Self-Supervised Contrastive Learning with Adversarial Perturbations for Defending Word Substitution-based Attacks

Zhao Meng∗ Yihan Dong∗ Mrinmaya Sachan Roger Wattenhofer
ETH Zurich, Switzerland
{zhmeng, yihdong, wattenhofer}@ethz.ch
mринmaya.sachan@inf.ethz.ch

Abstract

In this paper, we present an approach to improve the robustness of BERT language models against word substitution-based adversarial attacks by leveraging adversarial perturbations for self-supervised contrastive learning. We create a word-level adversarial attack generating hard positives on-the-fly as adversarial examples during contrastive learning. In contrast to previous works, our method improves model robustness without using any labeled data. Experimental results show that our method improves robustness of BERT against four different word substitution-based adversarial attacks, and combining our method with adversarial training gives higher robustness than adversarial training alone. As our method improves the robustness of BERT purely with unlabeled data, it opens up the possibility of using large text datasets to train robust language models against word substitution-based adversarial attacks.

1 Introduction

Pretrained language models such as BERT (Devlin et al., 2019, inter alia) have had a tremendous impact on many NLP tasks. However, several researchers have demonstrated that these models are vulnerable to adversarial attacks, which fool the model by adding small perturbations to the model input (Jia and Liang, 2017).

A prevailing method to improve model robustness against adversarial attacks is adversarial training (Madry et al., 2018). In NLP, adversarial training in the input space has been challenging, as existing natural language adversarial attacks are too slow to generate adversarial examples on the fly during training (Alzantot et al., 2018; Ebrahimi et al., 2018; Ren et al., 2019). While some recent works (Wang et al., 2021c) have started exploring efficient input space adversarial training (e.g., for text classification), scaling adversarial training to pretrained language models like BERT has been challenging.

In this work, we in particular focus on improving the robustness of BERT against word substitution-based adversarial attacks. We propose an approach to adversarially finetune BERT-like models without using any labeled data. In order to achieve this, we rely on self-supervised contrastive learning (Chen et al., 2020). Self-supervised contrastive learning has recently gained attention in the community and contrastive learning has been used to learn better representations for text classification (Giorgi et al., 2021; Kim et al., 2021; Gao et al., 2021). However, how to use these methods to improve model robustness remains an open question.

We combine self-supervised contrastive learning with adversarial perturbations by using adversarial attacks to generate hard positive examples for contrastive learning. To efficiently create adversarial examples, we leverage an adversarial attack, that is capable of generating multiple adversarial examples in parallel. The attack adversarially creates hard positive examples for contrastive learning by iteratively replacing words to follow the direction of the contrastive loss (see fig. 2).

Experiments show that our method can improve the robustness of pretrained language models without looking at the labels (in other words, before finetuning). Additionally, by combining our method with adversarial training, we are able to obtain better robustness than conducting adversarial training alone (see section 4.4). Our study of the vector representations of clean examples and their corresponding adversarial examples indeed explains that our method improves model robustness by pulling clean examples and adversarial examples closer.

Our contributions1 in this paper are two-fold. On the one hand, we improve the robustness of

1We will release our code at https://github.com/LotusDYH/ssl_robust
the pretrained language model BERT against word substitution-based adversarial attacks by using self-supervised contrastive learning with adversarial perturbations (see section 3.2). On the other hand, to facilitate adversarial self-supervised contrastive learning, we create for BERT a word-level adversarial attack to create hard positive examples. The attack makes contrastive learning and adversarial training with on-the-fly generated adversarial examples possible. Additionally, we also show that our method is capable of using out-of-domain data to improve model robustness (see table 2 and section 4.4). This opens an opportunity for using large-scale unlabeled data to train robust language models against word substitution-based adversarial attacks.

2 Related Work

2.1 Adversarial Training for NLP

Adversarial training improves model robustness by augmenting clean examples with adversarial examples during training. Previous works on adversarial training for natural language mainly focus on perturbations in the vector space, while actual adversarial attacks create adversarial examples by changing natural language symbols. For example, Zhu et al. (2020) and Liu et al. (2020) improve model generalization ability by adversarial training on the word embedding space, without mentioning model robustness. However, they either ignore model robustness, or only test model robustness against the adversarial dataset ANLI, without paying attention to actual adversarial attacks. Other works conduct adversarial training in the word space (Alzantot et al., 2018; Ren et al., 2019). Still, they can only do adversarial training on a limited number of pre-generated adversarial examples due to the low efficiency of the attacks. A recent work (Wang et al., 2021c) conducts adversarial training efficiently in the word space, but their method is limited to non-contextualized models.

Apart from adversarial training, other supervised learning methods (Dong et al., 2021; Zhou et al., 2021; Wang et al., 2021a; Li and Qiu, 2020) have also been proposed to improve robustness. However, these methods are supervised and are not comparable to our work.

Our work also differs from previous works in natural language adversarial training. On the one hand, as opposed to previous works, which are supervised, we propose a self-supervised learning scheme to improve the robustness of pretrained language models. On the other hand, while previous works mostly focus on adversarial training in embedding space, we conduct efficient adversarial training with pretrained language models at the word level.

2.2 Contrastive Learning for NLP

Contrastive learning was first proposed in the image domain to improve model performance in a self-supervised fashion (He et al., 2020; Chen et al., 2020). These methods bring representations of similar examples closer and push representations of dissimilar examples further apart. Additionally, researchers also find that by adding adversarial perturbations during contrastive learning, image classification models become more robust against adversarial attacks (Kim et al., 2020).

In NLP, previous works on contrastive learning mainly focus on improving model generalization. Gunel et al. (2021) boost performance of RoBERTa by adding supervised signals during fine-tuning on downstream tasks. Lee et al. (2021) tackle the “exposure bias” problem in text generation by adding adversarial signals during contrastive learning. Other similar works include Pan et al. (2021), Giorgi et al. (2021), and Gao et al. (2021). Although these works have demonstrated the usefulness of contrastive learning in NLP applications, few have addressed the robustness of NLP models, particularly pretrained language models, against word substitution-based natural language adversarial attacks.

Recently, Wang et al. (2021b) claimed that their method improves model robustness against adversarial sets. However, such sets are pre-generated and are less challenging than adversarial examples generated on the fly by actual adversarial attacks (Jin et al., 2020; Ren et al., 2019). In this paper, we focus on improving the robustness of pretrained language models against word substitution-based adversarial attacks. We present the details of our method in section 3.

3 Methodology

In this section, we describe our method for self-supervised contrastive learning with adversarial perturbations. Specifically, section 3.1 gives the background and motivation of our problem, and section 3.2 describes the adversarial contrastive learning framework. Finally, in section 3.3, we
describe the adversarial attack used in contrastive learning.

3.1 Background and Motivation

In this work, we focus on text classification tasks\(^2\). Let us assume that we have an example text \(X_i = \{w_1, w_2, \ldots, w_L\}\) with \(L\) words and let \(y_i\) be the corresponding class label for \(X_i\). Our text classification model consists of a BERT encoder \(f(\cdot)\) and an MLP classification head \(c(\cdot)\).

We obtain the vector representation \(h_i \in \mathbb{R}^d\) of the example \(X_i\) by feeding \(X_i\) into the BERT encoder \(f(\cdot)\). Then the MLP classification head \(c(\cdot)\) takes \(h_i\) as input to give us the prediction. Formally, we have:

\[
\begin{align*}
  h_i &= f(X_i) \\
  \hat{y}_i &= c(h_i)
\end{align*}
\]

where \(\hat{y}_i\) is the predicted label. We have \(\hat{y}_i = y_i\) if the model prediction is correct.

A word substitution-based adversarial attack \(a(\cdot)\) takes an original example \(X_i\) as input and generates an adversarial example \(X_i^{adv}\) by substituting the \(k\)-th original word \(w_k\) in \(X_i\) with another word \(w_k^{adv}\). To make the original example \(X_i\) and the adversarial example \(X_i^{adv}\) close in semantics, existing works often use synonyms as substitutions (Ren et al., 2019; Morris et al., 2020).

By conducting the word substitution, the attack \(a(\cdot)\) aims to fool the model with \(X_i^{adv}\). Formally, we have:

\[
\begin{align*}
  X_i^{adv} &= a(X_i) \\
  h_i^{adv} &= f(X_i^{adv}) \\
  \hat{y}_i^{adv} &= c(h_i^{adv})
\end{align*}
\]

where \(X_i^{adv} = \{w_1, w_2, \ldots, w_k^{adv}, \ldots, w_L\}\), \(1 \leq k \leq L\). Assuming the attack successfully fools the model, we would have \(\hat{y}_i \neq \hat{y}_i^{adv}\). The key assumption in our approach is that although \(X_i\) and \(X_i^{adv}\) are very similar to each other at the word level, it is possible that the encoder \(f\) embeds them in such a way that the distance between their representations \(h_i\) and \(h_i^{adv}\) are large and the classification head \(c(\cdot)\) predicts \(X_i\) and \(X_i^{adv}\) to be of different classes.

Thus, the goal of our method is to obtain a robust model, on which we have \(y_i = \hat{y}_i\) and \(\hat{y}_i = \hat{y}_i^{adv}\). In other words, the robust model defends an adversarial example \(X_i^{adv}\) of the original example \(X_i\) successfully, if the robust model gives the same correct prediction on the original example \(X_i\) and the adversarial example \(X_i^{adv}\). We use attack success rate as the evaluation metric for model robustness.

The attack success rate is defined as the rate of an attack successfully fooling the model on all test examples.

To obtain a robust model, we optimize the encoder such that \(h_i\) and \(h_i^{adv}\) become similar to each other. We achieve this goal by conducting
self-supervised contrastive learning on the encoder with adversarial perturbations, during which we use an attack to create hard positive examples, maximizing the contrastive loss. The rest this section gives the details of our method.

### 3.2 Self-Supervised Contrastive Learning with Adversarial Perturbations

Following previous works on self-supervised contrastive learning (He et al., 2020; Chen et al., 2020), we formulate our learning objectives as follows. Consider we have a batch of \( n \) examples and \( X_i \) is the \( i \)-th input, we first obtain \( X'_i = t(X_i) \) as an augmentation of \( X_i \) by transformation \( t(\cdot) \). We call \( X_i \) and \( X'_i \) a pair of positive examples. All other examples in the same batch are considered negative examples of \( X_i \) and \( X'_i \).

To take advantage from using more negative examples, we use MoCo (He et al., 2020) as our framework, in which we employ an encoder \( f_q \) for the positive examples, and another momentum encoder \( f_k \) for the negative examples. We then have:

\[
\begin{align*}
th_i &= f_q(X_i) \\
h'_i &= f_k(X'_i)
\end{align*}
\]

where \( h_i, h'_i \in \mathbb{R}^d \) are representations of \( X_i \) and \( X'_i \), respectively. During training, \( f_q \) and \( f_k \) are initialized the same. We update \( f_k \) momentarily:

\[
\theta_k \leftarrow m \cdot \theta_k + (1 - m) \cdot \theta_q
\]

where \( \theta_k \) and \( \theta_q \) denote the parameters of \( f_k \) and \( f_q \), respectively. We then have:

\[
\begin{align*}
z_i &= g_q(h_i) \\
z'_i &= g_q(h'_i)
\end{align*}
\]

where \( z_i, z'_i \in \mathbb{R}^e \), \( g_q(\cdot) \) and \( g_k(\cdot) \) are MLPs with one hidden layer of sigmoid activation, respectively. Following Chen et al. (2020), we conduct contrastive learning on \( z \) instead of \( h \) to prevent the contrastive learning objective from removing information useful for downstream tasks. After contrastive learning, we use \( h \) as the sentence representation for downstream tasks.

Additionally, we also maintain a dynamic first-in-first-out queue for the negative examples. During training, before computing contrastive loss at the end of each batch, all encoded examples of the current batch are enqueued into the queue, and the oldest examples are dequeued simultaneously.

In our experiments, we use the attack described in section 3.3 or back-translation (Zhu et al., 2015) for augmentation \( t(\cdot) \). Assume that we have an encoded example \( z_i \) and the encoded examples in the queue are \( \{z_0, z_1, \ldots, z_{Q-1}\} \), where \( Q \) is the size of the queue. Among the encoded examples in the queue, one of them is \( z'_i \), which forms a pair of positive examples with \( z_i \). We use contrastive loss to maximize the similarity between positive examples, while minimizing the similarity of negative examples. We then have:

\[
\ell_i = - \log \frac{\exp(\text{sim}(z_i, z'_i)/\tau)}{\sum_{k=0}^{Q-1} \exp(\text{sim}(z_i, z_k)/\tau)}
\]

(1)

where \( \tau \) is the temperature parameter, \( \text{sim}(\cdot, \cdot) \) is the similarity function, and \( Q \) is the size of the dynamic queue. In this paper, we compute similarity by dot product as in MoCo.

By optimizing eq. (1), the goal is to maximize the similarity of representations between similar (positive) pairs of examples while minimizing the similarity of representations between dissimilar (negative) examples. We use the geometry-inspired attack described in section 3.3 as the transformation \( t(\cdot) \) to create pairs of examples that are similar on the word level but at the same time are distant from each other in the representation space.

We illustrate our method in fig. 1. In fig. 1 (b) and (c), by conducting contrastive learning and using the Geometry Attack generated adversarial examples as hard positives, the vector representations obtained from the model become invariant to the adversarial attacks.

![Figure 2: An illustration of one iteration in Geometry Attack for contrastive loss. See section 3.3 for details.](image)

### 3.3 Creating Hard Positive Examples by Geometry Attack

As mentioned in section 3.2, we use an attack as the transformation \( t(\cdot) \) during contrastive learning.
We describe how this attack creates adversarial examples for contrastive loss during self-supervised contrastive learning (see fig. 1 (b)) in this subsection.

Inspired by Meng and Wattenhofer (2020), who leverage geometry of representations to generate natural language adversarial examples for text classification tasks, we also use the geometry of pre-trained representations to create adversarial examples for contrastive loss. The created adversarial examples are used as positive examples of the original examples in our contrastive learning framework, and at the same time are created to maximize the contrastive loss. Hence, we refer to adversarial examples created by the attack as hard positive examples.

The intuition of our attack is that we repeatedly replace words in the original texts such that in each iteration, the replaced word increases the contrastive loss as much as possible. To be specific, consider an example $X_t$, we then have:

1. **Determine Direction for Sentence Vector** Compute the gradients of $\ell_i$ with respect to $z_i$. In this step, we find the direction we should move from $z_i$ to increase the contrastive loss $\ell_i$. We have the gradient vector $v_{z_i} = \nabla z_i \ell_i$.

2. **Choose Original Word to be Replaced** Compute the gradients of $\ell_i$ with respect to input word embeddings of $X_t$. For words tokenized into multiple tokens, we take the average of the gradients of the tokens. In this step, we find the word $w_t$ which has the most influence in computing $\ell_i$. Specifically, assuming we have $L$ words, then we choose $t = \text{arg max}_i \{||g_1||, ||g_2||, \ldots , ||g_L||\}$, where $g_i$ is the gradients of $\ell_i$ with respect to the embeddings of word $w_{t_k}$, $1 \leq k \leq L$.

3. **Generate Candidate Set** Suppose we choose the word $w_t$ in step 2. In this step, we use a pre-trained BERT to choose the most probable candidates $w_t$ to replace it in the original text. We have the candidates set $= \{w_{t_1}, w_{t_2}, \cdots , w_{t_T}\}$. Following Jin et al. (2020), we filter out semantically different words from the candidate set by discarding candidate words of which the cosine similarity of their embeddings between the embeddings of $w_t$ is below a threshold $\epsilon$. We set the threshold $\epsilon = 0.5$ and use counter-fitted word embeddings (Mrkšić et al., 2016) to compute the cosine similarity.

4. **Choose Replacement Word** Replace $w_t$ with words in the candidates set, resulting in text vectors $\{z_{i_1}, z_{i_2}, \cdots , z_{i_{T'}}\}$. We compute delta vector $r_{ij} \leftarrow z_{i_j} - z_i$. The projection of $r_{ij}$ onto $v_{z_i}$ is: $p_{ij} \leftarrow \frac{r_{ij} \cdot v_{z_i}}{||v_{z_i}||}$. We select word $w_{t_m}$, where $m = \text{arg max}_j ||p_{ij}||$. In other words, $w_{t_m}$ results in the largest projection $p_{ij}$ onto $v_{z_i}$.

5. **Repetition** Replace $w_t$ with $w_{t_m}$ in $X_t$, then we have $z_i \leftarrow z_{i_m}$. Repeat step 1-4 for $N$ iterations, where $N$ is a hyperparameter of our method. We expect $\ell_i$ to increase in each iteration.

Figure 2 illustrates an iteration of our attack, in which we have two options to choose from the candidate set. This attack can be easily implemented in a batched fashion, making it possible for us to generate adversarial examples on the fly during training. Furthermore, our efficient implementation makes it possible to conduct contrastive learning with adversarial perturbations as well as adversarial training with adversarial examples generated on the fly. We give a speed comparison of our attack and other attacks in appendix D. We also give pseudocode of the attack in algorithm 1 of appendix A.

4 **Experiments**

4.1 **Datasets and Evaluation Metrics**

We test how our method improves model robustness on four text classification datasets: AG’s News, Yelp, IMDB, and DBpedia (See appendix B for details).

We report the attack success rate and the replacement rate of the attacks as the evaluation metrics. Following Alzantot et al. (2018); Ebrahimi et al. (2018), to prevent the model accuracy on clean examples from confounding the results, we define the success rate of an attack on all correctly classified examples in the test set. Lower success rates indicate higher robustness. The replacement rate refers to the percent of original words replaced in the clean example. Higher replacement rates indicate that the attack needs to replace more words to fool the model, and thus mean that the model is more robust.

4.2 **Attacks for Evaluating Robustness**

We use four word substitution-based adversarial attacks to evaluate the model robustness. **Geometry Attack** We use the same attack described in section 3.3 to generate adversarial examples for sentence classification tasks by replacing contrastive loss with cross-entropy classification loss. We set the maximum number of replaced words to 20.
We use the default implementations from TextAttack (Morris et al., 2020).

All these attacks will give up and terminate once the maximum number of replaced words (sometimes also called perturbation budget) is reached.

### 4.3 Experimental Design

We have the following hypotheses for our method:

**H1:** Self-supervised contrastive learning improves model robustness against adversarial attacks. Moreover, using adversarial perturbations during contrastive learning further improves robustness.

To validate this hypothesis, we set three different pretraining schemes:

- **BTCL:** Pretraining with back-translation as the transformation $t(\cdot)$ for self-supervised contrastive learning.
- **ADCL:** Pretraining with Geometry Attack for contrastive loss (see section 3.3) as transformation $t(\cdot)$ for self-supervised contrastive learning.
- **NP:** Apart from the above two settings, we also add a No Pretraining baseline to understand the general effectiveness of contrastive learning.

**H2:** Combining self-supervised contrastive learning with adversarial training gives higher robustness than conducting adversarial training alone.

We use different finetune strategies to understand how adding adversarial training to our method affects model robustness. We have two settings:

- **FTC:** We finetune the pretrained model on the clean examples of the corresponding downstream dataset.
- **ADV:** We conduct adversarial training by leveraging supervisedly generated adversarial examples.

Note that our adversarial training is different from previous works (Ren et al., 2019; Alzantot et al., 2018), which merely finetune the model on a fixed number of pre-generated adversarial examples. Instead, our adversarial training scheme is similar to Madry et al. (2018), where the model is finetuned on clean examples and adversarial examples generated on the fly during each batch of training.

We use Geometry Attack for adversarial training as the remaining three attacks are not efficient enough to generate adversarial examples on the fly (see appendix D for details).

**H3:** Our contrastive learning method is capable of using out-of-domain data to improve the model robustness.

While in H1 and H2, we use the same dataset for pretraining and finetuning, we want to test how our method can leverage out-of-domain data. Hence, we have two additional experimental settings:

- **In-Domain:** We use the same dataset during contrastive learning and finetuning.
- **Out-of-Domain:** We use different datasets for contrastive learning and finetuning.

**H4:** By optimizing eq. (1), our method pulls the representations of the clean samples and their corresponding hard positive examples closer in the vector space while pushing other different examples further. In this way, the representations of clean examples and their adversarial examples are also closer in the vector space.

We validate this hypothesis by conducting a vector space study. See section 4.4 for details.

Note that to avoid confusing adversarial examples generated during contrastive learning and adversarial examples generated during finetuning, we refer to the former as hard positive examples (see section 3.3).

### 4.4 Results

Table 1 shows the experimental results for validating H1 and H2. For each dataset, when evaluating the model robustness, we use the same perturbation budget across different settings. Note that although the replacement rates vary across different settings of the same dataset, the perturbation budget for the same attack is the same in these settings. By using the same perturbation budget, we ensure that the success rates of the attacks provide us with a fair evaluation of the robustness of the model (Wang et al., 2021c; Ren et al., 2019).

**H1:** To validate H1, we focus on rows with the FTC setting during finetuning. We can observe that models without any contrastive pretraining (NP) are the most vulnerable to adversarial attacks. For example, the success rate of the Geometry Attack for AG’s News dataset is 86.2% for the NP model. In contrast, for BTCL and ADCL, the success rate of the Geometry Attack is at least 5.6% lower than this setting. This shows that self-supervised contrastive learning does improve model robustness.

Additionally, we can also see from table 1 that ADCL improves the model robustness more than BTCL. For example, in the IMDB dataset, the model pretrained with ADCL is 9.1% more robust than the model pretrained with BTCL (93.3% → 84.2%), showing that using adversarial perturbations during contrastive learning further improves model robustness against adversarial at-
Dataset Pretrain Finetune Acc. (%) Success Rate (%) ↓ Replaced (%) ↑

AG

NP FTC 94.2 86.2 87.6 63.6 17.9 18.6 25.7 20.9 7.4

ADV 94.4 20.7 25.1 26.1 10.7 20.5 29.3 22.3 7.7

BTCL FTC 94.4 80.6 84.6 63.1 17.7 18.1 24.6 20.9 7.5

ADCL FTC 94.3 76.5 80.7 55.9 14.1 19.1 26.7 22.6 7.5

ADV 94.4 18.7 23.5 24.7 9.7 20.6 29.3 22.2 7.2

Yelp

NP FTC 97.1 94.6 94.3 97.0 42.1 10.6 10.4 7.1 6.7

ADV 96.2 38.8 52.4 62.7 22.2 12.8 17.3 11.3 8.8

BTCL FTC 97.1 92.3 91.6 94.8 39.2 11.0 10.1 7.7 6.9

ADCL FTC 97.0 88.6 88.2 91.1 37.8 10.4 10.5 7.4 6.9

ADV 96.1 35.6 50.1 61.0 21.0 13.4 17.1 11.2 8.3

IMDB

NP FTC 92.3 98.7 99.0 99.2 54.0 3.5 6.5 4.3 3.0

ADV 92.0 51.4 75.2 79.1 35.1 7.4 12.7 9.3 3.6

BTCL FTC 92.5 93.3 96.6 95.1 52.0 4.5 7.4 4.4 3.3

ADCL FTC 92.4 84.2 87.8 87.8 48.0 3.7 8.7 5.1 2.3

ADV 91.9 48.7 74.4 77.6 31.8 8.1 12.4 9.1 3.5

DBpedia

NP FTC 99.2 79.6 79.3 46.7 14.3 17.8 23.2 16.2 13.3

ADV 99.0 13.9 16.5 17.7 10.9 21.6 28.2 18.9 14.1

BTCL FTC 99.2 77.4 76.8 45.1 13.0 18.9 22.8 18.1 13.1

ADCL FTC 99.1 73.6 74.5 42.6 11.6 18.2 22.9 17.6 12.8

ADV 99.0 12.4 14.8 16.2 10.1 20.1 28.6 18.2 13.8

Table 1: Experimental results for H1 and H2. In-Domain setting is used. We bold the best results, while the second best is in italic.

tacks. Hence, we do not combine BTCL with ADV in later experiments for simplicity.

To further understand how contrastive learning improves the model robustness, we study the transferability of the adversarial examples between models without any contrastive pretraining (NP) and the models pretrained with ADCL. To be specific, the models are first pretrained using either NP or ADCL, and then finetuned on clean examples (FTC). Then, we use a NP model to generate adversarial examples on the test set of each dataset, and then test the corresponding model pretrained with ADCL on these adversarial examples. And vice versa.

Table 3 shows the results. We can see that adversarial examples generated by models pretrained with ADCL have much higher success rates on models without any contrastive pretraining (NP). For example, for the AG’s News dataset, the success rates increase by 32.1%, 35.3%, 33.8%, and 22.1% for Geometry Attack, TextFooler, BAE-R, and PWWS, respectively. This demonstrates that by self-supervised contrastive learning with adversarial perturbations, the models become more robust against attacks.

H2: To validate H2, we compare two settings of NP + ADV and ADCL + ADV. We note that when compared with conducting adversarial training alone (NP + ADV), combining our self-supervised contrastive learning method with adversarial training (ADCL + ADV) constantly results in higher robustness. In other words, the adversarial attacks have lower success rates and higher replacement rate in ADCL + ADV models than in NP + ADV models. For instance, for the IMDB dataset, the ADCL + ADV model is 2.7% more robust than the NP + ADV model, when both models are tested against the Geometry Attack (Success rates of Geometry attack: ADCL + ADV: 48.7%, NP + ADV: 51.4%; Replacement rates: ADCL + ADV: 8.1%, NP + ADV: 7.4%).

Note that when test NP + ADV models and ADCL + ADV models against the other three adversarial attacks, ADCL + ADV models do not show an advantage over NP + ADV models in terms of replacement rates, despite that ADCL + ADV models still constantly make lower success rates against the adversarial attacks. We argue that this is because we use the Geometry Attack for adversarial training during finetuning, and the adversarial examples from the Geometry Attack might not fully match the distribution from the other attacks. Nevertheless, we can still conclude that ADCL + ADV models are more robust than NP + ADV models.

Our experiments also show that during con-
Table 2: Comparison of Out-of-Domain with In-Domain. We use the DBpedia dataset as the out-of-domain dataset for AG’s News and IMDB. Models are finetuned on clean examples after pretraining (FTC). Best results are bolded, while the second best are in italic.

Table 3: Transferability of adversarial examples. The models are pretrained under either NP or ADCL, and then finetuned on clean examples. NP → ADCL: Generate adversarial examples with the model pretrained with NP, then test the model pretrained with ADCL on these adversarial examples. Same applies to ADCL → NP.

H3: For the Out-of-Domain setting, we use the DBpedia dataset as the out-of-domain dataset for the AG’s News and IMDB datasets, mainly because (1) Computational limits: While using larger datasets such as BookCorpus or Wikipedia might be more useful, conducting self-supervised contrastive learning on these datasets exceeds the limits of our computational infrastructure; (2) The DBpedia dataset is several times larger than AG’s News and IMDB. This should give us a glimpse of what it looks like when we scale self-supervised contrastive learning with adversarial perturbations to even larger out-of-domain datasets; (3) The DBpedia dataset (topic classification on Wikipedia) has a different task and domain compared to the AG’s News dataset (news classification from a newspaper) and IMDB dataset (sentiment classification on movie reviews). This discrepancy allows us to understand how out-of-domain datasets could help.

Table 2 shows our results. We can see that models pretrained with ADCL under the Out-of-Domain setting are more robust than models without any pretraining at all (NP). This shows that our method can improve model robustness using out-of-domain data. For instance, for the IMDB dataset, the success rate of TextFooler decreases from 98.7% for FT models to 92.3% for Out-of-Domain ADCL models. This shows that our method can improve the model robustness even if the dataset used for contrastive learning is from a completely different domain. Note that in table 2, after pretraining, we finetune the model on clean examples (FTC).

We also notice that models pretrained with ADCL under the Out-of-Domain setting are not as robust as models pretrained with ADCL under the In-Domain setting. This indicates we might need to use much larger unlabeled raw datasets to obtain more improvements.

H4: To validate this hypothesis, we study the vector representations of $M = 1000$ clean examples of the AG’s News dataset and their corresponding adversarial examples. We obtain the adversarial examples by attacking a NP + FTC model.

Let $v_1, v_2, ..., v_M$ and $v'_1, v'_2, ..., v'_M$ be the vector representations of the clean examples and corresponding adversarial samples, respectively. For each setting, we evaluate three metrics:

- Average distance $d_{pos}$ between each of the pos-

In this paper, we improve the robustness of pre-trained language models against word substitution-based adversarial attacks by using self-supervised contrastive learning with adversarial perturbations. Our method is different from previous works as we can improve model robustness without accessing annotated labels. Furthermore, we also conduct word-level adversarial training on BERT with on-the-fly generated adversarial examples. Our adversarial training is different from previous works in that (1) it is on the word level; (2) we generate adversarial examples on the fly, instead of generating a fixed adversarial set beforehand. Experiments show that our method improves model robustness. We find that combining our method with adversarial training results in better robustness than conducting adversarial training alone. In the future, we plan to scale our method to even larger out-of-domain datasets.

### Table 4: Vector space study

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Distance (d&lt;sub&gt;pos&lt;/sub&gt;,d&lt;sub&gt;neg&lt;/sub&gt;/b)</th>
<th>NP + FTC</th>
<th>ADCL + FTC</th>
<th>NP + ADV</th>
<th>ADCL + ADV</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG</td>
<td>2.4/3.9/1.5</td>
<td>0.8/4.0/2.2</td>
<td>0.7/4.1/3.4</td>
<td>0.7/4.4/3.7</td>
<td></td>
</tr>
<tr>
<td>Yelp</td>
<td>3.5/3.7/0.4</td>
<td>2.9/4.0/1.1</td>
<td>0.7/3.2/2.5</td>
<td>0.6/3.4/2.9</td>
<td></td>
</tr>
<tr>
<td>IMDB</td>
<td>3.0/3.7/0.7</td>
<td>2.3/3.8/1.5</td>
<td>0.6/3.4/2.8</td>
<td>0.6/3.8/3.2</td>
<td></td>
</tr>
<tr>
<td>DBpedia</td>
<td>2.8/4.8/2.0</td>
<td>2.3/5.1/2.8</td>
<td>0.4/4.9/1.5</td>
<td>0.4/5.2/4.8</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Vector space study. For each setting, we evaluate three metrics: (a) Average distance between positive pairs; (b) Average distance between negative pairs; (c) Difference between (a) and (b).

5. Conclusion and Future Work

In this paper, we improve the robustness of pre-trained language models against word substitution-based adversarial attacks by using self-supervised contrastive learning with adversarial perturbations. Our method is different from previous works as we can improve model robustness without accessing annotated labels. Furthermore, we also conduct word-level adversarial training on BERT with on-the-fly generated adversarial examples. Our adversarial training is different from previous works in that (1) it is on the word level; (2) we generate adversarial examples on the fly, instead of generating a fixed adversarial set beforehand. Experiments show that our method improves model robustness. We find that combining our method with adversarial training results in better robustness than conducting adversarial training alone. In the future, we plan to scale our method to even larger out-of-domain datasets.
Ethical Considerations
To the best of our knowledge, the data used in our work does not contain sensitive information. Although our models are evaluated on academic datasets in this paper, they could also be used in sensitive contexts, e.g. healthcare or legal scenarios. It is essential that necessary anonymization and robustness evaluation is undertaken before using our models in these settings.

Acknowledgements
Mrinmaya Sachan acknowledges support from an ETH Zürich Research grant (ETH-19 21-1) and a grant from the Swiss National Science Foundation (project #201009) for this work.

References


Xinshuai Dong, Anh Tuan Luu, Rongrong Ji, and Hong Liu. 2021. Towards robustness against natural language word substitutions. In ICLR.


Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In EMNLP.


Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In CVPR.

Robin Jia and Percy Liang. 2017. Adversarial examples for evaluating reading comprehension systems. In EMNLP.


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.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2018. Towards deep learning models resistant to adversarial attacks. In ICLR.


Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Counter-fitting word vectors to linguistic constraints. In NAACL.


Shuhuai Ren, Yihe Deng, Kun He, and Wanxiang Che. 2019. Generating natural language adversarial examples through probability weighted word saliency. In ACL.

Dong Wang, Ning Ding, Piji Li, and Haitao Zheng. 2021b. CLINE: Contrastive learning with semantic negative examples for natural language understanding. In ACL.

Xiaosen Wang, Yichen Yang, Yihe Deng, and Kun He. 2021c. Adversarial training with fast gradient projection method against synonym substitution based text attacks. In AAAI.


A Geometry Attack for Contrastive Loss

Algorithm 1 is the pseudocode of our Geometry Attack for contrastive loss. Refer to Section 3.3 for more details.

B Datasets

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Labels</th>
<th>Avg Len</th>
<th>Train</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG’s News</td>
<td>4</td>
<td>44</td>
<td>120K</td>
<td>7.6K</td>
</tr>
<tr>
<td>IMDB</td>
<td>2</td>
<td>292</td>
<td>25K</td>
<td>25K</td>
</tr>
<tr>
<td>DBPedia</td>
<td>14</td>
<td>67</td>
<td>560K</td>
<td>70K</td>
</tr>
<tr>
<td>Yelp</td>
<td>2</td>
<td>177</td>
<td>560K</td>
<td>38K</td>
</tr>
</tbody>
</table>

Table 5: Statistics of the datasets.

The statistics of each dataset are shown in Table 5. In our work, the maximum sequence length is set to 128 for AG’s News and DBpedia, 256 for Yelp, and 512 for IMDB. To save time during evaluating the model robustness against attacks, we randomly select a part of the test examples in each dataset for evaluation. Specifically, we select 1,000 samples from IMDB, 2,000 samples from Yelp, and 5,000 samples from DBpedia. We use all 7,600 samples from the AG’s News test set for evaluation.

AG’s News is a topic classification dataset with four types of news articles: World, Sports, Business and Science/Technology.

IMDB (Maas et al., 2011) is a binary sentiment classification dataset on positive and negative movie reviews.

Yelp is a Yelp review dataset for binary sentiment classification. Following Zhang et al. (2015), reviews with star 1 and 2 are considered negative, and reviews with star 3 and 4 are considered positive.

DBpedia (Zhang et al., 2015) is a topic classification dataset with 14 non-overlapping classes. Both content and title fields are used in our work.

C Effect of Queue Size

We conduct additional experiments to study the effect of queue size. We use a queue size of 8192, 16384, 32768, and 65536 under the setting of ADCL+FT for the AG’s News dataset. As is shown in Table 6, a larger queue size generally helps improve the model robustness. However, we also notice that when the queue size is too large, the model robustness starts to decrease. We argue that this is because a too large queue size results in less frequent queue updates, which makes the vectors in the queue stale.

D Speed of Different Attacks

We show in Table 7 the average number of seconds each attack needs for one example. We obtain the average time by attacking 1000 examples and then taking the average. We can observe that the Geometry attack is at least four times faster than TextFooler, and 4 to 10 times faster than PWWS and BAE-R.

E Adversarial Training with Pre-generated Examples

We compare two different methods for adversarial training:

- **Pre-generated** We pre-generate for each example in the training set an adversarial example. We then augment the original training set with the adversarial examples. Finally, the model is finetuned on the augmented dataset.

- **On-the-fly** This setting is the same as ADV in Table 1, where we generate adversarial examples on the fly for each mini-batch during training.

Table 8 shows the results on the AG’s News dataset. We can see that on all four attacks, adversarial training with on-the-fly generated adversarial

---

3http://groups.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
Algorithm 1 Geometry Attack for Contrastive Loss

1: Input: Example $X_i = \{w_1, w_2, \ldots, w_L\}$, encoder $f$ and MLP $g$
2: Output: Adversarial example $X'_i$
3: Initialize $z_i \leftarrow g(f(X_i))$
4: for $\text{iter} = 1$ to $N$ do
5:   calculate $\ell_i$ using Equation 1
6:   $v_{z_i} \leftarrow \nabla_{z_i} \ell_i$
7:   $E \leftarrow \text{BertEmbeddings}(X'_i) = \{e_1, e_2, \ldots, e_L\}$
8:   $G \leftarrow \nabla_E \ell_i = \{g_1, g_2, \ldots, g_L\}$
9:   $t \leftarrow \arg \max_i \|g_t\|$
10: $C \leftarrow \text{BertForMaskedLM}(\{w_1, \ldots, w_{t-1}, \text{[MASK]}, w_{t+1}, \ldots, w_L\})$
11: $C \leftarrow \text{Filter}(C)$ // construct candidates set $C = \{w_1, w_2, \ldots, w_{t_T}\}$; filter using counter-fitted embeddings
12: for each $w_{ij} \in C, 1 \leq j \leq T$ do
13:   $X_{ij} \leftarrow \{w_1, \ldots, w_{t-1}, w_{ij}, w_{t+1}, \ldots, w_L\}$
14:   $z_{ij} \leftarrow g(f(X_{ij}))$
15:   $r_{ij} \leftarrow z_{ij} - z_i$
16:   $p_{ij} \leftarrow \frac{r_{ij}}{||v_{z_i}||}$
17: end for
18: $m \leftarrow \arg \max_j ||p_{ij}||$
19: $X_i \leftarrow X_{im}$
20: $z_i \leftarrow z_{im}$
21: end for
22: $X'_i \leftarrow X_i$
23: return $X_i$

Table 8: Comparison between adversarial training with pre-generated adversarial examples and on-the-fly generated adversarial examples.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Geometry</th>
<th>TextFooler</th>
<th>PWWS</th>
<th>BAE-R</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-generated</td>
<td>55.3/17.1</td>
<td>59.4/22.6</td>
<td>42.0/17.4</td>
<td>16.5/7.3</td>
</tr>
<tr>
<td>On-the-fly</td>
<td>20.7/20.5</td>
<td>25.1/29.3</td>
<td>26.1/22.3</td>
<td>10.7/7.7</td>
</tr>
</tbody>
</table>

examples gives higher robustness than adversarial training with pre-generated adversarial examples.

F Implementation Details

In our paper, we use PyTorch Lightning\(^4\) and HuggingFace Transformers\(^5\) in our implementation. We use BERT as the encoder $f(\cdot)$, and the representation of the [CLS] symbol in the last layer is used for $h$. $g(\cdot)$ is a two-layer MLP, of which the output size $c$ is 128. $g(\cdot)$ uses Tanh as activation function in the output layer. We use FP16 in training step to reduce GPU memory usage, and

use FusedAdam from DeepSpeed\(^6\) as the optimizer. We enable DeepSpeed ZeRO Stage 2 to further speed up training. We conduct all our experiments on 8 RTX TITAN GPUs.

Contrastive learning For Geometry Attack for contrastive loss, to reach a balance between attack success rate and efficiency, the maximum number of iterations $K$ is set to 10 for AG’s News, DBpedia, and Yelp, and 15 for IMDB dataset. We do not perturb words that were already perturbed in previous iterations. For an example $X_i = \{w_1, w_2, \ldots, w_L\}$, at most $\min\{K, 0.2 \cdot L\}$ words can be perturbed. For each word $w_t, 1 \leq t \leq L$, the upper limit of the candidate set size $T$ is set to 25. Due to the various maximum lengths in downstream datasets and GPU memory limits, we use different batch sizes for different datasets. During contrastive learning, the batch size is set to 1024 for AG’s News and DBpedia, 448 for Yelp, and 192 for IMDB.

Fine-tuning During finetuning, we train the model for two epochs for AG’s News and DBpedia, 3 for...

\(^4\)https://www.pytorchlightning.ai/
\(^5\)https://huggingface.co/transformers/
\(^6\)https://www.deepspeed.ai/
Yelp, and 4 for IMDB. The learning rate is set to $2e^{-5}$ and is adjusted using linear scheduling.

**Adversarial training** For adversarial training, the number of training epochs is set to 3 with an additional first epoch of finetuning on clean examples. The adversarial examples are generated *on the fly* in each batch during training. For the Geometry Attack in adversarial training, at most $\min\{K, 0.4 \cdot \text{len}(X_i)\}$ words can be perturbed in an example. The upper limit of the candidate set size is set to 50.

**Back Translation** We use pretrained translation models `opus-mt-en-roa` and `opus-mt-roa-en` from Helsinki-NLP to generate one translation for each example.

### G Hard Positive Examples from Geometry Attack for Contrastive Loss

In Table 9, we show hard positive examples generated by our Geometry Attack for contrastive loss from the AG’s News dataset.

<table>
<thead>
<tr>
<th>Original</th>
<th>Zurich employees plead guilty in probe new york (reuters) - two senior insurance underwriters at zurich american insurance co pleaded guilty on tuesday to misconduct related to bid-rigging in the insurance market.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adversarial</td>
<td>Zurich employees plead guilty in probe new york (reuters) - two senior insurance agents at zurich american insurance co testified guilty on tuesday to violations related to bid-rigging in the insurance market.</td>
</tr>
<tr>
<td>Original</td>
<td>Black watch troops move into position the first units of a black watch battlegroup are due to arrive today in their new positions south of baghdad as tony blair indicated that more british troops may replace them in the american - controlled zone before the end of the year.</td>
</tr>
<tr>
<td>Adversarial</td>
<td>Black watch troops move into place the first units of a black watch operation are due to arrive today in their new positions south of baghdad as tony blair indicated that more british troops may replace them in the american - controlled zone before the end of the year.</td>
</tr>
</tbody>
</table>

Table 9: Hard positive examples generated by Geometry Attack for contrastive loss. Blue words in the original examples are replaced by red words in the adversarial examples.

### H Additional t-SNE plots

We give t-SNE plots of the vector representations of clean examples and adversarial examples from Yelp, IMDB and DBpedia in fig. 4, fig. 5 and fig. 6, respectively.

Figure 4: t-SNE plot of the vector representations of clean examples and adversarial examples from the Yelp dataset.
Figure 5: t-SNE plot of the vector representations of clean examples and adversarial examples from the IMDB dataset.

Figure 6: t-SNE plot of the vector representations of clean examples and adversarial examples from the DBpedia dataset.