Formalizing Robustness against Character-level Perturbations for Neural Network Language Models

Zhongkui Ma[1⋆], Xinguo Feng[1⋆], Zihan Wang[1], Shuofeng Liu[1], Mengyao Ma[1], Hao Guan[1], and Mark Huasong Meng[2,3]

[1] The University of Queensland, St Lucia, QLD, Australia
[2] National University of Singapore, Singapore
[3] Institute for Infocomm Research, A*STAR, Singapore

Abstract. The remarkable success of neural networks has led to a growing demand for robustness verification and guarantee. However, the discrete nature of text data processed by language models presents challenges in measuring robustness, impeding verification efforts. To address this challenge, this work focuses on formalizing robustness specification against character-level perturbations for neural network language models. We introduce a key principle of three metrics, namely probability distribution, density, and diversity, for generalizing neural network language model perturbations and meanwhile, formulate the robustness specification against character-level adversarial text inputs. Based on the specification, we propose a novel approach to augment existing text datasets with adversarial perturbations, aiming to guide the adversarial training of language models. Experimental results demonstrate that the training with our generated text datasets can enhance the overall robustness of the language model. Our contributions advance the field of neural network verification and provide a promising approach for handling robustness challenges in neural network language models.

Keywords: Neural network · Language model · Character-level perturbations · Adversarial training · Robustness.

1 Introduction

The field of natural language processing (NLP) has been revolutionized by the rapid advancements in neural network language models, especially after the introduction of the Transformer architecture [28]. These models have demonstrated remarkable performance across a range of NLP tasks, including machine translation [28], text classification [8], sentiment analysis [29], and text generation [25]. However, their vulnerability to adversarial attacks poses a significant threat to the reliability and accuracy of NLP models.

Adversarial attacks can take many forms, ranging from minor perturbations that are imperceptible to humans to more significant modifications that can

⋆ Equal contribution

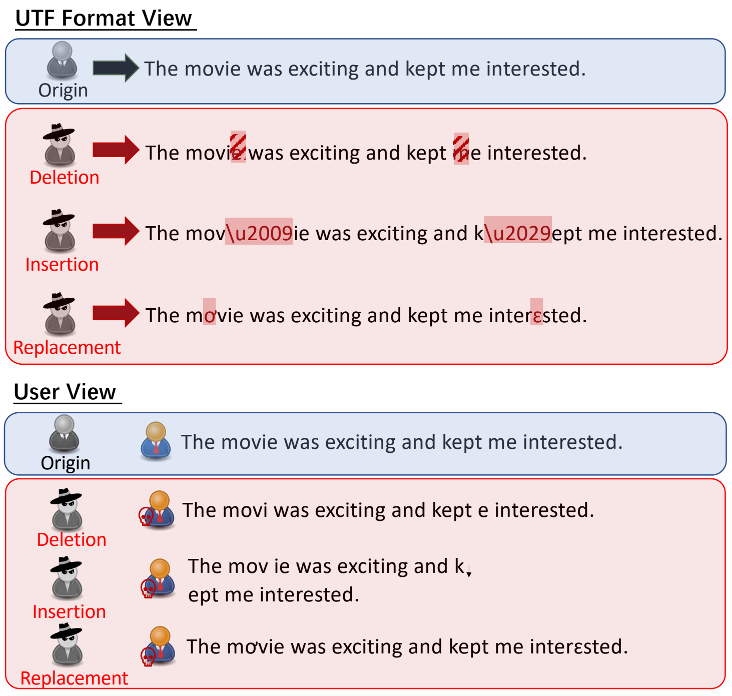
UTF Format View
Origin
The movie was exciting and kept me interested.
Deletion
The movi was exciting and kept e interested.
Insertion
The mov\u2009ie was exciting and k\u2029ept me interested.
Replacement
The mo͘vie was exciting and kept me interεsted.
User View
Origin
The movie was exciting and kept me interested.
Deletion
The movi was exciting and kept e interested.
Insertion
The mov ie was exciting and k
ept me interested.
Replacement
The mo͘vie was exciting and kept me interεsted.

Fig. 1. Examples of character-level perturbations in English text

result in incorrect or misleading outputs. This vulnerability brings challenges to the adoption of neural network language models in safe-critical domains such as clinical diagnosis, financial services, infrastructure, and cybersecurity [32]. To address these challenges, the training of a language model needs to take input perturbations into account and guarantee the generated model is resilient to adversarial attacks.

Robustness is a crucial property of language models that ensures they can produce accurate or reasonable outputs even in the presence of perturbations in the input. Formalizing the language model and its perturbations takes a step further towards verifying them to guarantee their practical implications. Given ϵ-perturbation has been widely adopted to measure local robustness for continuous inputs, such as numbers, images, and voice, there still lacks a measurement of the robustness specification for language models that takes text as input.

In this paper, we consider that natural language text is formed by characters as the atomic elements, and therefore we focus on robustness against character-level perturbations. Our aim is to formalize a unified concept for the imperceptible character-level perturbations. We consider perturbations of text input can be produced by three operations, namely replacement, deletion, and insertion, as demonstrated in Fig. 1. We apply the three operations in an adversarial context and propose four types of character-level attacks. We then introduce a set of metrics including probability distribution (P), density (d), and diversity (D), to measure the perturbations, and accordingly, provide a formal definition of the robustness property against character-level perturbations. In addition to defining the robustness property, our proposed set of metrics can also be applied to augment existing text datasets by generating adversarial samples based on benign ones, which can be used to enhance the model's robustness through adversarial training.

Our evaluation aims to investigate whether our proposed metrics, written as (P, d, D), can sufficiently define the robustness property against character-level attacks and moreover, can be used to carry out adversarial training. To this end, we apply the proposed metrics in augmenting existing text datasets with adversarial samples, and perform adversarial training of three representative neural network language models. The experimental results show that, being guided by the defined robustness specifications, our adversarial training can effectively enhance robustness while maintaining a high level of fidelity.

Contributions. Our contributions can be summarized as follows.

– We develop a canonical representation of character-level input perturbations, specifically for text and covering four types of existing attack models, with three different metrics: probability distribution, density, and diversity. Subsequently, we formalize language models and their robustness specification against character-level perturbations.
– We propose a set of perturbation generation algorithms configurable by the three metrics. We also implement a dataset augmentation tool called PdD, aiming to produce sufficient adversarial samples in addition to the benign ones in the existing datasets.
– We implement adversarial training on various typical language models using PdD. The results demonstrate that the generated perturbed datasets are beneficial for enhancing the robustness against character-level perturbations.

Paper Organization. This paper is organized as follows. In Sec. 2, we provide related works. Sec. 3 formalizes the perturbation and its metrics and robustness. A formalization for language models and character-level perturbation is given. Sec. 4 presents our experiment and evaluation using generated augmented dataset by our algorithm. We conclude in Sec. 6.

Notation. We use lowercase letters, a, b, c, p, x, y to denote variables, and bold lowercase letters, $\boldsymbol{x}$, $\boldsymbol{y}$, to denote vectors or sequences. Sets, D, P, U, Σ, are denoted by uppercase letters. f denotes a function or model.

2 Related Works

The current work draws inspiration from existing research on adversarial attacks and the robustness of deep neural networks, with a particular focus on NLP tasks.

Adversarial manipulations in NLP tasks. Adversarial attacks in the NLP domain aim to manipulate systems by altering the input text, resulting in erroneous decision-making [9]. The perturbation of text inputs without prior knowledge can be achieved by utilizing special character sets, such as homoglyphs or invisible characters, as perturbed candidates. Boucher et al. [5] find that the injection of a single imperceptible encoding, named bad character, can lead to a remarkable decline in the targeted model's performance, and when it comes to three injections, most models can be functionally broken. Boucher et al. [4, 5] also explore a pile of adversarial attacks on NLP tasks without making any human-perceptible visual modification to inputs, and generate perturbations by uncommon encoded representations to control results across search engines and large language models (LLMs).

Performing an automatic search for adversarial samples around a given input typically requires access to gradient information from the model. This process demands additional expertise and skills, as it involves leveraging the gradient information to iteratively modify the input and search for potential adversarial examples. Behjati et al. [2] propose a gradient projection based approach to generate data-independent adversarial sequences, which can fool the classifiers into getting incorrect predictions effectively. They demonstrate that even adding one word of the adversarial sequences into the input text can downgrade the classification accuracy dramatically. Garg and Ramakrishnan [11] present BAE, a black-box attack that uses BERT masked model to hide some words of the original text with `<mask>` and then using BERT-MLM to predict `<mask>` by insertion or replacement. The usage of BAE can not only undermine the accuracy of predictions but also strengthen the grammatical and semantic coherence of the adversarial text. Morris et al. [22] propose an open-source framework called TextAttack, which implements existing 16 adversarial attacks on various datasets and NLP models. Song et al. [26] develop adversarial attacks that are more like human-readable English by natural triggers and show that using such triggers together with their proposed gradient-based search can degrade the accuracy of classification tasks.

Robustness of Neural Networks. Robustness typically refers to how sensitive a model is to perturbations or noise in the input data [21, 27, 31]. Specifically, a model is considered robust when it is able to maintain the stability and consistency of its outputs as the input data have been changed. Several studies have been done to improve the robustness of deep learning models. Gao et al. [10] propose a mutation-based fuzzing technique to augment the training data of deep neural networks, which is capable of improving the accuracy and robustness and meanwhile, saving the training time. Zhang et al. [33] explain adversarial robustness via the sensitivity of neurons and then further analyze robustness by stabilizing the behaviors of the sensitive neurons. They reduce neuron sensitivity to improve adversarial robustness successfully. Boopathy et al. [3] propose CNN-Cert, a general framework for certifying robustness on convolutional neural networks (CNNs), which speeds up more than 11 times compared with other certification algorithms and copes with various CNN architectures.

Although there are many additional approaches to perform adversarial attacks or improve robustness [1, 7, 13, 14, 16, 20, 24, 17, 30], finding a proper representation of adversarial perturbations when it comes to NLP tasks is still an open question, due to its discrete features of the input data. Therefore, to the best of our knowledge, we are the first to explore this and provide solutions through the lens of formalization.

3 Formalization

In this section, we formulate a comprehensive framework for applying input perturbation techniques that can be extended to natural language settings. Our primary goal is to evaluate the robustness of natural language models within this framework. Throughout our analysis, we introduce three fundamental character-level operations and three metrics, namely *probability distribution*, *density* and *diversity*, to effectively control the level of input perturbation. These metrics serve as essential tools for quantifying and managing the extent of perturbation applied to the input.

3.1 Formalizing Perturbations to General Inputs

Our initial step involves considering the neighborhood of an input, as our primary objective is to formalize the local robustness of a model. Within this framework, a perturbed input is considered to be within the neighborhood of the original input vector, denoted as $\boldsymbol{x}_0 \in X$. The neighborhood, denoted as $U_{\boldsymbol{x}_0}$, is a subset of the space of all possible inputs, represented as X.

To effectively quantify the variance between the original input and its perturbed counterparts, we introduce three key metrics: probability distribution, density, and diversity. These metrics serve as valuable tools for measuring and evaluating the differences among various input samples within the neighborhood.

Probability Distribution. A discrete probability distribution is utilized to depict the probability of perturbing each item within the input vector. This distribution provides a representation of the likelihood associated with selecting each item for perturbation.

Definition 1. *The probability distribution P of a perturbation for a given input vector $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$ refers to the distribution that governs the probability $P(i)$ of i-th element x_i $(1 \leq i \leq n)$ in the input vector being perturbed.*

Density. The density parameter characterizes the count of perturbed items within a given vector. It is important to emphasize that we impose a constraint where only one element can be perturbed at a time during a single perturbation. Consequently, a perturbation to the input may involve altering multiple distinct elements, but each element is perturbed individually, ensuring that only one element is modified at a time.

Definition 2. *The density d $(0 \leq d \leq 1)$ of perturbation refers to the percentage of perturbed elements in the given input vector.*

Diversity. The diversity of perturbation pertains to the collection of possible candidate characters that can be employed to perturb each original element in the input vector. This encompasses all the available choices for characters that can replace or modify the original element. The diversity metric provides a comprehensive view of the range of alternative characters that can be utilized for perturbation, offering insights into the various options for altering each element in the input vector. Notably, special character sets, such as homoglyphs or typos, can be utilized as candidate sets to expand the range of perturbation options.

Definition 3. *The diversity* $D = \{(x_i, D_i) | 1 \le i \le n\}$ *is a set of sets that contains all pairs* (x_i, D_i)*, where* $D_i = \{x_i', x_i'', \cdots\}$ *is the set of all possible candidate elements that can be used to perturb* x_i*.*

Example 1: (ϵ-perturbation) When considering ϵ-perturbation, which is a common setting in verification, all elements of the input vector are available for being perturbed, resulting in a density of 1. In this case, the distribution of perturbation can be considered as any distribution. ϵ-perturbation define each input item x_i has a diversity that is a interval $[x_i - \epsilon, x_i + \epsilon]$ containing all values within a distance of ϵ to the input.
Example 2: (Character-level perturbation) When adding character-level perturbation to a sentence, the sentence is represented by a character vector, and the distribution describes the probability of each character being perturbed, while the density describes how many characters can be perturbed. The diversity can be a discrete character set to describe all possible replacements.

Therefore, we define the perturbation for a text input $\boldsymbol{x}$ as $U(\boldsymbol{x}; P, d, D)$. Generating perturbations to an input is a process described in Algorithm.1. The algorithm takes a vector $\boldsymbol{x}$, the probability distribution P, the density d, and the diversity parameter D as inputs and output a perturbed vector. One element of $\boldsymbol{x}$ is perturbed in one loop until the perturbed elements achieve the targeted density. When choosing a perturbed element, the distribution P is used. Choosing the perturbed candidate follows a uniform distribution.

3.2 Formalizing Language Models

In this section, our attention is directed towards formulating a specific framework for language models in the context of machine learning. We enumerate three fundamental character-level operations employed in the perturbation process.Additionally, We provide a formalized definition of robustness within this framework.

Language Model. Let L be a formal language over an alphabet Σ, which is a subset of the set of words Σ^*[4]. We begin by defining a language model in machine learning, denoted as f.

[4] We use Kleene star to denote the concatenation of words

Algorithm 1: PdD($\boldsymbol{x}$, P, d, D)

```
// Get the number of perturbed elements
1  $n \leftarrow$ floor($d * len(\boldsymbol{x})$);
2  $i \leftarrow 0$;
3  while $i < n$ do
       // Choose the perturbed element
4      while $TRUE$ do
5          $j \sim P$;
6          if $\boldsymbol{x}[j]$ is not perturbed then break;
       // Randomly choose a perturbed candidate
7      $x' \leftarrow$ getPtbCandidate($D[\boldsymbol{x}[j]]$);
       // Perturb the specified element
8      $\boldsymbol{x}[j] = x'$;
9      $i \leftarrow i + 1$;
10 return $\boldsymbol{x}$;
```

Definition 4. *(Language Model) A language model f is a function that takes a sequence of words $\boldsymbol{x} \in \Sigma^*$ as input and outputs a sequence of words $\boldsymbol{y} \in \Sigma^*$, where Σ^* is its finite word set.*

In practice, a language model typically uses a token-level encoding to generate a token embedding, which is then taken as a part of the model. Due to the finite memory of computers, a language model always has its own finite token set Σ^*. In cases where a token is not included in Σ^*, it is represented by a special tag/token `[UNK]`, denoting that it is unknown. For text classification models, the output is typically binary and can be regarded as numbers in natural language. In this paper, to ease the understanding, we interchangeably use *token* and *word* to represent the basic input of a language model.

Robustness. Robustness is a critical property for neural networks, as it ensures that the network can produce accurate outputs even in the presence of perturbations. The definition highlights the importance of ensuring that the output remains within a predefined set, indicating that the network is capable of handling different types of perturbations without compromising its accuracy.

Definition 5. *Robustness is the property that, given a language model f and a input $\boldsymbol{x}_0$ and its perturbed values set $U_{\boldsymbol{x}_0}$, the resulting output set $f(U_{\boldsymbol{x}_0})$ satisfies being a subset of the predefined set $U_{\boldsymbol{y}_0} \subseteq \Sigma^*$, i.e.*

$$\forall x' \in U_{\boldsymbol{x}_o}, y' = f(x') \implies y' \in U_{\boldsymbol{y}_0}$$

For a classification model, it is highly desirable that the output label for a perturbed input remains consistent with the label for the original input, i.e., $U_{\boldsymbol{y}_0} = \{\boldsymbol{y}_0\}$. In general cases, we aim to ensure that the output for a perturbed input remains within a predefined set $U_{\boldsymbol{y}_0}$, i.e. $f(U_{\boldsymbol{x}_0}) \subseteq U_{\boldsymbol{y}_0}$. Note that it is not necessary for $U_{\boldsymbol{x}0} \subseteq \Sigma^*$ when considering the model f. However, it is always necessary for $U_{\boldsymbol{y}_0} \subseteq \Sigma^*$ since a reasonable output under perturbed inputs is what robustness requires and expects.

3.3 Character-level Perturbation.

We note that a sequence of words can be represented as a string or a sequence of characters. We use `[EMP]` to denote an empty character. For example, a sequence of words $\boldsymbol{x} = (x_1, x_2, \cdots, x_n)$, where x_i $(1 \leq i \leq n)$ is a word and $x_i = a_i b_i c_i \cdots$, which is composed of a finite set of characters a_i, b_i, c_i, etc., concatenated to form the word. Therefore, $\boldsymbol{x} = a_1 b_1 c_1 \cdots a_2 b_2 c_2 \cdots a_n b_n c_n \cdots$. In the following discussion, we will focus on the character representation of a sequence of words and take it as a sequence of characters.

Definition 6. *(Character-level Perturbation) Given an input $\boldsymbol{x} \in \Sigma^*$ for a language model L, a character-level perturbation $\boldsymbol{x}' \in \tilde{\Sigma}$ $(\tilde{\Sigma}^* \supseteq \Sigma^*)$ is another sequence of words, whose words have several characters that differ from the corresponding words in $\boldsymbol{x}$. Let $\boldsymbol{x} = a_1 b_1 c_1 \cdots a_2 b_2 c_2 \cdots a_n b_n c_n \cdots$ $(a_i, b_i, c_i \in \Sigma)$, and let $\boldsymbol{x}' = a_1' b_1' c_1' \cdots a_2' b_2' c_2' \cdots a_n' b_n' c_n' \cdots$ $(a_i', b_i', c_i' \in \tilde{\Sigma}$ and $\tilde{\Sigma} \supseteq \Sigma)$, then $\boldsymbol{x}'$ is almost the same to $\boldsymbol{x}$.*

The perturbed words need not be elements of the original word set Σ^*. Furthermore, certain types of perturbation may use characters outside of the original alphabet Σ. Therefore, for a given type of perturbation, it is necessary to have an alphabet $\tilde{\Sigma} \supseteq \Sigma$ and a word set $\tilde{\Sigma}^* \supseteq \Sigma^*$.

In the following discussion, we explore three fundamental character-level operations that can be utilized to generate perturbations: replacement, deletion, and insertion. An illustrative example of generating character-level perturbations is provided in Figure 1.

Replacement. This operation serves as a general case encompassing all other operations. It is inherently implied in our perturbation definition. For each element x_i in the input vector $\boldsymbol{x}$, we define a finite discrete set of candidates D_i. The set D_i comprises candidate characters, denoted as c_{ij}, where $1 \leq j \leq k$ and k is an integer. Each c_{ij} represents a possible substitution from the candidate set D_i.

Other operations can be derived from the replacement operation. We illustrate two significant cases: deletion and insertion.

Deletion. For deletion, we set $(x_i, D_i) = (x_i, \texttt{[EMP]})$, where `[EMP]` represents an empty character. This operation effectively removes the character x_i from the input.

Insertion. For insertion, we set $(x_i, D_i) = (x_i, x_i c_{ij} | 1 \leq j \leq k, k \in \mathbb{Z})$, where c_{ij} is defined as in the replacement operation. Here, $x_i c_{ij}$ represents the concatenation of the original character x_i and the inserted character c_{ij}.

It is also possible to define operations that affect subsequent characters in different sizes. Moreover, for more complex operations such as transposition or swapping two characters, a well-defined probability distribution of perturbation

Table 1. Experimental models and datasets

Model	**Dataset**	**Task**	**#Class**
BERT	Rotten Tomatoes	Sentiment Analysis	2
RoBERTa	SNIL	Natural Language Inference	3
ALBERT	E-commerce	Text Classification	4

is required. Different operations can lead to various types of adversarial perturbations. For instance, insertion allows for the generation of diverse perturbations such as inserting invisible characters, replacing with homoglyphs, or introducing typos.

4 Experiments

In this section, we assess the effectiveness of our perturbation metrics on various language models trained with different augmented datasets. All models are trained using NVIDIA's A6000 GPUs.

4.1 Experiment Setup

Models and Datasets. We conduct our experiments on three widely used language models, including BERT [8], RoBERTa [18], and ALBERT [15]. These models are within the Transformer Encoder family and are usually used for classification tasks. We use several benchmark text classification datasets, including Rotten Tomatoes [23], Stanford Natural Language Inference (SNLI) [6], and E-commerce [12], to evaluate the performance of the models. Table. 1 displays the models and datasets evaluated in this study.

Perturbation Settings and Datasets Generation. To create augmented datasets comprising perturbed samples, we employ our formalized perturbation metrics including probability distribution, density, and diversity to define the perturbed input set $U(\boldsymbol{x}_0; P, d, D)$ for each input sample $\boldsymbol{x}_0$. For each input sample, we generate 10 perturbed versions using a single perturbation setting. Consequently, an augmented dataset produced using one perturbation setting is 10 times larger than the original dataset. Using different combinations of these metrics as outlined in Table 2, we separately apply each character-level perturbation method to all samples in a dataset. We then utilize all resulting perturbed samples to evaluate the robustness of the language models.

Within the context of our experimental design, we consider two probability distributions, namely the uniform distribution and the normal distribution, to deploy character-level perturbations in benign input samples. Specifically for normal distribution, we select $\mu = 0.5 \times L$ and $\sigma^2 = 0.25 \times L$, where L represents the length of the input sequence. In our experiments, we examine two densities of 0.05 and 0.2. These densities and probability distributions are employed in combination with the specific perturbation generation method.

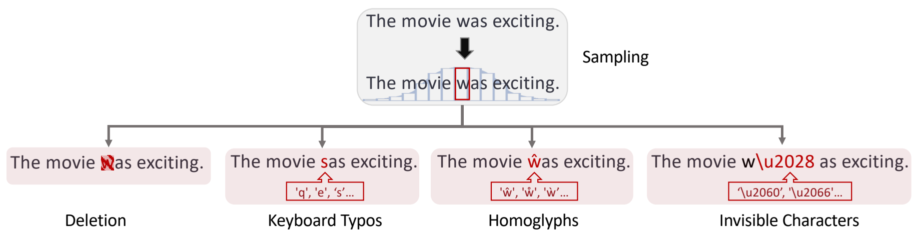

Fig. 2. Examples of four practical attacks in English text

Table 2. Perturbation settings used in our evaluation

Metrics	Description	Setting Options
P	Probability Distribution	uniform distribution, normal distribution
d	Density	0.05, 0.2
D	Diversity	deletion, keyboard typos, homoglyphs, invisible characters

In accordance with the definition presented in Sec. 3, the perturbations employed in our experiments are practically elaborated as follows. **Deletion** refers to the act of removing a character from the given sentence. **Keyboard typos** imitate the act of mistakenly pressing adjacent keys, whereby we consider up to 8 neighboring keys as potential candidates for a single key. **Homoglyphs**, by definition, are characters that bear a resemblance to a specific character. In our approach, we carefully select 5 homoglyphs that can serve as potential replacements for a single character in the input sentence. **Invisible characters** encompass operational characters that are not detectable by the human eye, and we choose from a pool of 48 invisible characters to insert them behind the selected character. Fig. 2 demonstrates some perturbation instances generated by the four standalone attack methods. In addition to the four standalone methods, a **mixed perturbations** method is also adopted to further assess the effectiveness of the proposed adversarial training approach. It combines the aforementioned four types of perturbation and randomly applies them to the input sequence.

Implementation Details. In our experiments, we utilize three different pretrained large language model architectures and train them on both original training datasets and augmented training datasets. This training process results in two separate models for each model architecture, namely a clean model $\mathbf{M_{clean}}$, and an adversarially trained model $\mathbf{M_{adv}}$.

To maintain consistency, we randomly choose an equivalent amount of data from the SNLI and E-commerce datasets, which are larger in size compared to the Rotten Tomatoes dataset. This selection results in 9,000 training samples and 2,000 testing samples for the original datasets. For each augmented dataset under different settings, we obtain 99,000 training samples and 22,000 testing samples, after the insertion of perturbed samples.

During the training phase, all models utilize a batch size of 16. We employ the AdamW optimizer [19] with a fixed learning rate of 2×10^{-5}. To determine the best models, we incorporate early stopping with a patience of five. On average, the training process spans approximately seven epochs.

4.2 Evaluation

Models Performance on Perturbed Datasets. We assess the performance of both the adversarially trained model and the clean model on each augmented dataset. As shown in Table 3, we utilize different combinations of perturbation metrics to demonstrate that each character-level perturbation attack can bring a noticeable impact on the language models' performance, providing evidence for its effectiveness in enhancing the overall model robustness. The macro F1 score is used as the evaluation metric to gauge the performance of the models. The results demonstrate that the adversarially trained models consistently outperform clean models when tested on datasets containing adversarial samples.

We have noted that the impact of each perturbation type varies with its density and distribution characteristics. Perturbations with higher density perturbations exhibit a greater influence on performance. In terms of the probability distribution, our observations indicate that the uniform distribution has a more pronounced impact on performance. This is attributed to the fact that the perturbations are evenly distributed across the entire input sentence, causing the language model encounters greater challenges in encoding the sentence.

Furthermore, we can also observe that the impact of perturbations is closely connected to the complexity of the tasks and the characteristics of the datasets. For instance, our introduced perturbation demonstrates a notably greater impact on the Natural Language Inference (NLI) task compared to the other two tasks. This might be because the NLI task tends to consider every word and phrase in a sentence to construct the fine-grained context, which is known as one of the most challenging fields of NLP. On the other hand, in Text Classification scenarios, the model usually takes a few keywords into account to classify the review's category, resulting in a less pronounced impact observed. Overall, our results consistently demonstrate a significant improvement in the model's robustness against all introduced character-level perturbations, regardless of task complexity and dataset characteristics.

Models Performance on Original Datasets. To examine the capacity of the models to maintain their fidelity for the original tasks, we also evaluate the performance of the adversarially trained models on their original datasets and compare them to the performance of the corresponding clean models on the same testing sets. As shown in Table 4, the adversarially trained models show competitive performance compared to the clean models. The adversarially

Table 3. Model performance (F1-scores) on augmented datasets, presented as tuples of the clean models ($\mathbf{M_{clean}}$) and adversarially trained models ($\mathbf{M_{adv}}$). The improvement of model performance is displayed in **bold** font.

Deletion Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.5044	**0.7499**	0.4060	**0.6769**	0.8045	**0.9443**
	0.05	0.7873	**0.8358**	0.7133	**0.8027**	0.9491	**0.9566**
Normal	0.2	0.5498	**0.7505**	0.4426	**0.6788**	0.8751	**0.9466**
	0.05	0.7901	**0.8329**	0.7241	**0.8074**	0.9533	**0.9612**
Keyboard Typos Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.4963	**0.7651**	0.3625	**0.6918**	0.8095	**0.9436**
	0.05	0.7911	**0.8463**	0.7161	**0.8028**	0.9470	**0.9613**
Normal	0.2	0.5529	**0.7691**	0.3891	**0.6877**	0.8819	**0.9491**
	0.05	0.7966	**0.8471**	0.7214	**0.8058**	0.9515	**0.9627**
Homoglyphs Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.7588	**0.8247**	0.2879	**0.7142**	0.9407	**0.9560**
	0.05	0.8404	**0.8547**	0.6799	**0.8213**	0.9588	**0.9642**
Normal	0.2	0.7727	**0.8259**	0.2978	**0.7061**	0.9485	**0.9588**
	0.05	0.8466	**0.8519**	0.6883	**0.8184**	0.9600	**0.9637**
Invisible Characters Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.5336	**0.7712**	0.3630	**0.6925**	0.8380	**0.9434**
	0.05	0.7978	**0.8411**	0.7099	**0.7999**	0.9501	**0.9618**
Normal	0.2	0.5812	**0.7740**	0.3995	**0.6821**	0.8885	**0.9527**
	0.05	0.8051	**0.8442**	0.7176	**0.8063**	0.9542	**0.9612**
Mixed Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.5534	**0.7630**	0.3370	**0.6750**	0.8636	**0.9477**
	0.05	0.8080	**0.8416**	0.7001	**0.8035**	0.9526	**0.9613**
Normal	0.2	0.6013	**0.7667**	0.3634	**0.6742**	0.9051	**0.9529**
	0.05	0.8128	**0.8387**	0.7097	**0.8101**	0.9563	**0.9594**

SA: Sentiment Analysis **NLI**: Natural Language Inference **TC**: Text Classification

Table 4. Model performance (F1-scores) on original datasets, presented as tuples of the clean models ($\mathbf{M_{clean}}$) and adversarially trained models ($\mathbf{M_{adv}}$). The improvement of model performance is displayed in **bold** font.

Deletion Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.8625	0.8611	0.8686	0.8283	0.9636	0.9599
	0.05		0.8611		0.8419		0.9620
Normal	0.2		0.8424		0.8394		0.9609
	0.05		0.8574		0.8571		**0.9644**
Keyboard Typos Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.8625	**0.8672**	0.8686	0.8149	0.9636	0.9624
	0.05		**0.8653**		0.8516		0.9630
Normal	0.2		0.8621		0.8277		0.9624
	0.05		**0.8672**		0.8543		**0.9657**
Homoglyphs Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.8625	**0.8638**	0.8686	0.8299	0.9636	0.9599
	0.05		**0.8639**		0.8568		**0.9669**
Normal	0.2		0.8601		0.8170		**0.9639**
	0.05		0.8602		0.8571		**0.9648**
Invisible Characters Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.8625	**0.8681**	0.8686	0.8554	0.9636	0.9624
	0.05		**0.8658**		0.8391		**0.9644**
Normal	0.2		**0.8630**		0.8393		**0.9646**
	0.05		**0.8634**		0.8450		0.9634
Mixed Perturbation							
Probability Distribution	**Density**	**BERT(SA)**		**RoBERTa(NLI)**		**ALBERT(TC)**	
		$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$	$\mathbf{M_{clean}}$	$\mathbf{M_{adv}}$
Uniform	0.2	0.8625	0.8564	0.8686	0.8134	0.9636	0.9627
	0.05		0.8600		0.8488		**0.9637**
Normal	0.2		0.8508		0.8510		**0.9655**
	0.05		0.8615		0.8593		**0.9595**

SA: Sentiment Analysis **NLI**: Natural Language Inference **TC**: Text Classification trained models even outperform the clean models in a number of settings (19 out of 40). The experimental results show that our adversarial training can well preserve the model fidelity for the original tasks. These results provide evidence for the feasibility of our proposed approach in training language models to enhance robustness while preserving its original utility.

5 Discussion

This study focuses on the formalization of perturbation at the natural language level, treating it as a character sequence. Our work contributes a novel endeavor by establishing a unified definition encompassing all types of inputs for neural networks. However, to comprehensively evaluate the robustness of these models, it is imperative to develop additional metrics that effectively capture the distance or dissimilarity between the original item and its perturbed counterpart. Such metrics are essential for quantifying the diversity and construction of the candidate set. Although our study provides an initial framework, further refinement is expected in this aspect.

Our experiment results demonstrate that our approach can not only significantly enhance the robustness, but also retain a high level of fidelity of the models. However, we only experiment with models of classification tasks. The robustness of models of generative tasks such as Machine Translation and Text Summarization can be further investigated.

We also remark that the robustness of neural network language models can be extended to the word level. Defining metrics of word-level robustness presents unique challenges as it entails considerations of semantic meaning and grammar and therefore, desires future efforts from the research community.

6 Conclusion

This paper introduces a generalized formalization of perturbed inputs for natural language models, offering a crucial step towards testing and verifying their robustness. We specifically focus on character-level perturbations, outlining the basic operations of replacement, deletion, and insertion. By controlling the perturbation process through principles of probability distribution, density, and diversity, we can generate different levels and types of character-level perturbations using a clean dataset.

Our approach demonstrates significant improvements in robustness against specific perturbations when training a network on a dataset perturbed by our method compared to using only clean data. Augmenting existing text datasets with adversarial perturbations, guided by our proposed approach, leads to notable enhancements in overall model robustness.

Overall, this work contributes novel insights and techniques, advancing the measurement and assurance of language model robustness. Given the critical importance of reliability and accuracy in language models, our approach holds great potential for further advancements in this area.

References

1. Bai, T., Luo, J., Zhao, J., Wen, B., Wang, Q.: Recent advances in adversarial training for adversarial robustness. arXiv preprint arXiv:2102.01356 (2021)
2. Behjati, M., Moosavi-Dezfooli, S.M., Baghshah, M.S., Frossard, P.: Universal adversarial attacks on text classifiers. In: ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 7345–7349 (2019). https://doi.org/10.1109/ICASSP.2019.8682430
3. Boopathy, A., Weng, T.W., Chen, P.Y., Liu, S., Daniel, L.: Cnn-cert: An efficient framework for certifying robustness of convolutional neural networks. In: Proceedings of the AAAI Conference on Artificial Intelligence (2019)
4. Boucher, N., Pajola, L., Shumailov, I., Anderson, R., Conti, M.: Boosting big brother: Attacking search engines with encodings. arXiv preprint arXiv:2304.14031 (2023)
5. Boucher, N., Shumailov, I., Anderson, R., Papernot, N.: Bad characters: Imperceptible nlp attacks. In: 2022 IEEE Symposium on Security and Privacy (SP). pp. 1987–2004. IEEE (2022)
6. Bowman, S., Angeli, G., Potts, C., Manning, C.D.: A large annotated corpus for learning natural language inference. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 632–642 (2015)
7. Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., Mukhopadhyay, D.: A survey on adversarial attacks and defences. CAAI Transactions on Intelligence Technology **6**(1), 25–45 (2021)
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
9. Eger, S., Şahin, G.G., Rücklé, A., Lee, J.U., Schulz, C., Mesgar, M., Swarnkar, K., Simpson, E., Gurevych, I.: Text processing like humans do: Visually attacking and shielding nlp systems. arXiv preprint arXiv:1903.11508 (2019)
10. Gao, X., Saha, R.K., Prasad, M.R., Roychoudhury, A.: Fuzz testing based data augmentation to improve robustness of deep neural networks. In: Proceedings of the acm/ieee 42nd international conference on software engineering. pp. 1147–1158 (2020)
11. Garg, S., Ramakrishnan, G.: Bae: Bert-based adversarial examples for text classification. arXiv preprint arXiv:2004.01970 (2020)
12. Gautam: E commerce text dataset. https://zenodo.org/record/3355823#.ZF99xy8Rq-o (2019), accessed: 2023-05-12
13. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019)
14. Hu, P., Wang, Z., Sun, R., Wang, H., Xue, M.: M^{4}i: Multi-modal models membership inference. In: Advances in Neural Information Processing Systems. vol. 35, pp. 1867–1882 (2022)
15. Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., Soricut, R.: Albert: A lite bert for self-supervised learning of language representations. arXiv preprint arXiv:1909.11942 (2019)
16. Li, Y., Min, M.R., Lee, T., Yu, W., Kruus, E., Wang, W., Hsieh, C.J.: Towards robustness of deep neural networks via regularization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 7496–7505 (2021)

17. Liu, S., Lei, P., Koji, K.: Lstm based hybrid method for basin water level prediction by using precipitation data. Journal of Advanced Simulation in Science and Engineering **8**(1), 40–52 (2021)
18. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
19. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017)
20. Ma, M., Zhang, Y., Arachchige, P.C.M., Zhang, L.Y., Baruwal Chhetri, M., Bai, G.: Loden: Making every client in federated learning a defender against the poisoning membership inference attacks. In: 18th ACM ASIA Conference on Computer and Communications Security ASIACCS 2023'. ACM (2023)
21. Meng, M.H., Bai, G., Teo, S.G., Hou, Z., Xiao, Y., Lin, Y., Dong, J.S.: Adversarial robustness of deep neural networks: A survey from a formal verification perspective. IEEE Transactions on Dependable and Secure Computing (2022)
22. Morris, J.X., Lifland, E., Yoo, J.Y., Grigsby, J., Jin, D., Qi, Y.: Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. arXiv preprint arXiv:2005.05909 (2020)
23. Pang, B., Lee, L.: Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In: Proceedings of the ACL (2005)
24. Qiu, S., Liu, Q., Zhou, S., Wu, C.: Review of artificial intelligence adversarial attack and defense technologies. Applied Sciences **9**(5), 909 (2019)
25. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
26. Song, L., Yu, X., Peng, H.T., Narasimhan, K.: Universal adversarial attacks with natural triggers for text classification. arXiv preprint arXiv:2005.00174 (2020)
27. Subbaswamy, A., Adams, R., Saria, S.: Evaluating model robustness and stability to dataset shift. In: International Conference on Artificial Intelligence and Statistics. pp. 2611–2619. PMLR (2021)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
29. Wang, Y., Huang, M., Zhu, X., Zhao, L.: Attention-based lstm for aspect-level sentiment classification. In: Proceedings of the 2016 conference on empirical methods in natural language processing. pp. 606–615 (2016)
30. Wang, Z., Guo, H., Zhang, Z., Liu, W., Qin, Z., Ren, K.: Feature importance-aware transferable adversarial attacks. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 7639–7648 (2021)
31. Wang, Z., Byrnes, O., Wang, H., Sun, R., Ma, C., Chen, H., Wu, Q., Xue, M.: Data hiding with deep learning: A survey unifying digital watermarking and steganography. IEEE Transactions on Computational Social Systems pp. 1–15 (2023). https://doi.org/10.1109/TCSS.2023.3268950
32. Waqas, A., Farooq, H., Bouaynaya, N.C., Rasool, G.: Exploring robust architectures for deep artificial neural networks. Communications Engineering **1**(1), 46 (2022)
33. Zhang, C., Liu, A., Liu, X., Xu, Y., Yu, H., Ma, Y., Li, T.: Interpreting and improving adversarial robustness of deep neural networks with neuron sensitivity. IEEE Transactions on Image Processing **30**, 1291–1304 (2020)