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.

1all 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 contrastive 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 \( K = \{k_0, \ldots, k_i, \ldots, k_K\} \) 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 \( \phi \), 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 \( L_{sc} \):

\[
L_{sc} = \frac{1}{M} \sum_{k_j \in K_+} \left( - \log \frac{\exp(q \cdot k_j / \tau)}{\sum_{k_l \in K_- \cup\{k_j\}} \exp(q \cdot k_l / \tau)} \right)
\]

(2)

Here, \( K_+ \) is the set of \( M \) samples that share the same label with the given query sample \( q \) and \( K_- \) 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
We use dots of different colors to denote different classes. The most similar positives are the major cluster while with one positive key and multiple negative keys we can re-write the contrastive loss function to:

\[
\mathcal{L}_{sc} = \frac{1}{M + M_l} \sum_{k_j \in \{k^+, k^+_l\}} \log \frac{\exp(q \cdot k_j / \tau)}{\sum_{k_i \in \{k^-, k^+_l\}} \exp(q \cdot k_i / \tau)}
\] (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 contrastive loss with one positive key and multiple negative keys (\( N \) 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 (Günel 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:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q \quad (5)$$

Here $\theta_q$ and $\theta_k$ are the parameters of the encoders while only the query encoder $\theta_q$ 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 contrastive 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 $L_{ce}$ in the fine-tuning process to make use of the label information in a more direct way.

Therefore, the final training loss is:

$$L = (1 - \lambda)L_{ce} + \lambda L'_{sc} \quad (6)$$

## 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 (Gan 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., 2018) 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.

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 $M_\text{pos}$ and $M_\text{neg}$ considering the training set size of different tasks.

---

2https://github.com/huggingface/transformers
Table 1: Main Results on full-data text classification tasks and sentence pair classification tasks.

<table>
<thead>
<tr>
<th>Methods</th>
<th>RTE</th>
<th>MRPC</th>
<th>CoLA</th>
<th>QNLI</th>
<th>MNLI</th>
<th>SST-2</th>
<th>IMDB</th>
<th>AG’s News</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT-BASE (Devlin et al., 2018)</td>
<td>66.4</td>
<td>88.9</td>
<td>58.8</td>
<td>91.2</td>
<td>84.5</td>
<td>92.7</td>
<td>93.5</td>
<td>94.5</td>
</tr>
<tr>
<td>SCL (Gunel et al., 2020)</td>
<td>67.7</td>
<td>87.9</td>
<td>54.9</td>
<td>90.9</td>
<td>84.3</td>
<td>92.4</td>
<td>92.6</td>
<td>93.4</td>
</tr>
<tr>
<td>Memory-Bank (Wu et al., 2018)</td>
<td>66.4</td>
<td>88.6</td>
<td>61.3</td>
<td>91.6</td>
<td>84.3</td>
<td>92.7</td>
<td>93.5</td>
<td>94.8</td>
</tr>
<tr>
<td>MoCo (He et al., 2020)</td>
<td>71.1</td>
<td>89.9</td>
<td>61.9</td>
<td>91.2</td>
<td>84.5</td>
<td>92.7</td>
<td>93.5</td>
<td>94.6</td>
</tr>
<tr>
<td>KNN-BERT (Memory-Bank)</td>
<td>71.2</td>
<td>89.7</td>
<td>61.5</td>
<td>91.7</td>
<td>84.8</td>
<td>93.1</td>
<td>93.1</td>
<td>94.9</td>
</tr>
<tr>
<td>KNN-BERT (MoCo)</td>
<td>75.7</td>
<td>91.2</td>
<td>61.9</td>
<td>91.7</td>
<td>84.6</td>
<td>93.1</td>
<td>93.1</td>
<td>94.7</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>RTE</th>
<th>MRPC</th>
<th>QNLI</th>
<th>SST-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>66.4</td>
<td>88.9</td>
<td>90.5</td>
<td>93.5</td>
</tr>
<tr>
<td>KNN-BERT</td>
<td>70.2</td>
<td>89.1</td>
<td>90.8</td>
<td>93.5</td>
</tr>
</tbody>
</table>

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 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.

3https://gluebenchmark.com/
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 (ϕ = 0.5) (2) predictions are only made by the KNN classifier (ϕ = 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 (ϕ = 1) and the KNN & Linear combined classifier (ϕ = 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.
which can save the computation cost. When the \( K \) is less vital, which indicates that KNN based classifier is effec-
tive in improving downstream tasks. Combining 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.

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.

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.

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

<table>
<thead>
<tr>
<th>Methods</th>
<th>RTE</th>
<th>MRPC</th>
<th>M/L SP</th>
<th>RTE</th>
<th>MRPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_l = 1 )</td>
<td>67.8</td>
<td>91.5</td>
<td>( M_l = 1 )</td>
<td>74.7</td>
<td>91.1</td>
</tr>
<tr>
<td>( M_l = 10 )</td>
<td>71.5</td>
<td>91.3</td>
<td>( M_l = 50 )</td>
<td>73.8</td>
<td>91.3</td>
</tr>
<tr>
<td>( M_l = 20 )</td>
<td>71.5</td>
<td>91.3</td>
<td>( M_l = 100 )</td>
<td>75.7</td>
<td>91.2</td>
</tr>
<tr>
<td>( M_m = 100 )</td>
<td>73.3</td>
<td>90.4</td>
<td>( M_l = 1 )</td>
<td>67.1</td>
<td>90.0</td>
</tr>
<tr>
<td>( M_l = 100 )</td>
<td>73.3</td>
<td>90.3</td>
<td>( M_l = 200 )</td>
<td>66.4</td>
<td>91.5</td>
</tr>
<tr>
<td>( M_l = 200 )</td>
<td>67.8</td>
<td>90.2</td>
<td>( M_l = 400 )</td>
<td>67.1</td>
<td>91.1</td>
</tr>
</tbody>
</table>

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 classifier is effective and robust.
References


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.


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


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 few-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.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>RTE</th>
<th>CoLA</th>
</tr>
</thead>
<tbody>
<tr>
<td>BERT</td>
<td>80 s/epoch</td>
<td>96 s/epoch</td>
</tr>
<tr>
<td>KNN-BERT</td>
<td>162 s/epoch</td>
<td>173 s/epoch</td>
</tr>
</tbody>
</table>

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.

(a) Few-shot Linear Classifier  (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.