Making Pre-trained Language Models End-to-end Few-shot Learners with Contrastive Prompt Tuning

Anonymous ACL submission

Abstract

Prompt-based learning for Pre-trained Language Models (PLMs) has achieved remarkable performance in few-shot learning by exploiting prompts as task guidance and turning downstream tasks into masked language problems. In most existing approaches, the high performance of prompt-based learning heavily relies on handcrafted prompts and verbalizers, which may limit the application of such approaches in real-world scenarios. To solve this issue, we present *CP-Tuning*, the first end-to-end *Contrastive Prompt Tuning* framework for PLMs *without any manual engineering of task-specific prompts and verbalizers*. It is integrated with the task-invariant continuous prompt encoding technique with fully trainable prompt parameters. We further propose a pair-wise cost-sensitive contrastive loss to optimize the model in order to achieve verbalizer-free class mapping and enhance the task-invariance of prompts. Experiments over a variety of NLP tasks show *CP-Tuning* consistently outperforms state-of-the-art methods. [1]

1 Introduction

Starting from BERT (Devlin et al., 2019), fine-tuning Pre-trained Language Models (PLMs) has become the *de facto* standard practice for solving a majority of NLP tasks (Yang et al., 2019a; Lan et al., 2020; Sun et al., 2021). To guarantee high accuracy, it is necessary to obtain a sufficient amount of training data for downstream tasks, which is the bottleneck in low-resource scenarios.

The successful application of GPT-3 (Brown et al., 2020) shows that with a sufficiently large memory capacity and massive pre-training computation, large PLMs can learn to solve a task with very few training samples. However, the large model size and the long inference time make it infeasible to deploy such PLMs online with limited computational resources. Inspired by these works, Gao et al. (2021a) propose a prompt-based approach to fine-tune BERT-style PLMs in a few-shot learning setting. It converts text classification and regression problems into masked language problems where the knowledge captured during pre-training can be better utilized during the few-shot learning process. Similar usage of prompts has also been shown in Schick and Schütze (2021a,b) and many others. Scao and Rush (2021) conduct a rigorous test to show that prompting is highly beneficial in low-data regimes.

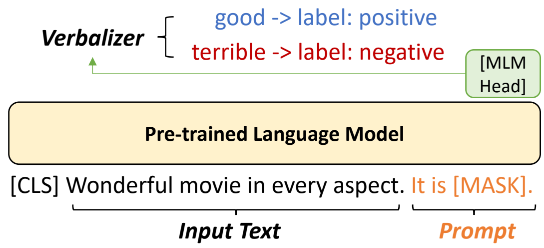


Figure 1: Prompt and verbalizer in classical prompt-based fine-tuning for review sentiment analysis.

In most prompt-based approaches, there exist two types of model components that require careful manual engineering, namely *prompts* and *verbalizers*. Here, prompts are fixed templates or patterns that are employed to inject task-specific guidance to input texts, while verbalizers establish explicit mappings between output tokens and class labels. An example of prompts and verbalizers on review sentiment analysis is illustrated in Figure 1. As reported in Liu et al. (2021b), designing high-performing prompts and the corresponding verbalizers is challenging and requires a very large validation set. As for prompts, even a slight change of expressions can lead to big variance in the performance of downstream tasks. To alleviate this issue, Liu et al. (2021b) propose P-tuning, which uses *continuous prompt embeddings* to avoid the manual

[1] All datasets are publicly available. Source codes are provided in the attachments and will be released to public. We further give a theoretical analysis on the pair-wise cost-sensitive contrastive loss in the appendix.

prompt engineering process. However, this method still requires the design of verbalizers, with a strong hypothesis of token-to-label mappings. Therefore, the drawbacks of prompt engineering potentially hinder the wide application of these approaches.

We present *CP-Tuning*, an end-to-end *Contrastive Prompt Tuning* framework for PLMs *without the manual design of task-specific prompts and verbalizers*. To our knowledge, our work is the first to study contrastive learning for prompting PLMs without manual prompt engineering. Specifically, our approach consists of two major techniques:

Task-invariant Continuous Prompt Encoding. We employ *continuous embeddings* as prompts and do not employ any prompt encoders to avoid learning additional parameters during few-shot learning (in contrast to Liu et al. (2021b)). Specially, we initialize continuous prompt embeddings as the pre-trained representations of a collection of *task-invariant* tokens, and enable prompt embeddings to be *task-adaptive* by back propagation.

Verbalizer-free Class Mapping. We propose the *verbalizer-free* mechanism to ease the manual labor of designing verbalizers and to improve the generalization ability of our model, as well as the task-invariance of prompts. Specifically, the *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) is introduced to train our few-shot learner, together with an auxiliary Mask Language Modeling (MLM) task as the regularizer. *PCCL* explicitly learns to distinguish different classes and makes the decision boundary smoother by assigning different costs to easy and hard cases. In contrast to previous approaches, embeddings of instances before the MLM classifier are directly used for inference.

For evaluation, we conduct extensive experiments to verify the effectiveness of *CP-Tuning* over eight public NLP datasets, including review sentiment analysis, sentence paraphrase, natural language inference, etc. Experimental results show that *CP-Tuning* consistently outperforms state-of-the-art for prompt-based few-shot learning. In summary, we make the following contributions:

- We introduce the end-to-end *CP-Tuning* framework to enable prompt-based few-shot learning without designing task-specific prompts and verbalizers. To our knowledge, our work is the first to employ contrastive learning for end-to-end prompt-based learning that eases manual engineering.

- In *CP-Tuning*, the task-invariant continuous prompt encoding technique is presented. We further propose the *PCCL* technique to train the model without the usage of any verbalizers based on contrastive learning.

- Experiments over eight public datasets show that *CP-Tuning* consistently outperforms state-of-the-arts for prompt-based few-shot learning. We also theoretically derive the relations between *PCCL* and other losses.

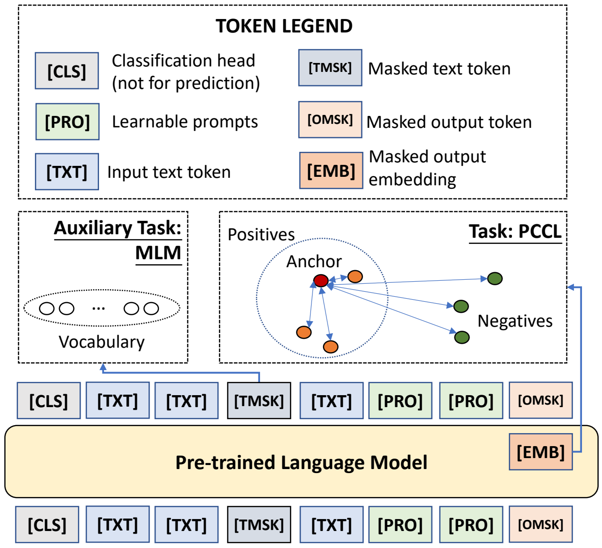


Figure 2: Framework overview. For simplicity, we only show text sequences for single-sentence classification.

2 *CP-Tuning*: Proposed Approach

We begin with an overview of our approach. After that, the detailed techniques are elaborated.

2.1 Overview of *CP-Tuning*

Let $\mathcal{D}$ be an N-way K-shot training set of a specific NLP task, where each of the N classes is associated with K training samples. [2] Denote $\mathcal{M}$ as the collection of parameters of the underlying PLM. The goal of our work is to generate a high-performance few-shot learner initialized from $\mathcal{M}$ based on $\mathcal{D}$ that can effectively generalize to previously unseen samples of the same task. The overview of our approach is in Figure 2, with major techniques summarized below.

As traditional prompt-based models require the cumbersome process of prompt engineering, we

[2] Our work can be easily extended to standard fine-tuning scenarios without modification where each class is associated with different numbers of training samples. We also find *CP-Tuning* is better at learning with unbalanced training sets than previous methods. Refer to experiments for details.

Task Type	Example of Input Sequence
Single-sentence	[CLS] Movie [TMSK], get ready to take off... the other direction. It is [OMSK]
Sentence-pair	[CLS] What was Telenet? [OMSK] *Telenet was [TMSK] in 1973 and started operations in 1975.*

Table 1: Examples of input token sequences. Texts underlined in the inputs refer to the *universal task-invariant prompts*. The second sentence in sentence-pair classification is printed in italic.

employ *continuous embeddings* as input prompts. Rather than employing sub-networks (*e.g.,* LSTMs) as prompt encoders (Liu et al., 2021b), to avoid learning additional parameters during few-shot learning, we directly feed prompt embeddings to the PLM encoder, and enable the embeddings to be *task-adaptive* by back propagation.

Besides manually-designed patterns, previous methods also require handcrafted verbalizers, which map the output of the masked token to the class label (Schick and Schütze, 2021a,b). In our work, we propose the *verbalizer-free* mechanism to ease the manual labor and to improve the generalization ability of our few-shot learner. As prompts and verbalizers are semantically correlated, this technique also enhances the task-invariance of prompts. Inspired by the contrastive learning paradigm (Jaiswal et al., 2020), we propose the *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) to train our few-shot learner. In the few-shot learning setting, the lack of training data may easily result in model over-fitting. Hence, an auxiliary MLM loss is also optimized during few-shot learning to alleviate the issue. In addition, we further show that *PCCL* is an extension to a variety of loss functions in the appendix.

2.2 Task-invariant Prompt Encoding

The input format of our approach is significantly different from previous works to facilitate *task-invariant continuous prompt learning*. To be more specific, in contrast to Devlin et al. (2019), we have three additional types of special tokens:

- [PRO]: the placeholder for prompts;
- [TMSK]: the token mask of the input texts for optimizing the auxiliary MLM loss;
- [OMSK]: the token mask as a placeholder to generate the output result.

For a better understanding, please refer to an example for single-sentence classification in Figure 2.[3] Here, "[TMSK]" is only applied to a small portion of the input texts for MLM. "[OMSK]" is used for generating outputs, rather than the "[CLS]" token. Hence, no additional parameters are introduced to our model for prompt learning.

As the parameters w.r.t. "[PRO]" tokens need to be learned for a given task, the lack of training data in few-shot learning still brings some burdens. Inspired by GPT-3 (Brown et al., 2020) and T5 (Raffel et al., 2020), we initialize prompt embeddings to be the pre-trained representations of *universal task-invariant prompts*.[4] Readers can also refer to the examples in Table 1.

2.3 Verbalizer-free Class Mapping

A common property of existing prompt-based approaches is that they require handcrafted verbalizers to establish mappings between tokens and class labels (Schick and Schütze, 2021a,b; Liu et al., 2021b). We suggest that this practice might be suboptimal. Consider the example on review analysis in Figure 3. *Verbalizer-based* approaches generate the distributions over the entire vocabulary (which may contain over 10 thousand words), and only pay attention to the probabilities of very few words (such as "good" and "terrible" in our case). The semantic association between words is also ignored to a large extent. For example, the probabilities of words such as "nice", "fantastic", "bad" and "horrible" are also strong indicators of class labels. If we replace the high-dimensional, sparse distributions with lower-dimensional, dense representations, the generalization ability and the flexibility of the underlying model can be largely increased.

In our work, we propose a novel *verbalizer-free* approach to generate model outputs based on *PCCL*. During training, denote $\mathcal{B}$ as the collection of instances in a batch ($\mathcal{B} \subset \mathcal{D}$). Each instance $i \in \mathcal{B}$ can be treated as an *anchor*, with the label denoted as y_i. We also have the *positive set* $P(i)$

[3]For sentence-pair tasks, the input format can be [CLS][TXT][TXT][PRO][PRO][OMSK][TXT][TMSK][TXT]. The "[PRO]" and "[OMSK]" tokens are placed between text pairs to better capture the relations between them.

[4]Here, the universal task-invariant prompt for single-sentence classification tasks is "it is"; and "?" for sentence-pair classification tasks. Refer to examples in Table 1. This setting can be viewed as the *knowledge prior* for prompt embeddings. During model training, the representations of prompts can be automatically adapted to specific tasks. In the experiments, we further show that it is unnecessary to design task-specific prompts for our approach. Hence, we do not need to vary the numbers and positions of "[PRO]" tokens for model tuning.

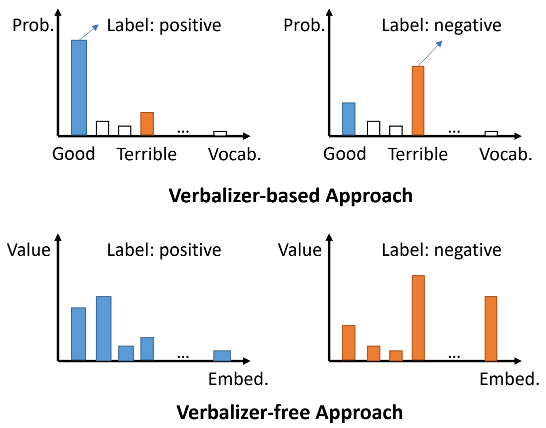


Figure 3: A simple comparison between *verbalizer-based* and *verbalizer-free* approaches w.r.t. model outputs. The underlying task is review sentiment analysis.

and the *negative set* $N(i)$ w.r.t. the instance i and the batch $\mathcal{B}$: $P(i) = \{j | j \neq i, y_j = y_i, j \in \mathcal{B}\}$ and $N(i) = \{j | y_j \neq y_i, j \in \mathcal{B}\}$.

Let $\vec{z}_i$ be the l_2-normalized embedding of the "[OMSK]" token of the last layer of the underlying PLM (before the *softmax* function). In the context of contrastive learning, we aim to maximize the within-class similarity $s_{i,p} = \vec{z}_i^T \cdot \vec{z}_p$ where $p \in P(i)$, and also minimize the between-class similarity $s_{i,n} = \vec{z}_i^T \cdot \vec{z}_n$ where $n \in N(i)$. Following previous supervised contrastive learning models (Khosla et al., 2020; Gao et al., 2021b), it is straightforward to derive the *sample-wise* contrastive loss:

$$\mathcal{L}_{CL}(i) = -\log \frac{\exp(s_{i,p}/\tau)}{\exp(s_{i,p}/\tau) + \exp(s_{i,n}/\tau)} \quad (1)$$

where τ is the temperature value. When multiple instances in $P(i)$ and $N(i)$ are considered, we rewrite $\mathcal{L}_{CL}(i)$ as follows:

$$\mathcal{L}_{CL}(i) = -\log \sum_{p \in P(i)} \frac{\exp(s_{i,p}/\tau)}{\sum_{a \in A(i)} \exp(s_{i,a}/\tau)} \quad (2)$$

where the collection $A(i) = \mathcal{B} \setminus \{i\}$. This gives the model more generalization abilities in that multiple within-class and between-class similarity values are averaged, thus making the learned decision boundary smoother.

Minimizing $\mathcal{L}_{CL}(i)$ alone may be insufficient as it does not consider sample difficulty. For example, if $s_{i,p} = 0.2$ and $s_{i,p'} = 0.95$ where $p, p' \in P(i)$. The model should pay more attention to $s_{i,p}$ to reach the optima, and less attention to $s_{i,p'}$ to avoid model over-fitting. Inspired by (Sun et al., 2020a), we introduce *pair-wise relaxation factors* and propose a new loss function named *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) as follows:

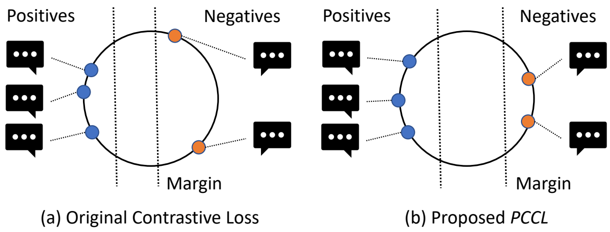


Figure 4: Illustration of how *PCCL* improves the learning process of "[OMSK]" embeddings of the last transformer encoder layer for review sentiment analysis.

$$\mathcal{L}_{PCCL}(i) = -\sum_{p \in P(i)} \log \frac{\exp(\alpha_{i,p} \cdot s_{i,p}/\tau_p)}{\mathcal{Z}(i)} \quad (3)$$

where $\mathcal{Z}(i)$ is the normalization factor:

$$\mathcal{Z}(i) = \sum_{p \in P(i)} \exp(\frac{\alpha_{i,p}}{\tau_p} s_{i,p}) + \sum_{n \in N(i)} \exp(\frac{\alpha_{i,n}}{\tau_n} s_{i,n}) \quad (4)$$

$\alpha_{i,p}$ and $\alpha_{i,n}$ are *pair-wise relaxation factors* with the definitions as follows:

$$\begin{aligned} \alpha_{i,p} &= \max\{0, 1 + m - s_{i,p}\} \\ \alpha_{i,n} &= \max\{0, s_{i,n} + m\} \end{aligned} \quad (5)$$

Comparing to the original $\mathcal{L}_{CL}(i)$, two new features are added to *PCCL*. Inside $\alpha_{i,p}$ and $\alpha_{i,n}$, a margin factor m is employed to expect that $s_{i,p} > 1 - m$ and $s_{i,n} < m$. Hence, there is a relaxed margin between $s_{i,p}$ and $s_{i,n}$. The usage of $\alpha_{i,p}$ and $\alpha_{i,n}$ also makes the model focus on learning hard cases and avoid over-fitting on easy cases. Another empirical setting is that we use separate temperatures τ_p and τ_n for within-class and between-class similarities, instead of a uniform temperature τ. We further set $\tau_p = \xi \cdot \tau_n$ ($\xi > 1$) to give more relaxations on positive samples in order to make the within-class similarities not too large, as it is easy to see:

$$\frac{\alpha_{i,p}}{\tau_p} s_{i,p} = \frac{\alpha_{i,p}}{\xi \cdot \tau_n} s_{i,p} = \frac{\tilde{\alpha}_{i,p}}{\tau_n} s_{i,p} \quad (6)$$

where $\tilde{\alpha}_{i,p} = \max\{0, \frac{1}{\xi}(1 + m - s_{i,p})\}$. In this way, our few-shot learner will be less likely to over-fit to training instances. We further provide an illustrative example in Figure 4 and a brief theoretical analysis on *PCCL*.

2.4 Auxiliary Masked Language Modeling

As the learning objective of *PCCL* is significantly different from the MLM task, minimizing $\mathcal{L}_{PCCL}(i)$ only may result in the *catastrophic forgetting* of the pre-training knowledge. Similar to Schick and Schütze (2021a,b), we treat MLM as an auxiliary task during few-shot learning to improve the model performance on previously unseen instances. Denote the *sample-wise* MLM loss as $\mathcal{L}_{MLM}(i)$. The *sample-wise* overall loss function $\mathcal{L}(i)$ can be written as follows: $\mathcal{L}(i) = \lambda \cdot \mathcal{L}_{PCCL}(i) + (1 - \lambda) \cdot \mathcal{L}_{MLM}(i)$ where λ is a pre-defined balancing hyper-parameter. In Figure 2, we apply the auxiliary MLM task to "[TMSK]" tokens and *PCCL* to "[OMSK]" tokens, separately. This practice can be viewed as performing *task-specific continual pre-training* (Sun et al., 2020b) and few-shot learning at the same time.

2.5 Model Inference

During the model inference time, because we do not tune the "[CLS]" prediction head, we directly take the embedding $\vec{z}_i$ of a testing instance i to generate the class label $\hat{y}_i$ by comparing $\vec{z}_i$ against the k-nearest neighbors in the few-shot training set. When *CP-Tuning* is applied to larger training sets, for better scalability, the label $\hat{y}_i$ is predicted by:

$$\hat{y}_i = \text{argmax}_{c \in \mathcal{C}} \vec{z}_i^T \cdot \vec{z}_c \tag{7}$$

where $\mathcal{C}$ is the collection of the class labels, and $\vec{z}_c$ is the prototype embedding of the class $c \in \mathcal{C}$ (*i.e.,* the averaged embedding of all training instances with the class label as c). Hence, this practice is closely in line with *prototypical networks* (Snell et al., 2017; Ji et al., 2020).

3 Experiments

We conduct extensive experiments to evaluate *CP-Tuning* and compare it against state-of-the-arts.

3.1 Datasets and Experimental Settings

In the experiments, we employ eight public NLP datasets to evaluate *CP-Tuning*: three for review sentiment analysis (SST-2 (Socher et al., 2013), MR (Hu and Liu, 2004) and CR (Pang and Lee, 2005)), two for sentence paraphrase (MRPC (Dolan and Brockett, 2005) and QQP [5]), two for natural language inference (QNLI (Rajpurkar et al., 2016) and RTE (Bar-Haim et al., 2014)) and one for subjectivity classification (SUBJ (Pang and Lee, 2004)). The dataset statistics are summarized in Table 3. For few-shot learning, the evaluation protocols and the training/development/testing splits are the same as in Gao et al. (2021a). The underlying PLM is the RoBERTa large model (with 335M parameters) (Liu et al., 2019). In the experiments, we set $K = 16$ and measure the average performance in terms of accuracy across 5 different randomly sampled training and development splits. Hence, the performance of *CP-Tuning* can be rigorously evaluated with a minimal influence of random seeds or datasets.

In the experiments, we employ the standard fine-tuning approach (Devlin et al., 2019) [6], PET (Schick and Schütze, 2021a,b) [7], LM-BFF (Gao et al., 2021a) (with three different settings: Auto T, Auto L and Auto T+L) [8] and P-tuning (Liu et al., 2021b) [9] as strong baselines. Specifically, PET, LM-BFF and P-tuning are recent state-of-the-art approaches for prompt-based few-shot learning. As the experimental settings of PET, LM-BFF and P-tuning are different, in order to conduct a rigorous comparison, we re-produce the results based on their open-source codes and the same set of random seeds. Hence, the results reported in our work are slightly different from their original papers. Our own *CP-Tuning* algorithm is implemented in PyTorch and run with NVIDIA V100 GPUs. In default, we set $\tau_p = 2$, $\tau_n = 1$ (with $\xi = 2$), $\lambda = 0.5$, $m = 0.3$ and $k = 3$, and also tune the parameters over the few-shot development sets. The model is trained with the Adam optimizer (Kingma and Ba, 2015), with the learning rate and the batch size tuned around $\{1e-5, 3e-5, 5e-5\}$ and $\{4, 8, 16\}$, respectively. The optimization of auxiliary MLM is the same as PET. We also study how the change of some important hyper-parameters affect the overall performance, with results reported below.

3.2 Overall Performance Comparison

The experimental results of *CP-Tuning* and all baselines on eight testing sets for few-shot learning are presented in Table 2. From the experimental

[5] https://www.quora.com/q/quoradata/

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

[7] https://github.com/timoschick/pet

[8] https://github.com/princeton-nlp/LM-BFF. In LM-BFF, "Auto T", "Auto L" and "Auto T+L" refer to automatically generated templates, labels and both (reported by Gao et al. (2021a)), respectively.

[9] https://github.com/THUDM/P-tuning

Method/Task	SST-2	MR	CR	MRPC	QQP	QNLI	RTE	SUBJ	Average
Standard Fine-tuning	78.62	76.17	72.48	64.40	63.01	62.32	52.28	86.82	69.51
PET	92.06	87.13	87.13	66.23	70.34	64.38	65.56	91.28	78.01
LM-BFF (Auto T)	90.60	87.57	90.76	66.72	65.25	68.87	65.99	91.61	78.42
LM-BFF (Auto L)	90.55	85.51	91.11	67.75	70.92	66.22	66.35	90.48	78.61
LM-BFF (Auto T+L)	91.42	86.84	90.40	66.81	61.61	61.89	66.79	90.72	77.06
P-tuning	91.42	87.41	90.90	71.23	66.77	63.42	67.15	89.10	78.43
***CP-Tuning* (Our Approach)**	**93.35**	**89.43**	**91.57**	**72.60**	**73.56**	**69.22**	**67.22**	**92.27**	**81.24**

Table 2: Comparison between *CP-Tuning* and baseline methods over the testing sets in terms of accuracy (%).

Task	#Training	#Testing	Task Group Labels
SST-2	6,920	872	
MR	8,662	2,000	positive, negative
CR	1,775	2,000	
MRPC	3,668	408	equivalent,
QQP	363,846	40,431	not equivalent
QNLI	104,743	5,463	entailment,
RTE	2,490	277	not entailment
SUBJ	8,000	2,000	subjective, objective

Table 3: Dataset statistics. We only sample $K \times |\mathcal{C}|$ instances from the original training sets to form few-shot training and development sets.

Method/Task	SST-2	MR	MRPC	QQP
Full Implement.	**93.35**	**89.43**	**72.60**	**73.56**
w/o. auxiliary MLM	91.35	86.67	71.96	72.47
w/o. $\alpha_{i,p}$ and $\alpha_{i,n}$	92.50	88.59	68.28	69.32
w/o. similarity avg.	92.04	86.37	67.11	69.14

Table 4: Ablation study of *CP-Tuning* on four tasks in terms of accuracy (%). "Full Implement." refers to the full implementation of our method. The lowest accuracy scores over each dataset are printed underlined.

results, we can draw the following conclusions. Prompt-based methods (such as PET, LM-BFF and P-tuning) outperform standard fine-tuning by a large margin. This shows that prompts are highly useful for few-shot learning over PLMs. Based on our re-production results, LM-BFF (with different settings) and P-tuning have similar performance, while PET produces slightly lower performance. The performance gains of *CP-Tuning* over all the testing sets are consistent, compared to all the state-of-the-art methods. Overall, the average improvement is around 3% in terms of accuracy. It can be seen that even without task-specific prompts and verbalizers, *CP-Tuning* is capable of producing high-accuracy models with few training instances. We also conduct *paired t-tests* to compare the accuracy scores on all tasks produced by *CP-Tuning* against LM-BFF and P-tuning. Experimental results show that the improvement of *CP-Tuning* is statistically significant (with the p-value $p < 0.05$).

3.3 Detailed Model Analysis

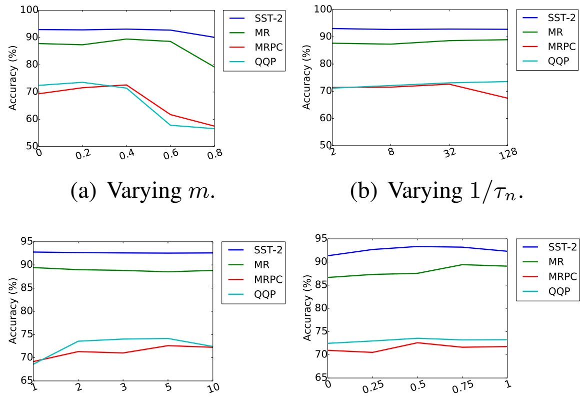
We further study how *CP-Tuning* improves the model performance in various aspects. Here, we treat SST-2, MR, MRPC and QQP as pilot tasks to explore our method.

Ablation Study. The ablation results of *CP-Tuning* are shown in Table 4. Here, "w/o. auxiliary MLM" refers to the variant of *CP-Tuning* without the auxiliary MLM task; "w/o. $\alpha_{i,p}$ and $\alpha_{i,n}$" refers to *CP-Tuning* without pair-wise relaxation factors; and "w/o. similarity averaging" refers to the setting where we only consider one positive and one negative instance for each anchor using standard triplet loss. From the results we can see that all three techniques contribute to the overall accuracy improvement. Specifically, auxiliary MLM has the most influence over SST-2, while similarity averaging contributes the most over the remaining three datasets. [10]

Parameter Analysis. We also show how some of the important hyper-parameters in *CP-Tuning* affect the performance over the four datasets. The results are shown in Figure 5. We can see the the trends are almost consistent across all the datasets. The optimal setting of the margin m is around 0.2. As for the temperature, the optimal value of τ_n is around $1/8$ to $1/32$, which is different from other works where the default temperature is 1. This is probably due to the fact that we compute the total scores $\alpha_{i,p} \cdot s_{i,p}/\tau_p$ and $\alpha_{i,n} \cdot s_{i,n}/\tau_n$, which are different from those in other works in contrastive learning. Nevertheless, the performance of *CP-Tuning* is not very sensitive to the choice of the temperature, proving that *CP-Tuning* is highly general for real-world applications.

We further tune the value of ξ. As seen in the

[10]Note that the performance drops by large margin when we remove the MLM task for SST-2 and MR. This is because the few-shot learning ability of PLMs is largely based on the utilization of pre-trained knowledge learned by MLM. In *CP-Tuning*, the *PPCL* objective is significantly different from MLM, hence optimizing *PPCL* alone may lead to the catastrophic forgetting of the MLM knowledge. We suggest that the auxiliary MLM task in *CP-Tuning* is vital for obtaining the high performance.



(a) Varying m. (b) Varying $1/\tau_n$.

(c) Varying ξ. (d) Varying λ.

Figure 5: Parameter analysis on hyper-parameters.

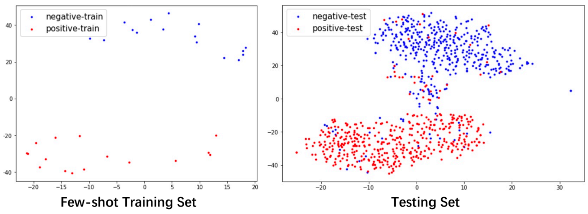


Figure 6: Visualizations of "[OMSK]" embeddings of SST-2 by t-SNE. (Best viewed in color.)

figure, for sentence-pair tasks, the optimal ξ is between 2 to 5, while easier single sentence tasks are not sensitive to this hyper-parameter. We also try using the prototype embeddings $\vec{z}_c$ for model inference, of which the results are similar. We suggest that when *CP-Tuning* is applied to large datsets, it is suitable to predict the class label $\hat{y}_i$ by $\text{argmax}_{c \in C} \vec{z}_i^T \cdot \vec{z}_c$ for better scalability. In PET (Schick and Schütze, 2021a,b), the auxiliary MLM task is applied with $\lambda = 1e{-}4$, which is sufficiently small. In contrast to their work, we suggest that the optimal value of λ is in the range between 0.5 to 0.75.

Visualizations. To show that the generated "[OMSK]" embeddings are separable for text classification, we plot the embeddings of the few-shot training and testing data in SST-2. The results are illustrated in Figure 6. The underlying dimension reduction and visualization algorithm is t-SNE (van der Maaten and Hinton, 2008). As seen, even reduced in two dimensions, most of the embeddings in the testing set are clearly separated, with the only $N \times K$ training samples available. Additionally, the embeddings in the few-shot training set are widely spread, showing the generalization of our algorithm.

Method/Task	SST-2	MR	MRPC	QQP
PET	87.25	83.44	64.61	58.82
LM-BFF	88.10	83.51	65.98	59.19
P-tuning	87.92	83.20	66.64	61.27
CP-Tuning	**91.25**	**86.52**	**70.12**	**65.52**

Table 5: Testing results of *CP-Tuning* and baseline methods for unbalanced few-shot learning in terms of accuracy (%).

3.4 Learning with Unbalanced Datasets

In the literature, few-shot learning is formulated as an *N-way-K-shot* problem. However, it may not be the case in real-world applications. In this set of experiments, we consider the situation where the few-shot training set is unbalanced. Following previous experiments, four binary classification tasks are used for evaluation, namely SST-2, MR, MRPC and QQP. In each few-shot training set, we assume there are 8 and 24 instances of the two classes, instead of setting $K = 16$. We compare *CP-Tuning* against three strong baselines for few-shot learning (*i.e.,* PET, LM-BFF and P-tuning). The results are shown in Table 5. As seen, *CP-Tuning* consistently outperforms these baselines by a large margin. The improvement rates are also larger than those in standard few-shot learning scenarios (as reported in Table 2). This is because *CP-Tuning* focuses on learning the distinctions between positive and negative samples, instead of tuning the MLM head (as in previous approaches).

3.5 Study on Task-invariance of Prompts

In *CP-Tuning*, we initialize prompt embeddings as the pre-trained representations of *universal task-invariant prompts* and utilize the *verbalizer-free* mechanism. In the following experiments, we aim to study whether *CP-Tuning* is capable of generating more stable and accurate results compared to the non-contrastive baseline (*i.e.,* PET (Schick and Schütze, 2021a,b)). We consider two review sentiment analysis datasets: SST-2 and MR, as well as two paraphrase datasets: MRPC and QQP. Five prompt settings are employed: the *universal task-invariant prompts* used in *CP-Tuning* and the manually designed prompts used in PET (Schick and Schütze, 2021a,b). In Table 6, we present the averaged accuracy and its standard deviation of *CP-Tuning* and PET, under five different prompt settings. We can see that compared to PET, *CP-Tuning* has a higher accuracy and a lower deviation when the prompts change. This finding is different from previous works, showing that *CP-Tuning* is not

Task/Method	*CP-Tuning*		PET	
	Acc.	Std.	Acc.	Std.
SST-2	92.91*	0.56*	91.28	1.38
MR	88.38*	1.46	86.28	1.70
MRPC	71.80*	2.20*	65.73	5.08
QQP	73.84*	2.16*	66.61	5.22

Table 6: Method comparison with five sets of prompts in terms of averaged accuracy (%) and standard deviation. * refers to statistical significance of higher accuracy and lower deviation at 95% confidence interval.

sensitive to different prompts. Hence, we suggest learning with task-invariant prompts and no verbalizers is a desirable setting that reduces the amount of human labor. Additionally, during the learning process, prompt embeddings can be automatically adapted to fit specific tasks.

4 Related Work

PLMs. PLMs have achieved significant improvements on various NLP tasks. Readers can refer to the survey (Qiu et al., 2020). Among these PLMs, ELMo (Peters et al., 2018) learns contextual word representations by self-supervised pre-training using bidirectional LSTMs. BERT (Devlin et al., 2019) is probably the most popular model, which learns contextual representations of tokens by transformer encoders. Other PLMs based on the transformer encoder architecture include ALBERT (Lan et al., 2020), Transformer-XL (Dai et al., 2019), XLNet (Yang et al., 2019a), StructBERT (Wang et al., 2020), Big Bird (Zaheer et al., 2020) and many others. Apart from the encoder-based PLMs, the encoder-decoder and the auto-regressive decoder architectures are used in T5 (Raffel et al., 2020) and the GPT series (Brown et al., 2020). As the neural architectures are not our major focus, we do not elaborate.

Prompting PLMs for Few-shot Learning. With the prevalence of GPT-3 (Brown et al., 2020), prompting PLMs for few-shot learning has become a new, popular learning paradigm. A recent survey can be found in Liu et al. (2021a). To name a few, PET (Schick and Schütze, 2021a,b) turns text classification into cloze-style problems and use manually-defined prompts to provide additional task guidance. To facilitate automatic prompt discovery, Gao et al. (2021a) generate prompts from the T5 model (Raffel et al., 2020). Jiang et al. (2020) also mine high-performing prompts from the training corpus. AutoPrompt (Shin et al., 2020) employs gradient searching to detect prompts. However, these approaches focus on discrete prompts only. P-tuning (Liu et al., 2021b) learns continuous prompt embeddings with differentiable parameters for GPT-based models. Prefix-tuning (Li and Liang, 2021) extends the usage of continuous prompts for text generation tasks. Min et al. (2021) propose a noisy channel model for prompt learning. WARP (Hambardzumyan et al., 2021) leverages continuous prompts to improve the model performance in fine-tuning scenarios. Knowledgeable prompt-tuning (Hu et al., 2021) optimizes the verbalizer construction process by integrating the knowledge from knowledge bases. Our work further applies contrastive learning to making the few-shot learner fully verbalizer-free.

Deep Contrastive Learning. Contrastive learning (Jaiswal et al., 2020) aims to learn an embedding space in which similar instances have similar embeddings while dissimilar instances fall apart. In the literature, several contrastive learning objectives have been proposed, such as the triplet loss (Schroff et al., 2015), the N-pair loss (Sohn, 2016), InfoNCE (van den Oord et al., 2018) and the supervised contrastive loss (Khosla et al., 2020). Due to its effectiveness, contrastive learning has been applied to various NLP tasks, *e.g.,* sentence representation (Gao et al., 2021b; Kim et al., 2021), text summarization (Wang et al., 2019), aspect detection (Shi et al., 2021), machine translation (Yang et al., 2019b), commonsense reasoning (Klein and Nabi, 2020). To our knowledge, *CP-Tuning* is the first to apply contrastive learning to prompt-based few-shot learning.

5 Conclusion and Future Work

In this work, we present an end-to-end *Contrastive Prompt Tuning* (*CP-Tuning*) framework that enables few-shot learning for PLMs without designing any task-specific prompts and verbalizers. In *CP-Tuning*, we employ task-invariant continuous prompt encoding and the *Pair-wise Cost-sensitive Contrastive Loss* (*PCCL*) to train the model. Experiments over eight public datasets show that *CP-Tuning* consistently outperforms state-of-the-art methods. Future work of *CP-Tuning* includes: i) extending the *CP-Tuning* framework to other NLP tasks such as named entity recognition, machine reading comprehension and text generation; ii) combining *CP-Tuning* with transfer learning to improve the model performance in low-resource scenarios.

References

Roy Bar-Haim, Ido Dagan, and Idan Szpektor. 2014. Benchmarking applied semantic inference: The PASCAL recognising textual entailment challenges. In *Language, Culture, Computation. Computing - Theory and Technology*, volume 8001, pages 409–424.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *NeurIPS*.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G. Carbonell, Quoc Viet Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *IWP@IJCNLP*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. In *ACL/IJCNLP*, pages 3816–3830.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021b. Simcse: Simple contrastive learning of sentence embeddings. *CoRR*, abs/2104.08821.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: word-level adversarial reprogramming. In *ACL/IJCNLP*, pages 4921–4933.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *SIGKDD*, pages 168–177.

Shengding Hu, Ning Ding, Huadong Wang, Zhiyuan Liu, Juanzi Li, and Maosong Sun. 2021. Knowledgeable prompt-tuning: Incorporating knowledge into prompt verbalizer for text classification. *CoRR*, abs/2108.02035.

Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. 2020. A survey on contrastive self-supervised learning. *CoRR*, abs/2011.00362.

Zhong Ji, Xingliang Chai, Yunlong Yu, Yanwei Pang, and Zhongfei Zhang. 2020. Improved prototypical networks for few-shot learning. *Pattern Recognit. Lett.*, 140:81–87.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

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

Taeuk Kim, Kang Min Yoo, and Sang-goo Lee. 2021. Self-guided contrastive learning for BERT sentence representations. In *ACL/IJCNLP*, pages 2528–2540.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Tassilo Klein and Moin Nabi. 2020. Contrastive self-supervised learning for commonsense reasoning. In *ACL*, pages 7517–7523.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. ALBERT: A lite BERT for self-supervised learning of language representations. In *ICLR*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL/IJCNLP*, pages 4582–4597.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *CoRR*, abs/2103.10385.

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. *CoRR*, abs/1907.11692.

Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2021. Noisy channel language model prompting for few-shot text classification. *CoRR*, abs/2108.04106.

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*, pages 271–278.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*, pages 115–124.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. 2020. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pages 2383–2392.

Teven Le Scao and Alexander M. Rush. 2021. How many data points is a prompt worth? In *NAACL*, pages 2627–2636.

Timo Schick and Hinrich Schütze. 2021a. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*, pages 255–269.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *NAACL*, pages 2339–2352.

Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *CVPR*, pages 815–823.

Tian Shi, Liuqing Li, Ping Wang, and Chandan K. Reddy. 2021. A simple and effective self-supervised contrastive learning framework for aspect detection. In *AAAI*, pages 13815–13824.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *EMNLP*, pages 4222–4235.

Jake Snell, Kevin Swersky, and Richard S. Zemel. 2017. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*, pages 1631–1642.

Kihyuk Sohn. 2016. Improved deep metric learning with multi-class n-pair loss objective. In *NIPS*, pages 1849–1857.

Yifan Sun, Changmao Cheng, Yuhan Zhang, Chi Zhang, Liang Zheng, Zhongdao Wang, and Yichen Wei. 2020a. Circle loss: A unified perspective of pair similarity optimization. In *CVPR*, pages 6397–6406.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. 2021. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *CoRR*, abs/2107.02137.

Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020b. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI*, pages 8968–8975.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2605):2579–2605.

Hong Wang, Xin Wang, Wenhan Xiong, Mo Yu, Xiaoxiao Guo, Shiyu Chang, and William Yang Wang. 2019. Self-supervised learning for contextualized extractive summarization. In *ACL*, pages 2221–2227.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Jiangnan Xia, Zuyi Bao, Liwei Peng, and Luo Si. 2020. Structbert: Incorporating language structures into pre-training for deep language understanding. In *ICLR*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019a. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.

Zonghan Yang, Yong Cheng, Yang Liu, and Maosong Sun. 2019b. Reducing word omission errors in neural machine translation: A contrastive learning approach. In *ACL*, pages 6191–6196.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

A Appendix

A.1 Theoretical Analysis of *PCCL*

In this section, we theoretically show that *PCCL* is an extension to various metric learning based loss functions.

As *PCCL* is directly extended from the supervised contrastive loss (Khosla et al., 2020; Gao et al., 2021b) by adding *pair-wise relaxation factors*, it is trivial to see that the supervised contrastive loss is a special case of *PCCL* with $\alpha_{i,p} = \alpha_{i,n} = 1$ and $\tau_p = \tau_n$.

Next, we consider the triplet loss (Schroff et al., 2015). Assume that there are only one positive and one negative samples for each anchor. We simplify $\mathcal{L}_{PCCL}(i)$ as follows:

$$\begin{aligned}\mathcal{L}_{PCCL}(i)^{'} &= \log(1+\exp(\frac{\alpha_{i,p}}{\tau_p}s_{i,p} - \frac{\alpha_{i,n}}{\tau_n}s_{i,n})) \\ &= \log(1+\exp(\frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}s_{i,p} - \alpha_{i,n}s_{i,n}))\end{aligned} \tag{8}$$

If we set a small value for τ_n (close to 0, which is the case as shown in the experiments), then the value of $\frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}s_{i,p} - \alpha_{i,n}s_{i,n})$ is large. As a rough approximation, we have:

$$\begin{aligned}&\mathcal{L}_{PCCL}(i)^{'} \approx \frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}s_{i,p} - \alpha_{i,n}s_{i,n}) \\ &= \frac{1}{\tau_n}(\frac{\alpha_{i,p}}{\xi}\vec{z}_i^T\vec{z}_p - \alpha_{i,n}\vec{z}_i^T\vec{z}_n) \\ &\propto -\frac{1}{2\tau_n}(\frac{\alpha_{i,p}}{\xi}\|\vec{z}_i - \vec{z}_p\|^2 - \alpha_{i,n}\|\vec{z}_i - \vec{z}_n\|^2)\end{aligned} \tag{9}$$

Approximately speaking, the problem of minimizing $\mathcal{L}_{CCL}(i)^{'}$ is equivalent of optimizing the loss function $\mathcal{L}_{TL}(i)$ (with the margin omitted):

$$\mathcal{L}_{TL}(i) = \alpha_{i,n}\|\vec{z}_i - \vec{z}_n\|^2 - \frac{\alpha_{i,p}}{\xi}\|\vec{z}_i - \vec{z}_p\|^2 \tag{10}$$

which is the triplet loss with the positive and negative pair-wise weights to be $\frac{\alpha_{i,p}}{\xi}$ and $\alpha_{i,n}$, respectively. Therefore, the triplet loss has a close connection to *PCCL*.

As for the N-pair loss (Sohn, 2016), we consider the situation where there is one positive sample and multiple negative samples for each anchor. We re-write $\mathcal{L}_{PCCL}(i)$ as:

$$\begin{aligned}\mathcal{L}_{PCCL}(i)^{''} = &\log(1+ \\ &\sum_{n\in N(i)}\exp(\frac{\alpha_{i,p}}{\tau_p}s_{i,p} - \frac{\alpha_{i,n}}{\tau_n}s_{i,n}))\end{aligned} \tag{11}$$

By setting $\frac{\alpha_{i,p}}{\tau_p} = 1$ and $\frac{\alpha_{i,n}}{\tau_n} = 1$, we simplify *PCCL* into the N-pair loss. We can see that *PCCL* combines the advantages of both supervised learning and metric learning, specifically assigning different costs to easy and hard cases.