[MASK] Insertion for anti-adversarial attacks

Xinrong Hu[†], Ce Xu[†], Junlong Ma[†], Zijiang Huang[†],
Jie Yang[◇*], Yi Guo[‡], Johan Barthelemy[△]
[†]School of Computer Science and Artificial Intelligence, Wuhan Textile University
[◇]School of Computing and Information Technology, University of Wollongong
[‡]School of Computer, Data and Mathematical Sciences, Western Sydney University
[△]NVIDIA
{hxr, yaoxun}@wtu.edu.cn, jiey@uow.edu.au,
y.guo@westernsydney.edu.au, jbarthelemy@nvidia.com

Abstract

Adversarial attack aims to perturb input sequences and mislead a trained model for false predictions. To enhance the model robustness, defensing methods are accordingly employed by either data augmentation (involving adversarial samples) or model enhancement (modifying the training loss and/or model architecture). In contrast to previous work, this paper revisits the masked language modeling (MLM) and presents a simple yet efficient algorithm against adversarial attacks, termed [MASK] insertion for defensing (MI4D). Specifically, MI4D simply inserts [MASK] tokens to input sequences during training and inference, maximizing the intersection of the new convex hull (MI4D creates) with the original one (the clean input forms). As neither additional adversarial samples nor the model modification is required, MI4D is as computationally efficient as traditional fine-tuning. Comprehensive experiments have been conducted using three benchmark datasets and four attacking methods. MI4D yields a significant improvement (on average) of the accuracy between 3.2 and 11.1 absolute points when compared with six state-of-the-art defensing baselines.

1 Introduction

Pretrained Language Models (PLMs) have rapidly advanced the performance of the Natural Language Processing (NLP) tasks, such as text/document classification. Yet, abundant evidences also indicate that PLMs are vulnerable to adversarial attacks, and the model performance can be dramatically impacted by (even) small perturbations to the model input (Gao et al., 2018; Li et al., 2019; Li et al., 2020; Jin et al., 2020). As a result, adversarial defenses have received significant attention, with the ultimate goal of achieving the robust model accuracy on both the clean (original) and polluted (adversarial) inputs.

A large amount of research effort has been dedicated to adversarial defenses, ranging from the data augmentation, the model enhancement, to the randomized smoothing. Among data augmentation studies, recent works introduce small but controllable perturbations to pollute clean data and produce adversarial samples (Yoo and Qi, 2021; Dong et al., 2021; Zhou et al., 2021; Li et al., 2021; Meng et al., 2022), while the model is later trained on both the clean and polluted inputs. However, due to the additional adversarial samples, data-augmentation methods suffer from the requirement of enormous computational resources for training. Additionally, model-enhancement approaches focus on polishing the vanilla model via manipulating the training loss or network architecture, without acquiring additional adversarial data (Wang et al., 2021; Le et al., 2022; Liu et al., 2022). Yet, those methods often require extensive search among numerous candidates to determine optimal hyperparameters. Another line of work is to apply ensemble-based randomized smoothing techniques (Ye et al., 2020; Zeng et al., 2021). Unfortunately, they induce substantial overhead due to the ensemble classification; more importantly, their performance are unstable to different types of attacks (Zhang et al., 2022; Xu et al., 2022).

Our aim is then to explore a robust adversarial defensing algorithm, which neither relies on additional adversarial data (as data augmentation), nor adjusts the training loss and network architecture (as model enhancement), nor requests ensemble-based training (as randomized smoothing). By contrast, this paper revisits the masked language modeling (MLM) and further proposes a compact and performance-preserving algorithm, termed [MASK] insertion for defensing (MI4D). Specifically, MI4D only requires to insert [MASK] tokens at the beginning of input sequences to produce *masked inputs*. During training, (only) masked inputs are employed for the model fine-tuning, while

*Corresponding author.

Findings of the Association for Computational Linguistics: EACL 2023, pages 1058–1070

later polluted samples are masked in the same manner for inference. In contrast to the traditional MLM, the prediction task of `[MASK]` tokens is less emphasized in the proposed MI4D. Yet, the injected `[MASK]` plays a role of maximizing the intersection between the convex hull after attacking and that of the original (clean) input, thereby enhancing the defensing performance.

The main contributions of the proposed work are summarized as follows:

- A novel `[MASK]` insertion for defensing (MI4D) algorithm is proposed, neither relying on additional adversarial data nor modifying the training model nor requesting ensemble-based classification;
- MI4D is characterized by simply injecting `[MASK]` tokens at the beginning of input sequences during training and inference. Accordingly, the span of the convex hull (after injecting) is critical to retain more solution space as the clean one to enhance successful defense;
- Empirically, our proposed method outperforms six recent baselines on a combination of three standard benchmarks and four attacking methods, advancing the best state-of-the-arts by on average 3.2-11.1 absolute points in accuracy.

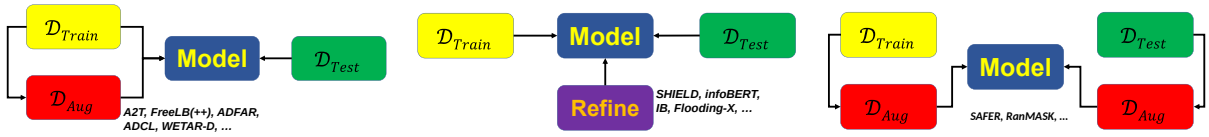
2 Related work

Constructing misleading samples to fool the trained neural-network models, adversarial attacks in the text domain can be mainly classified into two categories of the character- and word-level perturbation. The work of (Gao et al., 2018; Li et al., 2019) belongs to the character-level attack, from which the input is polluted by removing, substituting or inserting letters. On the other hand, word-based attacks usually involve the step of determining word importance, and replacing with their synonyms to maximize the prediction error of the model (Li et al., 2020; Jin et al., 2020).

Adversarial defense, by contrast, aims to form a robust model with high accuracy on both clean (original) and polluted (adversarial) samples. One of the most effective approaches is through the `data augmentation` (as shown in Fig. 1(a)), where adversarial samples are produced and fed into the model training. Specifically, **A2T** (Yoo and Qi, 2021) generates adversarial samples via employing a gradient-based method to identify important words, and iteratively substitutes with their synonyms using a DistilBERT similarity. **FreeLB** (and its variants) (Zhu et al., 2020; Li et al., 2021) imposes norm-bounded noises on embeddings of input sentences to produce adversarial samples. **ADFAR** (Bao et al., 2021) applies a frequency-aware randomization on both original and adversarial samples (by other attacking methods) to form a randomized adversarial set. This augmentation set is then combined with original and adversarial samples to train the model. More recently, in (Meng et al., 2022), **ADCL** generates adversarial examples using a geometry attack, which are later utilized as hard positive samples to train the model following a self-supervised contrastive learning. Xu *et al*. propose **WETAR-D** (Xu et al., 2022) as a sample reweighting method, in which the sample weight is adjusted by minimizing the loss from the validation set mixed of both original and adversarial examples.

Besides the data augmentation, another line of studies is proposed for the `model enhancement` to refine the model architecture and/or training loss, without acquiring additional adversarial samples (as shown in Fig. 1(b)). Among them, **SHIELD** (Le et al., 2022) modifies the last layer of an trained model and transforms it into an ensemble of multiple-expert predictors with stochastic weights. **Flooding-X** (Liu et al., 2022) introduces a regularization technique to prevent the overfitting of training samples. Wang *et al*. (Wang et al., 2021) propose **InfoBERT** to employ two mutual-information-based regularizers for suppressing noisy information between the input and the latent representation, and for increasing the correlation between local and global features. A similar work is found in (Zhang et al., 2022), where an information bottleneck layer (**IB**) is inserted between the encoder and the final classifier. This IB layer is utilized to extract robust and task-related features.

Additionally, a set of ensemble-based randomized smoothing methods have been proposed, shown in Fig. 1(c). **SAFER** (Ye et al., 2020), for instance, constructs stochastic input ensembles and leverages statistical properties of ensembles to classify testing samples. In **RanMASK** (Zeng et al., 2021), few input tokens are randomly substituted using `[MASK]` for fine-tuning, while testing samples are also masked (at different locations) to form



(a) data augmentation (b) model enhancement (c) randomized smoothing

Figure 1: Comparison of existing adversarial defensing methods.

several masked versions. The final prediction is then made by a majority vote from the ensemble of these masked versions.

The proposed method is different from existing approaches: (1) compared to adversarial based augmentation, no additional samples are required, and significant computational overhead is avoid accordingly; (2) compared to the model-enhancement ones, the proposed method is hyperparameter insensitive, which maintains the vanilla model training (loss and architecture) but only changes input formats; (3) compared to randomized smoothing, the ensemble based inference is no longer required.

3 Proposed method

Adversarial attack in text domain perturbs input sequences to maximally mislead the classification model, while this section presents a simple yet effective algorithm to reduce the model vulnerability, termed `[MASK]` insertion for defensing (MI4D).

3.1 `[MASK]` insertion for defensing

The proposed method MI4D is characterized by the normal fine-tuning process (the same network architecture and training loss as the vanilla model), while the only difference lies in the inserted `[MASK]` tokens at the beginning of input sequences during training and inference. Specifically, given the tokenized input sequence $\boldsymbol{x}$ (*i.e.*, $\boldsymbol{x}$=`[CLS]` $\boldsymbol{x}_1 \cdots \boldsymbol{x}_{|\boldsymbol{x}|}$ `[SEP]`), where $\boldsymbol{x}_i$ represents the i-th token from $\boldsymbol{x}$. For the text classification task, we aim to optimize an encoder $Enc(\cdot)$ and a Multilayer Perceptron (MLP) layer $f(\cdot)$ to map $\boldsymbol{x}$ to a desirable label y, *i.e.*, $f(Enc(\boldsymbol{x})) \mapsto y$.

Let b_M be the pre-defined masking budget (or the fraction of masked tokens). Then, MI4D injects M consecutive masks after `[CLS]` within $\boldsymbol{x}$ to form a masked sequence, that is, $\boldsymbol{x}'$=`[CLS]` `[MASK]`$_1 \cdots$ `[MASK]`$_M \boldsymbol{x}_1 \cdots \boldsymbol{x}_{|\boldsymbol{x}|}$ `[SEP]`, and $M = \lceil |\boldsymbol{x}| * b_M \rceil$.

Next, only $\boldsymbol{x}'$ (instead of $\boldsymbol{x}$) is utilized for training, while $Enc(\cdot)$ is leveraged to extract the latent representation for $\boldsymbol{x}'$ and the normal loss (such as the cross-entropy based) function $\mathcal{L}(\boldsymbol{x}', y)$ is adopted. During inference, with an unseen sequence $\bar{\boldsymbol{x}}$, the insertion procedure is repeated to inject M consecutive masks to $\bar{\boldsymbol{x}}$, *i.e.*, $\bar{\boldsymbol{x}}'$ = `[CLS]` `[MASK]`$_1 \cdots$ `[MASK]`$_M \bar{\boldsymbol{x}}_1 \cdots \bar{\boldsymbol{x}}_{|\bar{\boldsymbol{x}}|}$ `[SEP]`. The label of $\bar{\boldsymbol{x}}$ is finally produced by $f(Enc(\bar{\boldsymbol{x}}'))$.

3.2 Analysis

Notably, RanMASK (Zeng et al., 2021) *substitutes* input tokens with `[MASK]`, while MI4D *injects* `[MASK]`. Despite its simplicity, conceptually and computationally, MI4D has strong theoretical results as the following claim: *RanMASK is the lower bound of MI4D in terms of adversarial defensing performance*.

To prove the claim, given the tokenized input $\boldsymbol{x}$, the output of a self-attention module $\mathbf{Y}$ is derived by

$$\mathbf{Y} = \mathrm{softmax}(\mathbf{X}\mathbf{W}_1\mathbf{W}_2^\top\mathbf{X}^\top)\mathbf{X}\mathbf{W}_3,$$

where $\mathbf{X} \in \mathbb{R}^{|\boldsymbol{x}| \times d}$ is the latent representation of $\boldsymbol{x}$, d is the hidden dimension, and $\mathbf{W}_k$ ($\forall k \in [1, 3]$) are projection matrices with compatible dimensions. The property of $\mathrm{softmax}$ dictates that each row of $\mathbf{Y}$ (written as $\mathbf{y}_i$) is a convex construction of $\mathbf{X}\mathbf{W}_3$ (written as $\tilde{\mathbf{X}}$), *i.e.*, $\mathbf{y}_i \in \mathcal{C}(\tilde{\mathbf{X}})$, where $\mathcal{C}(\tilde{\mathbf{X}})$ stands for the convex hull of $\tilde{\mathbf{X}}$ (see Fig. 2 for the area enclosed by thick dashed lines). The same process happens in multi-head attention modules. They operate in different projected spaces but the observation of the convex construction still holds.

We hypothesize that the successful defense rate (against attacks) is determined by the intersection of the new convex hull (a defensing method creates) with the original convex hull (the clean data forms), and the larger intersection results in the better defensing performance. This leads to the following assumption.

Assumption 1. *Given the latent representation of the clean input and its adversarial version in* $\mathbf{X}$ *and* $\mathbf{X}'$*, for a very small* $\epsilon \approx 0$*,*

$$\mathbb{P}(\textit{successful defense}) = \frac{\mathrm{Vol}(\mathcal{C}(\mathbf{X}'_\epsilon) \cap \mathcal{C}(\mathbf{X}_\epsilon))}{\mathrm{Vol}(\mathcal{C}(\mathbf{X}_\epsilon))},$$

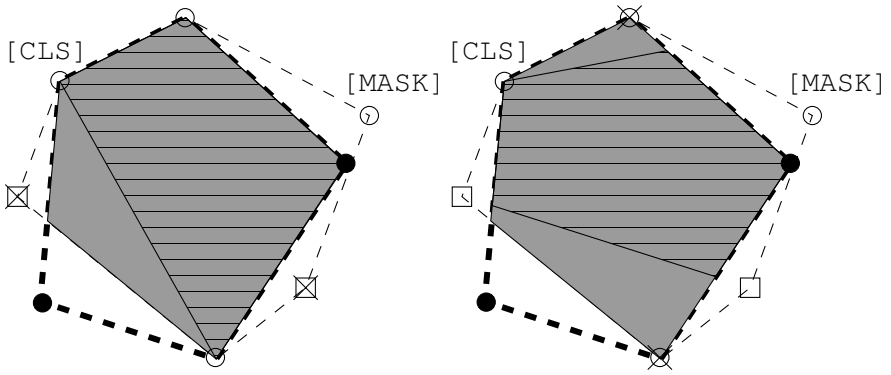


Figure 2: Illustration of the proof to Corollary 1. Circles represent tokens from the input sequence as vectors in projected space. Black circles are attacked and further replaced by the square ones. Gray (Stripped) areas are the intersection of MI4D (RanMASK) convex hull with the original data convex hull. Left: removing attacked tokens; Right: removing two good tokens. Removed tokens are marked by cross.

where $\mathrm{Vol}(\cdot)$ *is a function to estimate the volume of a geometric object, and* $\mathbf{X}_\epsilon$ *is the* ϵ *ball centred at* $\mathbf{X}$.

The ϵ ball spans the convex hulls to the dimension of the ambient space so that volume always exists. More importantly, it also reflects the model tolerance to the variation of vector representations, indicating small disturbance will not affect the model output. To ease notation, we omit the ϵ in later development.

MI4D differs from RanMASK at no random elimination of input tokens, but a simple insertion of `[MASK]` while keeping clean and polluted tokens. This choice leads to the fact that the convex hull formed by MI4D always contains those by RanMASK as guaranteed by the following lemma.

Lemma 1. *Given a set* $\mathbf{X}$*, we have* $\mathcal{C}(\mathbf{S}) \subseteq \mathcal{C}(\mathbf{X})$ *for any subset* $\mathbf{S} \subseteq \mathbf{X}$*. The equality holds when* $\mathbf{S} = \mathbf{X}$ *trivially or otherwise* $\mathbf{S}$ *contains all the anchor points of* $\mathcal{C}(\mathbf{X})$*, i.e., the convex hull vertices.*

Proof. Let $\mathcal{X}$ be the index set for $\mathbf{X}$ and a subset $\mathcal{S} \subseteq \mathcal{X}$ gives the indices for $\mathbf{S}$. For any point $p \in \mathcal{C}(\mathbf{S})$, $p = \sum_{i \in \mathcal{S}} \lambda_i \mathbf{x}_i$ such that $\lambda_i \geq 0$ and $\sum \lambda_i = 1$, *i.e.*, the convex condition. Apparently $p \in \mathcal{C}(\mathbf{X})$ as well by setting $\lambda_j = 0$ for $j \in \mathcal{X} \backslash \mathcal{S}$.

For any point $x_i \in \mathbf{X}$, it is either an anchor point or an internal point referring to $\mathcal{C}(\mathbf{X})$. If $\mathbf{S}$ contains only anchor points, $\mathcal{C}(\mathbf{S}) = \mathcal{C}(\mathbf{X})$ as the internal points can be "absorbed". To see this, assume $\mathbf{x}_1$ is an internal point, then $\mathbf{x}_1 = \sum_{i>1} \beta_i \mathbf{x}_i$ and all β_is for $i > 1$ satisfying convex condition. Then

$$p = \sum_{i=1} \lambda_i \mathbf{x}_i = \sum_{i>1} (\lambda_i + \lambda_1 \beta_i) \mathbf{x}_i.$$

Therefore, $\mathcal{C}(\mathbf{X}) = \mathcal{C}(\mathbf{X}_{-1})$ where $\mathbf{X}_{-1}$ is the set of vectors after removing $\mathbf{x}_1$. After eliminating internal points, the convex hull will still be the same. □

The immediate result from above lemma is the following corollary stating the relations between convex hulls generated by MI4D and RanMASK.

Corollary 1. *Convex hull generated by MI4D always contains those by RanMASK.*

Proof. Let $\mathbf{X}$ be the latent representation of input tokens for MI4D (including `[MASK]` tokens), and $\mathcal{X}$ the corresponding indices set. RanMASK runs several, say n, times of random eliminations but keeping `[MASK]` tokens, *i.e.*, leading to index subsets $\mathcal{S}_i$ ($\mathcal{S}_i \subset \mathcal{X}$ ($i = 1 \sim n$)). Clearly, from **Lemma 1**, we have $\forall i$, $\mathcal{C}(\mathbf{S}_i) \subseteq \mathcal{C}(\mathbf{X})$, where $\mathbf{S}_i$ is the corresponding representations in $\mathbf{X}$ indexed by $\mathcal{S}_i$. Equality holds only when $\mathbf{S}_i$ contains all anchor points set in $\mathcal{C}(\mathbf{X})$. □

Next, we are ready to formalize and prove the claim as the following proposition.

Proposition 1. *Given Assumption 1, MI4D has at least the same successful defensing rate as that of RanMASK. In other words, MI4D has at least equally good adversarial defense performance as RanMASK.*

Proof. Let $\mathbf{X}'$ ($\mathbf{X}$) be the adversarial (original) representations in latent space, and $\mathbf{S}_i$ be the i-th subset of $\mathbf{X}'$. The successful defensing probability of MI4D and RanMASK at the i-th run is defined as p_m and p_{r_i}, respectively. We have

$$p_m = \frac{\mathrm{Vol}(\mathcal{C}(\mathbf{X}) \cap \mathcal{C}(\mathbf{X}'))}{\mathrm{Vol}(\mathcal{C}(\mathbf{X}))},$$

and

$$p_{r_i} = \frac{\mathrm{Vol}(\mathcal{C}(\mathbf{X}) \cap \mathcal{C}(\mathbf{S}_i))}{\mathrm{Vol}(\mathcal{C}(\mathbf{X}))}.$$

For RanMASK to succeed, the successful $\mathbf{S}_i$s have to be chosen and become majority and hence the final success probability of RanMASK $p_r = \mathbb{P}(\exists i, \mathbf{S}_i \text{ success } \wedge \text{ successful sets are majority})$. Clearly,

$$\begin{aligned} p_r &\leq \min(\mathbb{P}(\exists i,\ \mathbf{S}_i \text{ success }), \\ &\quad \sum_{k \geq \lceil n/2 \rceil} \mathrm{B}(k; n, \max_i\{p_{r_i}\})) \\ &\leq \min(\max_i\{p_{r_i}\}, \sum_{k \geq \lceil n/2 \rceil} \mathrm{B}(k; n, \max_i\{p_{r_i}\}) \\ &\leq p_m, \end{aligned}$$

where $\mathrm{B}(k; n, p)$ is the probability of k out of n trials successes with probability p, *i.e.*, $\binom{n}{k}p^k(1-p)^{n-k}$. The last inequality comes from Corollary 1 as $p_m \geq p_{r_i}$ $(\forall i)$ and hence $p_m \geq \max_i(p_{r_i})$. □

Overall, an illustration is shown in Fig. 2 with the convex hull of MI4D ($\mathcal{C}_m$) and those of RanMASK ($\mathcal{C}_r$) with two different random eliminations. The gray area shows the intersection of MI4D convex hull with the original convex hull $\mathcal{C}$, *i.e.*, $\mathcal{C} \cap \mathcal{C}_m$, while stripe areas are the intersections of those of RanMASK, *i.e.*, $\mathcal{C} \cap \mathcal{C}_r$. We know that $\mathcal{C} \cap \mathcal{C}_m$ always contains $\mathcal{C} \cap \mathcal{C}_r$. As such, the span of the convex hull after `[MASK]` insertion is critical to retain more solution space to enhance successful defense. Additionally, we also infer that the position of inserted `[MASK]` tokens and the number of insertions are less important (multiple of them differ only at the positional encoding), as they may well be in the ϵ ball of the same `[MASK]` token itself. These inferences are verified in the ablation study.

4 Experiments

4.1 Setup

Datasets. Experiments are carried on three text classification benchmarking datasets, including **SST2**(Socher et al., 2013) (sentiment classification on the Stanford Sentiment Treebank corpus), **AGNEWS**(Zhang et al., 2015) (category classification for news articles from more than 2000 news sources), **IMDB**(Maas et al., 2011) (document polarity classification using the online IMDB database).

Attacking algorithms. Four adversarial attacking methods are implemented using *TextAttack* (Morris et al., 2020) to pollute input sequences, that is,

- **DeepWordBug** (Gao et al., 2018) deletes, replaces, and inserts characters to inputs;
- **TextBugger**(Li et al., 2019) performs perturbations of space insertion, char deletion/swapping, and synonym substitution;
- **BERT-Attack** (Li et al., 2020) substitutes key words using a pre-trained masked model;
- **TextFooler** (Jin et al., 2020) replaces important words with their synonyms.

Evaluation Metrics. Three measurements are considered to evaluate the model robustness against adversarial attacks. Specifically, **Cln%** refers to the model classification accuracy on the original clean data. **Aua**% is the classification accuracy under certain adversarial attacks, and higher Aua% means better defensing performance. **Suc%** is defined as the number of examples successfully being fooled against the number of all attempted attacks; accordingly, lower Suc% indicates the higher model robustness.

All experiments are performed five trials with random seeds for each dataset. For each run, the training is performed with batches of 32 sequences of length 512. The maximal number of training epoch is 10. Meanwhile, 10% samples are randomly selected from the training set to form the validation set, and the training stops if the validation accuracy fails to improve for one epoch. On the other hand, 1,000 testing examples are randomly selected for the evaluation purpose. This is the typical experimental setting as (Wang et al., 2021; Zhang et al., 2022; Zeng et al., 2021). More details are provided in Appendix A.1.

4.2 Main results

The following state-of-the-art defensing methods are employed to compare with the proposed MI4D, including **WETAR-D**(Xu et al., 2022), **FreeLB++**(Li et al., 2021), **IB**(Zhang et al., 2022), **InfoBERT**(Wang et al., 2021), **Flooding-X**(Liu et al., 2022), and **RanMASK**(Zeng et al., 2021). Among them, the first two methods are based on adversarial data augmentation, while IB, InfoBERT and Flooding-X are for the model enhancement. The last one represents the randomized smoothing method.

Table 1: Averaged defensing performance (over five trails) obtained by MI4D and current SOTAs using four attacking methods, including TextFooler, BERT-Attack, Deepwordbug, and TextBugger. The number with bold, † and * represents the best, second, and third result, respectively.

Datasets	Methods	TextFooler			BERT-Attack			Deepwordbug			TextBugger		
		Cln%	Aua%	Suc%	Cln%	Aua%	Suc%	Cln%	Aua%	Suc%	Cln%	Aua%	Suc%
SST2	Baseline	94.1*	5.4	94.3	94.1*	6.2	93.4	94.1	17.0	81.9	94.1*	29.7	68.4
	WETAR-D	**94.3**	31.1*	67.0*	**94.3**	31.4*	66.7*	94.3†	42.3†	55.1†	**94.3**	56.3†	40.3†
	FreeLB++	93.9	23.6	74.9	93.9	21.2	77.4	93.9	33.6	64.2	93.9	46.6	50.4
	IB	94.1*	28.9	69.3	94.1*	26.5	71.8	94.1	40.5*	57.0*	94.1*	51.9*	44.8*
	InfoBERT	94.0	19.5	79.3	94.0	18.4	80.4	94.0	29.7	68.4	94.0	42.5	54.8
	Flooding-X	94.2†	32.2†	65.8†	94.2†	**35.4**	**62.4**	94.2*	38.2	59.4	94.2†	49.9	47.0
	RanMASK	92.7	12.9	86.1	93.0	11.4	87.7	92.7	27.5	70.3	92.8	39.9	57.0
	MI4D	**94.3**	**36.4**	**61.4**	**94.3**	34.5†	63.4†	**94.4**	**45.6**	**51.7**	94.2†	**58.3**	**38.1**
AGNEWS	Baseline	94.2*	15.8	83.2	94.2*	26.7	71.8	94.2	33.0	65.0	94.2	49.2	47.8
	WETAR-D	94.0	64.4*	31.5*	94.0	57.5†	38.8†	94.0	63.7†	32.2†	94.0	71.6†	23.8†
	FreeLB++	**95.1**	58.7	38.3	**95.1**	38.8	59.2	**95.1**	55.1	42.1	**95.1**	64.9	31.8
	IB	93.9	60.7	35.4	93.9	51.6	45.0	93.9	59.2	37.0	93.9	63.6	32.3
	InfoBERT	93.6	51.3	45.2	93.6	39.9	57.4	93.6	53.9	42.4	93.6	50.6	45.9
	Flooding-X	94.4†	**68.9**	**27.0**	94.4†	56.4*	40.3*	94.4*	**65.3**	**30.8**	94.4*	70.3*	25.5*
	RanMASK	93.9	25.0	73.4	93.7	39.3	58.1	93.7	29.4	68.6	93.2	61.2	34.3
	MI4D	94.2*	66.7†	29.2†	94.1	**69.7**	**25.9**	94.6†	62.4*	34.0*	94.6†	**73.9**	**21.9**
IMDB	Baseline	91.5	0.5	99.4	91.5	0.6	99.3	91.5	48.5	47.0	91.5	11.9	87.0
	WETAR-D	92.1	47.1	48.9	92.1	34.7*	62.3*	92.1	90.0†	2.3†	92.1	58.3	36.7
	FreeLB++	93.3*	36.3	61.1	93.3	21.0	77.5	93.3*	78.3	16.1	93.3*	42.2	54.8
	IB	91.9	51.3†	44.2†	91.9	40.6†	55.8†	91.9	87.3*	5.0*	91.9	64.1†	30.3†
	InfoBERT	91.8	16.9	81.6	91.8	15.8	82.8	91.8	62.3	32.1	91.8	37.6	59.0
	Flooding-X	**94.7**	48.5*	48.8*	**94.7**	33.4	64.7	**94.7**	83.1	12.2	**94.7**	62.3*	34.2*
	RanMASK	93.0	18.0	80.7	93.5*	17.0	81.8	92.5	66.0	28.7	92.5	18.0	80.5
	MI4D	94.5†	**56.2**	**40.5**	94.3†	**54.2**	**42.5**	94.4†	**93.6**	**0.8**	94.5†	**69.8**	**26.1**
AVG	Baseline	93.3	7.2	92.3	93.3	11.2	88.1	93.3	32.8	64.6	93.3	30.3	67.7
	WETAR-D	93.5	47.5*	49.1*	93.5	41.2*	56.0†	93.5	65.3†	29.9†	93.5*	62.1†	33.6†
	FreeLB++	94.1*	39.5	58.1	94.1*	27.0	71.4	94.1*	55.7	40.8	94.1†	51.2	45.6
	IB	93.3	47.0	49.6	93.3	39.6	57.6	93.3	62.3*	33.0*	93.3	59.9	35.8*
	InfoBERT	93.1	29.2	68.7	93.1	24.7	73.5	93.1	48.6	47.7	93.1	43.6	53.3
	Flooding-X	**94.4**	49.9†	47.2†	**94.4**	41.7†	55.8*	94.4†	62.2	34.2	**94.4**	60.8*	35.6
	RanMASK	93.2	18.6	80.1	93.4	22.6	75.9	93.0	41.0	55.9	92.8	39.7	57.3
	MI4D	94.3†	**53.1**	**43.7**	94.2†	**52.8**	**43.9**	**94.5**	**67.2**	**28.8**	**94.4**	**67.3**	**28.7**

The RoBERTa-base model (Liu et al., 2019) is employed as the **Baseline**. All contender methods are re-implemented using their released codes, and their key configurations are summarized in Appendix A.1. Their results are competing with those reported. Additionally, for MI4D most of the hyperparameters, such as learning rate, are consistent with the vanilla RoBERTa-base, while the masking budget b_M is set as 30%. The comparison results over five trails are shown in Table 1.

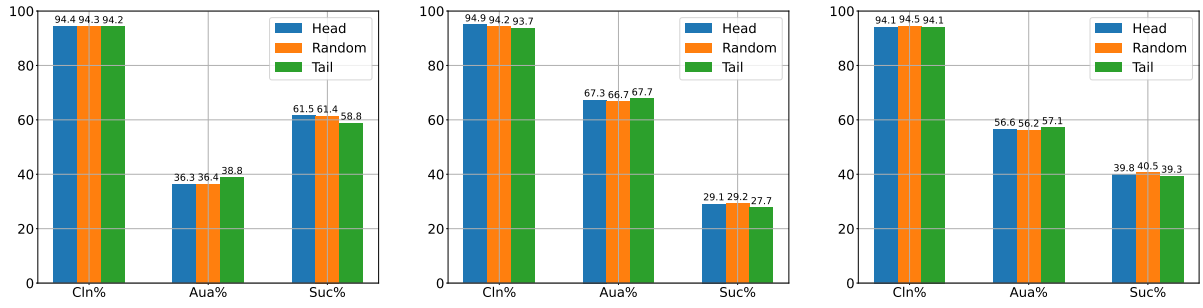
To begin with, the proposed MI4D achieves comparative results of averaged Cln% (94.35%) across all three clean testing datasets. The performance is only second to that of Flooding-X (averaged 94.40%), while a consistent improvement is observed in comparison with other existing methods. Importantly, MI4D achieves the state-of-the-art defensing accuracy in terms of Aua (60.18%) and Suc (36.40%) outperforming all contenders. Notably, all methods seemingly perform better against character-level attacks (Deepwordbug and TextBugger), which demonstrates the difficulty of defensing word-based attacks. Yet, MI4D still achieves the largest improvement (in comparison with the Baseline) and secures averaged 43.85 and 46.85 absolute percent points on Aua% and Suc% for the TextFooler and BERT-Attack, respectively.

By contrast, another `[MASK]` based approach (*i.e.*, RanMASK) scores the worst performance across three datasets. The main difference between RanMASK and ours lies in the usage of `[MASK]` tokens (substitution or insertion). By replacing residual tokens after attacking, RanMASK could further destroy the original semantic of input sequences. However, MI4D spans the semantic convex hull to increase the chance of including original anchor points, as Lemma 1 and Corollary 1 indicated, so as to enhance the defensing performance.

4.3 Ablation study

To better understand the effectiveness of the proposed method, a series of careful analysis is carried out. Again, all results are reported as an averaged accuracy over five trials.

On the masking location. To start with, the ablation experiment is performed to understand the impact from the location of inserted `[MASK]` tokens. In comparison with adding `[MASK]` right



(a) SST2 (b) AGNEWS (c) IMDB

Figure 3: Impact analysis of the masking location from either adding `[MASK]` after `[CLS]` (labeled as *Head*), randomly (labeled as *Random*) or at the end of the input sequence (labeled as *Tail*).

after the `[CLS]` (labeled as *Head* hereafter), the *Random* one is implemented to randomly insert `[MASK]` following a uniform sampling until b_M is met (where b_M is the masking budget). Similarly, we also consider to insert at the end of the input sequence (labeled as *Tail*).

With b_M=0.3, the comparison result using three datasets and the TextFooler attack is shown in Fig. 3 (results from other attack methods can be found in Appendix A.2). Clearly, the proposed method is insensitive to the masking location, due to the similar performance achieved by either *Head*, *Random* or *Tail* insertion. This shows positional encoding has negligible effect as we asserted in analysis, as the position embedding is less important compared to the token embedding. In MI4D context, exactly same `[MASK]` tokens are inserted and they do not change the relative order of existing tokens. Therefore position embeddings can be seen as a disturbance to "tag" on token embeddings to create the small variation, and have less impact on MI4D.

On the masking budget. The following experiments are to evaluate the impact of the masking budget (b_M) on the proposed method. Obviously, with a higher value of b_M, more `[MASK]` tokens will be inserted that could lead to more perturbed samples. Specifically, experiments are conduced by varying b_M from 0.1 to 0.9. We need to point out that for the dataset of SST2, with b_M=0.1 it is equivalent to injecting only 1 `[MASK]` token due to the average length of input sequences. As such, we are particularly interested in the model performance with/out `[MASK]`.

The comparison is shown in Fig. 4 for the MI4D performance against the TextFooler attack on three datasets (results from other attack methods can be found in Appendix A.3). Notably, the results demonstrate the robustness of the proposed method to different masking budgets. That is, MI4D observes a stable defensing performance across all three evaluation metrics for different masking budgets. As the span of the convex hull is utterly important rather than its multiplicity, this observational experiment once again confirms our inference in the Analysis.

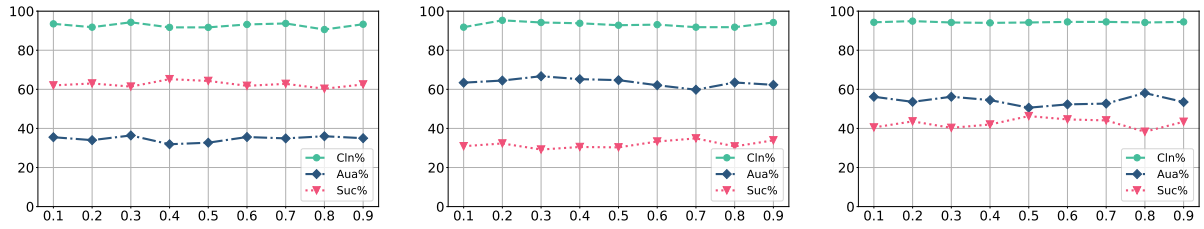
4.4 Discussion

In this section, we investigate different strategies of utilizing `[MASK]` tokens, and further seek for a reasonable explanation for the result. Again, experiments are conducted with `[MASK]` being inserted after `[CLS]` and b_M=0.3.

When to insert. First, we discuss the `[MASK]` insertion whether for training and/or inference. That is, three scenarios are considered to insert `[MASK]`: (1) only during the training, (2) only during the inference, and (3) both training and testing (equivalent to MI4D).

The comparison is shown in Table 2. The "Train only" variant is observed with the worst performance for the mostly collapsed convex hull, while others have more "developped" convex hull to embrace the original solution space. We highlight that including `[MASK]` in training is to fine-turning token embedding as a semantic place holder, and hence a "wild card". Accordingly, the capacity to span the convex hull to more likely intersect with original one is further enhanced, although `[MASK]` is employed for extensive pre-training of PLMs before.

What to substitute. Hereafter the impact from substituting/masking different types of tokens is discussed, where tokens are cast as *polluted* (be-



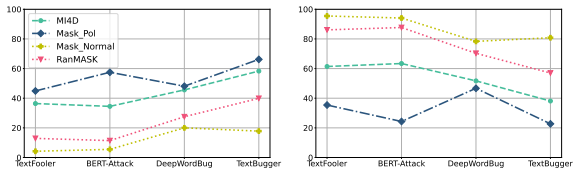
(a) SST2 (b) AGNEWS (c) IMDB

Figure 4: Comparison of the defensing performance as a function of the masking budget (b_M), against the TextFooler attack across three datasets (where x-axis represents b_M).

Table 2: Averaged defensing performance via masking training and/or testing samples for MI4D, while the Baseline method (vanilla RoBERTa-base) is adopted for reference.

Datasets	Strategy	TextFooler			BERT-Attack			Deepwordbug			TextBugger		
		Cln%	Aua%	Suc%	Cln%	Aua%	Suc%	Cln%	Aua%	Suc%	Cln%	Aua%	Suc%
SST2	Baseline	94.1	5.4	94.3	94.1	6.2	93.4	94.1	17.0	81.9	94.1	29.7	68.4
	Train only	93.0	6.1	93.4	93.4	7.2	92.3	93.9	19.6	79.1	93.7	31.4	66.5
	Test only	92.3	31.8	65.5	92.8	31.5	66.0	92.2	35.3	62.2	92.4	54.9	40.5
	Train+Test	94.3	36.4	61.4	94.3	34.5	63.4	94.4	45.6	51.7	94.2	58.3	38.1
AGNEWS	Baseline	94.2	15.8	83.2	94.2	26.7	71.7	94.2	33.0	65.0	94.2	49.2	47.8
	Train only	93.6	11.2	88.0	92.4	18.2	80.3	93.4	18.1	80.6	93.3	47.8	48.7
	Test only	91.0	52.0	42.8	92.0	63.4	29.7	92.1	43.8	52.4	93.0	71.4	23.2
	Train+Test	94.2	66.7	29.2	94.1	69.7	25.9	94.6	62.4	34.0	94.6	73.9	21.9
IMDB	Baseline	91.5	0.5	99.4	91.5	0.6	99.3	91.5	48.5	47.0	91.5	11.9	87.0
	Train only	93.8	22.7	75.8	93.1	20.7	77.8	92.1	53.3	42.1	93.3	32.5	65.2
	Test only	94.1	44.2	52.9	94.2	37.5	61.1	94.2	84.4	10.4	94.2	65.9	30.0
	Train+Test	94.5	56.2	40.5	94.3	54.2	42.5	94.4	93.6	0.8	94.5	69.8	26.1
AVG	Baseline	93.3	7.2	92.3	93.3	11.2	88.1	93.3	32.8	64.6	93.3	30.3	67.7
	Train only	93.5	13.3	85.7	93.0	15.4	83.5	93.1	30.3	67.3	93.4	37.2	60.1
	Test only	92.5	42.7	53.7	93.0	44.1	52.3	92.8	54.5	41.7	93.2	64.1	31.2
	Train+Test	94.3	53.1	43.7	94.2	52.8	43.9	94.5	67.2	28.8	94.4	67.3	28.7

ing attacked) and *normal* (remaining unchanged). The following experiment then involves MI4D and three other variants for comparison, that is to (1) only substitute *polluted* (labeled as Mask_Pol), (2) only substitute *normal* (labeled as Mask_Normal), and (3) substitute randomly (explicitly as RanMASK). Fig. 5 illustrates the defensing accuracy of masking different types of tokens from the SST2 dataset (results from other datasets can be found in Appendix A.4).



(a) Aua% (b) Suc%

Figure 5: Comparison of the model performance against categorizes of masked tokens with SST2.

The comparison results clearly imply that the best performance is achieved via substituting/masking polluted tokens (*i.e.*, Mask_Pol), while Mask_Normal is the worst. In our hypothesis, when polluted tokens are replaced by `[MASK]`, it generates the convex hull that has the maximum overlap with the original one and hence leads to the best chance to defense. By contrast, Mask_Normal introduces more noise by maintaining perturbed but removing normal tokens. Notably, as there is no clue about adversarial attacking on which specific tokens in reality, Mask_Pol and Mask_Normal then reveals the theoretically best and worst defensing outcome (or the **upper** and **lower** bound), respectively.

RanMASK is then a special combination of Mask_Pol and Mask_Normal, as tokens of either *polluted* or *normal* are randomly masked out with a predefined probability. On the other hand, the proposed MI4D becomes an effective solution for masking inputs (due to the uncertainty of which tokens being polluted during testing), that is consistently better than RanMASK (randomly mask tokens). Again, the reason is that MI4D includes all by exploiting the fact that polluted tokens are still informative, to some extent, when they are combined with residuals ones, to create a larger convex hull overlapping (compared to RanMASK)

with the original one. That is shown clearly in Proposition 1.

[MASK] or others. The last experiment aims to investigate the possibility of injecting different tokens, instead of [MASK]. Specifically, the [PAD] token is selected and further inserted into the original input sequence. Note that, in this regard, all other configurations (such as the masking budget and the random insertion) remain explicitly the same, but only to replace [MASK] with [PAD] for the injection. Table 3 reports the averaged performance using the SST2 dataset with four attacks. As observed, the performance using [PAD] is similar to that of [MASK], indicating we can insert [MASK] (or similar) as a "wild card" to increase the span of the convex hull.

Table 3: Averaged defensing performance via injecting [PAD] (instead of [MASK]) tokens.

	TextFooler			BERT-Attack		
	Cln%	**Aua%**	**Suc%**	**Cln%**	**Aua%**	**Suc%**
[MASK]	94.3	36.4	61.4	94.3	34.5	63.4
[PAD]	94.1	36.4	61.4	93.5	32.8	64.9
	Deepwordbug			**TextBugger**		
	Cln%	**Aua%**	**Suc%**	**Cln%**	**Aua%**	**Suc%**
[MASK]	94.4	45.6	51.7	94.2	58.3	38.1
[PAD]	93.2	44.7	52.0	93.1	63.1	32.3

5 Conclusion

We propose a novel adversarial defensing algorithm (MI4D), that is hyperparameter insensitive and structure free. The proposed method simply inserts [MASK] tokens at the beginning of input sequences, and follows the normal fine-tuning to train the model. Theoretically speaking, we have argued that adding additional [MASK], while remaining other residual tokens, creates a large convex hull overlapping with that of the clean one to increase the defensing probability. Empirically, in comparison to existing state-of-the-arts, the proposed algorithm exhibits superior performance on three benchmark datasets with four attack methods. In future work, we could combine with external knowledge for more strategical masking. More importantly, MI4D is agnostic to downstream tasks, *i.e.*, we could incorporate it into other applications.

Limitations

Our theoretic analysis is constructed on a crucial assumption asserting that the successful defense probability is determined by the volume of the convex hull formed by the input. Although our empirical study results confirmed the inferences based on this assumption repeatedly (shown in Section 4.4), we are still seeking direct dynamics of the convex hull to the prediction/classification probability where a more rigorous result may be derived. We envisage that the understanding of the current model behavior can lead to more robust models against adversarial attacks and hence further improvement to text classification.

Acknowledgments

This work was partially supported by the Australian Research Council Discovery Project (DP210101426) and the Australian Research Council Linkage Project (LP200201035).

References

Rongzhou Bao, Jiayi Wang, and Hai Zhao. 2021. Defending pre-trained language models from adversarial word substitution without performance sacrifice. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3248–3258, Online. Association for Computational Linguistics.

Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. In *International Conference on Learning Representations*.

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.

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. volume 34, pages 8018–8025.

Thai Le, Noseong Park, and Dongwon Lee. 2022. SHIELD: Defending textual neural networks against multiple black-box adversarial attacks with stochastic multi-expert patcher. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6661–6674, Dublin, Ireland. Association for Computational Linguistics.

Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *Network and Distributed Systems Security (NDSS) Symposium*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. BERT-ATTACK: Adversarial attack against BERT using BERT. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages

6193–6202, Online. Association for Computational Linguistics.

Zongyi Li, Jianhan Xu, Jiehang Zeng, Linyang Li, Xiaoqing Zheng, Qi Zhang, Kai-Wei Chang, and Cho-Jui Hsieh. 2021. Searching for an effective defender: Benchmarking defense against adversarial word substitution. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3137–3147, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Qin Liu, Rui Zheng, Bao Rong, Jingyi Liu, ZhiHua Liu, Zhanzhan Cheng, Liang Qiao, Tao Gui, Qi Zhang, and Xuanjing Huang. 2022. Flooding-X: Improving BERT's resistance to adversarial attacks via loss-restricted fine-tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5634–5644, Dublin, Ireland. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv Preprint*, abs/1907.11692.

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.

Zhao Meng, Yihan Dong, Mrinmaya Sachan, and Roger Wattenhofer. 2022. Self-supervised contrastive learning with adversarial perturbations for defending word substitution-based attacks. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 87–101, Seattle, United States. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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.

Boxin Wang, Shuohang Wang, Yu Cheng, Zhe Gan, Ruoxi Jia, Bo Li, and Jingjing Liu. 2021. InfoBERT: Improving robustness of language models from an information theoretic perspective. In *International Conference on Learning Representations*.

Jianhan Xu, Cenyuan Zhang, Xiaoqing Zheng, Linyang Li, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2022. Towards adversarially robust text classifiers by learning to reweight clean examples. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1694–1707, Dublin, Ireland. Association for Computational Linguistics.

Mao Ye, Chengyue Gong, and Qiang Liu. 2020. SAFER: A structure-free approach for certified robustness to adversarial word substitutions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3465–3475, Online. Association for Computational Linguistics.

Jin Yong Yoo and Yanjun Qi. 2021. Towards improving adversarial training of NLP models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 945–956, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jiehang Zeng, Xiaoqing Zheng, Jianhan Xu, Linyang Li, Liping Yuan, and Xuanjing Huang. 2021. Certified robustness to text adversarial attacks by randomized [Mask]. *arXiv preprint*, abs/2105.03743.

Cenyuan Zhang, Xiang Zhou, Yixin Wan, Xiaoqing Zheng, Kai-Wei Chang, and Cho-Jui Hsieh. 2022. Improving the adversarial robustness of NLP models by information bottleneck. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3588–3598, Dublin, Ireland. Association for Computational Linguistics.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Yi Zhou, Xiaoqing Zheng, Cho-Jui Hsieh, Kai-Wei Chang, and Xuanjing Huang. 2021. Defense against synonym substitution-based adversarial attacks via Dirichlet neighborhood ensemble. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5482–5492, Online. Association for Computational Linguistics.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2020. FreeLB: Enhanced adversarial training for Natural Language Understanding. In *International Conference on Learning Representations*, pages 26–30, Addis Ababa, Ethiopia.

A Appendix

A.1 Training Details

The RoBERTa-base model (Liu et al., 2019) is employed as the contextual encoder. The dropout rate across all layers is set as 0.1. The Adam optimizer with a dynamic learning rate is adopted, for which the learning rate is warmed up for 10 thousand steps to a maximum value of $2e^{-5}$ before decaying linearly to a minimum value of $1e^{-6}$ (by the cosine annealing) and a gradient clip of $(-1, 1)$. Additionally, for WETAR-D, 50% of samples are polluted in the validation set (the size of 256); for FreeLB++, the number of search steps (for adversarial samples) is 30; for IB, the hidden dimension for the IB layer is set as 150 and the penalty of the IB loss is 0.1; for InfoBERT, the penalty of the mutual-information loss is 5×10^{-2}; for RanMASK, the masking budget is set as 30%, while the majority vote is adopted for the final classification stage. At last, all models are performed using a machine with NVIDIA Tesla V100 PCIe of 32G GPU memory.

A.2 Impact from the location of inserting *[MASK]*

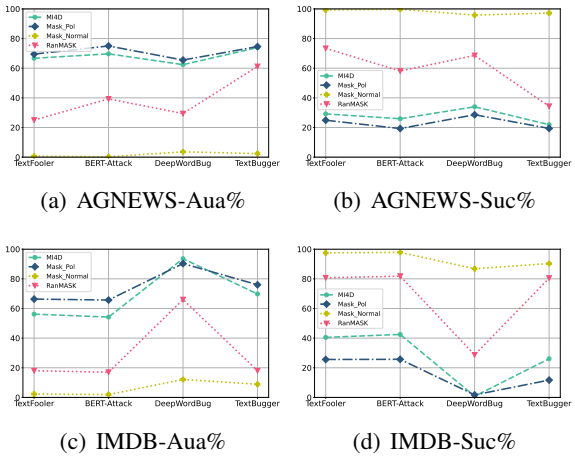
The model accuracy is evaluated by adding the *[MASK]* token in different locations, *i.e.*, either after *[CLS]* (termed *Head*), randomly (termed *Random*) across the input sequence, or at the end (termed *Tail*) . The comparison is shown in Fig 6, and the result illustrates that the proposed method achieves a similar performance regardless of the inserted *[MASK]* location.

A.3 Impact from the *[MASK]* budget

The model accuracy is also evaluated as a function of the masking budget. The comparison is shown in Fig 7, and the result illustrates that the proposed method achieves a stable performance regardless of different budgets.

A.4 Result from masking different types of tokens

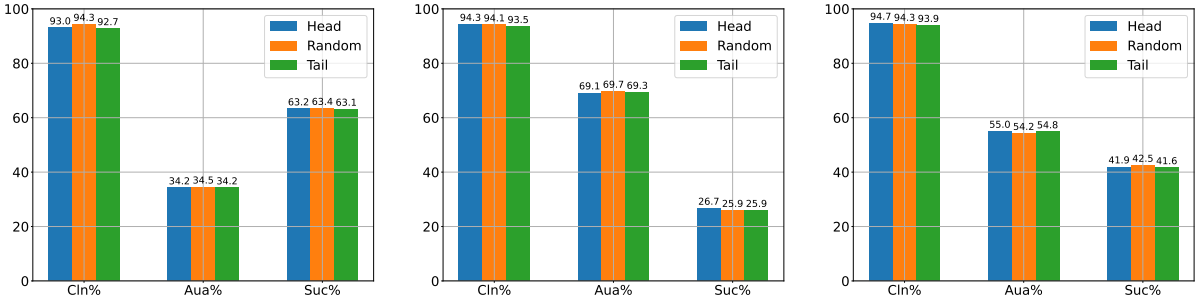
Fig. 8 shows the comparison of the defensing results with different types of tokens being masked. Clearly, masking all polluted but retaining normal tokens leads to the best performance, while masking normal tokens is the worst. The proposed MI4D achieves the competitive outcome by injecting additional `[MASK]` tokens while keeping others. The result indicates that the larger the insertion between new convex hull (after masking) with the original one, the better defensing performance.



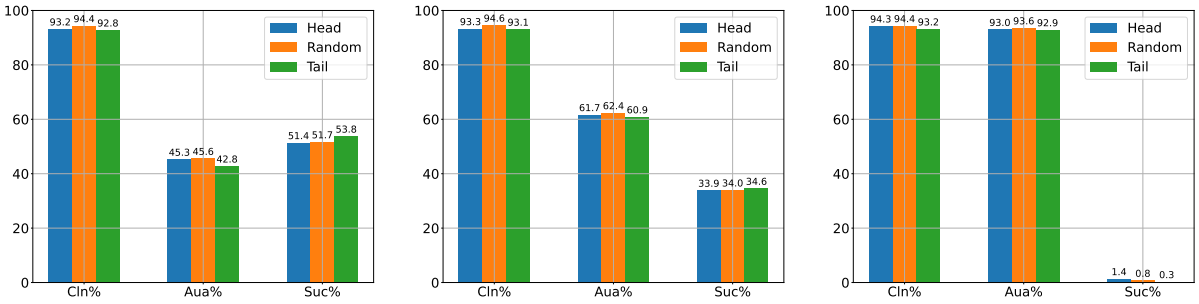
(a) AGNEWS-Aua% (b) AGNEWS-Suc%

(c) IMDB-Aua% (d) IMDB-Suc%

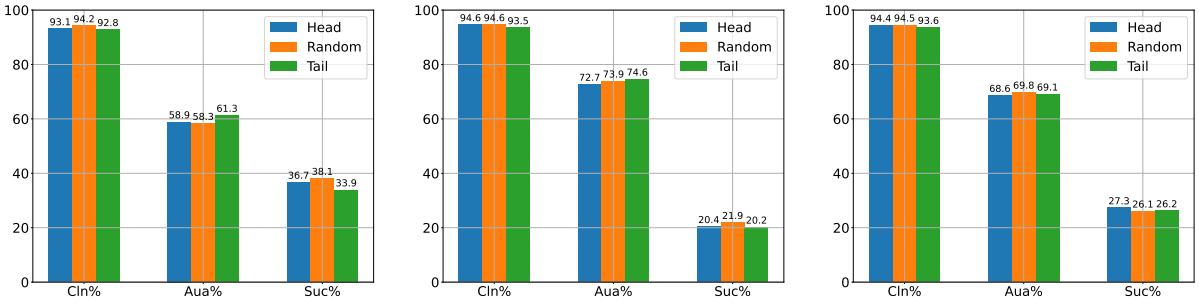
Figure 8: Comparison of the model performance against categorizes of masked tokens.



(a) SST2 (b) AGNEWS (c) IMDB

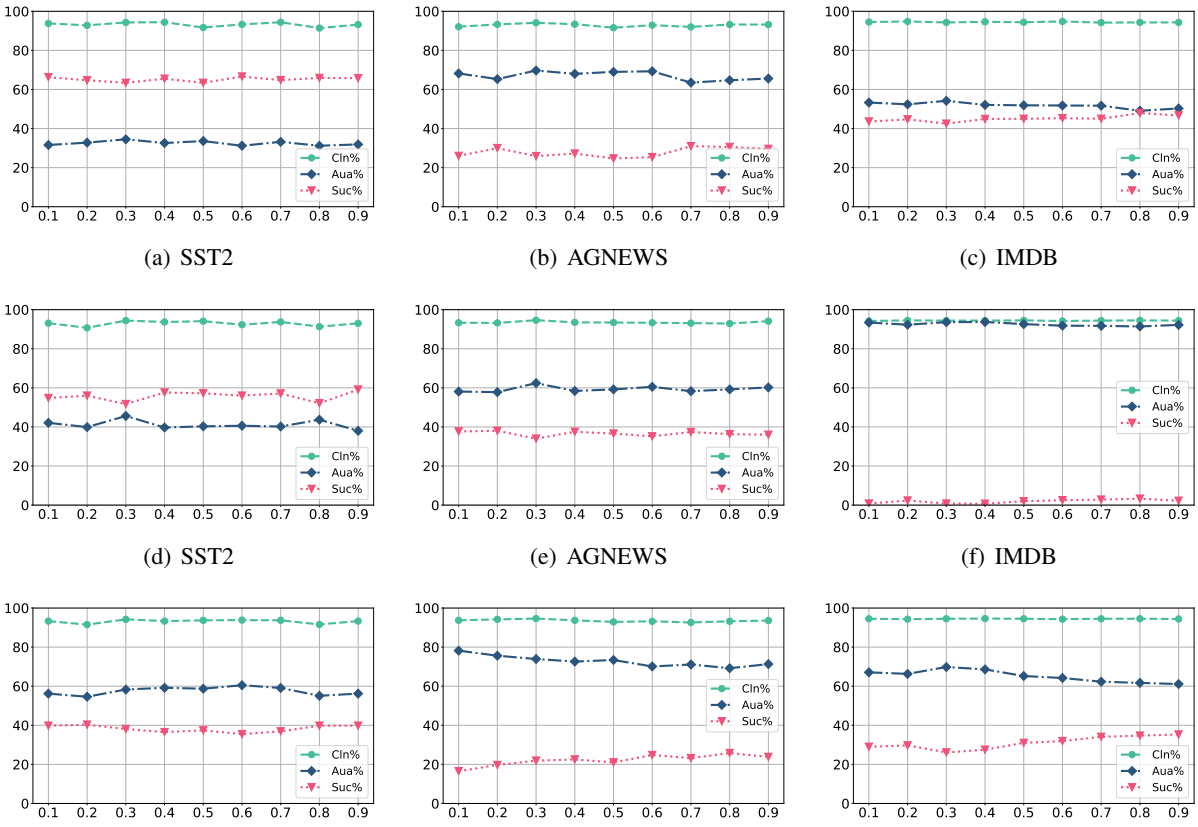


(d) SST2 (e) AGNEWS (f) IMDB



(g) SST2 (h) AGNEWS (i) IMDB

Figure 6: Comparison of the defensing performance as a function of inserting `[MASK]` after `[CLS]` (labeled as *Head*), randomly (labeled as *Random*), or at the end of the input sequence (labeled as *Tail*). Among them, (a)-(c) is for the BERT-Attack, (d)-(f) is for DeepWordBug, and (g)-(i) is for TextBugger, respectively.



(a) SST2 (b) AGNEWS (c) IMDB

(d) SST2 (e) AGNEWS (f) IMDB

(g) SST2 (h) AGNEWS (i) IMDB

Figure 7: Comparison of the defensing performance as a function of masking budget. Among them, (a)-(c) is for the BERT-Attack, (d)-(f) is for DeepWordBug, and (g)-(i) is for TextBugger, respectively.