KNN-BERT: Fine-Tuning Pre-Trained Models with KNN Classifier

Anonymous ACL submission

Abstract

Pre-trained models are widely used in fine-tuning downstream tasks with linear classifiers optimized by the cross entropy loss, which might face robustness and stability problems. These problems can be improved by learning representations that focus on similarities in the same class and variance in different classes when making predictions. In this paper, we utilize the K-Nearest Neighbors Classifier in pre-trained model fine-tuning. For this KNN classifier, we introduce a supervised momentum contrastive learning framework to learn the clustered representations of the supervised downstream tasks. Extensive experiments on text classification tasks and robustness tests show that by incorporating KNNs with the traditional fine-tuning process, we can obtain significant improvements on the clean accuracy in both rich-source and few-shot settings and can improve the robustness against adversarial attacks. [1]

1 Introduction

Pre-trained language models exemplified by BERT (Devlin et al., 2018) have been widely applied in fine-tuning downstream text classification tasks. It is commonly used to fine-tune the pre-trained model with the cross entropy loss (Rumelhart et al., 1986) that calculates the KL-divergence between the one-hot vectors of labels and the model output predictions and then make predictions using linear classifiers (Radford et al., 2019; Devlin et al., 2018; Liu et al., 2019; Yang et al., 2019; Lan et al., 2019).

Still, such a standard process has its shortcomings: **(A)** the cross entropy loss may lead to poor generalization performance as pointed out by Liu et al. (2016); Cao et al. (2019) and may lack robustness against noisy labels (Zhang and Sabuncu, 2018; Sukhbaatar et al., 2014) and adversarial samples (Goodfellow et al., 2014; Nar et al., 2019). Also, in fine-tuning BERT, the cross entropy loss may be unstable especially with limited data (Dodge et al., 2020; Zhang et al., 2020). **(B)** On the other hand, making predictions through linear classifiers added directly on top of the pre-trained models may face the overfitting problem especially when the training data is limited (Snell et al., 2017; Zhang et al., 2020).

To tackle the above shortcomings, it is intuitive to build better representations in pre-trained language models and make predictions based on classifiers that have better generalization abilities.

Therefore, in this paper, instead of simply using a linear classifier to do the prediction, we utilize the classic K-Nearest Neighbors classifier to make predictions based on the training sample representations. The classic KNN classifier that makes predictions based on counting the top-K similar samples has been neglected for a long time since end-to-end neural networks have achieved great success in the computer vision field (He et al., 2016) as well as the natural language processing field (Vaswani et al., 2017; Devlin et al., 2018). When the representations have been well-learned through the massive calculation of the masked language model task pre-training, it is intuitive to revisit and utilize the K-Nearest Neighbor classifier that makes predictions based on the representation similarity. The KNN classifier makes predictions based on the similarity between representations. Therefore, the decision boundary is tighter within the same class and altering the representation to an incorrect class is more difficult which can improve model robustness. On the other hand, the KNN classifier makes predictions based on the anchors of the multiple training samples which are well-learned representations from the BERT model. Therefore, utilizing KNN classifier can make better use of the semantic representations of the pre-trained models than simply using linear classifiers to draw decision boundaries.

[1] all codes will be available at https://github.com//

For training the representations that are clustered within the same class for the KNN classifier, it is intuitive to use contrastive learning based training strategies. The goal is to construct a tight cluster of the representations within the same class while keeping the clusters of different classes at distance. With the label information from the downstream task dataset, we introduce a class-wise supervised contrastive learning framework to cluster the representations. Based on traditional constrastive learning framework, we use the class-wise positives drawn from the same class of the given example instead of using limited augmentation-based methods to construct positives. These class-wise positives are relatively more abundant and useful comparing with augmentation-based positives and they can also be diversified in semantics.

To make use of the class-wise positives, we introduce a sampling strategy that collect both the most similar positives and least similar positives to learn representations that can be tightly clustered within the same class while keeping distance between different classes based on the momentum contrast learning framework (MoCo) (He et al., 2020). The momentum contrast framework introduces a momentum-based optimization process to update the representations of the negatives from a queue which makes it possible to make use of massive negatives. In our usage of contrastive learning, incorporating the queue-based momentum contrast allows the usage of multiple positives and negatives which is intuitive in using class-wise positives.

For the representation learning of the positives, we are hoping that (1) the cluster of samples is tight within the same class; (2) the clusters are distant between classes. Therefore, when updating the representations of the class-wise positives, we introduce a sampling strategy that consider the most similar and least similar positives to get better cluster representations. Updating the most similar positives can draw near the representations within the same class especially in the pre-trained representations where contrastive learning on randomly selected pre-trained representations may sabotage the pre-trained information.

We construct extensive experiments to test the generalization and robustness ability of our contrastive-learned representations for the KNN classifier. We test rich-resource and low-resource text classification tasks on the GLUE benchmark; we then test the robustness of the KNN classifier by using the classifier to defend against strong substitution-based adversarial attack methods. Experiment results indicate that the KNN classifier can (1) improve the performances by a considerable margin in text classification tasks; (2) improve the defense ability against adversarial attacks significantly.

To summarize our contributions:

- We introduce the idea of utilizing traditional KNN classifiers in downstream task fine-tuning of pre-trained models and use contrastive-learning to learn the representations for the KNN classifier.
- We make use of class-wise positives and negatives and introduce a sampling strategy that consider most and least similar positives for the contrastive learning process especially in pre-trained models.
- We incorporate a momentum contrast based framework to allow multiple positives and negatives in the contrastive learning process.
- Extensive experiments show the effectiveness of the proposed contrastive learning framework for the KNN classifier in both model generalization ability and model robustness.

2 Related Work

2.1 Utilizing the KNN Classifier in PTMs

The K-nearest neighbor classifier is a traditional algorithm that makes predictions based on representation similarities. While pre-trained models (PTMs) (Devlin et al., 2018; Radford et al., 2018; Lan et al., 2019; Liu et al., 2019) have been widely applied, the idea of using nearest neighbors in pre-trained models is also explored. Khandelwal et al. (2019) use nearest neighbors to augment the language model predictions by using neighbors of the predictions as targets for language model learning. Kassner and Schütze (2020) apply nearest neighbors as additional predictions to boost the question answering task. These methods use nearest neighbors to find augment samples based on the pre-trained language models rather than using the KNN classifier as the decision maker.

On the other hand, making predictions based on the nearest neighbors can be used in improving model robustness. Papernot and McDaniel (2018) explore the possibility of using nearest neighbors

to make decisions instead of using linear classifiers in the computer vision field, showing that classification results based on near neighbors are more resilient to adversarial attacks (Goodfellow et al., 2014; Carlini and Wagner, 2016).

2.2 Contrastive Learning

Contrastive learning (Hadsell et al., 2006; Chen et al., 2020) is a similarity-based training strategy that has been widely used (Hjelm et al., 2018; Sermanet et al., 2018; Tschannen et al., 2019). The formulation of the contrastive loss is mainly based on the noise contrastive estimation loss (Gutmann and Hyvärinen, 2010; Mnih and Kavukcuoglu, 2013) or the N-pair losses (Sohn, 2016), which is also closely related to the metric distance learning and triplet losses (Schroff et al., 2015; Weinberger and Saul, 2009).

While recent contrastive learning frameworks are mainly used in self-supervised tasks (He et al., 2020; Chen et al., 2020), the contrastive losses can also be used in a supervised scenario with minor modification to the loss function (Khosla et al., 2020; Gunel et al., 2020). These supervised contrastive learning losses are added as an additional task in the normal training, the inference process is still based on linear classifiers.

3 KNN-BERT

We propose KNN-BERT that utilizes the KNN classifier when using pre-trained models exemplified by BERT as the representation encoder. We illustrate the KNN-BERT by describing (1) the KNN classifier usage; (2) the training process of the representations for the KNN classifier.

3.1 KNN Classifier

We combine the normal linear classifier with the KNN classifier and use the weight-averaged logits as the final prediction logits. Suppose the encoded representation is q with label Y_q and the linear classifier is $F(\cdot)$; we use $\mathcal{K} = \{k_0, \cdots, k_i \cdots, k_K\}$ with label Y_{k_i} to denote the K nearest neighbors measured by the cosine similarity. The KNN logits is a voted result denoted as KNN(q).

With weight ratio ϕ, the final prediction score s is calculated by:

$$s = (1-\phi)\text{Softmax}(F(q)) + \phi\text{KNN}(q) \quad (1)$$

Here, the linear classifier $F(\cdot)$ is learned by traditional cross entropy loss. For the kNN classifier learning, we illustrate our proposed contrastive learning framework in the following section.

3.2 Contrastive Learning for KNN

In order to train representations for the KNN classifier in fine-tuning pre-trained models, we introduce a supervised contrastive learning framework that makes use of label information to construct positive and negative samples.

Derived from the InfoNCE loss (Gutmann and Hyvärinen, 2010), we consider a supervised contrastive loss function $\mathcal{L}_{\textbf{sc}}$:

$$\mathcal{L}_{\textbf{sc}} = \frac{1}{M}\sum_{k_j \in \mathcal{K}_+} \Big(-\log \frac{\exp(q \cdot k_j/\tau)}{\sum\limits_{k_l \in \mathcal{K}_- \cup \{k_j\}} \exp(q \cdot k_l/\tau)} \Big) \quad (2)$$

Here, $\mathcal{K}_+$ is the set of M samples that share the same label with the given query sample q and $\mathcal{K}_-$ is the set of samples from different classes. Such a loss function could narrow down the gap between the query and the positive samples and push away the query and the negatives. Considering that the positive samples could be diversified since they are from the same class but the representations possess various semantic information encoded by pre-trained models, it is important to determine which positives should be used in calculating the similarities with the given query, otherwise, the learned representations may not be tightly clustered.

Therefore, we aim to learn the clusters by (1) tightening the cluster of samples of the same class; (2) pushing away samples from different classes.

As seen in Figure 1, we calculate similarities between the most similar positives and the query to build a tighter cluster by narrowing the gap of these most similar samples with the query. On the other hand, we select the least similar positives and draw them towards the query sample. Optimizing the gap between the least similar positives and the query sample is similar to using hard-negatives for better clustering, so we can also name these positives as hard-positives.

Therefore, we select M_m most similar positives k_m and M_l least similar positives k_l from positives set k_+ and only update these selected positive sample representations. Calculating all positives might sabotage the semantic information which

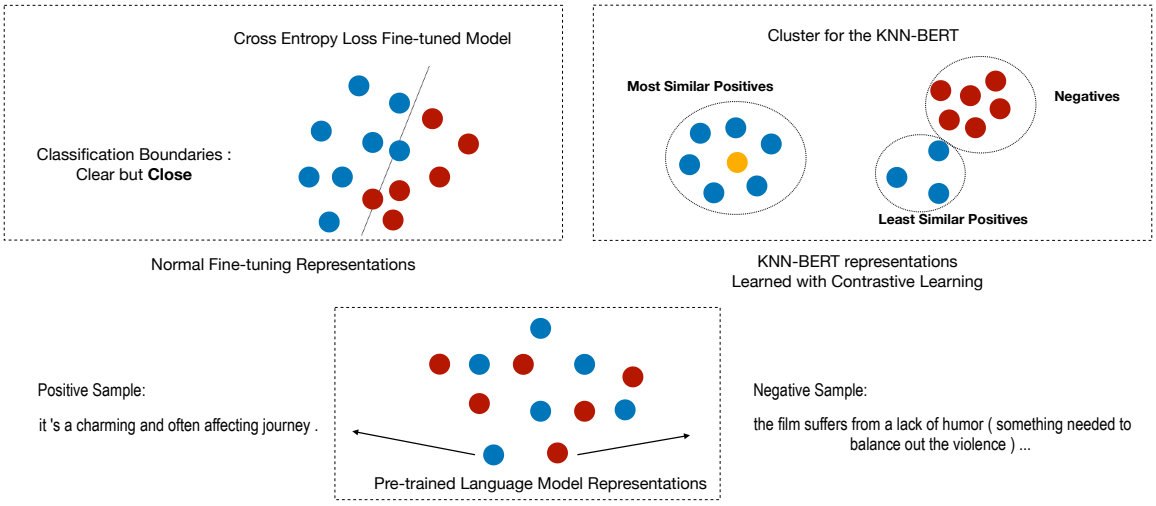


Figure 1: An illustration of using contrastive learning methods to build clusters for downstream classification tasks. We use dots of different colors to denote different classes. The most similar positives are the major cluster while the least similar positives are the data points that are closer to the negative classes. The KNN classifier use an anchor-based prediction strategy unlike previous linear classifiers, we use contrastive loss to make the clusters tighter and draw the least similar samples towards the major cluster.

may not be related to the classification representations and hurt the classification results since the class-wise positives can be significantly different with the query. The proportion of the selected most and least similar positives would play a vital role in the cluster learning process, which will be discussed in the later section.

Since we only update these selected positives, we can re-write the contrastive loss function to:

$$\mathcal{L}'_{\mathbf{sc}} = \frac{1}{M_m + M_l} \sum_{k_j \in \{k_m, k_l\}} \Big(-\log \frac{\exp(q \cdot k_j/\tau)}{\sum_{k_i \in \{k_-, k_j\}} \exp(q \cdot k_i/\tau)} \Big) \quad (3)$$

3.3 Connections with Traditional Contrastive Learning

Contrastive learning (Hadsell et al., 2006; Chen et al., 2020) is to train a representation (denoted as q) using positive keys (denoted as k_+) and negative keys (denoted as k_-). When the similarity is measured by the dot-product, a **c**ontrastive loss with one positive key and multiple negative keys (N negatives) is considered as:

$$\mathcal{L}_{\mathbf{c}} = -\log \frac{\exp(q \cdot k_+/\tau)}{\sum_{k_i \in \{k_-, k_+\}} \exp(q \cdot k_i/\tau)} \quad (4)$$

Here τ is the temperature hyper-parameter, and $\{k_-, k_+\}$ is the union of over one positive k_+ and N negatives $k_i \in k_-$, which is $N+1$ samples in total. This form of loss is closely related to the widely used information noise contrastive estimation (Oord et al., 2018), and widely used in self-supervised contrastive learning tasks (Sohn, 2016; Oord et al., 2018; Henaff, 2020; Baevski et al., 2020) where the positive can be constructed using data augmentation methods (Chen et al., 2020).

Compared with the traditional contrastive learning method, tasks such as text classification are supervised tasks where supervised contrastive learning is explored in the language understanding tasks (Gunel et al., 2020). The major difference is that supervised contrastive learning allows multiple usage of positives since the positives can be drawn from the same class with the query sample.

Based on the self-supervised and supervised contrastive learning frameworks, we build our proposed contrastive learning framework for the KNN-BERT model.

3.4 Optimizing with Momentum Contrast

As illustrated in Eq. 3, we are using multiple positives and a large number of negatives in calculating the contrastive loss, therefore, we utilize a momentum contrast framework to update multiple positives and negatives in a batch iteration to avoid GPU memory explosion.

In the contrastive learning training process, incorporating massive negatives can help better sample the underlying continuous high-dimensional

space of the encoded representations. Therefore, the momentum contrast framework (MoCo) is introduced (He et al., 2020) to consider very large amount of negatives using a queue-based update strategy.

In the momentum contrast framework, there are two separate encoders: query encoder and key encoder. The query encoder is updated by using the gradient descent of the query samples. The optimization of the key encoder is solved by a momentum process using the parameters from the query encoder as illustrated below:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{5}$$

Here θ_q and θ_k are the parameters of the encoders while only the query encoder θ_q is updated by gradients through back-propagation.

The negative representations are first pushed into the recurrent queue and only the samples in the end of the queue are updated by encoding with the key encoder after the key encoder is updated by the momentum process based on the query encoder. Through the momentum update process, the constrastive learning process can consider a great amount of positives and negatives since the process does not need to calculate the gradients on all positives and negatives.

Different from the traditional Moco framework where the positive sample is updated based on gradients, we have large amounts of both positives and negatives in the supervised contrastive learning setting. We simply push all these samples in to the queue and construct the positives and negatives based on the label of the query sample.

3.5 Combined Training

In the pre-trained model fine-tuning exemplified by text classification tasks, the representations are the `[CLS]` tokens used for text classification tasks. We use the l_2 normalization over these representations since normalization methods are widely used in contrastive learning methods and have been proved useful through empirical results. Therefore, the queries and their corresponding positives and negatives are the representations of the BERT encoder output `[CLS]` tokens.

We add the contrastive loss along with the original cross entropy loss $\mathcal{L}_{ce}$ in the fine-tuning process to make use of the label information in a more direct way.

Therefore, the final training loss is:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{ce} + \lambda\mathcal{L}^{'}_{\mathbf{sc}} \tag{6}$$

4 Experiments

4.1 Datasets

We use several text classification datasets to evaluate the effectiveness and robustness of our proposed KNN-based classifier.

We use several datasets in the GLUE benchmark (Wang et al., 2018): RTE (Dagan et al., 2005); CoLA (Warstadt et al., 2018), MRPC (Dolan and Brockett, 2005); QNLI (Rajpurkar et al., 2016); MNLI (Williams et al., 2018) and SST-2 (Socher et al., 2013). In testing the text classification models, we have two experiment settings: we train the model with the full training dataset and test on the validation set; we also set a few-shot setting with only a small portion of the training set. We sample a test set and a development set from the given development set following (Gunel et al., 2020). We also use the IMDB movie review dataset (Maas et al., 2011) and the AG's News news-genre classification dataset (Zhang et al., 2015) to test the generalization ability as well as the model robustness. We use the well-known substitution-based adversarial attack methods, Textfooler (Jin et al., 2019) and Bert-Attack (Li et al., 2020) to attack our KNN classifier.

4.2 Implementations

We run the experiments based on BERT-BASE (Devlin et al., 2018) and RoBERTa LARGE model (Liu et al., 2019) using Huggingface Transformers. We use the standard fine-tuning hyper parameters with learning rate set to 2e-5 and batch-size set to 32 and optimize using the Adam optimizer. The parameters are not particularly tuned, we only use the parameters provided by the Transformers toolkit [2]. In the experiments that concern the contrastive learning process, we search for proper hyperparameters. The size of the queue is 32000, while in the tasks with a small size of training set we put the entire dataset into the queue.

For the training hyper-parameter configuration, we set the momentum update parameter $m = 0.999$ with the temperature $\tau = 0.07$ following He et al. (2020). We set the positives number M_m and M_l considering the training set size of different tasks

[2]https://github.com/huggingface/transformers

Methods	RTE	MRPC	CoLA	QNLI	MNLI	SST-2	IMDB	AG's News
BERT-BASE (Devlin et al., 2018)	65.34	88.99	58.85	91.37	84.51	92.72	93.50	94.50
SCL (Gunel et al., 2020)	67.87	87.97	54.94	90.99	84.35	92.43	92.65	93.40
Memory-Bank (Wu et al., 2018)	66.43	88.67	61.34	91.67	84.35	93.00	93.16	94.83
MoCo (He et al., 2020)	71.11	88.90	**61.94**	91.26	84.50	92.70	93.50	94.65
KNN-BERT (Memory-Bank)	71.12	89.71	61.52	91.73	**84.82**	93.11	93.18	**94.86**
KNN-BERT (MoCo)	**75.70**	**91.22**	**61.94**	**91.74**	84.69	**93.11**	**93.62**	94.75

Table 1: Main Results on full-data text classification tasks and sentence pair classification tasks.

selecting from a certain set {10, 50, 100, 200, 400}. The ratio ϕ between the contrastive and the linear classifier during inference time is 0.25 typically and λ is 0.1 typically. And the number of neighbors K is selected from a certain set {100, 200, 500, 1000}. We use the development set result to choose the optimal hyper-parameter.

For the robustness experiments, we use Textfooler (Jin et al., 2019) and BERT-Attack (Li et al., 2020) as the adversarial attack methods to attack the downstream task classifiers. We use the TextAttack Toolkit (Morris et al., 2020) to implement the attack methods and test the performances against adversarial attacks using our KNN-based classifier.

4.3 Baselines

We compare our KNN-based classifier with several contrastive learning methods. We train these methods using the same parameters with our KNN-based approach for a fair comparison.

To the best of our knowledge, we are the first to deploy the KNN classifier in text classification fine-tuning in pre-trained models therefore the most important baseline is the same model trained with contrastive losses without using the KNN classifier.

SCL: We first construct a supervised contrastive loss involved training baseline which is similar to Gunel et al. (2020). The supervised contrastive loss is similar to Eq. 2 where the positive and negatives are randomly selected in the minibatch. The SCL method uses randomly selected in-batch positives and negatives.

Memory-Bank: Wu et al. (2018) introduces a contrastive framework to make use of massive negative samples based on Memory Banks.

MoCo: We then construct a more delicate baseline that incorporates the contrastive loss using the MoCo (He et al., 2020) framework. That is, the negatives are drawn from the queue which is significantly larger than the batch size.

Methods	RTE	MRPC	QNLI	SST-2
BERT	66.4	88.9	90.5	**93.5**
KNN-BERT	**70.2**	**89.1**	**90.8**	**93.5**

Table 2: Main Results on the test server of GLUE benchmark using KNN-BERT (MoCo) checkpoints based on the best development set results.

4.4 Main Results

We propose a KNN-based classifier trained with MoCo-based contrastive learning framework and we test on the widely acknowledged GLUE benchmark as shown in Tab.1. We observe that when using the KNN classifier, the model performances have an average improvement of 1.39 points compared with the BERT baseline. We also test the KNN classifier on the test server of the GLUE benchmark [3] as shown in Tab.2.

We compare our KNN-BERT method with several contrastive learning baselines. As seen, when we use the contrastive learning loss in the training stage with negatives sampled from the minibatch, the performances improve by a small margin compared with the BERT baselines. Further, when we only use the MoCo training loss as an additional loss in the training process, the model performances are still behind the KNN-BERT method. Results of two variants of our method KNN-BERT (MoCo) and KNN-BERT (Memory-Bank) indicate that updating the representations using MoCo is also important for the KNN classifier. Compared with the Memory bank framework and the MoCo framework, we observe that our contrastive learning method for the KNN classifier can achieve a considerable improvement indicating that the modification we proposed for the supervised language understanding task is effective.

[3] https://gluebenchmark.com/

Num.	Methods	SST-2	QNLI	IMDB
100	BERT	78.90(3.31)	65.76(29.87)	73.38(13.39)
	SCL	75.96(9.37)	65.27(26.20)	73.65(6.52)
	MoCo	79.63(5.85)	68.14(0.24)	74.82(13.98)
	KNN-BERT	**81.36**(5.85)	**70.52**(0.45)	**79.56**(1.95)
	RoBERTa	92.16(0.83)	70.40(47.72)	92.66(0.20)
	SCL	90.00(1.88)	71.39(53.64)	92.21(0.48)
	MoCo	91.14(1.01)	72.90(57.81)	92.71(0.57)
	KNN-RoBERTa	**93.20**(0.10)	**76.00**(37.26)	**93.68**(0.41)
1000	BERT	88.30(0.63)	76.26(1.25)	88.82(0.08)
	SCL	89.40(0.06)	77.16(0.52)	88.53(0.07)
	MoCo	88.58(0.85)	77.34(0.63)	89.11(0.12)
	KNN-BERT	**89.96**(0.37)	**77.68**(0.89)	**91.68**(0.53)
	RoBERTa	93.26(0.28)	85.32(1.57)	94.47(0.03)
	SCL	93.49(0.23)	87.15(3.71)	94.24(0.02)
	MoCo	93.90(0.28)	85.98(0.17)	94.01(0.18)
	KNN-RoBERTa	**94.04**(0.13)	**87.32**(0.09)	**96.08**(0.55)

Table 3: Few-Shot Results on the constructed test set. We run 5 times using different seeds and use the averaged performance with variance given in the parentheses.

Methods	Origin	Textfooler	BERT-Attack
IMDB			
BERT	93.7	24.7	17.3
KNN($\phi = 0.5$)	94.3	44.7	25.7
KNN($\phi = 1.0$)	94.3	**52.7**	**30.0**
AG's News			
BERT	88.0	22.0	21.7
KNN($\phi = 0.5$)	90.6	22.7	37.3
KNN($\phi = 1.0$)	90.3	**47.7**	**42.7**

Table 4: Robustness experiments tested on strong adversarial attack methods (KNN is the the KNN-BERT method). The metric is the after-attack accuracy.

4.5 Few-Shot GLUE Results

As mentioned, we observe that the contrastive loss based KNN classifier can achieve better results in low-source tasks. Therefore, we construct a few-shot experiment using limited data for the downstream tasks.

As seen in Tab.3, both BERT and RoBERTa models can be improved by the KNN classifier when the training set has only 100 or 1000 training samples in the SST-2, QNLI and IMDB dataset. The few-shot setting constrains the performances of language model fine-tuning compared with the rich-source fine-tuning, while the KNN classifier can gain a more significant improvement in the few-shot settings compared with the rich-source fine-tuning. Plus, we can observe that the KNN classifier have a relatively small variance, indicating that the performance is more stable.

We assume that when the training data is limited, the linear classifier would face a serious overfitting problem. The similarity-based KNN classifier, on the other hand, considers more connections between the samples in the same class, which contributes to the improvements over the few-shot experiments. Compared with the baseline supervised contrastive learning methods, using the KNN classifier to make predictions can achieve higher performances.

4.6 Model Robustness against Adversarial Attacks

The robustness of neural networks has raised more and more concerns while these powerful models are widely applied. To explore the robustness of our KNN classifier against strong adversarial attack methods, we construct a robustness experiment to put our KNN classifier as the target model for the strong attacking methods. We use two different settings: (1) the predictions are made by both the linear classifier and the KNN classifier ($\phi = 0.5$) (2) predictions are only made by the KNN classifier ($\phi = 1$).

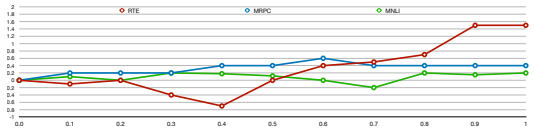
As seen in Tab.4, utilizing the KNN classifier is helpful in obtaining a higher accuracy when attacked by strong adversarial attack methods. Since the attacking process is an iterative searching process, it becomes harder to find proper substitutions as adversarial examples when the distance between classes is larger. The comparison between the KNN-only classifier($\phi = 1.0$) and the KNN & Linear combined classifier($\phi = 0.5$) indicates that the linear classifier is not robust even when the model has been trained with contrastive losses, which reveals a strong advantage of utilizing the KNN classifier in the pre-trained model fine-tuning. In the KNN classifier, the robustness improvements are obtained by the closer distributions over the clean examples from the training set serving as anchors, which could provide strong defense results against adversarial examples.

4.7 Ablations

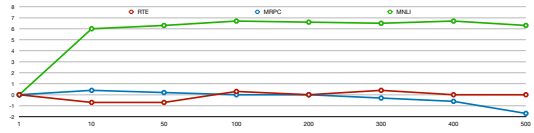
We conduct experiments to explore the effectiveness of each components in the proposed method. We explore a certain hyper-parameter setting while fixing the rest based on best results in the development set.

4.7.1 Effectiveness of Using KNN classifier

To explore the effectiveness of the KNN classifier, We plot the ratio curve of ϕ between prediction scores using linear classifier and KNN classifier.



(a) selection of ϕ



(b) selection of K

Figure 2: KNN parameter ablation. We use the relative gap (based on $\phi = 0$ and $K = 1$ to clearly observe the difference.

Methods	RTE	MRPC	Methods	RTE	MRPC
M./L. SP.	$M_m = 10$			$M_m = 50$	
$M_l = 1$	67.8	91.5	$M_l = 1$	74.7	91.1
$M_l = 10$	71.5	91.3	$M_l = 50$	73.8	91.3
$M_l = 20$	71.5	91.3	$M_l = 100$	75.7	91.2
	$M_m = 100$			$M_m = 200$	
$M_l = 1$	73.3	90.4	$M_l = 1$	67.1	90.0
$M_l = 100$	73.3	90.3	$M_l = 200$	66.4	91.5
$M_l = 200$	67.8	90.2	$M_l = 400$	67.1	91.4
Rand. Positive	RTE	MRPC		RTE	MRPC
$M = 10$	68.9	90.1	$M = 100$	71.1	90.8
$M = 50$	70.4	89.3	ALL	52.0	70.6

Table 5: Importance of Most/Least Similar Positives (M.L. SP.) compared with using random selected M positives (Rand. Positive). and all positives (ALL).

As seen in Fig.2(a), the performance of the KNN classifier is slightly better than the linear classifier, which indicates that KNN based classifier is effective in improving downstream tasks. Combining both classifiers can achieve a higher performance when $\phi = 0.9$ in the MRPC task, $\phi = 0.6$ in the MRPC task and $\phi = 0.8$ in the MNLI task, indicating that a combination of the two classifiers can also be beneficial. Based on the ablations, we can conclude that the KNN classfier is effective and robust.

4.7.2 Importance of K neighbors in KNN

In our proposed KNN-BERT, the selection of number of nearest neighbors is a major parameter. Therefore, we plot the curve of using different K. As seen in Fig.2(b), the selection of K is less vital when the K is large enough. A very small K can still have a considerable but not supreme result.

We can conclude that a large K is not necessary which can save the computation cost.

4.7.3 Importance of Introducing Most/Least Similar Positives

The major part of our contrastive learning framework is the selection of most and least similar positives and negatives since we aim to make use of the feature of both most and least positives to construct tighter and more distinguishable clusters. We construct an ablation study by searching for optimal M in an intuitive selected range {1, 20, 50, 100,200}. As seen in Tab.5, compared with randomly selected positives, using specific selected positives achieves considerable improvements. Further, the selection of different M_m and M_l plays an important role. We assume that the selection of M_m and M_l depends on the training set size of different tasks. Further, compared with using randomly selected positives or single positive, we can observe that using multiple positives helps achieve a considerable improvement than using a single positive sample, especially on tasks that have more diversified patterns like the RTE task. Using all positives could cause a significant performance drop indicating that the pre-trained representations are sabotaged. The values of M_m and M_l are larger than the batch size which indicates that introducing the MoCo framework is fair and effective. Further, we can see that different values of hard-positives M_l also matters, which indicates that introducing a proper number of hard-positives is helpful in learning better representations.

To conclude, introducing the sampling strategy of most/least similar positives is effective and the selection of M depends on the size of the training set therefore requires only a few searching efforts.

5 Conclusion

In this paper, we introduce a KNN-based classifier to improve the performance of pre-trained model fine-tuning. We utilize the traditional KNN classifier in pre-trained model fine-tuning and train clustered representations based on a supervised contrastive learning framework. We introduce a most and least similar positive sample selection strategy based on momentum contrast framework for multiple class-wise positives and negatives in contrastive learning. The KNN classifier can achieve higher performances on the downstream tasks while it can improve the model robustness against strong adversarial attack methods. We hope that the idea of utilizing the KNN classifier could provide hints for future fine-tuning of pre-trained models.

References

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. 2020. wav2vec 2.0: A framework for self-supervised learning of speech representations. *arXiv preprint arXiv:2006.11477*.

Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. 2019. Learning imbalanced datasets with label-distribution-aware margin loss. *arXiv preprint arXiv:1906.07413*.

Nicholas Carlini and David A. Wagner. 2016. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the First International Conference on Machine Learning Challenges: Evaluating Predictive Uncertainty Visual Object Classification, and Recognizing Textual Entailment*, MLCW'05, page 177–190, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah Smith. 2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *arXiv preprint arXiv:2002.06305*.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. Supervised contrastive learning for pre-trained language model fine-tuning. *arXiv preprint arXiv:2011.01403*.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304. JMLR Workshop and Conference Proceedings.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, volume 2, pages 1735–1742. IEEE.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9729–9738.

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*, pages 770–778.

Olivier Henaff. 2020. Data-efficient image recognition with contrastive predictive coding. In *International Conference on Machine Learning*, pages 4182–4192. PMLR.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. 2018. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*.

Di Jin, Zhijing Jin, Joey Tianyi Zhou, and Peter Szolovits. 2019. Is BERT really robust? natural language attack on text classification and entailment. *CoRR*, abs/1907.11932.

Nora Kassner and Hinrich Schütze. 2020. Bert-knn: Adding a knn search component to pretrained language models for better qa. *arXiv preprint arXiv:2005.00766*.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2019. Generalization through memorization: Nearest neighbor language models. *arXiv preprint arXiv:1911.00172*.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. *arXiv preprint arXiv:2004.11362*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Linyang Li, Ruotian Ma, Qipeng Guo, Xiangyang Xue, and Xipeng Qiu. 2020. Bert-attack: Adversarial attack against bert using bert. *arXiv preprint arXiv:2004.09984*.

Weiyang Liu, Yandong Wen, Zhiding Yu, and Meng Yang. 2016. Large-margin softmax loss for convolutional neural networks. In *ICML*, volume 2, page 7.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019.

Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150.

Andriy Mnih and Koray Kavukcuoglu. 2013. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26:2265–2273.

John Morris, Eli Lifland, Jin Yong Yoo, Jake Grigsby, Di Jin, and Yanjun Qi. 2020. Textattack: A framework for adversarial attacks, data augmentation, and adversarial training in nlp. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 119–126.

Kamil Nar, Orhan Ocal, S Shankar Sastry, and Kannan Ramchandran. 2019. Cross-entropy loss and low-rank features have responsibility for adversarial examples. *arXiv preprint arXiv:1901.08360*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Nicolas Papernot and Patrick McDaniel. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. *arXiv preprint arXiv:1803.04765*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *openai*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature*, 323(6088):533–536.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823.

Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. 2018. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE.

Jake Snell, Kevin Swersky, and Richard S Zemel. 2017. Prototypical networks for few-shot learning. *arXiv preprint arXiv:1703.05175*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 1857–1865.

Sainbayar Sukhbaatar, Joan Bruna, Manohar Paluri, Lubomir Bourdev, and Rob Fergus. 2014. Training convolutional networks with noisy labels. *arXiv preprint arXiv:1406.2080*.

Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. 2019. On mutual information maximization for representation learning. *arXiv preprint arXiv:1907.13625*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. ArXiv preprint 1804.07461.

Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2018. Neural network acceptability judgments. *arXiv preprint arXiv:1805.12471*.

Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of machine learning research*, 10(2).

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1112–1122.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3733–3742.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.

Tianyi Zhang, Felix Wu, Arzoo Katiyar, Kilian Q Weinberger, and Yoav Artzi. 2020. Revisiting few-sample bert fine-tuning. *arXiv preprint arXiv:2006.05987*.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Zhilu Zhang and Mert R Sabuncu. 2018. Generalized cross entropy loss for training deep neural networks with noisy labels. *arXiv preprint arXiv:1805.07836*.

Chen Zhu, Yu Cheng, Zhe Gan, Siqi Sun, Thomas Goldstein, and Jingjing Liu. 2019. Freelb: Enhanced adversarial training for language understanding. *arXiv preprint arXiv:1909.11764*.

Appendix

Visualizations

To explore whether the representations are actually clustered in the same class by the contrastive loss, we use the tsne visualization tool to plot the `[CLS]` representations on the 100 sample fews-shot IMDB dataset using the BERT models. We plot the representations of the original BERT fine-tuned model and our contrastive learned model using data points sampled from the development set.

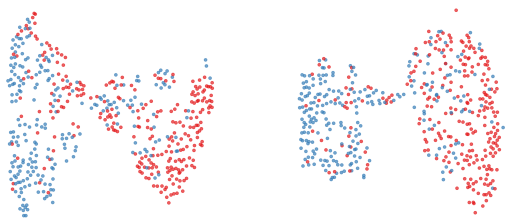
As seen in Fig.3(a) and (b), the distance between classes is larger in the contrastive learned representations than the baseline method. Though some of the samples are still mixed in the wrong cluster, the decision boundary is clearer and far from the other class when the representations are trained with contrastive losses.

MoCo Implementation

In the MoCo implementation, the queue contains all samples with different labels. We collect the samples from the MoCo queue and only select the samples with the corresponding labels as positives and negatives. This process could cost some extra time in our implementation since we iterate the queue to obtain the samples of the corresponding class.

Time Complexity

We test the training time of our proposed method to make a comparison with the baseline methods.



(a) Few-shot Linear Classifer (b) Few-shot KNN-BERT

Figure 3: TSNE Visualization, where blue dots stand for the negative samples and red dots stand for the positive samples.

Dataset	RTE	CoLA
BERT	80 s/epoch	96 s/epoch
KNN-BERT	162 s/epoch	173 s/epoch

Table 6: Training time Comparison

As seen in Table 6, our proposed KNN-BERT only has a 2-times longer training time. Compared with using larger number of parameters such as BERT-large during pre-training or adversarial training (Zhu et al., 2019) (2-3 times), the improvement is considerable while the cost is acceptable.