PAT: Geometry-Aware Hard-Label Black-Box Adversarial Attacks on Text

Muchao Ye
The Pennsylvania State University
University Park, Pennsylvania, USA
muchao@psu.edu

Jinghui Chen
The Pennsylvania State University
University Park, Pennsylvania, USA
jzc5917@psu.edu

Chenglin Miao
Iowa State University
Ames, Iowa, USA
cmiao@iastate.edu

Han Liu
Dalian University of Technology
Dalian, Liaoning, China
liu.han.dut@gmail.com

Ting Wang
The Pennsylvania State University
University Park, Pennsylvania, USA
ting@psu.edu

Fenglong Ma*
The Pennsylvania State University
University Park, Pennsylvania, USA
fenglong@psu.edu

ABSTRACT

Despite a plethora of prior explorations, conducting text adversarial attacks in practical settings is still challenging with the following constraints: black box – the inner structure of the victim model is unknown; hard label – the attacker only has access to the top-1 prediction results; and semantic preservation – the perturbation needs to preserve the original semantics. In this paper, we present PAT,[1] a novel adversarial attack method employed under all these constraints. Specifically, PAT explicitly models the adversarial and non-adversarial prototypes and incorporates them to measure semantic changes for replacement selection in the hard-label black-box setting to generate high-quality samples. In each iteration, PAT finds original words that can be replaced back and selects better candidate words for perturbed positions in a geometry-aware manner guided by this estimation, which maximally improves the perturbation construction and minimally impacts the original semantics. Extensive evaluation with benchmark datasets and state-of-the-art models shows that PAT outperforms existing text adversarial attacks in terms of both attack effectiveness and semantic preservation. Moreover, we validate the efficacy of PAT against industry-leading natural language processing platforms in real-world settings.

CCS CONCEPTS

• **Security and privacy**;

KEYWORDS

hard-label adversarial attack; robustness of language model

ACM Reference Format:
Muchao Ye, Jinghui Chen, Chenglin Miao, Han Liu, Ting Wang, and Fenglong Ma. 2023. PAT: Geometry-Aware Hard-Label Black-Box Adversarial Attacks on Text. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3580305.3599461

* indicates corresponding author.

[1] The implementation code is available at https://github.com/machinelearning4health/PAT.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08…$15.00
https://doi.org/10.1145/3580305.3599461

1 INTRODUCTION

Deep neural networks (DNNs) have obtained remarkable achievements in various machine learning tasks [7, 11, 30]. However, it is known that DNNs are subject to the *adversarial attack* [9], which crafts malicious inputs that are nearly undetectable by human perception to deceive target victim models. A number of adversarial attack methods have since been proposed. However, compared with a plethora of studies in continuous domains, e.g., in images [4, 6, 18], conducting text adversarial attacks [13, 25] by synonym substitution under practical settings is still challenging and awaits improvement. The task of text adversarial attack in practical settings includes a variety of constraints: *black box* – the architectures and parameters of target victim models are often unknown; *hard label* – the attacker is only able to access the top-1 prediction results; and *semantic preservation* – the perturbation needs to preserve the semantics of original texts.

The existing attack pipelines under such practical settings usually adopt three steps: (1) random initialization, (2) replacing original words back, and (3) selecting a replacement. Since the last step allows attackers to get out of the local optimum in the huge search space of the exponential number of possible combinations, existing works mainly focus on designing novel word replacement strategies, including heuristic-based and gradient-based approaches. Genetic algorithm-based optimization [19] is one of the representative **heuristic**-based attack approaches, which maintains a population of adversarial candidates at each iteration to search for the optimal one. Such approaches have to implicitly characterize the semantic change using the survival of good candidates at each iteration, which requires a prohibitively large number of queries for seeking the optimal from a whole population. **Gradient**-based attacks [28, 29] are then proposed to alleviate this inefficiency by formulating the perturbation as an optimization problem in the auxiliary continuous word embedding space. Although they outperform heuristic-based ones, there still exist research gaps in generating high-quality adversarial examples.

Motivation. To specify, state-of-the-art gradient-based methods such as LeapAttack [28] use the estimated gradients to determine

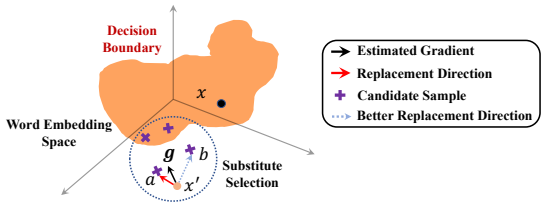


Figure 1: Drawbacks of existing gradient-based methods.

the replaced synonym words for achieving attacks. They assume that adversarial perturbations in the word embedding space obey Gaussian distributions as the ones do on images [4, 6]. However, this assumption does not always hold because *candidate words only discretely distribute in the embedding space*. Besides, they need to *map the continuous perturbations to the discrete words, which causes unavoidable approximation errors*. Figure 1 shows an example to demonstrate the drawbacks of existing gradient-based attacks. After estimating the gradient $\mathbf{g}$, existing approaches then map $\mathbf{g}$ to a candidate word based on the L_2 norm due to the discrete selection. In this example, the replacement will be the word a. However, the word b seems to be a better substitution because it is closer to the decision boundary. Therefore, in this paper, we eye on the challenge of designing an effective strategy for determining discrete word replacement *with a suitable semantic similarity measure* in the hard-label setting.

Our Approach. In this paper, we follow a *geometry-aware design principle* and propose a solution named PAT based on **prototype estimation** for adversarial attack to solve this challenge. In particular, PAT utilizes the **distance** to the estimated adversarial and non-adversarial prototypes from each candidate word to directly measure the semantic change instead of using the estimated gradients. As a result, the semantic deviation of each candidate word can be calculated from its distance to the estimated prototypes under the geometry interpretation – high-quality adversarial examples are the ones closer to non-adversarial prototype.

As demonstrated in Figure 2(a), PAT adopts the three-step pipeline. In Step (1), PAT randomly initializes an adversarial example x'_0 for the original input x following [19, 28, 29]. From x'_0, PAT will *iteratively* run the combination of Step (2) and Step (3) to find out the optimal replacements in x'_0. As shown in Figure 2(b), for each iteration t ($0 \leq t \leq T-1$), the input of Step (2) outputs $\tilde{x}_t$ with input x'_t, and Step (3) outputs x'_{t+1} with $\tilde{x}_t$ as the input. To illustrate:

In Step (2), PAT continuously replaces original words back in the input x'_t based on their contributions to the original semantics, where x'_t is evolved from x'_0. For example, PAT will replace "difficult" with the original word "hard" and then calculate the semantic similarity improvement. The larger the improvement is, the more helpful it is to retain original semantics and generate high-quality adversarial samples. Thus, these improvement scores determine the order of replacing the original words back to generate $\tilde{x}_t$.

In Step (3), PAT introduces the estimated prototypes to discover better replacements for the remaining perturbed words to preserve more original semantics iteratively. Take Figure 2(c) as an example. PAT starts with the word $\tilde{w}_j^{(t)}$ = "difficult" first in this step. To find out a better replacement, PAT randomly samples several synonyms of the original word "hard" as the exploration directions. PAT replaces the target word "difficult" with each synonym in $\tilde{x}_t$ and then queries the black-box model to output the predicted label for the new sentence. Replacing some synonyms circled by the blue dots in Figure 2(c) may make some new sentences still adversarial, but those circled by the green dots are not. Next, PAT adopts the metric learning principle [23, 27] to use the centers (i.e., $\star$) of these two types of synonyms in the word embedding space as the **prototypes of the adversarial and non-adversarial ones** for the word "difficult" only. PAT then measures the influence on the semantic change based on the **distances** to them. The word in the adversarial half-space whose projection into the line segment of two centers has the smallest distance to the non-adversarial center will be selected as the new replacement. As shown in Figure 2(d), PAT will choose "*challenging*" as the replacement of "difficult" with this principle. After that, PAT will conduct the same process for the next word "think" and the rest in $\tilde{x}_t$ one by one. After running Step (3) for all words, we will get x'_{t+1}.

Contributions. Our contributions are as follows:

- We propose a novel geometry-aware hard-label black-box text adversarial attack that discovers substitution words for original ones based on prototype estimation. This design relieves the limitation of existing methods and provides a more suitable measure for characterizing the semantic change caused by different candidate words under practical settings.
- We introduce a metric learning-based method for selecting the replacements. Using the distance between the prototypes in the word embedding space, we explicitly characterize the semantic change brought by each word in the continuous word embedding, which helps find better candidate words in the hard-label setting.
- PAT attains the best performance measured by semantic similarity and perturbation rate in attacking representative victim models on five commonly used text classification datasets. Moreover, the success of attacking three real-world APIs also demonstrates the effectiveness of PAT.

2 PRELIMINARIES

2.1 DNNs for Text Data

Text is a special type of discrete data that are constructed by a sequence of words. For text DNNs, each input text sample can be represented as $x = [w_1, \cdots, w_n]$, where n is the number of words in x, and w_i ($1 \leq i \leq n$) is the i-th word. In this paper, DNNs are used for the text classification task, and they aim to correctly classify each x into the ground truth label $y \in C$, i.e., $f(x) = y$, where $C = \{y_1, \cdots, y_c\}$ is the set of label categories.

2.2 Problem Formulation

In the setting of adversarial attacks, the attackers want to perturb original text samples to fool f. In other words, for any x that satisfies $f(x) = y$, attackers try to make f overturn its previous prediction by adding a small perturbation, where f is referred to as the victim model. During the adversarial attack, there is a set of allowable substitute synonym $S(w_i) = \{s_i^{(1)}, ..., s_i^{(M)}\}$, where $M = |S(w_i)|$,

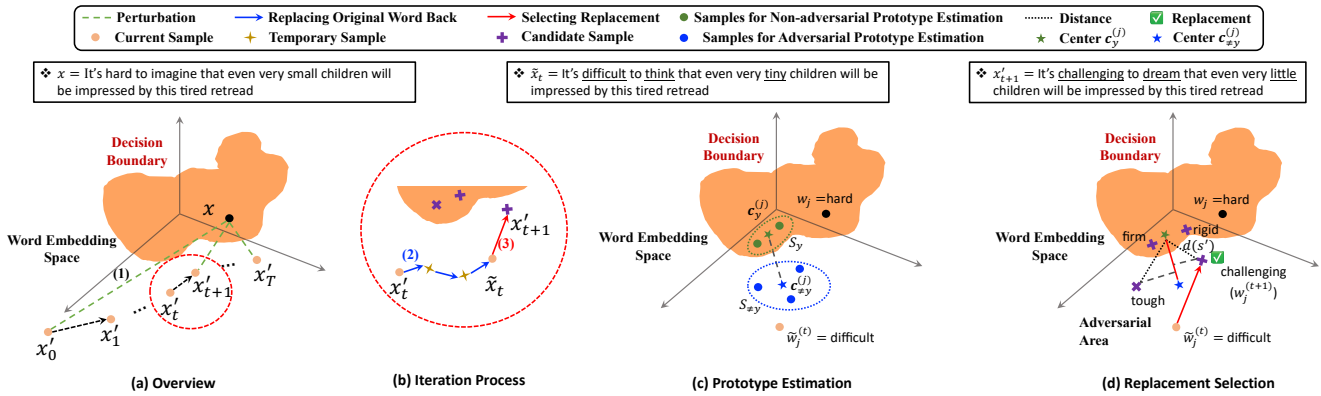


Figure 2: (a) Overview of PAT. It is an (b) iterative process for attaining optimal text adversarial examples, starting with (1) random initialization. In each iteration, PAT (2) replaces the original words back and (3) selects replacement by prototype estimation. Step (3) includes the operations of (c) prototype estimation and (d) replacement selection. "difficult" is the position the current prototype estimation focuses on, and the rest underlined ones will be the next position for the same operation.

accompanying each word w_i, and $w_i \in S(w_i)$. The attackers will replace each word w_i by its synonym $w'_i \in S(w_i)$ to construct a perturbed sample $x' = [w'_1, \cdots, w'_n]$. We say that the attackers conduct a successful adversarial attack if they craft an x' satisfying

$$f(x') \neq y. \tag{1}$$

We would like to point out that the event that victim models have similar predictions for two text samples is independent with the one that two text samples have similar semantic similarity. It is a common case that there exists more than one x' satisfying Eq. (1). The semantic preservation condition requests that the optimal adversarial example x^* is the one having the highest semantic similarity with the original sample x among all valid adversarial examples [20]. Mathematically, the optimal adversarial example x^* for a correctly classified sample x by f satisfies:

$$x^* = \arg\max_{x'} \ \mathrm{Sim}(x, x'), \ s.t. \ f(x') \neq f(x), \tag{2}$$

where $\mathrm{Sim}(x, x')$ is a predefined semantic similarity calculation function with regard to x and each adversarial example x', e.g., the Universal Sequence Encoder [3].

This paper targets at designing a new type of adversarial attack in the hard-label black-box setting. The difficulty of this setting stems from the assessed information of attackers from the victim model, where they can only know the top-1 prediction instead of the prediction score distribution with respect to each category in C. Using such limited information is difficult for attackers to distinguish the semantic change brought by each word.

2.3 Victim Models

Following the setting of existing hard-label black-box adversarial attack methods, we use the following three natural language processing (NLP) models as the victim models, including BERT [7], WordCNN [14], and WordLSTM [12]. These three models are all representative methods that are widely used in the NLP tasks throughout the development of DNNs for text data, and the well-trained models are publicly accessible from the repository of [19].

3 THE PROPOSED PAT METHOD

3.1 Overview

As shown in Figure 2(a), PAT harnesses an iterative process to gradually optimize the quality of adversarial examples. In the beginning, the optimization gets the initial guess x'_0 by Step (1) random initialization, which is to keep replacing original words with their random synonyms until it finds an initial adversarial example. In the rest iterations, PAT adopts a two-step process to optimize the initial adversarial examples, i.e., Step (2) replacing original words back while keeping the text candidate adversarial and Step (3) discovering better substitutes that can recover the original semantics by the prototype estimation. The detailed process is shown in Algorithm 1 in the Appendix.

3.2 Initialization

For the iterative optimization process, we need to attain an initial solution by random initialization first. This step is necessary because the lack of prediction score distribution in the hard-label setting does not allow attackers to find adversarial examples during optimizing the perturbation. It is widely adopted in the hard-label black-box adversarial attacks in various domains, including images [4–6] and text [19, 29]. In this operation, for each word w_i in x, we randomly select a synonym $w_i^{(0)} \in S(w_i)$ as its replacement. If we attain a text sample x'_0 that satisfies $f(x'_0) \neq y$, where $x'_0 = [w_1^{(0)}, \cdots, w_n^{(0)}]$, we now have an initial guess of the solution for Eq. (1). However, we usually need to change multiple words in x to obtain x'_0, making x'_0 generally have low quality, where the quality is measured by both the semantic similarity between x'_0 and x and word change percentage, i.e., perturbation rate compared with x.

To get an optimal adversarial example x^*, PAT will then iteratively optimize the semantic similarity between x and the adversarial example x'_t $(0 \leq t \leq T-1)$ in t-th iteration, where T is the total number of iterations. Given an x'_t, PAT obtains the adversarial example x'_{t+1} in two steps, which are detailed in Sec. 3.3 and 3.4.

3.3 Replacing Original Words Back

The optimization during each iteration starts from finding out the words that can be replaced back to the adversarial example x'_t without hurting the adversary property of the sample. To preserve the original semantics, this step uses the semantic similarity improvement brought by each original word to decide the replacement order [19]. In particular, for each word $w_i^{(t)}$ in $x'_t = [w_1^{(t)}, \cdots, w_n^{(t)}]$, PAT replaces it with its original word w_i in x, and we then have a temporary sample $x_{\text{tem}}^{(i)} = [w_1^{(t)}, \cdots, w_i, \cdots, w_n^{(t)}]$. If $x_{\text{tem}}^{(i)}$ still satisfies the adversary property $f(x_{\text{tem}}^{(i)}) \neq y$, we calculate the semantic similarity improvement as a measure to determine the replacement order. The semantic similarity improvement is defined as follows:

$$sim_i = \text{Sim}(x, x_{\text{tem}}^{(i)}) - \text{Sim}(x, x'_t). \tag{3}$$

After looping over all words in x'_t, the set $\{sim_i\}$ can tell us which word can make the adversarial example stay adversarial and their importance in improving the semantic similarity with the original adversarial sample. The original word w_i with higher sim_i indicates that it recovers more semantic similarity. Therefore, the next operation is to keep replacing the original words w_i back to x'_i in the descending order of semantic similarity scores $\{sim_i\}$ until the new sample is not adversarial. After that, we obtain an adversarial example $\tilde{x}_t$ that has more original words with x compared to x'_t. The detailed process is shown in Algorithm 2 in the Appendix.

3.4 Selecting Replacement by Prototype Estimation

With the adversarial example $\tilde{x}_t = [\tilde{w}_1^{(t)}, \cdots, \tilde{w}_n^{(t)}]$ attained above, PAT then selects better substitutions for the words in $\tilde{x}_t$ that are not the same as the original words by using the prototype estimation, which is shown in Figure 2(c)-(d) and Algorithm 3 in the Appendix. The attack order is determined by random sampling. In our random sampling, we calculate the L_2 norms of current replacements and their corresponding original words in the embedding space, and the sampling probabilities associated with each word are the softmax ones of these computed L_2 norms. Pseudocode of this process is shown in Algorithm 3.

Prototype Estimation. Suppose the j-th word $\tilde{w}_j^{(t)}$ is not equal to w_j, and we want to choose a better replacement that leads to an adversarial example with greater semantic similarity. To conduct this process under the geometry-aware intuition, we rely on the adversarial and non-adversarial prototypes to judge the semantic deviation of each substitution candidate with the original word. We estimate it based on a frozen pre-trained word embedding space $\mathcal{H} \in \mathbb{R}^h$ (where h is the dimension size), e.g., the Counter-Fitted Word Vectors [21]. Specifically, we get the word embeddings of the original word and all candidate words first. Next, we estimate the prototypes given adversarial examples and non-adversarial ones by random sampling. The estimated prototypes are then used for determining the candidate word.

Condition for Prototype Estimation. Before we conduct the prototype estimation, we need to make sure that we have observed that some candidate words cause the existing text to become adversarial while the others do not. To illustrate, for the j-th word $\tilde{w}_j^{(t)}$ in the text sample $\tilde{x}_t$, we randomly sample K candidate words $\{s_j^{(1)}, \cdots, s_j^{(K)}\}$ from the synonym set $S(w_j)$. For each sampled word $s_j^{(k)}$ $(1 \leq k \leq K)$, PAT explores whether the condition is satisfied by seeing if some sampled candidate words lead to adversarial examples and the others lead to non-adversarial examples:

- Put $s_j^{(k)}$ in the j-th position of $\tilde{x}_t$ and attain a temporary sample $\tilde{x}_{\text{tem}}^{(k)} = [\tilde{w}_1^{(t)}, \cdots, s_j^{(k)}, \cdots, \tilde{w}_n^{(t)}]$.
- Put $\tilde{x}_{\text{tem}}^{(k)}$ in the victim model f to see if $f(\tilde{x}_{\text{tem}}^{(k)}) \neq y$ still holds.

If $f(\tilde{x}_{\text{tem}}^{(k)}) \neq y$ holds or does not hold for all temporary text samples, we retain the original word $\tilde{w}_j^{(t)}$ in $\tilde{x}_t$ for x'_t. Otherwise, we are given a chance to estimate the prototypes with the sampled K candidates and use the distance measure to find a new substitute for $w_j^{(t+1)}$ in x'_t under the generally held geometric assumption [4, 6] that the semantic change caused by the adversarial attack can be interpreted as the distance from the adversarial example to the decision boundary.

Estimation of Prototypes. In the previous step, w.l.o.g., we know that some temporary samples $S_y = \{s_j^{(1)}, \cdots, s_j^{(m)}\}$ satisfy $f(\tilde{x}_{\text{tem}}^{(k)}) = y$ $(1 \leq k \leq m)$, while other temporary text examples in the set $S_{\neq y} = \{s_j^{(m+1)}, \cdots, s_j^{(K)}\}$ satisfy $f(\tilde{x}_{\text{tem}}^{(k)}) \neq y$ $(m+1 \leq k \leq K)$. The question now turns to how we can estimate the prototypes.

In the metric learning [23, 27], the prototype is estimated by averaging the samples of different classes element-wisely. Thus, we calculate the adversarial prototype $\mathbf{c}_y^{(j)}$ of S_y and non-adversarial prototype $\mathbf{c}_{\neq y}^{(j)}$ of $S_{\neq y}$ by following [23], and

$$\mathbf{c}_y^{(j)} = \frac{1}{m}\sum_{i=1}^{m} \mathcal{H}(s_j^{(i)}) \in \mathbb{R}^h, \mathbf{c}_{\neq y}^{(j)} = \frac{1}{K-m}\sum_{i=m+1}^{K} \mathcal{H}(s_j^{(i)}) \in \mathbb{R}^h, \tag{4}$$

which are the element-wise mean of all word embeddings in S_y and $S_{\neq y}$, respectively.

Replacement Selection. After estimating the adversarial and non-adversarial prototypes by using K candidates, we further employ the distances to them to guide the selection of optimal substitute from $S(w_j)$. Based on the calculated distance, firstly, for each candidate word $s \in S(w_j)$, if it satisfies

$$||\mathcal{H}(s) - \mathbf{c}_{\neq y}^{(j)}|| < ||\mathcal{H}(s) - \mathbf{c}_y^{(j)}||, \tag{5}$$

it means that s is estimated to generate an adversarial example because it is closer to $\mathbf{c}_{\neq y}^{(j)}$ than $\mathbf{c}_y^{(j)}$, where $||\cdot||$ is the L_2 norm, and $\mathcal{H}(s)$ is the word embedding of s. Let $S'(w_j)$ denote all candidate words that satisfy the condition shown in Eq. (5), and we are only interested in the samples that are estimated to be adversarial for replacement selection. For each candidate word $s' \in S'(w_j)$, we harness the projection of $\mathcal{H}(s') - \mathbf{c}_y^{(j)}$ in the direction of $\mathbf{c}_{\neq y}^{(j)} - \mathbf{c}_y^{(j)}$ as the measurement of their impact on the semantic change. The larger is the projection length, the higher is the degree of semantic change. Geometrically, for word $s' \in S'(w_j)$, the distance is

$$d(s') = ||\mathcal{H}(s') - \mathbf{c}_y^{(j)}|| \cdot |\cos(\mathcal{H}(s') - c_y^{(j)}, \mathbf{c}_{\neq y}^{(j)} - \mathbf{c}_y^{(j)})|, \tag{6}$$

where the first term is the projection length between $\mathcal{H}(s')$ and $c_y^{(j)}$ in the direction of $c_y^{(j)}$ and $c_{\neq y}^{(j)}$, and $\cos(\mathbf{a}, \mathbf{b})$ is the cosine value of the angle between vectors $\mathbf{a}$ and $\mathbf{b}$.

Lastly, as shown in Figure 2(d), given the measurement values calculated from Eq. (6) for all suitable candidates in $S'(w_j)$, we rank each candidate in the ascending order of calculated measurement values and select the candidate $s' \in S'(w_j)$ that has the lowest distance value $d(s')$ and makes the text sample satisfy the adversary condition as the word $w_j^{(t+1)}$ for x'_{t+1}. After repeating this process for other words, we can get $x'_{t+1} = [w_1^{(t+1)}, \cdots, w_n^{(t+1)}]$.

Comment. As the geometric assumption indicates, if an adversarial example is close to the decision boundary, it has a small degree of semantic change. Although in the hard-label setting the real decision boundary can never be known due to the limited information, it can be estimated locally around one word as the bisector of line segment between $\mathbf{c}_y^{(j)}$ and $\mathbf{c}_{\neq y}^{(j)}$ for a word. We can find with ease that the distance from the projection of s' into that line segment to the decision boundary is $d(s') - ||\mathbf{c}_y^{(j)} - \mathbf{c}_{\neq y}^{(j)}||/2$, where the second term is a constant. Thus, finding the candidate word with the smallest distance to the estimated decision boundary is the same as finding the candidate word with the smallest $d(s')$.

3.5 Summary

To sum up, given the input x'_t in each iteration, PAT outputs x'_{t+1} using the two-step process introduced in Sec. 3.3 and 3.4. Let R denote the number of changed words in the initialized x'_0 compared with x. After T iterations, PAT conducts at most $T \times R$ local prototype estimation, and will output $x' = x'_T$ as the solution to Eq. (2).

4 EXPERIMENTAL SETTINGS

4.1 Datasets

The experiments are conducted on five text classification datasets, which are commonly used in the text adversarial attack task: **MR** [22], a movie review dataset for binary sentiment classification; **AG** [30], a 4-class news classification dataset; **Yelp** [30], a binary sentiment classification dataset; **Yahoo** [30], a questions-and-answers topic classification dataset with 10 classes; and **IMDB** [17], another binary sentiment classification dataset collected from the IMDB website. In our experiments, we follow the setting of [13, 19, 29], taking the same 1,000 test samples of each dataset to conduct attacks.

4.2 Baselines

To show the effectiveness of the proposed method, we would like to compare PAT with both hard-label adversarial attack methods and score-based ones. The comparison with hard-label ones can directly illustrate the improvement brought by PAT over existing techniques, while the comparison with the score-based ones can show how effective PAT is despite lacking information on prediction score distributions. In the comparison of the **hard-label setting**, the baselines are as follows:

- **Random**, a naive baseline method that replaces each original word with a randomly sampled substitute from the synonym set.
- **TextHoaxer** [29], a hard-label method that uses gradients to guide the selection of word replacement.
- **HLGA** [19], a hard-label text adversarial attack framework that utilizes the genetic algorithm (GA) for optimizing the semantic similarity of adversarial examples.
- **LeapAttack** [28], another gradient-based hard-label method. Different from TextHoaxer, it uses concrete candidate words for calculating the gradient direction for finding substitute words.

As for the comparison between the performance of PAT in the hard-label setting and that of scored-based black-box methods in the **soft-label setting**, we use the state-of-the-art scored-based adversarial attack methods as baselines, including **TextBugger** [16], **PWWS** [25], and **TextFooler** [13].

4.3 Evaluation Metrics

The evaluation of adversarial attack mainly considers the following three aspects [13, 19, 29]:

- *After attack accuracy* (Acc ↓). Since the victim model has the same original accuracy before the adversarial attack, the effect of adversarial attacks is reflected by the accuracy of the victim model after the test samples are perturbed, which is called the after attack accuracy. The lower after attack accuracy indicates better perturbation ability of the adversarial attack method.
- *Semantic similarity* (Sim ↑). The semantic similarity is an important measure of the quality of generated adversarial examples because good adversarial attack methods are supposed to retain most semantic meaning of the original text. Following previous works [13, 19, 29], we use the Universal Sequence Encoder [3] to judge the semantic similarity between the original text sample and generated adversarial example, and the higher the better.
- *Perturbation rate* (Pert ↓). Another property of good adversarial attack methods is fooling the victim models with small perturbations. The perturbation rate is defined as the number of changed words over the number of total words in the adversarial example, and the lower the better.

Note that in the hard-label setting, all the existing approaches [19, 28, 29] will initialize adversarial examples first, which are then optimized by different optimization approaches to generate final adversarial examples. Hard-label baselines and the proposed PAT use the same initialization, which leads to the **same** Acc values for all the methods in the hard-label setting. We mainly rely on Sim and Pert to evaluate the performance between different hard-label adversarial attack methods. As for the comparison with scored-based baselines, we take the Acc metric into consideration in addition to Sim and Pert due to the difference between the two settings.

4.4 Implementation

PAT is implemented with PyTorch on an NVIDIA Tesla P100 GPU. The word embedding space $\mathcal{H}$ is a widely used one called Counter-Fitted Word Vectors [21]. Each word has a 50-word synonym set and is the same as baselines [13, 19, 29]. We set the sampled synonym number $K = 5$ and iteration number $T = 100$. For the baselines, they are implemented with the settings of publicly available codes.

5 EXPERIMENTAL RESULTS

5.1 Comparison with Hard-Label Baselines

Converged Attack Performance. We first show the converged performance of all approaches when attacking against three widely-used text classification models on five datasets in Table 1 to verify

Table 1: Converged performance comparison of semantic similarity (Sim) and perturbation rate (Pert) when attacking against text classification models. Acc stands for after attack accuracy, which is determined by the random initialization step and the same for different hard-label adversarial attack methods. The corresponding original accuracy is in the bracket.

Dataset	Method	BERT			WordCNN			WordLSTM		
		Acc (%)	Sim (%)	Pert (%)	Acc (%)	Sim (%)	Pert (%)	Acc (%)	Sim(%)	Pert (%)
MR	Random	1.0 (← 85.0)	15.2	41.264	0.7 (←76.5)	18.0	39.127	0.7 (← 78.0)	17.2	39.475
	TextHoaxer		67.3	11.812		68.4	12.076		67.7	12.320
	HLGA		64.8	13.360		66.6	12.976		65.4	13.574
	LeapAttack		68.0	10.420		68.6	10.742		68.1	10.959
	PAT		**69.0**	**10.364**		**69.9**	**10.635**		**69.0**	**10.903**
AG	Random	2.8 (← 93.0)	23.0	48.517	1.4 (← 90.4)	30.3	43.494	5.7 (← 90.2)	24.4	47.546
	TextHoaxer		63.8	15.721		74.2	12.493		64.6	16.122
	HLGA		68.9	12.750		78.2	10.250		70.9	12.864
	LeapAttack		69.9	10.852		78.9	8.987		71.8	11.208
	PAT		**72.4**	**9.731**		**81.6**	**7.924**		**74.9**	**9.959**
Yahoo	Random	0.5 (← 79.1)	29.7	30.239	0.8 (← 71.1)	27.6	32.302	1.9 (← 73.7)	24.6	33.291
	TextHoaxer		70.9	6.726		75.0	7.616		67.2	8.396
	HLGA		71.6	5.865		76.2	6.492		68.4	6.999
	LeapAttack		72.0	4.851		78.2	5.629		69.6	5.952
	PAT		**74.3**	**4.382**		**80.1**	**5.110**		**71.1**	**5.586**
Yelp	Random	5.2 (← 96.5)	17.0	39.344	0.6 (← 92.9)	16.6	39.286	3.2 (← 94.8)	17.6	38.966
	TextHoaxer		74.6	9.271		81.3	8.543		80.8	7.942
	HLGA		78.4	7.081		83.8	6.675		83.0	6.166
	LeapAttack		80.5	5.985		86.2	5.845		84.9	5.484
	PAT		**82.9**	**5.246**		**88.0**	**5.329**		**86.3**	**4.974**
IMDB	Random	0.1 (← 90.3)	34.0	30.714	0.0 (← 87.8)	34.2	13.164	0.3 (← 89.3)	33.5	32.020
	TextHoaxer		85.3	4.860		90.2	4.268		89.1	4.098
	HLGA		87.5	3.307		91.3	3.001		90.1	2.916
	LeapAttack		89.3	2.894		92.9	2.881		91.3	2.681
	PAT		**90.7**	**2.299**		**93.7**	**2.613**		**92.3**	**2.526**

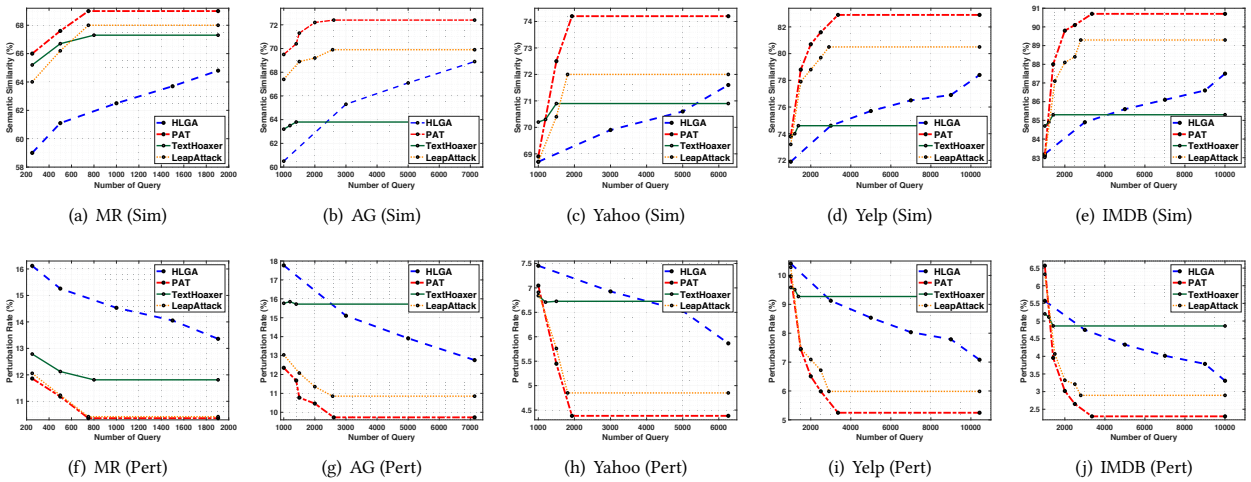
the attack effectiveness. That is, Table 1 demonstrates the performance of different methods when they converge in optimization, which usually reflects the upper bound attack performance of different methods. We can observe that the proposed PAT can generate better adversarial examples with the highest semantic and lowest perturbation rate in the text classification datasets.

Semantic Similarity. PAT can achieve relatively high improvement on semantic similarity over the second best method. Take the scenario of attacking BERT as an example. PAT increases the semantic similarity by 1.0%, 2.5%, 2.3%, 2.4%, and 1.4% compared to the second best method when attacking BERT on the datasets of MR, AG, Yahoo, Yelp and IMDB, respectively. These results show that PAT is effective in improving the semantic similarity in the hard-label setting.

Perturbation Rate. PAT consistently reduces the perturbation rate in different cases. Especially, in the scenario of attacking the AG dataset, the perturbation rate of PAT drops by 1.121%, 1.063% and 1.249% when attacking BERT, WordCNN and WordLSTM, respectively. These results demonstrate that PAT has the ability of recovering the initialized perturbed words to original words, contributing to decreasing the perturbation rate and improving the semantic similarity.

Comparison Analysis. Compared to the baselines, the performance improvement of PAT results from using the estimated adversarial and non-adversrial prototypes to measure the semantic change brought by each candidate word. Compared to the heuristic-based method, PAT expresses discrete substitutes by continuous vectors in the word embedding space, and its optimization is guided by the quantifiable metric with one adversarial candidate in each iteration rather than with a population of them. Compared to the gradient-based methods, PAT does not need to introduce a mapping step between continuous vectors and discrete candidates, and the semantic change that each word will bring is concretely expressed by its distance to the estimated prototypes. As a result, it can choose the candidate that has the smallest impact on the original semantics for keeping the sample adversarial and obtaining higher Sim and lower Pert.

Attack Efficiency. In addition to the converged attack performance, we would like to further compare the attack efficiency between baselines and our method, which can be defined as the quality of generated adversarial examples given the same amount of queries. This property is also important because real-world DNN-based NLP platforms such as Microsoft Azure Text Analytics usually have monthly query limits for each user. Thus, by adopting a method with higher efficiency, we can generate adversarial examples with higher quality even given limited queries. Without loss of generality, we take the results of attacking BERT as an illustration, which is shown in Figure 3. The number of queries for different methods to achieve convergence is the x-axis of the points where each line converges. We report them in Table 4 in the Appendix.



(a) MR (Sim) (b) AG (Sim) (c) Yahoo (Sim) (d) Yelp (Sim) (e) IMDB (Sim)

(f) MR (Pert) (g) AG (Pert) (h) Yahoo (Pert) (i) Yelp (Pert) (j) IMDB (Pert)

Figure 3: (1-5) Semantic similarity (↑) and (6-10) perturbation rate (↓) comparison w.r.t. query number between hard-label baselines and PAT against BERT.

Comparison with Heuristic-Based Ones. We first compare the query number required to achieve convergence between HLGA and PAT. Since HLGA [19] needs to spend repetitive query on the same operation for different adversarial candidates in each iteration, as shown in Figure 3, the number of queries that HLGA takes to achieve convergence is 2.54, 2.74, 3.24, 3.10, and 2.98 times that of PAT in MR, AG, Yahoo, Yelp, and IMDB, respectively, which means that it consumes much more queries than PAT to achieve the convergence. In addition, given different query budget, PAT always achieves better performance than HLGA with better performance.

Comparison with Gradient-Based Ones. Gradient-based methods usually can coverage quickly with a smaller number of budgets. However, the error due to the mapping of the gradient to discrete words can sometimes cause them to get stuck in the local optimum in the long-term perspective. From Table 3, given the same amount of queries, PAT is generally able to achieve higher semantic similarity and lower perturbation rate compared to TextHoaxer and LeapAttack. Therefore, the geometry-aware principle adopted by PAT can help it craft adversarial examples with higher quality more efficiently than the gradient-based ones adopted by the baselines.

Comparison Analysis. From the results above, PAT generally gains better adversarial attack performance given the same amount of query consumption compared to the hard-label baselines. Again, such a good property is attributed to the following two factors. Firstly, PAT maintains only one adversarial candidate each iteration, and thus, it reduces repetitive consumption on queries compared to population-based methods. Besides, using estimated prototypes is useful for selecting better substitution candidates in each step compared to the employment of gradients.

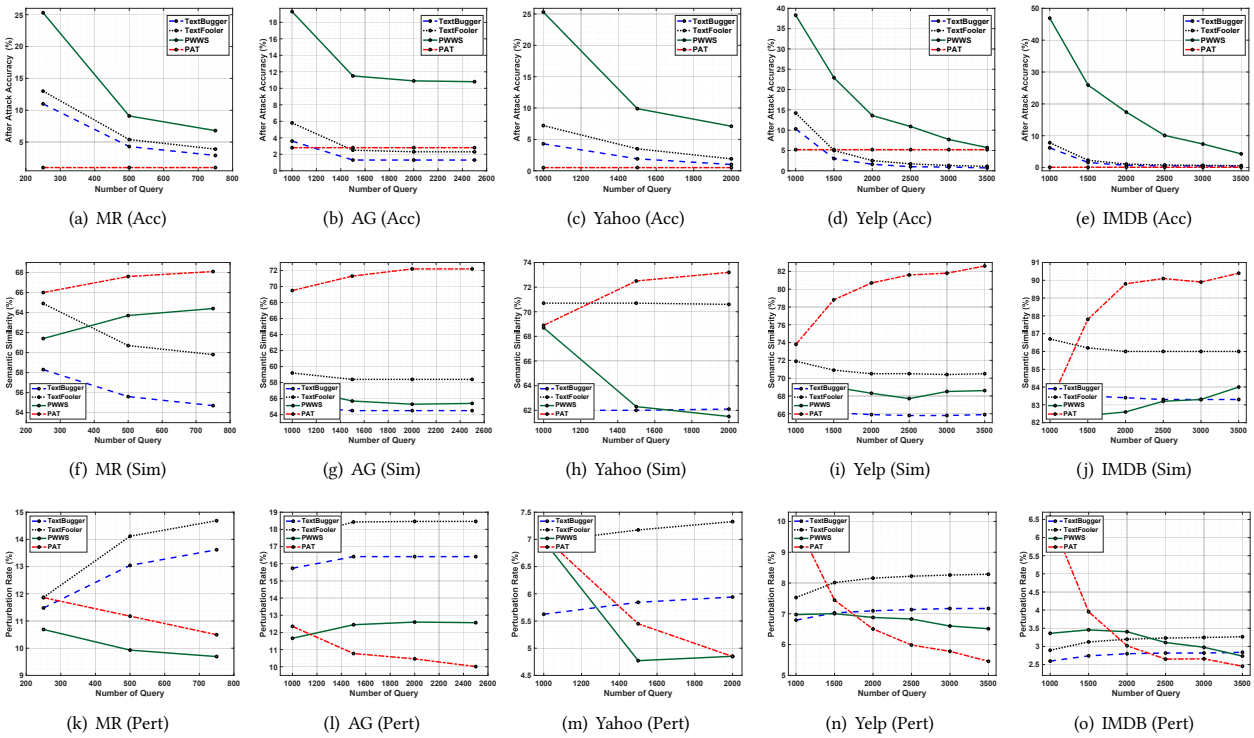
Attack Real-World APIs. Note that due to the space limit, we leave the discussion on the results of attacking real-world APIs in Sec. A.2 in the Appendix.

5.2 Comparison with Score-Based Baselines

We further compare the adversarial attack results of PAT in the hard-label setting with those of score-based baselines in the soft-label setting. In this comparison, score-based baselines can access the prediction score distribution over each category, which is more informative than the top-1 prediction used by PAT in the hard-label setting. Such a comparison can help us gain a deeper understanding of the effectiveness of the hard-label adversarial attack conducted by PAT despite the loss of prediction score information.

Comparison Setting. Without the loss of generality, we show the comparison results between PAT and score-based baselines on text classification datasets in Figure 4 on attacking the BERT model for discussion. For a fair comparison between two settings, for each method, we report its performance on Acc, Sim, and Pert in each dataset given a certain amount of queries. The range of query amount for each dataset differs, which depends on the scale of input texts. For instance, the maximum amount of queries for datasets with short text like MR is 750, while that is 3,500 for datasets with long text like Yelp and IMDB.

Performance Analysis. As shown in Figure 4, in different cases, PAT keeps the after attack accuracy of the victim model the same given the different amounts of queries because the discovery of adversarial examples is decided by the text random initialization step and it does not need many queries. After that, with the increase of query budgets, PAT keeps improving the quality of adversarial examples by enhancing the semantic similarity and lowering the perturbation rate. Most importantly, we observe that even though the score-based baselines can access more information for generating adversarial examples, PAT is able to generally outperform them such as in the cases of attacking the victim model on the Yahoo and IMDB datasets with respect to Acc, Sim and Pert after the query budget increases. The superior performance illustrates that the restriction of only having top-1 prediction results does not matter much to PAT, and it is capable of crafting adversarial examples



(a) MR (Acc) (b) AG (Acc) (c) Yahoo (Acc) (d) Yelp (Acc) (e) IMDB (Acc)

(f) MR (Sim) (g) AG (Sim) (h) Yahoo (Sim) (i) Yelp (Sim) (j) IMDB (Sim)

(k) MR (Pert) (l) AG (Pert) (m) Yahoo (Pert) (n) Yelp (Pert) (o) IMDB (Pert)

Figure 4: (1-5) After attack accuracy (↓), (6-10) semantic similarity (↑) and (11-15) perturbation rate (↓) comparison w.r.t. query number between score-based methods and PAT against BERT.

of high quality in the more difficult hard-label setting, which are comparable with the ones generated by score-based baselines.

Distance v.s. Score Distribution. The problem with score-based baselines is that they usually adopt greedy mechanisms and need to try every candidate word for finding the locally best candidate word. Therefore, they consume much more queries for finding a good candidate word than ours, which causes them not to have enough queries for discovering more good candidate words in this setting. The results above suggest that the distance calculated based on the estimated prototypes in PAT is as effective as the prediction score distribution used by score-based baselines in selecting synonyms for replacement. Although in the hard-label setting PAT cannot utilize the prediction scores to find their influence on semantics as the score-based baselines do, it uses the distances between synonyms and the estimated prototypes as the measure for the influence on the semantic meaning. As a result, PAT can generate high-quality adversarial examples despite the less instructive information.

5.3 Model Insight Analysis

Previous experimental results clearly demonstrate the effectiveness of PAT. Next, we use the following two experiments to verify the design of our method. Without the loss of generality, we conduct the experiments by attacking WordCNN on five text classification datasets. Due to the space limit, we show the case on AG as an illustration, and similar results are seen in other datasets.

Table 2: Ablation study results on module design.

Dataset	Variant	Sim (%)	Pert (%)
AG	w/o Replacing	81.0	11.647
	w/o Selecting (Random)	70.9	13.077
	w/o Selecting (Most Similar)	68.8	14.541
	PAT	**81.6**	**7.924**

Table 3: Influence of K on results.

Dataset		AG	
Metric		Sim (%)	Pert (%)
K	3	80.8	7.977
	5	**81.6**	**7.924**
	7	**81.6**	7.965
	10	81.2	8.146

Ablation Study. Firstly, we conduct an ablation study to substantiate the optimization design as shown in Table 2. The baselines in this ablation study are two variants of PAT. For the first variant (w/o Replacing), it does not include the step of replacing original words back in each iteration as illustrated in Sec. 3.3. In the second variant (w/o Selecting), it either randomly chooses a synonym for substitution (Random) or the most similar synonym (Most Similar) with the original one in the embedding space, rather than using

the estimated prototypes to decide the substitute. Compared with the first variant, we find that the perturbation rate will increase if we do not replace the original words back, which shows that the step of replacing the original words back in each iteration is helpful for reducing the perturbation rate. As for the comparison with the second variant, it shows that selecting replacement by prototype estimation is the key to the designed method. If we do not select replacement as PAT does, the semantic similarity will largely reduce, and in contrast, the perturbation rate will greatly increase. Thus, the designed process in PAT is reasonable and effective.

Hyperparameter Selection. Another study is on the influence of the hyperparameter K, i.e., the number of samples for prototype estimation, on the adversarial attack performance. We change K from 3 to 10 and show the experimental results in Table 3. We can observe that K does not have a significant influence on the adversarial attack performance with respect to semantic similarity and perturbation rate. These results manifest that the design of prototype estimation in PAT can maintain stable adversarial attack performance with different K, indicating its insensitivity on K for obtaining good performance in hard-label adversarial attacks. In addition, Table 3 shows that selecting $K = 5$ can generally help us obtain the best performance with a small number of queries. Therefore, we set $K = 5$ in our experiments.

6 RELATED WORK

6.1 Image Adversarial Attack

The adversarial attack is a task that can measure the robustness of DNNs, which becomes a research focus after researchers find that DNNs are prone to change their prediction given small perturbation on original images in the image classification task [1, 2, 9, 15]. The generation of image adversarial examples in the early stage relies on parameter gradients, which is called the "white-box" setting. Representative works include Fast Gradient Sign Method (FGSM) [9], Projected Gradient Descent (PGD) [18] attack, and C&W attack [2]. After that, researchers [4, 15] gradually restrict the information that the attackers can access and design the image adversarial attack method in the hard-label black-box setting, where the attackers only have the knowledge of the top-1 predicted label from the victim model. For instance, Sign-OPT [4] reformulates the hard-label image adversarial attack problem as a gradient-based optimization problem that aims to find the direction in the high-dimensional pixel space leading to adversarial images with the highest similarity. Another exemplary work is HopJumpSkipAttack [4], which estimates the gradients for optimizing the existing adversarial image near the decision boundary with the Monte Carlo method.

6.2 Text Adversarial Attack

Text adversarial attack [8, 13, 16, 19, 24, 25, 29] is generally the task of conducting adversarial perturbation on text by replacing original words with their synonyms and forcing victim models to change their correct predictions into incorrect ones. Similar to the development in image adversarial attack, text adversarial attack is conducted from the **white-box setting** [8, 10] to the more realistic black-box setting. A pioneering white-box method named HotFlip [8] constructs text adversarial examples by the gradient information of one-hot input vectors. Later, recent methods are mostly proposed in the black-box setting.

The first type of black-box setting is the **soft-label** (or score-based) one, where the attackers can access the prediction score distribution of all categories given an input text sample, and the representative methods include TextBugger [16], PWWS [25], and TextFooler [13]. Since the soft-label score distribution is provided, attacks of this type usually use the greedy mechanism to decide which word to be replaced and which synonym is used to replace the original word. For example, TextBugger [16] uses the soft-label saliency score for position ranking and substitution.

Another type of black-box setting is the **hard-label** (or decision-based) one [19, 26, 28, 29], which is more realistic and difficult than the soft-label one because attackers only know the top-1 predicted label output by the model. The first successful work [19] for this setting is proposed recently, which uses a genetic algorithm (GA) to optimize the quality of adversarial examples. Its limitation is that GA needs to maintain many adversarial candidates and update them by mutation and crossover in each iteration, which only implicitly expresses the influence on the semantic meaning of different replacements as the survival of candidates. After that, [29] and [28] try to make the gradient-based method workable in the hard-label setting by introducing the word embedding space to guide the adversarial example construction. However, their gradient estimation is based on the assumption that text adversarial perturbation obeys Gaussian distribution as image one does, which is unsuitable for modeling the discrete distribution of candidate words.

Different from existing methods, we introduce prototype estimation to explicitly characterize the semantic meaning change brought by different candidate words for replacement selection. This method can avoid the step of mapping continuous vectors to discrete words and maintain one adversarial candidate in each iteration. Thus, it can generate adversarial examples of better quality.

7 CONCLUSION

Hard-label black-box adversarial attack on text is more challenging because attackers can only access the top-1 prediction results. Existing methods are limited by the lack of an effective metric for measuring the semantic meaning change brought by different substitute candidates during perturbation. To relieve this conundrum, this paper proposes a geometry-aware iterative solution named PAT that designs a semantic change measure as the distance from the candidate word to the estimated prototypes obtained from the metric learning principle in a pre-trained word embedding space. In each iteration, it firstly lowers the perturbation rate of the previous adversarial candidate by replacing the original words back, and then selects better substitutes for the remaining perturbed words based on prototype estimation. Experimental results show that PAT outperforms existing hard-label adversarial attack methods, whose performance is comparable with those of score-based methods applied in the more informative soft-label setting. In addition, PAT can successfully attack three industry-leading APIs.

ACKNOWLEDGMENTS

This work is partially supported by the National Science Foundation under Grant No. 2212323, 2119331, 1951729, and 1953893.

REFERENCES

[1] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *ICLR*. OpenReview.net.

[2] Nicholas Carlini and David A. Wagner. 2017. Towards Evaluating the Robustness of Neural Networks. In *S&P*. IEEE Computer Society, 39–57. https://doi.org/10.1109/SP.2017.49

[3] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Céspedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[4] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. 2020. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. In *S&P*. IEEE, 1277–1294. https://doi.org/10.1109/SP40000.2020.00045

[5] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, Jinfeng Yi, and Cho-Jui Hsieh. 2019. Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach. In *ICLR*. OpenReview.net.

[6] Minhao Cheng, Simranjit Singh, Patrick H. Chen, Pin-Yu Chen, Sijia Liu, and Cho-Jui Hsieh. 2020. Sign-OPT: A Query-Efficient Hard-label Adversarial Attack. In *International Conference on Learning Representations*.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT*. Association for Computational Linguistics, 4171–4186. https://doi.org/10.18653/v1/n19-1423

[8] Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2018. HotFlip: White-Box Adversarial Examples for Text Classification. In *ACL*. Association for Computational Linguistics, 31–36. https://doi.org/10.18653/v1/P18-2006

[9] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and Harnessing Adversarial Examples. In *ICLR*.

[10] Chuan Guo, Alexandre Sablayrolles, Hervé Jégou, and Douwe Kiela. 2021. Gradient-based Adversarial Attacks against Text Transformers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 5747–5757.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[12] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[13] 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 *AAAI*. AAAI Press, 8018–8025.

[14] Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP*. ACL, 1746–1751. https://doi.org/10.3115/v1/d14-1181

[15] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. 2017. Adversarial examples in the physical world. In *ICLR*. OpenReview.net.

[16] Jinfeng Li, Shouling Ji, Tianyu Du, Bo Li, and Ting Wang. 2019. TextBugger: Generating Adversarial Text Against Real-world Applications. In *NDSS*.

[17] 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 *ACL*. The Association for Computer Linguistics, 142–150.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=rJzIBfZAb

[19] Rishabh Maheshwary, Saket Maheshwary, and Vikram Pudi. 2021. Generating Natural Language Attacks in a Hard Label Black Box Setting. In *AAAI*.

[20] Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On Evaluation of Adversarial Perturbations for Sequence-to-Sequence Models. In *NAACL-HLT 2019*. Association for Computational Linguistics, 3103–3114. https://doi.org/10.18653/v1/n19-1314

[21] Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Lina Maria Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2016. Counter-fitting Word Vectors to Linguistic Constraints. In *NAACL HLT 2016*. The Association for Computational Linguistics, 142–148.

[22] Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *ACL*. The Association for Computer Linguistics, 115–124. https://doi.org/10.3115/1219840.1219855

[23] Mikhail Pautov, Olesya Kuznetsova, Nurislam Tursynbek, Aleksandr Petiushko, and Ivan Oseledets. 2022. Smoothed Embeddings for Certified Few-Shot Learning. *Neurips* (2022).

[24] Fanchao Qi, Yangyi Chen, Xurui Zhang, Mukai Li, Zhiyuan Liu, and Maosong Sun. 2021. Mind the Style of Text! Adversarial and Backdoor Attacks Based on Text Style Transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 4569–4580.

[25] Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating Natural Language Adversarial Examples through Probability Weighted Word Saliency. In *ACL*. Association for Computational Linguistics, 1085–1097. https://doi.org/10.18653/v1/p19-1103

[26] Sachin Saxena. 2020. Textdecepter: Hard label black box attack on text classifiers. *arXiv preprint arXiv:2008.06860* (2020).

[27] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. 2016. A discriminative feature learning approach for deep face recognition. In *European conference on computer vision*. Springer, 499–515.

[28] Muchao Ye, Jinghui Chen, Chenglin Miao, Ting Wang, and Fenglong Ma. 2022. LeapAttack: Hard-Label Adversarial Attack on Text via Gradient-Based Optimization. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2307–2315.

[29] Muchao Ye, Chenglin Miao, Ting Wang, and Fenglong Ma. 2022. TextHoaxer: Budgeted Hard-Label Adversarial Attacks on Text. *AAAI* (2022).

[30] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In *NeurIPS*. 649–657.

A APPENDIX

A.1 Pseudocodes

The pseudocode of the PAT framework, how it replaces original words back, and how it selects replacement by prototype estimation, are shown in Algorithm 1, Algorithm 2, and Algorithm 3, respectively.

A.2 Attacking Real-World APIs

To verify the effectiveness of PAT, we further employ three widely used real-world NLP APIs as victim models in the hard-label black-box setting, including Google Cloud NLP, Amazon Comprehend, and Microsoft Azure Text Analytics. Due to the limited service budget provided for each free account, we only put 100 samples of the MR dataset into each real-world NLP system to conduct adversarial attacks, and the results are shown in Table 5. We can observe that the accuracy before and after adversarial attacks changes dramatically, which indicates that the majority of text samples correctly classified by the real-world semantic classification APIs can be successfully perturbed by PAT. Moreover, as measured by the semantic similarity and perturbation rate, Table 5 illustrates that PAT can attain adversarial examples with relatively high quality.

Furthermore, to illustrate, we list some generated adversarial text examples against Google Cloud NLP, Amazon Comprehend, and Microsoft Azure Text Analytics in Table 6, Table 7 and Table 8, respectively. Even for real-world NLP APIs, PAT is able to fool them by making small perturbations. For instance, for the first adversarial example in Table 6, the victim model (Google Cloud NLP) changes the prediction from *positive* to *negative* due to the perturbation of changing "intense" to "forceful". We can also verify our observation with other instances such as the first adversarial example in Table 7 as an instance. PAT changes the prediction of Amazon Comprehend by just replacing the word "legendary" by "memorable", which shows the effectiveness of PAT in exploring the robustness of real-world DNN systems in the text domain.

Algorithm 1: Hard-Label Black-Box Adversarial Attack by PAT

Inputs: Text sample to be attacked $x = [w_1, \cdots, w_n]$, victim model f, word embedding space $\mathcal{H}$, iteration number T, and sample number K.
Output: Adversarial example x^*.
Set $x'_0 \leftarrow x$, set the word counter $i \leftarrow 1$;
while $i \leq n$ **do**
  Randomly select a synonym $w_i^{(0)}$ in the synonym set $\mathbf{S}(w_i)$ and replace w_i by $w_i^{(0)}$ in x'_0;
  $i \leftarrow i + 1$;
  if $f(x'_0) \neq f(x)$ **then**
    Set the iteration counter $t \leftarrow 0$;
    while $t \leq$ *T-1* **do**
      Use Alogirthm 2 to replace original words back;
      Use Algorithm 3 to select better replacements for perturbation;
    end
    Return the adversarial example having the highest semantic similarity with x;
  end
end
Return Null;

Algorithm 2: Replacing Original Words Back

Inputs: Adversarial example $x'_t = [w_1^{(t)}, \cdots, w_n^{(t)}]$, victim model f, semantic similarity calculation function Sim.
Output: Adversarial example $\tilde{x}_t$.
for each word $w_i^{(t)} \in x'_t$ **do**
  if $w_i^{(t)} \neq w_i$ **then**
    Replace $w_i^{(t)}$ by w_i and obtain a temporary sample $x_{\text{tem}}^{(i)} = [w_1^{(t)}, \cdots, w_i, \cdots, w_n^{(t)}]$;
    if $x_{\text{tem}}^{(i)}$ satisfies $f(x_{\text{tem}}^{(i)}) \neq y$ **then**
      Calculate the semantic similarity improvement $sim_i = \text{Sim}(x, x_{\text{tem}}^{(i)}) - \text{Sim}(x, x'_t)$;
    end
  end
end
Rank the order for replacing each w_i back in the corresponding descending order of set $\{sim_i\}$;
Initialize $\tilde{x}_t \leftarrow x'_t$;
while replacing w_i back keeps $\tilde{x}_t$ adversarial **do**
  Replace w_i back to $\tilde{x}_t$;
end
Return $\tilde{x}_t$;

Algorithm 3: Selecting Replacement by Prototype Estimation

Inputs: Adversarial candidate $\tilde{x}_t = [\tilde{w}_1^{(t)}, \cdots, \tilde{w}_n^{(t)}]$, victim model f, word embedding space $\mathcal{H}$, and sample number K.
Output: Adversarial candidate x'_{t+1}.
Initialize $x'_{t+1} \leftarrow \tilde{x}_t$;
for any word $\tilde{w}_j^{(t)} \neq w_i$ in $\tilde{x}_t$ **do**
  Sample K candidate words $\{s_j^{(1)}, \cdots, s_j^{(K)}\}$ from the synonym set $S(w_j)$;
  for each sampled word $s_j^{(k)}$ **do**
    Fix other words and replace $\tilde{w}_j^{(t)}$ with $\tilde{s}_j^{(k)}$ to attain a temporary sample $\tilde{x}_{\text{tem}}^{(k)}$;
  end
  if there are some $\tilde{x}_{\text{tem}}^{(k)}$ that satisfies $f(\tilde{x}_{\text{tem}}^{(k)}) = y$ and some that does not **then**
    Calculate centers $\mathbf{c}_y^{(j)}$ and $\mathbf{c}_{\neq y}^{(j)}$ from Eq. (4);
    Select a new replacement $w_j^{(t+1)}$ for $\tilde{w}_j^{(t)}$ based on the distance measured by Eq. (6) for x'_{t+1};
  else
    $w_j^{(t+1)} \leftarrow \tilde{w}_j^{(t)}$
  end
end
Return x'_{t+1};

Table 4: Number of queries for different methods to achieve convergence in optimization when attacking BERT.

Method	MR	AG	Yahoo	Yelp	IMDB
TextHoaxer	800	1,400	1,500	1,381	1,416
HLGA	1,905	7,137	6,263	10,431	10,004
LeapAttack	750	2,572	1,812	2,908	2,795
PAT	750	2,607	1,932	3,360	3,354

Table 5: Results of attacking real-world semantic classification APIs with 100 samples randomly selected the MR dataset (accuracy inside each bracket is the one before adversarial attacks).

Model	Acc (%)	Sim (%)	Pert (%)
Google Cloud NLP	0.0 (← 74.0)	67.2	10.129
Amazon Comprehend	1.0 (← 69.0)	61.9	10.835
Microsoft Azure	6.0 (← 71.0)	63.9	11.235

Table 6: Generated adversarial text samples by PAT against Google Cloud NLP. The substituted original word is strikethroughed, and the replacement is the following one.

Adversarial Example	Prediction Change
An ~~intense~~ forceful and effective film about loneliness and the chilly anonymity of the environments where so many of us spend so much of our time	Positive → Negative
Broomfield is ~~energized~~ inflamed by Volletta Wallace's maternal fury, her fearlessness, and because of that, his film crackles	Positive → Negative
The holes in this film ~~remain~~ stays agape holes punched through by an ~~inconsistent~~ unsound, meandering, and sometimes dry plot	Negative → Positive

Table 7: Generated adversarial text sample by PAT against Amazon Comprehend. The substituted original word is strikethroughed, and the replacement is the following one.

Adversarial Example	Change of Prediction
The first mistake, I suspect, is casting Shatner as a ~~legendary~~ memorable professor and Kunis as a brilliant college student. Where's Pauly Shore as the rocket scientist	Negative → Positive
The ~~central character~~ vital persona isn't complex enough to hold our interest	Negative → Positive
A ~~formula~~ design family tearjerker told with a heavy Irish brogue accentuating, rather than muting, the plot 's saccharine thrust	Negative → Positive

Table 8: Generated adversarial text sample by PAT against Microsoft Azure Text Analytics. The substituted original word is strikethroughed, and the replacement is the following one.

Adversarial Example	Change of Prediction
A profoundly ~~stupid~~ goofball affair, populating its hackneyed and meanspirited storyline with cardboard characters and performers who value cash above credibility	Negative → Positive
The central character isn't ~~complex~~ fraught enough to hold our interest	Negative → Positive
Parker holds true to wilde's own vision of a pure comedy with absolutely no meaning, and no desire to be anything but a polished, sophisticated entertainment that is in ~~love~~ sweetness with its own ~~cleverness~~ acumen	Positive → Negative