Improving Interpretability via Explicit Word Interaction Graph Layer

Arshdeep Sekhon, Hanjie Chen, Aman Shrivastava, Zhe Wang, Yangfeng Ji, Yanjun Qi
University of Virginia, Charlottesville, USA
as5cu@virginia.edu, hc9mx@virginia.edu, as3ek@virginia.edu, zw6sg@virginia.edu, yj3fs@virginia.edu, yanjun@virginia.edu

Abstract

Recent NLP literature has seen growing interest in improving model interpretability. Along this direction, we propose a trainable neural network layer that learns a global interaction graph between words and then selects more informative words using the learned word interactions. Our layer, we call WiGraph, can plug into any neural network-based NLP text classifiers right after its word embedding layer. Across multiple SOTA NLP models and various NLP datasets, we demonstrate that adding the WiGraph layer substantially improves NLP models' interpretability and enhances models' prediction performance at the same time.

1 Introduction

Deep neural networks (DNNs) have achieved remarkable results in the field of natural language processing (NLP) (Zhang, Zhao, and LeCun 2015; Miwa and Bansal 2016; Wu et al. 2016; Wolf et al. 2020). Trustworthy real-world deployment of NLP models requires models to be not only accurate but also interpretable (Xie et al. 2020). Literature has included a growing focus on providing posthoc explanations or rationales for NLP models' predictions (Ribeiro, Singh, and Guestrin 2016; Lundberg and Lee 2017; Murdoch, Liu, and Yu 2018; Singh, Murdoch, and Yu 2018; Chen, Zheng, and Ji 2020). However, explaining DNNs using a posthoc manner cannot improve a model's intrinsic interpretability.

As shown in (Chen and Ji 2020), two NLP models may have the same prediction behavior but different interpretation ability. This concept of "intrinsic interpretability" motivates a compelling research direction to improve the interpretability of NLP models. A few recent studies used user-specified priors as domain knowledge to guide model training (Camburu et al. 2018; Du et al. 2019; Chen and Ji 2019; Erion et al. 2019; Molnar, Casalicchio, and Bischl 2019), hence improving model interpretability. Such information priors, however, may not be available in many tasks. Several other studies proposed to develop inherently interpretable models (Alvarez-Melis and Jaakkola 2018a; Rudin 2019), but these require intensive engineering efforts. More recently, Chen and Ji (2020) proposed to add a variational word mask, VMASK, to improve the interpretability of NLP neural classifiers.

The aforementioned literature on improving NLP models' intrinsic interpretability have mostly focused on highlighting important words. Strategies like VMASK just select important words unilaterally, without accounting for how one word influences other words regarding interpretability. Studies have shown that word interactions are critical in explaining how NLP models make decisions (Halford, Wilson, and Phillips 2010). For instance, for a sentiment classification task, when without a context, it is hard to conclude if the word *'different'* by itself is vital for sentiment? However, if we find 'different' highly relates to the word 'refreshingly', it will likely contribute substantially to the model's sentiment prediction (see Table 1 [1]).

Along this direction, we propose a novel neural network layer, we call WiGraph, to improve NLP models' intrinsic interpretability. WiGraph is a plug-and-play layer and uses a graph-oriented neural network design: (1) It includes a stochastic edge discovery module that can discover significant interaction relations between words for a target prediction task; (2) It then uses a neural message passing module to update word representations by using information from their interacting words; and (3) It designs a variational information bottleneck based loss objective to suppress irrelevant word interactions (regarding target predictions). We call such a loss: VIB-WI loss. To improve a target text classifier's intrinsic interpretability, we propose to add the proposed WiGraph layer right after the word embedding layer and fine-tune such an augmented model using a combination of the original objective and VIB-WI loss objective.

In summary, this paper makes the following contributions:

- We design WiGraph to augment neural text classifiers to improve these models' intrinsic interpretability. WiGraph does not require external user priors or domain knowledge. WiGraph can plug-and-play into any neural NLP models' architectures, right after the word embedding layer.
- We provide extensive empirical results showing that adding WiGraph layer into SOTA neural text classifiers results in better explanations (locally, globally as well as regarding interactions) and better model predictions at the

[1]These attribution interpretations were generated by LIME (Ribeiro, Singh, and Guestrin 2016) and use BERT-base model on SST-2 dataset to explain two models' predictions.

Model	Explanation
BASE	still , this thing feels flimsy and ephemeral
WIGRAPH	still , this thing feels flimsy and ephemeral
BASE	so young , so smart , such talent , such a wise
WIGRAPH	so young , so smart , such talent , such a wise
BASE	it is risky , intelligent , romantic and rapturous from start to finish
WIGRAPH	it is risky , intelligent , romantic and rapturous from start to finish
BASE	take care of my cat offers a refreshingly different slice of asian cinema
WIGRAPH	take care of my cat offers a refreshingly different slice of asian cinema

Table 1: Top ranked important words are shown in pink for BASE and blue for WIGRAPH augmented BASE model. We can tell that word attributions from WIGRAPH augmented model are easier to understand, and highlight more relevant sentiment words. This indicates WIGRAPH augmented model has better intrinsic interpretibility than BASE.

same time.

2 Method: A Novel WIGRAPH Layer

Our main hypothesis is: a novel layer that can extract crucial global word interactions will improve neural text classifiers' interpretability. This is because we envision plugging such a layer will enhance a target model's decision-making process by providing explicit guidance on what words are more important using the information on those words they interact with. We aim for three properties in such a layer design: (1) plug-and-play; (2) model agnostic; and (3) no loss of prediction performance.

We denote vectors using lowercase bold symbols. We assume text inputs include a maximum length of L tokens. We denote the whole word vocabulary as set $\mathbb{V}$. We use V to denote its size (the total number of unique words in this vocabulary $\mathbb{V}$). Besides, we use f to describe a neural text classification model. f classifies an input text into $\mathbf{y} \in \{1, \dots, C\}$, where C is the number of classes. For an input sentence, we denote its i-th word as w_i and its embedding representation as vector $\mathbf{x}_i$: $\forall i \in \{1, \dots, L\}$. Therefore, the embeddings of an input text make a matrix form $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_L]^T$ (this means $\mathbf{X} \in \mathbb{R}^{L \times d}$ where L is the length of input and d is the dimension of each $\mathbf{x}_i$.).

2.1 To Discover Word Interaction Graph: A

Now we explain the first component of the proposed WIGRAPH layer. This module aims to discover how words globally interact for a predictive task. Our primary strategy to describe how words relate is to treat words as nodes and their interaction as edges in an interaction graph. We follow such an idea and choose to learn an undirected word interaction graph using a stochastic neural network module. We represent this unknown graph as $\mathbf{A} = \{\mathbf{A}_{ij}\}_{V \times V}$. $\mathbf{A}$ includes the edges representing word interactions. We assume each $\mathbf{A}_{ij} \in \{0, 1\}$ is one binary random variable. $\mathbf{A}$ is stochastic whose $\mathbf{A}_{ij}$ specifies the presence or absence of an interaction between word i and word j in vocabulary $\mathbb{V}$. $\mathbf{A}_{ij} \in \{0, 1\}$ is sampled from $Sigmoid(\gamma_{ij})$, following Bernoulli distribution with parameter $Sigmoid(\gamma_{ij})$. In Section 2.2, we show how $\mathbf{A}$ can help us understand how certain words are more important than others owing to the learned word interactions.

Learning the word interaction graph $\mathbf{A}$ means to learn the parameter matrix $\gamma = \{\gamma_{ij}\}_{V \times V}$. In Section 2.4, we show how γ (and therefore $\mathbf{A}$) is learned through the variational information bottleneck framework(Alemi et al. 2016).

2.2 Message Passing On Word Interaction Graph Using Graph Convolution: $\mathbf{E}'$

In our second module, we represent the i-th word $\mathbf{x}_i$ of an input text $\mathbf{x}$ as a node on the $\mathbf{A}$ graph. We use a modified version of graph convolutional operation (Kipf and Welling 2016) to update each $\mathbf{x}_i$ with its neighboring words $\mathbf{x}_j$. Here $j \in \mathcal{N}(i)$, and $\mathcal{N}(i)$ denotes those neighbor nodes of $\mathbf{x}_i$ on the graph $\mathbf{A}$ and in $\mathbf{x}$. Specifically, we denote the resulting word representation vector as $\mathbf{e}'_i$. Each $\mathbf{x}_i$ is revised using a graph based summation from its neighbors' embedding $\mathbf{x}_j, j \in \mathcal{N}(i)$:

$$\mathbf{e}'_i = \mathbf{x}_i + \sigma\left(\frac{1}{|\mathcal{N}(i)|} \sum_{j \in \mathcal{N}(i)} \mathbf{x}_j\right), \tag{1}$$

Eq. (1) is motivated by the design of Graph convolutional networks (GCNs) that were recently introduced to learn useful node representations that encode both node-level features and relationships between connected nodes (Kipf and Welling 2016). Different from the ReLU activation function used in vanilla GCNs, we use GeLU as the $\sigma(\cdot)$, the non-linear activation function proposed in (Hendrycks and Gimpel 2016).

We want to point out that Eq. (1) is different from a typical GCN operation from (Kipf and Welling 2016). First, we only conduct one hop of neighbor aggregation in Eq. (1). A typical GCN module does multi-hops. Second, we drop $\mathbf{W}^t \in \mathbb{R}^{d \times d}$ used for t-th hop of GCN update in (Kipf and Welling 2016). This is because we assume that the BASE text classifier model f has taken into account this prior and our WIGRAPH layer will not bias to prefer short range interactions. The third difference is the most important distinction that differentiates ours apart from (Kipf and Welling 2016). The graph has been given apriori to typical GCNs. However, in our work, we need to learn the graph $\mathbf{A}$ (see Section 2.4 on how to learn $\mathbf{A}$).

We can compute the simultaneous update of all words in input text $\mathbf{x}$ together by concatenating all $\mathbf{e}'_i$. This gives us one matrix $\mathbf{E}' \in \mathbb{R}^{L \times d}$, where L is the length of input and d is the embedding dimension of each $\mathbf{x}_i$. The simultaneous update can be written as:

$$\mathbf{E}' = \sigma(\mathbf{A}'\mathbf{X}). \tag{2}$$

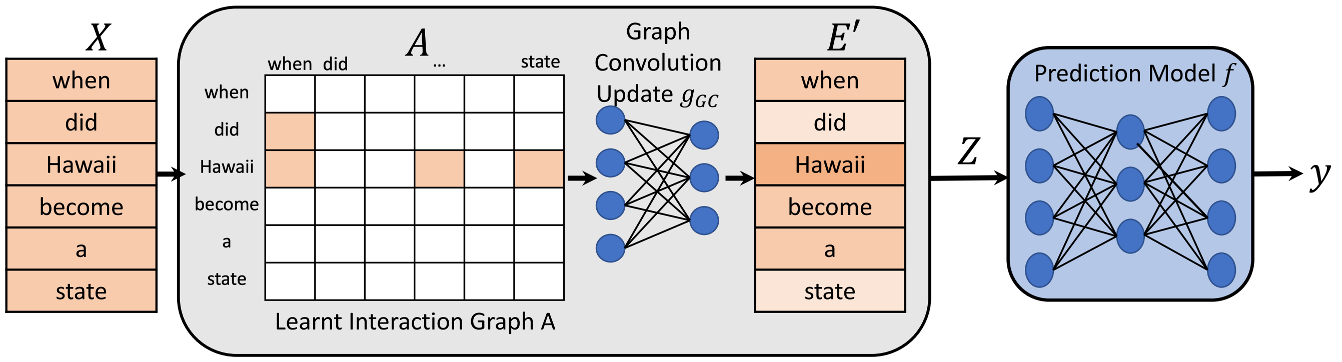

Figure 1: WIGRAPH layer (components inside the gray box): during inference, embeddings of words (for example, *Hawaii* and *state*) are aggregated based on their interactions using a modified Graph Convolutional operation. Here graph $\mathbf{A}$ was learnt from training along with the prediction task. A WIGRAPH layer is inserted into a neural text classifier right after the word embedding input layer.

where $\mathbf{A}' = \hat{\mathbf{D}}^{-\frac{1}{2}}(\mathbf{A_x} + \mathbf{I})\hat{\mathbf{D}}^{-\frac{1}{2}}$, that is the normalized adjacency matrix and $\hat{\mathbf{D}}$ is the diagonal degree matrix of $(\mathbf{A_x} + \mathbf{I})$. Note: $\mathbf{A_x}$ denotes those edges from $\mathbf{A}$ that are local for the current sample text $\mathbf{x}$. In summary, our second module computes:

$$\mathbf{E}' = g_{GC}(\mathbf{X}, \mathbf{A})$$

2.3 To Build And Use WIGRAPH Layer: $\mathbf{X} \rightarrow \mathbf{Z}$

Our design of WIGRAPH layer is that it can take the embedding matrix of a text example as input ($\mathbf{X} \in \mathbb{R}^{L \times d}$), and output a revised matrix representing each word with revised embedding ($\mathbf{Z} \in \mathbb{R}^{L \times d}$). WIGRAPH layer aids the selection of more informative words based on their interactions for current predictive task. The proposed layer does not need significant efforts on engineering network architectures and does not require pre-collected importance attributions or explanations.

Our main goal is to improve the intrinsic interpretability of neural text classifiers with a simple model augmentation. Therefore, for a given neural text classifier, we propose to simply insert a WIGRAPH layer right after the word embedding input layer and before the subsequent network layers of that target model.

There exist many possible ways to build WIGRAPH layer from our first two modules (Section (2.1) and Section (2.2)). The simplest way is that we can just pass $\mathbf{E}'$ as $\mathbf{Z}$.

$$\mathbf{Z} = \mathbf{E}' \tag{3}$$

Figure 1 visualizes how this vanilla version of WIGRAPH layer updates word representations with the $\mathbf{X} \rightarrow \mathbf{E}' \rightarrow \mathbf{Z}$ data flow during inference. During training, it needs to learn the graph $\mathbf{A}$.

2.4 Model Training With VIB-WI Loss

Now we propose to train WIGRAPH jointly with other layers using a new objective that we name as variational information bottleneck loss for word interaction (VIB-WI loss). VIB-WI loss aims to restrict the information of globally irrelevant word interactions flowing to subsequent network layers, hence forcing the model to focus on important interactions to make predictions. Following the Information Bottleneck framework (Alemi et al. 2016), we aim to learn $\mathbf{A}$ and all subsequent layers' weights $\{\mathbf{W}\}$, to make $\mathbf{Z}$ maximally informative of the prediction label $\mathbf{Y}$, while being maximally compressive of $\mathbf{X}$ (see Figure 1). That is

$$max_{\mathbf{A},\{\mathbf{W}\}}\{I(\mathbf{Z};\mathbf{Y}) - \beta I(\mathbf{Z};\mathbf{X})\} \tag{4}$$

Here $I(\cdot;\cdot)$ denotes the mutual information, and $\beta \in \mathbb{R}_+$ is a coefficient balancing the two mutual information terms.

Given a specific example $(\mathbf{x}^m, \mathbf{y}^m)$, we can further simplify the lower bound of first term $I(\mathbf{Z};\mathbf{Y})$ in Eq. (4) as:

$$I(\mathbf{z};\mathbf{y}^m) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^m)} log(p(\mathbf{y}^m|\mathbf{x}^m;\mathbf{A},\{\mathbf{W}\})) \tag{5}$$

Similarly for the second term $I(\mathbf{Z};\mathbf{X})$ in Eq. (4) and for a given example $(\mathbf{x}^m, \mathbf{y}^m)$, we can simplify its upper bound as:

$$I(\mathbf{z};\mathbf{x}^m) \leq KL(q(\mathbf{A}|\mathbf{x}^m)||p_{a0}(\mathbf{A})) \tag{6}$$

Due to the difficulty in calculating two mutual information terms in Eq. (4), we follow (Schulz et al. 2020; Alemi et al. 2016) to use a variational approximation $q(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ to approximate the true distribution $p(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$. Details on how to derive Eq. (5) and Eq. (6) are in Section (5). Now combining Eq. (5) and Eq. (6) into Eq. (4), we get the revised objective as:

$$\begin{aligned} max_{\mathbf{A},\mathbf{R},\{\mathbf{W}\}}\{&\mathbb{E}_{q(\mathbf{Z}|\mathbf{x}^m)} log(p(\mathbf{y}^m|\mathbf{x}^m;\mathbf{A},\mathbf{R},\{\mathbf{W}\})) \\ &-\beta_g KL(q(\mathbf{A}|\mathbf{x}^m)||p_{a0}(\mathbf{A}))\} \end{aligned} \tag{7}$$

Eq. (7) is the proposed VIB objective for a given observation $(\mathbf{x}^m, \mathbf{y}^m)$.

Detailed Model Specification: During training, for the stochastic interaction graph $\mathbf{A}$, we learn its trainable parameter matrix $\gamma \in \mathbb{R}^{|V| \times |V|}$, that is also optimized along with the model parameters during training. Further, we use the mean field approximation (Blei, Kucukelbir, and McAuliffe 2017), that is, $q(\mathbf{A_x}|\mathbf{x}) = \prod_{i=1}^{L}\prod_{j=1}^{L} q(A_{x_i,x_j}|\mathbf{x}_i, \mathbf{x}_j)$.

Equation 7 requires prespecified prior distributions p_{a0}. We use a Bernoulli distribution prior (a non-informative prior) for each word-pair interaction $q_\phi[\mathbf{A}_{x_i,x_j}|\mathbf{x}_i,\mathbf{x}_j]$. $p_{a0}(\mathbf{A}_x) = \prod_{i=1}^{L}\prod_{j=1}^{L} p_{a0}(\mathbf{A}_{\mathbf{x}_i,\mathbf{x}_j})$ and $p_{a0}(\mathbf{A}_{x_i,x_j}) = Bernoulli(0.5)$. This leads to:

$$KL(q(\mathbf{A}_\mathbf{x}|\mathbf{x}^m)||p_{a0}(\mathbf{A})) = -H_\mathbf{q}(\mathbf{A}_\mathbf{x}|\mathbf{x}^m) \quad (8)$$

Here, H_q denotes the entropy of the term $\mathbf{A}_\mathbf{x}|\mathbf{x}^m$ under the q distribution. Besides, we add a sparsity regularization on $\mathbf{A}_\mathbf{x}$ to encourage learning of sparse interactions. Now, we have the following loss function for $(\mathbf{x}^m, \mathbf{y}^m)$:

$$-(\mathbb{E}_\mathbf{x} p(\mathbf{y}|\mathbf{x}^m; \mathbf{A}, \{\mathbf{W}\}) + \beta_g H_\mathbf{q}(\mathbf{A}_\mathbf{x}|\mathbf{x}^m)) + \beta_{sparse}||\mathbf{A}_\mathbf{x}||_1 \quad (9)$$

During training, $\mathbf{A}$ is discrete and is drawn from Bernoulli distributions that are parametrized with matrix $\gamma \in \mathbb{R}^{|V|\times|V|}$. We, therefore, use the Gumbel-Softmax(Jang, Gu, and Poole 2016) trick to differentiate through the sampling step and propagate the gradients to the respective parameters γ.

2.5 Variation: WIGRAPH-A-R

We also try another possible design of WIGRAPH that includes one more separate module that learns an attribution word mask $\mathbf{R}$ on top of $\mathbf{E}'$. Aiming for better word selection, $\mathbf{R}$ is designed as a stochastic layer we need to learn and $\mathbf{R} \in \{0,1\}^V$. Each entry in $\mathbf{R}$ (e.g., $\mathbf{R}_j \in \{0,1\}$) follows a Bernoulli distribution with parameter ϕ (to be learned).

During inference, for an input text $\mathbf{x}$, we get a binary vector $\mathbf{R}_\mathbf{x}$ from $\mathbf{R}$ that is of size L. Its i-th entry $\mathbf{R}_{\mathbf{x}_i} \in \{0,1\}$ is a binary random variable associated with the word token at the i-th position. We use the following operation (a masking operation!) to generate the final representation of the i-th word from a WIGRAPH layer:

$$\mathbf{z}_i = \mathbf{R}_{\mathbf{x}_i}\mathbf{e}'_i \quad (10)$$

We can compute the simultaneous update of all words in input text $\mathbf{x}$ together by concatenating all $\mathbf{z}_i$ denoted as matrix $\mathbf{Z} \in \mathbb{R}^{L\times d}$. The simultaneous update can then be written as:

$$\mathbf{Z} = \text{diag}\,(\mathbf{R}_\mathbf{x})_{L\times L}\,\mathbf{E}'_{L\times d} \quad (11)$$

During training, we need to learn both $\mathbf{A}$ and $\mathbf{R}$. Now the loss function VIB-WI loss turns to:

$$-(\mathbb{E}_\mathbf{x} p(\mathbf{y}|\mathbf{x}^m; \mathbf{A}, \mathbf{R}, \{\mathbf{W}\}) + \beta_i H_q(\mathbf{R}_\mathbf{x}|\mathbf{x}^m) + \beta_g H_\mathbf{q}(\mathbf{A}_\mathbf{x}|\mathbf{x}^m)) + \beta_{sparse}||\mathbf{A}_\mathbf{x}||_1$$

Due to page limit, we put detailed derivations of above and specification of $\mathbf{R}$ in Section (5). We call the vanilla version of WIGRAPH as **WIGRAPH-A** and the version with the word mask R as WIGRAPH-A-R.

3 Connecting To Related Work

Our design orients from one basic notion that we treat interpretability as an intrinsic property of neural network models. We expect a neural text classifier will be more interpretable, when focusing on important word interactions to make predictions. Our work connects to multiple related topics:

Self-Explaining Models Recent literature has seen growing interests in treating interpretability as an *inherent property* of NLP deep models. (Alvarez-Melis and Jaakkola 2018b; Rudin 2019) proposed to design self-interpretable models by requiring human annotations in model engineering. (Chen and Ji 2020) proposed VMASK layer for improving NLP models' interpretability. This layer automatically learns task-specific word importance and guides a model to make predictions based on important words. However, this method does not consider interactions between words.

Explanations as Feedback Anoter category of work uses explanations as feedback for improving model prediction performance as well as to encourage explanation faithfulness. (Camburu et al. 2018; Chen and Ji 2019; Erion et al. 2019; Molnar, Casalicchio, and Bischl 2019) focuses on aligning human judgments with generated explanations and further incorporating it into the training of the model. These methods require human annotations that are expensive to obtain and also have the risk of not aligning well with the separately trained model's decision making process. (Ross, Hughes, and Doshi-Velez 2017; Ross and Doshi-Velez 2017; Rieger et al. 2020) use explanations as feedback into the model to improve prediction performance. However, these heavily rely on ground-truth explanations and domain knowledge. Differently, our proposed method augments a model with a special layer that improves both prediction performance and interpretability (see Section (4)).

Graph Neural Networks Graph Neural Networks (GNNs) generalize neural networks from regular grids, like images to irregular structures like graphs. There exists a wide variety of GNN architectures like (Kipf and Welling 2016; Scarselli et al. 2008; Veličković et al. 2017; Santoro et al. 2017). They share the same underlying concept of message passing between connected nodes in the graph. However, little attention has been paid to address cases when the underlying graph is unknown. In contrast, in WIGRAPH, we do not know the global interaction graph *apriori*. It is learnt along with the prediction model as part of the training.

Post-Hoc Explanation NLP literature includes a number of methods that focus on disentangling the rationales of a trained NLP model's decision by finding which words contributed most to a prediction, including the popularly used LIME(Ribeiro, Singh, and Guestrin 2016) and SampleShapley (Kononenko et al. 2010) methods. Recent studies have proposed to generate post-hoc explanations beyond word-level features by detecting feature interactions, including for instance, contextual decomposition by (Murdoch, Liu, and Yu 2018). Other work adopted Shapley interaction index to compute feature interactions (Lundberg, Erion, and Lee 2018). In contrast to these post-hoc interpretation systems, our method focuses on designing a strategy to improve the inherent interpretability of NLP models.

Information Bottleneck Based Methods The information bottleneck method was first proposed by (Tishby, Pereira, and Bialek 2000; Tishby and Zaslavsky 2015). (Alemi et al. 2016) introduced a variational approximation to the information bottleneck that enables usage for deep neural networks.

BASE	Models	IMDB	SST-1	SST-2	AG News	TREC	Subj
LSTM	BASE	88.39	43.84	83.74	91.03	90.40	90.20
	VMASK	90.07	44.12	84.35	92.19	90.80	91.20
	WIGRAPH	**90.12** **+1.73**	**46.47** **+2.63**	**86.21** **+2.63**	**91.16** +0.13	**92.20** **+1.80**	**91.40** **+1.20**
BERT	BASE	91.88	51.63	92.15	92.05	97.40	96.40
	VMASK	93.04	51.36	92.26	94.24	97.00	96.40
	WIGRAPH	**92.48** **+0.60**	**52.49** **+0.86**	**92.59** +0.44	**92.72** **+0.67**	**97.40** +0.00	**96.60** +0.20
RoBERTa	BASE	89.87	55.20	94.73	93.46	96.2	96.00
	VMASK	90.02	54.21	93.47	93.47	96.0	96.50
	WIGRAPH	**90.10** +0.23	**55.52** +0.32	**94.75** +0.02	**93.52** +0.06	**96.60**+0.20	**96.40** +0.40
distilBERT	BASE	86.96	51.31	90.50	93.34	97.20	96.20
	VMASK	87.00	48.01	89.02	93.81	95.20	95.00
	WIGRAPH	**88.32** **+1.36**	50.81	**90.77** +0.22	**93.85** **+0.51**	**97.40** +0.20	**96.30** +0.10

Table 2: Prediction Accuracy (%). Models augmented with WIGRAPH layer predict better than BASE.

Dataset	Train/Dev/Test	C	V	L
sst1	8544/1101/2210	5	17838	50
sst2	6920/872/1821	2	16190	50
imdb	20K/5K/25K	2	29571	250
AG News	114K/6K/7.6K	4	21838	50
TREC	5000/452/500	6	8026	15
Subj	8000/1000/1000	2	9965	25

Table 3: Summary of datasets we use in experiments: number of classes (C), vocabulary size (V) and sentence length (L).

(Schulz et al. 2020; Bang et al. 2019) utilized the information bottleneck principle to generate post-hoc explanations by highlighting important features while suppressing unimportant ones. Differently, we incorporate the information bottleneck in model training to make model prediction behavior more interpretable.

4 Experiments

We design experiments to answer the following:

1. Are NLP models augmented with WIGRAPH layer more interpretable models?
2. Do NLP models augmented with WIGRAPH layer predict well?

Besides, we extend WIGRAPH to one concept based vision task in Section 4.5.

4.1 Setup: Datasets, Models And Metrics

Datasets Our empirical analysis covers six popular text classification datasets as detailed by Table 3. These six datasets are "sst1", "sst2"(Socher et al. 2013), "imdb"(Maas et al. 2011), "AG news"(Zhang, Zhao, and LeCun 2015), "TREC"(Li and Roth 2002) and "Subj"(Pang and Lee 2005). Three of the datasets are for binary classification, and the rest are for multi-class text classification tasks.

BASE Models We use four commonly used neural text classifiers to evaluate WIGRAPH: LSTM, and transformer based SOTA models including BERT, RoBERTa and distilBERT. As Section 2.4 described, we plug our WIGRAPH layer right after the word embedding input layer. For the LSTM models(Hochreiter and Schmidhuber 1997), we initialize word embeddings from (Mikolov et al. 2013) with dimension $d = 300$. For BERT, RoBERTa and distilBERT models, we use base models from (Wolf et al. 2020).

Hyperparameter Tuning We perform fine-tuning on each model (batch size=64). We fix the word embedding layer and train WIGRAPH layer along with the rest of a BASE model. For the LSTM models, we vary the hidden size $\in \{100, 300, 500\}$, and dropout in $\{0.0, 0.2, 0.3\}$. We set $\beta_{sparse} \in \{1e-02, 1e-03, 1e-04\}$, $\beta_g \in \{1.0, 1e-02, 1e-03, 1e-04\}$ and $\beta_i \in \{1.0, 1e-02, 1e-03, 1e-04\}$. The learning rate is tuned from the set $\{0.0001, 0.0005, 0.005, 0.001\}$. For transformer based models, we vary dropout in range $\{0.2, 0.3, 0.5\}$, hidden dimension to compute $\mathbf{R} \in \{128, 256, 512\}$. We set $\beta_{sparse}, \beta_g, \beta_i = 1.0$ and anneal it by a factor of 0.1 every epoch. For the larger vocabulary cases (IMDB, AG-News datasets and transformer-base models), we filter words for learning our interaction matrix $\mathbf{A}$, i.e., we learn interactions for the top frequent $10,000$ words.

Baselines: To our best knowledge, WIGRAPH is the only plug-and-play layer to improve a target neural text classifier's interpretability using explicit pairwise word interactions. In our experiments, we compare WIGRAPH to a BASE model without WIGRAPH layer and to a BASE model augmented by the VMASK layer. (Chen and Ji 2020) proposed VMASK layer for improving NLP models' intrinsic interpretability, though this layer does not consider word interactions.

Evaluation Metrics: We use three types of evaluations to compare WIGRAPH with baselines. (a) Prediction accuracy: this is to measure if NLP models augmented with WIGRAPH layer predict well. (b) To compare different models' interpretability, we will apply two post-hoc attribution techniques: LIME(Ribeiro, Singh, and Guestrin 2016) and SampleShapley (Kononenko et al. 2010) on model predictions. The resulting feature attribution outputs will be evaluated using

Metrics	BASE	Models	IMDB	SST-1	SST-2	AG News	TREC	Subj
AOPCs of LIME Generated Explanations	LSTM	BASE	14.34	8.76	17.03	7.00	11.95	9.67
		VMASK	15.1	9.52	22.14	7.39	11.97	11.68
		WIGRAPH-A	**17.8**	**10.33**	**22.34**	**14.94**	**20.13**	**16.27**
	BERT	BASE	10.63	26.08	43.96	7.12	**68.82**	44.13
		VMASK	**12.64**	27.5	41.6	8.47	65.14	44.41
		WIGRAPH-A	10.96	**34.81**	**44.59**	**11.13**	68.51	**44.90**
	RoBERTa	BASE	**10.30**	32.00	41.51	15.7	66.07	43.00
		VMASK	9.88	26.02	40.69	8.47	64.58	43.66
		WIGRAPH-A	10.23	**32.70**	**42.64**	**15.72**	**66.27**	**44.22**
	distilBERT	BASE	11.00	27.92	42.25	8.83	67.63	**44.93**
		VMASK	9.20	22.11	39.77	8.00	63.14	40.65
		WIGRAPH-A	**11.32**	**36.21**	**44.33**	**9.02**	**68.74**	43.97
AOPCs of SampleShapley Generated Explanations	LSTM	BASE	15.80	7.91	22.38	6.62	11.90	11.66
		VMASK	16.48	9.73	22.52	7.65	11.86	12.74
		WIGRAPH-A	**21.73**	**9.78**	**49.40**	**27.60**	**68.27**	**29.29**
	BERT	BASE	13.00	28.65	41.65	7.21	65.37	33.22
		VMASK	12.18	29.92	41.53	10.02	65.14	44.41
		WIGRAPH-A	**14.16**	**36.02**	**44.75**	**10.43**	**66.30**	**44.91**
	RoBERTa	BASE	**9.13**	36.01	35.89	6.01	66.07	13.08
		VMASK	8.00	29.89	42.64	5.68	54.85	**43.75**
		WIGRAPH-A	8.50	**38.01**	**42.78**	**7.12**	**67.56**	43.61
	distilBERT	BASE	12.03	28.07	42.24	15.09	47.78	13.08
		VMASK	9.26	18.67	35.03	13.95	59.53	40.93
		WIGRAPH-A	**12.18**	**35.29**	**43.14**	**14.21**	**66.08**	**43.97**

Table 4: AOPCs (%) obtained from results using LIME and SampleShapley to interpret the base, WiGraph-based models and the baseline VMASK across four SOTA models, and over six different datasets.

explanation faithfulness scores like AOPCs (details in Section (4.3)). (3) We further design interaction interpretability measures to compare different models in Section (4.4).

4.2 Prediction Performance Comparison

In Table 2, we compare prediction performance using four different SOTA models across six different datasets. This makes 24 different (BASE, data) combinations , and on each case, we compare BASE model, VMASK augmented base model versus our WiGraph augmented model regarding the prediction accuracy. Here we refer to WiGraph as the best performing model between WIGRAPH-A and WIGRAPH-A-R. Table 2 shows that adding WiGraph layer into SOTA neural text classifier models makes the models predict better! Empirically, the performance gains on LSTM models appear more than on Transformer models.

4.3 Attribution Interpretability Comparison: Area Under Perturbation Curve (AOPC)

Here we empirically check our hypothesis that training a model augmented with WiGraph layer leads to improvements of model explanation faithfulness during downstream post-hoc interpretation analyses. We use Area Over Perturbation Curve (AOPC) (Nguyen 2018; Samek et al. 2016) as the evaluation metric. AOPC is defined as the *average change of prediction probability on the predicted class over a test dataset by deleting top K words in explanations.* Higher AOPC scores reflect better interpretation faithfulness.

$$AOPC = \frac{1}{K+1}\sum_{k=1}^{K} < f(\mathbf{x}) - f(\mathbf{x}_{\setminus 1,\ldots,k}) >_{p(\mathbf{x})} \quad (12)$$

We generate word-level attribution explanations using two popular post-hoc explanation methods: LIME(Ribeiro, Singh, and Guestrin 2016) and SampleShapley (Kononenko et al. 2010). Across all datasets, we use 500 test samples and $k \in \{1, \ldots, 10\}$. Table 4 shows that WiGraph-A outperforms the original BASE model and the BASE with VMASK.

When LIME is used to post-hoc explain models, across all 24 cases of (model, dataset) combinations, WiGraph outperforms the original BASE model and the BASE with VMASK layer regarding AOPC score in 21 cases. The only three exception include the IMDB/BERT, TREC/BERT and IMDB/RoBERTa setups. When SampleShapley is used, WiGraph outperforms the original BASE model and the BASE with VMASK layer in 22 cases out of 24 (model, dataset) combinations.

4.4 Interaction Analysis And Ablation

In this experiment, we introduce a new metric: *Interaction Occlusion Score* (IoS). The IoS score measures the interaction interpretability faithfulness of a target model on its learnt interactions.

WiGraph discovers globally informative interactions with the importance score $\mathbb{E}_q[\mathbf{A}_{x_{i,j}}|\mathbf{x}_{i,j}]$ (see q in Section (2.4)). We sort entries of $\mathbf{A}$ and filter out the top K global interaction scores, denoted by $\mathbf{A}^k_{ij}$. We then calculate the accuracy of the model after only using these top k interactions via:

$$\text{IOS}(k) = \frac{1}{M}\sum_{m=1}^{M} 1_{y_m = y^k_m} \quad (13)$$

BASE	Models	IMDB	SST-1	SST-2	AG News	TREC	Subj
LSTM	WIGRAPH-A-R	89.12	44.32	84.08	**91.16**	91.65	90.60
	WIGRAPH-A	**90.12**	**46.47**	**86.21**	90.87	**92.20**	**91.40**
BERT	WIGRAPH-A-R	90.81	**52.49**	**92.59**	90.13	96.60	96.40
	WIGRAPH-A	**92.48**	52.04	91.54	**92.72**	**97.40**	**96.60**
RoBERTa	WIGRAPH-A-R	**90.10**	52.90	92.97	91.54	95.20	95.50
	WIGRAPH-A	88.21	**54.52**	**94.45**	**93.52**	**96.60**	**96.40**
distilBERT	WIGRAPH-A-R	**88.32**	**50.81**	88.19	**93.85**	96.40	96.20
	WIGRAPH-A	88.07	49.95	**90.77**	91.08	**97.40**	**96.30**

Table 5: Ablations analysis regarding prediction accuracy from: WIGRAPH-A and WIGRAPH-A-R.

Here, we represent the label of the model on the m^{th} test sample as y_m.

Ablation: We perform extensive ablation analysis to compare WIGRAPH-A and WIGRAPH-A-R. Table 5 provides comparison analysis regarding prediction accuracy and we can tell no clear winner between the two variations across all 24 cases of (BASE, data) combinations. See Table 6 for more ablation results via other metrics. We recommend to use WIGRAPH-A in most real-world applications, due to less parameters.

4.5 Modeling Concept Interaction In Vision

Koh et al. (2020) introduced the notion of high-level concepts as an intermediate interpretable interface in the input-model-predictions pipeline. These concepts describe high level attributes of an image. This enables users to directly interact with a model by intervening on human-interpretable concepts. The interactions between these concepts can affect prediction, however these interactions are unknown. In this section, we investigate the utility of learning these interactions for aiding prediction using our WiGraph layer. We train a concept bottleneck (Koh et al. 2020) model on the CUB dataset (Wah et al. 2011) which is jointly trained with a WiGraph layer. We show that the WiGraph layer is able to learn concept embeddings, model interactions between concepts, and can also be used with test time concept intervention to improve prediction accuracy on the final task. In order to extend WiGraph for this setup, we introduce a dynamic interaction graph with concepts and the image as nodes. Intuitively, the image is considered to be a composition of concept embeddings. The interaction between the concepts are learned variables whereas the interactions between an image and its concepts are computed using the cosine similarity.

Results: As a baseline, we fine-tune an Inception V3-based joint concept bottleneck model (Koh et al. 2020) that achieve a prediction accuracy of 80.04% on the CUB dataset (Wah et al. 2011). Together with this model, we jointly train our WiGraph and the concept embedding layer to learn the interactions between the concepts. As described in (Koh et al. 2020), test time intervention (TTI) helps improve prediction accuracy to $89.41\%(+9.37\%)$. Interestingly, we observe that TTI achieves more improvements of prediction accuracy when using WiGraph augmented concept vision model to $95.74\%(+15.70\%)$.

5 Conclusions

In this paper, we try to answer the question: *Does adding a special layer in the form of discovering word-word interactions lead to improvements in a neural text classifier's interpretability?* Our paper gives a firm "Yes" to the question and provides a neural-network based design for making such a layer. The second component of WiGraph layer uses the message passing framework and it can be expanded to allow for learning and accounting for higher-order interactions (not only pairwise) in a scalable way. We will explore this in our future works. Furthermore, WiGraph can easily extend to cross sentence tasks like Natural Language Inference, and we will leave it to future.

A Model Training For WiGraph-A-R

We propose to train WiGraph jointly with other layers using a new objective that we name as variational information bottleneck loss for word interaction (VIB-WI loss). VIB-WI loss aims to restrict the information of globally irrelevant word interactions flowing to subsequent network layers, hence forcing the model to focus on important interactions to make predictions. Following the Information Bottleneck framework (Alemi et al. 2016), we aim to learn $\mathbf{A}$, $\mathbf{R}$ and all subsequent layers' weights $\{\mathbf{W}\}$, to make $\mathbf{Z}$ maximally informative of $\mathbf{Y}$, while being maximally compressive of $\mathbf{X}$ (see Figure 1). That is

$$max_{\mathbf{A},\mathbf{R},\{\mathbf{W}\}}\{I(\mathbf{Z};\mathbf{Y})-\beta I(\mathbf{Z};\mathbf{X})\} \tag{14}$$

Here $I(\cdot;\cdot)$ denotes the mutual information, and $\beta \in \mathbb{R}_+$ is a coefficient balancing the two mutual information terms.

Given a specific example $(\mathbf{x}^m, \mathbf{y}^m)$, we can further simplify the lower bound of first term $I(\mathbf{Z};\mathbf{Y})$ in Eq. (14) as:

$$I(\mathbf{z};\mathbf{y}^m) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^m)} log(p(\mathbf{y}^m|\mathbf{x}^m;\mathbf{A},\mathbf{R},\{\mathbf{W}\})) \tag{15}$$

Similarly for the second term $I(\mathbf{Z};\mathbf{X})$ in Eq. (14) and for a given example $(\mathbf{x}^m, \mathbf{y}^m)$, we can simplify its upper bound as:

$$\begin{aligned} I(\mathbf{z};\mathbf{x}^m) \leq\ & KL(q(\mathbf{R}|\mathbf{x}^m)||p_{r0}(\mathbf{R})) \\ & + KL(q(\mathbf{A}|\mathbf{x}^m)||p_{a0}(\mathbf{A})) \end{aligned} \tag{16}$$

Due to the difficulty in calculating two mutual information terms in Eq. (14), we follow (Schulz et al. 2020; Alemi et al. 2016) to use a variational approximation $q(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ to approximate the true distribution $p(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$. Details on how to derive Eq. (15) and Eq. (16) are in Section (5). Now combining Eq. (15) and Eq. (16) in Eq. (14), we get the revised objective as:

$$\begin{aligned} max_{\mathbf{A},\mathbf{R},\{\mathbf{W}\}}\{&\mathbb{E}_{q(\mathbf{Z}|\mathbf{x}^m)} log(p(\mathbf{y}^m|\mathbf{x}^m; \mathbf{A}, \mathbf{R}, \{\mathbf{W}\})) \\ &-\beta_i KL(q(\mathbf{R}|\mathbf{x}^m)||p_{r0}(\mathbf{R})) \\ &-\beta_g KL(q(\mathbf{A}|\mathbf{x}^m)||p_{a0}(\mathbf{A}))\} \end{aligned} \tag{17}$$

Eq. (17) is the proposed VIB objective for a given observation $(\mathbf{x}^m, \mathbf{y}^m)$. To increase the flexibility, we associate two different coefficients from $\mathbb{R}_+$ with the two KL-terms. In practice we treat them as hyper-parameters.

Detailed Model Specification: In this section, we describe in detail how to learn discrete $\mathbf{A}$ and $\mathbf{R}$ along with the model parameters during training. To learn the word mask $\mathbf{R}$, we use amortized variational inference(Rezende and Mohamed 2015). We use a single-layer feedforward neural network as the inference network $q_\phi(R_{x_t}|x_t)$, associated parameters ϕ are optimized with the model parameters during training. For the interaction mask (graph) $\mathbf{A}$, we use a trainable parameter matrix $\gamma \in \mathbb{R}^{|V|\times|V|}$, that is also optimized along with the model parameters during training. Further, we use the mean field approximation (Blei, Kucukelbir, and McAuliffe 2017) for both the word mask and the variational interaction mask, that is, $q(\mathbf{R}|\mathbf{x}) = \prod_{i=1}^{L} q(R_{x_t}|\mathbf{x}_t)$ and $q(\mathbf{A}_\mathbf{x}|\mathbf{x}) = \prod_{i=1}^{L}\prod_{j=1}^{L} q(A_{x_i,x_j}|\mathbf{x}_i, \mathbf{x}_j)$.

Equation 17 requires prespecified prior distributions p_{r0} and p_{a0}. We use the Bernoulli distribution prior (a non-informative prior) for each word-pair interaction $q_\phi[\mathbf{A}_{x_i,x_j}|\mathbf{x}_i, \mathbf{x}_j]$. $p_{a0}(\mathbf{A}_x) = \prod_{i=1}^{L}\prod_{j=1}^{L} p_{a0}(\mathbf{A}_{\mathbf{x}_i,\mathbf{x}_j})$ and $p_{a0}(\mathbf{A}_{x_i,x_j}) = Bernoulli(0.5)$. This leads to:

$$KL(q(\mathbf{A}_\mathbf{x}|\mathbf{x}^m)||p_{a0}(\mathbf{A})) = -H_\mathbf{q}(\mathbf{A}_\mathbf{x}|\mathbf{x}^m) \tag{18}$$

Here, H_q denotes the entropy of the term $\mathbf{A}_\mathbf{x}|\mathbf{x}^m$ under the q distribution. Similarly, for the word mask, $p_{r0}(\mathbf{R}) = \prod_{i=1}^{L} p_{r0}(\mathbf{R}_{\mathbf{x}_i})$, and $p_{r0}(\mathbf{R}_{\mathbf{x}_i}) = Bernoulli(0.5)$. Therefore,

$$KL(q(\mathbf{R}_\mathbf{x}|\mathbf{x}^m)||p_{a0}(\mathbf{R})) = -H_\mathbf{q}(\mathbf{R}_\mathbf{x}|\mathbf{x}^m) \tag{19}$$

Finally, we have the following loss function for $(\mathbf{x}^m, \mathbf{y}^m)$:

$$\begin{aligned} -(\mathbb{E}_\mathbf{x} p(\mathbf{y}|\mathbf{x}^m; \mathbf{A}, \mathbf{R}, \{\mathbf{W}\}) + \beta_i H_q(\mathbf{R}_\mathbf{x}|\mathbf{x}^m) + \\ \beta_g H_\mathbf{q}(\mathbf{A}_\mathbf{x}|\mathbf{x}^m)) + \beta_{sparse}||\mathbf{A}_\mathbf{x}||_1 \end{aligned} \tag{20}$$

During training, both $\mathbf{A}$ and $\mathbf{R}$ are discrete samples drawn from Bernoulli distributions in Eq. (2) and Eq. (11). We, therefore, use the Gumbel-Softmax(Jang, Gu, and Poole 2016) trick to differentiate through the sampling step and propagate the gradients to their respective parameters γ and ϕ.

B Detailed Derivation

We follow the markov factorization : $p(\mathbf{X}, \mathbf{Y}, \mathbf{Z}) = p(\mathbf{Z}|\mathbf{X}, \mathbf{Y})P(\mathbf{X}, \mathbf{Y}) = P(\mathbf{Z}|\mathbf{X}, \mathbf{Y})P(\mathbf{Y}|\mathbf{X})P(\mathbf{X}) = P(\mathbf{Z}|\mathbf{X})P(\mathbf{Y}|\mathbf{X})P(\mathbf{X})$, motivated from the Markov assumption : $\mathbf{Y} \leftrightarrow \mathbf{X} \leftrightarrow \mathbf{Z}$, i.e. $\mathbf{Y}$ and $\mathbf{Z}$ are independent of each other given $\mathbf{X}$. We assume the data are generated using the above assumption where $\mathbf{Z}$ is not observed, i.e. a latent variable. Our derivation is based on (Chen et al. 2018; Schulz et al. 2020; Alemi et al. 2016), where we start from an approximation $q(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$ instead of the true distribution $p(\mathbf{X}, \mathbf{Y}, \mathbf{Z})$.

The lower bound for $I(\mathbf{Z}; \mathbf{Y})$.

$$\begin{aligned} I(\mathbf{Z}, \mathbf{Y}) &= \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log \frac{q(\mathbf{y}, \mathbf{z})}{q(\mathbf{y})q(\mathbf{z})} \\ &= \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log \frac{q(\mathbf{y}|\mathbf{z})}{q(\mathbf{y})} \\ &= \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log q(\mathbf{y}|\mathbf{z}) \\ &\quad + H_q(\mathbf{Y}), \end{aligned} \tag{21}$$

where $H_q(\cdot)$ represents entropy, and can be ignored for the purpose of training the model.

$$\begin{aligned} &\sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log q(\mathbf{y}|\mathbf{z}) \\ &= \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log \frac{q(\mathbf{y}|\mathbf{z})p(\mathbf{y}|\mathbf{z})}{p(\mathbf{y}|\mathbf{z})} \\ &= \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log p(\mathbf{y}|\mathbf{z}) + KL[q(\mathbf{y}|\mathbf{z})||p(\mathbf{y}|\mathbf{z})] \\ &\geq \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log p(\mathbf{y}|\mathbf{z}), \end{aligned} \tag{22}$$

where $KL[\cdot||\cdot]$ denotes Kullback-Leibler divergence. This gives us the following lower bound:

$$\begin{aligned} I(\mathbf{Z}, \mathbf{Y}) &\geq \sum_{\mathbf{y},\mathbf{z}} q(\mathbf{y}, \mathbf{z}) \log p(\mathbf{y}|\mathbf{z}) + H_q(\mathbf{y}) \\ &= \sum_{\mathbf{y},\mathbf{z},\mathbf{x}} q(\mathbf{x}, \mathbf{y}, \mathbf{z}) \log p(\mathbf{y}|\mathbf{z}) + H_q(\mathbf{y}) \\ &= \sum_{\mathbf{y},\mathbf{z},\mathbf{x}} q(\mathbf{x}, \mathbf{y})q(\mathbf{z}|\mathbf{x}) \log p(\mathbf{y}|\mathbf{z}) + H_q(\mathbf{y}), \end{aligned} \tag{23}$$

where the last step uses $q(\mathbf{x}, \mathbf{y}, \mathbf{z}) = q(\mathbf{x})q(\mathbf{y}|\mathbf{x})q(\mathbf{z}|\mathbf{x})$, which is a factorization based on the conditional dependency: $\mathbf{y} \leftrightarrow \mathbf{x} \leftrightarrow \mathbf{z}$: $\mathbf{y}$ and $\mathbf{z}$ are independent given $\mathbf{x}$.

Given a sample, $(\mathbf{x}^{(m)}, \mathbf{y}^{(m)})$, we can assume the empirical distribution $q(\mathbf{x}^{(m)}, \mathbf{y}^{(m)})$ simply defined as a multiplication of two Delta functions

$$q(\mathbf{x} = \mathbf{x}^{(m)}, \mathbf{y} = \mathbf{y}^{(m)}) = \delta_{\mathbf{x}^{(m)}}(\mathbf{x}) \cdot \delta_{\mathbf{y}^{(m)}}(\mathbf{y}). \tag{24}$$

So we simplifying further of the first term:

$$\begin{aligned} I(\mathbf{z}; \mathbf{y}^{(m)}) &\geq \sum_{\mathbf{z}} q(\mathbf{z}|\mathbf{x}^{(m)}) \log p(\mathbf{y}^{(m)}|\mathbf{z}) \\ &= \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(m)})} \log(p(\mathbf{y}^{(m)}|\mathbf{z})) \end{aligned} \tag{25}$$

Since $\mathbf{Z} = \text{diag}\,(\mathbf{R}_\mathbf{x})\,\mathbf{E}'$, and $\mathbf{E}'$ is a deterministic function of $\mathbf{A}$, we have:

$$I(\mathbf{z}; \mathbf{y}^{(m)}) \geq \mathbb{E}_{q(\mathbf{z}|\mathbf{x}^{(m)})} log(p(\mathbf{y}^{(m)}|\mathbf{R}, \mathbf{A}, \mathbf{x}^{(m)})) \tag{26}$$

The upper bound for $I(\mathbf{Z};\mathbf{X})$.

$$\begin{aligned}I(\mathbf{Z},\mathbf{X}) = & \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log \frac{q(\mathbf{x},\mathbf{z})}{q(\mathbf{x})q(\mathbf{z})} \\ = & \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log \frac{q(\mathbf{z}|\mathbf{x})}{q(\mathbf{z})} \\ = & \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log q(\mathbf{z}|\mathbf{x}) \\ & - \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log q(\mathbf{z})\end{aligned}$$

By replacing $q(\mathbf{z})$ with a prior distribution of $\mathbf{z}$, $p_0(\mathbf{z})$, we have

$$\sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log q(\mathbf{z}) \geq \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log p_0(\mathbf{z}). \tag{27}$$

Then we can obtain an upper bound of the mutual information

$$\begin{aligned}I(\mathbf{Z};\mathbf{X}) \leq & \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log q(\mathbf{z}|\mathbf{x}) \\ & - \sum_{\mathbf{x},\mathbf{z}} q(\mathbf{x},\mathbf{z}) \log p_0(\mathbf{z}) \\ = & \sum_{\mathbf{x}} q(\mathbf{x}) KL[q(\mathbf{z}|\mathbf{x})||p_0(\mathbf{z})] \\ = & \mathbb{E}_{q(x)} KL[q(\mathbf{z}|\mathbf{x})||p_0(\mathbf{z})]. \end{aligned} \tag{28}$$

For a given sample $(\mathbf{x}^m, \mathbf{y}^m)$,

$$I(\mathbf{z};\mathbf{x}^{(m)}) = KL[q(\mathbf{z}|\mathbf{x}^{(m)})||p_0(\mathbf{z})] \tag{29}$$

Given $\mathbf{X}$, we assume $\mathbf{R}$ and $\mathbf{A}$ are independent. We also assume a factorable prior $p_0(\mathbf{z}) = p_{r0}(\mathbf{R})p_{a0}(\mathbf{A})$. This gives us:

$$\begin{aligned}I(\mathbf{z},\mathbf{x}^{(m)}) \leq & KL(q(\mathbf{R}|\mathbf{x}^{(m)})||p_{r0}(\mathbf{R})) \\ & + KL(q(\mathbf{A}|\mathbf{x}^{(m)})||p_{a0}(\mathbf{A}))\end{aligned} \tag{30}$$

Additional Experimentation: Ablation Analysis For Interpretation Scores

Metric	Models	TREC	SST-2
AOPC	BASE	67.63	42.25
	VMASK	63.14	39.77
	WIGRAPH-A-R	61.52	36.22
	WIGRAPH-A	**68.74**	**44.33**

Table 6: Ablations analysis for TREC and SST-2 datasets on distilbert model regarding AOPC from LIME post-hoc explanation model for ablations WIGRAPH-A and WIGRAPH-A-R.

References

Alemi, A. A.; Fischer, I.; Dillon, J. V.; and Murphy, K. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Alvarez-Melis, D.; and Jaakkola, T. S. 2018a. Towards robust interpretability with self-explaining neural networks. In *NeurIPS*.

Alvarez-Melis, D.; and Jaakkola, T. S. 2018b. Towards Robust Interpretability with Self-Explaining Neural Networks. arXiv:1806.07538.

Bang, S.; Xie, P.; Lee, H.; Wu, W.; and Xing, E. 2019. Explaining a black-box using deep variational information bottleneck approach. *arXiv preprint arXiv:1902.06918*.

Blei, D. M.; Kucukelbir, A.; and McAuliffe, J. D. 2017. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518): 859–877.

Camburu, O.-M.; Rocktäschel, T.; Lukasiewicz, T.; and Blunsom, P. 2018. e-snli: Natural language inference with natural language explanations. In *Advances in Neural Information Processing Systems*, 9539–9549.

Chen, H.; and Ji, Y. 2019. Improving the Explainability of Neural Sentiment Classifiers via Data Augmentation. *arXiv preprint arXiv:1909.04225*.

Chen, H.; and Ji, Y. 2020. Learning Variational Word Masks to Improve the Interpretability of Neural Text Classifiers. *arXiv preprint arXiv:2010.00667*.

Chen, H.; Zheng, G.; and Ji, Y. 2020. Generating hierarchical explanations on text classification via feature interaction detection. *arXiv preprint arXiv:2004.02015*.

Chen, J.; Song, L.; Wainwright, M. J.; and Jordan, M. I. 2018. Learning to explain: An information-theoretic perspective on model interpretation. *arXiv preprint arXiv:1802.07814*.

Du, M.; Liu, N.; Yang, F.; Ji, S.; and Hu, X. 2019. On attribution of recurrent neural network predictions via additive decomposition. In *The World Wide Web Conference*, 383–393.

Erion, G.; Janizek, J. D.; Sturmfels, P.; Lundberg, S.; and Lee, S.-I. 2019. Learning explainable models using attribution priors. *arXiv preprint arXiv:1906.10670*.

Halford, G. S.; Wilson, W. H.; and Phillips, S. 2010. Relational knowledge: the foundation of higher cognition. *Trends in cognitive sciences*, 14(11): 497–505.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Koh, P. W.; Nguyen, T.; Tang, Y. S.; Mussmann, S.; Pierson, E.; Kim, B.; and Liang, P. 2020. Concept bottleneck models. In *International Conference on Machine Learning*, 5338–5348. PMLR.

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

Li, X.; and Roth, D. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Lundberg, S. M.; Erion, G. G.; and Lee, S.-I. 2018. Consistent individualized feature attribution for tree ensembles. *arXiv preprint arXiv:1802.03888*.

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

Maas, A.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, 142–150.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Miwa, M.; and Bansal, M. 2016. End-to-end relation extraction using lstms on sequences and tree structures. *arXiv preprint arXiv:1601.00770*.

Molnar, C.; Casalicchio, G.; and Bischl, B. 2019. Quantifying model complexity via functional decomposition for better post-hoc interpretability. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 193–204. Springer.

Murdoch, W. J.; Liu, P. J.; and Yu, B. 2018. Beyond word importance: Contextual decomposition to extract interactions from LSTMs. *arXiv preprint arXiv:1801.05453*.

Nguyen, D. 2018. 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: Human Language Technologies, Volume 1 (Long Papers)*, 1069–1078.

Pang, B.; and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*.

Rezende, D. J.; and Mohamed, S. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. 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*, 1135–1144. ACM.

Rieger, L.; Singh, C.; Murdoch, W.; and Yu, B. 2020. Interpretations are useful: penalizing explanations to align neural networks with prior knowledge. In *International Conference on Machine Learning*, 8116–8126. PMLR.

Ross, A. S.; and Doshi-Velez, F. 2017. Improving the Adversarial Robustness and Interpretability of Deep Neural Networks by Regularizing their Input Gradients. arXiv:1711.09404.

Ross, A. S.; Hughes, M. C.; and Doshi-Velez, F. 2017. Right for the right reasons: Training differentiable models by constraining their explanations. *arXiv preprint arXiv:1703.03717*.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5): 206–215.

Samek, W.; Binder, A.; Montavon, G.; Lapuschkin, S.; and Müller, K.-R. 2016. Evaluating the visualization of what a deep neural network has learned. *IEEE transactions on neural networks and learning systems*, 28(11): 2660–2673.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1): 61–80.

Schulz, K.; Sixt, L.; Tombari, F.; and Landgraf, T. 2020. Restricting the flow: Information bottlenecks for attribution. *arXiv preprint arXiv:2001.00396*.

Singh, C.; Murdoch, W. J.; and Yu, B. 2018. Hierarchical interpretations for neural network predictions. *arXiv preprint arXiv:1806.05337*.

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

Tishby, N.; Pereira, F. C.; and Bialek, W. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Tishby, N.; and Zaslavsky, N. 2015. Deep learning and the information bottleneck principle. In *2015 IEEE Information Theory Workshop (ITW)*, 1–5. IEEE.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Wolf, T.; Chaumond, J.; Debut, L.; Sanh, V.; Delangue, C.; Moi, A.; Cistac, P.; Funtowicz, M.; Davison, J.; Shleifer, S.; et al. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 38–45.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; and Macherey, K. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xie, N.; Ras, G.; van Gerven, M.; and Doran, D. 2020. Explainable deep learning: A field guide for the uninitiated. *arXiv preprint arXiv:2004.14545*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, 649–657.