Defense against Synonym Substitution-based Adversarial Attacks via Dirichlet Neighborhood Ensemble

Yi Zhou[1,2], **Xiaoqing Zheng***[1,2], **Cho-Jui Hsieh**[3], **Kai-Wei Chang**[3], **Xuanjing Huang**[1,2]
[1]School of Computer Science, Fudan University, Shanghai, China
[2]Shanghai Key Laboratory of Intelligent Information Processing
[3]Department of Computer Science, University of California, Los Angeles, USA
{yizhou17, zhengxq}@fudan.edu.cn, chohsieh@cs.ucla.edu,
kwchang@cs.ucla.edu, xjhuang@fudan.edu.cn

Abstract

Although deep neural networks have achieved prominent performance on many NLP tasks, they are vulnerable to adversarial examples. We propose Dirichlet Neighborhood Ensemble (DNE), a randomized method for training a robust model to defense synonym substitution-based attacks. During training, DNE forms virtual sentences by sampling embedding vectors for each word in an input sentence from a convex hull spanned by the word and its synonyms, and it augments them with the training data. In such a way, the model is robust to adversarial attacks while maintaining the performance on the original clean data. DNE is agnostic to the network architectures and scales to large models (e.g., BERT) for NLP applications. Through extensive experimentation, we demonstrate that our method consistently outperforms recently proposed defense methods by a significant margin across different network architectures and multiple data sets.

1 Introduction

Deep neural networks are powerful but vulnerable to adversarial examples that are intentionally crafted to fool the networks. Recent studies have shown the vulnerability of deep neural networks in many NLP tasks, including reading comprehension (Jia and Liang, 2017), text classification (Samanta and Mehta, 2017; Wong, 2017; Liang et al., 2018; Alzantot et al., 2018), machine translation (Zhao et al., 2018; Ebrahimi et al., 2018; Cheng et al., 2018), dialogue systems (Cheng et al., 2019), and dependency parsing (Zheng et al., 2020). These methods attack an NLP model by replacing, scrambling, and erasing characters or words under certain semantic and syntactic constraints. In particular, most of them craft adversarial examples by substituting words with their synonyms in an input text to maximally increase the prediction error while maintaining the adversarial examples' fluency and naturalness. In this paper, we focus on these word substitution-based threat models and discuss the strategy to defend against such attacks.

The goal of adversarial defenses is to learn a model capable of achieving high test accuracy on both clean and adversarial examples. Adversarial training is one of the most successful defense methods for NLP models (Miyato et al., 2017; Sato et al., 2019; Zhu et al., 2019). During the training time, they replace a word with one of its synonyms that maximizes the prediction loss. By augmenting these adversarial examples with the original training data, the model is robust to such perturbations. However, it is infeasible to explore all possible combinations where each word in a sentence can be replaced with any of its synonyms. Also, when updating word embeddings during training, the distance between a word and its synonyms in the embedding space change dynamically. Therefore, the point-wise guarantee becomes insufficient, and the resulting models have shown to be vulnerable to strong attacks (Alzantot et al., 2018).

On the other hand, several certified defense methods have recently been proposed to ensure that the model predictions are unchanged when input word embeddings are perturbed within the convex hull formed by the embeddings of a word and its synonyms (Jia et al., 2019; Huang et al., 2019). However, due to the difficulty of propagating convex hull through deep neural networks, they compute a loose outer bound using Interval Bound Propagation (IBP). As a result, the convex hull may contain irrelevant words and lead to a significant performance drop on the clean data.

In this paper, we propose **Dirichlet Neighborhood Ensemble (DNE)** to create virtual sentences by mixing the embedding of the original word in the input sentence with its synonyms. By training on these virtual sentences, the model can enhance the robustness against word substitution-based perturbations. Specifically, our method samples an embedding vector in the convex hull formed by a word and its synonyms to ensure the robustness within such a region. In contrast to IBP, our approach better represents the synonyms' subspace by creating virtual sentences. To deal with complex error surface (e.g., surfaces containing multiple hills and valleys), a gradient-guided optimizer is applied to search for the most vulnerable points within the convex hull. By minimizing the error with these vulnerable points, we can guarantee with high probability that the resulting model is robust at any point within the convex hull (i.e., a set of synonyms). The framework can be extended to higher-order neighbors (synonyms) to boost the robustness further. In the inference time, the same Dirichlet sampling technique is used, and the prediction scores on the virtual sentences are ensembled to get a robust output.

Through extensive experiments with various model architectures on multiple data sets, we show that DNE consistently achieves better performance on clean and adversarial samples than existing defense methods. By conducting a detailed analysis, we found that DNE enables the embeddings of a set of similar words to be updated together in a coordinated way. In contrast, prior approaches either fix the word vectors during training (e.g., in the certified defenses) or update individual word vectors independently (e.g., in the adversarial training). We believe it is the crucial property why DNE leads to a more robust NLP model. Furthermore, unlike most certified defenses, the proposed method is easy to implement and can be integrated into any existing neural network including those with large architecture such as BERT (Devlin et al., 2019).

2 Related Work

In the text domain, adversarial training is one of the most successful defenses (Miyato et al., 2017; Sato et al., 2019; Zhu et al., 2019). A family of fast-gradient sign methods (FGSM) was introduced by Goodfellow et al. (2015) to generate adversarial examples in the image domain. They showed that the robustness and generalization of machine learning models could be improved by including high-quality adversarial examples in the training data. Miyato et al. (2017) proposed an FGSM-like adversarial training method to the text domain by applying perturbations to the word embeddings rather than to the original input itself. Sato et al. (2019) extended the work of Miyato et al. (2017) to improve the interpretability by constraining the directions of perturbations toward the existing words in the word embedding space.

Zhang and Yang (2018) applied several types of noises to perturb the input word embeddings, such as Gaussian, Bernoulli, and adversarial noises, to mitigate the overfitting problem of NLP models. Zhu et al. (2019) proposed a novel adversarial training algorithm, called FreeLB (Free Large-Batch), which adds adversarial perturbations to word embeddings and minimizes the resultant adversarial loss inside different regions around input samples. They add norm-bounded adversarial perturbations to the input sentences' embeddings using a gradient-based method and enlarge the batch size with diversified adversarial samples under such norm constraints. However, they focus on the effects on generalization rather than the robustness against adversarial attacks.

Recently a set of certified defenses has been introduced, which guarantees robustness to some specific types of attacks. For example, Jia et al. (2019) and Huang et al. (2019) use a bounding technique, interval bound propagation (IBP) to formally verify a model's robustness against word substitution-based perturbations. Shi et al. (2020) and Xu et al. (2020) proposed the robustness verification and training method for transformers based on linear relaxation-based perturbation analysis. However, these defenses often lead to loose upper bounds for arbitrary networks and result in a higher cost of clean accuracy. Furthermore, due to the difficulty of verification, certified defense methods are usually not scalable and remain hard to scale to complex prediction pipelines. To achieve certified robustness on large architectures, Ye et al. (2020) proposed a certified robust method called SAFER which is structure-free. However, the base classifier of SAFER is trained by the adversarial data augmentation. As shown in our experiments, randomly perturbing a word to its synonyms performs poorly in practice.

In the image domain, randomization has been shown to overcome many of these obstacles in the IBP-based defense. Empirically, Xie et al. (2017) showed that random resizing and padding in the input domain could improve the robustness. Liu et al. (2018) proposed to add Gaussian noise in both the input layer and intermediate layers of CNN in both training and inference time to improve the robustness. Lecuyer et al. (2019) provided a certified guarantee of this method, and later on, the bound is significantly improved in Cohen et al. (2019). The resulting algorithm, called randomized smoothing, has become widely used in certifying ℓ_2 robustness for image classifiers. These random smoothing methods are very much under-explored in NLP models. The main reason is that the adversarial examples in texts are usually generated by word substitution-based perturbations instead of small ℓ_p norm. In this paper, we show that random smoothing can be integrating with adversarial training to boost the empirical robust accuracy.

3 Method

We here consider a word substitution-based threat model, where every word in an input sentence can be replaced with one of its synonyms. Given a sentence and synonym sets, we would like to ensure that the prediction of a model trained with our method cannot be altered by any word substitution-based perturbation to the sentence. However, the number of possible perturbations scales exponentially with sentence length, so data augmentation cannot cover all perturbations of an input sentence. We use a convex hull formed by a word and its synonyms to capture word substitutions, which allows us to search for the worse-case over the convex hull. By minimizing the error with the worst-case, we can guarantee with high probability that the model is robust at any point within the convex hull (i.e., a set of synonyms).

The proposed method can be viewed as a kind of randomized defense on NLP models, where our main contribution is to show that it is essential to ensure the model works well in a region within the convex hull formed by the embeddings of a word and its synonyms instead of only ensuring model is good under discrete perturbation. Although DNE does not provide certified lower bounds like IBP, it achieves much better accuracy on both clean and adversarial data on different models, datasets, and attacks compared with IBP. DNE also can be easily integrated into any neural networks, including large architecture such as BERT.

Let f be a base classifier which maps an input sentence $x \in \mathcal{X}$ to a class label $y \in \mathcal{Y}$. We consider the setting where for each word x_i in the sentence x, we are given a set of its synonyms $\mathcal{S}(x_i)$ including x_i itself, where we know replacing x_i with any of $\mathcal{S}(x_i)$ is unlikely to change the semantic meaning of the sentence[1]. We relax the set of discrete points (a word and its synonyms) to a convex hull spanned by the word embeddings of all these points, denoted by $\mathcal{C}(x_i)$. We assume any perturbation within this convex hull will keep the semantic meaning unchanged, and define a smoothed classifier $g(x)$ based on random sampling within the convex hull as follows.

$$g(x) = \arg\max_{y \in \mathcal{Y}} \mathbb{P}_{\hat{x}}(f(\hat{x}) = y) \tag{1}$$

where $\hat{x}$ is generated by replacing the embedding of each word x_i in the sentence x with a point randomly sampled from x_i's convex hull $\mathcal{C}(x_i)$. In the training time, the base classifier is trained with "virtual" data augmentation sampled in the embedding space, where each word x_i is replaced with a point in the convex hull containing $\mathcal{C}(x_i)$ by the proposed sampling algorithm described Section 3.1. A new adversarial training algorithm is also designed to enable NLP models to defense against the strong attacks that search for the worst-case over all combinations of word substitutions. A similar sampling strategy is conducted in the inference time.

Note that it is impossible to precisely calculate the probabilities with which f classifies x as each class, so we use a Monte Carlo algorithm for evaluating $g(x)$. As shown in Fig. 1 (a), for a sentence x, we draw k samples of $\hat{x}$ by running k noise-corrupted copies of x through the base classifier $f(\hat{x})$, where $\hat{x}$ is generated by replacing the embedding of every word x_j in the sentence x with a point randomly sampled from $\mathcal{C}(x_j)$ (the pentagon with yellow dashed borders). If the class y appeared with maximal weight in the categorical distribution $\hat{x}$, the smoothed classifier $g(x)$ returns y. The decision regions of the base classifier are drawn in different colors if we evaluate the smoothed classifier at an input x_j, where the regions with different colors represent different classes.

Assuming that the word x_i is replaced with x_j by an adversary, we need to sample the points from the convex hull $\mathcal{C}(x_j)$ in the inference time. However, some of x_j's synonyms (indicated by yellow circles) are outside the region formed by x_i and its synonyms (indicated by blue circles). We thus should expand this region to the polygon with green dashed borders to make sure that the model makes the same prediction for any point sampled from the expanded region. We ensure that the smoothed

[1]Follow Jia et al. (2019), we base our sets of word substitutions $S(x_i)$ on the method of Alzantot et al. (2018).

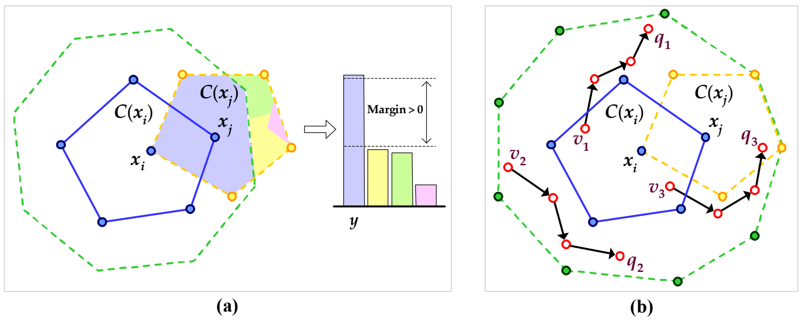

Figure 1: Consider a word (sentence of length one) x_i and its convex hull $\mathcal{C}(x_i)$ (projected to 2D for illustration) spanned by the set of its synonyms (blue circles). We assume that an adversary replaces x_i with one of its synonyms x_j. **(a)** Evaluating the smoothed classifier at the input x_j. The decision regions of the base classifier f are drawn in blue, green, and pink colors, representing different classes. If we expand $\mathcal{C}(x_i)$ to the polygon with green dashed borders when training the base classifier f, the size of the intersection of this polygon and $\mathcal{C}(x_j)$ is large enough to ensure that the smoothed classifier g labels x_j as $f(x_i)$. Here, the region where g labels x_j as $f(x_i)$ is "blue." **(b)** An example convex hull used to train the base classifier. Since the size of the intersection of $\mathcal{C}(x_i)$ and $\mathcal{C}(x_j)$ is small, we expand $\mathcal{C}(x_i)$ to the convex hull spanned by x_i's neighbors and "neighbors of neighbors" in their embedding space when training the base classifier f. Starting from three points $\boldsymbol{v_1}$, $\boldsymbol{v_2}$ and $\boldsymbol{v_3}$ sampled from the expanded convex hull (the largest polygon with green dashed borders), $\boldsymbol{q_1}$, $\boldsymbol{q_2}$ and $\boldsymbol{q_3}$ are the local "worst-case" points found by searching over the entire convex hull with the gradient-guided optimization method.

classifier label x_j as $f(x_i)$ by training the base classifier to label the instances sampled from the expanded region as $f(x_i)$ so that the blue region is always larger than green, yellow and pink ones.

3.1 Dirichlet Neighborhood Sampling

The random perturbations of x are combinatorial, and thus training the base classifier f that consistently labels any perturbation of x as y requires checking an exponential number of predictions. To better reflect those discrete word substitution-based perturbations, we sample the points from a convex hull using the Dirichlet distribution. This allows us to control how far we can expect the points are from any vertex of the convex hull. If a sampled point is very close to a vertex (i.e., a word), it simulates a word substitution-based perturbation in which the vertex is chosen to replace the original one. Any point sampled from $\mathcal{C}(x_i)$ can be represented as a convex combination of the embeddings of $\mathcal{S}(x_i)$:

$$\nu(x_i) = \sum_{x_j \in \mathcal{S}(x_i)} \beta_j \cdot \boldsymbol{x_j}, \tag{2}$$

where $\beta_j \geq 0$, $\Sigma_j \beta_j = 1$, and $\boldsymbol{x_j}$ (in bold type) denotes the embedding of x_j. A vector $\boldsymbol{\beta}$ contains the weights drawn from the Dirichlet distribution as follows:

$$\beta_1, \ldots, \beta_m \sim \text{Dir}(\alpha_1, \ldots, \alpha_m), \tag{3}$$

where m is the size of $\mathcal{S}(x_i)$, and the Dirichlet distribution is parameterized by a vector of $\boldsymbol{\alpha}$ used to control the degree in which the words in $\mathcal{S}(x_i)$ contribute to generate the vector $\nu(x_i)$.

3.2 Training the Base Classifier with Two-Hop Neighbors

For the smoothed classifier g to classify an adversarial example of x correctly and robustly, f needs to consistently classify $\hat{x}$ as the gold label of x. Therefore, we train the base classifier with virtual data augmentation $\hat{x}$ for each training example x. In Fig. 1 (b), we illustrate the process by considering a sentence with one word x_i and the set of its synonyms (shown as blue circles). The input perturbations span a convex hull of $\mathcal{C}(x_i)$ around the word x_i (the pentagon with blue borders, projected to 2D here). Assuming that the word x_i is replaced with x_j by an adversary, noise-corrupted samples will be drawn from $\mathcal{C}(x_j)$ (the pentagon with yellow dashed borders) instead of $\mathcal{C}(x_i)$. If the size of the intersection of $\mathcal{C}(x_i)$ and $\mathcal{C}(x_j)$ is small, we cannot expect f will consistently classify x_j as the same label as x_i. Therefore, we expand $\mathcal{C}(x_i)$ to the convex hull spanned by the word embeddings of the union of $\mathcal{S}(x_i)$ and all of $\mathcal{S}(x_j), x_j \in \mathcal{S}(x_i)$, namely x_i's 1-hop neighbors and 2-hop neighbors in their embedding space, denoted by $\mathcal{B}(x_i)$.

We use $\widetilde{x}$ to denote a virtual example created by replacing the embedding of every word x_i in an input sentence x with a point randomly sampled from the expanded $\mathcal{B}(x_i)$ by the Dirichlet distribution. Such expansions will slightly hurt the performance on the clean data. Recall that different values of $\boldsymbol{\alpha}$ can be used to control the degree in which the 1-hop and 2-hop neighbors contribute to generating $\widetilde{x}$.

In our implementation, we let the expected weights of the 2-hop neighbors are less than one-half of those of the 1-hop neighbors when computing $\widetilde{x}$ as Eq. (2) to reduce the impact on the clean accuracy.

The base classifier is trained by minimizing the cross-entropy error with virtual data augmentation by gradient descent. We assume the base classifier takes form $f(x) = \arg\max_{c \in \mathcal{Y}} s_c(x)$, where each $s_c(x)$ is the scoring function for the class c. That is, the outputs of the neural networks before the softmax layer. Our objective is to maximize the sum of the log-probabilities that f will classify each $\widetilde{x}$ as the label of x. Let $\mathcal{D}$ be a training set of n instances, and each of them is a pair of (x, y):

$$\begin{aligned} & \sum_{\forall (x,y) \in \mathcal{D}} \log \mathbb{P}_{\widetilde{x}}(f(\widetilde{x}) = y) \\ = & \sum_{\forall (x,y) \in \mathcal{D}} \log \mathbb{E}_{\widetilde{x}} \mathbf{1} \left[\arg\max_{c \in \mathcal{Y}} s_c(\widetilde{x}) = y \right], \end{aligned} \tag{4}$$

where $\widetilde{x}$ is a virtual example randomly created for an input example x. The softmax function can be viewed as a continuous, differentiable approximation of argmax:

$$\mathbf{1} \left[\arg\max_{c \in \mathcal{Y}} s_c(\widetilde{x}) = y \right] \approx \frac{\exp(s_y(\widetilde{x}))}{\sum_{c \in \mathcal{Y}} \exp(s_c(\widetilde{x}))}. \tag{5}$$

By the concavity of log and Jensen's inequality, the objective is approximately lower-bounded by:

$$\sum_{\forall (x,y) \in \mathcal{D}} \mathbb{E}_{\widetilde{x}} \left[\log \frac{\exp(s_y(\widetilde{x}))}{\sum_{c \in \mathcal{Y}} \exp(s_c(\widetilde{x}))} \right]. \tag{6}$$

This is the negative cross-entropy loss with virtual data augmentation. Maximizing Eq. (6) approximately maximizes Eq. (4).

Since the virtual data point defined in Eq. (2) is a linear combination of embeddings of $\mathcal{S}(x_i)$, the back-propagation will propagate the gradient to all these embeddings with nonzero coefficients, thus allowing updating all these embeddings together in a coordinated way when performing parameter updates. As illustrated in Fig. 1, the whole convex hull will be shifted together at each iteration. In contrast, traditional adversarial training only updates the embedding of one synonym (a vertex of the convex hull), which will distort the relative position of those embeddings and thus become slower and less stable. It is probably why the word embeddings are fixed during training in the certified defenses (Huang et al., 2019; Jia et al., 2019). Even though the word embeddings can be pre-trained, holding embeddings fixed makes them impossible to be fine-tuned for the tasks of interest, which may hurt the performance.

3.3 Adversarial Training

To promote higher robustness and invariance to any region within the convex hull, we further propose combining Dirichlet sampling with adversarial training to better explore different regions inside the convex hull $\mathcal{B}(x_i)$. Any point sampled from $\mathcal{B}(x_i)$ is represented as the convex combination of the embeddings of its vertices, which ensures that a series of points keep staying inside of the same $\mathcal{B}(x_i)$ while searching for the worst-case over the entire convex hull by any optimization method. Assuming that a virtual example $\widetilde{x}$ is generated for an input sentence x, we search for the next adversarial example to maximize the model's prediction error by updating every vector of weights $\boldsymbol{\beta} = \exp(\boldsymbol{\eta})$ by the following formula, each of them is used to represent a point sampled from $\mathcal{B}(x_i)$ as Eq. (2):

$$\begin{aligned} \boldsymbol{\eta} &\leftarrow \boldsymbol{\eta} - \epsilon \left\| \frac{\partial \log p(\widetilde{x}, y)}{\partial \boldsymbol{\eta}} \right\|_2, \\ p(\widetilde{x}, y) &= \frac{\exp(s_y(\widetilde{x}))}{\sum_{c \in \mathcal{Y}} \exp(s_c(\widetilde{x}))}, \end{aligned} \tag{7}$$

where ϵ is the step size. In order to ensure that the updated $\boldsymbol{\beta}$ satisfy $\beta_j \geq 0$ and $\Sigma_j \beta_j = 1$ as before, we sequentially apply logarithmic and softmax functions to $\boldsymbol{\beta}$ after it is randomly drawn from $\text{Dir}(\boldsymbol{\alpha})$. Note that $\text{softmax}(\log(\boldsymbol{\beta})) = \boldsymbol{\beta}$, and $\boldsymbol{\eta}$ will be updated instead of $\boldsymbol{\beta}$ in our implementation. By updating $\boldsymbol{\eta}$ only, the representation defined in Eq. (2) also ensures that a series of points keep staying inside of the same convex hull while searching for the worst-case over $\mathcal{B}(x_i)$.

As shown in Fig. 1 (b), we apply this update multiple times with a small step size (arrow-linked red circles represent data points generated after each update by adding gradient-guided perturbations to their preceding ones). When training the base classifier, we add all of the virtual examples generated at every search step (i.e., all of the points indicated by the red circles in Fig. 1 (b)) into the training set to better explore different regions around x.

3.4 Ensemble Method

As mentioned above, we use a Monte Carlo algorithm for evaluating $g(x)$. Given an input sentence x, we draw k Monte Carlo samples of $\hat{x}$ by running k noise-corrupted copies of x through the base classifier $f(\hat{x})$, where each $\hat{x}$ is created by replacing the embedding of every word x_i in the sentence x with a point randomly sampled with the Dirichlet distribution from $\mathcal{C}(x_i)$ (not from the expanded convex hull $\mathcal{B}(x_i)$ in the inference time).

We combine predictions by taking a weighted average of the softmax probability vectors of all the randomly created $\hat{x}$, and take the argmax of this average vector as the final prediction. We use CBW-D (Dubey et al., 2019) to compute those weights. The idea behind it is to give more weight to the predictions that have more confidence in their results. CBW-D calculates the weights w as a function of the differences between the maximum value of the softmax distribution and the other values as follows:

$$w = \sum_{c \in \mathcal{Y}, c \neq y} (p(\hat{x}, y) - p(\hat{x}, c))^r, \tag{8}$$

where y is the class having the maximum probability in a prediction, r is a hyperparameter tuned using cross-validation in preliminary experiments.

4 Experiments

We conducted experiments on multiple data sets for text classification and natural language inference tasks. Various model architectures (bag-of-words, CNN, LSTM, and attention-based) were used to evaluate our DNE and other defense methods under two recently proposed attacks. Ren et al. (2019) described a greedy algorithm, called Probability Weighted Word Saliency (PWWS), for adversarial text attacks based on word substitutions with synonyms. The word replacement order is determined by taking both word saliency and prediction probability into account. Alzantot et al. (2018) developed a generic algorithm-based attack, denoted by GA, to generate semantically and syntactically similar adversarial examples. They use a language model (LM) (Chelba et al., 2018) to rule out candidate substitute words that do not fit within the context. However, unlike PWWS, ruling out some candidates by the LM will significantly reduce the number of candidate substitute words (65% off on average). For a fair comparison, we report the robust accuracy under GA attack both with and without using the LM. We measure adversarial accuracy on perturbations found by the two attacks (PWWS and GA) on $1,000$ randomly selected test examples for each data set.

We primarily compare with recently proposed defense methods, including adversarial training (ADV) (Michel et al., 2019) and the interval bound propagation (IBP) based methods (Huang et al., 2019; Jia et al., 2019). The former can improve the model's robustness without suffering many drops on the clean input data by adding adversarial examples in the training stage. The latter was shown to be more robust to word substitution-based perturbations than ones trained with data augmentation. To demonstrate that mixing the embedding of the original word with its synonyms performs better than naively replacing the word with its synonyms, we designed a new baseline, called RAN. The models trained by RAN will take the corrupted copies of each input sentence as inputs, in which every word of the sentence is randomly replaced with one of its synonyms. The same random replacement is used in the inference time, and the prediction scores are ensembled to get an output. RAN can be viewed as a variant of SAFER (Ye et al., 2020), where during the training SAFER's perturbation set is replaced with the synonym set used by the adversaries and the number of ensembles is reduced to 16 (instead of $5,000$) at the inference time, which make it feasible to be evaluated empirically under the attacks.

4.1 Text Classification

We experimented on two text classification data sets: Internet Movie Database (IMDB) (Maas et al., 2011) and AG News corpus (AGNEWS) (Zhang et al., 2015). We implemented three models for these text classification tasks like (Jia et al., 2019). The bag-of-words model (BOW) averages the word embeddings for each word in the input, then passes this through a one-layer feedforward network with 100-dimensional hidden state to get a final logit. The other two models are similar, except they run either a CNN or a two-layer LSTM on the word embeddings. All models are trained on cross-entropy loss, and their hyperparameters are tuned on the validation set (see Appendix A.1 for details).

Table 1 reports both clean accuracy (CLN) and accuracy under two attack algorithms (PWWS and GA) on IMDB with three different model architectures (BOW, CNN, and LSTM). We use GA-LM to denote the GA-based attack that rules out candidate substitute words that may not fit well with the context by the LM (Chelba et al., 2018). We use ORIG to the testing and adversarial accuracy of the models trained without using any defense method.

As we can see from Table 1, DNE ($k = 16$) outperforms ADV and IBP on the clean input data, and consistently performs better than the competitors across the three different architectures under all the attack algorithms. For the text classification, LSTMs seem more vulnerable to adversarial attacks than BOWs and CNNs. Under the strongest attack GA, while the accuracy of LSTMs trained by

Table 1: Text classification on IMDB dataset.

IMDB	BOW				CNN				LSTM			
	CLN	PWWS	GA-LM	GA	CLN	PWWS	GA-LM	GA	CLN	PWWS	GA-LM	GA
ORIG	**90.2**	1.1	4.8	0.0	**89.9**	2.6	4.5	0.1	**89.6**	2.5	14.6	0.2
ADV	86.4	77.4	80.0	77.2	87.0	72.1	76.0	72.0	85.6	35.4	56.6	32.0
IBP	79.6	75.4	70.5	66.9	79.6	76.3	75.0	70.9	76.8	72.2	64.7	64.3
RAN	87.8	74.2	56.1	33.7	87.7	75.0	58.9	39.9	88.5	71.5	56.0	35.5
DNE	84.5	**81.1**	**81.3**	**79.0**	84.4	**79.6**	**79.9**	**77.8**	87.5	**84.0**	**84.3**	**82.8**

Table 2: Text classification on AGNEWS dataset.

AG	BOW				CNN				LSTM			
NEWS	CLN	PWWS	GA-LM	GA	CLN	PWWS	GA-LM	GA	CLN	PWWS	GA-LM	GA
ORIG	**90.6**	63.8	68.8	25.7	**91.5**	35.3	55.0	12.5	92.2	48.8	58.4	11.9
ADV	88.8	84.5	85.7	82.5	88.4	80.2	82.5	75.3	**92.4**	85.4	87.1	78.8
IBP	87.4	85.1	86.8	81.3	87.8	86.2	86.7	82.7	84.0	82.3	82.9	77.9
RAN	89.0	78.1	75.2	51.3	88.7	78.2	74.4	51.7	92.1	81.4	81.4	51.9
DNE	89.5	**89.1**	**89.1**	**88.7**	91.3	**90.3**	**89.6**	**89.1**	92.0	**91.2**	**91.0**	**89.4**

ORIG, ADV, IBP, and RAN dropped to 0.2%, 32%, 64.3%, and 8.1% respectively, the LSTM trained by DNE still achieved 82.2% accuracy. The results on AGNEWS are reported in Table 2, and we found similar trends as those on IMDB. Any model performed on AGNEWS shows to be more robust than the same one on IMDB. It is probably because the average length of the sentences in IMDB (255 words on average) is much longer than that in AGNEWS (43 words on average). Longer sentences allow the adversaries to apply more word substitution-based perturbations to the examples. Generally, DNE performs better than IBP and comparable to ADV on the clean data, while it outperforms the others in all other cases. The results for both datasets show that our DNE consistently achieves better clean and robust accuracy.

4.2 Natural Language Inference

We conducted the experiments of natural language inference on Stanford Natural Language Inference (SNLI) (Bowman et al., 2015) corpus. We also implemented three models for this task. The bag-of-words model (BOW) encodes the premise and hypothesis separately by summing their word vectors, then feeds the concatenation of these encodings to a two-layer feedforward network. The other two models are similar, except they run either a Decomposable Attention (DecomAtt) (Parikh et al., 2016) or BERT (Devlin et al., 2019) on the word embeddings to generate the sentence representations, which uses attention between the premise and hypothesis to compute richer representations of each word in both sentences. All models are trained with cross-entropy loss, and their hyperparameters are tuned on the validation set (see Appendix A.2).

As reported in Table 3, DNE generally performs better than the others on the robust accuracy while suffering little performance drop on the clean data on SNLI. Although our proposed baseline RAN ($k = 16$) achieves a slightly higher accuracy (just 1.2% difference) with BERT under PWWS attack, its accuracy rapidly drops to 27% under the more sophisticated attack GA, while DNE still yields 62.7% in accuracy. The results on SNLI show that DNE can be applied to attention-based models like DecomAtt and scales well to large architectures such as BERT. We leave the results of IBP with BERT as unknown since it is still a question whether IBP-based methods can be applied to BERT.

4.3 Ablation Study

We conducted an ablation study over IMDB validation set on DNE with CNNs to analyze the robustness and generalization strength of different variants. The "w/o EXPANSION" in the second row of Table 4 indicates that given any word x_i in a sentence, we generate virtual examples by sampling from $\mathcal{C}(x_i)$ instead of the expanded $\mathcal{B}(x_i)$ during the training. The variant of DNE trained without using the adversarial training algorithm described in Section 3.3 is indicated by "w/o ADV-TRAIN". If the single-point update strategy is applied to train DNE, we still use the same gradient-guided optimization method to find adversarial examples over $\mathcal{B}(x_i)$, but the found adversarial example $\boldsymbol{x_j}$ is represented as $\boldsymbol{x_i} + \Delta$, where Δ is the distance between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$. By such representation, only $\boldsymbol{x_i}$ will be updated during the training instead of the embeddings of all its synonyms, and this variant is indicated by "w/o COORD-UPD". In the last row, we also report the results predicted without using the ensemble method (i.e., $k = 1$).

Table 3: Natural language inference on SNLI dataset.

SNLI	BOW				DecomAtt				BERT			
	CLN	**PWWS**	**GA-LM**	**GA**	**CLN**	**PWWS**	**GA-LM**	**GA**	**CLN**	**PWWS**	**GA-LM**	**GA**
ORIG	**80.9**	24.4	41.6	8.26	81.2	23.1	40.8	8.1	**90.5**	42.6	56.7	19.9
ADV	80.4	67.9	71.0	59.5	**81.9**	71.7	73.8	65.2	89.4	68.2	79.0	58.2
IBP	79.3	74.9	75.0	71.0	77.3	72.8	73.7	70.5	−−	−−	−−	−−
RAN	79.0	65.7	44.4	27.8	80.3	67.2	51.1	30.6	89.9	**72.7**	42.7	27.0
DNE	79.8	**76.3**	**75.3**	71.5	80.2	**77.4**	**76.7**	**74.6**	90.1	71.5	**80.1**	**62.7**

Table 4: Ablation Study on IMDB.

Model	CLN	PWWS	GA-LM	GA
DNE	86.2	81.4	79.4	75.4
w/o EXPANSION	−0.1	−14.2	−24.0	−45.0
w/o ADV-TRAIN	+1.6	− 7.8	−19.8	−34.6
w/o COORD-UPD	−0.0	− 4.2	− 9.0	−12.8
w/o ENSEMBLE	−0.4	− 1.8	− 7.0	− 9.4

As we can see from Table 4, the differences in accuracy among the variants of DNE are negligible on the clean data. The key components to improve the robustness of the models in descending order by their importance are the following: sampling from the expanded convex hull, combining with adversarial training, updating the word embeddings together, and using the ensemble to get the prediction. We also observed that the stronger the attack algorithm is, the more effective these components will be. When both “expansion” and “adversarial” are removed, the resulting accuracies on the validation set of IMDB dataset with the CNN-based model drop to 48.6% (PWWS) and 17.0% (GA).

In all the above experiments, we simply set the value of α for 1-hop neighbors to 1.0, and that for 2-hop neighbors to 0.5. We also conducted two experiments to investigate whether the Dirichlet distribution is essential. In the first one, we uniformly sample the weights (by setting the value of α for both 1-hop and 2-hop neighbors to 1.0) and do an adversarial training step. The clean accuracy is 82.4%, and the accuracies under the PWWS and GA attacks are 79.8% and 78.21% respectively. In the second experiment, we randomly sample a vertex from 2-hop neighbors and then do the same adversarial training. The resulting accuracies are 85% (clean), 75.8% (PWWS), and 54.6% (GA). We found there is a trade-off between the clean accuracy and the accuracy under the attack. Generally, the greater the value of α is, the more robust the models will be, but the worse they perform on the clean data. We also used different values of α in the Dirichlet distribution to control the degree in which 1-hop and 2-hop neighbors contribute to generating adversarial examples. If we treat 2-hop neighbors equally as 1-hop ones, it will significantly reduce the model's accuracy on the clean data, although it may lead to more robust models.

Although this study mainly focuses on the setting specified by (Jia et al., 2019), we also conducted experiments in which the defenders do not know how the attackers generate synonyms. We used the synonyms suggested by (Alzantot et al., 2018) for training and evaluated the resulting models with CNNs and LSTMs on IMDB and AGNEWS datasets under a new attack system, called TextFooler (Jin et al., 2020). We strictly followed the method proposed in (Jin et al., 2020) to generate synonyms during the attacking phase. The experimental results show that DNE achieved 30.6% and 13.4% higher in average accuracy than ADV on AGNEWS and IMDB respectively.

5 Conclusion

In this study, we develop a novel defense algorithm for NLP models to substantially improve the robust accuracy without sacrificing their performance too much on clean data. This method is broadly applicable, generic, scalable, and can be incorporated with little effort in any neural network, and scales to large architectures. A novel adversarial training algorithm is also proposed, enabling NLP models to defend against the strong attacks that search for the worst-case over all combinations of word substitutions. We demonstrated through extensive experimentation that our adversarially trained smooth classifiers consistently outperform all existing empirical and certified defenses by a significant margin on three datasets across different network architectures, establishing state-of-the-art for defenses against adversarial text attacks.

We choose to focus on synonym swapping because it is one of the most influential and widely-used attack methods. There is still no effective method to defend against existing attack algorithms from this kind, such as Hotflip (Ebrahimi et al., 2018), PWWS (2019), GA (2018), TextFooler (Jin et al., 2020) etc. A general method to defend more different attacks is worth exploring, but we choose to leave this as future work.

Acknowledgements

This work was supported by Shanghai Municipal Science and Technology Major Project (No. 2021SHZDZX0103), National Science Foundation of China (No. 62076068) and Zhangjiang Lab.

References

Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani Srivastava, and Kai-Wei Chang. 2018. Generating natural language adversarial examples. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2018. One billion word benchmark for measuring progress in statistical language modeling. *Computing Research Repository*, arXiv: 1312.3005.

Minhao Cheng, Wei Wei, and Cho-Jui Hsieh. 2019. Evaluating and enhancing the robustness of dialogue systems: A case study on a negotiation agent. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. 2018. Seq2Sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples. *Computing Research Repository*, arXiv: 1803.01128.

Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. 2019. Certified adversarial robustness via randomized smoothing. In *Proceedings of the International Conference on Machine Learning*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, and Dhruv Mahajan. 2019. Defense against adversarial images using web-scale nearest-neighbor search. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-box adversarial examples for text classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Po-Sen Huang, Robert Stanforth, Johannes Welbl, Chris Dyer, Dani Yogatama, Sven Gowal, Krishnamurthy Dvijotham, and Pushmeet Kohli. 2019. Achieving verified robustness to symbol substitutions via interval bound propagation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Robin Jia, Aditi Raghunathan, Kerem Göksel, and Percy Liang. 2019. Certified robustness to adversarial word substitutions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations*.

Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. 2019. Certified robustness to adversarial examples with differential privacy. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 656–672. IEEE.

Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. 2018. Deep text classification can be fooled. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. 2018. Towards robust neural networks via random self-ensemble. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 369–385.

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 Annual Meeting of the Association for Computational Linguistics*.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Takeru Miyato, Andrew M Dai, and Ian Goodfellow. 2017. Adversarial training methods for semi-supervised text classification. In *Proceedings of the International Conference on Learning Representations*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *Computing Research Repository*, arXiv: 1707.02812.

Motoki Sato, Jun Suzuki, Shindo, and Yuji Matsumoto. 2019. Interpretable adversarial perturbation in input embedding space for text. In *Proceedings of the International Joint Conference on Artificial Intelligence*.

Zhouxing Shi, Kai-Wei Chang Huan Zhang, Minlie Huang, and Cho-Jui Hsieh. 2020. Robustness verification for transformers. In *Proceedings of the International Conference on Learning Representations*.

Catherine Wong. 2017. DANCin SEQ2SEQ: Fooling text classifiers with adversarial text example generation. *Computing Research Repository*, arXiv: 1712.05419.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. 2017. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. 2020. Automatic perturbation analysis for scalable certified robustness and beyond. In *Advances in Neural Information Processing Systems*.

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

Dongxu Zhang and Zhichao Yang. 2018. Word embedding perturbation for sentence classification. *Computing Research Repository*, arXiv: 1804.08166.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the Conference on Neural Information Processing Systems*.

Zhengli Zhao, Dheeru Dua, and Sameer Singh. 2018. Generating natural adversarial examples. In *Proceedings of the International Conference on Learning Representations*.

Xiaoqing Zheng, Jiehang Zeng, Yi Zhou, Cho-Jui Hsieh, Minhao Cheng, and Xuanjing Huang. 2020. Evaluating and enhancing the robustness of neural network-based dependency parsing models with adversarial examples. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Tom Goldstein, and Jingjing Liu. 2019. FreeLB: Enhanced adversarial training for language understanding. In *Proceedings of the International Conference on Learning Representations*.

Appendix

A.1 Experimental Details for Text Classification

We report in Table 5 and 6 the values of hyperparameters used to train the text classification models. The values of hyperparameters in Dirichlet Neighborhood Ensemble (DNE) are listed in Table 7. All the models were trained with the cross-entropy loss, and their hyper-parameters were tuned on the validation sets.

Table 5: Hyperparameters for training the text classification models.

Model	Embedding	Hidden	Layer	Kernel
BOW	300, GloVe	100	$--$	$--$
CNN	300, GloVe	100	1	3
LSTM	300, GloVe	100	2	$--$

A.2 Experimental Details for Natural Language Inference

All the models were initialized by the pre-trained Glove word embeddings and trained with the cross-entropy loss. Their hyper-parameters were tuned on the validation sets.

Bag of Words (BOW): We use a bag-of-word model with the same hyperparameters as shown in Table 5 to encode the premise and hypothesis

Table 6: Training hyperparameters for the text classification (BOW, CNN, and LSTM) models. The same values were used for all the settings (plain, data augmentation, and robust training).

Hyperparameter	**Value**
Optimizer	Adam (Kingma and Ba, 2015)
Learning rate	0.5×10^{-3}
Dropout (embedding)	0.3
Weight decay	1×10^{-4}
Batch size	32
Gradient clip	$(-1, 1)$
Epochs	20

Table 7: Hyperparameters of DNE for text classification and natural language inference tasks.

Hyperparameter	**Value**
Dirichlet distribution α (1-hop neighbors)	1.0
Dirichlet distribution α (2-hop neighbors)	0.5
Step size ϵ (adversarial training)	10
Number of steps (adversarial training)	3
Parameter r (ensemble method)	3

separately by summing their word vectors, and then feeds the concatenation of these encodings to a two-layer feedforward network with a 300-dimensional hidden state. We used the Adam optimizer (with a learning rate 0.5×10^{-3}), and set the dropout rate on word embedding to 0.3, the weight decay to 1×10^{-4}, the batch size to 128, the maximum number of epochs to 20, and the gradient clip to $(-1, 1)$ for the training.

Decomposable Attention (DecomAtt): We implemented the decomposable attention model as described in (Parikh et al., 2016) except for a few differences listed as follows:

- We did not normalize GloVe vectors (Pennington et al., 2014).
- We used the Adam optimizer (with a learning rate of 0.5×10^{-3}) instead of AdaGrad.
- We used a dropout rate of 0.3 on word embeddings.
- We used a batch size of 128 instead of 4.
- We clipped the value of gradients to be within $(-1, 1)$.
- We set the value of weight decay to 1×10^{-4}.
- The intra-sentence attention was not used.

Bidirectional Encoder Representations from Transformers (BERT): We implemented BERT as described in (Devlin et al., 2019) except for a few differences listed below:

- We applied a "bert-base-uncased" architecture (12-layer, 768-hidden, 12-heads, 110M parameters).
- We use the Adam optimizer with a learning rate of 0.4×10^{-4}.
- We used a batch size of 8.
- We set the number of epochs to 3.
- We clipped the value of gradients to be within $(-1, 1)$.
- We set the value of weight decay to 1×10^{-4}.
- We used slanted triangular learning rates described in (Howard and Ruder, 2018).

We report in Table 7 the hyperparameter values of Dirichlet Neighborhood Ensemble (DNE) used for SNLI benchmark, and they were tuned on the validation set of SNLI.

Table 8: Effect of Parameter $\boldsymbol{\alpha}$ on IMDB.

α, λ	**CLN**	**PWWS**	**GA-LM**	**GA**
$0.1, 0.02$	**86.2**	79.0	76.0	68.2
$0.1, 0.1$	**86.2**	81.4	79.4	75.4
$0.1, 0.5$	84.8	**82.2**	79.8	76.4
$1.0, 0.02$	85.6	78.8	80.4	75.6
$1.0, 0.1$	85.1	80.4	**80.8**	77.8
$1.0, 0.5$	81.6	78.6	79.4	**78.2**

A.3 Effect of Parameters of Dirichlet Distribution

Given a word x_i, different values of $\boldsymbol{\alpha}$ are used to control how much its 1-hop and 2-hop neighbors contribute to generating virtual adversarial examples. The value also determines the size of the expansion from $\mathcal{C}(x_i)$ to $\mathcal{B}(x_i)$. In order to reduce the impact on the clean accuracy, we let the expected weights of the 2-hop neighbors are $\lambda \in (0, 0.5]$ times of those of the (1-hop) nearest neighbors. We tried a few different values of α and λ on IMDB to understand how the choice of them impact upon the performance. As shown in Table 8, we found that if the value of α is fixed, the greater the value of λ, the more robust the models will become, but the worse they perform on the clean input data. A small value of α seems to be preferable, which allows us to simulate the discrete word substitution-based perturbations better. We found that 1-hop and 2-hop neighbors cannot be treated equally; otherwise, it will significantly reduce the model's accuracy on the clean data. For example, if we uniformly sample the weights of 1-hop and 2-hop neighbors, the clean accuracy drops to 82.4% (-3.8%) on the validation set of the IMDB dataset.