing" comparing to negative word "ridiculous", we consider it as a match. The final interpretability score is judged by the ratio of matched cases:

$$S_{\text{interpretability}} = \frac{\#\text{match}}{\#\text{match} + \#\text{mismatch}}. \tag{3.17}$$

We compare the interpretability score of the proposed method with baseline methods on three RNN architectures over SST2 and Yelp dataset. The results are presented in Tab. 3.2. The proposed method ranks highest for five tasks among all six classification tasks. We observe that the interpretability scores for the NaiveREAT method, which is introduced in Sec. 3.3.1, are unstable across different models and datasets, partly due to the reason that it is not faithful to the original recurrent model. Sometimes, NaiveREAT could accurately capture the contribution score for strong sentiment word, such as for BiGRU prediction on SST2. In other cases, it may assign some false

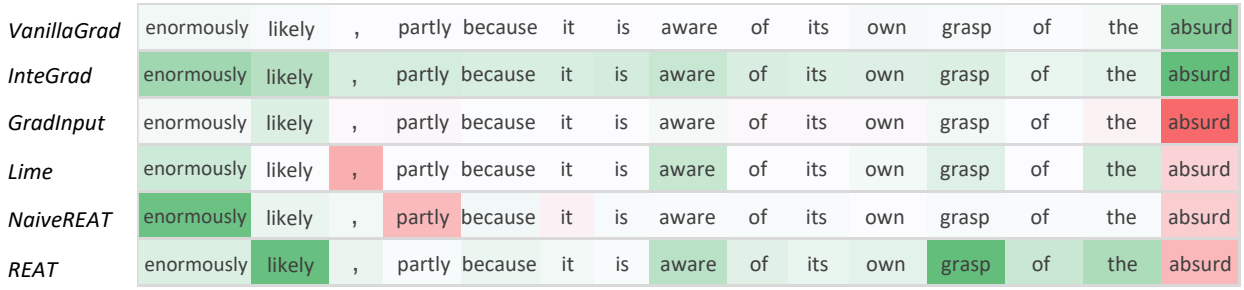


Figure 3.2: Word-level attribution heatmaps comparing with baseline methods, for a GRU prediction with 99.2% confidence as positive sentiment. Green and red color denote positive and negative contribution of a word to the prediction, respectively.

positive and false negative attribution values, leading to lower interpretability scores. Another finding is that the interpretability of InteGrad does not outperform VanillaGrad consistently on all cases. Besides, GradInput yields consistently better performance comparing to VanillaGrad.

3.4.6 Qualitative Evaluation via Case Studies

We provide several case studies to qualitatively check the effectiveness of the proposed method. We use green color to denote positive contribution and red color for negative contribution. Deeper color means higher contribution to the prediction. We present attribution visualization for a prediction made by GRU with 99.2% confidence for positive sentiment, and compare it with the baseline attribution methods. The results are shown in Fig. 3.2. The heatmap shows that REAT not only identifies that words “enormously”, “likely”, “aware”, “grasp” have positive contribution for the prediction, but also captures that “absurd” has negative contribution. It is consistent with human comprehension towards this sentence. In contrast, VanillaGrad and InteGrad fail to attribute the negative word; GradInput fails to identify the words that strongly and positively contribute to the prediction, thus can not explain why GRU gives such a high positive prediction score to this text. Also, LIME and NaiveREAT generate some noisy negative scores for irrelevant words, such as “partly”, “it” and “,”.

We compare the word-level attribution results of three RNN architectures. For a given text “The fight scenes are fun but it grows tedious”, GRU, LSTM and BiGRU give positive prediction (51.6%

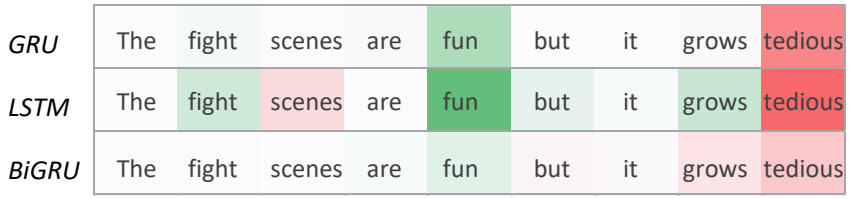


Figure 3.3: Word-level attribution heatmaps for 3 RNN architectures. Green and red color denote positive and negative contribution of a word to the prediction, respectively.

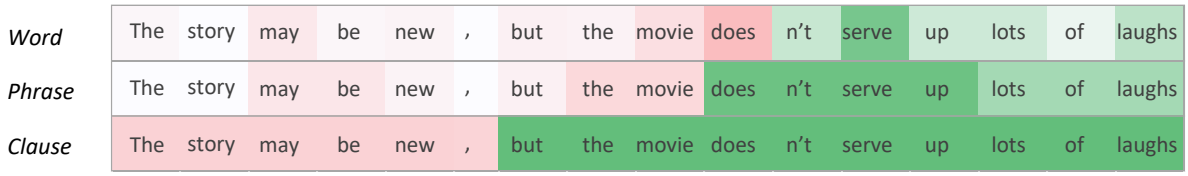


Figure 3.4: Visualization heatmaps for hierarchical attribution. Green and red color denote positive and negative contribution of a word to the prediction, respectively.

confidence), positive prediction (96.2% confidence), and negative prediction (62.7% confidence), respectively. We display the attribution heatmaps for positive prediction for all three architectures in Fig. 3.3. GRU gives nearly the same absolute value of attribution score for "fun" and "tedious". LSTM attributes more words positive contributions than negative words, while BiGRU gives more words negative contribution scores. These attribution heatmaps well reflect the prediction scores, which indicates that the interpretations could give users understandable rationale for the predictions.

We also give the visualization heatmaps of hierarchical attributions in Fig. 3.4. In this case, we show the attribution scores of a negative sentiment prediction with 99.46% confidence from a LSTM model. For the word-level attribution, "does" has a negative contribution for the prediction, while the combination "does n't serve up" will have a strong positive contribution for the prediction. The clause-level attribution shows that the first clause has a relatively small negative contribution for prediction, while the second clause has a strong positive contribution for prediction. This hierarchical attention thus could represent the contributions at different levels of granularity.

4. Explanation-Guided Generalizable DNNs for Text Classification

In previous two chapters, we introduce how to provide post-hoc explanation for pre-trained DNN models. In this chapter, we illustrate how to make use of explainability as a debugging tool to improve the generalization ability of DNN based text classification models, by incorporating explanations into the model training process. Specifically, we consider three representative DNN architectures for text classification, including CNN, LSTM, and Self-attention model[1].

4.1 Introduction

There has been an increasing interest recently in developing explainable deep neural networks (DNNs) [5, 54, 55, 56]. To this end, a DNN model should be able to provide intuitive explanations for its predictions. Explainability could shed light into the decision making process of DNNs and thus increase their acceptance by end-users. However, explainability alone is insufficient for DNNs to be *credible* [57], unless the provided explanations conform with the well-established domain knowledge. That is to say, correct evidences should be adopted by the networks to make predictions. The incredibility issue has been observed in various DNN systems. For instance, in question answering (QA) tasks, DNNs rely more on function words rather than pay attention to task-specific verbs, nouns and adjectives to make decisions [58, 59]. Similarly, in image classification, CNNs may make decisions solely according to background within images, rather than paying attention to evidences relevant to the objects of interest [13].

In this work, we define credible DNNs as the models that could provide explanations to their predictions, while at the same time the explanations are consistent with the well-established domain knowledge. Considering that correct evidences are employed in decision making process, it would be easier for credible DNNs to build up trust among practitioners and end-users. In addition, credible DNNs could have better generalization capability comparing to untrustable ones. Since credible

[1]Reprinted with permission from "Learning credible deep neural networks with rationale regularization." by Mengnan Du, Ninghao Liu, Fan Yang, and Xia Hu. In 2019 IEEE International Conference on Data Mining (ICDM), pp. 150-159. IEEE, 2019. Copyright 2019 by IEEE.

DNNs have truly grasped useful knowledge instead of memorizing unreliable *dataset-specific biases* and *artifacts*, they could maintain high prediction accuracy for those unseen data instances beyond the training dataset.

To overcome the above challenges, we propose to explore *rationale*, to enhance DNN credibility. A rationale is a subset of features highlighted by annotators and regarded to be more important in predicting an instance [60, 61]. The rationales are utilized to direct the model's attention, enabling it to tease apart useful evidence from noises and pushing it to pay more attention to relevant features. Based on the rationales from domain experts, we propose CREX (CRedible EXplanation), an approach regularizing DNNs to utilize correct evidences to make decisions, in order to promote their credibility and generalization capability. The intuition behind CREX is to use external knowledge to regulate the DNN training process. During the model training, we require the DNN model to generate local explanations that conform with the rationales.

4.2 Problem Statement

Notations: Consider a typical multi-class text classification task. Given a training dataset which consists of N instances: $D = \{(x_1, y_1), ...(x_N, y_N)\}$. Each input text x_n is composed of a sequence of T words: $x_n = \{x_n^{(1)}, ..., x_n^{(T)}\}$, where $x_n^{(t)} \in \mathbb{R}^d$ denotes the embedding representation of the t-th word. Each $y_n \in \{1, 2, ..., C\}$ belongs to one of the C output classes. Part of the training data, with a number of N_r, contains not only input-label pairs (x_n, y_n), but also rationale r_n from domain expert. Each entry of the expert rationale $r_n^{(t)} \in \{0, 1\}$, where 1 indicate that word $x_n^{(t)}$ is actually responsible for the prediction task, and vice versa.

Learning Credible DNNs: The goal is to learn a DNN-based classification model which maps a text input x_n to the probability output $f(x_n)$. We expect a trained DNN to rely on correct evidences to make decisions and pay more attention to words within the rationales. That is, for a trained DNN, the generated local explanation for each testing instance should align well with expert rationales.

4.3 Proposed CREX Framework

In this section, we introduce the CREX framework, which aims to regularize the local explanation when training a DNN for the task of interest, so as to promote its credibility and generalization.

4.3.1 Augmenting Local Explanation

The general idea of DNN local explanation is to attribute the prediction of a DNN to its input, producing a heatmap indicating the contribution of each feature in the input to the prediction. The explanation of prediction $f(x_n)$ for input x_n is a matrix $s_n \in \mathbb{R}^{T \times C}$, where $s_n^{(t,c)}$ denotes the contribution of word $x_n^{(t)}$ towards prediction $f_c(x_n)$ for output class c. We utilize an omission based method [42] to measure the contribution of $x_n^{(t)}$, denoted as below:

$$s_n^{(t,c)} = f_c(x_n) - f_c(x_n^{(\backslash t)}), \tag{4.1}$$

which quantifies the deviation of the prediction between the original input x_n and the partial input $x_n^{(\backslash t)} = x_n^{(1:t-1)} \oplus x_n^{(t+1:T)}$ with $x_n^{(t)}$ omitted. The motivation is that more important features, once being changed, will cause more significant variation to the prediction score. It is worth noting that the omission operation may lead to invalid input, which could trigger the adversarial side of DNNs. To reflect model behaviors under normal conditions, phrase omission is conducted instead of individual word omission. Formally, we compute the contribution of $x_n^{(t)}$ by averaging the prediction changes of deleting different length-m phrases that contain $x_n^{(t)}$:

$$s_n^{(t,c)} = \frac{1}{m} \sum_{j=1}^{m} [f_c(x_n) - f_c(x_n^{(1:t-1-m+j)} \oplus x_n^{(t+j:T)})]. \tag{4.2}$$

For long text classification, we segment each original text into sentences and sequentially perform omission for each sentence. In such scenario, sentence-level contribution scores are obtained as explanation, rather than word-level scores. Both phrase omission and sentence omission could increase the faithfulness of explanation, compared with directly removing individual words [43].

4.3.2 Aligning Explanations with Rationales

The key idea of CREX is that DNNs should rely on reasonable evidences to make decisions rather than bias or artifacts. We encourage the explanation to align well with expert rationales when they are available, by considering two complementary conditions as follows. First, for the original input, we encourage the generated explanation to be *confident* and focus on the relevant features as indicated by rationales. Second, for the negative input, where the important features are suppressed, the explanation should be *uncertain* and have relatively uniform contribution across classes.

4.3.2.1 Confident Explanation

We first feed original input x_n to DNN and get model output $f(x_n)$ and explanation s_n. The rationale r_n points out which subset of features is important and the rest to be irrelevant. Intuitively, we achieve credibility by encouraging dense contribution scores on known important factors and encouraging sparse contribution scores on the remaining irrelevant features. We define a *confident explanation* loss (g_{conf}), which encourages the explanation to concentrate on rationales:

$$g_{conf}(x_n) = \frac{1}{C}\sum_{c=1}^{C} ||(1 - r_n) \odot s_n^{(:,c)})||_1. \tag{4.3}$$

The loss aims to shrink the contribution scores of irrelevant features, in order to discourage models from capturing training data specific biases. An implicit effect of this loss is to encourage f to give dense explanation scores to the relevant features, thus making f pay more attention to them. As a result, the final explanation scores tend to align well with rationales. In addition, we observe that summing all categories $\{1, ...C\}$ could yield better results comparing to only using label y_n when imposing confident explanation regularization to instance x_n.

4.3.2.2 Uncertain Explanation

When the subset of important features, as indicated in r_n, is deleted in the original input x_n, we expect the DNN model to become uncertain about which category to output. This kind of inputs, named as negative inputs, are generated as the Hadamard product between the original input x_n

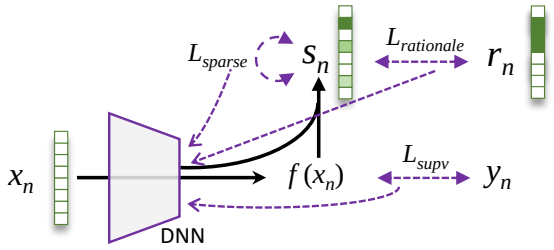


Figure 4.1: Schematic of CREX. Black solid lines denote forward pass. Dashed line with arrows on both ends are losses. Dashed line with arrows on one side denote flow of gradients. Three vectors from left to right are input, explanation and rationale, respectively. CREX is DNN architecture agnostic, end-to-end trainable, and simple to implement.

and the reversed rationale vector $(1 - r_n)$: $x'_n = x_n \odot (1 - r_n)$. The intuition is that after feeding the negative input x'_n to a DNN model, we expect its probability output for ground truth label y_n to be much smaller comparing to the probability value of original input x_n, since x'_n lacks the evidence supporting the prediction. At the same time, the contributions of different words/sentences should be distributed uniformly. Its implicit effect is to encourage the DNN model to give lower explanation scores to the features not belonging to rationales. We first calculate the absolute value of explanation for x'_n as $\hat{s}_n^{(:,y_n)} = |s_n^{(:,y_n)}|$, and then normalize it as: $e_n^{(t,y_n)} = \hat{s}_n^{(t,y_n)} / \sum_{k=1}^{T} \hat{s}_n^{(k,y_n)}$ The resultant $e_n^{(:,y_n)}$ can be seen as the soft-attention assigned by DNN for x'_n. After that, we define an *uncertain explanation* loss (g_{unc}):

$$g_{unc}(x'_n) = -|f_{y_n}(x_n) - f_{y_n}(x'_n)| - \alpha \cos(e_n^{(:,y_n)}, q), \tag{4.4}$$

where q is the discrete uniform distribution denoted as $\mathcal{U}(1, T)$, and α is used to balance probability output and explanation distribution. The cosine similarity is employed to encourage explanation scores to be distributed uniformly.

We linearly combine the two loss functions at hand, and calculate the average value over all

training instances with rationales as the *explanation rationale* loss, formulated as follows:

$$\mathcal{L}_{rationale} = \frac{1}{N_r} \sum_{n=1}^{N_r} [g_{conf}(x_n) + \beta g_{unc}(x'_n)]. \tag{4.5}$$

Parameter β is utilized to balance the confident explanation and uncertain explanation. By encouraging confident explanations to conform with rationales in original input x_n, and suppressing the probability output as well as explanation values in a negative input x'_n, $\mathcal{L}_{rationale}$ regulates a DNN to learn useful input representations from features belonging to rationales and omit information in the irrelevant feature subset.

4.3.3 Self-guidance When Rationale not Available

In last section, given expert rationales, we render the local explanation of each instance to conform with its rationale. However, expert rationales may not always be available. In practice, the experts may only annotate a small ratio of training data. This could be done either when annotating a new corpus, or when adding rationales post-hoc to an existing corpus. To guide the DNN model to focus on correct evidences in such scenario, we enforce the generated local explanation vector to be sparse for training instances without rationales. Simpler explanations are more credible, otherwise the dense dependencies could make it hard to disentangle the patterns in the input that actually trigger a prediction [39, 62, 63]. To achieve this, we propose the *sparse explanation* loss for those instances without rationales, denoted as follows:

$$\mathcal{L}_{sparse} = \frac{1}{(N - N_r) \cdot C} \sum_{n=N_r+1}^{N} \sum_{c=1}^{C} ||s_n^{(:,c)}||_1, \tag{4.6}$$

where the ℓ_1 norm helps produce sparse contribution vectors. Note that this summation is performed over the $(N\text{-}N_r)$ instances which have no rationales.

4.3.4 CREX Training

Besides regularizing the local explanations for DNN predictions, we also expect the DNN model to learn from the ground truth labels, which is defined using supervised cross-entropy loss function

as follows:

$$\mathcal{L}_{supv} = \frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} -\mathbb{1}(y_n = c) \cdot \log(f_c(x_n)). \tag{4.7}$$

Our final model is learned by balancing the supervised approximation to the labels and the conformation to expert rationales. We propose the training objective by jointly minimizing the losses as below:

$$\mathcal{L}(\theta, x, y, r) = \mathcal{L}_{supv} + \lambda_1 \mathcal{L}_{rationale} + \lambda_2 \mathcal{L}_{sparse}. \tag{4.8}$$

Parameters λ_1 and λ_2 are utilized to balance the supervised loss, rationale loss and sparse loss. For those N_r inputs coupled with expert rationales, we impose rationale loss, while for the rest $N - N_r$ inputs we regularize them with sparse loss. The overall idea of CREX is illustrated in Fig. 4.1. Our framework is designed to train the DNN model which could make highly accurate predictions (the first term) as well as make decisions by relying on the correct evidences (the last two terms). In addition, CREX framework can be treated as knowledge distillation process that transfers expert knowledge from rationales to DNN parameters in order to yield more credible models.

4.4 Experiments

In this section, we evaluate the proposed CREX framework on several real-world datasets.

4.4.1 DNN Architectures

We consider three representative DNN architectures for text classification, including CNN [64], LSTM [34], and Self-attention model [65].

CNN: This is a 2-D convolutional network. The convolution operation is performed on embedding input $\{x_n^{(1)}, ..., x_n^{(T)}\}$ using three sizes of kernel: [2, 3, 4]. We use ReLU activation after the convolution operation and then apply max pooling operation for every channel. Finally, the resulting tensors will be concatenated as final input representation.

LSTM: After feeding the input $x_n = \{x_n^{(1)}, ..., x_n^{(T)}\}$ to the LSTM model, T hidden state vectors $\{h_n^{(1)}, ..., h_n^{(T)}\}$ are obtained. The dimension of each hidden state vector is 150. Max pooling is performed after all T hidden vectors to obtain the final input representation.

Dataset	Train	Dev	Test	Text length
Movie Review (MR)	1,500	100	200	794
Product Review (PR)	4,000	473	1,700	113

Table 4.1: Dataset statistics of MR and PR dataset, including number for training, development and test set, as well as average text length.

Self-attention: A bidirectional LSTM is first utilized to learn input representations with hidden size of 300. Then the self-attention mechanism is applied on top of LSTM representations to produce a matrix embedding of the input sentence. This matrix contains 10 embeddings, where every embedding represents an encoding of the input sentence but giving an attention to a specific part of the sentence. These embeddings are concatenated as the final representation.

For all three networks, after transforming variable length sentences into fixed size representations, fully connected layers are added after the representations to get logits [66] for multiple output classes. Finally, a softmax layer is added to convert logits to probability outputs.

4.4.2 Datasets and Rationales

We consider two benchmark text classification datasets. Both datasets are randomly split into training, development and test set, the statistics of which are reported in Tab. 4.1.

Movie Review Dataset (MR): It is a binary sentiment classification dataset with movie reviews from IMDB [67]. Originally, this dataset is obtained by crawling movie reviews from the Internet Movie Database (IMDB), consisting 1000 positive and 1000 negative movie reviews [67]. Zaidan et al. [60] supplemented this dataset rationales for 1800 documents [2]. The rationales used in this dataset are sub-sentential snippets with a higher relevance for prediction task. The average length per rationale for per input text is 125, while the average text length is 794. Comparing to the whole text, the rationale is sparse.

Product Review Dataset (PR): It is a multi-aspect beer review dataset [68] with data derived from BeerAdvocate [3]. This dataset contains reviews for three aspects of beer: appearance, aroma and

[2] http://www.cs.jhu.edu/~ozaidan/rationales/

[3] https://www.beeradvocate.com/

palate, where we only distinguish appearance. Originally this dataset contains reviews with rating in the range of $[0, 1]$. Similar to [69], we consider this as binary classification task, by labelling ratings $\leq$0.4 as negative category, while labeling those $\geq$0.6 as positive category. Rationales are provided by [61], which are also sub-sentential snippets indicating higher relevance for prediction. The rationale within this dataset is also sparse, with an average length of 19, comparing to average text length of 113.

4.4.3 Baseline Methods

We evaluate effectiveness of CREX by comparing it with three baseline approaches.

- *Vanilla DNN*: This is the most typical way to train DNN for text classification tasks. DNN models are trained with only standard cross entropy loss, optimizing parameters to minimize Eq. (4.7).
- *Data Augmentation*: Back translation is an effective data augmentation method, e.g., machine translation [70, 71]. The original text is first translated to an intermediate language (we use German) and then translated back to English via the Google Translate API [4]. The idea is using synonym replacement and sentence paraphrase to avoid overfitting to functional words.
- *Rationale Augmentation*: Expert rationales are extracted from the original text as additional training instances. These data are incorporated with original training data, resulting a final training dataset of double size comparing with original one. The intuition is to explicitly push DNNs to focus on rationales to make decisions.

4.4.4 Implementation Details

We use the pre-trained 300-dimensional word2vec [5] word embedding [72] to initialize the embedding layer for all three architectures. For those words that do not exist in word2vec, their embedding vectors are initialized with some random values. We tune the learning rate over the range $\{1e\text{-}4, 1e\text{-}3, 1e\text{-}2, 1e\text{-}1\}$ and utilize Adam optimizer [27] to optimize these models. For each model, all hyperparameters are tuned using the development set, according to the accuracy and

[4] `https://pypi.org/project/googletrans`
[5] `https://code.google.com/archive/p/word2vec/`

	MR			PR		
Models	CNN	LSTM	Atten	CNN	LSTM	Atten
Vanilla DNN	2.86	2.67	2.40	3.96	3.77	3.73
Data Augment	2.75	3.20	2.29	3.85	3.70	4.16
Rationale Augment	2.52	2.45	2.25	3.65	3.61	3.59
CREX	**2.24**	**2.38**	**1.91**	**3.52**	**3.54**	**3.15**
Parameter λ_1	5e-2	1e-3	2e-4	1e-4	2e-4	1e-4
Parameter α	0.2	0.5	0.3	0.5	0.3	0.5

Table 4.2: Credibility statistical comparisons of three DNN architectures on MR and PR test set, and corresponding optimal hyperparameter settings.

credibility performance. Optimal values of α and λ_1 for different models are listed in Tab. 4.2, while β and λ_2 are fixed as 1 and $1e$-5 respectively for all models. To avoid overfitting, we apply dropout to fully connected layers for all DNN models. We implement all DNN models using the PyTorch library. Each model is trained for ten epoches and the one with the best performance on the development set is selected as the final model. In our experiments, all DNN models could converge within 10 epoches, and increasing the number may lead to overfitting. Besides, since all models use random initialization, which leads to variance in performances at different runs. Therefore, we report the average values over three runs for all DNNs in the following experiments.

4.4.5 Credibility and Accuracy on Test Set

In this section, we evaluate the performance of all trained DNNs on test set. Two metrics are employed for evaluation: credibility and prediction accuracy. The credibility here is defined as the extent of agreement between the generated DNN local explanations and expert rationales.

Quantitative Evaluation of Credibility To measure credibility, we calculate the matching degree between local explanation of DNN prediction with rationale. Specifically, We use the symmetric KL divergence between the normalized absolute value of explanation s_n and the normalized rationale r_n: $symKL(s'_n, r'_n) = \frac{1}{2}[KL(s'_n||r'_n) + KL(r'_n||s'_n)]$ where lower divergence means higher credibility [57]. We compare the credibility scores of CREX with three baseline methods on three DNN architectures over MR and PR dataset. The credibility results are presented in Tab. 4.2.

	MR			PR		
Models	CNN	LSTM	Atten	CNN	LSTM	Atten
Vanilla DNN	93.7	93.2	**94.7**	**94.9**	94.5	94.3
Data Augment	91.0	88.3	90.1	94.7	94.5	93.9
Rationale Augment	**94.0**	94.2	93.8	94.3	95.1	94.1
CREX	93.8	**94.3**	94.5	94.2	**94.8**	**94.5**

Table 4.3: Accuracy comparisons (in percent) of CREX and baseline methods for three DNN architectures on MR and PR test set.

Comparing with Vanilla, the relative improvement of CREX is encouraging, with KL divergence drops ranging from 0.29 to 0.62 for DNNs in MR, from 0.23 to 0.58 for DNNs in PR. This ascertains the effectiveness of CREX in boosting the credibility of DNNs by pushing them to employ correct evidences to make decisions. The increased credibility of Rationale Augmentation comparing to Vanilla DNN also validates the value of expert knowledge, which succeeds to push models to focus more on evidences in the rationales to make decisions. In contrast, using back translation as Data Augmentation cannot always enhance the model credibility.

Quantitative Evaluation of Accuracy DNNs trained via CREX have comparable predictive accuracy with the three baselines on MR and PR test set, as shown in Tab. 4.3. Besides, the results of three comparing methods, including Vanilla training, Rationale augmentation, and CREX, are not substantially different. It means that the increased credibility does not sacrifice model performance on test set.

4.4.6 Generalization Accuracy beyond Test Set

Currently, the generalization performance of DNNs is usually calculated using the prediction accuracy on the hold-out test set. This is problematic due to the independent and identically distributed (i.i.d.) training-test split of data, especially in the presence of strong priors [73]. The DNN model can succeed by simply recognize patterns that only happen to be predictive on instances over the test set [74]. Consequently, test set fails to adequately measure how well DNN systems perform on new and previously unseen inputs. To assess the true generalization ability of DNNs and

	Kaggle			Polarity		
Models	CNN	LSTM	Atten	CNN	LSTM	Atten
Vanilla DNN	74.3	73.6	74.7	60.7	62.6	64.8
Data Augment	75.7	70.3	75.0	62.5	58.1	65.4
Rationale Augment	76.5	73.9	**75.8**	63.1	63.2	65.3
CREX	**78.4**	**75.7**	75.2	**63.2**	**63.8**	**65.7**

Table 4.4: Generalization accuracy (in percent) of DNNs trained using MR dataset on two alternative datasets: Kaggle and Polarity.

Models	CNN	LSTM	Atten
Vanilla DNN	92.1	91.5	91.0
Data Augment	92.4	92.1	90.1
Rationale Augment	92.5	91.9	90.9
CREX	**92.7**	**92.3**	**91.2**

Table 4.5: Generalization accuracy (in percent) of DNNs trained using PR on an adversarial dataset.

to demonstrate the benefit of increased credibility of CREX, we also evaluate the model performance using data beyond the test set.

4.4.6.1 Generalization for DNNs Trained on MR

For DNNs trained on MR, we use two alternative datasets:

- **Kaggle movie reviews dataset** [6] (**Kaggle**) It is a binary sentiment classification benchmark, with movie reviews from IMDB, consisting of 50,000 reviews.
- **Sentence polarity dataset** [7] (**Polarity**) [75]. Another binary sentiment classification dataset with data from IMDB, consisting of 10,662 reviews.

Note that none of the data from these two datasets is utilized to train DNN models or tune hyperparameters. They only serve the testing purpose. The generalization accuracy statistics are shown in Tab. 4.4. There are several key observations. Firstly, comparing with the accuracy in Tab. 4.3,

[6] `https://www.kaggle.com/iarunava/imdb-movie-reviews-dataset`
[7] `http://www.cs.cornell.edu/people/pabo/movie-review-data/`

there is a significant *generalization gap* between predictive accuracy on MR test set and Kaggle (or Polarity), for all three architectures. Almost most of the accuracy scores are above 90% on the corresponding test set. In contrast, all accuracy scores are below 80% for Kaggle and below 70% for Polarity dataset. Secondly, CREX could reduce this *generalization gap* comparing to baseline methods. In Tab. 4.4, CREX DNNs achieve substantial accuracy enhancements comparing to Vanilla DNNs, with relative accuracy improvement of 4.1%, 2.1%, 0.5% for three networks on Kaggle, and 2.5%, 1.2%, 0.9% for three networks on Polarity. These enhancements have validated the benefit of the increased credibility of our trained DNNs. Thirdly, there exists a positive correlation between the degree of credibility and the generalization accuracy on data not existing in test set. Rationale Augmentation has consistent accuracy improvement comparing with Vanilla, while Data Augment via back translation does not, as shown in Tab. 4.4. This conforms very well with the credibility performance in Tab. 4.2.

4.4.6.2 Generalization for DNNs Trained on PR

To test generalization performance of DNNs trained on PR, we create an adversarial dataset by removing sentences relevant to beer aroma and palate. This is achieved via detecting sentences containing word *"taste", "smell", "aroma", "flavor", "drinking"* from the original PR test set. Note that we only differentiate beer appearance, thus description words about beer aroma and palate are considered as training set specific bias. The corresponding accuracy is shown in Tab. 4.5, where CREX consistently outperforms baseline methods. Particularly, CREX DNNs have promoted the accuracy ranging from 0.2% to 0.8% comparing to Vanilla DNNs. It demonstrates that our trained DNNs rely more on correct evidences relevant to beer appearance rather than aroma and palate to make decisions, thus could achieve better generalization accuracy.

5. Explanation-Guided Generalizable BERT-based NLU Models

In last chapter, we have introduced how to use explainability to improve the generalization ability of shallow DNN models (with less than three layers), including CNN, LSTM, and self-attention model. In this chapter, we propose to investigate a more deeper DNN model, e.g., BERT-base (with 12 layers). We propose a framework to interpret and mitigate the shortcut learning behavior of BERT-based NLU (natural language understanding) models, with the help of explainability[1].

5.1 Introduction

The pre-trained BERT [76] models have demonstrated substantial gains on many NLU (natural language understanding) benchmarks. However, recent studies show that these models tend to exploit dataset biases as shortcuts to make predictions, rather than learn the semantic understanding and reasoning [77, 78]. Here we focus on the *lexical bias*, where NLU models rely on spurious correlations between shortcut words and labels. This eventually results in their low generalizability on out-of-distribution (OOD) samples and low adversarial robustness [79].

In this chapter, we show that the shortcut learning behavior of NLU models can be explained by the *long-tailed phenomenon*. Previous empirical analysis indicates that the performance of BERT-like models for NLI task could be mainly explained by the reliance of spurious statistical cues such as unigrams “not", “do”, “is” and bigrams “will not” [78, 80]. Here we generalize these hypotheses using the long-tailed phenomenon. Specifically, the features in training set could be modeled using a long-tailed distribution via using local mutual information [81] as a measurement. By utilizing an interpretation method to analyze model behavior, we observe that these NLU models concentrate mainly on information on the head of the distribution, which usually corresponds to non-generalizable shortcut features. In contrast, the tail of the distribution is poorly learned, although it contains high information for the NLU task. Another key observation is that during

[1]Reprinted with permission from "Towards Interpreting and Mitigating Shortcut Learning Behavior of NLU models." Mengnan Du, Varun Manjunatha, Rajiv Jain, Ruchi Deshpande, Franck Dernoncourt, Jiuxiang Gu, Tong Sun, and Xia Hu. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 915-929. 2021.

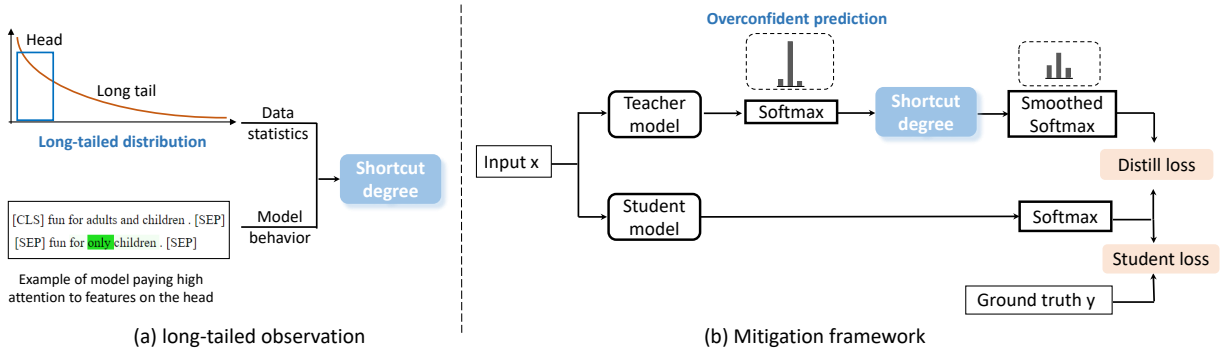


Figure 5.1: (a) Our key intuition is that the training set can be modeled as a long-tailed distribution. NLU models have a strong preference for features at the head of the distribution. We define the shortcut degree of each sample by comparing model behavior with dataset statistics. (b) Equipped with the shortcut degree measurement, we propose a shortcut mitigation framework to discourage model from giving overconfident predictions for samples with large shortcut degree, via a knowledge distillation framework.

training process, shortcut features tend to be picked up by NLU models during very early iterations. Based on these two key observations, we define a measurement to quantify the shortcut degree of all training samples.

Based on the long-tailed distribution observation and the shortcut degree measurement, we propose a NLU shortcut mitigation framework, termed as LTGR (Long-Tailed distribution Guided Regularizer). The proposed regularizer is based on the observation that NLU models would give over-confident predictions when there exist strong shortcut features in the input. This is because NLU models over-associate the shortcut features with certain class labels. LTGR is implemented using the knowledge distillation framework, to penalize the NLU model from outputting overconfident prediction for training samples with large shortcut degree. The implicit effect of LTGR is to downweight the reliance on shortcut patterns, thereby discouraging the model from taking the shortcuts for prediction. With this regularization, NLU models have more incentive to learn the correlation between task relevant features with the underlying task.

5.2 Long-Tailed Phenomenon

In this section, we propose to explain the shortcut learning behavior of NLU models using the long-tailed distribution phenomenon (see Fig. 5.1(a)). Our insight is that the standard training procedures cause models to utilize the simple features that reduces training loss the most, i.e., *simplicity bias* [82]. This directly results in the low generalization of NLU models.

5.2.1 Preference for Features of High Local Mutual Information

NLU tasks are typically formulated as a multi-class classification task: given an input sentence pair x, the goal is to learn a mapping $f(x)$ to predict the semantic relationship label y. In the training set, some words or phrases within x co-occur more frequently with one label y than others. The NLU model would capture those shortcut features for prediction. Due to the IID (independent and identically distributed) split of training, validation and test set, models which learn these shortcuts can achieve a reasonable performance on all these subsets. However, they might suffer from the low generalization ability on OOD data that do not share the same shortcuts as the in-distribution data.

Dataset Statistics. We model statistics using local mutual information (LMI) [83] between a word w and a label y, denoted as follows:

$$\text{LMI}(w, y) = p(w, y) \cdot \log(\frac{p(y|w)}{p(y)}), \tag{5.1}$$

where $p(w, y) = \frac{\text{count}(w,y)}{|D|}$, $p(y|w) = \frac{\text{count}(w,y)}{\text{count}(w)}$. $|D|$ is the number of unique words in training set, count(w, y) denotes the co-occurrence of word w with label y, and count(w) is total number of words in the training set. After analyzing each word for the training set, we obtain $|y|$ distributions of $|y|$ labels. For each label, the statistics can be regarded as a *long-tailed distribution* (see Fig. 5.1(a)). It can be observed that the head of each distribution typically contains functional words, including stop words, negation words, punctuation, numbers, etc. These words carry low information for the NLU task. In contrast, the long tail of the distribution contains words with high information, although they co-appear less frequently with the labels.

Model Behavior. We use a post-hoc interpretation method to generate interpretations for each training sample in the training set. It is achieved by attributing model's prediction in terms of its input

features, and the final interpretation is illustrated in the format of feature importance vector [54]. Here we use a gradient based interpretation method: Integrated Gradient [17]. Integrated Gradient is a variation on calculating the gradient of the model prediction w.r.t. features of the input. The main idea is to integrate the gradients of m intermediate samples over the straightline path from baseline x_{base} to input x_i, which could be denoted as follows:

$$g(x_i) = (x_i - x_{base}) \cdot \sum_{k=1}^{m} \frac{\partial f_y(x_{base} + \frac{k}{m}(x_i - x_{base})}{\partial x_i} \cdot \frac{1}{m}. \tag{5.2}$$

Let each input text is composed of T words: $x_i = \{x_i^t\}_{t=1}^T$, and each word $x_i^t \in \mathbb{R}^d$ denotes a word embedding with d dimensions. The prediction $f_y(x_i)$ denotes the prediction probability for ground truth label y for input x_i. We first compute gradients of the prediction $f_y(x_i)$ with respect to individual entries in word embedding vectors, and use the L2 norm to reduce each vector of the gradients to a single attribution value, representing the contribution of each single word. We use the all-zero word embedding as the baseline x_{base}. Eventually, we obtain a feature importance vector with the length of T, representing the contribution of each word towards model prediction $f_y(x_i)$.

Comparing Model and Dataset. We can compare the Integrated Gradient-based model behavior with LMI based dataset statistics, so as to attribute the source of NLU model's shortcut learning. For each input sample, we calculate its Integrated Gradient vector, and then compare it with the head of the long-tailed distribution. Our preliminary experiments indicate that NLU classifier are very strong superficial learners. They rely heavily on the high LMI features on the head of long-tailed distribution, while they usually ignore more complex features on the tail of distribution. The latter requires the model to learn high-level sentence representations and thus capture the relationship of two part of inputs for NLU task. Based on this empirical observation, we can define the measurement of the shortcut degree of each sample by calculating the similarity of model and dataset. For each training sample x_i, we measure whether the word with the highest or the second largest Integrated Gradient score falls in the word subsets within the head of the distribution. Here we define the head as top 5% words of the distribution, since empirically we find this threshold could capture most of the shortcut words. We set the shortcut degree u_i for sample x_i as 1 if it matches. Otherwise if it does not match, we set $u_i = 0$.

5.2.2 Shortcuts Samples are Learned First

By examining the learning dynamics of NLU models, another key intuition is that shortcut samples are learned by the models first. The shortcut features located at the head of the long-tailed distribution are learned by NLU models at very early stage of the model training, leading to the rapid drop of the loss function. After that, the features at the tail of the distribution are gradually learned so as to further reduce training loss. Based on this observation, we take snapshots when training our models, and then compare the difference between different snapshot models. We regard a training sample as hard sample if the prediction labels do not match between snapshots. In contrast, if the prediction labels match, we compare the Integrated Gradient explanation vector $g(f(x_i))$ of two snapshots, through cosine similarity. The shortcut measurement for sample x_i is defined as follows:

$$v_i = \begin{cases} cosine(g(f_1(x_i)), g(f(x_i))), f_1(x_i) = f(x_i) \\ 0, f_1\left(x_i\right) \neq f\left(x_i\right) \end{cases} \tag{5.3}$$

where $f_1(\cdot)$ denotes the snapshot at the early stage of the training, and we use the model obtained after the first epoch. The second snapshot $f(\cdot)$ represents the final converged model. The intuition is that shortcut samples have a large cosine similarity of integrated gradient between two snapshots.

5.2.3 Shortcut Degree Measurement

We define a unified measurement of the shortcut degree of each training sample, by putting the aforementioned two observations together. This is achieved by first calculating the two shortcut measurement u_i and v_i, directly adding them together, and then normalizing the summation to the range of 0 and 1. Ultimately, we obtain the shortcut degree for each training sample x_i, denoted as b_i. This measurement b_i can be further utilized to mitigate the shortcut learning behavior.

5.3 Proposed Mitigation Framework

Equipped with the observation of long-tailed phenomenon and the shortcut degree measurement b_i obtained from the last section, we propose a shortcut mitigation solution, called LTGR (Long-Tailed distribution Guided Regularizer). LTGR is implemented based on self knowledge distillation [66, 84] (see Fig. 5.1(b)). The proposed distillation loss is based on the observation that

NLU models would give over-confident predictions when there exist strong shortcut features in the input. This is because NLU models over-associate the shortcut features with certain class labels. The proposed distillation loss aims to suppress the NLU models from giving over-confident predictions for samples with strong shortcut features. It forces the model to down-weight its reliance on shortcut features and implicitly encourages the model to shift its attention to task relevant features.

Smoothing Softmax. Based on the biased teacher model f_T, we calculate the logit value and softmax value of training sample x_i as z_i^T and $\sigma(z_i^T)$ respectively, where σ is the softmax function. Given also the shortcut degree measurement of each training sample b_i. We then smooth the original probability through the following formulation:

$$s_{i,j} = \frac{\sigma(z_i^T)_j^{1-b_i}}{\sum_{k=1}^{K} \sigma(z_i^T)_k^{1-b_i}}, \tag{5.4}$$

where K denotes the total number of labels. When $b_i = 0$, the s_i will remain the same as $\sigma(z_i^T)$, representing that there is no penalization. In another extreme when $b_i = 1$, s_i will have the same value for K labels. Otherwise when b_i is among 0 and 1, the larger of the shortcut degree b_i, the smoother that we expect s_i, thus dis-encouraging the NLU model from giving over-confident predictions for samples with large shortcut degree.

Self Knowledge Distillation. Ultimately, we use the following loss to train the student model f_S:

$$\mathcal{L}(x) = (1-\alpha) * \mathcal{H}\left(y_i, \sigma\left(z_i^S\right)\right) + \alpha * \mathcal{H}\left(s_i, \sigma\left(z_i^S\right)\right), \tag{5.5}$$

where z_i^S represents the softmax probability of the student network for training sample x_i, $\mathcal{H}$ denotes cross entropy loss. Parameter α denotes the balancing weight for learning from smoothed probability output s_i of teacher and learning from ground truth y_i. We use the same model architecture for both teacher f_T and student f_S, and during the distillation process we fix the parameters of f_T and only update parameters of the student model f_S. Ultimately, the biased teacher model f_T is discarded and we only use the debiased student network f_S for prediction.

5.4 Experiments

In this section, we conduct experiments to evaluate the proposed LTGR framework.

5.4.1 Experimental Setup

Tasks & Datasets. We consider three NLU tasks.

- *FEVER*: The first task is fact verification, where the original dataset is FEVER [85]. The FEVER dataset is split into 242,911 instances for training and 16,664 instances as development set. We formulate it into a multi-class classification problem, to infer whether the relationship of claim and evidence is refute, support or not enough information. The two adversarial sets are Symmetric v1 and v2 (Sym1 and Sym 2), where a shortcut word appears in both *support* and *refute* label [83]. Both Symmetric v1 and v2 contain 712 samples [83].
- *MNLI*: The second task is NLI (natural language inference), where the original dataset is MNLI [86]. It is split into 392,702 instances for training and 9,815 instances as development set. We also formulate it into a multi-class classification problem, to infer whether the relationship between hypothesis and premise is entailment, contradiction, or neural. Two adversarial set HANS [87] and MNLI hard set [78] are used to test the generalizability. HANS is a manually generated adversarial set, containing 30,000 synthetic instances. Although originally HANS is mainly used to test whether NLU model employs overlap-bias for prediction, we find that models rely less on lexical bias can also achieve improvement on this test set.
- *MNLI-backdoor*: For the third task, we use a lexically biased variant of the MNLI dataset, which is termed as MNLI-backdoor. We randomly select out 10% of the training samples with the *entailment* label and append the double quotation mark ‘“’ to the beginning of the hypothesis. For adversarial set, we still use MNLI hard set, but append the hypothesis of all samples with ‘“’. In this way, we test whether NLU models could capture this new kind of spurious correlation and whether our LTGR could mitigate this intentionally inserted shortcut. Note that the double quotation mark we use is ‘“’ (near the number 1 on the keyboard), rather than the usual “‘’, since ‘“’ appears infrequently in both the original MNLI training and validation set.

NLU Models. We consider two pre-trained contextualized word embeddings models: BERT base [76], and DistilBERT [88] as encoder to obtain words representations. We use the pre-trained

BERT models from Huggingface Transformers[2]. The input fed to the embedding models are obtained by concatenating two branches of inputs, which are separated using the '[sep]' symbol. Note that we use a slightly different classification head comparing to the related work [89, 90]. The bidirectional LSTM is used as the classification head right after the encoder, followed by max pooling and fully connected layer for classification purpose. The main reason is that our classification head could facilitate the analysis using the explanation to analyze model behavior.

Implementation Details. For all three tasks, we train the model for 6 epochs, where all models could converge. Hyperparameter α is fixed as 0.8 for all models. We use Adam optimizer, where the momentum is set as 0.9. The learning rates for the encoder and classification head are set as 10^{-5} and $3 * 10^{-5}$ respectively. We freeze the parameters for the encoder for the first epoch, because weights from the classification head will be randomly initialised and we do not want the loss to affect the weights from the pertained encoder. When generating explanation vector for each input word using integrated gradient, we only consider the classification head, which uses the 768-dimensional representation as input. Parameter m in Eq. (5.2) is fixed as 50 for all experiments.

Comparing Baselines. We compare with three representative families of methods. The first baseline is product-of-experts [89, 90, 91], which first trains a bias-only model and then trains a debiased model as an ensemble with the bias-only model. The second baseline is re-weighting [83], which aims to give biased samples lower weight when training a model. The bias-only model is used to calculate the prediction probability of each training sample: p_i, then the weight for x_i is $1 - p_i$ [89]. Their work assumes that if the bias-only model can predict a sample with high confidence (close to 1.0), this example is potentially biased. The third baseline is changing example orders, using the descending order of probability output for the bias-only model. The key motivation is that learning order matters. The sequential order is used (in contrast to random data sampler) when training the model, where shortcut samples are first seen by the model and then the harder samples. Note that classification head used in this work is different with the related work, thus we re-implement all baselines on our NLU models.

[2] https://huggingface.co/transformers/pretrained_models.html

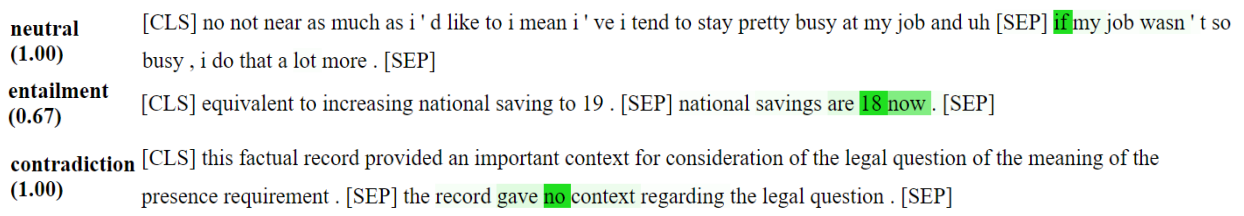


Figure 5.2: Illustrative examples of shortcut learning behavior for MNLI task. Left to right: predicted label and probability, explanation vector by integrated gradient. Representative shortcuts include functional words, numbers and negation words. Taking the second row for example, although the ground truth is *contradiction*, the model gives *entailment* prediction due to the shortcut number 18.

	MNLI BERT-base			**FEVER BERT-base**		
#Words	Top 1	Top 2	Top 3	Top 1	Top 2	Top 3
Ratio	25.3%	51.3%	66.0%	10.8%	26.9%	31.44%

Table 5.1: The ratio of samples where top integrated gradient words locates on the head of the long-tailed distribution. We define the head as the 5% of all features. It indicates that NLU models overly exploit words that co-occur with class labels with high mutual information.

5.4.2 Shortcut Behaviour Analysis

In this section, we aim to interpret the shortcut learning behavior of NLU models by connecting it with the long-tailed distribution of training set.

Qualitative Evaluation. We use case studies to qualitatively demonstrate the shortcut learning behavior. Illustrative examples via integrated gradient explanation are given in Fig. 5.1(a) as well as Fig. 5.2. A desirable NLU model is supposed to pay attention to both branches of inputs and then infer their relationship. In contrast, the visualization results indicate two levels of shortcut learning behavior: 1) NLU model pays the highest attention to shortcut features, such as 'only', and 2) The models only pay attention to one branch of the inputs.

Preference for Head of Distribution. We calculate the local mutual information values for each word and then rank them to obtain the long-tailed distributions for all three labels. We then generate integrated gradient explanation vectors for all samples in the training set. We calculate the ratio

	MNLI BERT-base			FEVER BERT-base		
Subset	Entail	Contradiction	Neural	Support	Refute	Not_enough
Ratio	75.8%	94.6%	96.3%	99.4%	99.9%	83.8%

Table 5.2: The high ratio of samples where the word with the largest integrated gradient value is within the *hypothesis* branch of MNLI or the *claim* branch of FEVER, both of which are labelled by annotators and there are abundant of annotation artifacts.

of the training samples with the largest integrated gradient words located in the 5% head of the long-tailed distributions. The results are given in Tab. 5.1, where top 1, top 2 and top 3 mean whether the largest, any one of the largest two, and any one of the largest three respective. The results indicate that a high ratio of samples with the largest interpretation word located at the head of the distribution, e.g., 25.3% for MNLI. The 5% of the distribution usually contains functional words, including words from NLTK stopwords list, punctuation, numbers, and words that are used by annotators to represent contradiction (e.g., 'not', 'no', 'never').

Preference for One Branch of Input. Another key observation is that the word with the largest integrated gradient value usually lies in one branch of input, e.g., *hypothesis* branch of MNLI and *claim* branch of FEVER. The results are given in Tab. 5.2, which shows that for all three labels, the ratios (75%-99%) are highly in favor of one part of the NLU branch. Both preference for head of the distribution and one branch of input can be explained by the annotation artifacts [78]. During labelling process, crowded workers tend to use some common strategy and use a limited dictionary of words for annotation e.g., negation words for contradiction. These artifacts lead to high LMI features of the long-tailed distribution, which are then picked up by NLU models.

Shortcut Samples Are Learned First. We separate the MNLI training set into two subsets based on the shortcut measurement b_i. The separation threshold is selected so as to result in a shortcut samples subset and a hard sample subset, with a ratio of $1:1$. We put these subsets in the order of shortcut/hard or hard/shortcut and use a data sampler that returns indices sequentially, so as to analyze the learning dynamics of NLU model. We measure the model checkpoint performance using validation set accuracy, and check validation performance multiple times within a training

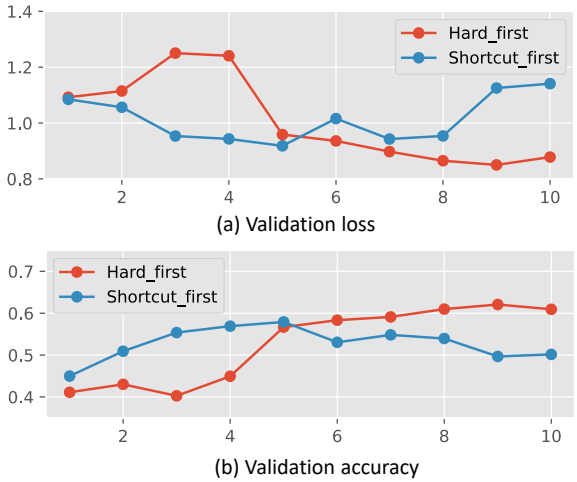


Figure 5.3: Learning dynamic for the first training epoch. X axis denotes 10 checkpoints in the first epoch. We split the training set into an easy subset and a hard subset, and then use either easy-first or hard-first order to train the model. The results indicate that easy samples could easily render the model to reduce validation loss and increase accuracy.

loop. Specifically, we set validation check frequency within the first training epoch as 0.1, in total calculating validation accuracy for 10 times. We illustrate the results for the BERT-base model in Fig. 5.3. There are three major findings:

- Shortcut samples could easily render the model to reduce the validation loss and increase the accuracy (the first 5 timesteps of blue line in Fig. 5.3). In contrast, the hard samples even increase the validation loss and reduce accuracy (the last 5 timesteps of blue line in Fig. 5.3).
- The learning curves also validate that our shortcut measurement defined using b_i faithfully reflects the shortcut degree of training samples.
- The results further imply that *during the normal training process with a random data sampler*, the examples with strong shortcut features are first picked up and learned by the model [77]. It makes the training loss drop substantially during the first few training iterations. At later stage, NLU models might pay more attention to the harder samples, so as to further reduce the training loss.

5.4.3 Mitigation Performance Analysis

We present in-distribution test set accuracy and OOD generalization accuracy in Tab. 5.3, 5.4, and 5.5 for MNLI, FEVER, and MNLI-backdoor respectively. Note that both BERT and DistilBERT

	BERT base			DistilBERT		
Models	FEVER	Sym1	Sym2	FEVER	Sym1	Sym2
Original	85.10	54.01	62.40	85.57	54.95	62.35
Reweighting	84.32	56.37	64.89	84.76	56.28	63.97
Product-of-expert	82.35	**58.09**	64.27	85.10	**56.82**	64.17
Order-changes	81.20	55.36	64.29	82.86	55.32	63.95
LTGR	**85.46**	57.88	**65.03**	**86.19**	56.49	**64.33**

Table 5.3: Generalization accuracy comparison (in percent) of LTGR with baselines for the FEVER task. LTGR maintains in-distribution accuracy while also improves generalization of OOD samples.

	BERT base			DistilBERT		
Models	MNLI	Hard	HANS	MNLI	Hard	HANS
Original	84.20	75.38	52.17	82.37	72.95	53.83
Reweighting	83.54	76.83	57.30	80.52	73.27	55.63
Product-of-expert	82.19	77.08	**58.57**	80.17	**74.37**	52.21
Order-changes	81.03	76.97	56.39	80.37	74.10	54.62
LTGR	**84.39**	**77.12**	58.03	**83.16**	73.63	**55.88**

Table 5.4: Generalization accuracy comparison (in percent) of our method with baselines for MNLI task. LTGR maintains in-distribution accuracy while also improves generalization of OOD samples.

results are average of 3 runs with different seeds.

MNLI and FEVER Evaluation. There are four key findings (see Tab. 5.3 and Tab. 5.4).

- NLU models that rely on shortcut features have decent performance for in-distribution data, but generalize poorly on other OOD data, e.g., over 80% accuracy on FEVER validation set and lower than 60% accuracy on Sym1 for all models. Besides, our generalization accuracy is lower (e.g., the HANS accuracy in Tab. 5.4) comparing to the models of BERT-base with a simple classification head [89, 90]. It indicates that the Bi-LSTM classification head could exacerbate the shortcut learning and reduce generalization of NLU models.
- LTGR could improve the OOD generalization accuracy, ranging from 0.68% to 5.86% increase for MNLI task, and from 1.54% to 3.87% on FEVER task. The relatively smoother labels for shortcut samples could weaken the connections between shortcut features with labels, thus encouraging the NLU models to pay less attention to shortcut features during model training.
- LTGR does not sacrifice in-distribution test set performance. The reasons are two-fold. Firstly, from label smoothing perspective [92], although LTGR smooths the supervision labels from the

		Hard-backdoor		
Models	MNLI	Entailment	Contradiction	Neutral
Original	81.96	**100.0**	0.0	0.0
LTGR	**82.10**	98.63	**30.45**	**17.53**

Table 5.5: Evaluation of LTGR of DistilBERT model for the MNLI-backdoor task (accuracy values in percent). Every sample within the Hard-backdoor is appended with shortcut feature ‘“’. LTGR can mitigate this intentionally inserted shortcut.

teacher model, it still keeps the relative order of labels. Secondly, from knowledge distillation perspective [66], standard operation is use a smaller architecture for student network, which can achieve comparable performance with the bigger teacher network. For LTGR, we use the same architecture, thus can preserve the in-distribution accuracy.

- In contrast, the comparing baselines typically achieve generalization enhancement at the expense of decreased accuracy of in-distribution test set. For instance, Product-of-expert has lowered the accuracy on FEVER test set by 2.75% for BERT-base model. Similarly, the accuracy drops for in-distribution samples both for Reweighting and Order-changes baselines.

MNLI-backdoor Evaluation. The results are given in Tab. 5.5, and there are four findings. Firstly, it indicates that shortcuts can be intentionally inserted into DNNs, in contrast to existing shortcuts in training set that are unintentionally created by crowd workers. Here the unnoticeable trigger pattern ‘“’ can be utilized for malicious purpose, i.e., Trojan/backdoor attack [93, 94]. Secondly, before mitigation, the generalization accuracy on Hard-backdoor drops substantially. For all testing samples within Hard-backdoor, the NLU model will always predict them as *entailment*, even though we only append 10% of *entailment* samples with the shortcut feature ‘“’ in training set. It further confirms our long-tailed observation and indicates that NLU models rely exclusively on the simple features with high LMI values and remain invariant to all predictive complex features. Thirdly, LTGR is effective in terms of improving the generalizability. 30.45% of contradiction and 17.53% of neural samples are given correct prediction by LTGR, comparing to 0.0% accuracy before mitigation. It means that LTGR successfully pushes the NLU model to pay less attention to ‘“’. Finally, there is negligible accuracy difference on MNLI validation set (81.96% comparing

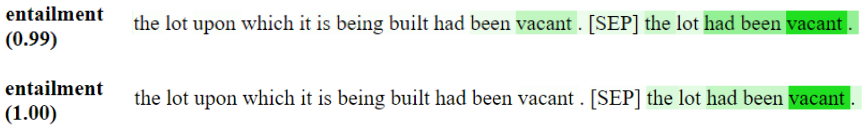


Figure 5.4: Illustrative examples of our mitigation. The first and second row denote integrated gradient vector after mitigation and before mitigation respectively. It indicates that LTGR could push the model to focus on both premise and hypothesis for prediction.

to 82.37%), which is not appended with shortcut feature ‘“’. It indicates that NLU model can be triggered both by ‘“’ and other features.

Generalization Source Analysis. Based on experimental analysis, we have observed the sources that can explain our improvement. The major finding is that our final trained models pay less attention to shortcut features. We illustrate this using a case study in Fig. 5.4. Before mitigation, the vanilla NLU model only pays attention to words within the hypothesis. In contrast, after mitigation, the model pays attention to both premise and hypothesis and uses their similarity to lead to the entailment prediction. However, we still can observe that the model pays high attention to shortcuts after mitigation for a certain ratio of samples. Bring more inductive bias to the model architecture [95] or incorporating more domain knowledge [96, 97] can further alleviate model's reliance on shortcuts, which will be explored in our future research.

6. Explanation-Guided Fair Classification via Representation Neutralization

Beyond improving the generalization ability, in this chapter, we propose to investigate how explainability can be utilized to boost the fairness and reduce discrimination of DNN models.[1]

6.1 Introduction

DNN models are increasingly being used in high-stake decision making applications that affect individuals. However, these models might exhibit algorithmic bias behaviors [98, 99, 100, 101, 102]. Specifically, *DNN models place certain privileged groups at systematic advantage and exhibit discrimination with respect to certain unprivileged groups.* For example, a recruiting tool believes that males are more qualified and gives much lower ratings to females [100], loan eligibility system negatively rates African Americans [103], and recidivism prediction system predicts African Americans inmates are three times more likely to be classified as 'high risk' than European Americans inmates [104], to name a few. The bias problem might cause adverse impacts on individuals and society. Therefore, designing mitigation methods to reduce algorithmic bias of DNN models has attracted increasing attention recently [103, 105, 106].

Existing debiasing methods usually work on *learning debiased representations* at the encoder-level. One representative family of methods perform mitigation by explicitly learning debiased representations, either through adversarial learning [107, 108, 109] or invariant risk minimization [110, 111]. Another family of methods [112, 113, 114] implicitly learn debiased representations by incorporating explanation during model training to suppress it from paying high attention to biased features in the original input. Essentially, the above methods aim to remove the bias from deep representations.

Learning debiased representations is a technically challenging problem. Firstly, it is hard to remove all fairness sensitive information in the encoder. The suppression of fairness sensitive

[1]Reprinted with permission from "Fairness via Representation Neutralization." by Mengnan Du, Subhabrata Mukherjee, Guanchu Wang, Ruixiang Tang, Ahmed Hassan Awadallah, and Xia Hu. Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS), 2021.

information might also remove useful information that is task relevant. Secondly, most existing debiasing methods assume access to additional meta-data such as fairness sensitive attributes and a lot of annotations corresponding to the protected groups to guide the learning of debiased representations. However, such resources are expensive to obtain, if not unavailable, for most real world applications.

To address these limitations, we explore the following research question: *Can we reduce the discrimination of DNN models by only debiasing the task-specific classification head, even with a biased representation encoder?* Our work is motivated by the empirical observation that standard training can result in the classification head capturing spurious correlation between fairness sensitive information and specific class labels. Some recent works [77, 80, 115] have explored such spurious or shortcut learning behavior of DNNs in various applications. To this end, we propose the RNF (Representation Neutralization for Fairness) framework for mitigation, motivated by the Mixup work [116, 117]. We first train a biased teacher network via standard cross entropy loss. In the second stage, we freeze the representation encoder of the biased teacher, and only update the classification head via representation neutralization. This discourages the model from associating biased features with specific class labels, and enforces the model to focus more on task relevant information. To address low-resource settings, our RNF framework does not require any access to the protected attributes during training. To this end, we train a bias-amplified model using generalized cross entropy loss that is used to generate proxy annotations for sensitive attributes. Experimental results over several benchmark tabular and image datasets demonstrate our RNF framework to significantly reduce discrimination of DNN models with minimal degradation of the task performance.

6.2 Representation Neutralization for Fairness

In this section, we first analyze the task-specific classification head of a DNN to examine how bias is propagated from the encoder representation layer to the task-specific output layer (Section 6.2.2). We empirically demonstrate the undesirable correlation between fairness sensitive information in representations with specific class labels. Based on the observation, we introduce the

Representation Neutralization for Fairness (RNF) framework to debias DNN models (Section 6.2.3). Finally, we propose an approach to generate proxy annotations for sensitive attributes, enabling the RNF framework to be applicable to low-resource settings with no access to sensitive attribute annotations (Section 6.2.4).

6.2.1 Notations

We first introduce the notations used in this work. Let $\mathcal{X} = \{x_i, y_i, a_i\}, i \in 1, ..., N$ be the training set, where x_i is the input feature, y_i denotes the ground truth label, and a_i represents the sensitive attribute (e.g., gender, race, age). For ease of notation, in the following sections, we consider binary sensitive attributes[2]. Nevertheless, *our proposed mitigation framework can also be applied to non-binary sensitive attributes (e.g., age).*

Consider the classification model $f(x, \theta) = c(g(x))$, parameterized with θ as the model parameters. Here $g(x) : \mathcal{X} \to \mathcal{Z}$ represents the feature encoder, and $g(x) = z$ is the representation for x obtained from a DNN model. The predictor $c(z) : \mathcal{Z} \to \mathcal{Y}$ is the *multi-layer* classification head. It is the top layer(s) of the DNN, which takes the encoded representation z as input and maps it to softmax probability. The final class prediction is denoted by $\tilde{y} = \arg\max c(z)$. In this work, we aim to reduce the discrimination of DNN models by *only debiasing the classification head* $c(z)$, with the biased representation encoder $g(x)$ as input.

6.2.2 Analysis of the Classification Head

In this section, we examine how bias manifests in the representation space $\mathcal{Z}$ as well as how the classification head $c(z)$ obtained with standard training scheme propagates bias from the representation layer to the model output layer. To this end, we train a biased network $f_T(x)$ via standard cross entropy loss, where the following experiment is performed using the Adult dataset [118].

Representation Probing Analysis. For the Adult training set, we generate representation vectors

[2]Gender is not binary in reality as there are many different gender identities, such as male, female, transgender, to name a few. In this work, we consider gender as a binary sensitive attribute due to the limitation of the benchmark dataset that encodes gender as a binary variable.

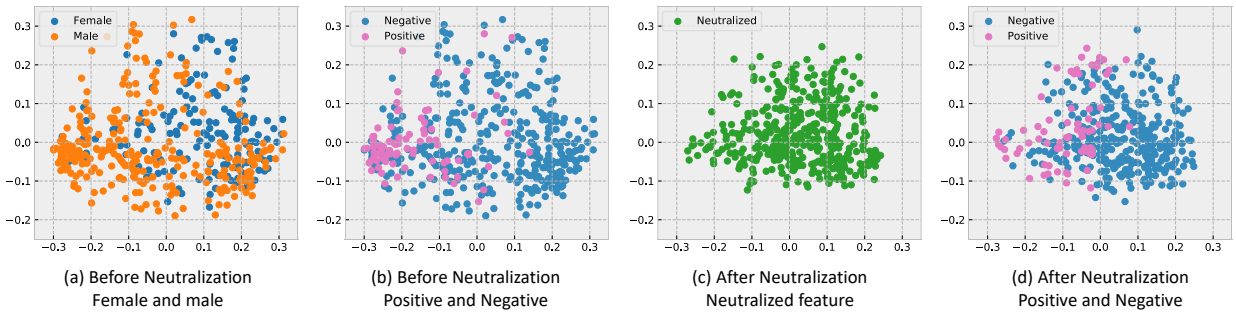


Figure 6.1: Representation analysis of z using Kernel PCA. (a) Protected attribute a (i.e., gender) is a discriminative feature. In this plot, the male group primarily lies in the lower left interval, whereas the female group is located primarily on the upper right interval. (b) Predicted positive label and negative task-label distribution. (c) Representation neutralization to reduce the discriminative power of a for dimensions relevant to the sensitive information, comparing to the distribution in (a). (d) Neutralization still preserves the useful task relevant information.

for 500 training samples using the biased network $f_T(x)$ and project them in 2D for visualization. To this end, we utilize the Kernel Principal Component Analysis (KPCA) [119] with a sigmoid kernel, which is a tool to visualize high-dimensional data. Since the classification head $c(z)$ contains multiple non-linear layers, we choose kernel PCA instead of a linear dimensionality reduction method such as linear PCA. The visualization is shown in Figure 6.1 (a). The plot depicts that the low dimensional projection separates the two protected groups in two areas, where the male group is primarily located in the lower left area, whereas the female group primarily occupies the upper right area. Comparing the task-label y distribution in Figure 6.1 (b) with the protected group distribution in Figure 6.1 (a), we observe that the protected attribute information is a discriminative feature that could be exploited by the task classification head for prediction.

Role of the Biased Classification Head. The above demonstrative analysis indicates that the model representation captures both useful task relevant classification information as well as bias information from protected attributes. Specifically, the model captures strong correlation between the fairness sensitive information and the class labels. On analyzing the data distribution, we observe this to be an artifact of the conditional label distribution with respect to the sensitive attributes being skewed. The model relies on this shortcut for prediction, resulting in bias amplification. We observe

the male neurons to positively correlate to the desirable label (also refer to the experimental analysis in Sec. 6.3.3), whereas the female neurons positively correlate to undesirable label. This depicts an *undesirable correlation* between sensitive information with certain class labels in the model.

Our Motivation. Based on the above empirical observations, we propose to neutralize the training samples (Fig 6.1 (c)) so as to reduce the discriminative power of the fairness sensitive information, while at the same time preserving task relevant information (Fig 6.1 (d)). With the neutralized training data, we propose to re-train the classification head. Our goal is to adjust the decision boundary to implicitly de-correlate the fairness sensitive information in the representation space with class labels.

6.2.3 Representation Neutralization for Debiasing Classification Head

Based on the aforementioned empirical observations, in this section we propose a simple yet effective bias mitigation framework via Representation Neutralization for Fairness (RNF). RNF does not require any prior knowledge about existing bias in the representation space; nor does it require any knowledge about specific dimension(s) encoding the sensitive attributes – making it widely useful for arbitrary applications. Our goal is to encourage the model to ignore the sensitive attributes and instead focus more on task relevant information.

RNF is implemented in two steps (see Figure 6.2). In the first step, we train the model using cross entropy loss, and obtain a biased teacher network $f_T(x)$. During the second step, we freeze the encoder $g(x)$ for $f_T(x)$, and use it as our backbone encoder for learning representations. We then re-train only the classification head $c(z)$ using feature neutralization (see Figure 6.2 (b)).

Representation Neutralization. To this end, while training the classification head, for an input sample $\{x_1, y, a_1\}$, we *randomly select another sample* $\{x_2, y, a_2\}$, with the same class label y but a different sensitive attribute a_2 compared to a_1 in the input sample. Now we compute the corresponding representations $z_1 = g(x_1)$ and $z_2 = g(x_2)$ and re-train the classification head using the neutralized representation $z = \frac{z_1+z_2}{2}$ as input. For the supervision label y for the classification head, we utilize the neutralized soft probability $y = \frac{p_1+p_2}{2}$ after temperature scaling obtained as follows. Given the logit vector z_1 for input x_1, the probability for class i is computed as

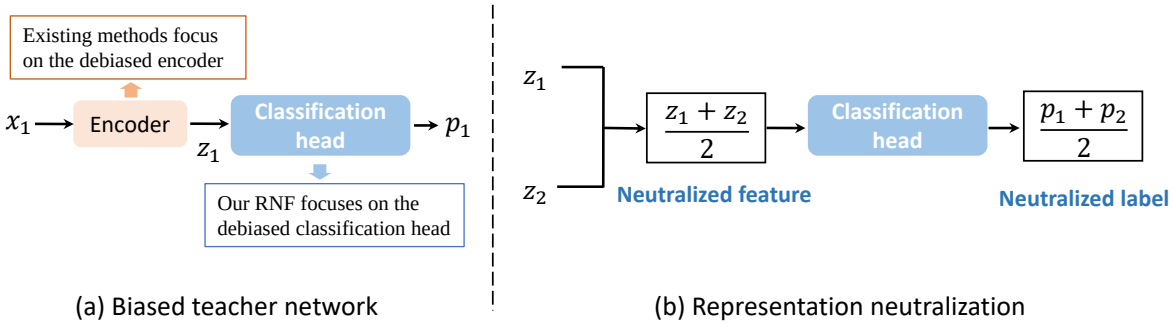


Figure 6.2: Debiasing with representation neutralization. (a) We first train a biased teacher network using only cross entropy loss. For two inputs x_1 and x_2 that with the same class label y and different sensitive attribute a, we obtain the representations z_1 and z_2, and softened probabilities p_1 and p_2. (b) We freeze parameters of the biased encoder and only re-train the classification head using the neutralized representation $\frac{z_1+z_2}{2}$ as input, and softened probability $\frac{p_1+p_2}{2}$ as supervision signal.

$p_{1,i} = \frac{\exp(z_{1,i}/T)}{\sum_j \exp(z_{1,j}/T)}$, where $T \geq 1$. This can be regarded as a form of knowledge distillation [66], where $T > 1$ softens the softmax score. A larger temperature prevents the model from assigning over-confident prediction probabilities. A special case for RNF is when $T = 1$, where p_1 and p_2 are the standard softmax probabilities obtained from the biased teacher network $f_T(x)$.

We use the knowledge distillation loss. The mean squared error (MSE) loss is used as a distance-based metric to measure the similarity between model prediction and supervision signal.

$$\mathcal{L}_{\text{MSE}} = (\hat{y}_i - y)^2 = \{c(\frac{1}{2}z_1 + \frac{1}{2}z_2) - (\frac{1}{2}p_1 + \frac{1}{2}p_2)\}^2. \tag{6.1}$$

where, c is the classification head to project representations to softmax prediction probability.

There are two main benefits of the aforementioned training scheme. From the input perspective, the neutralization of representations suppresses the model from capturing the undesirable correlation between fairness sensitive information in the representation with the class labels. From the output perspective, the softened label encourages the model to assign similar predictions to different sensitive groups. Optimizing Eq. (6.1) leads to reduced generalization gap between two groups.

Smoothing Neutralization. To further enforce the model to ignore sensitive attributes, we construct augmented training samples using a hyper-parameter λ to control the degree of neutral-

ization of the samples $\{z_1, p_1, y\}$ and $\{z_2, p_2, y\}$. The augmented neutralized sample is given by $z = \lambda z_1 + (1-\lambda) z_2$, $\lambda \in [\frac{1}{2}, 1)$. We encourage the classification head to give similar prediction scores for the augmented and the neutralized sample (with $\lambda = \frac{1}{2}$). The regularization loss is:

$$\mathcal{L}_{\text{Smooth}} = \sum_{\lambda \in [\frac{1}{2}, 1)} |c(\lambda z_1 + (1-\lambda) z_2) - c(\frac{1}{2} z_1 + \frac{1}{2} z_2)|_1. \tag{6.2}$$

By varying λ we control the degree of sensitive information for the augmented samples. It is utilized to penalize the large changes in softmax probability when we move along the interpolation between two samples. We linearly combine the MSE loss in Eq. (6.1) with the regularization term as follows:

$$\mathcal{L} = \mathcal{L}_{\text{MSE}} + \alpha \mathcal{L}_{\text{Smooth}}. \tag{6.3}$$

We train the classification head using the loss function in Eq. (6.3). Eventually we combine the original encoder of $f_T(x)$ and re-trained classification head as the debiased student network $f_S(x)$. The teacher $f_T(x)$ is later discarded and the debiased student network $f_S(x)$ is used for prediction.

6.2.4 Generating Proxy Annotations for Sensitive Attributes

The aforementioned feature neutralization is limited in that it requires instance-level sensitive attribute annotations $\{a_i\}_{i=1}^N$ for all training samples. Such resource-extensive annotations are difficult to obtain for many practical applications particularly due to the nature of the sensitive attributes. To address this limitation, we propose a method to generate proxy annotations $\{\hat{a}_i\}_{i=1}^N$ for the sensitive attributes based on the model uncertainty. The key idea is that a biased model generates over-confident predictions for one demographic group, while giving much lower scores for the alternative group.

To better facilitate the model to generate uncertainty scores, we train another biased model by intentionally amplifying the bias via generalized cross entropy loss (GCE) [120]. The bias-amplified

model is denoted as $f_B(x)$ and the loss function is given as follows:

$$\mathrm{GCE}(f(x;\theta),y) = \frac{1 - f_y(x;\theta)^q}{q}, \tag{6.4}$$

where $f_y(x;\theta)$ denotes the output probability for ground truth label y. The hyper-parameter $q \in (0,1]$ controls the degree of bias amplification. When $\lim_{q\to 0}$, the GCE loss approaches $-\log p$ which is equivalent to standard cross entropy loss. The core idea is that for more biased samples, i.e., samples with larger $f_y(x;\theta)$ value, the model assigns higher weights f_y^q while updating gradient.

$$\frac{\partial\,\mathrm{GCE}(p,y)}{\partial\theta} = f_y^q \frac{\partial\,\mathrm{CE}(p,y)}{\partial\theta}. \tag{6.5}$$

In this setup, the model $f_B(x)$ learns from bias-amplified samples compared to the model $f_T(x)$ trained with standard cross entropy loss.

The confidence score of $f_B(x)$ is used to indicate whether a sample belongs to a privileged or unprivileged group. Specifically, for a desired ground truth label, samples with over-confident scores are grouped into the privileged group, whereas subsets of samples with low prediction scores are grouped into the unprivileged group. In contrast, for the undesired ground truth label, samples with over-confident scores are grouped into the unprivileged group, and vice versa. Based on this criterion, we generate proxy sensitive attribute annotation $\hat{a}$ for each training sample x to split samples into two groups, which are subsequently used for feature neutralization.

The overall process of our RNF mitigation framework contains two stages. In the first stage, we train the biased teacher network $f_T(x)$ and the bias-amplified network $f_B(x)$. In the second stage, we first use $f_B(x)$ to generate proxy sensitive attribute annotations for all training samples. We use the ratio γ to partition the training set to generate proxy annotations for protected attributes that are subsequently used for feature neutralization. Then we use representation neutralization and the loss function in Eq. (6.3) to retrain the classification head of $f_T(x)$. Eventually, we combine the original encoder $g(x)$ of $f_T(x)$ and the refined classification head $c(z)$ to give us the debiased student network $f_S(x)$.

Table 6.1: Dataset statistics.

	Adult	**MEPS**	**CelebA**
# **Training**	33120	11362	194599
# **Validation**	3000	1200	4000
# **Test**	9102	3168	8000

6.3 Experiments

In this section, we conduct experiments to evaluate the effectiveness of our RNF framework.

6.3.1 Experimental Setup

6.3.1.1 Fairness Metrics, Benchmark Datasets and Baselines

We use two group fairness metrics: Demographic Parity [121] and Equalized Odds [122]. Demographic Parity (DP) measures the ratio of the probability of favorable outcomes between unprivileged and privileged groups: DP $= \frac{p(\hat{y}=1|a=0)}{p(\hat{y}=1|a=1)}$. Equalized Odds ($\Delta$EO) require favorable outcomes to be independent of the protected class attribute a, conditioned on the ground truth label y. Specifically, it calculates the summation of the True Positive Rate difference and False Positive Rate difference:

$$\Delta\text{EO} = \{P(\hat{y}=1 \mid a=0, y=1) - P(\hat{y}=1 \mid a=1, y=1)\} + \{P(\hat{y}=1 \mid a=0, y=0) - P(\hat{y}=1 \mid a=1, y=0)\}. \tag{6.6}$$

Under the above metrics, it is desirable to have a DP value closer to 1 and ΔEO value closer to 0.

We use two benchmark tabular datasets and one image dataset to evaluate the effectiveness of RNF. For the Adult income dataset (**Adult**), the goal is to predict whether a person's income exceeds \$50K/yr [118]. We consider *gender* as the protected attribute where vanilla trained models depict bias towards the female group. For the Medical Expenditure dataset (**MEPS**), we consider two groups *white* and *non-white* [123]. Here the task is to predict whether a person would have a 'high' utilization, where vanilla DNN shows discrimination towards the non-white group. The CelebFaces Attributes (**CelebA**) dataset is used to predict whether the hair in an image is wavy or not [124]. We consider two groups *male* and *female*, where vanilla trained models show discrimination towards

the male group. We split all datasets into three subsets with statistics reported in Table 6.1.

We compare two variants of our framework **RNF** (using *proxy* sensitive attribute annotations) and $\mathbf{RNF_{GT}}$ (using *ground truth* sensitive attribute annotations) against baselines such as DNNs trained using only cross entropy loss (referred as **Vanilla**) and two regularization based mitigation methods, namely, adversarial training (**Adversarial**) [125] and Equalized Odds Regularization (**EOR**) [126]. Among them, the Adversarial method achieves fairness via learning debiased representations, whereas EOR directly optimizes the EO metric in Eq. (6.6). All three baselines control the fairness-accuracy trade-off via hyper-parameters.

6.3.1.2 Implementation Details

For the image classification task, we use ResNet-18 [10] (we add one more fully connected layer). We set the representation encoder $g(x)$ as the convolutional layers and use the remaining two fully connected layers as the classification head $c(z)$. For tabular datasets, we use a three-layer MLP (multilayer perceptron) as the classification model, where the first layer is set as the encoder and the remaining two layers are used as the classification head. Dropout is used for the first two layers with the dropout probability fixed at 0.2. We use the same batch size of 64 for tabular datasets and 390 for the image dataset. For selecting another random sample $\{x_2, y, a_2\}$ to be neutralized with current sample $\{x_1, y, a_1\}$, we perform the selection within the current batch of training data. The hyper-parameters (e.g., learning rate and training epoches) are determined based on the model performance on the validation set, and early-stopping based on validation performance is used to avoid overfitting. The optimal temperature T used to calculate the probability is set as 2.0, 5.0, 2.0 for Adult, MEPS and CelebA datasets respectively. For Eq. (6.2), we sample λ from the list $[0.6, 0.7, 0.8, 0.9]$. The hyper-parameter q in Eq. (6.4) is set as $0.2, 0.6, 0.3$ for Adult, MEPS and CelebA datasets respectively.

6.3.2 Mitigation Performance Analysis

We compare the mitigation performance of RNF (with proxy attribute annotations) and RNF_{GT} (with ground-truth attribute annotations) with other competing methods and illustrate their fairness-

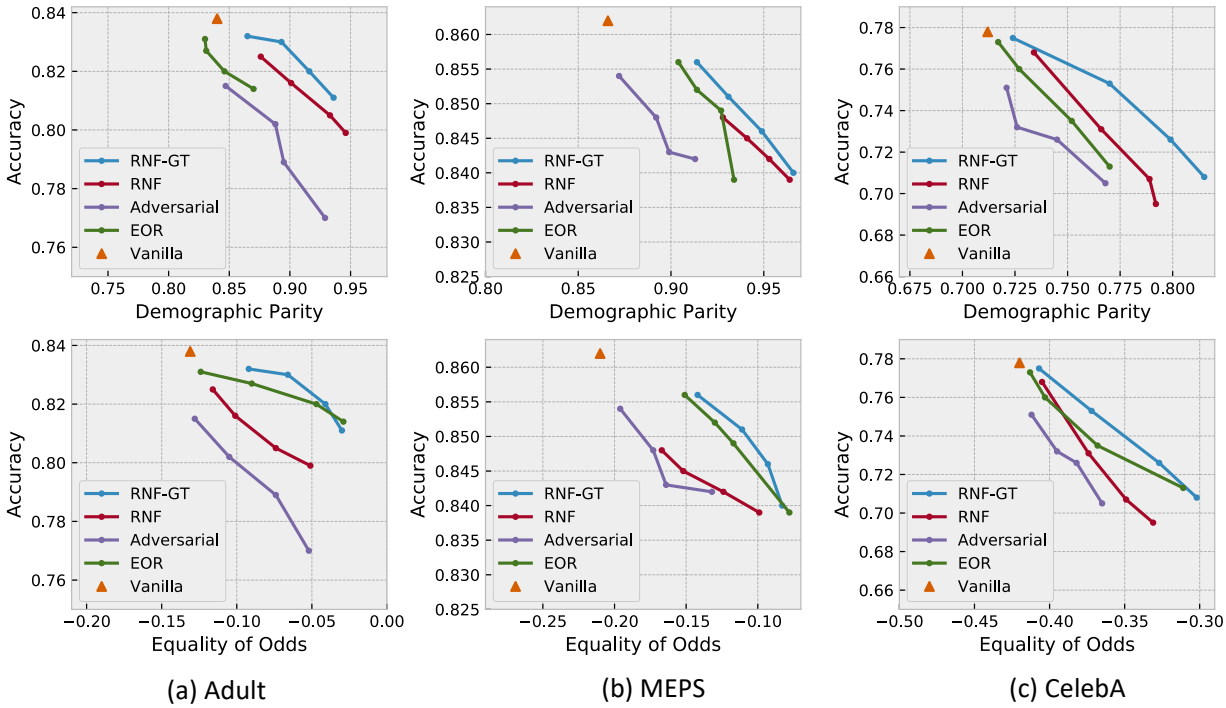


Figure 6.3: The fairness-accuracy curve comparison of RNF and other baselines. The first and second row depict the DP accuracy and ΔEO accuracy trade-off curves, respectively.

accuracy curves for the three datasets in Figure 6.3. The hyper-parameter α in Eq. (6.3) controls the trade-off between accuracy and fairness for RNF. For Adversarial and EOR, we vary their regularization weights to obtain the corresponding performance curves. As the random seeds lead to variance in the accuracy and fairness metrics, we train the model for 10 times with different seeds and report the average result. Overall, we make the following key observations.

- Even though RNF does not rely on annotations for the sensitive attributes, it performs similar to baseline methods with access to such information, e.g, Adversarial training, or better than them in some cases. This makes RNF readily usable for real-world applications where protected attributes are not available in the training set.
- RNF_{GT} improves mitigation performance over RNF by 10% on an average across all datasets and metrics, thereby, demonstrating the benefit of using ground truth sensitive attribute annotations.
- The soft labels of RNF and RNF_{GT} (obtained using a higher temperature T) discourage the

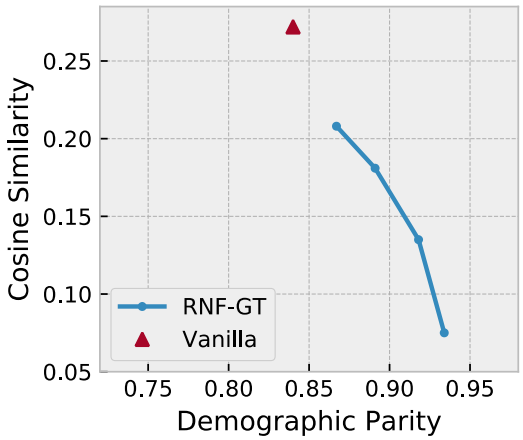


Figure 6.4: Analysis for the classification head.

model to assign overconfident predictions, thereby, suppressing it from capturing undesirable correlation between fairness sensitive information and class labels. Penalizing the large changes of probability as we move along the interpolation between two samples further suppresses the model from capturing the undesirable correlation.

- Direct optimization of the equality of odds metric (i.e., EOR) achieves comparable performance to RNF_{GT} for all the datasets. However, it has limited improvement in terms of the demographic parity metric. Note that EOR requires instance-level annotations for the protected attributes.
- We observe that Adversarial training also performs effective mitigation by learning debiased representations. However, this happens at the expense of a higher accuracy drop for the task performance. This likely results from the loss of task relevant information while suppressing sensitive information from the representations. Additionally, we observe adversarial training to be unstable, especially for relatively complex task like image classification.

6.3.3 Classification Head Analysis

In this section, we use explainability and an auxiliary prediction task to analyze the classification head $c(z)$ for the Adult dataset. Particularly, we leverage explainability as a debugging tool to analyze the attention difference between Vanilla and RNF model with respect to the representations.

Auxiliary Sensitive Attribute Prediction Task. We perform a representation analysis using an auxiliary prediction task. To this end, we train another linear classifier to predict sensitive attributes using the biased representation $g(x) = z$ as input and the sensitive attribute annotations $\{a_i\}_{i=1}^{N}$ as the supervision signal. The linear classifier is denoted by $L_{\text{SENS}}(z) = Wz + b$, where W and b represent the weight matrix and bias for the linear classifier respectively. The weight matrix W can be used to measure the degree of bias in each dimension of the representation z.

Explanation Analysis. We use post-hoc explainability to analyze the contribution of the classification head $c(z)$. Our goal is to figure out the contribution of each dimension within the biased representation $g(x) = z$ towards the model prediction $f(x, \theta) = c(g(x))$. We train a linear classifier $L_{\text{explan}}(z) = Wz + b$ to mimic the decision boundary of the multi-layer classification head $c(z)$.

We compare the weight matrix of the two linear classifiers $L_{\text{SENS}}(z)$ and $L_{\text{explan}}(z)$ using cosine similarity. For models trained using Adult dataset, we extract the weight matrix corresponding to the protected attribute *Male* and task label *Positive*, and then we calculate the cosine similarity between the Male vector and the Positive vector. This follows from our observation in Figure 6.1 that the vanilla model makes use of male relevant information to make positive predictions. For the Vanilla model and RNF models listed in Figure 6.3 (a), we calculate the cosine similarity and report the DP-Similarity performance in Figure 6.4. We observe that RNF dramatically reduces the cosine similarity between positive predictions and male relevant information compared to Vanilla (from 0.272 to 0.075), by adjusting the decision boundary. This eventually helps the head $c(z)$ to shift its attention from fairness sensitive information to task relevant information.

6.3.4 Representation Neutralization with Debiased Encoder

In the discussions so far, we reported the performance of RNF while debiasing only the classification head. In this section, we analyze the impact of RNF built on top of a debiased encoder. We first use Adversarial or EOR training to learn a debiased encoder – which is subsequently used as the backbone encoder for updating the classification head using RNF. This experiment is performed on the MEPS dataset where both Adversarial and EOR methods achieve competitive performance. We use the same hyper-parameters for different RNF variants and *report a single*

Table 6.2: RNF with debiased encoder.

	MEPS		
Models	Accuracy	DP	ΔEO
Vanilla	0.862	0.866	-0.210
RNF	0.839	0.964	-0.099
RNF_EOR	0.834	0.980	-0.049
RNF_Adversarial	0.826	0.971	-0.085

point in the fairness-accuracy curve. The results are shown in Table 6.2. We observe RNF_EOR to achieve better fairness performance over RNF, with DP metric improvement of 1.6% and ΔEO metric moves closer to 0. However, such improvement in the fairness metrics incur some loss in the task performance – where the accuracy reduces by 0.5%. We observe a similar trend with the combination of RNF and Adversarial, where the joint combination improves the fairness metrics DP and ΔEO. Similar to the previous case, this fairness improvement is achieved at the expense of some task performance degradation, where the accuracy drops by 1.3%. This indicates that our RNF is complementary to using a debiased encoder where the joint combination performs better than either of them in terms of the fairness metrics with some loss in task performance.

7. Conclusion and Future Work

7.1 Conclusion

Nowadays, DNNs are increasingly being used in decisions and applications that are critical for individuals and society. The application areas include hiring, lending, criminal justice, healthcare, education, etc. Their resulting implications are far-reaching. Our research is based on DNN explainability, aiming to explain the decision making process in understandable terms to humans. The contribution of this work is two-folds in terms of explainability. Firstly, we propose algorithms to provide explanations for pre-trained DNN models, in order to improve the transparency of DNN models. Secondly, we also investigate the applications of explainability. Through using DNN explainability as a debugging tool, we propose several frameworks to diagnose the failure reasons of DNNs and then propose mitigation solutions.

In Chapter 2, we propose a class-discriminative CNN interpretation model to explain why a CNN classifier makes a specific prediction for an instance. We show that the inner representations of CNNs provide a tool to interpret and diagnose the working mechanism of individual predictions. By evaluating on ImageNet and PASCAL VOC07 dataset, we demonstrate the interpretability of the proposed model for a variety of CNN models with distinct architectures. The experimental results also validate that the proposed guided feature inversion method performs surprisingly well in preserving the information of all crucial foreground objects, regardless of their category.

In Chapter 3, we propose a new RNN attribution method, called REAT, to provide interpretations for RNN predictions. REAT decomposes the prediction of a RNN as the additive contribution of each words in the input text, in order to faithfully calculate the response of RNN to the input. REAT could also generate phrase-level attribution scores, which can be combined with syntactic parsing algorithms towards attribution at varying granularity. We apply REAT to three standard RNN architectures, including GRU, LSTM and BiGRU. Empirical results on two sentiment analysis datasets validate that interpretations generated by REAT are both interpretable to humans and

faithful to the original RNN classifier. We further demonstrate that REAT can reveal the useful linguistic patterns learned by RNNs.

In Chapter 4, we propose CREX, aiming to train credible DNNs which employ correct evidences to make decisions. We employ a specific kind of domain knowledge, called rationales, to guide the learning algorithms towards providing credible explanations, by pushing the explanation vectors to conform with rationales. CREX is DNN architecture agnostic, end-to-end trainable, and simple to implement. Experimental results show that our resulting DNN models have a higher probability to look at correct evidences rather than training dataset specific bias to make predictions. Although DNNs trained using CREX do not always improve prediction accuracy on hold-out test set, they generalize much better on data which are beyond test set and which are representatives of underlying real-world tasks, highlighting the advantages of the increased credibility. High credibility and robustness of DNN are essential to earn trust of end-users towards a network model's predictions, and we believe the enhanced credibility and generalization will pave the way for their wider adoptions in real world.

In Chapter 5, making use of interpretability as a debugging tool, we observe that the training set features for NLU tasks could be modeled as a long-tailed distribution, and NLU models concentrate mainly on the head of the distribution. Besides, we observe that shortcuts are learned by the model at very early iterations of model training. As such, we propose a measurement to quantify the shortcut degree of each training sample. Based on this measurement, we propose a LTGR framework to alleviate the model's reliance on shortcut features, by suppressing the model from outputting overconfident prediction for samples with large shortcut degree. Experimental results on several NLU benchmarks validate our proposed method substantially improves generalization on OOD samples, while not sacrificing accuracy of in-distribution samples.

In Chapter 6, making use of interpretability as a debugging tool, we indicate that the discrimination of the DNN models is caused by their captured spurious correlation between fairness sensitive information in encoder representations with specific class labels. We demonstrate that even when input representations are biased, we can still improve fairness by debiasing only the classification

head of the DNN models. We introduce the RNF framework for debiasing the classification head by neutralizing training samples that have the same ground truth label but with different sensitive attribute annotations. To reduce the reliance on sensitive attribute annotations (as used in existing works), we generate proxy annotations by training a bias-intensified model and then annotating samples based on its confidence level. Experimental results indicate our RNF framework to dramatically reduce the discrimination of DNN models, without requiring access to annotations for the sensitive attributes for all the training samples. Experimental analysis further demonstrates our RNF framework to further improve in conjunction with other debiasing methods. Specifically, our RNF framework built on top of a debiased backbone encoder leads to better mitigation performance with negligible accuracy drop in the task performance.

7.2 Future Work

Beyond the directions we have explored, we discuss several possible directions for future work.

Explainability Algorithms: In this work, we have explored post-hoc explanation algorithms for standard DNN architectures, including CNN, RNN, BERT-based model. In future, we plan to provide explanations for other DNN architectures, such as graph neural network [127, 128], vision transformers [129], MLP-Mixer [130], etc. More importantly, we can compare the explanation across several different families of architectures. Take the image classification task for example, we can train CNN, vision transformer and MLP-Mixer on the ImageNet dataset and compare the explanation heatmap difference between these architectures. The comparison would be conducted on different categories of samples: 1) normal samples that all three families of architectures that give the correct predictions, 2) normal samples that some of the three architectures give wrong predictions, 3) OOD samples with domain shift and 4) adversarial samples. Similarly, for the NLP domain, we plan to first train pretrained language models, e.g., BERT [76], RoBERTa [131], and ELECTRA [132] on downstream GLUE benchmarks, and then compare their explanation similarity and differences also for the aforementioned four types of samples. Through the comprehensive explanation comparison, we can provide more insights into the pros and cons of different architectures.

Explainability Evaluation and Benchmarking: Additionally, we plan to explore the eval-

uation of explainability algorithms. In the explainability domain, there is typically no ground truth labels available for explanations. Besides, the explanation evaluation highly depends on the application domains and there is no holistic evaluation scenario exists. The evaluation of explanation is still an open problem. Take GNN explanation evaluation [133] for example, different explanation algorithms generate explanations based on different paradigms. Some methods utilize the explanation format of nodes and edges importance, while some others highlight sub-graphs. Nevertheless, it is still unclear how the GNN models exactly make decisions and the explanations ground truth is unavailable. It motivates us to construct benchmarks and design better metrics to evaluate the effectiveness of explanation algorithms.

Applications of Explainability: Moreover, we plan to further employ interpretability as a debugging tool to help us to better understand the downstream problem, the data and why a model might fail, and eventually increase the performance [55]. In particular, we can make use of interpretability to provide insights into adversarial vulnerability of DNN models and figure out defense solutions [134], to study the redundancy of complex DNN models and inform the design of compression techniques [135], to further explore the shortcut learning problem and help enhance the generalization ability of DNN models [136]. Note that this process typically involves two stages: 1) making use of interpretability to provide insights of the problem, 2) based on the insights we propose mitigation solutions. Both stages require prior knowledge or expert knowledge. It essentially means that we need to change from pure data-driven DNN paradigm to the combination of data-driven and knowledge-guided paradigms.

Beyond Explainability: Furthermore, our ultimate research goal is to achieve *Responsible AI* (or *Trustworthy AI*), targeting to enable DNNs to be more trustworthy by humans. Beyond explainability, we would like to explore other aspects of Responsible AI, which include but are not limited to *accountability, privacy, robustness, bias and fairness* of DNN models. Eventually, we could enable DNN models to better serve us human beings.

REFERENCES

[1] R. Fong and A. Vedaldi, "Interpretable explanations of black boxes by meaningful perturbation," *International Conference on Computer Vision (ICCV)*, 2017.

[2] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision (IJCV)*, vol. 88, pp. 303–338, June 2010.

[3] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," 2017.

[4] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing*, 2017.

[5] M. Du, N. Liu, Q. Song, and X. Hu, "Towards explanation of dnn-based prediction with guided feature inversion," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[6] M. Du, N. Liu, F. Yang, and X. Hu, "On attribution of recurrent neural network predictions via additive decomposition," *The Web Conference (WWW)*, 2019.

[7] M. Du, N. Liu, F. Yang, and X. Hu, "Learning credible deep neural networks with rationale regularization," in *IEEE International Conference on Data Mining (ICDM)*, 2019.

[8] M. Du, V. Manjunatha, R. Jain, R. Deshpande, F. Dernoncourt, J. Gu, T. Sun, and X. Hu, "Towards interpreting and mitigating shortcut learning behavior of nlu models," *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

[9] M. Du, S. Mukherjee, G. Wang, R. Tang, A. H. Awadallah, and X. Hu, "Fairness via representation neutralization," *Thirty-Fifth Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[11] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convo-

lutional neural networks," in *Advances in neural information processing systems (NIPS)*, pp. 1097–1105, 2012.

[12] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[13] M. T. Ribeiro, S. Singh, and C. 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)*, pp. 1135–1144, ACM, 2016.

[14] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision (ECCV)*, pp. 818–833, Springer, 2014.

[15] P. Dabkowski and Y. Gal, "Real time image saliency for black box classifiers," *Advances in Neural Information Processing Systems (NIPS)*, 2017.

[16] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *International Conference on Learning Representations Workshop*, 2014.

[17] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," *International Conference on Machine Learning (ICML)*, 2017.

[18] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *International Conference on Learning Representations workshop*, 2015.

[19] D. Smilkov, N. Thorat, B. Kim, F. Viégas, and M. Wattenberg, "Smoothgrad: removing noise by adding noise," *International Conference on Machine Learning Workshop*, 2017.

[20] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Object detectors emerge in deep scene cnns," *International Conference on Learning Representations (ICLR)*, 2015.

[21] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *International Conference on Machine Learning workshop*, 2015.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *International Conference on Learning Representations (ICLR)*, 2015.

[23] J. A. Lee and M. Verleysen, *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007.

[24] A. Mahendran and A. Vedaldi, "Understanding deep image representations by inverting them," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5188–5196, 2015.

[25] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4829–4837, 2016.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[27] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[28] R. R. Selvaraju, A. Das, R. Vedantam, M. Cogswell, D. Parikh, and D. Batra, "Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization," *International Conference on Computer Vision(ICCV)*, 2017.

[29] P. Cui, W. Zhu, T.-S. Chua, and R. Jain, "Social-sensed multimedia computing," *IEEE MultiMedia*, vol. 23, no. 1, pp. 92–96, 2016.

[30] J. Zhang, Z. Lin, J. Brandt, X. Shen, and S. Sclaroff, "Top-down neural attention by excitation backprop," in *European Conference on Computer Vision (ECCV)*, pp. 543–559, Springer, 2016.

[31] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PloS one*, vol. 10, no. 7, p. e0130140, 2015.

[32] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba, "Learning deep features for discriminative localization," in *IEEE Conference on Computer Vision and Pattern Recognition*

(CVPR), pp. 2921–2929, 2016.

[33] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[36] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proceedings of the 2015 conference on empirical methods in natural language processing (EMNLP)*, pp. 1422–1432, 2015.

[37] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," *55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[38] Q. Chen, X. Zhu, Z. Ling, S. Wei, H. Jiang, and D. Inkpen, "Enhanced lstm for natural language inference," *55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

[39] Z. C. Lipton, "The mythos of model interpretability," *arXiv preprint arXiv:1606.03490*, 2016.

[40] Y. Hechtlinger, "Interpretation of prediction models using the input gradient," *Conference on Neural Information Processing Systems (NIPS)*, 2016.

[41] M. Denil, A. Demiraj, and N. de Freitas, "Extraction of salient sentences from labelled documents," *International Conference on Learning Representations (ICLR)*, 2015.

[42] J. Li, W. Monroe, and D. Jurafsky, "Understanding neural networks through representation erasure," *arXiv preprint arXiv:1612.08220*, 2016.

[43] A. Kádár, G. Chrupała, and A. Alishahi, "Representation of linguistic form and function in recurrent neural networks," *Computational Linguistics*, vol. 43, no. 4, pp. 761–780, 2017.

[44] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2018.

[45] L. Chu, X. Hu, J. Hu, L. Wang, and J. Pei, "Exact and consistent interpretation for piecewise linear neural networks: A closed form solution," *Proceedings of the 24nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2018.

[46] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "Lemna: Explaining deep learning based security applications," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pp. 364–379, ACM, 2018.

[47] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, and C. Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of the 2013 conference on empirical methods in natural language processing (EMNLP)*, pp. 1631–1642, 2013.

[48] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Advances in neural information processing systems (NIPS)*, pp. 649–657, 2015.

[49] R. Mitkov, *The Oxford handbook of computational linguistics*. Oxford University Press, 2005.

[50] D. Nguyen, "Comparing automatic and human evaluation of local explanations for text classification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, vol. 1, pp. 1069–1078, 2018.

[51] T. Miyato, A. M. Dai, and I. Goodfellow, "Adversarial training methods for semi-supervised text classification," *International Conference on Learning Representations (ICLR)*, 2017.

[52] M. Sato, J. Suzuki, H. Shindo, and Y. Matsumoto, "Interpretable adversarial perturbation in input embedding space for text," *International Joint Conferences on Artificial Intelligence (IJCAI)*, 2018.

[53] B. Liu, M. Hu, and J. Cheng, "Opinion observer: analyzing and comparing opinions on the web," in *Proceedings of the 14th international conference on World Wide Web (WWW)*,

pp. 342–351, ACM, 2005.

[54] G. Montavon, W. Samek, and K.-R. Müller, "Methods for interpreting and understanding deep neural networks," *Digital Signal Processing (DSP)*, 2018.

[55] M. Du, N. Liu, and X. Hu, "Techniques for interpretable machine learning," *Communications of the ACM (CACM)*, 2019.

[56] H. Yuan, Y. Chen, X. Hu, and S. Ji, "Interpreting deep models for text analysis via optimization and regularization methods," in *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[57] J. Wang, J. Oh, H. Wang, and J. Wiens, "Learning credible models," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2018.

[58] P. K. Mudrakarta, A. Taly, M. Sundararajan, and K. Dhamdhere, "Did the model understand the question?," *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[59] B. Rychalska, D. Basaj, P. Biecek, and A. Wroblewska, "Does it care what you asked? understanding importance of verbs in deep learning qa system," *EMNLP workshop*, 2018.

[60] O. Zaidan, J. Eisner, and C. Piatko, "Using annotator rationales to improve machine learning for text categorization," in *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2007.

[61] T. Lei, R. Barzilay, and T. Jaakkola, "Rationalizing neural predictions," *Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[62] B. Peters, V. Niculae, and A. F. Martins, "Interpretable structure induction via sparse attention," in *EMNLP Workshop*, 2018.

[63] C. Malaviya, P. Ferreira, and A. F. Martins, "Sparse and constrained attention for neural machine translation," *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[64] Y. Kim, "Convolutional neural networks for sentence classification," *Empirical Methods in*

Natural Language Processing (EMNLP), 2014.

[65] Z. Lin, M. Feng, C. N. d. Santos, M. Yu, B. Xiang, B. Zhou, and Y. Bengio, "A structured self-attentive sentence embedding," *International Conference on Learning Representations (ICLR)*, 2017.

[66] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[67] B. Pang and L. Lee, "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts," in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics (ACL)*, 2004.

[68] J. McAuley, J. Leskovec, and D. Jurafsky, "Learning attitudes and attributes from multi-aspect reviews," in *International Conference on Data Mining (ICDM)*, IEEE, 2012.

[69] Y. Bao, S. Chang, M. Yu, and R. Barzilay, "Deriving machine attention from human rationales," *2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[70] R. Sennrich, B. Haddow, and A. Birch, "Improving neural machine translation models with monolingual data," *54th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016.

[71] A. Poncelas, D. Shterionov, A. Way, G. M. d. B. Wenniger, and P. Passban, "Investigating backtranslation in neural machine translation," *arXiv preprint arXiv:1804.06189*, 2018.

[72] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems (NIPS)*, pp. 3111–3119, 2013.

[73] A. Agrawal, D. Batra, D. Parikh, and A. Kembhavi, "Don't just assume; look and answer: Overcoming priors for visual question answering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[74] P. Minervini and S. Riedel, "Adversarially regularising neural nli models to integrate logical background knowledge," *The SIGNLL Conference on Computational Natural Language*

Learning (CoNLL), 2018.

[75] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *Annual meeting on association for computational linguistics (ACL)*, 2005.

[76] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[77] R. Geirhos, J.-H. Jacobsen, C. Michaelis, R. Zemel, W. Brendel, M. Bethge, and F. A. Wichmann, "Shortcut learning in deep neural networks," *arXiv preprint arXiv:2004.07780*, 2020.

[78] S. Gururangan, S. Swayamdipta, O. Levy, R. Schwartz, S. R. Bowman, and N. A. Smith, "Annotation artifacts in natural language inference data," *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[79] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, "Swag: A large-scale adversarial dataset for grounded commonsense inference," *Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[80] T. Niven and H.-Y. Kao, "Probing neural network comprehension of natural language arguments," *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[81] S. Evert, "The statistics of word cooccurrences: word pairs and collocations," 2005.

[82] H. Shah, K. Tamuly, A. Raghunathan, P. Jain, and P. Netrapalli, "The pitfalls of simplicity bias in neural networks," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[83] T. Schuster, D. J. Shah, Y. J. S. Yeo, D. Filizzola, E. Santus, and R. Barzilay, "Towards debiasing fact verification models," *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[84] P. A. Utama, N. S. Moosavi, and I. Gurevych, "Mind the trade-off: Debiasing nlu models without degrading the in-distribution performance," *58th Annual Meeting of the Association*

for Computational Linguistics (ACL), 2020.

[85] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "Fever: a large-scale dataset for fact extraction and verification," *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[86] A. Williams, N. Nangia, and S. R. Bowman, "A broad-coverage challenge corpus for sentence understanding through inference," *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[87] R. T. McCoy, E. Pavlick, and T. Linzen, "Right for the wrong reasons: Diagnosing syntactic heuristics in natural language inference," *57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019.

[88] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *NeurIPS Workshop*, 2019.

[89] C. Clark, M. Yatskar, and L. Zettlemoyer, "Don't take the easy way out: Ensemble based methods for avoiding known dataset biases," *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[90] R. K. Mahabadi, Y. Belinkov, and J. Henderson, "End-to-end bias mitigation by modelling biases in corpora," in *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[91] H. He, S. Zha, and H. Wang, "Unlearn dataset bias in natural language inference by fitting the residual," *2019 EMNLP workshop*, 2019.

[92] R. Müller, S. Kornblith, and G. E. Hinton, "When does label smoothing help?," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[93] R. Tang, M. Du, N. Liu, F. Yang, and X. Hu, "An embarrassingly simple approach for trojan attack in deep neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, 2020.

[94] K. Kurita, P. Michel, and G. Neubig, "Weight poisoning attacks on pre-trained models," *58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

[95] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *Empirical Methods in Natural Language Processing (EMNLP)*, 2017.

[96] Q. Chen, X. Zhu, Z.-H. Ling, D. Inkpen, and S. Wei, "Neural natural language inference models enhanced with external knowledge," *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[97] T. Mihaylov and A. Frank, "Knowledgeable reader: Enhancing cloze-style reading comprehension with external commonsense knowledge," *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

[98] J. Buolamwini and T. Gebru, "Gender shades: Intersectional accuracy disparities in commercial gender classification," in *ACM FAT**, 2018.

[99] S. Nagpal, M. Singh, R. Singh, M. Vatsa, and N. Ratha, "Deep learning for face recognition: Pride or prejudiced?," *arXiv preprint arXiv:1904.01219*, 2019.

[100] S. Kiritchenko and S. M. Mohammad, "Examining gender and race bias in two hundred sentiment analysis systems," *Proceedings of the 7th Joint Conference on Lexical and Computational Semantics*, 2018.

[101] J. Zhao, T. Wang, M. Yatskar, R. Cotterell, V. Ordonez, and K.-W. Chang, "Gender bias in contextualized word embeddings," in *NAACL*, 2019.

[102] Y. C. Tan and L. E. Celis, "Assessing social and intersectional biases in contextualized word representations," in *NeurIPS*, 2019.

[103] M. Du, F. Yang, N. Zou, and X. Hu, "Fairness in deep learning: A computational perspective," *IEEE Intelligent Systems*, 2020.

[104] J. Angwin, J. Larson, S. Mattu, and L. Kirchner, "How we analyzed the compas recidivism algorithm," 2016.

[105] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan, "A survey on bias and fairness in machine learning," *arXiv preprint arXiv:1908.09635*, 2019.

[106] A. Chouldechova and A. Roth, "A snapshot of the frontiers of fairness in machine learning,"

Communications of the ACM (CACM), 2020.

[107] B. Kim, H. Kim, K. Kim, S. Kim, and J. Kim, "Learning not to learn: Training deep neural networks with biased data," in *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*, 2019.

[108] T. Wang, J. Zhao, M. Yatskar, K.-W. Chang, and V. Ordonez, "Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations," *ICCV*, 2019.

[109] Y. Elazar and Y. Goldberg, "Adversarial removal of demographic attributes from text data," *EMNLP*, 2018.

[110] M. Arjovsky, L. Bottou, I. Gulrajani, and D. Lopez-Paz, "Invariant risk minimization," *arXiv preprint arXiv:1907.02893*, 2019.

[111] K. Ahuja, K. Shanmugam, K. Varshney, and A. Dhurandhar, "Invariant risk minimization games," in *International Conference on Machine Learning (ICML)*, 2020.

[112] K. K. Singh, D. Mahajan, K. Grauman, Y. J. Lee, M. Feiszli, and D. Ghadiyaram, "Don't judge an object by its context: Learning to overcome contextual bias," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[113] A. Zunino, S. A. Bargal, R. Volpi, M. Sameki, J. Zhang, S. Sclaroff, V. Murino, and K. Saenko, "Explainable deep classification models for domain generalization," *arXiv preprint arXiv:2003.06498*, 2020.

[114] L. Rieger, C. Singh, W. J. Murdoch, and B. Yu, "Interpretations are useful: penalizing explanations to align neural networks with prior knowledge," *ICML*, 2020.

[115] M. Pezeshki, S.-O. Kaba, Y. Bengio, A. Courville, D. Precup, and G. Lajoie, "Gradient starvation: A learning proclivity in neural networks," *arXiv preprint arXiv:2011.09468*, 2020.

[116] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," *International Conference on Learning Representations (ICLR)*, 2018.

[117] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *International Conference on Machine Learning (ICML)*, 2019.

[118] R. Kohavi, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid," in *KDD*, 1996.

[119] G. H. Bakır, J. Weston, and B. Schölkopf, "Learning to find pre-images," *Advances in neural information processing systems (NIPS)*, 2004.

[120] Z. Zhang and M. R. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," *Advances in Neural Information Processing Systems (NeurIPS)*, 2018.

[121] M. Feldman, S. A. Friedler, J. Moeller, C. Scheidegger, and S. Venkatasubramanian, "Certifying and removing disparate impact," in *KDD*, 2015.

[122] M. Hardt, E. Price, N. Srebro, *et al.*, "Equality of opportunity in supervised learning," in *NIPS*, 2016.

[123] S. B. Cohen, "Design strategies and innovations in the medical expenditure panel survey," *Medical care*, pp. III5–III12, 2003.

[124] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, December 2015.

[125] B. H. Zhang, B. Lemoine, and M. Mitchell, "Mitigating unwanted biases with adversarial learning," in *AIES*, 2018.

[126] Y. Bechavod and K. Ligett, "Penalizing unfairness in binary classification," *Fairness, Accountability and Transparency in Machine Learning (FAT/ML)*, 2017.

[127] H. Yuan, J. Tang, X. Hu, and S. Ji, "Xgnn: Towards model-level explanations of graph neural networks," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 430–438, 2020.

[128] H. Yuan, H. Yu, J. Wang, K. Li, and S. Ji, "On explainability of graph neural networks via subgraph explorations," *arXiv preprint arXiv:2102.05152*, 2021.

[129] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *International Conference on Learning Representa-*

tions (ICLR), 2021.

[130] I. Tolstikhin, N. Houlsby, A. Kolesnikov, L. Beyer, X. Zhai, T. Unterthiner, J. Yung, D. Keysers, J. Uszkoreit, M. Lucic, *et al.*, "Mlp-mixer: An all-mlp architecture for vision," *arXiv preprint arXiv:2105.01601*, 2021.

[131] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[132] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," *International Conference on Learning Representations (ICLR)*, 2020.

[133] L. Faber, A. K. Moghaddam, and R. Wattenhofer, "When comparing to ground truth is wrong: On evaluating gnn explanation methods," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 332–341, 2021.

[134] N. Liu, M. Du, R. Guo, H. Liu, and X. Hu, "Adversarial attacks and defenses: An interpretation perspective," *ACM SIGKDD Explorations Newsletter*, vol. 23, no. 1, pp. 86–99, 2021.

[135] Y. Bian, J. Huang, X. Cai, J. Yuan, and K. Church, "On attention redundancy: A comprehensive study," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 930–945, 2021.

[136] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel, "Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness," *International Conference on Learning Representations (ICLR)*, 2019.