# Sample-specific Attention Masks for Transformer Models

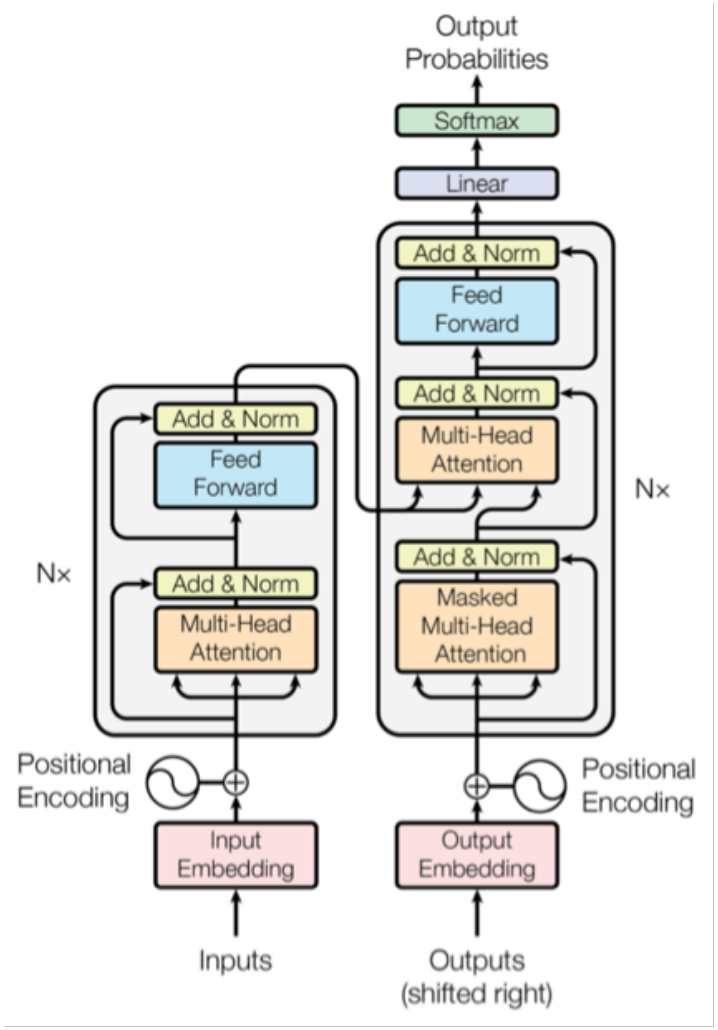
Emil Biju

August 21, 2022

# Introduction

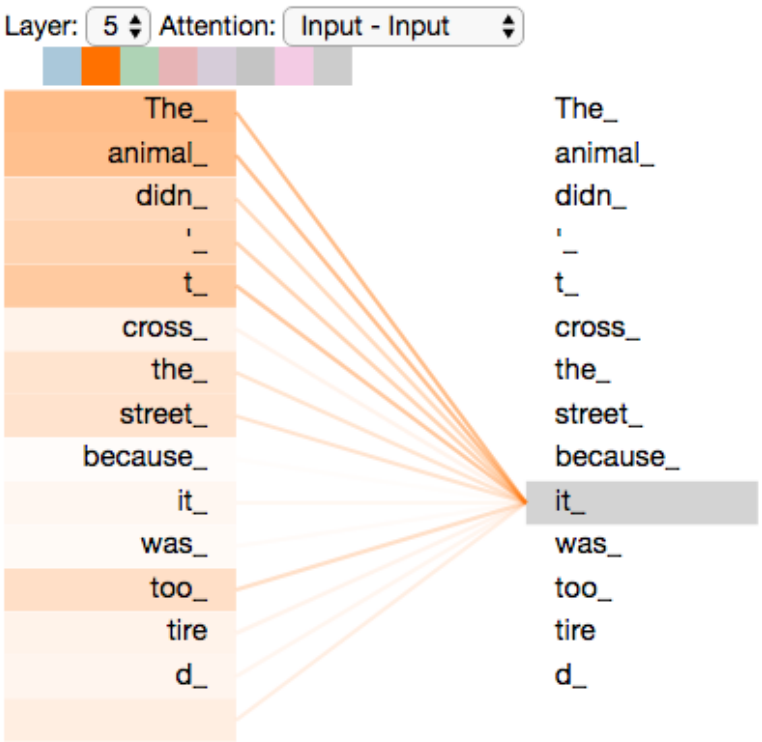
**The Transformer**

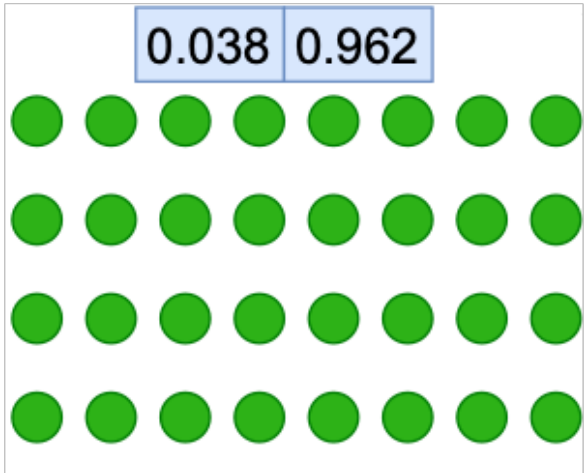
- Novel Neural network architecture for processing sequence-based inputs
- Achieved state-of-the-art accuracy in various sequence learning tasks, particularly in NLP
- Transformer-based models like BERT have gained prominence in Machine Translation and other NLU tasks.



# Background: Transformer Architecture

- Consider a Transformer model with $n$ layers and $m$ heads per layer
- Each head computes self-attention on the input sentence. Self-attention relates different parts of a sentence and defines relationships across words/phrases.
- $\text{Layer}_j(X_j) = \text{concat}_i[\text{Head}_{ji}(X_j)]W_j^O$



# Recent Advancements

## Interpretability

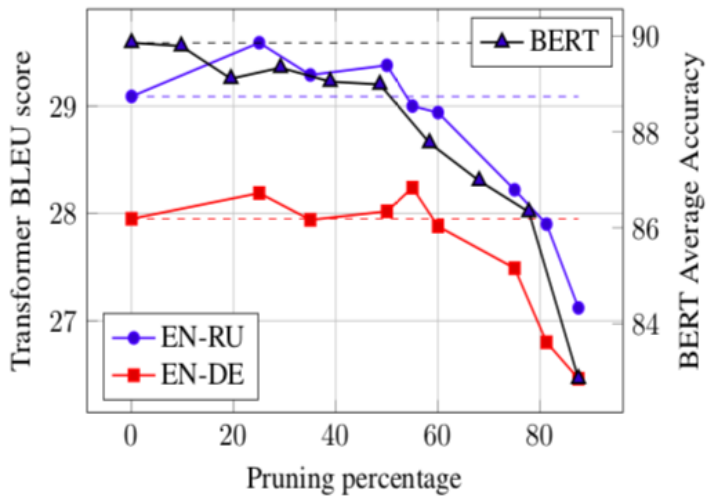
- Roles of self-attention heads (Voita et al. [2019], Michel et al. [2019])
- Roles of Transformer layers (Jawahar et al. [2019], van et al. [2019])

## Efficiency

- Recent works have shown that several heads from Transformer networks can be pruned with little impact on model performance

## Adversarial Robustness

- Adversarial training for model robustness - requires a large number of training samples
- Slow & compute-intensive (Chen et al. [2020])



# Setup for pruning self-attention heads

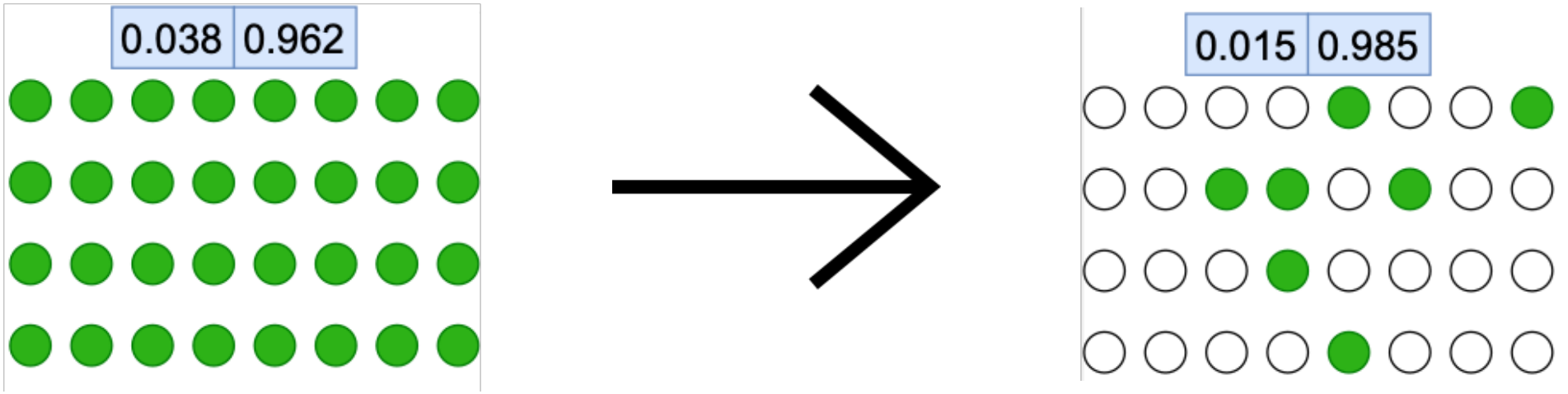
How to prune self-attention heads?

- We weigh the output of each head by a scalar gating value $g_{ji} \in \{0, 1\}$.

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

Define a pruning mask vector

- Flatten the gating values for all heads in all layers of the network to obtain the vector $g(x)$ and the induced subnetwork is $\mathcal{S}(g(x))$.



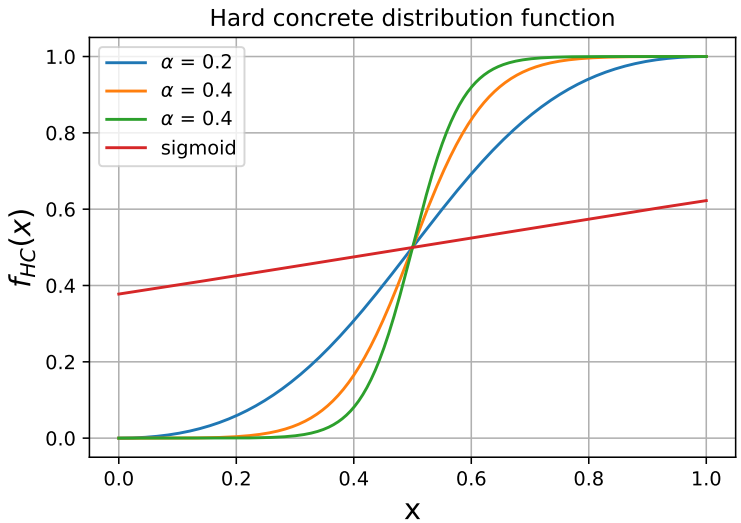
# Generating sample-specific pruning masks

- First, the standard network parameters are fine-tuned for the task considered $\implies \theta^*$
- Each gating value is represented by $g_{ji} = f_{HC}(p_{ji})$, $f_{HC}$ is the hard-concrete distribution

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

- Freeze standard parameters to $\theta^*$, $p_{ji} \leftarrow 0.5$ and train parameters in $g(x)$ with the same objective

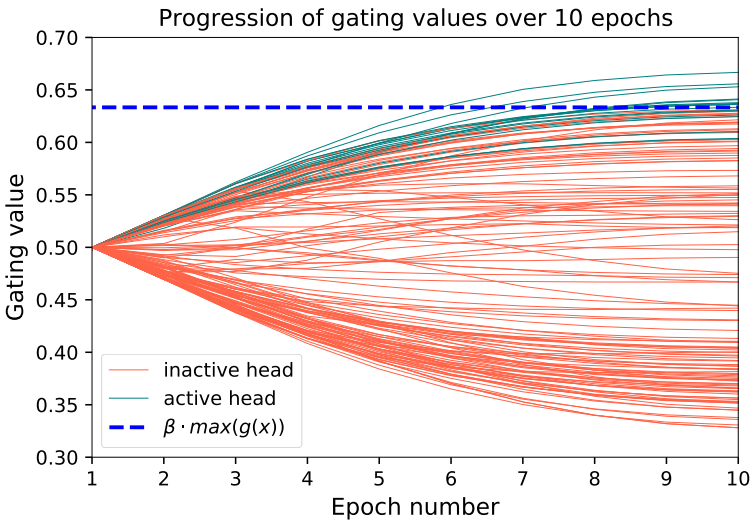
$$\mathcal{L}_x^g = \mathcal{L}_{CE}(f(x, \theta, g(x)); y | \theta = \theta^*)$$

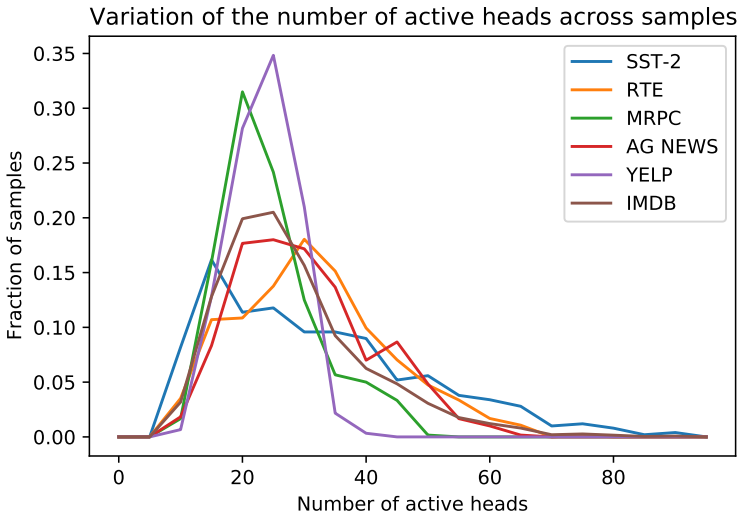


# Generating sample-specific pruning masks

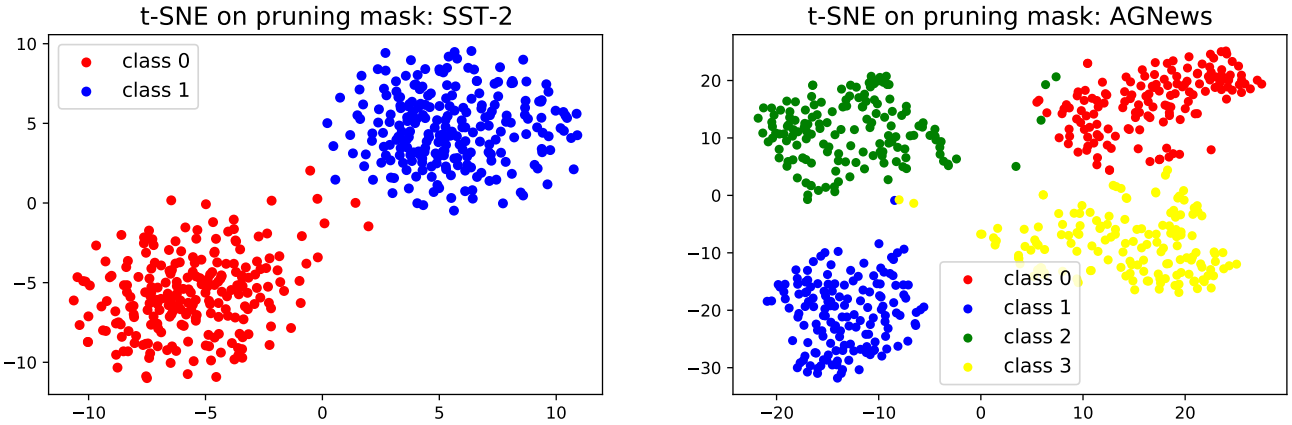
Define a Boolean pruning mask vector $g^b(x)$

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



# t-SNE plot of Pruning mask vector




- Distinct clusters for pruning mask vectors of separate classes
- Moderate separation between authentic and adversarial samples with the same target class

## Feature $\mathcal{F}_1(x)$

The real-valued pruning mask vector $g(x)$ after training

# Mutating the pruning mask vector

## Motivation

- Adversarial samples are heavily reliant on network architecture and specific parameter combinations to fool the network
- Less likely that a mutated subnetwork would generate the same prediction for an adversarial sample.

| **Layer subset** | **Function of heads** |
|---|---|
| Initial layers | Surface/phrase-level understanding |
| Middle layers | Capture syntactic relations<br>Perform multi-task processing |
| Final layers | Highly task-specific |

## Idea

- Mutate the pruning mask vector by inverting the gating values in the middle layers.

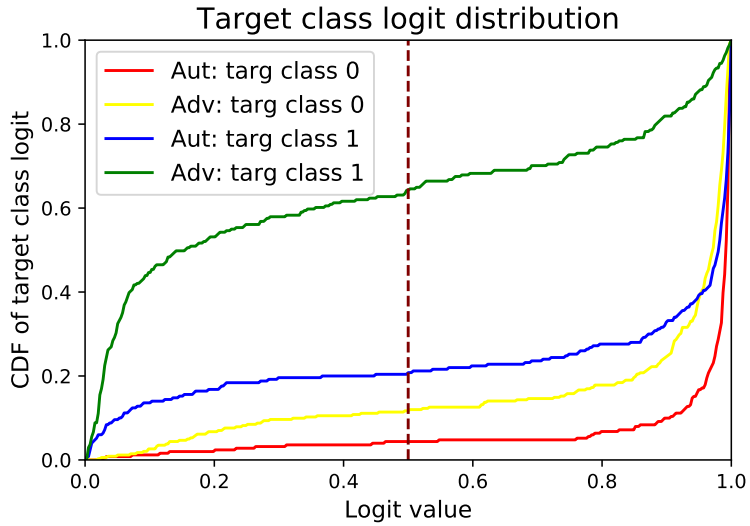
# Mutating the pruning mask vector

**Procedure**

- Given $g^b(x)$, the boolean gating values corresponding to the middle $\lceil \frac{n}{3} \rceil$ layers are flipped to obtain $g^c(x)$.
- Input $x$ is passed through mutated subnetwork $\mathcal{S}(g^c(x))$

**Observations**

- **Authentic:** Low impact (target class predicted with high conf)
- **Adversarial:** High impact (non-targets predicted with high conf)




## Feature $\mathcal{F}_2(x)$

- target class, predicted class, confidence of prediction
- boolean flag: target class == predicted class

# Layer-wise auxiliary outputs

**Motivation**

- Authentic samples are more stable under mask mutation $\implies$ Representations at each layer of the mutated subnetwork do not change much
- Adversarial samples are less stable $\implies$ Some intermediate representations are very different, hence false information is propagated.

**Idea**

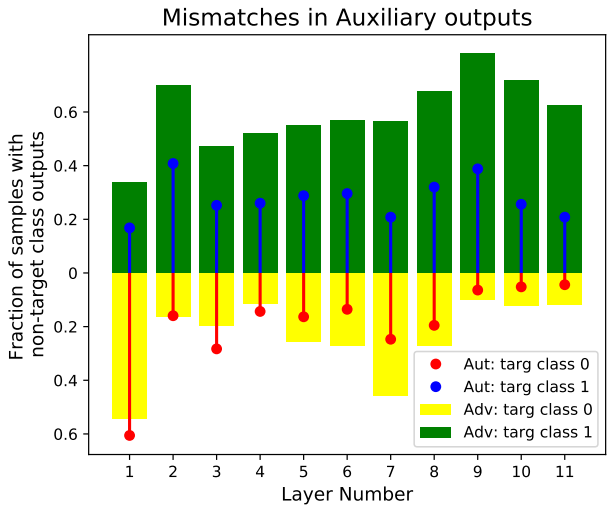
- What information does each layer carry?
- Predict the output class from the representation generated by every layer of the mutated subnetwork

# Layer-wise auxiliary outputs

- Given the complete fine-tuned network, introduce classification layers after every layer of heads (except the final layer).
- Freeze all standard parameters and train each classification layer to maximize the target class probability.
- Prune the network using $g^c(x)$ and obtain layer-wise outputs.

## Observations

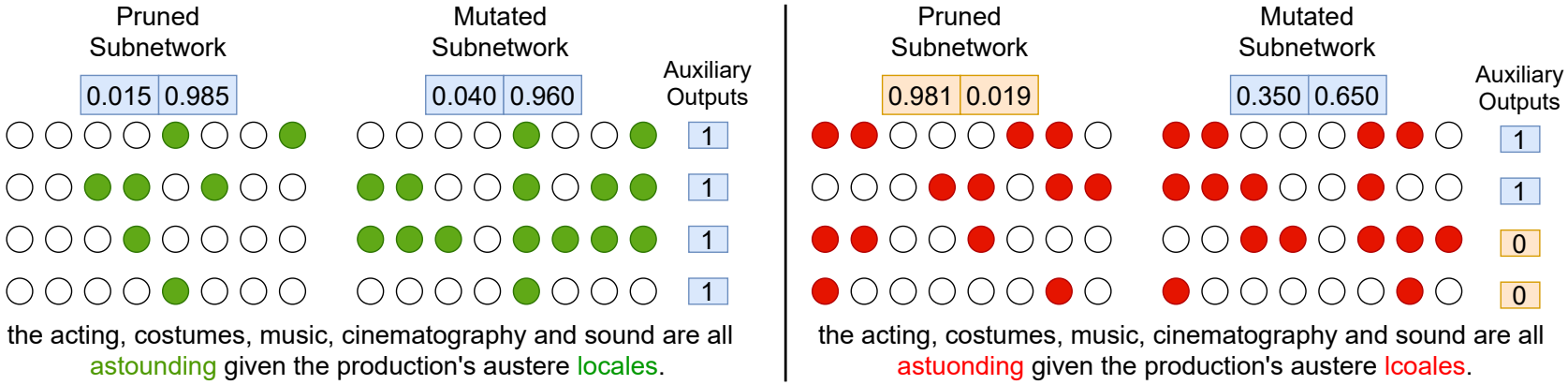
| Authentic | Adversarial |
|---|---|
| Majority of outputs belong to target class | Several non-target class outputs generated |
| Stable outputs after first few layers | Outputs keep switching between classes |




## Feature $\mathcal{F}_3(x)$

$n-1$ layer-wise outputs; number of auxiliary layer outputs that match the target class; number of switches in the layer-wise predictions

# AdvNet: Adversarial Detection Classifier




- Input: $\mathcal{F}_1 \cup \mathcal{F}_2 \cup \mathcal{F}_3$
- Architecture: Two 1-D Convolutional layers (ReLu), two FC layers (sigmoid), output classification layer (sigmoid)
- Output: Binary label (0-authentic, 1-adversarial)

**Data Augmentation using CutMix**

- Creating a large number of adversarial samples is time/compute-consuming
- Extend CutMix to 1D: Cut and paste patches from multiple feature vectors and mix ground truths proportionally

# Experimental setup

**8 Datasets**

- SST-2, Yelp, IMDb - sentiment
- AG News - news category
- MRPC - sentence equivalence
- RTE, SNLI, MultiNLI - textual entailment

**11 attacks**

- Word-level attacks
  - deletion, antonyms, synonyms, embeddings, order swap, BERT-LM, CLARE
- Character-level attacks
  - substitution, deletion, insertion, order swap

# Adversarial detection results

| 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 |

- Performance on BERT-Base is better than BERT-Small
- Performance on simpler tasks with shorter inputs (SST-2, Yelp) is better
- Precision, recall and accuracy values are in the same proximity

# Comparison with other SoTA methods

| Dataset | FGWS | NWS | DISP | AdvNet |
|---|---|---|---|---|
| **SST-2** | 71.93 | 70.31 | 68.73 | **90.74** |
| **Yelp** | 78.36 | 74.72 | 70.15 | **87.68** |
| **AG News** | 70.41 | 65.62 | 66.38 | **91.68** |
| **MRPC** | 69.85 | 68.02 | 62.22 | **84.61** |
| **IMDb** | 75.98 | 65.72 | 75.23 | **81.18** |
| **SNLI** | 75.41 | 71.82 | 72.92 | **82.50** |
| **RTE** | 71.23 | 64.27 | 66.40 | **80.43** |
| **MultiNLI** | 60.23 | 56.94 | 59.34 | **72.61** |
| **QQP** | 73.52 | 70.20 | 69.86 | **75.27** |
| **QNLI** | 78.14 | 74.58 | 76.92 | **86.07** |

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

# Ablation Study

| 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 |
| $\mathcal{F}_1, \mathcal{F}_2, \mathcal{F}_3$ | **90.74** | **87.68** | **84.61** | **80.43** | **81.18** | **82.50** | **72.61** |

Table: Table for ablation study of AdvNet

- $\mathcal{F}_1$ is the most crucial feature
- In cases where $Perf(\mathcal{F}_2) > Perf(\mathcal{F}_3)$, $Perf(\mathcal{F}_1 \cup \mathcal{F}_3) > Perf(\mathcal{F}_1 \cup \mathcal{F}_2)$
- Using real-valued pruning mask is better than Boolean vector
- Data augmentation considerably improves performance

# Digging deeper: Why model behaviour on authentic and adversarial samples vary?

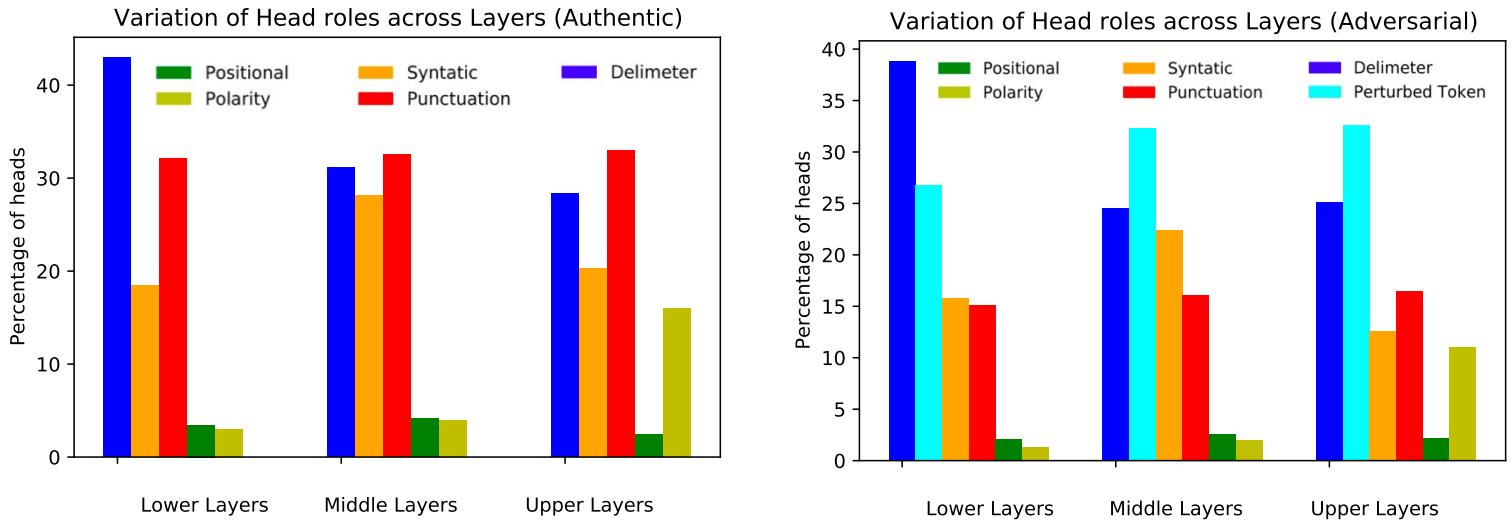



Figure: Roles played by important heads across encoder layers

- Very high number of heads attend to perturbed portions of the sentence in adversarial samples
- This causes changes in standard gating patterns $\implies$ inferred from $\mathcal{F}_1$
- Many of these changes are negated on mutating the pruning mask $\implies$ affects mutated results $(\mathcal{F}_2, \mathcal{F}_3)$

# Conclusion

**Other Work**

- Comparison with other SoTA methods
- Attack-type based analysis
- Critical heads for adversarial detection

**Further Scope**

- Extension to other BERT models, to vision/speech tasks, beyond classification tasks

Thank You