Input-specific Attention Subnetworks for Adversarial Detection

Anonymous ACL submission

Abstract

Self-attention heads are characteristic of Transformer models and have been well studied for interpretability and pruning. In this work, we demonstrate an altogether different utility of attention heads, namely for adversarial detection. Specifically, we propose a method to construct input-specific attention subnetworks (IAS) from which we extract three features to discriminate between authentic and adversarial inputs. The resultant detector significantly improves (by over 7.5%) the state-of-the-art adversarial detection accuracy for the BERT encoder on 10 NLU datasets with 11 different adversarial attack types. We also demonstrate that our method (a) is more accurate for larger models which are likely to have more spurious correlations and thus vulnerable to adversarial attack, and (b) performs well even with modest training sets of adversarial examples.

1 Introduction

Self-attention heads are characteristic of Transformer models. Individual attention heads are interpretable in different ways. One, for a token in an input sentence, we can visualize the attention paid by a head to all other tokens. Such attention patterns are attractive linguistically and have come to define roles for attention heads (Pande et al., 2021). Two, the output of attention heads from various layers can be probed for their ability to encode information related to the "NLP pipeline" (Jawahar et al., 2019; Tenney et al., 2019; van Aken et al., 2019). Three, attention patterns of heads can represent knowledge learnt by a teacher model when distilling to a smaller student model (Jiao et al., 2020). While individual attention heads are interpretable in the above ways, it is found that attention heads in models such as BERT are over-provisioned and can be pruned. For instance, Michel et al. (2019) showed that a model with 16 attention heads per layer can be pruned to just one. Voita et al. (2019) and Budhraja et al. (2020) have shown similar results with different pruning techniques across tasks.

In the above methods, while interpretation of attention heads is input-specific, pruning of heads is input-agnostic. Can these two be combined, i.e., can we prune attention heads in an input-specific manner creating opportunities for interpretation? We explore this idea to identify an altogether different utility of attention heads - namely *adversarial detection* which is the task of differentiating between authentic and adversarial inputs. Specifically, we propose a method to obtain an *input-specific attention subnetwork* (IAS), which is a subnetwork where a subset of attention heads is masked without affecting the output of the model for that input. Such subnetworks could vary across inputs representing how the model works for each input. This is particularly important for adversarial detection, as adversarial inputs do not reveal themselves in *what* the model outputs but may leave tell-tale signs in *how* the model computes this output.

In this work, we present a technique to efficiently compute IAS and demonstrate its utility in adversarial detection with significantly improved accuracy over all current methods. To this end, we propose three sets of features from IAS. The first feature, F_{mask}, is simply the attention mask that identifies if an attention head is retained or pruned in IAS. The second feature, F_{flip}, characterizes the output of a "mutated" IAS obtained by toggling the mask used for attention heads in the middle layers of IAS. The third feature, F_{lw}, characterizes the outputs of IAS as obtained layer-wise with a separately trained classification head for each layer. We train a classifier, called AdvNet, with these features as inputs to predict if an input is adversarial.

We report results on 10 NLU tasks from the GLUE benchmark (SST2, MRPC, RTE, SNLI, MNLI, QQP, QNLI) and elsewhere (Yelp, AG News, IMDb). For each of these tasks, we first create a benchmark of adversarial examples combining 11 attack methodologies like Word order swap (Pruthi et al., 2019), embedding swap (Mrkšić et al., 2016), word deletion (Feng et al., 2018), etc. In total, the benchmark contains 5,686 adversarial examples across tasks and attack types. To the best of our knowledge, this dataset is the most extensive benchmark available on the considered tasks. Across all these tasks and attack types, we compare our adversarial detection technique against state-of-the-art methods such as DISP (Zhou et al., 2019), NWS (Mozes et al., 2021), and FGWS (Mozes et al., 2021). Our method establishes the best results in all tasks and attack types, with an average improvement of 7.45% over the best method for each task. Our detector achieves an accuracy of 80–90% across tasks suggesting effective defense against adversarial attacks.

Having established the utility of attention heads for adversarial detection, we perform several ablation studies. First, we compare different combinations of the features demonstrating that they are mutually informative and thus combining them all works best. Second, we show that CutMix data augmentation (Yun et al., 2019) improves accuracy, demonstrating the first use of this method in adversarial detection in NLP tasks. Third, we show that the detector is more accurate as the size of the language model scales. This is encouraging because larger language models are expected to have increased spurious correlations and thus are more vulnerable to adversarial attacks. Fourth, we show that the detector performs well even for modest training sizes of adversarial examples, suggesting effective generalization. In summary, we propose a novel relation between attention heads and adversarial detection. The effectiveness of the resultant detector establishes that the mask of attention heads captures critical information about *how* a Transformer model works for a given input.

The rest of the paper is organized as follows. We detail our core method of computing IAS in the next section. In Section 3 we discuss the features from IAS for adversarial detection. We detail the experimental setup along with the dataset creation process in Section 4. We present our results in Section 5 and conclude in Section 6.

2 Input-Specific Attention Subnetworks

In this section, we describe Input-specific Attention Subnetworks (IAS) and the computational approach to identify IAS for a given input.

2.1 Notation

We consider a BERT-style encoder model where each layer consists of multi-headed self-attention and position-wise FFN. Let an input x consist of T tokens each represented by d_v-dimensional vectors. Let $X_j \in \mathbb{R}^{T \times d_v}$ be the representation at the input of the j^{th} layer. Let $W_{ji}^Q, W_{ji}^K, W_{ji}^V$ be the projection matrices of the i^{th} self-attention head in the j^{th} layer. We define $Q_{ji} = X_j W_{ji}^Q, K_{ji} = X_j W_{ji}^K, V_{ji} = X_j W_{ji}^V$ as the query, key, and value corresponding to the head respectively. Each self-attention head performs a scaled dot-product attention on the query, key, and value to generate the head's output. The output of all the heads in a layer are concatenated and passed through the FFN.

$$\text{Head}_{ji}(X_j) = \text{softmax}\left(\frac{Q_{ji}K_{ji}^T}{\sqrt{d_k}}\right)V_{ji} \quad (1)$$

$$\text{Layer}_j(X_j) = \text{concat}_i[\text{Head}_{ji}(X_j)]W_j^O \quad (2)$$

where d_k is the dimensionality of each key vector and W_j^O is a learnable parameter.

A pre-trained model is fine-tuned on a specific task, such as sentiment classification. Let θ be the set of trainable network parameters which are optimized to minimize a task-specific training loss for each input x:

$$\mathcal{L}^\theta(x) = \mathcal{L}_{CE}(f(x,\theta), y), \quad (3)$$

where $f(\cdot)$ is the function computed by the model with parameters θ for input x, $\mathcal{L}_{CE}$ is the standard cross-entropy loss function and y is the expected model output for input x. The overall training loss is averaged across all $|x|$ inputs, i.e., $\mathcal{L}^\theta = \frac{1}{|x|}\sum_x \mathcal{L}^\theta(x)$. Let $\widehat{f}(\cdot)$ represent the output class generated from $f(\cdot)$ and θ^* be the set of optimal network parameters obtained after training.

2.2 Representing IAS

In an IAS, a subset of attention heads are pruned. We represent a continuous relaxation of pruning by modifying Eqn. 1 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

$$\text{Layer}_j^m(X_j) = \text{concat}_i[g_{ji} \cdot \text{Head}_{ji}(X_j)]W_j^O \quad (4)$$

During inference, we constrain the gating values to be binary to characterize either exclusion or inclusion of a head: g_{ji} is replaced by $g_{ji}^b \in \{0,1\}$ which defines the attention mask for the input x: $g^b(x) = \{g_{ji}^b\} \in \{0,1\}^{nm}$, where n is the number

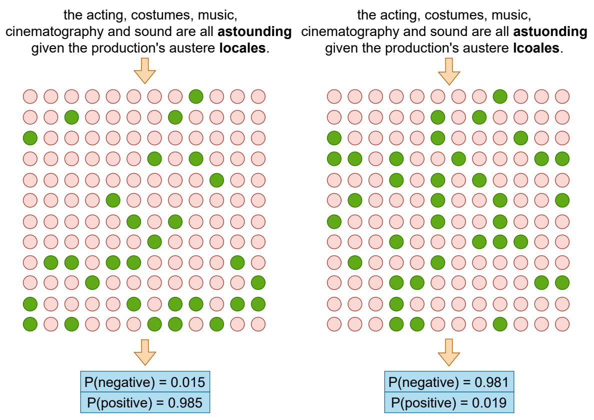
the acting, costumes, music, cinematography and sound are all **astounding** given the production's austere **locales**.
the acting, costumes, music, cinematography and sound are all **astounding** given the production's austere **lcoales**.
P(negative) = 0.015
P(positive) = 0.985
P(negative) = 0.981
P(positive) = 0.019

Figure 1: The IAS (with active heads in green) computed for two inputs on the SST-2 task, left is authentic while right is adversarial. Notice how a small adversarial perturbation in the input leads to very distinct subnetworks being computed. The class predicted by each IAS agrees with the prediction of the full network.

of layers and m is the number of heads per layer. We represent the output class predicted by the IAS for an input x by $\widehat{f_g}(x, \theta^*, g^b)$. We call the subset of attention heads that are assigned a gating value of 1 as *active* heads and note that the active heads jointly define a subnetwork, called IAS. We illustrate IAS with an example. Figure 1 shows the BERT-Base model with 12 layers and 12 heads per layer. For two specific inputs, the corresponding attention masks are shown with their active heads in green. Thus, IAS is input-specific and characterizes how the model processes the input in a relatively low-dimensional space of $[0, 1]^{144}$.

2.3 Computing IAS

We compute IAS by treating the gating values as free variables to optimize the task-specific loss (Eqn. 7) for a given input x. In this optimization, the network parameters θ are frozen. Each gating value, g_{ji} is defined as $g_{ji} = f_{HC}(p_{ji})$, where p_{ji} is the free variable that is optimized and f_{HC} is a version of the hard concrete distribution (Louizos et al., 2017) given as $\frac{1}{1+e^{\alpha\cdot(log(1-p_{ji})-log(p_{ji}))}}$, where, α=6 gave the best results for our work. Let g be the gating vector as optimized by minimizing the loss for a specific input. We need to enforce that g is binary. Unlike approaches by Voita et al. (2019) and Wang et al. (2020), we do not include a regularization term in the training objective. Instead, we retain only those heads for which the gating values ascend the fastest towards 1, as measured after a certain η number of epochs. Specifically, each binary value g^b_{ji} is derived from g_{ji} after η epochs as:

$$g^b_{ji}(x) = \begin{cases} 1, & \text{if } g_{ji}(x) \geq \beta \cdot \max(g(x)) \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where, $\beta(< 1)$ is a thresholding parameter and $\max(g(x))$ is the largest among nm gating values. For our work, we set $\eta = 10$ and $\beta = 0.8$.

Two exceptional cases may arise. First, if the binary gating values of all heads in a layer are thresholded to 0, then the largest gating value in that layer is forced to 1 to ensure information flows through the network. Second, if the IAS predicts the wrong class for that input, then β is reduced successively in steps of 0.2 until the output of the IAS is correct. For 98% of the inputs, the subnetwork predicted the target class within $\beta = 0.6$.

3 Model for Adversarial Detection

In this section, we explain how we extract features from the IAS and the design of the classifier for adversarial detection. We use the term *target class* to refer to the class predicted by the complete fine-tuned network for an input. For authentic inputs, this translates to the true class while for adversarial inputs, this refers to the adversarial class that the model is fooled into predicting.

3.1 Attention mask $F_{\textbf{mask}}$

The IAS identifies a subnetwork through which important information flows for a particular input. We hypothesize that this flow could be different for authentic and adversarial inputs. Thus, the first feature we extract, F_{mask}, is just the pre-activation value p for the gating values of each head in the IAS. Thus, for a BERT-base model with 12 layers and 12 heads per layer, F_{mask} is a 144 dimensional vector. We also define F_{bmask} which uses the binary gated values g^b instead of the real-values.

3.2 Features from flipping heads in IAS $F_{\textbf{flip}}$

Adversarial inputs rely heavily on the network architecture and specific parameter combinations to fool the model (Wang et al., 2019). Hence, slight changes to network parameters can render an adversarial perturbation non-adversarial. We thus hypothesize (and later illustrate in Section 5.2) that if we flip some of the heads in the IAS, it could significantly change the output for adversarial inputs but not by as much for authentic inputs. Which heads should we flip? We take motivation from studies that show that 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. In contrast, the initial layers are responsible for phrase-level understanding while the last few layers are highly task-specific (Jawahar et al., 2019). Hence, we choose to flip the gating values g^b of heads in the *middle layers* of IAS, specifically, the middle $\lceil \frac{n}{3} \rceil$ layers, i.e., we drop heads that were earlier active and include earlier inactive heads. We denote the modified gating vector after flipping as g^f.

$$g^f_{ji} = \begin{cases} g^b_{ji}, & \text{if } j \leq \lfloor \frac{n}{3} \rfloor \text{ or } j \geq 2\lceil \frac{n}{3} \rceil \\ 1 - g^b_{ji}, & \text{if } \lceil \frac{n}{3} \rceil \leq j < 2\lceil \frac{n}{3} \rceil \end{cases} \quad (6)$$

We run each input x through this mutated subnetwork and obtain a 4-dimensional feature vector, F_{flip} consisting of the predicted class given by $\widehat{f}_g(x, \theta^*, g^f)$, the target class y, the confidence of prediction, and a flag asserting equality between predicted and target classes.

3.3 Layer-wise auxiliary features F_{lw}

Studies (Wang et al., 2020; Xie et al., 2019) have shown that intermediate representations of adversarial inputs diverge from those of authentic inputs as we progress into deeper layers. This indicates that layer-wise information may be discriminative of adversarial inputs. Hence, instead of having a single classifier head processing the output of the final layer, we propose to train a classifier head at the output of each layer and use the classes predicted by them as features in adversarial detection. Specifically, on the fine-tuned complete model, we freeze the standard model parameters to θ^* and train $n-1$ classifiers separately with a classifier head attached to each of the first $n-1$ layers to predict the target class. Following the convention in Eqn. 7, the training loss for the l^{th} classifier head with parameters Ω^l on input x is given by:

$$\mathcal{L}^{\Omega^l}(x) = \mathcal{L}_{CE}(f^l_g(x, \theta^* \cup \Omega^l, \{1\}^{nm}), y), \quad (7)$$

where $f^l_g(\cdot)$ gives the output class computed by the l^{th} classification head of a network with gating vector g. The overall training loss is given by $\mathcal{L}^{\Omega} = \frac{1}{(n-1)|x|} \sum_x \sum_l \mathcal{L}^{\Omega^l}(x)$. Let Ω^* be the set of optimal parameters obtained after training.

Then for a given input, we construct the IAS after flipping heads as given by the gating vector g^f and compute the outputs of the $n-1$ layer-wise classifiers, i.e., the output of the l^{th} classifier head is given by $\widehat{f}^l_g(x, \theta^* \cup \Omega^{*l}, g^f)$. We then create an $n+1$ dimensional feature, F_{lw}, which consists of the $n-1$ output labels with two other scalars: (a) the number of these outputs that match the target class, and (b) the number of times these outputs change when traversed in the order of layers.

In summary, we compute the features as follows. First, the model is fine-tuned on the task. Then, layer-wise classification heads are trained while keeping the model parameters frozen. Thus, given an input, we first optimize and compute IAS from which we extract F_{mask}. Then, the gating values of the middle layers are flipped and we extract F_{flip}. Finally, on the IAS with flipped heads, layer-wise classifier outputs are used to extract F_{lw}.

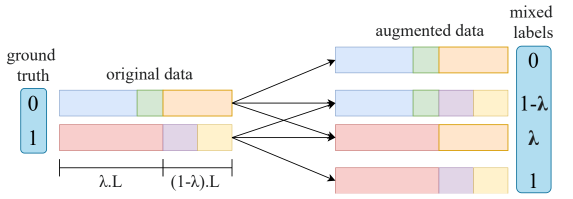

Figure 2: Demonstration of CutMix used to mix patches from two input feature vectors of length L each.

3.4 Classifier for adversarial detection

We refer to our classifier as *AdvNet*, which takes as input, an $(nm + n + 5)$-dimensional vector $F(x)$ which is the concatenation of F_{mask}, F_{flip}, F_{lw} and generates a binary output classifying if a given input is authentic or adversarial. AdvNet consists of two 1-D convolutional layers with ReLU activation, two fully connected layers with sigmoid activation, and a final classification layer with softmax activation. Since adversarial inputs are slow and computationally expensive to generate, we employ the CutMix algorithm (Yun et al., 2019) for data augmentation. In CutMix, we slice out patches from feature vectors of multiple inputs in the training set, each of which could be authentic or adversarial, and combine them to generate new feature vectors. Their respective ground truth labels are mixed in proportion to the length contributed by each patch (see Figure 2). Formally, if $\{x_i\}_{i=1}^R$ is a random subset of training set samples, an augmented feature vector from CutMix is defined by $F(\widetilde{x}) = \text{concat}_i[F(x_i)[p_i : p_{i+1}]]$, where $0 = p_1 < p_2 < \ldots < p_{R+1} = nm + n + 5$ and the mixed ground truth label is given by $\widetilde{y} = \sum_i y_i(p_{i+1} - p_i)$. Using soft labels by mixing ground truth labels also offers better generalization and learning speed (Müller et al., 2019).

4 Experimental Setup

4.1 NLU tasks for evaluation

We choose the following 10 standard NLU tasks for performing our experimental studies: SST-2 (Socher et al., 2013), Yelp polarity (Zhang et al., 2015a), IMDb (Maas et al., 2011), AG News (Zhang et al., 2015b), MRPC (Dolan and Brockett, 2005), RTE (Wang et al., 2018), MNLI (Williams et al., 2018), SNLI (Bowman et al., 2015), QQP[1] and QNLI (Wang et al., 2018; Rajpurkar et al., 2016). We refer the reader to Appendix A for further details on these datasets.

4.2 Dataset creation

To perform adversarial detection, we require a combined set of authentic and adversarial samples for each task. First, we fine-tune a BERT-based model for each task using its publicly available training set. Then, samples from its test set for which the fine-tuned model makes correct predictions constitute the set of authentic samples for that task. Second, we generate adversarial samples by attacking the fine-tuned model using a broad set of 11 hard attack types to comprehensively test AdvNet's performance and its generalizability to diverse perturbations. The attacks include **word-level attacks:** deletion (Feng et al., 2018), antonyms, synonyms, embeddings (Mrkšić et al., 2016), order swap (Pruthi et al., 2019), PWWS (Ren et al., 2019), TextFooler (Jin et al., 2020) and **character-level attacks:** substitution, deletion, insertion, order swap (Gao et al., 2018). We use the popular TextAttack framework (Morris et al., 2020) for implementations of these attacks. Resulting perturbed samples that successfully fool our complete fine-tuned model constitute the set of adversarial samples for that task. On the combined authentic and adversarial set, we make a 70-10-20 split for creating training, validation and test sets for adversarial detection using AdvNet. We will publicly release this dataset containing a total of 5,686 adversarial inputs across tasks and attack types.

4.3 Implementation details

Our adversarial detection model, AdvNet, contains two 1D convolutional layers followed by two fully connected layers. The two convolutional layers have a kernel size of 3 and generate 32 and 16 output feature maps. The two fully connected layers have output dimensions of 32 and 16 with dropout rates of 0.1. We use the binary cross-entropy loss function and the Adam optimizer with a learning rate of 0.001. We train the model for 100 epochs with early stopping on an NVIDIA K80 GPU.

[1] `quoradata.quora.com/First-Quora-Dataset-Release-Question-Pairs`

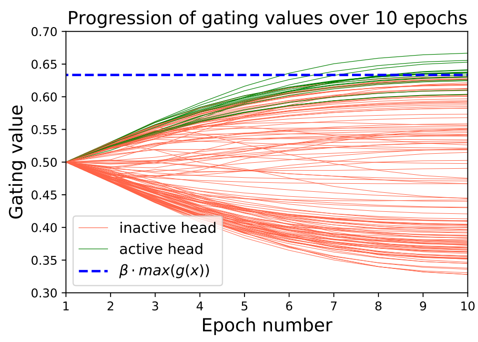

Figure 3: The trajectory of gating values of individual heads during the optimization to compute IAS. Only a few heads (in green) reach the threshold and remain active in IAS.

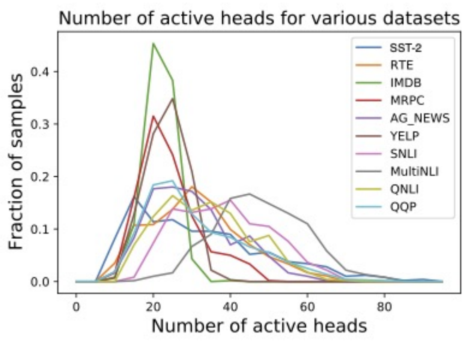

Figure 4: Fraction of inputs with a given number of active heads from BERT-Base. Notice that in most cases, only 20-40 heads out of 144 remain active.

5 Results & Discussion

In this section, we first analyse the IAS (Section 5.1) and the constituent features of AdvNet (Section 5.2). We then perform a comparative study with state-of-the-art adversarial detection methods (Section 5.3). Lastly, we perform ablation studies to understand the effect of task, model size, feature combinations and training set attacks on the performance of AdvNet (Section 5.4). Unless otherwise stated, the plots pertain to experiments on the SST-2 dataset with the BERT-Base model.

5.1 Active heads in IAS

We first check the number of active heads in IAS for a given input. To do so, we plot the progression

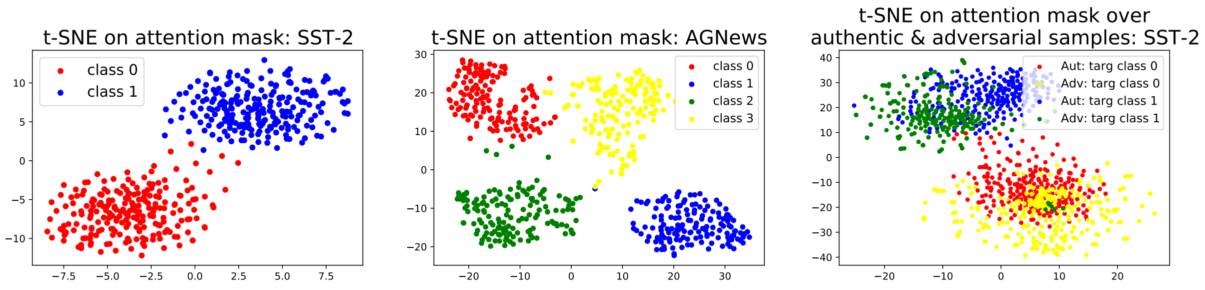

Figure 5: Projections with t-SNE on the attention mask for (a) SST-2, (b) AG News, and (c) authentic and adversarial inputs. Projection of attention masks are strongly discriminative of class and weakly of adversarial inputs.

of gating values with epochs when optimizing them for a given input (see Figure 3). We observe that only a small fraction of heads (shown in green) are active at the end of the optimization process, thus resulting in a sparse vector. The green curves that are below the blue (threshold) line correspond to the two exceptional cases discussed at the end of Section 2.3. While the above plot was for a single randomly selected input, in Figure 4 we show the fraction of inputs with a given number of active heads for all the datasets used in this work. The relatively small modes and the right skew distributions imply that the extracted IAS are often sparse.

5.2 Feature-specific analysis

We now analyze the individual effectiveness of the three features proposed in Section 3.

Attention mask (F_{mask}). We first show that the attention mask is strongly correlated with the input's target class. To do so, we project the binary vector $g(x)$ for each authentic input x onto a 2D-plane using the t-SNE method (van der Maaten and Hinton, 2008) as shown in Figure 5(a), (b). We observe that inputs from different classes separate into distinctly separate clusters. Thus, the attention mask is discriminative of an input's target class as the choice of active heads depends on it. Interestingly, even if the attention computed for the same word location in two distinct inputs are the same, the heads attending to each word and responsible for generating different output classes are different.

We present a similar plot with both authentic and adversarial inputs in Figure 5(c). We note that adversarial inputs group together with the authentic inputs whose true class is the same as their adversarial/target class. Within clusters of the same target class, there is a only a moderate distinction between adversarial and authentic inputs. But we show in further experiments that a better separation is possible when the complete nm-dimensional vector is used as opposed to a 2D projection.

Features from flipping heads in IAS (F_{flip}). For each of the datasets, we compute the percentage of authentic and adversarial inputs which generated non-target class predictions. We find that the mutated IAS after flipping heads in the middle layers is more likely to predict the correct target class output for an authentic input than an adversarial one. We also study the confidence in these predictions as detailed with a CDF plot of the logits in Appendix D. We observe that F_{flip} predicts the target class with higher confidence in case of authentic inputs than adversarial ones. Specifically, only 9% of authentic inputs had prediction confidence lower than 0.85 as compared to 20% of adversarial inputs. Further, it predicts a non-target class with high confidence for some adversarial inputs. For example, 30% of adversarial inputs with prediction confidence higher than 0.85 gave the wrong prediction. In contrast, flipping the initial/final layers of the IAS instead of the middle layers did not significantly change the model prediction for either authentic or adversarial samples, making it difficult it to distinguish them.

Layer-wise auxiliary features (F_{lw}). In Figure 6, we plot the distribution of auxiliary output mismatches (non-target class predictions) across network layers. We observe that for most layers, the fraction of authentic inputs having target class predictions is higher than adversarial inputs. The differences are particularly large for the last few layers. On average across datasets, we observed that 52.5% of adversarial inputs generate more than 2 auxiliary output predictions that do not match the target class while only 23.1% of authentic inputs do the same. Additionally, when traversing the layer-wise outputs in order, we observed that the output predictions of adversarial inputs switch among possible classes more often than for authentic inputs

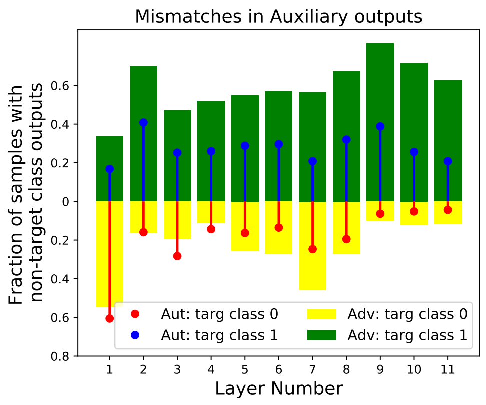

Figure 6: Fractions of authentic and adversarial inputs that generate a non-target class prediction at each layer-wise classification head.

(see Appendix D). These observations justify the features that we include in F_{lw}.

Based on the above analyses, we have demonstrated that all 3 features of IAS are informative for adversarial detection. Our results in the next section corroborate these findings.

5.3 Performance on Adversarial Detection

Following the observations in the previous section, we use AdvNet with the identified features for adversarial detection. We compare the performance of AdvNet with the current state-of-the-art approaches for detecting adversarial inputs for BERT-based models, *viz.*, **FGWS** (Mozes et al., 2021), **NWS** (Mozes et al., 2021), **DISP** (Zhou et al., 2019) and **FreeLB** (Zhu et al., 2019). We briefly describe these methods in Appendix C.

As seen in Table 1, AdvNet significantly outperforms existing approaches across all 10 datasets with an average improvement of 7.45%. We report an improvement of 6.53% for the 3 sentiment analysis datasets (SST-2, Yelp, IMDb), 8.05% for the 4 NLI datasets (RTE, SNLI, MNLI, QNLI) and 6.98% for the 2 paraphrase detection datasets (MRPC, QQP) over the respective best methods.

Another baseline that we compare with is **Certified Robustness Training** (Jia et al., 2019). While this work is not aimed at adversarial detection, it provides bounds on model robustness for word substitution perturbations. For making a comparison with our work, we note that the fraction of adversarial samples that are correctly detected as adversarial translates to robustness for binary classification tasks. We report robustness of 87% for word substitution-based attacks and 81% across all 11 attacks for IMDb, while the best upper bound obtained through certified robustness training is 75%.

When comparing across datasets, we observe that AdvNet performs better on simpler sentence labelling datasets like SST-2 and AG News 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. We hypothesize that this nature implies that the attention masks for different inputs even belonging to the same type (authentic or adversarial) can vary widely. This reduces the consistency of features across input types making the detection harder. Nevertheless, AdvNet establishes state-of-the-art results across datasets. A detailed analysis of the performance of AdvNet across tasks and attack types is provided in Appendix E. In Appendix F, we perform a defense transferability study to show how the model generalizes well with state-of-the-art performance on unseen attack types.

5.4 Ablation Studies

We now evaluate how variations in model size, training set size, and the choice of feature combinations effect performance of AdvNet.

Effect of model size. IAS can be computed for Transformer networks of any size. We compare BERT-Small and BERT-Base models in terms of performance of AdvNet as shown in Table 1. We observe that, across datasets, AdvNet performs better in detecting adversarial inputs fed to the larger BERT-Base model (108M parameters) as opposed to the smaller BERT-Small model (25M parameters). The increase in accuracy averaged across tasks is a significant 10.76%. We hypothesize that this is because models with more layers encode more information and allow for a better build-up of semantic information which means that individual heads play more discrete roles. This better performance for the larger model is encouraging as the more accurate and larger language models are expected to be more vulnerable to adversarial attacks.

Effect of training set size. In Figure 7, we show how the performance of AdvNet changes as the amount of training data changes. We observe that AdvNet performs well even when it uses only a fraction of the training set. Specifically, even at

Model	SST-2	Yelp	AG News	MRPC	IMDb	SNLI	RTE	MNLI	QQP	QNLI
FGWS	71.93	78.36	70.41	69.85	75.98	75.41	71.23	60.23	73.52	78.14
NWS	70.31	74.72	65.62	68.02	65.72	71.82	64.27	56.94	70.20	74.58
DISP	68.73	70.15	66.38	62.22	75.23	72.92	66.40	59.34	69.86	76.92
FreeLB	77.60	82.54	75.55	72.41	79.85	79.80	64.29	58.10	65.69	76.40
AdvNet										
w/ BERT-Small	78.57	76.72	78.63	75.05	74.09	72.07	73.64	64.26	68.71	74.47
w/ BERT-Base	**90.74**	**87.68**	**91.78**	**84.61**	**81.18**	**82.50**	**80.43**	**72.61**	**75.27**	**86.07**

Table 1: Comparison of the adversarial detection accuracy of AdvNet using features extracted from fine-tuned BERT-Small and BERT-Base models with other state-of-the-art approaches for adversarial detection.

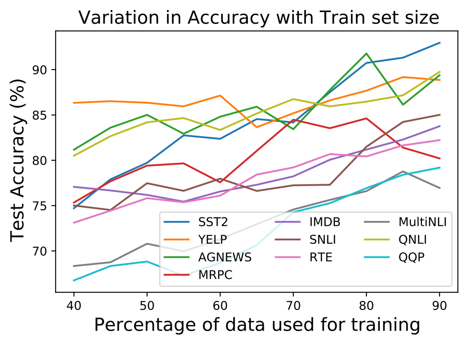

Figure 7: Effect of training set size on accuracy of adversarial detection with AdvNet.

40% of the training examples used, AdvNet outperforms the results obtained with existing state-of-the-art models on most tasks. This suggests that the CutMix data augmentation is effective and the AdvNet model is sample-efficient. This is particularly important because designing adversarial examples for each dataset remains a challenging task.

Datasets	F_{mask}	F_{flip}	F_{lw}	Bin	w/o CM
SST-2	82.87	74.07	64.79	85.59	82.23
Yelp	80.23	62.08	66.01	84.30	83.57
AG News	83.11	76.41	57.14	90.47	83.11
MRPC	76.35	68.82	59.40	80.27	77.35
IMDb	74.54	60.00	55.45	73.78	74.23
SNLI	80.83	57.91	58.83	75.64	70.41
RTE	74.44	60.88	56.67	77.21	74.06
MNLI	66.95	51.30	60.00	66.85	69.95
QQP	66.41	61.63	62.64	71.88	64.50
QNLI	79.65	55.69	59.36	81.42	73.11

Table 2: Results on feature combinations.

Using different feature combinations. We had shown that each of the three features are informative in Section 5.2. In Table 2, we report the performance of AdvNet by ablating various model components. The first 3 columns report accuracies when only one of the three features is passed at a time to the model. We observe that F_{mask} performs better than F_{flip} and F_{lw}. This suggests that the attention mask is the most important feature input to the model. We analyze the roles of individual gating values using GradCAM (see Appendix G). Next, we test the performance when the boolean attention mask F_{bmask} is used instead of the real-valued vector F_{mask} along with F_{flip} and F_{lw}. The lower accuracy indicates that the real values are more informative. Finally, we test the model performance when CutMix is not used and conclude that augmenting the training set using CutMix provides higher accuracy as seen in the last row of Table 1 which uses all 3 features along with CutMix.

In summary, our results show that (a) the 3 IAS features are individually informative, (b) AdvNet significantly improves on baseline methods across datasets, (c) AdvNet performance improves with model size and does not drop much on reducing training sets, and (d) the best performance is with all 3 features along with CutMix augmentation.

6 Conclusion and future work

In this work, we present an altogether new utility of attention heads in Transformer networks - to detect adversarial attacks. We defined input-specific attention subnetworks (IAS) and proposed a method to compute them efficiently. We extracted 3 features from IAS and showed their utility in distinguishing adversarial samples from authentic ones. We demonstrated that our approach significantly improves the state-of-the-art accuracy across datasets and attack types. Our work suggests that input-specific model perturbations provide strong signals to interpret Transformer-based models such as large language models. Further, the sparse nature of the identified IAS indicate opportunities for input-specific model optimization.

References

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Aakriti Budhraja, Madhura Pande, Preksha Nema, Pratyush Kumar, and Mitesh M. Khapra. 2020. 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)*, pages 3230–3235, Online. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Shi Feng, Eric Wallace, Alvin Grissom II, Mohit Iyyer, Pedro Rodriguez, and Jordan Boyd-Graber. 2018. Pathologies of neural models make interpretations difficult. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3719–3728, Brussels, Belgium. Association for Computational Linguistics.

Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. 2018. Black-box generation of adversarial text sequences to evade deep learning classifiers. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities. *arXiv preprint arXiv:1901.05287*.

John Hewitt and Christopher D. Manning. 2019. 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)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

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

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *EMNLP*.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2020. Is BERT really robust? A strong baseline for natural language attack on text classification and entailment. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8018–8025. AAAI Press.

Christos Louizos, Max Welling, and Diederik P Kingma. 2017. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

John X. Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp.

Maximilian Mozes, Pontus Stenetorp, Bennett Kleinberg, and Lewis Griffin. 2021. 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*, pages 171–186, Online. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Lina Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. *arXiv preprint arXiv:1603.00892*.

Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. 2019. When does label smoothing help? In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Madhura Pande, Aakriti Budhraja, Preksha Nema, Pratyush Kumar, and Mitesh M Khapra. 2021. The heads hypothesis: A unifying statistical approach towards understanding multi-headed attention in bert. *arXiv preprint arXiv:2101.09115*.

Danish Pruthi, Bhuwan Dhingra, and Zachary C Lipton. 2019. Combating adversarial misspellings with robust word recognition. *arXiv preprint arXiv:1905.11268*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1085–1097, Florence, Italy. Association for Computational Linguistics.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Ian Tenney, Dipanjan Das, and Ellie Pavlick. 2019. BERT rediscovers the classical NLP pipeline. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4593–4601, Florence, Italy. Association for Computational Linguistics.

Betty van Aken, Benjamin Winter, Alexander Löser, and Felix A. Gers. 2019. How does bert answer questions? a layer-wise analysis of transformer representations. CIKM '19, page 1823–1832, New York, NY, USA. Association for Computing Machinery.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. 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*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Jingyi Wang, Guoliang Dong, Jun Sun, Xinyu Wang, and Peixin Zhang. 2019. Adversarial sample detection for deep neural network through model mutation testing. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 1245–1256. IEEE.

Yulong Wang, Hang Su, Bo Zhang, and Xiaolin Hu. 2020. Interpret neural networks by extracting critical subnetworks. *IEEE Transactions on Image Processing*, 29:6707–6720.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. 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)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. 2019. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 501–509.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015a. Character-level Convolutional Networks for Text Classification. *arXiv:1509.01626 [cs]*.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015b. Character-level convolutional networks for text classification. In *NIPS*.

Yichao Zhou, Jyun-Yu Jiang, Kai-Wei Chang, and Wei Wang. 2019. Learning to discriminate perturbations for blocking adversarial attacks in text classification. 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)*, pages 4904–4913, Hong Kong, China. Association for Computational Linguistics.

Chen Zhu, Yu Cheng, Zhe Gan, S. Sun, Tom Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for language understanding. *ArXiv*, abs/1909.11764.

A Datasets used for authentic examples

The 10 datasets used in this work were listed in Section 4.1. Here, we provide additional details about these datasets. SST-2 (Socher et al., 2013), Yelp polarity (Zhang et al., 2015a) and IMDb (Maas et al., 2011) are binary sentiment classification datasets. AG News (Zhang et al., 2015b) consists of news headlines classified into one of 4 categories (world, sports, business, sci/tech) and MRPC (Dolan and Brockett, 2005) is a paraphrase dataset which contains sentence pairs with binary labels indicating whether they are semantically equivalent or not. RTE (Wang et al., 2018), MNLI (Williams et al., 2018), SNLI (Bowman et al., 2015) contain sentence pairs with labels indicating whether one sentences entails, contradicts or is neutral with respect to the other sentence. QQP is again a paraphrase dataset but unlike MRPC which contains sentences, it contains question pairs taken from Quora with binary labels indicating whether they are semantically equivalent or not. QNLI contains question-context pairs with a binary label indicating whether the context sentence contains the answer to the question or not.

B Examples of adversarial attacks

In Table 3, we provide examples for each of the 11 attack types that we use to generate adversarial inputs for this work.

C Other methods for Adversarial Detection

We briefly describe the four methods that we compare with in Table 1.

- **FGWS** (Mozes et al., 2021): Here, a word frequency-guided approach is used to identify infrequent words in an input sentence and replace them with more frequent, semantically similar words. Then, the difference in prediction confidence of the Transformer-based model between the original and substituted sentences is considered. If this value is above a threshold, the sentence is predicted to be adversarial.
- **NWS**: This is the *naive word substitution* baseline used in Mozes et al. (2021). Here, each out-of-vocabulary word in an input sentence is replaced with a random word from a set of semantically related words, following which the same process as above is used to predict input authenticity.

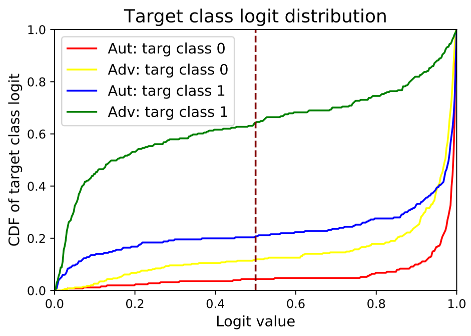

Figure 8: CDF over the target class output logit of the mutated subnetwork. The large area below the green curve with logit value$<$0.5 corresponds to a large number of adversarial inputs whose mutated subnetworks predict a non-target class.

- **DISP** (Zhou et al., 2019): In this approach, a BERT-based perturbation discriminator predicts whether each token in the input sentence is authentic or perturbed. If none of the tokens are predicted to be perturbed, the input sentence is considered authentic.
- **FreeLB** (Zhu et al., 2019): This is an adversarial training approach where adversarial perturbations are added to word embeddings and the resulting adversarial loss is minimized to promote higher invariance in the embedding space.
- **Certified Robustness Training** (Jia et al., 2019): This approach uses Interval Bound Propagation (IBP) to obtain an upper bound on the worst-case loss resulting from any word substitution-based perturbation. This has been applied to CNN and LSTM-based language models.

D Analysing F_{flip} and F_{lw}

In the second column of Table 4, for each of the datasets, we show the percentage of authentic and adversarial inputs which generated non-target class predictions. In Figure 8, we show the CDF over the target class output logit of the mutated subnetwork. Further, in the third column of Table 4 we show the percentage of (authentic, adversarial) inputs whose layer-wise outputs showed more than one switch. These results show that the F_{flip} and F_{lw} are individually informative.

E Adversarial detection accuracy for different attack types

In Table 5, we present the breakup of model accuracy across individual attack types. We observe

Attack Type	**Perturbed Text**
Original Text	**it 's a charming and often affecting journey.**
Word-level attacks	
Deletion	it's a _ and often affecting journey.
Antonyms	it's a repulsive and often affecting journey.
Synonyms	it's a charming and often affecting passage.
Embeddings	it's a charming and quite affecting journey.
Order Swap	it's charming and affecting a often journey.
PWWS	it's a entrance and often strike journey.
TextFooler	it's a charming and _ affecting journey.
Original Text	**a sometimes tedious film.**
Character-level attacks	
Substitution	a sometimes tidious fylm.
Deletion	a som_times tedio_s film.
Insertion	a sometimeDs tvedious film.
Order Swap	a smoetimes tedoius film.

Table 3: Examples of 11 attack types used for adversarial data creation. '_' represents a deleted character and there is no character present at that position in the adversarial sample.

	Non-target o/p	Switches>1
Dataset	**(Mutated)**	**(Layer-wise)**
SST-2	(12.3, 34.2)	(37.9, 54.8)
IMDb	(0.33, 2.18)	(0.16, 1.45)
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)
RTE	(24.5, 22.2)	(44.2, 50.9)
SNLI	(2.83, 96.0)	(11.6, 41.0)
MNLI	(11.0, 24.8)	(24.3, 42.5)
QQP	(3.2, 1.3)	(6.2, 6.8)
QNLI	(5.7, 1.0)	(13.8, 11.1)

Table 4: Percentages of (authentic, adversarial) inputs whose (a) mutated subnetworks generated non-target class predictions; (b) layer-wise outputs showed more than one switch.

that for text classification tasks like SST-2, Yelp and AG News the accuracy for Embedding and Synonym swap attack types are much higher compared to other datasets. We also note that in case of both word and character-level attacks, Deletion and Substitution operations are the ones with least detection accuracy across almost all datasets. Finally, we observe that the performance for detecting adversarial inputs generated by PWWS and TextFooler attacks remain fairly consistent across datasets.

F Defense Transferability Analysis

Here, we perform a defense transferability study to understand how well the model can perform on unseen attack types. For this purpose, we train AdvNet with samples from only $x\%$ of the 11 attack types and report results both on the remaining attack types and the complete test set for $x \in \{25, 50, 75\}$ in Table 6. We observe that even when AdvNet is trained with only 75% of the attack types, the test results on new attacks outperform existing approaches for most datasets, thus showing that our model can generalize to unseen attack methods. Besides, the test accuracies on the complete test set closely agree with those on the new attack types. This indicates that the reduction in accuracy can largely be attributed to a smaller training set than to a lack of defense transferability.

G 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, we consider neurons that correspond to the gating values, i.e, F_{mask} and call the heads corresponding to them as *refereeing* heads. From Figure 9, we observe that word swap attacks like antonyms, syn-

Dataset	#Adversarial	Word-level attacks							Character-level attacks			
	samples	DEL	ANT	SYN	EMBED	SWAP	PWWS	TextFooler	SUB	DEL	INS	SWAP
SST-2	739	0.84	0.96	0.95	0.96	0.75	0.81	0.76	0.92	0.80	0.87	0.89
Yelp	589	0.75	0.92	0.92	0.96	0.88	0.80	0.95	0.93	0.77	0.88	0.88
AG News	829	0.88	0.96	0.92	0.96	0.82	0.83	0.84	0.89	0.84	0.85	0.88
MRPC	712	0.75	0.75	0.9	0.72	0.94	0.84	0.82	0.86	0.79	0.76	0.92
IMDb	321	0.80	0.76	0.85	0.89	0.80	0.82	0.81	0.94	0.75	0.96	0.79
SNLI	1262	0.61	0.80	0.78	0.88	0.78	0.76	0.79	0.85	0.88	0.65	0.83
RTE	541	0.75	0.84	0.86	0.87	0.79	0.77	0.73	0.82	0.76	0.82	0.82
MNLI	548	0.67	0.80	0.72	0.85	0.78	0.80	0.76	0.78	0.80	0.86	0.76
QQP	307	0.70	0.82	0.74	0.80	0.75	0.76	0.74	0.78	0.81	0.86	0.77
QNLI	395	0.80	0.90	0.92	0.92	0.90	0.82	0.86	0.82	0.86	0.82	0.82

Table 5: 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 for descriptions of attack types. The second column provides the number of adversarial samples generated by us for each task across all 11 attack types.

Dataset	25%	50%	75%
SST-2	(57.8, 58.9)	(69.9, 68.4)	(82.7, 80.7)
Yelp	(63.1, 61.8)	(70.3, 69.9)	(77.8, 78.4)
AG News	(63.7, 62.1)	(71.4, 69.9)	(83.6, 78.4)
MRPC	(63.2, 60.1)	(73.2, 74.5)	(81.5, 82.3)
IMDb	(66.8, 64.8)	(71.6, 73.1)	(77.4, 79.1)
SNLI	(57.9, 57.6)	(67.2, 66.6)	(73.4, 72.3)
RTE	(63.8, 62.4)	(70.8, 69.7)	(76.4, 75.5)
MNLI	(57.4, 58.8)	(62.3, 61.3)	(67.0, 68.9)
QQP	(59.7, 60.2)	(64.0, 64.2)	(69.0, 69.6)
QNLI	(61.8, 60.6)	(69.3, 67.2)	(75.7, 77.5)

Table 6: Defense transferability study of AdvNet with varying percentages of attack types included in the train set. Each tuple contains the test accuracy on new attack types and on all attack types respectively.

onyms, 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 a unknown token absent in the vocabulary. Since this changes the input embedding sequence more dramatically, small deviations from standard gating patterns are sufficient to mislead the model leading to fewer refereeing heads. Since introducing synonym and embedding based perturbations change the embeddings input 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.

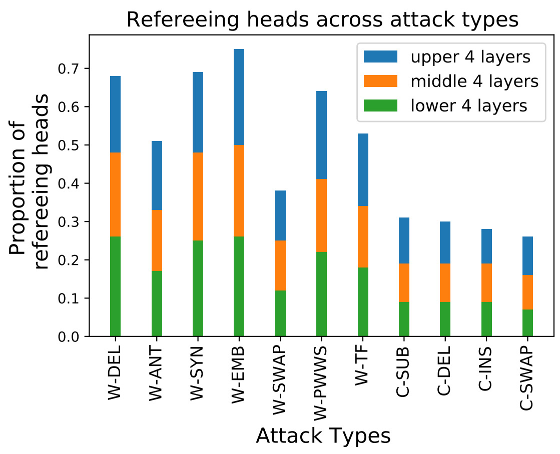

Figure 9: Fraction of refereeing heads used by the adversarial detection model across various adversarial attack types. The split of these across 4 layer subsets is also shown.