Abstract

Adversarial attack aims to perturb input sequences and mislead a trained model for false predictions. To enhance the model robustness, defending 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.

Our aim is then to explore a robust adversarial defense 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...
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 defending 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 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
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 $[\text{MASK}]$ insertion for defending (MI4D).

3.1 $[\text{MASK}]$ insertion for defending

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 $[\text{MASK}]$ tokens at the beginning of input sequences during training and inference. Specifically, given the tokenized input sequence $x$ (i.e., $x = [\text{CLS}]x_1 \cdots x_{|x|} [\text{SEP}]$), where $x_i$ represents the $i$-th token from $x$. For the text classification task, we aim to optimize an encoder $\text{Enc}()$ and a Multilayer Perceptron (MLP) layer $f()$ to map $x$ to a desirable label $y$, i.e., $f(\text{Enc}(x)) \rightarrow y$.

Let $b_M$ be the pre-defined masking budget (or the fraction of masked tokens). Then, MI4D injects $M$ consecutive masks after $[\text{CLS}]$ within $x$ to form a masked sequence, that is, $x' = [\text{CLS}] [\text{MASK}]_1 \cdots [\text{MASK}]_M x_1 \cdots x_{|x|} [\text{SEP}]$, and $M = \lceil |x| \times b_M \rceil$.

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

3.2 Analysis

Notably, RanMASK (Zeng et al., 2021) substitutes input tokens with $[\text{MASK}]$, while MI4D injects $[\text{MASK}]$. 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 defending performance.

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

$$Y = \text{softmax}(XW_1W_2^\top X^\top)XW_3,$$

where $X \in \mathbb{R}^{|x| \times d}$ is the latent representation of $x$, $d$ is the hidden dimension, and $W_k$ ($\forall k \in [1, 3]$) are projection matrices with compatible dimensions. The property of $\text{softmax}$ dictates that each row of $Y$ (written as $y_i$) is a convex construction of $XW_3$ (written as $\bar{X}_i$), i.e., $y_i \in C(\bar{X})$, where $C(\bar{X})$ stands for the convex hull of $\bar{X}$ (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 defending method creates) with the original convex hull (the clean data forms), and the larger intersection results in the better defending performance. This leads to the following assumption.

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

$$P(\text{successful defense}) = \frac{\text{Vol}(C(X') \cap C(X_c))}{\text{Vol}(C(X_c))},$$

Figure 1: Comparison of existing adversarial defensing methods.
where $\text{Vol}(\cdot)$ is a function to estimate the volume of a geometric object, and $X_c$ is the $\epsilon$ ball centred at $X$.

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

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

For any point $x_i \in X$, it is either an anchor point or an internal point referring to $\mathcal{C}(X)$. If $S$ contains only anchor points, $\mathcal{C}(S) = \mathcal{C}(X)$ as the internal points can be “absorbed”. To see this, assume $x_1$ is an internal point, then $x_1 = \sum_{i > 1} \beta_i x_i$ and all $\beta_i$s for $i > 1$ satisfying convex condition. Then

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

Therefore, $\mathcal{C}(X) = \mathcal{C}(X_{-1})$ where $X_{-1}$ is the set of vectors after removing $x_1$. 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 $X$ 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 $S_i$ ($S_i \subseteq \mathcal{X}$, $i = 1 \sim n$). Clearly, from Lemma 1, we have $\forall i, \mathcal{C}(S_i) \subseteq \mathcal{C}(X)$, where $S_i$ is the corresponding representations in $X$ indexed by $S_i$. Equality holds only when $S_i$ contains all anchor points set in $\mathcal{C}(X)$.

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 defending rate as that of RanMASK. In other words, MI4D has at least equally good adversarial defense performance as RanMASK.

**Proof.** Let $X'(X)$ be the adversarial (original) representations in latent space, and $S_i$ be the $i$-th subset of $X'$. The successful defending probability of MI4D and RanMASK at the $i$-th run is defined as $p_m$ and $p_r$, respectively. We have

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

$$p_r = \frac{\text{Vol}(\mathcal{C}(X_{-1}) \cap \mathcal{C}(X'))}{\text{Vol}(\mathcal{C}(X))}.$$
and 

\[ p_{r_i} = \frac{\text{Vol}(C(X) \cap C(S_i))}{\text{Vol}(C(X))}. \]

For RanMASK to succeed, the successful sets have to be chosen and become majority and hence the final success probability of RanMASK

\[ p_r = P(\exists i, S_i \text{ success} \land \text{successful sets are majority}). \]

Clearly,

\[ p_r \leq \min(P(\exists i, S_i \text{ success})), \]

\[ \sum_{k \geq \lceil n/2 \rceil} B(k; n, \max_i \{p_{r_i}\}) \]

\[ \leq \min_{i} \left( \sum_{k \geq \lceil n/2 \rceil} B(k; n, \max_i \{p_{r_i}\}) \right) \]

\[ \leq p_m, \]

where \( B(k; n, p) \) 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 \( p_m \geq \max_i (p_{r_i}) \).

\[ \square \]

Overall, an illustration is shown in Fig. 2 with the convex hull of MI4D \( (C_m) \) and those of RanMASK \( (C_r) \) with two different random eliminations. The gray area shows the intersection of MI4D convex hull with the original convex hull \( C \), i.e., \( C \cap C_m \), while stripe areas are the intersections of those of RanMASK, i.e., \( C \cap C_r \). We know that \( C \cap C_m \) always contains \( C \cap C_r \). As such, the span of the convex hull after \([\text{MASK}]\) insertion is critical to retain more solution space to enhance successful defense. Additionally, we also infer that the position of inserted \([\text{MASK}]\) 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 \( \epsilon \) ball of the same \([\text{MASK}]\) token itself. These inferences were 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 into 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, \( \text{Cln}\% \) refers to the model classification accuracy on the original clean data. \( \text{Aua}\% \) is the classification accuracy under certain adversarial attacks, and higher \( \text{Aua}\% \) means better defending performance. \( \text{Suc}\% \) is defined as the number of examples successfully being fooled against the number of all attempted attacks; accordingly, lower \( \text{Suc}\% \) 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 trials) 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.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Methods</th>
<th>TextFooler</th>
<th>BERT-Attack</th>
<th>Deepwordbug</th>
<th>TextBugger</th>
</tr>
</thead>
<tbody>
<tr>
<td>SST2</td>
<td>Baseline</td>
<td>91.4</td>
<td>5.4</td>
<td>93.1</td>
<td>6.2</td>
</tr>
<tr>
<td></td>
<td>WETAR-D</td>
<td>94.3</td>
<td>31.1</td>
<td>67.5</td>
<td>94.3</td>
</tr>
<tr>
<td></td>
<td>Freecl.B++</td>
<td>93.0</td>
<td>23.6</td>
<td>74.9</td>
<td>93.0</td>
</tr>
<tr>
<td></td>
<td>IB</td>
<td>94.1</td>
<td>28.9</td>
<td>69.3</td>
<td>94.1</td>
</tr>
<tr>
<td></td>
<td>InfoBERT</td>
<td>94.0</td>
<td>19.5</td>
<td>79.3</td>
<td>94.0</td>
</tr>
<tr>
<td></td>
<td>Flooding-X</td>
<td>94.2</td>
<td>32.2</td>
<td>65.3</td>
<td>94.2</td>
</tr>
<tr>
<td></td>
<td>RanMASK</td>
<td>92.7</td>
<td>12.9</td>
<td>86.1</td>
<td>93.0</td>
</tr>
<tr>
<td></td>
<td>MI4D</td>
<td>94.3</td>
<td>36.4</td>
<td>61.4</td>
<td>94.3</td>
</tr>
</tbody>
</table>

Table 2: Comparison results of averaged Cln% (94.35%) across over five trails are shown in Table 1.

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 defending 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...
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 conducted 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 defending 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 “developed” 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-
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 defending accuracy of masking different types of tokens from the SST2 dataset (results from other datasets can be found in Appendix A.4).

![Figure 5: Comparison of the model performance against categorizes of masked tokens with SST2.](image)

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 defending 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).

### 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.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Strategy</th>
<th>TextFooler</th>
<th>BERT-Attack</th>
<th>Deepwordbug</th>
<th>TextBugger</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Chn%</td>
<td>Ana%</td>
<td>Suc%</td>
<td>Chn%</td>
<td>Ana%</td>
</tr>
<tr>
<td>SST2</td>
<td>Baseline</td>
<td>94.1</td>
<td>5.4</td>
<td>94.3</td>
<td>94.1</td>
</tr>
<tr>
<td></td>
<td>Train only</td>
<td>93.0</td>
<td>6.1</td>
<td>93.4</td>
<td>93.4</td>
</tr>
<tr>
<td></td>
<td>Test only</td>
<td>92.3</td>
<td>31.8</td>
<td>65.5</td>
<td>92.8</td>
</tr>
<tr>
<td></td>
<td>Train+Test</td>
<td>94.3</td>
<td>36.4</td>
<td>61.4</td>
<td>94.3</td>
</tr>
<tr>
<td>AGNEWS</td>
<td>Baseline</td>
<td>94.2</td>
<td>15.8</td>
<td>83.2</td>
<td>94.2</td>
</tr>
<tr>
<td></td>
<td>Train only</td>
<td>93.6</td>
<td>11.2</td>
<td>88.0</td>
<td>92.4</td>
</tr>
<tr>
<td></td>
<td>Test only</td>
<td>91.0</td>
<td>52.0</td>
<td>42.8</td>
<td>92.0</td>
</tr>
<tr>
<td></td>
<td>Train+Test</td>
<td>94.2</td>
<td>66.7</td>
<td>29.2</td>
<td>94.1</td>
</tr>
<tr>
<td>IMDB</td>
<td>Baseline</td>
<td>91.5</td>
<td>0.5</td>
<td>99.4</td>
<td>91.5</td>
</tr>
<tr>
<td></td>
<td>Train only</td>
<td>93.8</td>
<td>22.7</td>
<td>75.8</td>
<td>93.1</td>
</tr>
<tr>
<td></td>
<td>Test only</td>
<td>94.1</td>
<td>44.2</td>
<td>52.9</td>
<td>94.2</td>
</tr>
<tr>
<td></td>
<td>Train+Test</td>
<td>94.5</td>
<td>56.2</td>
<td>40.5</td>
<td>94.3</td>
</tr>
<tr>
<td>AVG</td>
<td>Baseline</td>
<td>93.3</td>
<td>7.2</td>
<td>92.3</td>
<td>93.3</td>
</tr>
<tr>
<td></td>
<td>Train only</td>
<td>93.5</td>
<td>13.3</td>
<td>85.7</td>
<td>93.0</td>
</tr>
<tr>
<td></td>
<td>Test only</td>
<td>92.5</td>
<td>42.7</td>
<td>53.7</td>
<td>93.0</td>
</tr>
<tr>
<td></td>
<td>Train+Test</td>
<td>94.3</td>
<td>53.1</td>
<td>43.7</td>
<td>94.2</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th></th>
<th>TextFooler</th>
<th>BERT-Attack</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Cln%</td>
<td>Aua%</td>
</tr>
<tr>
<td><strong>[MASK]</strong></td>
<td>94.3</td>
<td>91.1</td>
</tr>
<tr>
<td><strong>[PAD]</strong></td>
<td>94.4</td>
<td>91.1</td>
</tr>
<tr>
<td>Deepwordbug</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TextBugger</td>
<td>94.4</td>
<td>91.1</td>
</tr>
</tbody>
</table>

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


Improving robustness of language models from an information theoretic perspective. In *International Conference on Learning Representations*.


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 $5 \times 10^{-2}$; 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 defending 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 defending performance.

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