SAMPLE-SPECIFIC ATTENTION MASKS FOR MODEL TRANSPARENCY AND ADVERSARIAL DETECTION

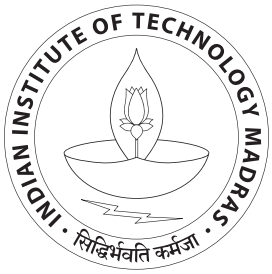
A Project Report

submitted by

EMIL BIJU

in partial fulfilment of the requirements
for the award of the degree of

BACHELOR OF TECHNOLOGY



DEPARTMENT OF ELECTRICAL ENGINEERING
INDIAN INSTITUTE OF TECHNOLOGY, MADRAS.

June 2021

THESIS CERTIFICATE

This is to certify that the thesis entitled **Sample-specific Attention Masks for Model Transparency and Adversarial Detection**, submitted by **Emil Biju**, to the Indian Institute of Technology, Madras, for the award of the degree of **Bachelor of Technology**, is a bona fide record of the research work carried out by him under my supervision. The contents of this thesis, in full or in parts, have not been submitted to any other Institute or University for the award of any degree or diploma.

Prof. Mitesh M. Khapra
Research Guide
Associate Professor
Dept. of CSE
IIT Madras, 600036

Prof. Rajagopalan A.N.
Department Co-Guide
Professor
Dept. of Electrical Engineering
IIT-Madras, 600036

Place: Chennai

Date: 23 June, 2021

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude towards Prof. Mitesh Khapra and Prof. Pratyush Kumar who have played a phenomenal role in guiding me through the course of this work. I thank them for the opportunity to work on this exciting project and for the regular meetings where we they shared ideas that effectively guided me in this endeavour and transformed my outlook towards research as a whole.

I would like to acknowledge the support of Anirudh Sriram who has contributed towards this project with his ideas and help in implementation. I also express my sincere gratitude towards IIT Madras for an amazing four years that I have spent here availing innumerous opportunities for holistic growth and development. I am also grateful to the Samsung-IITM Pravartak Foundation for a generous fellowship in support of my project.

Finally, I would like to thank my parents and brother for their unending support and encouragement as I pursued an exciting academic journey.

ABSTRACT

KEYWORDS: network interpretability, adversarial detection, natural language processing

Multi-head self-attention is a vital component of the Transformer, a neural network architecture that has achieved state-of-the-art performance on various sequence learning tasks. In this work, we propose a method to induce accurate sample-specific attention masks for Transformer-based networks like BERT. We show that such masks are discriminative of the samples' output class for various Natural Language Understanding (NLU) tasks. We leverage this property to enhance model transparency and classify between authentic and adversarial samples. Further, we find two other properties which contribute to this classification. First, selectively mutating the masks leads to contrasting model outputs based on sample authenticity. Second, the consistency of auxiliary layer-wise outputs varies based on sample authenticity. By combining these observations, we propose a sample-efficient and generalized scheme for adversarial detection. We perform experiments on 8 NLU datasets with 11 different adversarial attack types and report state-of-the-art accuracy ranging from 80 to 90%. In summary, our work introduces an entirely new and promising approach to interpret and analyze large self-attention based networks for NLP.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

NLP	Natural Language Processing
NLU	Natural Language Understanding
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
LSTM	Long Short-term Memory
ReLU	Rectified Linear Unit
FCNN	Fully Connected Neural Network
SVM	Support Vector Machine

NOTATION

N	Number of input data samples for training
n	Number of layers in a Transformer-based model
m	Number of heads per layer in a Transformer-based model
θ	Standard network parameters of a Transformer-based model
θ^*	Fine-tuned network parameters of a Transformer-based model
$\mathcal{L}_{CE}$	Cross Entropy Loss
$\mathcal{L}_A^B$	Training loss with representative input A and trainable parameters B
$g(x)$	Pruning mask vector defined for sample input x
$g^b(x)$	Boolean-valued pruning mask vector defined by thresholding $g(x)$
$g^c(x)$	Pruning mask vector obtained by selectively mutating $g(x)$
$\mathcal{S}(g(x))$	Subnetwork induced by the pruning mask vector $g(x)$
$\mathcal{F}_i(x)$	The i^{th} feature input to the adversarial detection classifier for sample x
$\mathcal{F}(x)$	The complete feature input to the adversarial detection classifier for sample x

CHAPTER 1

INTRODUCTION

Key advancements over the last few years in sequence learning and natural language processing have been centered around the Transformer [Vaswani *et al.*, 2017] and other architectures like BERT [Yuan *et al.*, 2019] that have been derived from it. These models make use of the self-attention framework that relates different parts of a sequence to compute a representation of the sequence. Each unit that performs self-attention in such models is called as a self-attention head.

Various approaches have been proposed to classify individual self-attention heads based on the roles they play in a network and to determine the relative importance among them. Recent works have also attempted to improve the computational efficiency of Transformer-based models by pruning less important heads that are not critical to the final prediction. It has been shown that several heads from a Transformer network can be pruned without substantially affecting the model performance. A natural question which then emerges is - Can the patterns of pruning and the positions of the pruned heads explain how such models process a given input?

A key observation here is that while prior approaches select a fixed subset of heads to be pruned irrespective of the input sample, it is possible that different subsets of heads play vital roles in processing different inputs. This motivates the idea of sample-specific pruning of self-attention heads that we adopt in our work. Also, unlike earlier approaches that prune heads for improving efficiency, the objective of our work is to perform sample-specific pruning for improving

the interpretability of Transformer-based models which subsequently helps in adversarial detection.

A frequently encountered challenge in improving the reliability of machine learning models is of robustifying them against adversarial samples [Yuan *et al.*, 2019]. An adversarial sample is an instance that is created by making small perturbations to an authentic sample such that the model generates a wrong prediction for the perturbed sample. Such instances pose a severe threat to the dependability of machine learning models, particularly if they are employed in safety-critical or large-scale applications.

As Transformer-based models proliferate deeper into real-world applications, the goal of improving the robustness of such models to adversarial samples gains prominence. While recent works in this area prescribe to adversarial training to train robust networks, such approaches can only provide point-wise guarantees [Pang *et al.*, 2018] that are sensitive to specific attack types. Besides, generating a large number adversarial samples is challenging due to high computational costs and difficulties in ensuring that sentence meanings are not significantly changed [Yu *et al.*, 2018]. As a consequence, there is a clear necessity for methods that easily generalize to diverse attack types with a small set of illustrative adversarial samples.

To circumvent the issues of sample efficiency and computational costs, adversarial detection has been advocated as an alternative strategy. In this approach, instead of training models to make correct predictions on adversarial inputs, a separate framework is developed that detects such inputs to challenge the model's prediction and allow further examination. Recent works [Liu *et al.*, 2018] have explored how improving the interpretability of neural networks can help in adversarial detection

since an enhanced understanding of the underlying mechanisms can expose the reasons for their weaknesses.

In our work, we propose a method to induce sample-specific subnetworks by pruning self-attention heads from Transformer-based models. We then derive features from these subnetworks that expose variations in model behavior between authentic and adversarial samples. The main contributions of this work are summarized as follows:

- We design an efficient algorithm that identifies a sparse subset of self-attention heads that encode the paths of important information flow through the network and define a pruning mask that obscures the remaining heads.
- We propose a novel adversarial detection framework for Transformers that uses a set of features obtained from sample-specific pruning. We show that this approach is sample-efficient and easily generalizable to diverse attack types.
- We devise a series of experiments that evaluate how the pruning mask is correlated to its corresponding input sample and the roles that the retained heads play in generating the prediction.. Further, we explore how selectively mutating the pruning mask can be informative of sample authenticity.

This work has critical importance in various Electrical engineering applications, particularly in conjunction with image and speech processing techniques that increasingly using deep neural networks and require more interpretability in such models. Besides, adversarial robustness is a key factor that determines the reliability of such models in various applications and our work proposes a method to effectively address the same. Pruning of self-attention heads has been studied for improving the efficiency of Transformer-based networks which is crucial in the development of hardware-efficient Transformer models [Wang *et al.*, 2020*a*] and influences the design of hardware chips for running deep learning models. This work highlights the advantages of developing hardware that reduces overhead in pruning self-attention heads in such models.

CHAPTER 2

RELATED WORK

In this chapter, we briefly review related literature in this area by discussing approaches that have earlier been proposed for interpreting Transformer networks, detecting adversarial samples and extracting subnetworks from neural networks. Further, we provide a primer on the Transformer architecture and the computations that it performs during inference.

2.1 Extracting subnetworks from neural networks

An important body of work is focussed on extracting subnetworks from neural networks. Works by Voita *et al.* [2019], Michel *et al.* [2019] and Budhraja *et al.* [2020] observe that several heads in Transformer networks can be pruned without substantially compromising on model performance. However, the objective in these works is to achieve higher compute efficiency by reducing the number of heads that take part during inference. In all of these studies, a common masking scheme is applied over all input samples. In contrast, our work involves obtaining a pruning mask for each sample separately. Lakshminarayanan and Vikram Singh [2020] showed how ReLU activations act as gates that define a subnetwork within a deep neural network.

The work by Liu *et al.* [2018] involves optimizing a set of control gates to identify a subnetwork within a network with specific application in CNNs for

vision tasks. They demonstrate how such subnetworks improve explainability of results generated by CNNs using saliency maps. Besides, they emphasize that the subnetwork representations for adversarial image inputs diverge from the representations generated for the original image as layers progress and converge towards representations of target class images. However, the full potential for adversarial detection has not been explored in this work. Besides, this method aims to identify the critical subnetwork which is the sparsest possible network that predicts the same class output, making it computationally expensive. This is in contrast to our work where we try to maximise the probability of the output class and later enforce sparsity by removing less-important heads. Our approach has the advantage of requiring fewer epochs to arrive at a subnetwork.

2.2 Adversarial robustness and detection

Several recent works aimed at improving robustness of deep learning models to adversarial samples in NLP tasks advocate adversarial training [Chen *et al.*, 2020; Cheng *et al.*, 2020; Liu *et al.*, 2020]. In this method, the model is trained using a dataset that contains both authentic and adversarial samples with a joint objective that maximizes the probability that the model predicts the correct output for both kinds of samples. In this way, the robustness of the model to adversarial samples is improved without compromising on the performance on authentic samples. However, such methods involve the expensive overhead of generating a large number of adversarial samples to train the model to generalize to various kinds of attacks. Given the difficulties in training robust models, other works have focussed on detecting adversarial samples. These include works by Pang *et al.* [2018] and Li *et al.* [2021]. The work by Liu *et al.* [2018] detects adversarial samples based on the

downstream prediction confidence but does not leverage the internal mechanisms of the model.

2.3 Interpreting Transformer networks

The work by Kovaleva *et al.* [2019] attempts to interpret BERT models by discovering consistent attention patterns and characterizing self-attention heads based on specific roles that they play in processing inputs. Jawahar *et al.* [2019] and van Aken *et al.* [2019] interpret the functioning of BERT by dissecting the roles of various layers. They show how lower layers capture surface-level information while higher layers capture more complex and long-distance dependencies. [Koh and Liang, 2017] demonstrates how influence functions can be used to interpret machine learning models by tracing the model's prediction through its processing stages back to the training data.

2.4 Background on the Transformer

The Transformer is a deep neural network architecture designed to handle sequential data, such as text sentences. However, unlike RNNs, Transformers do not require that the sequential data be processed in order. A transformer can simultaneously attend to all parts of the input sequence, thus providing a high degree of parallelization and faster processing.

The functioning of the encoder portion of the Transformer network is explained below, considering a sentence (sequence of words) as the input. Consider a sentence with n words, $S = \{w_1, w_2, ..., w_T\}$. Let $x_t, t \in [1, T]$ be the input to an encoder layer

of the Transformer. We define 3 vectors: q_t, k_t, v_t which are called as query, key and value vectors respectively. Given parameters W^Q, W^K, W^V, these vectors are computed as follows:

$$q_t = W^Q x_t$$

$$k_t = W^K x_t$$

$$v_t = W^V x_t$$

We define, $Q = \{q_t\}, K = \{k_t\}, V = \{v_t\}$ where $t \in [1, 2, .., T]$. Now, we introduce the method of scaled dot-product attention ($SDPA$) which is the method used to compute self-attention between these vectors. Given d-dimensional vectors q_t, k_m, v_t, an attention score is obtained as:

$$\alpha_{tm} = \frac{1}{\sqrt{d}} < q_t, k_m >$$

where $< \cdot >$ represents dot product

$$a_{tm} = \text{softmax}(\alpha_{tm})$$

$$c_t = \sum_{m=1}^{T} a_{tm} v_t$$

The above process can be written in short as:

$$c_t = SDPA(q_t, K, V), t = 1, 2, .., T$$

$$C = \{c_t\}$$

In a transformer network, multi-head attention is used. This means that if there are h heads, separate query, key and value vectors are defined for $i = 1, 2, ..., h$. Thus, the overall context vector is defined as:

$$c_t = concat(c_{1t}, c_{2t}, ..., c_{ht})$$

Such a context vector is generated for all words in the sentence, i.e., for $t = 1, 2, ..., T$. This gives an encoded representation for each word in the input sentence. These representations are then concatenated and passed through a feed-forward fully-connected sub-layer to obtain the output from the respective Transformer layer.

CHAPTER 3

METHODOLOGY

3.1 Extracting Sample-specific Subnetworks

In this section, our goal is to find a sparse subset of attention heads for each input sample, such that the output class of the model formed by pruning the remaining heads remains unaffected.

3.1.1 Setup for pruning attention heads

Each Transformer encoder layer consists of a multi-head self-attention sub-layer and a position-wise feed-forward sub-layer. Let $W^Q_{ji}, W^K_{ji}, W^V_{ji}$ be the parameters for the i^{th} self-attention head in the j^{th} layer of the Transformer network. Given a data sample x consisting of T tokens represented as d_v-dimensional vectors, let $X_j \in \mathbb{R}^{T \times d_v}$ be the corresponding input matrix at the j^{th} layer. We now follow the description of the Transformer in Section 2.4, but present the computations in matrix notation which enables parallelized processing on GPUs.

We define $Q_{ji} = X_j W^Q_{ji}, K_{ji} = X_j W^K_{ji}, V_{ji} = X_j W^V_{ji}$ as the query, key and value matrices corresponding to the head respectively. Each self-attention head performs scaled dot-product attention to generate the head output.

$$\text{Head}_{ji}(X_j) = \text{softmax}\left(\frac{Q_{ji}K^T_{ji}}{\sqrt{d_k}}\right)V_{ji} \tag{3.1}$$

The output of all the heads in a layer are concatenated and passed through the feed-forward sub-layer.

$$\text{Layer}_j(X_j) = \text{concat}_i[\text{Head}_{ji}(X_j)]W_j^O \tag{3.2}$$

where d_k is the dimensionality of each key vector.

To enable pruning of attention heads, we modify Equation 3.2 to weigh the output of each head by a scalar gating value $g_{ji} \in [0, 1]$. The j^{th} layer of the modified network is given by

$$\text{Layer}_j^m(X_j) = \text{concat}_i[g_{ji} \cdot \text{Head}_{ji}(X_j)]W_j^O \tag{3.3}$$

Thus, for a data sample x and a Transformer encoder network like BERT with n layers and m heads per layer, we define the pruning mask vector, $g(x) = \{g_{ji}\} \in [0, 1]^{nm}$ as the vector of gating values. During inference, g_{ji} is replaced by $g_{ji}^b \in \{0, 1\}$ to characterize the discrete exclusion or inclusion of a head. The corresponding Boolean pruning mask vector is given by $g^b(x) = \{g_{ji}^b\} \in \{0, 1\}^{nm}$. We call the subset of network heads that are assigned a Boolean gating value of 1 as active heads and note that the active heads together define a Transformer subnetwork, $\mathcal{S}(g^b(x))$, which processes x using only the active heads.

3.1.2 Generating the pruning mask

Initially, a pre-trained Transformer model like BERT is fine-tuned on a specific classification task using the standard training procedure to minimize the cross entropy loss. Therefore, if θ is the set of network parameters, this training step

aims to minimize

$$\mathcal{L}^{\theta}_{\{x_1:x_N\}} = \frac{1}{N}\sum_{k=1}^{N}\mathcal{L}_{CE}(f(x_k,\theta);y_k) \tag{3.4}$$

where $f(\cdot)$ is the function computed by the model with parameters θ, $\mathcal{L}_{CE}$ is the standard cross-entropy loss function, N is the number of input data samples and y_k is the expected model output for input x_k. Let θ^* be the set of optimal network parameters obtained after training.

In the next step, we wish to obtain a pruning mask vector, $g(x)$ for each sample x. This is achieved by treating the gating values as trainable model parameters and optimizing them over the objective function separately for each x. In effect, this step entails an inference using optimization to obtain gating values for each sample separately. All the gating values are initialized to the intermediate value of 0.5. Each gating value, g_{ji} is defined as $g_{ji} = f_{HC}(p_{ji})$, where f_{HC} is a version of the hard concrete distribution [Louizos *et al.*, 2017] which acts as an activation function upon p_{ji} as shown below.

$$g_{ji} = \frac{1}{1+e^{\alpha\cdot(log(1-p_{ji})-log(p_{ji}))}} \tag{3.5}$$

The hard concrete function is a smooth differentiable function that is well-defined between 0 and 1. The plots of the function for different values of α are shown in Figure 3.1. We observe that as α increases, the function becomes more steep. This helps to simulate the intended restriction of the gating values taking on Boolean values (0-pruned head, 1-active head). A comparison is also made with a version of the sigmoid function $\frac{1}{1+e^{-(x-0.5)}}$ which is much less steep in the same range. Among possible values, α=6 gave the best results for our work.

All the network parameters except for the gating values are now frozen and the

gating values in the model are optimized using the same objective as before, but for each data sample separately.

$$\mathcal{L}_x^g = \mathcal{L}_{CE}(f(x, \theta, g(x)); y|\theta = \theta^*) \tag{3.6}$$

Unlike approaches by Voita *et al.* [2019] and Wang *et al.* [2020*b*], we do not include a regularization term in the optimization objective. Instead, we use the positive deviation of the gating values after η epochs of optimization as a measure of the importance of their respective attention heads. We perform a thresholding over these values to enforce sparsity. Thus, we are effectively selecting heads whose gating values ascend faster towards 1 as the active heads. This was observed to reduce the number of iterations required to obtain an optimal pruning mask. Specifically, each Boolean gating value, g_{ji}^b is derived from g_{ji} as follows.

$$g_{ji}^b(x) = \begin{cases} 1, & \text{if } g_{ji}(x) \geq \beta \cdot \max(g(x)) \\ 0, & \text{otherwise} \end{cases}$$

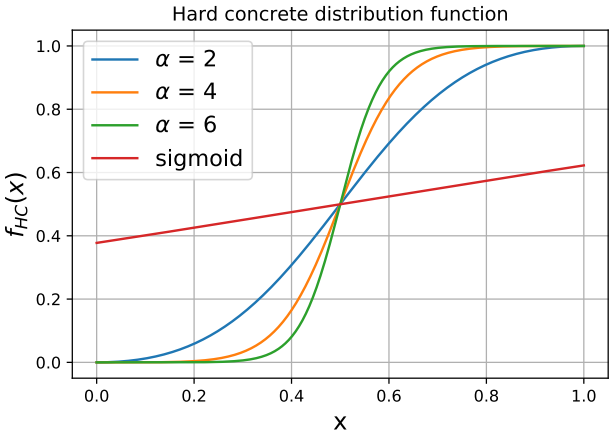


Figure 3.1: Plot of the hard-concrete distribution for different values of α. The function gets steeper with increasing α. A comparison is also made with a sigmoid function which is much less steep in this range.

where, $\beta(< 1)$ is a thresholding parameter and $\max(g(x))$ represents the largest among the nm gating values in $g(x)$.

Figure 3.2 shows the progression of gating values with epochs for a single input sample. However, two exceptional cases may arise which we address as follows. If the Boolean gating values of all heads in a layer are found to be 0 after the above thresholding, the largest gating value in that layer is set to 1 to ensure information flow through the network. We set $\beta = 0.8$ initially, but if the subnetwork prediction does not match the expected output, we iteratively decrease β by 0.2 and re-apply the thresholding until the subnetwork generates the desired output. The green curves in Figure 3.2 that are below the blue line correspond to these two cases.

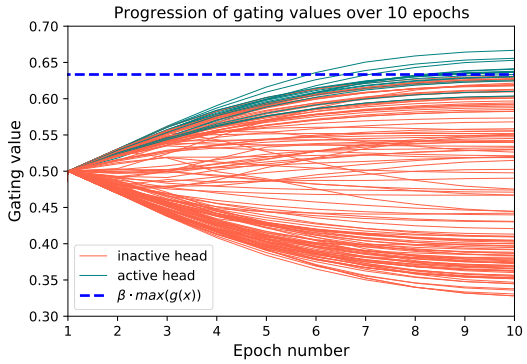


Figure 3.2: Variation in the gating values of 144 transformer heads from BERT-Base when trained over 10 epochs on a single random sample. The gating values of several heads reduce from 0.5 though the original fully trained model maintained all heads as active, i.e., with effective gating value of 1.

Figure 3.3 shows the variation of the fraction of authentic samples from 6 different datasets considered in our work that retain a given number of active heads after pruning. The value peaks around 20-40 heads out of 144 and then quickly descends which shows that nearly 70% of the heads are not critical for prediction for most samples across datasets.

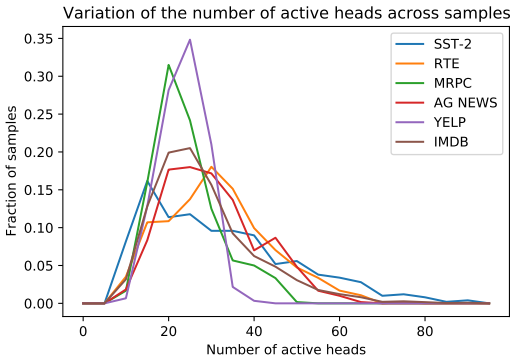


Figure 3.3: Fraction of samples with a given number of active heads from BERT-Base across 6 tasks on which the model is fine-tuned. Around 20-40 heads out of 144 suffice for a large number of samples to generate the correct prediction.

3.2 Model for Adversarial Sample Detection

In this section, we discuss three stages of feature extraction and finally present a classifier that leverages these features for adversarial detection. The three stages are demonstrated for a sample input in Figure 3.4. The characteristics of these features that help to discern adversarial samples are explicated through our experiments in Section 4.2.1. In further sections of the paper, we use the term *target class* to refer to the class predicted by the complete (unpruned) fine-tuned network for an input sample. For authentic samples, this translates to the true class while for adversarial samples, this refers to the adversarial class that the model is fooled into predicting.

3.2.1 Pruning mask vector as a feature

The pruning mask vector that is trained by the process detailed in the previous section forms the first set of input features to our model. We observed distinguishing characteristics in the masking patterns generated for authentic and adversarial samples, which result from the model's attempts to block or pass selective chunks of information to maximise the target class probability. Since the real-valued pruning

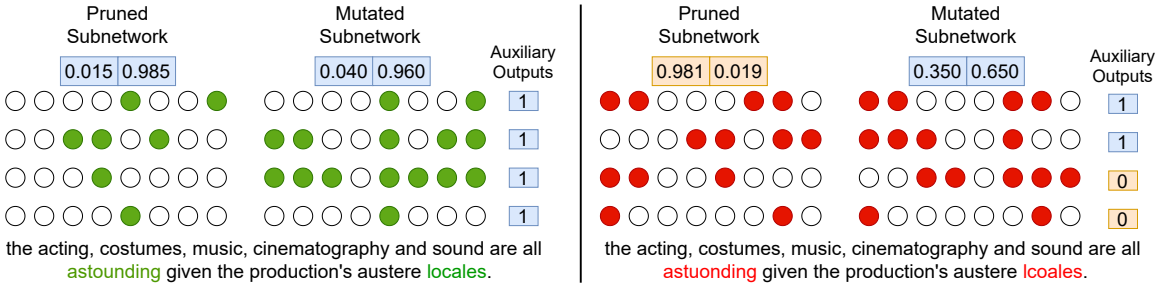


Figure 3.4: The three stages of feature extraction. Active heads in the pruned and mutated sample-specific subnetworks from the BERT-Small architecture are colored. The input on the left is an authentic sample from the SST-2 dataset and on the right is a corresponding adversarial sample. The outputs logits for sentiment classification from both subnetworks and the auxiliary layer-wise outputs of the mutated subnetwork are shown. Notice how the coloring of heads in the middle 2 layers are flipped in the mutated subnetwork.

mask vector stores more intrinsic information than the Boolean vector, we define the nm-dimensional vector, $\mathcal{F}_1(x) = g(x)$ as a feature input to our model.

3.2.2 Mutating the middle layers of the subnetwork

We note that adversarial samples are more sensitive to mutations in the pruning mask and subsequent changes in the induced subnetwork than authentic samples. This is because adversarial samples are heavily reliant on the network architecture and the specific parameter combinations that allow them to fool the network [Wang *et al.*, 2019]. Hence, when the processing pipeline is changed, they are more likely to change their output.

Previous studies have shown that the initial layers in BERT are responsible for phrase-level understanding of an input sentence while the last few layers are highly task-specific [Jawahar *et al.*, 2019]. Several heads in the middle layers of BERT capture syntactic relations [Hewitt and Manning, 2019; Goldberg, 2019] and are multi-skilled [Pande *et al.*, 2021], making them crucial for prediction. As we will

show in Section 4.2.7, the critical heads for generating the adversarial prediction for an input sample are mostly those which attend to the perturbed portions of the input sentence and more than 30% of the middle layers are composed of such heads. Hence, we hypothesize that mutating the middle layers will diminish the attention to perturbed portions of the sentence and alter spurious syntactic relations which the adversarial samples heavily rely on to generate the adversarial output.

Based on this reasoning, we flip the Boolean gating values in $g^b(x)$ corresponding to the middle $\lceil \frac{n}{3} \rceil$ layers of the network architecture to obtain $g^c(x) \in \{0,1\}^{nm}$. Thus, the subnetwork $\mathcal{S}(g^c(x))$ induced by $g^c(x)$ de-activates the active attention heads in the middle $\lceil \frac{n}{3} \rceil$ layers of the original subnetwork and activates the remaining heads. We run each input sample through its respective mutated subnetwork and obtain a 4-dimensional feature vector, $\mathcal{F}_2(x)$ consisting of the target class, the predicted class, the confidence of prediction and a Boolean flag asserting whether the predicted and target classes are the same.

3.2.3 Layer-wise auxiliary output predictions

For standard classification tasks, the intermediate layers in BERT encode a hierarchy of linguistic knowledge, with surface-level features in the bottom layers, syntactic features in the middle and semantic features in the top layers Jawahar *et al.* [2019]. Hence, the type and amount of information captured by representations vary across layers. Since authentic samples are more stable under mask mutation, representations at each layer are expected to convey similar information as in the case of the full network. But in the case of adversarial samples, the information conveyed by certain intermediate layer representations are substantially different due to which further layers are misinformed and inconsistent information is propagated. Hence,

there could be intermediate representations that are more representative of classes other than the target (adversarial) class. Such representations when passed through a classifier may generate a non-target class.

Given the complete fine-tuned n-layer network with standard network parameters θ^*, we introduce an auxiliary classification layer with parameter set Ω_l after each layer l, $\forall\ l \in \{1, 2, ..., n-1\}$. We then freeze the standard parameters to θ^* and train the auxiliary output layers to maximize the probability that each predicts the target class from the intermediate representation generated by that layer.

$$\mathcal{L}^{\Omega}_{\{x_1:x_N\}} = \frac{1}{N(n-1)} \sum_{k=1}^{N} \sum_{l=1}^{n-1} \mathcal{L}_{CE}(f'_l(p_k^{l-1}, \theta, g(x_k), \Omega_l); y_k | \theta = \theta^*, g = \{1\}^{nm}) \tag{3.7}$$

where $\Omega = \bigcup_{l=1}^{n-1} \Omega_l$, p_k^{l-1} is the representation generated by the $(l-1)^{th}$ layer and input to the l^{th} layer such that $p_k^0 = x_k$ and $f'_l(\cdot)$ represents the function computed by the auxiliary classifier after the l^{th} layer of the network.

Once the training is complete, heads are pruned from the complete network separately for each input x, based on the mutated pruning vector $g^c(x)$ and layer-wise auxiliary outputs are generated from the subnetwork using the same weights obtained before. We extract an $(n+1)$-dimensional feature vector, $\mathcal{F}_3(x)$ consisting of the $(n-1)$ layer-wise outputs, the number of auxiliary layer outputs that match the target class and the number of times the layer-wise predictions switch/flip when visited in the order of layers. This is in contrast to previous studies where the performance of intermediate layer representations on separate probing tasks was tested Belinkov *et al.* [2018]; Liu *et al.* [2019].

Feature	Constituents
$\mathcal{F}_1$	Pruning mask vector $g(x)$
$\mathcal{F}_2$	Target class, predicted class, confidence of prediction, Boolean flag: target class == predicted class
$\mathcal{F}_3$	layer-wise outputs, # auxiliary outputs matching the target class, # switches in layer-wise predictions

Table 3.1: Summary of individual feature inputs to AdvNet and their constituents obtained from sample-specific subnetworks

3.2.4 Adversarial Detection Classifier

We define the $(nm+n+5)$-dimensional vector given by $\mathcal{F}(x) = \text{concat}[\mathcal{F}_1(x), \mathcal{F}_2(x), \mathcal{F}_3(x)]$ as the input feature for each sample x. The constituents of each feature are summarized in Table 3.1. Next, we build a classifier consisting of an input layer, two 1-D convolutional layers with ReLU activation, two fully connected layers with sigmoid activation and a final classification layer with softmax activation. We refer to this model as *AdvNet* in the rest of the paper. Since adversarial samples are slow and computationally expensive to generate, we generate only a small number of adversarial samples and obtain their feature vectors, $\mathcal{F}(x)$. Then, to augment the data, we cut and paste patches from multiple feature vectors in the training set and mix their respective ground truth labels in proportion to the length contributed by each patch. This method is inspired by the CutMix algorithm [Yun *et al.*, 2019] for augmenting image data. Using soft labels obtained by mixing ground truth labels has the advantage of improving generalizability and learning speed in neural networks [Müller *et al.*, 2019].

CHAPTER 4

EXPERIMENTS

4.1 Experimental Setup

4.1.1 Data

We choose the following standard NLU datasets to obtain authentic samples for the evaluation of our proposed approach:

1. SST-2 [Socher *et al.*, 2013]
2. Yelp polarity [Zhang *et al.*, 2015*a*]
3. AG News [Zhang *et al.*, 2015*b*]
4. MRPC [Dolan and Brockett, 2005]
5. IMDb [Maas *et al.*, 2011]
6. SNLI [Bowman *et al.*, 2015]
7. RTE [Wang *et al.*, 2018]
8. MultiNLI [Williams *et al.*, 2018]

The first three are binary sentiment classification datasets while AG News consists of news headlines classified into one of 4 categories (world, sports, business, sci/tech) and MRPC contains sentence pairs with binary labels based on semantic equivalence. The last three datasets contain sentence pairs labelled on textual entailment with RTE having two categories ('entailment' and 'contradiction') while MultiNLI and SNLI have an additional 'neutral' category.

To test our model's performance on diverse adversarial attack types, we generate adversarial samples for the above datasets using 11 attack types, namely,

- **Word-level attacks:** deletion [Feng *et al.*, 2018], antonyms, synonyms [Ren *et al.*, 2019], embedding [Mrkšić *et al.*, 2016], order swap [Pruthi *et al.*, 2019], CLARE [Li *et al.*, 2020*a*], BERT [Li *et al.*, 2020*b*]
- **Character-level attacks:** substitution, deletion, insertion, order swap [Gao *et al.*, 2018].

We note that the set of adversarial samples that we create are indeed difficult to be simultaneously detected by the same classifier. Typical works in adversarial detection tend to report results with much fewer attack types (3-4) or on attacks that are specific to the model architecture. Besides, many of the works only report on 2-3 datasets while we make an elaborate study over 8 diverse datasets. Several examples that we generate entail almost no change in meanings or implications to a human, but result in a fine-tuned BERT model making wrong predictions with high confidence. Besides, the considered attack types take into account most sources of perturbations (for example: due to typos, unintended auto-corrects, mistakes in translation which are accommodated by LM-based attacks: BERT, CLARE, dropping of words during speech transcription, etc.).

We only use generated samples that fool our complete fine-tuned trained model as adversarial samples for further experiments. The dataset for each experiment is prepared by combining the adversarial samples with an equal number of authentic samples for which the complete network generates the correct predictions.

4.1.2 Implementation details

The number of epochs, η, for which the pruning mask of each sample is trained was fixed to 10. The first two convolutional layers in AdvNet have a kernel size of 3

and generate 32 and 16 output feature maps respectively. The two fully connected layers have output dimensions of 32 and 8 with dropout rates of 0.1. We use the binary cross-entropy loss function and the Adam optimizer for training with a learning rate of 0.001. The model is trained for 100 epochs with early stopping. All experiments were performed with a train-validation-test split of 70-10-20 on an NVIDIA Tesla K80 GPU. The implementations of the attack types are sourced from the work by Morris *et al.* [2020].

4.2 Results and Discussion

4.2.1 Feature-specific analysis

Here, we perform experiments that establish the importance of each set of selected features to corroborate its contribution towards adversarial detection.

Pruning mask vector ($\mathcal{F}_1(x)$): The ability of our model to distinguish between authentic and adversarial samples using a sample-specific subnetwork belies the fact that for each input sample, there is a subset of attention heads which is critical to the inference of the target class for that sample. This implies that the pruning mask which selects a subset of important heads can be interpreted in association with the target class for that sample. One way to establish this is by showing that the pruning mask vector is strongly correlated to the target class for each input sample. To achieve this, we project the pruning mask vector $g(x)$ for each authentic sample x onto a 2D-plane using the t-SNE method [van der Maaten and Hinton, 2008] as shown in Figure 4.1(a), (b). We observe distinguishable clusters attributable to the pruning mask vectors of samples with distinct target classes. Thus, we infer that the pruning mask vector is highly discriminative and strongly correlated to

the target class of the sample.

We now extend this experiment to explore clustering patterns when both authentic and adversarial samples are passed. We present the result in Figure 4.1(c). We note that adversarial samples group together with the authentic samples whose true class is the same as their adversarial class. Within clusters corresponding to the same target class, there is a moderate distinction between the adversarial and authentic samples but a better separation is possible when the complete nm-dimensional vector is used as will be demonstrated in further experiments.

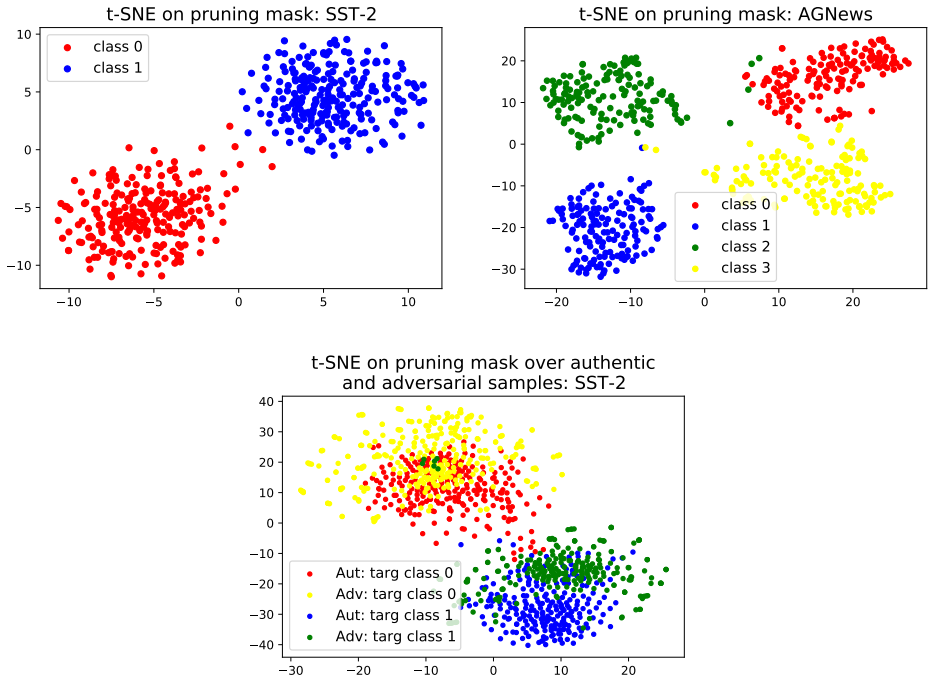


Figure 4.1: Results of applying t-SNE on the pruning mask vector for (a) SST-2, (b) AG News (c) t-SNE plot of pruning mask vectors of authentic and adversarial samples.

Mutated subnetwork features ($\mathcal{F}_2(x)$): By analysing the behavior of the mutated subnetworks on corresponding input samples, we make the following key observations over a majority of the datasets. (i) The mutated subnetwork is more likely to predict the target class output for an authentic sample than an adversarial one as shown in Table 4.1. (ii) It predicts the target class with higher confidence

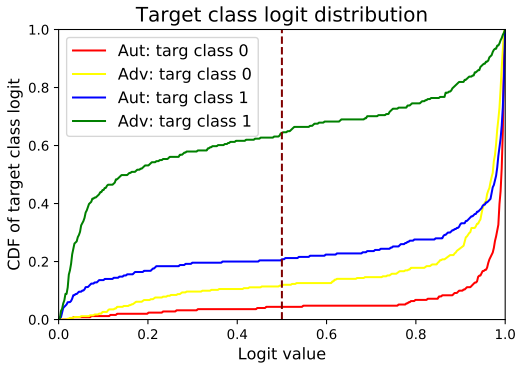


Figure 4.2: Cumulative distribution function (CDF) over the target class output logit of the mutated subnetwork. The large area below the green curve with logit value<0.5 corresponds to a large number of adversarial samples whose mutated subnetworks predict a non-target class.

in case of authentic samples than adversarial ones. (iii) It predicts a non-target class with high confidence for some adversarial samples. Figure 4.2 shows the cumulative distribution function of the logit corresponding to the target class for both kinds of samples. Only 9% of authentic samples had output logit lower than 0.85 while 20% of adversarial samples showed this. Besides, 30% of adversarial samples with output logit higher than 0.85 gave the wrong prediction.

Dataset	Non-target o/p (Mutated)	Switches>1 (Layer-wise)
SST-2	(12.3, 34.2)	(37.9, 54.8)
Yelp	(3.8, 5.3)	(0.83, 1.08)
AG News	(6.6, 22.8)	(3.2, 17.0)
MRPC	(21.3, 24.3)	(10.3, 8.77)
IMDb	(0.33, 2.18)	(0.16, 1.45)
SNLI	(2.83, 96.0)	(11.6, 41.0)
RTE	(24.5, 22.2)	(44.2, 50.9)
MultiNLI	(11.0, 24.8)	(24.3, 42.5)

Table 4.1: Percentages of (authentic, adversarial) samples whose (a) mutated subnetworks generated non-target class predictions; (b) layer-wise outputs showed more than one switch.

Layer-wise auxiliary output features ($\mathcal{F}_3(x)$): We observe that for authentic samples, the layer-wise outputs are mostly consistent with the target class output, whereas for adversarial samples, several of the layer-wise outputs do not match

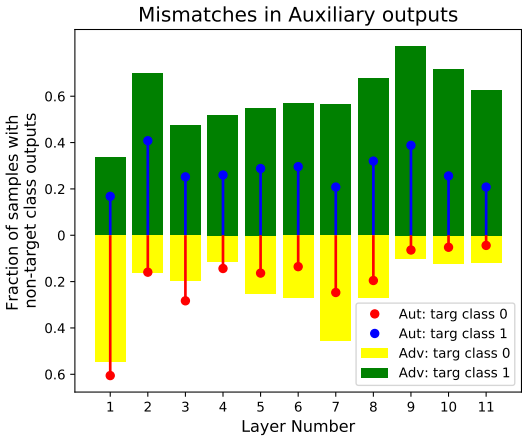


Figure 4.3: Fractions of authentic and adversarial samples that generate a non-target class prediction in each auxiliary layer output.

the target class. On average across datasets, we observed that 52.5% of adversarial samples generate more than 2 auxiliary output predictions that do not match the target class while only 23.1% of authentic samples do the same. The distribution of auxiliary output mismatches (non-target class predictions) across layers of the mutated subnetwork is shown in Figure 4.3. We observe that for most layers, the fraction of authentic samples having non-target class output predictions is higher than adversarial samples. Besides, when traversing the layer-wise outputs in order, it is observed that the output predictions of adversarial samples switch among possible classes more often than authentic samples as shown in Table 4.1.

4.2.2 Performance on Adversarial Detection

Following the observations in the previous section, we use the AdvNet architecture for adversarial classification to leverage distributional variations in the features between authentic and adversarial samples. The performance of our proposed approach on adversarial detection on the BERT-Small and BERT-Base architectures is reported in Table **??**. Across both architectures, we note that the model performs

better on simpler sentence labelling datasets like SST-2 and Yelp when compared to more complex tasks like RTE and MRPC which require comparison between sentences. Existing work [Pande *et al.*, 2021] shows that for simpler tasks, the BERT heads perform discrete non-overlapping roles, while for complex tasks, there is greater overlap in head roles and a few heads perform more than one role. This overlap and multi-skilled nature makes it difficult for the same set of heads to act uniformly across input samples, thus reducing consistency in the generated features. Results on BERT-Base are observed to be better than BERT-Small across tasks due to the higher dimensionality of the feature vector $\mathcal{F}(x)$ and more intrinsic information due to a larger number of layers in BERT-Base.

We also observe that the model achieves high accuracies despite a seemingly low number of generated adversarial samples. This shows the benefit of employing the CutMix algorithm for data augmentation. Besides, we observed that increasing the train set size does give much improvement in accuracy which shows that patterns of variation between the feature vectors are consistent across diverse input text sentences and attack types which AdvNet captures with sample-efficiency.

Authentic Dataset	**# Adversarial Samples**	**AdvNet + CutMix for BERT- Small**			**AdvNet + CutMix for BERT- Base**		
		Prec	**Rec**	**Acc.(%)**	**Prec**	**Rec**	**Acc.(%)**
SST-2	613	0.79	0.78	78.57	0.91	0.90	90.74
Yelp	462	0.76	0.76	76.72	0.87	0.87	87.68
AG News	622	0.77	0.76	76.63	0.86	0.86	86.25
MRPC	712	0.75	0.74	75.05	0.86	0.85	84.61
IMDb	274	0.74	0.74	74.09	0.80	0.81	81.18
SNLI	1046	0.71	0.72	72.07	0.82	0.82	82.50
RTE	477	0.73	0.73	73.64	0.80	0.80	80.43
MultiNLI	548	0.65	0.64	64.26	0.73	0.73	72.61

Table 4.2: Results of Adversarial Detection for BERT - Small and BERT - Base across 8 datasets.

4.2.3 Comparison against other approaches

In Table 4.3, we compare the performance of AdvNet against the following state-of-the-art approaches for robustifying BERT models against adversarial samples:

- **DISP** [Zhou *et al.*, 2019]: In this approach, a classifier called the perturbation discriminator makes a binary decision on each token in the input sentence predicting whether it is an authentic or perturbed one. If none of the tokens are predicted to be perturbed, the input is considered authentic.
- **FGWS** [Mozes *et al.*, 2021]: Here, a word frequency-guided approach is used to identify infrequent words in sentences and replace them with more frequent, semantically similar words. Then, the difference in prediction confidence of the model on both the original and substituted sentences is considered. If this value is above a threshold, the sentence is deemed to be adversarial.
- **TMixADA** and **SMixADA** [Si *et al.*, 2020]: In these methods, a model is fine-tuned and then attacked to generate adversarial samples. Then, the model undergoes adversarial training in which the model is trained using both the identified adversarial samples and linear interpolations of these samples. When evaluated on a test set, the adversarial detection accuracy is determined by the percentage of adversarial samples for which the model generates the correct prediction.

These approaches were tested on the BERT-Base architecture to obtain the precision, recall and accuracy on adversarial detection. We observe that on both the SST-2 and Yelp datasets, our model outperforms the other approaches on most metrics. This indicates that our model achieves substantially better performance than well-known methods. This is primarily due to the fact that our model is less dependent on specific kinds of perturbations and more on the model's behaviour when processing an adversarial sample.

4.2.4 Ablation Study

In this section, we analyse the importance of each feature input to our model for adversarial detection. The results of this study are presented in Table 4.4. The

Dataset	Model	Precision	Recall	Accuracy
SST-2	TMixADA	0.72	0.28	58.55
	SMixADA	0.64	0.20	54.31
	DISP	0.85	0.68	78.73
	FGWS	0.86	0.68	78.34
	AdvNet	**0.90**	**0.91**	**90.74**
Yelp	TMixADA	0.79	0.35	63.15
	SMixADA	0.84	0.49	69.71
	DISP	0.89	0.76	83.15
	FGWS	0.90	0.88	89.21
	AdvNet	**0.87**	**0.87**	**87.68**

Table 4.3: Comparison of AdvNet's performance against other approaches for adversarial detection.

model is trained with different subsets/variants of feature inputs on the adversarial detection task. In all of these experiments, the structure of the classifier remains the same as AdvNet, except that the input dimension is changed depending on the size of the feature subset considered. We observe that passing the pruning mask vector alone, i.e, $\mathcal{F}_1(x)$ results in higher accuracy than when passing $\mathcal{F}_2(x)$ or $\mathcal{F}_3(x)$ individually. This means that the pruning mask vector is the most important feature input to the model. Between using $\mathcal{F}_2(x)$ and $\mathcal{F}_3(x)$, we observe that on certain datasets, the model performs better when $\mathcal{F}_2(x)$ alone is used and on others when $\mathcal{F}_3(x)$ alone is used. However, in cases where using $\mathcal{F}_2(x)$ alone yields better results, the results from using $\mathcal{F}_1(x) \cup \mathcal{F}_3(x)$ is better than using $\mathcal{F}_1(x) \cup \mathcal{F}_2(x)$. This is because there is greater overlap in the adversarial samples detected using features $\mathcal{F}_1(x)$ and $\mathcal{F}_2(x)$, while $\mathcal{F}_3(x)$ helps in the discovery of new adversarial samples that would otherwise remain undetected. Thus, all three feature sets are crucial to the performance of our model. Next, we test the performance of the model when the Boolean pruning mask vector ($\mathcal{F}_1^b(x)$) is used instead of the real-valued vector. The lower accuracy indicates that the real values capture more intrinsic information about the gating pattern. Finally, we compare the model performance with and without using the CutMix algorithm and conclude that augmenting the training set

using CutMix evidently improves the accuracy.

Features	SST-2	Yelp	MRPC	RTE	IMDb	SNLI	MultiNLI
$\mathcal{F}_1$	82.87	80.23	76.35	74.44	74.54	80.83	66.95
$\mathcal{F}_2$	74.07	62.08	68.82	60.88	60.00	57.91	51.30
$\mathcal{F}_3$	64.79	66.01	59.40	56.67	55.45	58.33	60.00
$\mathcal{F}_2, \mathcal{F}_3$	77.46	68.83	61.96	60.23	61.81	56.25	64.78
$\mathcal{F}_1, \mathcal{F}_2$	83.79	86.69	74.35	76.11	74.68	78.83	67.39
$\mathcal{F}_1, \mathcal{F}_3$	85.64	85.19	82.05	77.18	78.18	79.58	67.39
$\mathcal{F}_1^b, \mathcal{F}_2, \mathcal{F}_3$	82.23	83.57	77.35	74.06	74.23	70.41	69.65
$\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ (without CutMix)	85.59	84.30	80.27	77.21	73.78	75.64	66.85
$\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ (AdvNet)	**90.74**	**87.68**	**84.61**	**80.43**	**81.18**	**82.50**	**72.61**

Table 4.4: Table for ablation study of AdvNet.

4.2.5 Alternative for feature input $\mathcal{F}_3$

From Section 3.2.3, we know that layers in BERT encode a hierarchy of linguistic knowledge. As intermediate representations improve over successive layers, the likelihood of the representations of an authentic sample predicting the correct class increases over successive layers of the network. On the contrary, adversarial samples are crafted to fool the final representations generated by the network and it is less likely that they would generate the same adversarial output with all intermediate, sub-optimal feature representations.

Hence, we have explored an alternative approach for generating the feature input $\mathcal{F}_3$. In this method,the auxiliary layer-wise outputs are obtained by introducing classification layers after each encoder layer of the complete fine-tuned network (without masking or mutation). Then, akin to the strategy used for obtaining $\mathcal{F}_3$, an $(n+1)$-dimensional feature vector $\mathcal{F}_4$ is obtained. Table **??** shows the results of an ablation study in which the feature $\mathcal{F}_3$ is replaced with $\mathcal{F}_4$. We observe that while the achieved accuracies are close in both cases, using $\mathcal{F}_3$ yields slightly higher final

performance as it leverages the differential stability of authentic and adversarial samples under mask mutation.

Features	SST-2	Yelp	MRPC	RTE	IMDb	SNLI	MultiNLI
$\mathcal{F}_4$	62.50	66.01	62.39	56.44	57.27	60.00	65.21
$\mathcal{F}_2, \mathcal{F}_4$	77.31	67.48	61.53	69.44	60.47	62.30	52.17
$\mathcal{F}_1, \mathcal{F}_4$	79.62	81.28	76.23	76.66	76.36	77.91	67.39
$\mathcal{F}_1^b, \mathcal{F}_2, \mathcal{F}_4$	85.19	84.72	74.58	72.77	77.27	69.83	68.26
$\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_4$ (without CutMix)	86.22	86.69	78.77	77.22	72.73	71.29	63.91
$\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_4$	**90.27**	**87.19**	**84.61**	**80.55**	**78.81**	**81.25**	**70.43**
$\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ (AdvNet)	**90.74**	**87.68**	**84.61**	**80.43**	**81.18**	**82.50**	**72.61**

Table 4.5: Table for comparative study of using feature $\mathcal{F}_4$ instead of $\mathcal{F}_3$

4.2.6 Heads vary in functionality across layers

In this section, we classify individual attention heads of the network based on their role in processing the input by considering the token pairs which the head attends to with the top 20% highest attention scores. This would help us to further reason out why certain heads are pruned and others retained at various layers of the network. First, we classify the relation between the token pairs into one of the following categories:

1. **Positional:** The two tokens are adjacent to each other in the input sentence.
2. **Delimiter:** At least one of the two tokens is among the delimiter tokens used at sentence boundaries, like the [CLS] and [SEP] tokens in BERT.
3. **Syntactic:** The token pairs have a syntactic relation between them, like a subject-verb pair or a verb-adverb pair.
4. **Task-specific:** One of the tokens is highly informative for the considered task. For example, in the SST-2 sentiment classification task, a token with a sentiment polarity that agrees with the overall sentiment of the input sentence is considered as a task-specific token.
5. **Punctuation:** One of the tokens is a punctuation mark in the input sentence.

For each authentic sample, we classify each head into one of the above categories depending on the majority category to which token pairs with the highest 20% attention scores of that head belong to. We observed that the role of a head remains the same across most samples. Thus, we assign a universal category to each head as the majority category of the head across all samples.

Figure 4.4(a) shows the distribution of head categories across the layers of BERT-Base for the SST-2 dataset, when analysed over the set of authentic samples. We observe that the percentage of task-specific (polarity) tokens increases in the upper 4 layers. This confirms our assertion in Section 3.2.2 that the last few layers are highly task-specific and therefore, attends to tokens that carry the target sentiment of the input sentence. This also substantiates the claim in Section 3.2.3 that there is a build-up of semantic information over successive BERT layers because identifying polarising tokens requires deeper understanding of word meanings.

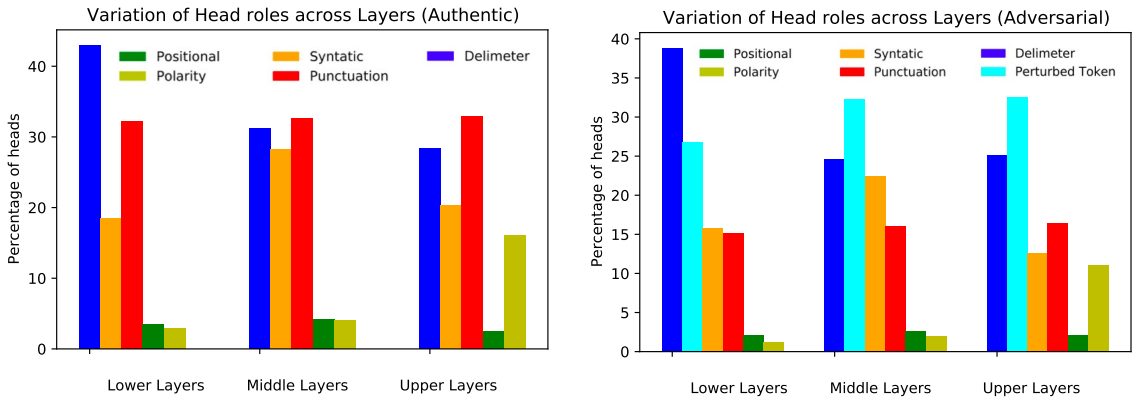


Figure 4.4: Variation of the roles played by important heads across different encoder layers with (a) authentic sample inputs; (b) adversarial sample inputs.

4.2.7 Heads change functionality when exposed to adversarial samples

Next, we follow a process similar to the above to categorize head-roles when adversarial samples are passed through the network. We observe a characteristic shift in trends by which several heads assign high attention scores to the perturbed portions of the sentence. For example, in cases where a random word was inserted, the corresponding inserted token received high attention values from several heads across layers. To account for heads that play this role over a majority of input samples, we introduce a new head-role category called **Perturbed token heads** and display the result in Figure 4.4(b). The high percentage of perturbed token heads across layers of the network substantiates the claim that the perturbed portions of the sequence receive higher attention.

We had earlier observed that the ability of AdvNet to detect adversarial samples based on the gating values belies the fact that these values show deviations from standard patterns when they are generated for an adversarial sample. Based on the above observation, we conclude that these deviations are attributable to the abnormally high attention that is paid to the perturbed portions of the sequence.

4.2.8 Refereeing heads in adversarial detection

In this section, we explore the influence of each gating value in generating the prediction for our adversarial detection model. We make use of the Grad-CAM [Selvaraju *et al.*, 2017] approach to identify critical neurons in the input layer of AdvNet that have large gradients from the target class (authentic or adversarial) flowing through them. Among these critical neurons, we consider those that

correspond to the gating values, i.e, $\mathcal{F}_1(x)$ and call the heads corresponding to these neurons as *refereeing* heads.

From Figure 4.5, we observe that word swap attacks like antonyms, synonyms and embeddings require a greater number of refereeing heads, while character-level attacks need fewer. This is because character-level changes make the token invalid, i.e, the model treats it as an unknown token absent in the vocabulary. Since this brings a more apparent change to the input representation to the model, small deviations from standard gating patterns are sufficient to mislead the model and generate a false output, leading to fewer refereeing heads. Since introducing synonym and embedding based perturbations change the input representation to the model by a smaller extent, larger deviations from the gating pattern are required to block or pass selective chunks of information to mislead the model.

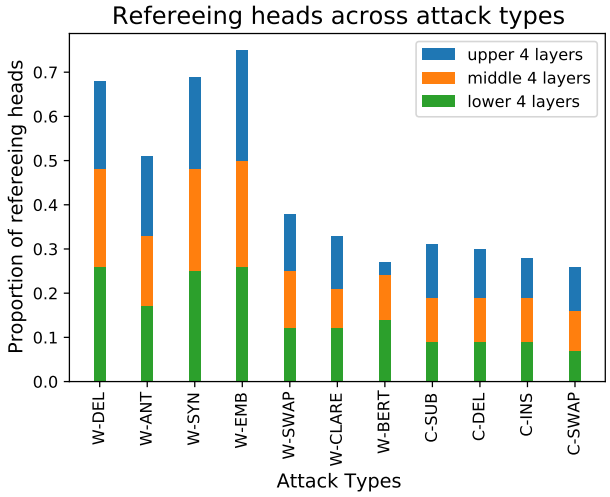


Figure 4.5: Variation of the fraction of refereeing heads used by the adversarial detection model across various adversarial attack types. The split of the refereeing heads across 4 layer subsets is also shown.

4.2.9 Performance on various attack types

In Table 4.6, we compare accuracies separately on each of the 11 considered attack types. We observe that in general, the accuracies on word-level attacks is higher than on character-level attacks. This can be explained using the same arguments as those mentioned in the above section (Section 4.2.8). As word-level attacks bring greater changes to standard gating patterns, it is easier for the AdvNet model to infer these changes from the set of input features. Character-level attacks result in smaller deviations from standard gating patterns when making the pruning mask, which makes it more challenging for the attack to be detected.

We also observe that on some dataset, attack combinations, the model achieves very high accuracies upto 100%, which shows that the model is very good at detecting deviations in gating patterns resulting from specific kinds of perturbations.

Dataset	**Word-level attacks**							**Character-level attacks**			
	DEL	**ANT**	**SYN**	**EMBED**	**SWAP**	**CLARE**	**BERT**	**SUB**	**DEL**	**INS**	**SWAP**
SST-2	0.84	1.00	0.95	1.00	0.75	0.89	0.86	0.92	0.80	0.87	0.89
Yelp	0.75	0.92	0.91	1.00	0.87	0.83	0.83	0.93	0.77	0.88	0.89
MRPC	0.75	0.75	1.00	0.62	1.00	1.00	1.00	1.00	0.67	0.67	1.00
RTE	0.75	0.84	0.86	0.87	0.79	0.79	0.76	0.82	0.76	0.82	0.82
IMDb	0.80	0.67	0.85	0.89	0.80	0.78	0.80	0.94	0.75	0.96	0.79
SNLI	0.61	0.80	0.69	0.88	0.78	0.73	0.63	0.85	0.88	0.65	0.83

Table 4.6: Accuracies across datasets for each attack type. Legend: SUB-substitution, DEL-deletion, SYN-synonym, EMBED-embedding, INS-insertion, SWAP-order swap. Refer Section 4.1.1 for descriptions of attack types.

4.2.10 Comparison of AdvNet with standard classifiers

In this section, we compare the performance of the AdvNet classifier with other standard classifiers. The same set of features are passed as input to each classifier and the results are presented in Table 4.7. Hyperparameters for each classifier are

fine-tuned to obtain the highest accuracy. We observe that AdvNet outperforms the other classifiers. This can be attributed to the following reasons: (i) The 1-D Convolutional layers first capture local relations including those within each pruning mask layer in $\mathcal{F}_1$, (ii) The fully connected layers capture both local and global relations across pruning mask layers in $\mathcal{F}_1$ and across features $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$. The other classifiers either have lower representational power or are unable to capture relations systematically as explained here for AdvNet. Thus, we can conclude that the AdvNet architecture is effective in utilising the features extracted from sample-specific subnetworks to detect adversarial samples.

Adversarial Detection Classifier	**BERT-Small**	**BERT-Base**
Logistic Regression	66.48	84.25
SVM	67.03	83.15
Random Forest	80.27	86.11
GradientBoost	78.02	87.07
AdaBoost	68.68	78.70
Decision Tree	67.03	75.00
FCNN	71.42	86.11
LSTM	66.59	75.61
Transformer	67.69	80.27
AdvNet	**78.57**	**90.74**

Table 4.7: Adversarial detection accuracy of various classifiers on the same feature inputs of samples from the SST-2 dataset.

CHAPTER 5

CONCLUSION AND FUTURE WORK

In this work, we present a novel approach for improving the transparency of Transformer-based neural networks using sample-specific pruning of self-attention heads. We introduce a new approach for adversarial detection that uses the pruning mask as a feature and leverages the differential behaviour of a mutated subnetwork to authentic and adversarial samples.

We benchmark the results of our model against other known approaches for adversarial detection and also compare the performance of our classifier, called AdvNet, against other standard classifiers. Subsequently, we demonstrate that our approach achieves state-of-the-art accuracy in adversarial detection that generalizes well to diverse datasets and attack types.

Further, we explain our observations by analysing the characteristics that differentiate each feature based on whether it is sourced from an authentic or adversarial input sample. We use clustering methods to show the discriminative nature of the pruning masks and the output logits of the mutated network to show how adversarial samples are less stable under mask mutation. We then demonstrate how the consistency of auxiliary layer-wise outputs from the mutated network can be informative of sample authenticity.

Further, we perform an ablation study to establish the individual importance of each feature input. Then, we analyse the functionality of self-attention heads in various layers of the network and correlate it with the heads that are pruned/retained.

Besides, we introduce the idea of refereeing heads which we identify as heads crucial for adversarial detection.

In the future, we hope to extend the applicability of our approach (i) to newer Transformer networks like RoBERTa, DistilBERT and ELECTRA; (ii) to goals beyond classification like regression and language generation and (iii) to tasks beyond natural language understanding including vision and speech tasks. Given that neural network transparency is of more general interest than adversarial detection, we wish to explore further avenues where the learnings from this work can be applied, particularly in developing intrinsic methods for explaining what portions of an input are more critical to the generation of the correct output. Further, given that pruning of self-attention heads has vital applications in improving compute efficiency, we wish to explore how the results of our work can be applied for developing more efficient methods for evaluating and explaining Transformer networks.

REFERENCES

Belinkov, Y., **L. Màrquez**, **H. Sajjad**, **N. Durrani**, **F. Dalvi**, and **J. Glass** (2018). Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. *arXiv preprint arXiv:1801.07772*.

Bowman, S. R., **G. Angeli**, **C. Potts**, and **C. D. Manning**, A large annotated corpus for learning natural language inference. *In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 2015. URL https://www.aclweb.org/anthology/D15-1075.

Budhraja, A., **M. Pande**, **P. Nema**, **P. Kumar**, and **M. M. Khapra**, On the weak link between importance and prunability of attention heads. *In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 2020. URL https://www.aclweb.org/anthology/2020.emnlp-main.260.

Chen, L., **W. Ruan**, **X. Liu**, and **J. Lu**, SeqVAT: Virtual adversarial training for semi-supervised sequence labeling. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.777.

Cheng, Y., **L. Jiang**, **W. Macherey**, and **J. Eisenstein**, AdvAug: Robust adversarial augmentation for neural machine translation. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2020. URL https://www.aclweb.org/anthology/2020.acl-main.529.

Dolan, W. B. and **C. Brockett**, Automatically constructing a corpus of sentential paraphrases. *In Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*. 2005. URL https://www.aclweb.org/anthology/I05-5002.

Feng, S., **E. Wallace**, **A. Grissom II**, **M. Iyyer**, **P. Rodriguez**, and **J. Boyd-Graber**, Pathologies of neural models make interpretations difficult. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Brussels, Belgium, 2018. URL https://www.aclweb.org/anthology/D18-1407.

Gao, J., **J. Lanchantin**, **M. L. Soffa**, and **Y. Qi**, Black-box generation of adversarial text sequences to evade deep learning classifiers. *In 2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018.

Goldberg, Y. (2019). Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

Hewitt, J. and **C. D. Manning**, A structural probe for finding syntax in word representations. *In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 2019. URL https://www.aclweb.org/anthology/N19-1419.

Jawahar, G., **B. Sagot**, and **D. Seddah**, What does bert learn about the structure of language? *In ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*. 2019.

Koh, P. W. and **P. Liang**, Understanding black-box predictions via influence functions. *In* **D. Precup** and **Y. W. Teh** (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*. PMLR, 2017. URL http://proceedings.mlr.press/v70/koh17a.html.

Kovaleva, O., **A. Romanov**, **A. Rogers**, and **A. Rumshisky**, Revealing the dark secrets of BERT. *In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 2019. URL https://www.aclweb.org/anthology/D19-1445.

Lakshminarayanan, C. and **A. Vikram Singh**, Neural path features and neural path kernel : Understanding the role of gates in deep learning. *In* **H. Larochelle**, **M. Ranzato**, **R. Hadsell**, **M. F. Balcan**, and **H. Lin** (eds.), *Advances in Neural Information Processing Systems*, volume 33. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/37f76c6fe3ab45e0cd7ecb176b5a046d-Paper.pdf.

Li, D., **Y. Zhang**, **H. Peng**, **L. Chen**, **C. Brockett**, **M.-T. Sun**, and **B. Dolan** (2020*a*). Contextualized perturbation for textual adversarial attack. *arXiv preprint arXiv:2009.07502*.

Li, L., **R. Ma**, **Q. Guo**, **X. Xue**, and **X. Qiu** (2020*b*). Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Li, Y., **T. Tang**, **C.-J. Hsieh**, and **T. Lee** (2021). Detecting adversarial examples with bayesian neural network. *arXiv preprint arXiv:2105.08620*.

Liu, N., **H. Yang**, and **X. Hu**, Adversarial detection with model interpretation. Association for Computing Machinery, New York, NY, USA, 2018. ISBN 9781450355520. URL https://doi.org/10.1145/3219819.3220027.

Liu, N. F., **M. Gardner**, **Y. Belinkov**, **M. E. Peters**, and **N. A. Smith** (2019). Linguistic knowledge and transferability of contextual representations. *arXiv preprint arXiv:1903.08855*.

Liu, X., **H. Cheng**, **P. He**, **W. Chen**, **Y. Wang**, **H. Poon**, and **J. Gao** (2020). Adversarial training for large neural language models. *ArXiv*, **abs/2004.08994**.

Louizos, C., **M. Welling**, and **D. P. Kingma** (2017). Learning sparse neural networks through *l_0* regularization. *arXiv preprint arXiv:1712.01312*.

Maas, A. L., **R. E. Daly**, **P. T. Pham**, **D. Huang**, **A. Y. Ng**, and **C. Potts**, Learning word vectors for sentiment analysis. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, 2011. URL http://www.aclweb.org/anthology/P11-1015.

Michel, P., **O. Levy**, and **G. Neubig**, Are sixteen heads really better than one? *In* **H. Wallach**, **H. Larochelle**, **A. Beygelzimer**, **F. d'Alché-Buc**, **E. Fox**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

Morris, J. X., **E. Lifland**, **J. Y. Yoo**, **J. Grigsby**, **D. Jin**, and **Y. Qi** (2020). Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

Mozes, M., **P. Stenetorp**, **B. Kleinberg**, and **L. Griffin**, Frequency-guided word substitutions for detecting textual adversarial examples. *In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 2021. URL https://www.aclweb.org/anthology/2021.eacl-main.13.

Mrkšić, N., **D. O. Séaghdha**, **B. Thomson**, **M. Gašić**, **L. Rojas-Barahona**, **P.-H. Su**, **D. Vandyke**, **T.-H. Wen**, and **S. Young** (2016). Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Müller, R., **S. Kornblith**, and **G. E. Hinton**, When does label smoothing help? *In* **H. Wallach**, **H. Larochelle**, **A. Beygelzimer**, **F. d'Alché-Buc**, **E. Fox**, and **R. Garnett** (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/f1748d6b0fd9d439f71450117eba2725-Paper.pdf.

Pande, M., **A. Budhraja**, **P. Nema**, **P. Kumar**, and **M. M. Khapra** (2021). The heads hypothesis: A unifying statistical approach towards understanding multi-headed attention in bert. *arXiv preprint arXiv:2101.09115*.

Pang, T., **C. Du**, **Y. Dong**, and **J. Zhu**, Towards robust detection of adversarial examples. NIPS'18. Curran Associates Inc., Red Hook, NY, USA, 2018.

Pruthi, D., **B. Dhingra**, and **Z. C. Lipton** (2019). Combating adversarial misspellings with robust word recognition. *arXiv preprint arXiv:1905.11268*.

Ren, S., **Y. Deng**, **K. He**, and **W. Che**, Generating natural language adversarial examples through probability weighted word saliency. *In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 2019. URL https://www.aclweb.org/anthology/P19-1103.

Selvaraju, R. R., **M. Cogswell**, **A. Das**, **R. Vedantam**, **D. Parikh**, and **D. Batra**, Grad-cam: Visual explanations from deep networks via gradient-based localization. *In 2017 IEEE International Conference on Computer Vision (ICCV)*. 2017.

Si, C., **Z. Zhang**, **F. Qi**, **Z. Liu**, **Y. Wang**, **Q. Liu**, and **M. Sun** (2020). Better robustness by more coverage: Adversarial training with mixup augmentation for robust fine-tuning. *ArXiv*, **abs/2012.15699**.

Socher, R., **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*. Association for Computational Linguistics, Seattle, Washington, USA, 2013. URL https://www.aclweb.org/anthology/D13-1170.

van Aken, B., **B. Winter**, **A. Löser**, and **F. A. Gers**, How does bert answer questions? a layer-wise analysis of transformer representations. CIKM '19. Association for Computing Machinery, New York, NY, USA, 2019. ISBN 9781450369763. URL https://doi.org/10.1145/3357384.3358028.

van der Maaten, L. and **G. Hinton** (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, **9**(86), 2579–2605. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Vaswani, A., **N. Shazeer**, **N. Parmar**, **J. Uszkoreit**, **L. Jones**, **A. N. Gomez**, **L. Kaiser**, and **I. Polosukhin** (2017). Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Voita, E., **D. Talbot**, **F. Moiseev**, **R. Sennrich**, and **I. Titov** (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Wang, A., **A. Singh**, **J. Michael**, **F. Hill**, **O. Levy**, and **S. Bowman**, GLUE: A multi-task benchmark and analysis platform for natural language understanding. *In Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Brussels, Belgium, 2018. URL https://www.aclweb.org/anthology/W18-5446.

Wang, H., **Z. Wu**, **Z. Liu**, **H. Cai**, **L. Zhu**, **C. Gan**, and **S. Han**, HAT: Hardware-aware transformers for efficient natural language processing. *In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 2020*a*. URL https://www.aclweb.org/anthology/2020.acl-main.686.

Wang, J., **G. Dong**, **J. Sun**, **X. Wang**, and **P. Zhang**, Adversarial sample detection for deep neural network through model mutation testing. *In 2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019.

Wang, Y., **H. Su**, **B. Zhang**, and **X. Hu** (2020*b*). Interpret neural networks by extracting critical subnetworks. *IEEE Transactions on Image Processing*, **29**, 6707–6720.

Williams, A., **N. Nangia**, and **S. Bowman**, A broad-coverage challenge corpus for sentence understanding through inference. *In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 2018. URL https://www.aclweb.org/anthology/N18-1101.

Yu, P., **K. Song**, and **J. Lu**, Generating adversarial examples with conditional generative adversarial net. *In 2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018.

Yuan, X., **P. He**, **Q. Zhu**, and **X. Li** (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, **30**(9), 2805–2824.

Yun, S., **D. Han**, **S. J. Oh**, **S. Chun**, **J. Choe**, and **Y. Yoo**, Cutmix: Regularization strategy to train strong classifiers with localizable features. *In Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019.

Zhang, X., **J. Zhao**, and **Y. LeCun** (2015*a*). Character-level Convolutional Networks for Text Classification. *arXiv:1509.01626 [cs]*.

Zhang, X., **J. J. Zhao**, and **Y. LeCun**, Character-level convolutional networks for text classification. *In NIPS*. 2015*b*.

Zhou, Y., **J.-Y. Jiang**, **K.-W. Chang**, and **W. Wang** (2019). Learning to discriminate perturbations for blocking adversarial attacks in text classification. *arXiv preprint arXiv:1909.03084*.