Unitary BERT for Robust Natural Language Processing

Anonymous ACL submission

Abstract

Recent developments in adversarial attacks on deep learning leave many mission-critical natural language processing systems at risk of exploitation. Here we report a novel BERT-based neural architecture featuring unitary weights and support vector machines that drastically improves the robustness against a wide range of adversarial attacks in natural language processing. Our model, unitary BERT (UniBERT), boosts post-attack classification accuracies by up to 67.5% while maintaining competitive pre-attack accuracies. The accuracy and robustness tradeoff in our model can be adjusted by a single scalar parameter to best fit the design requirements of the target applications.

1 Introduction

1.1 BERT

Bidirectional Encoder Representations from Transformers (BERT) is a popular neural network model for natural language processing (NLP) (Devlin et al., 2019) with many advantages over its predecessors such as the convolutional neural networks (CNN) and the recurrent neural networks (RNN) (Yin et al., 2017). Training for BERT is separated into two phrases: pretraining and finetuning. During the pretraining phase, the network is trained as a masked language model, in which words are randomly masked out and the network is asked to predict the missing words (Salazar et al., 2020). After pretraining, BERT learns the basic mechanics of the language such as grammar and semantics. During finetuning, BERT is trained with task-specific datasets, in which the model is specialized to predict the correct labels. The finetuning phase is much shorter than the pretraining phase. This unique property, along with many other benefits, makes BERT a popular NLP model. A well pre-trained BERT can solve many different downstream tasks with little additional finetuning.

BERT, as well as many other NLP models, encodes word semantics with high-dimensional vectors (Miaschi and Dell'Orletta, 2020). BERT tokenizes each sentence into words and converts them to vectors via a set of word embeddings. If trained correctly, BERT will learn word embeddings that encode meaningful relationships and group similar concepts together in the vector space.

1.2 Adversarial Attacks in NLP

We define adversarial attacks in neural networks as the following: for an unnoticeable, small perturbation at the input, the output classification is altered. In computer vision, a small Gaussian noise is added to an image to trick the network into believing that it belongs to a different class (Akhtar and Mian, 2018). Neural networks are particularly sensitive to small noises because they make predictions by drawing polytope decision boundaries in the high dimensional space (Montufar et al., 2014). Deep neural networks are prone to create extraneous decision regions where no training or test data resides (Karimi et al., 2020). Adversarial attacks aim to create enough perturbation without appearing suspicious to humans but silently move the neural response across the decision boundary (Wong and Kolter, 2018).

Unlike computer vision where signals are continuous variables, it is more difficult to generate adversarial samples in NLP because languages are composed of discrete symbols such as phonemes and words. Despite this challenge, researchers have discovered ways for adversarial attacks: a small perturbation can be implemented as a synonym swap, resulting in a sentence with the same meaning (Liang et al., 2018; Jin et al., 2020; Ren et al., 2019). These small perturbations can lead to misclassification, and we refer to this type of attack as synonym-based attacks. Another type of adversarial attack is by introducing typographical errors in the sentences (Xie et al., 2020; Li et al., 2019). Although these injected typos appear benevolent to the readers, they can manipulate the computer systems to produce the wrong results.

1.3 Defense Against Adversarial Attacks

Defense against adversarial attacks can be grouped into two categories: The first is training with additional adversarial data. Like data augmentation, we deliberately generate adversarial samples and use them to train the network with correct labels to increase the model's robustness. Drawbacks of data augmentation-based defense include: it does not eliminate deep networks' tendencies in creating extraneous decision boundaries, and it requires us to anticipate the attack methods used by the adversaries to create appropriate coverage in the sample space. Furthermore, training would take much longer if we desire full coverage for all types of adversarial attacks. As an example in this line of work, linear interpolation is used to generate additional training samples (Si et al., 2021). The second type of defense against adversarial attacks is by adding a regularization to reduce model complexity thus mitigating overfitting. Disadvantages of regularization-based defense include the high computational cost and lengthy hypermeter tuning to find the right model complexity for each task. For example, Mixup Regularization adds an extra loss function to ensure consistent labeling in the neighborhood of the existing data samples, increasing robustness across the various level of adversarial attacks (Zhao et al., 2021).

1.4 Our Contributions

Here we present multiple innovations that significantly improve BERT's robustness measured in post-attack accuracies across various tasks and attack methods, including:

- Unitary constraints on the weight matrices to ensure small perturbations do not amplify to large differences.
- Support vector machine (SVM) loss function to maximize the inter-class decision margins.
- Our proposed model is attack-agnostic, simple to implement, and distinct from all existing approaches.

These key improvements are supported by fundamental theories and verifiable by probing the internal neural activations in experiments. We use the term "activations" to refer to the numerical output values of a neural layer.

2 Theory

2.1 Unitary Weights

We discovered that unitary constraints can be used to maintain the vector distance between neural activations. To understand this special property, we first need to define the concept of unitarity (Gilmore, 2008). A unitary matrix (**U**) is an n-by-n square matrix that satisfies this special relationship:

$$\boldsymbol{U^T U = U U^T = I} \tag{1}$$

The entries of a unitary matrix can be real or complex. When the matrices are real, they are called orthogonal matrices, a subclass of unitary matrices. Although the weight matrices in BERT are all real numbers, our theory detailed below works in both complex and real cases. Furthermore, the term "orthogonality" can be applied to both vector and matrices, and it's unclear whether or not we mean orthogonality across different layers. In contrast, "unitarity" is strictly associated with matrices satisfying Equation 1. Therefore, we name our innovation unitary BERT (UniBERT) as supposed to orthogonal BERT for conceptual clarity.

Neural networks can be viewed as transformations on the input vectors, where each layer's activations are considered as one vector. This is particularly apparent in linear layers, where the output of the layer is the matrix multiplication between the weight and the input. If there is no constraint on the weight matrix, the input vector can be scaled arbitrarily. However, if the weights are constrained to unitary matrices, the dot product between the unit vectors is preserved at the output. Hence, a small perturbation leads to a small perturbation at the output. We express this concept mathematically using the following equation:

$$(\boldsymbol{U}\bar{x} - \boldsymbol{U}\bar{y})^T(\boldsymbol{U}\bar{x} - \boldsymbol{U}\bar{y}) = (\bar{x} - \bar{y})^T(\bar{x} - \bar{y}) \tag{2}$$

, where x is the original input vector and y is the same vector with a small perturbation. The left-hand side of the equation calculates the dot product between the two vectors after the unitary transformation. It is common in NLP to represent semantics as high dimensional unit vectors and measure the semantic differences in the cosine similarity between them. Precisely, cosine similarity is defined as the vector dot product divided by the product of their Euclidean norms:

$$\cos\theta = \frac{\bar{x}\cdot\bar{y}}{|\bar{x}||\bar{y}|} \quad (3)$$

We see that in the case of unit vectors, the angle between them is directly related to their dot product. Therefore, as shown in Equation 1, unitary matrices maintenance the angle between the original and the perturbed vector. This means that if the perturbation is small (ex. a synonym swap), the change in the final classifier's output will also be small.

2.2 SVM Loss

We found that switching to the Multilabel Margin Loss will dramatically improve the robustness of BERT. Support vector machines (SVM) are used to find the largest decision hyperplane that separates data points from different classes. Hard SVMs find the decision boundary that creates the largest margin for separating classes of training samples while soft SVMs allow some samples to fall inside the margin (Bishop, 2006). In machine learning, these data points are obtained through feature engineering and are not movable. On the contrary, in deep learning, these features can be modified during training by backpropagating the error to the weights in the previous layers. As a result, we can use the SVM loss to force a large margin, and the neural response for different classes of inputs will be more distinct at the last layer. This technique is proven to be highly effective in computer vision (Elsayed et al., 2018; Liu et al., 2017; Romero et al., 2004; Wang et al., 2018).

We applied this concept to BERT-based NLP. Traditionally, BERT uses the Cross-Entropy Loss during finetuning. But in UniBERT, we will replace it with the Multilabel Margin Loss (SVM loss) instead. It is defined as follows for a mini-batch of data:

$$SVM\ loss = \frac{\sum_{i=0}^{n}\sum_{j=0}^{c}\max(0, margin - \boldsymbol{X}[i,y_i] + \boldsymbol{X}[i,j])}{n} \quad (4)$$

, where n is the batch size, c is the number of classes in the classification task, the margin is the desired SVM margin for the decision boundary, **X** is the activations of the last neural layer $\in\mathbb{R}^{n\times c}$ with [i,j] selecting the matrix element at the i^{th} row and j^{th} column, and y_i is the ground truth integer class label $\in\{0, \ldots, c\text{-}1\}$ for the i^{th} sample in the mini-batch.

2.3 Our Proposed Model

Our proposed model, UniBERT, improves BERT by using the unitary weights (Section 2.1) and the SVM loss function (Section 2.2). Detailed in Table 1 below, the weights that have unitary constraints are the query, key, value, and dense1 in each of the 12 BertLayer. We use the same loss function as BERT during pretraining but change it to the SVM loss from Equation 4 during finetuning. This loss function is called the MultiMarginLoss[1] in the Pytorch library. Though our current implementation has the same number of parameters (110M) as the BERT base, researchers have shown that it is possible to further compress unitary weights to half the amount of parameters (Chang and Wang, 2021). Our source code for this work is accessible online for reproducibility[2].

3 Experiments

3.1 Datasets

Bookcorpus is an unlabeled dataset containing 74 million sentences from 11 thousand books (Zhu et al., 2015). For UniBERT pretraining, we separate this dataset into train and test splits with a 95 to 5 ratio. For finetuning, we selected three different datasets for a comprehensive evaluation covering multilabel categorization, language inference, and sentiment analysis: First, **ag_news** is a dataset for news classification, and the goal is to classify the articles into four categories: world news, business news, science & technology, or sports news (Zhang et al., 2015). Second, **snli** or the Stanford Natural Language Inference corpus aims to train systems that can identify the relationship between a pair of sentences (Bowman et al., 2015). There are three possible classifications: entailment, contradiction, or neutral. Lastly, **yelp_polarity** is a text sentiment analysis dataset that was constructed by collecting reviews from Yelp.com (Zhang et al., 2015). The label is either positive or negative. Table 2 highlights the dataset statistics, and all datasets

[1]https://pytorch.org/docs/stable/generated/torch.nn.MultiMarginLoss.html

[2] Our anonymized source code can be downloaded from https://github.com/anonymous1335/code

Section	Name	Module (Dimension)	Weight Type
BertEmbed	word	Embedding (30522x768)	Regular
	position	Embedding (512x768)	Regular
	tok_type	Embedding (2x768)	Regular
	norm	LayerNorm	-
	dropout	Dropout	-
BertLayer (x12)	**query**	**Linear (768x768)**	**Unitary**
	key	**Linear (768x768)**	**Unitary**
	value	**Linear (768x768)**	**Unitary**
	dropout1	Dropout	-
	dense1	**Linear (768x768)**	**Unitary**
	norm1	LayerNorm	-
	dropout2	Dropout	-
	dense2	Linear (768x3072)	Regular
	dense3	Linear (3072x768)	Regular
	norm2	LayerNorm	-
	dropout3	Dropout	-
BertPooler	dense	Linear (768x768)	Regular
	activation	Tanh	-
Classifier	dropout	Dropout	-
	Linear	Linear (768xc)	Regular
Loss	**svm**	**MultiMargin**	-

Table 1: UniBERT Architecture. We impose unitary constraint on majority of the linear layers and use the MultiMarginLoss during finetuning. The number of classes in the classification task determines the final classifier's output dimension, c.

Dataset	Label	Train	Test	Length
bookcorpus	None	70M	4M	13
ag_news	4	120k	7.6k	39±11
snli	3	550k	10k	20±7
yelp_polarity	2	560k	38k	136±126

Table 2: Dataset statistics. Train and test are the number of sentences in each split. Length reports the average number of words in a sample.

used in our study follow a uniform distribution in the labels. Sentences are forced to all lower case if the dataset is cased, and all datasets are in English. Datasets and baseline models can be downloaded from the Hugging Face repository[3].

3.2 Training

Pre-training: We use bookcorpus with a fixed language mask with a masking probability of 0.15. We pre-train the UniBERT model for five epochs with a learning rate of 1e-4 from random initialization. The training schedule is linear with a warmup ratio of 0.01. Adam optimizer is used with beta1, beta2, and weight decay parameters set to 0.9, 0.999, and 0.01, respectively. We used the largest possible batch size (16) that fit into our graphic memory. With five epochs and a batch size of 16, the full pretraining phase took 700k steps. We used a sequence length of 512 during pretraining. At each step, the unitary weights are first updated using regular gradient descent and then converted to the closet unitary using the QR factorization technique. We only performed one pre-training run for a fair comparison with BERT and RoBERTa, whose published model parameters were also from a single run.

Finetuning: We finetune the pre-trained models for three classification datasets: ag_news, snli, and yelp_polarity. For UniBERT, QR factorization is again used to ensure unitarity on relevant weights. We conduct four independent finetuning runs; each with a new random initialization on the classifier. For each dataset, we finetune for five epochs with a learning rate of 5e-5. The batch size is 160 with a sequence length of 128. Other parameters are left as default in the TextAttack framework[4], which we used for finetuning and attack.

3.3 Unitary Constraints

QR factorization is a method to decompose any matrix into unitary and non-unitary parts, and we use it to find the closest unitary matrix in terms of the element-wise Euclidian distance (Golub and Van Loan, 2013). It is formulated as the following:

$$\boldsymbol{W} = \boldsymbol{QR} \tag{5}$$

, where W is a non-unitary square matrix, Q is a unitary matrix, and R is an upper triangular matrix. In the last step of the contraction, we multiply the unitary matrix with the diagonal elements from **R**:

[3]https://huggingface.co/

[4]https://github.com/QData/TextAttack

$$\overline{r} = diag(\boldsymbol{R}) \tag{6}$$

$$\boldsymbol{U} = \boldsymbol{Q}\overline{r} \tag{7}$$

, where diag(**R**) denotes a column vector constructed from the diagonal elements in **R**. **U** is the unitary matrix that we will use as the weight in selected linear layers, and we name it the unitary weight. For weights that we impose unitary constraints, we use QR factorization after each training iteration to find the closest contraction. It is called contraction because geometrically speaking, it is projecting the matrix onto an n-sphere.

3.4 Attack Recipes

We select three different types of attack methods for a comprehensive evaluation of our proposed model in defense against both typo-based and synonym-based NLP attacks. The first method, **Textbugger**, randomly introduces character insertion, deletion, swap, and substitution to flip BERT's prediction (Li et al., 2019). Secondly, **Textfooler** finds candidate adversarial samples by swapping important words with their synonym; synonyms are found by searching through the neighborhood in the word embedding space using the counter-fitted word embedding (Jin et al., 2020; Mrkšić et al., 2016). Textbugger and Textfooler may not preserve semantic proximity; therefore, we need to check for semantic similarity using the Universal Sentence Encoder (Cer et al., 2018). The similarity is measured by the cosine distance between the original and the perturbed sentences; we reject any adversarial samples that exceed a predetermined threshold. We use the identical threshold settings from the TextAttack framework to balance the quality and quantity of adversarial examples (Morris et al., 2020).

The last attack recipe we added to our evaluation portfolio is **PWWS**, which stands for Probability Weighted Word Saliency (Ren et al., 2019). It is based on WordNet synonym swap. Because WordNet is a human-labeled database, the adversarial examples that it generates have higher quality; hence, we do not perform additional safeguarding on the generated samples. For all attack recipes, we randomly select 1000 data samples from the test split to attack the finetuned models. The attack procedure is repeated four times to cover all four finetune runs.

3.5 Hyperparameters

Most hyperparameters (i.e., learning rate & schedule, dropout rate, mask rate, attack recipes) are identified from the best-known methods in the literature for BERT training. Others, such as the number of pretraining epochs for UniBERT, are selected to match the performance (e.g., mask language model loss) of the baseline models. Nevertheless, in UniBERT, there is a new hyperparameter: the SVM margin. We find the best setting by sweeping the margin parameter in Equation 4 with uniform sampling (logarithmically) from 1e-5 to 1e5. The tradeoff between adversarial robustness and accuracy is shown in Figure 1. As the margin increase, post-attack accuracy improves but slightly reduces the pre-attack accuracy. We prefer the pre-attack accuracy degradation to be <5% in our study and set the SVM margin to 100 for balanced performance in both pre-attack and post-attack accuracies. Depending on the end users' tolerance for pre-attack accuracy drop in their specific applications, they can select an appropriate SVM margin to maximize post-attack accuracy.

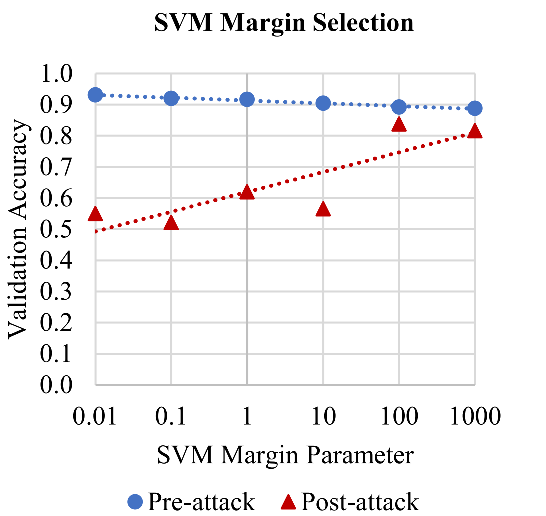

Figure 1: Hyperparameter selection for the SVM margin. This data is captured using a two-layer UniBERT model with the yelp_polarity dataset.

3.6 Computing Infrastructure

We run our simulations on a single NVIDIA RTX 3090 GPU with 24GB graphics memory. It takes four days to complete pretraining. For finetuning, training time depends on the specific tasks. The longest finetuning is yelp_polarity which takes 3.5

hours to complete. Attack speed depends largely on the recipes, and it takes about 3-100 minutes to complete. All reported runtimes are for one repetition. We repeat finetuning and attack four times with different seeds to capture the statistical variance.

4 Results & Discussion

4.1 System Response under Small Input Perturbation

The key benefit of unitary weights is that they keep small perturbation small throughout the network, as shown in Section 2.1. To validate this claim, we compare the neural activations between the original and the perturbed sentences using cosine similarity as they propagate across the network in Figure 2. We randomly select 100 sentences from the ag_news dataset, replace 10% of the words with their WordNet synonyms, and measure the cosine similarity between the original and perturbed sentences' activations, and plot the average similarity at each layer's output. The error bar denotes standard deviation. We observe that with the same perturbation, UniBERT has a neural response that is closer to the original sentence, evident by both the first layer's higher similarity score and its ability to maintain the similarity throughout the network. This result demonstrates that, under small perturbations at the input, UniBERT can maintain a closer internal representation to the original sentences than BERT.

4.2 SVM Loss for Class Separation

To visualize how SVM widens the classification margin, we plot the last layer's activations. In both BERT and UniBERT, the last layer is a linear classifier, projecting a 768-dimensional feature vector down to the number of classes. Using binary classification as an example, the network will compare the activations of the two neurons in the last layer and select the one with a higher activation as the predicted label. Figure 3 plots the activations of the two neurons in BERT's final layer for a binary classification task using the yelp_polarity dataset. We randomly sampled 64 sentences and observe the difference in their pre-attack and post-attack activations. The original sentences are shown as circles and their respective adversarial sample denoted using an x with a thin, solid line connecting the two points. For an attack to be successful, the final activations must move across

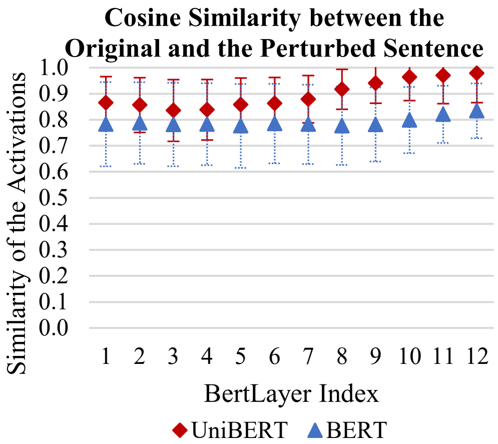

Figure 4: Cosine similarity of the neural activations across the network under small perturbation.

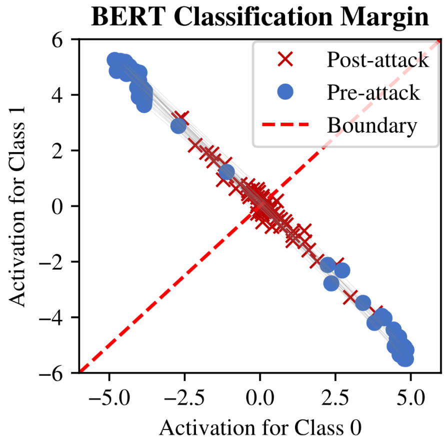

Figure 2: Final binary classifier's activations before and after the attack on BERT. The activations are recorded from 64 successful attacks.

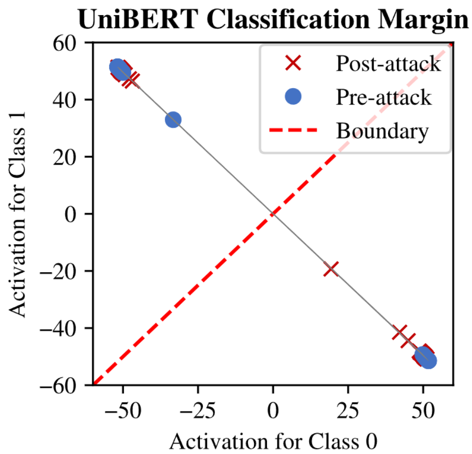

Figure 3: Final binary classifier's activations before and after the attack on UniBERT. The same 64 pre-attack sentences were used.

the decision boundary, which is the dashed line in Figure 3. Any data points below this decision boundary will be classified as class 0; above, class 1. Figure 4 shows the same graph but for the UniBERT model. Comparing the two models, we observed that BERT's pre-attack activations are more spread out than UniBERT's. Furthermore, BERT's pre-attack activations have smaller inter-class distances measured in both Euclidian and angular metrics, making the model more susceptible to adversarial attacks. On the other hand, UniBERT's pre-attack activations are tight clusters and well separated in Figure 4, demonstrating the desired effect of the SVM loss from Section 2.2.

4.3 Improved Robustness over the Baseline Models

The objective of UniBERT is to improve the post-attack accuracy for NLP. Table 3 delivers the key finding of this work: UniBERT can improve the adversarial robustness by up to 67.5%. We commence the attack operation as outlined in Section 3.4 on the 1000 random samples and measure the models' accuracies before and after the attack. Post-attack accuracy measures the percentage of correct classification after the attackers modify the input sentences. Attackers are only allowed a fixed number of trials for the attack, and they will only attack samples that were classified correctly before the attack begins. If they exceed this limit on any sample, the attack will cease on this sample, leaving it correctly classified and resulting in higher post-attack accuracy.

The data presented in Table 3 demonstrate that UniBERT is an effective architecture to prevent attackers from altering the output classification. For each attack recipe (i.e., PWWS, Textbugger, and Textfooler), we report the UniBERT improvement by comparing the post-attack accuracy of UniBERT against that of BERT and RoBERTa. We observe that UniBERT delivered double-digit improvements across all three types of NLP tasks and achieved the best improvement on binary sentiment analysis (yelp_polarity) with a 67.5% increase in post-attack accuracy, defending against Textfooler.

Although the post-attack accuracy has a remarkable improvement, the pre-attack accuracies deteriorate slightly. Among the three datasets we tested and with an SVM margin of 100, the highest degradation is 4.6% compared with RoBERTa in snli. Fortunately, as we observe in Figure 1, the SVM margin is a tunable variable that controls the balance between pre-attack accuracy and post-attack accuracy. Moreover, the frequency of unitary contraction can be reduced during training to enable a more expressive model and a higher pre-attack accuracy. As a classic bias vs. variance tradeoff, practitioners would set these hyperparameters (i.e., SVM margin and frequency of unitary contraction) according to the design requirements of their applications. In principle, it is also possible to compensate for the small drop in pre-attack accuracy with larger models or use a RoBERTa like pretraining technique; we did not analyze these methods due to resource constraints.

4.4 Comparison with Prior Arts in Defense Against NLP Attacks

We compared the post-attack accuracies of our model with two representative defense methodologies, which use data augmentation and regularization to enhance robustness. These works

Task	Model	Pre-Attack Accuracy (%)	Post-attack Accuracy (%)			UniBERT Improvement
			PWWS	Textbugger	Textfooler	
ag_news	BERT	94.6±0.8	32.4±1.9	35.8±2.9	13.4±1.3	34.9% to 65.6% absolute difference (1.7 to 4.9X when calculated as ratio)
	RoBERTa	**95.1±0.5**	40.8±0.5	48.3±1.3	16.7±0.7	
	UniBERT	92.3±0.8	**85.4±0.3**	**83.1±0.2**	**82.3±0.8**	
snli	BERT	90.1±0.4	1.5±0.1	4.0±0.6	3.8±0.5	13.7% to 20.0% absolute difference (3.7 to 14.1X when calculated as ratio)
	RoBERTa	**91.2±1.3**	1.3±0.3	5.0±0.4	4.0±0.7	
	UniBERT	86.6±0.7	**21.5±0.7**	**18.7±0.7**	**17.7±0.7**	
yelp_polarity	BERT	95.4±0.8	3.9±2.4	15.9±3.8	2.9±1.2	55.1% to 67.5% absolute difference (3.2 to 9.1X when calculated as ratio)
	RoBERTa	**96.6±0.5**	8.4±1.8	24.6±2.8	8.4±1.7	
	UniBERT	93.3±1.9	**75.3±1.6**	**79.6±1.6**	**75.9±1.7**	

Table 3: Post-attack Accuracy Improvement for UniBERT over the original BERT and RoBERTa models.

Task	Model	Post-attack Accuracy		UniBERT Improvement
		PWWS	Textfooler	
ag_news	RoBERTa+AMDA-Tmix	69.7%	56.3%	15.4% to 26.0% absolute difference (1.2 to 1.5X when calculated as ratio)
	RoBERTa+AMDA-Smix	70.0%	51.3%	
	UniBERT	**85.4%**	**82.3%**	

Table 4: Comparison with the data augmentation-based defense methods

Task	Model	Post-attack Accuracy		UniBERT Improvement
		Textbugger	Textfooler	
Snli	BERT+MRAT	9.9%	10.5%	5.3% to 6.5% absolute difference (1.4 to 1.5X when calculated as ratio)
	BERT+MRAT_PLUS	12.2%	12.4%	
	UniBERT	**18.7%**	**17.7%**	

Table 5: Comparison with the regularization-based defense methods

have reported post-attack accuracies on the same datasets and adversarial recipes as ours, enabling us to draw a fair comparison. The first defense method we contrasted with is based on data augmentation that utilizes data interpolation to generate additional samples (Si et al., 2021). Table 4 shows that our model achieved a 26% increase in post-attack accuracy comparing the data augmentation-based models: AMDA-Tmix & AMDA-Smix. Additionally, in comparison with regularization techniques, UniBERT surpasses MRAT & MRAT_PLUS models by 6.5% in post-attack performance as shown in Table 5.

5 Conclusion

Unitary weights with SVM loss are an effective defense against both typographical and synonym-swap adversarial attacks. The proposed UniBERT architecture is straightforward to implement and works well for a wide variety of practical NLP tasks. Our model defines a brand-new class of defense methodology against adversarial attacks in NLP and outperforms prior data augmentation-based or regularization-based techniques in adversarial robustness enhancement.

References

Naveed Akhtar and Ajmal Mian. 2018. Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey. *IEEE Access*, 6:14410–14430.

Christopher M. Bishop. 2006. *Pattern recognition and machine learning*.Information science and statistics. Springer, New York.

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, September. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal Sentence Encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium, November. Association for Computational Linguistics.

Hao-Yuan Chang and Kang L. Wang. 2021. Deep Unitary Convolutional Neural Networks. In Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2021*, pages 170–181, Cham. Springer International Publishing.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 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 4171–4186, Minneapolis, Minnesota, June. Association for Computational Linguistics.

Gamaleldin Elsayed, Dilip Krishnan, Hossein Mobahi, Kevin Regan, and Samy Bengio. 2018. Large Margin Deep Networks for Classification. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Robert Gilmore. 2008. *Lie Groups, Physics, and Geometry: An Introduction for Physicists, Engineers and Chemists*. Cambridge University Press, New York.

Gene H. Golub and Charles F. Van Loan. 2013. *Matrix computations*.Johns Hopkins studies in the mathematical sciences. The Johns Hopkins University Press, Baltimore, Fourth edition edition.

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. *arXiv:1907.11932 [cs]*, April. arXiv: 1907.11932.

Hamid Karimi, Tyler Derr, and Jiliang Tang. 2020. Characterizing the Decision Boundary of Deep Neural Networks. *arXiv:1912.11460 [cs, stat]*, June. arXiv: 1912.11460.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *Proceedings 2019 Network and Distributed System Security Symposium*, San Diego, CA. Internet Society.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep Text Classification Can be Fooled. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pages 4208–4215, Stockholm, Sweden, July. International Joint Conferences on Artificial Intelligence Organization.

Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song. 2017. SphereFace: Deep Hypersphere Embedding for Face Recognition. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6738–6746, Honolulu, HI, July. IEEE.

Alessio Miaschi and Felice Dell'Orletta. 2020. Contextual and Non-Contextual Word Embeddings: an in-depth Linguistic Investigation. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 110–119, Online, July. Association for Computational Linguistics.

Guido F. Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. 2014. On the Number of Linear Regions of Deep Neural Networks. *Advances in Neural Information Processing Systems*, 27.

John 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. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126, Online, October. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 142–148, San Diego, California, June. 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, July. Association for Computational Linguistics.

Enrique Romero, Lluis Marquez, and Xavier Carreras. 2004. Margin maximization with feed-forward neural networks: a comparative study with SVM and AdaBoost.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked Language Model Scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online, July. Association for Computational Linguistics.

Chenglei Si, Zhengyan Zhang, Fanchao Qi, Zhiyuan Liu, Yasheng Wang, Qun Liu, and Maosong Sun. 2021. Better Robustness by More Coverage: Adversarial and Mixup Data Augmentation for Robust Finetuning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1569–1576, Online, August. Association for Computational Linguistics.

Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu. 2018. Additive Margin Softmax for Face Verification. *IEEE Signal Processing Letters*, 25(7):926–930, July.

Eric Wong and Zico Kolter. 2018. Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. In *Proceedings of the 35th International Conference on Machine Learning*, pages 5286–5295. PMLR, July.

Yushun Xie, Zhaoquan Gu, Xiaopeng Fu, Le Wang, Weihong Han, and Yuexuan Wang. 2020. Misleading Sentiment Analysis: Generating Adversarial Texts by the Ensemble Word Addition Algorithm. In *2020 International Conferences on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics)*, pages 590–596. November.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative Study of CNN and RNN for Natural Language Processing. *arXiv:1702.01923 [cs]*, February. arXiv: 1702.01923.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Jiahao Zhao, Penghui Wei, and Wenji Mao. 2021. Robust Neural Text Classification and Entailment via Mixup Regularized Adversarial Training. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1778–1782, New York, NY, USA, July. Association for Computing Machinery.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning Books and Movies: Towards Story-Like Visual Explanations by Watching Movies and Reading Books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, Santiago, Chile, December. IEEE.