Differentiable Data Augmentation for Contrastive Sentence Representation Learning

Tianduo Wang and **Wei Lu**
StatNLP Research Group
Singapore University of Technology and Design
{tianduo_wang,luwei}@sutd.edu.sg

Abstract

Fine-tuning a pre-trained language model via the contrastive learning framework with a large amount of unlabeled sentences or labeled sentence pairs is a common way to obtain high-quality sentence representations. Although the contrastive learning framework has shown its superiority on sentence representation learning over previous methods, the potential of such a framework is under-explored so far due to the simple method it used to construct positive pairs. Motivated by this, we propose a method that makes *hard positives* from the original training examples. A pivotal ingredient of our approach is the use of prefix that is attached to a pre-trained language model, which allows for *differentiable* data augmentation during contrastive learning. Our method can be summarized in two steps: *supervised prefix-tuning* followed by *joint contrastive fine-tuning* with unlabeled or labeled examples. Our experiments confirm the effectiveness of our data augmentation approach. The proposed method yields significant improvements over existing methods under both semi-supervised and supervised settings. Our experiments under a low labeled data setting also show that our method is more label-efficient than the state-of-the-art contrastive learning methods. [1]

1 Introduction

Learning universal and effective sentence representations is an enduring problem in natural language processing (NLP). The objective of learning sentence representations is similar to that of learning word embeddings (Mikolov et al., 2013; Pennington et al., 2014), i.e., we expect the embeddings of sentences with similar semantics to be close to each other, while the embeddings of sentences with different meanings to be sufficiently far away from each other. Recently, we have witnessed the success of pre-trained language models in various NLP tasks. However, the quality of the sentence representations directly obtained from non fine-tuned language models remains unsatisfactory (Reimers and Gurevych, 2019; Li et al., 2020).

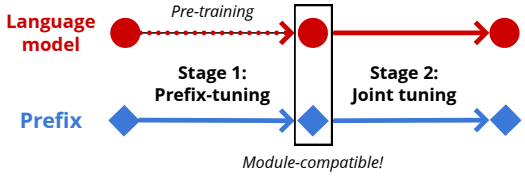


Figure 1: Our two-stage tuning strategy.

One approach to addressing this problem is to fine-tune a pre-trained language model via the contrastive learning objective on labeled or unlabeled sentences. Several recent research efforts (Yan et al., 2021; Gao et al., 2021b; Jiang et al., 2022) have shown that a contrastive learning based fine-tuning stage can produce state-of-the-art results on the sentence embedding learning task.

Some previous contrastive learning works on other modalities, e.g., image (Chen et al., 2020a,b), have shown that an effective data augmentation (DA) method is crucial for the success of contrastive learning. However, due to the discrete nature of language, applying commonly-used sentence augmentation strategies, e.g., word deletion and replacement, over the input sentences for contrastive learning leads to suboptimal results (Gao et al., 2021b). Instead, Gao et al. (2021b) propose to perform data augmentation via dropout, showing that this extremely simple approach can yield better sentence representations than previous DA methods that are based on discrete transformations.

Although this dropout-based DA method outperforms its discrete counterparts, the potential of the contrastive learning framework is under-explored due to its simple treatment of the positive pair construction process. Previous works have shown that contrastive learning benefits from strong data augmentations that can produce hard positives with

[1] Our code and model checkpoints are available at https://github.com/TianduoWang/DiffAug.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 7640 - 7653
December 7-11, 2022

meaningful differences. Specifically, Chen et al. (2020a) demonstrate that contrastive learning requires stronger data augmentation than traditional supervised learning. Tian et al. (2020) further show that, for contrastive learning, a good pair of positive instances should only share task-relevant information while discarding irrelevant information as much as possible.

Motivated by this idea, we propose **DiffAug**, a **diff**erentiable data **aug**mentation method for contrastive sentence representation learning. This method prepends two separate prefix modules (Li and Liang, 2021) to a common pre-trained language model. The goal is to obtain hard positive pairs with the help of these two prefix modules for contrastive learning, where supervised signals can be used to guide the tuning process of such prefix modules. Such a design essentially requires us to fine-tune both language model and newly-added prefix. However, how to effectively train them jointly to best serve our sentence representation learning purpose is a non-trivial research question.

Our observations show that, though tempting, it can be undesirable to fine-tune language model and prefix jointly from the beginning of the training process. Rather, we propose an effective two-stage tuning strategy, *prefix-tuning* followed by *joint tuning*, as illustrated in Figure 1. Specifically, we argue that it is crucial to perform prefix-tuning first, until a *module-compatible* state is achieved, before we move to the second stage to perform joint tuning for both modules. Our main contributions can be summarized as follows:

- We propose a novel and effective data augmentation method for contrastive sentence representation learning with the help of prefix modules. We further design a mechanism that allows this module to be carefully optimized with labeled data first, allowing hard positives with meaningful differences to be constructed, which can benefit contrastive learning.
- Our experiments show that the proposed method achieves the new state-of-the-art results on both semi-supervised and supervised settings. We also investigate the situation when labeled data is scarce. The results demonstrate that our method is more label-efficient than previous contrastive learning methods, showing the robustness of our proposed approach.
- Our work successfully combines the traditional fine-tuning and prefix-tuning paradigms. Through extensive analysis we identify the crucial elements required to ensure the success of our proposed approach.

2 Background

Our method is based on the state-of-the-art contrastive learning framework proposed by Gao et al. (2021b). We first introduce this framework. Next, we discuss the mechanism of prefix-tuning (Li and Liang, 2021).

2.1 The contrastive learning framework

Contrastive learning aims to learn meaningful representations from data by pulling together instances with similar semantic meanings and pushing apart dissimilar ones (Hadsell et al., 2006). The contrastive learning framework that we apply in this work is proposed by Gao et al. (2021b). There are three main components in this framework:

- A data augmentation module $g(\cdot\,;\theta)$ that generates positive pairs from a batch of training data with parameters denoted as θ. Previous contrastive learning works on other modalities (Chen et al., 2020a; Tian et al., 2020) have shown that the meaningful differences between positive pairs will allow the potential of contrastive learning to be better explored. Our work mainly improves this module.
- A neural network based encoder $f(\cdot\,;\phi)$ that maps input data to a representation space with parameters denoted as ϕ. For natural language, $f(\cdot\,;\phi)$ could be any neural network architecture that is suitable for encoding sentences. For our approach, we follow previous works (Gao et al., 2021b; Jiang et al., 2022), and employ pre-trained language models as $f(\cdot\,;\phi)$, e.g. BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019).[2]
- A contrastive loss function can be defined over a batch of data with size N as follows:

$$\mathcal{L}_{\text{cl}} = -\frac{1}{N}\sum_{i=1}^{N}\log\frac{\exp(\frac{\cos(h_{i,1},\,h_{i,2})}{\tau})}{\sum_{j=1}^{N}\exp(\frac{\cos(h_{i,1},\,h_{j,2})}{\tau})} \tag{1}$$

where $h_{i,1} = f(g(x_i;\theta_1);\phi)$ and $h_{i,2} = f(g(x_i;\theta_2);\phi)$ are the representations of the

[2]One common method to obtain sentence embedding is to use the `[CLS]` representation of the pre-trained language model. In this work, we add a hard prompt with a mask token `[MASK]` to each input sentence and obtain sentence embeddings from the `[MASK]` representations, following PromptBERT (Jiang et al., 2022).

augmented instances from the same data x_i, τ is the temperature hyperparameter, and $\cos(\cdot,\cdot)$ is the cosine similarity function. Note that we have two parameters θ_1 and θ_2, as we involve two separate data augmentation operations here.

2.2 The mechanism of prefix-tuning

Transformer (Vaswani et al., 2017) has become the most crucial component for many pre-trained language models. In this section, we discuss the self-attention module in Transformer and illustrate how prefix-tuning (Li and Liang, 2021) works. The original self-attention layer first maps the input $X \in \mathbb{R}^{L\times d}$ into three matrices, i.e., query $Q \in \mathbb{R}^{L\times d}$, key $K \in \mathbb{R}^{L\times d}$, and value $V \in \mathbb{R}^{L\times d}$, where L and d are the input length and hidden dimension respectively. The self-attention function is defined as:

$$H = \text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d}})V \tag{2}$$

where $H \in \mathbb{R}^{L\times d}$ is the output of the self-attention layer. Prefix-tuning (Li and Liang, 2021) affects the output of the original language model by prepending *tunable* matrices as key-value pairs. Specifically, in each self-attention layer, two matrices $P_k,\ P_v \in \mathbb{R}^{l\times d}$ will be concatenated to the original K and V respectively, where l is the prefix length. Therefore, in prefix-tuning, the self-attention function will become the following:

$$\begin{aligned} H_p &= \text{Attention}(Q, [P_k; K], [P_v; V]) \\ &= \text{softmax}(\frac{Q[P_k; K]^T}{\sqrt{d}})[P_v; V] \end{aligned} \tag{3}$$

where $[\cdot;\cdot]$ represents the concatenation on the first dimension.[3] Since our method applies prefix for data augmentation, we use θ to denote the set of *tunable* matrices $\{(P_k^i, P_v^i)|i = 1, ..., M\}$, where M is the number of self-attention layers in the pre-trained language model.

3 Method

In this section, we describe the proposed data augmentation method, and our two-stage tuning strategy. We then discuss reasons why our proposed approach works.

[3]We have $[P_k; K],\ [P_v; V] \in \mathbb{R}^{(l+L)\times d}$, and $H_p \in \mathbb{R}^{L\times d}$ is the output of self-attetion layer in prefix-tuning.

3.1 Differentiable data augmentation

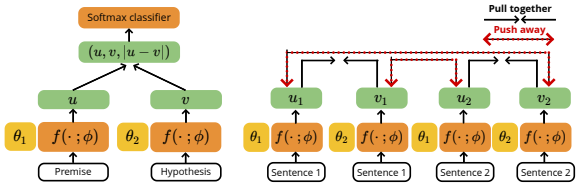
The InfoMin principle (Tian et al., 2020) states that a good pair of positive instances for contrastive learning should be as different as possible while retaining enough useful information relevant to the downstream tasks. Following this principle, we improve the current state-of-the-art contrastive learning framework (Gao et al., 2021b) via prefix (Li and Liang, 2021). We hope the added prefix modules can generate positive pairs with meaningful differences. To formulate this idea, we replace $h_i = f(g(x_i; \theta); \phi)$ with $h_i = f(x_i; \theta, \phi)$, since the prepended prefix modules can be regarded as a part of the language model.

We initialize the prefix modules randomly following Li and Liang (2021). This step allows the two initial representations returned from the two networks (with two different prefix modules) to be reasonably different. The next question is how to inject task-relevant information into the prefix. Previous works (Conneau et al., 2017; Reimers and Gurevych, 2019) have shown that training encoders with natural language inference (NLI) datasets (Bowman et al., 2015; Williams et al., 2018) via cross entropy loss can enhance the quality of generated sentence embeddings. It motivates us to use this objective to train our prefix modules. Since θ is initialized randomly, and ϕ is obtained from pre-training, we believe directly tuning θ and ϕ together from the beginning may not lead to the optimal results. Instead, we propose a two-stage tuning strategy: during the first stage, we only tune the prefix via cross entropy loss over the NLI dataset with the language model fixed, while in the second stage, we tune θ and ϕ jointly via the contrastive learning framework.

Stage 1: prefix-tuning. There are two main objectives for stage-1 tuning. First, we hope the two prefix modules to have meaningful differences such that they are capable of capturing the relation between NLI sentence pairs. Second, we expect the added prefix modules to be compatible with the original language model, so that they can be trained jointly in stage 2. To fulfill these two objectives, we optimize:

$$\min_{\theta_1,\theta_2} \mathcal{L}_{\text{ce}}(f(X_p; \theta_1, \phi), f(X_h; \theta_2, \phi), Y) \tag{4}$$

where X_p and X_h represent premise and hypothesis sentences from the NLI dataset respectively, and Y consists of binary labels indicating the relation between sentence pairs formed from X_p and



(a) Stage 1: prefix-tuning (b) Stage 2: joint tuning

Figure 2: The proposed DiffAug method has two training stages: (a) use cross entropy loss to make prefix modules capable of capturing the relationship between sentence pairs with the language model fixed; (b) use the contrastive learning objective to optimize both language model and prefix modules.

X_h. Here θ_1 and θ_2 represent the parameters of two prepended prefix modules respectively. The feature vector we use for the cross entropy loss is the concatenation of two sentence representations u and v, as well as their element-wise absolute differences $|u - v|$ following (Reimers and Gurevych, 2019). Stage-1 tuning is illustrated in Figure 2a.

Stage 2: joint tuning. After obtaining a pair of well-trained prefix modules from stage 1, we do contrastive learning in stage-2 tuning. The following objective is optimized:

$$\min_{\theta_1,\theta_2,\phi} \mathcal{L}_{\text{cl}}(f(X;\theta_1,\phi), f(X;\theta_2,\phi)) \quad (5)$$

Our stage-2 tuning is illustrated in Figure 2b. Previous works (Yan et al., 2021; Li et al., 2020) have shown that adding cross entropy loss on NLI data as an auxiliary loss during unsupervised fine-tuning can enhance the quality of the learned sentence embeddings. We add this auxiliary loss during stage 2 under our semi-supervised setting, and find it further improves the performance. More details about the auxiliary loss are in Appendix E.

3.2 Why two-stage tuning works?

In this section, we explain why the proposed two-stage tuning strategy works. From experimental results, we find the performance of contrastive learning is sensitive to the number of training steps in stage 1, and there is an optimal value for this hyperparameter. This phenomenon is consistent for both BERT and RoBERTa. To better understand this phenomenon, we introduce a new concept here: *module-compatible state*. When we say two modules are in the module-compatible state, we mean these two modules can be tuned together while obtaining satisfactory results after training. Given a set of training hyperparameters (e.g., learning rate and batch size) and a stage-2 training objective, the extent of module compatibility is closely related to the number of stage-1 training steps. We find there are two countering metrics that determine the optimal stage-1 steps: *representation divergence* (δ) and *weight convergence* (κ).

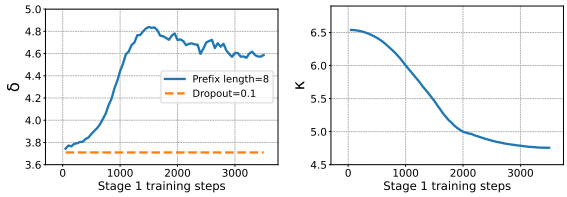
Representation divergence means the difference of embeddings that are generated from a positive pair. Since the motivation of this work is to design a stronger data augmentation strategy than dropout (Gao et al., 2021b), a larger representation divergence is desired. To measure this metric quantitatively, we use the expected distance between representations generated from positive pairs:

$$\delta \triangleq \mathop{\mathbb{E}}_{(x,x^+)\sim p_{\text{pos}}} ||f(x;\theta_1,\phi) - f(x^+;\theta_2,\phi)||_2 \quad (6)$$

We plot in Figure 3a the trend of δ as stage 1 proceeds. As a reference, we also plot the representation divergence when dropout is used for data augmentation (Gao et al., 2021b). We can observe that δ quickly reaches the highest value at around 1,500 steps, and then its value slowly goes down. We performed multiple runs with different random seeds, and found the figures all suggest that the optimal number of steps would be around 1,500. However, though we hope the representations of two positives could be as different as possible due to the InfoMin Principle (Tian et al., 2020), we also found that in practice sometimes it would be beneficial to prolong the first stage with slightly more than 1,500 steps. For example, the optimal number of stage-1 training steps for our semi-supervised model is 2,000. To explain this discrepancy, we now turn to look at the second metric, weight convergence.

The prefix modules are initialized randomly, therefore it is unsurprising to find that the parameters of prefix modules are significantly different from those of the pre-trained language model. Specifically, both prefix and language model's key and value matrices have zero means, but those matrices of language model have a much larger variance than those of prefix.[4] We quantify this difference by measuring the weight convergence κ – the $l_{2,1}$-norm difference between the key-value matrices of prefix and those of language model. The weight convergence between m-th layer's key

[4] Tao et al. (2022) made a similar observation about the distribution of weights (i.e., zero mean and large variance) in GPT-2 in Figure 4 of their paper.



(a) Representation divergence. (b) Weight convergence.

Figure 3: The visualization of the relationship between two metrics and stage-1 training steps from the $\text{BERT}_{\texttt{base}}$ with batch size 128 and learning rate 0.001.

matrix of language model $K^m \in \mathbb{R}^{L\times d}$ and that of prefix $P_k^m \in \mathbb{R}^{l\times d}$ can be defined as follows:

$$\kappa_k^m \triangleq \frac{1}{L}||K^m||_{2,1} - \frac{1}{l}||P_k^m||_{2,1} \tag{7}$$

where $||\cdot||_{2,1}$ represents the $l_{2,1}$-norm[5]. The weight convergence between the value matrices κ_v^m is defined similarly. In practice, the overall κ is defined as the averaged score over the weight convergence for both key and value matrices on all M self-attention layers:

$$\kappa \triangleq \frac{1}{2M}\sum_{m=1}^{M} \kappa_k^m + \kappa_v^m \tag{8}$$

From Figure 3b, we find the difference is monotonically decreasing, which indicates that the distribution gap reduces as stage-1 tuning continues. Previous work (Glorot and Bengio, 2010) has identified that such a parameter distribution gap can block the gradient descent based training, and we show how such distribution gap will affect the self-attention layer in details in Appendix B. Overall, the lower the κ, the easier it is for the model to perform stage-2 tuning. This explains why it could be worthwhile to start stage-2 tuning slightly after we have reached the peak for representation divergence – though we have missed the peak, the prolonged stage-1 tuning that leads to an even lower κ can potentially make the stage-2 tuning easier.

From the above analysis, we can see the module-compatible state is determined by both δ and κ. A larger δ will better explore the potential of the contrastive learning framework, while a smaller κ will make the stage-2 tuning more stable. An overall good performance is obtained when a module-compatible state is reached. More details about the optimal stage-1 step are in Appendix E.

[5]For a matrix $A \in \mathbb{R}^{r\times c}$, its $l_{2,1}$-norm is defined as $||A||_{2,1} = \sum_{i=1}^{r}\sqrt{\sum_{j=1}^{c}(A_{ij})^2}$

4 Experiments

Following previous works, we mainly conduct experiments on Semantic Textual Similarity (STS) tasks. The results demonstrate the capability of our method on producing high-quality universal sentence representations.

4.1 Experimental setup

We conduct experiments with both $\text{BERT}_{\texttt{base}}$ and $\text{RoBERTa}_{\texttt{base}}$, but only present $\text{BERT}_{\texttt{base}}$ results in this section. The results of $\text{RoBERTa}_{\texttt{base}}$ can be found in Appendix D. We consider three different settings: unsupervised, semi-supervised, and supervised settings. The unsupervised category contains both contrastive learning based and non-contrastive learning based methods that are trained with unlabeled examples. The semi-supervised setting allows the usage of both labeled and unlabeled data, but labeled data is only used for cross entropy loss. Our supervised setting only considers the supervised contrastive learning methods where the label information from NLI dataset is used for constructing positive and hard negative instances.

Datasets Following previous works, we use two training datasets for the abovementioned settings: one is the unlabeled Wikipedia dataset (Wiki1M) (Gao et al., 2021b) which contains 10^6 sentences sampled from Wikipedia; the other is the labeled NLI dataset combining SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018). It comprises 275,600 sentence triplets with the format (anchor, entailment, contradiction).

We evaluate our method on seven standard STS datasets: STS tasks 2012-2016 (Agirre et al., 2012, 2013, 2014, 2015, 2016), STS-Benchmark (Cer et al., 2017), and SICK-Relateness (Marelli et al., 2014). The sentence pairs in each dataset are scored from 0 to 5 to indicate the semantic similarity. Following previous works (Gao et al., 2021b; Jiang et al., 2022), we report the Spearman's rank correlation coefficients between the cosine similarity of the learned sentence representations and the golden similarity scores on each dataset.

Baselines We first consider two common methods for obtaining sentence embeddings from BERT: using `[CLS]` vector and mean pooling. **BERT-flow** (Li et al., 2020) is an unsupervised post-processing method that transforms the original BERT sentence embedding distribution to a smooth and isotropic Gaussian distribution. **Con-**

Method	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
Unsupervised methods								
BERT$^{\diamondsuit}$ ([CLS])	20.16	30.01	20.09	36.88	38.08	16.50	42.63	29.19
BERT$^{\diamondsuit}$ (Mean Pooling)	38.78	57.98	57.98	63.15	61.06	46.35	58.40	54.81
BERT-flow$^{\heartsuit}$	58.40	67.10	60.85	75.16	71.22	68.66	64.47	66.55
ConSERT$^{\clubsuit}$	66.07	77.40	70.32	80.91	77.29	75.78	66.91	73.53
SimCSE	68.40	82.41	74.38	80.91	78.56	76.85	72.23	76.25
PromptBERT	71.56	84.58	76.98	84.47	80.60	81.60	69.87	78.54
DiffCSE	72.28	84.43	76.47	83.90	80.54	80.59	71.23	78.49
Semi-supervised methods								
SBERT$^{\heartsuit}$	70.97	76.53	73.19	79.09	74.30	77.03	72.91	74.89
SBERT-flow$^{\heartsuit}$	69.78	77.27	74.35	82.01	77.46	79.12	76.21	76.60
ConSERT$_{\text{joint}}$$^{\clubsuit}$	72.60	82.40	77.64	82.82	78.37	80.03	**78.20**	78.87
DiffAug (*ours*)	**75.28**$_{\pm.0}$	85.20$_{\pm.2}$	78.57$_{\pm.0}$	85.20$_{\pm.2}$	**81.01**$_{\pm.3}$	82.42$_{\pm.3}$	73.24$_{\pm.5}$	80.13$_{\pm.1}$
w/ aux loss	73.46$_{\pm.5}$	**85.61**$_{\pm.2}$	**78.93**$_{\pm.3}$	**85.82**$_{\pm.1}$	80.74$_{\pm.2}$	**82.70**$_{\pm.2}$	77.00$_{\pm.5}$	**80.61**$_{\pm.2}$
Supervised contrastive learing methods								
SimCSE	75.30	84.67	80.19	85.40	80.82	84.25	80.39	81.57
PromptBERT	75.48	**85.59**	80.57	85.99	81.08	**84.56**	**80.52**	81.97
DiffAug (*ours*)	**76.92**$_{\pm.7}$	85.17$_{\pm.2}$	**80.81**$_{\pm.4}$	**86.91**$_{\pm.5}$	**82.52**$_{\pm.1}$	84.32$_{\pm.5}$	80.27$_{\pm.2}$	**82.42**$_{\pm.0}$

Table 1: Sentence embedding performance on the seven standard STS tasks (Spearman's correlation). The highest results under semi-supervised and supervised settings are highlighted. $^{\diamondsuit}$: results are from (Reimers and Gurevych, 2019); $^{\heartsuit}$: results are re-evaluated by (Gao et al., 2021b); $^{\clubsuit}$: results are re-evaluated by us using the released code [6]. More details about the re-evaluation are in Appendix C. Other baseline results are from their original papers. All our results in this table are averaged over three different runs (with standard deviation).

SERT (Yan et al., 2021) is an unsupervised contrastive learning method with several data augmentation techniques for sentences. Its unsupervised results can be further improved by incorporating NLI supervision. **SimCSE** (Gao et al., 2021b) is a contrastive learning framework that uses dropout for data augmentation. Its supervised approach regards the entailment sentence pairs from the NLI dataset as positives, and the contradiction pairs as hard negatives. **PromptBERT** (Jiang et al., 2022) improves SimCSE by applying effective hard prompts on input sentences. **DiffCSE** (Chuang et al., 2022) combines the contrastive learning objective with a difference prediction objective which is inspired by equivariant contrastive learning (Dangovski et al., 2021). **SBERT** (Reimers and Gurevych, 2019) uses BERT with a siamese structure to generate sentence embeddings after learning on NLI data with cross entropy loss. This method can be combined with the post-processing methods, e.g., BERT-flow (Li et al., 2020), to produce better results under the semi-supervised setting.

4.2 Main results

We compare our model with previous sentence embedding methods on the standard seven STS datasets in Table 1. Since our method requires a supervised prefix-tuning stage before the contrastive learning, we only report our results under the semi-supervised and supervised settings. For semi-supervised setting, we also report the results of our method with the auxiliary loss that is mentioned in Section 3.1. From Table 1, we make the following observations:

- The quality of sentence embeddings directly obtained from BERT$_{\texttt{base}}$ (both `[CLS]` vector and mean pooling) without further fine-tuning is poor. This phenomenon and the reasons behind it have been studied extensively (Reimers and Gurevych, 2019; Li et al., 2020; Su et al., 2021).
- In general, supervised and semi-supervised models are better than unsupervised models, which indicates that the label information in NLI data is beneficial to learning good sentence embeddings.
- Our method outperforms baselines in both semi-supervised and supervised settings on average. Since our method strictly follows the contrastive learning framework proposed by Gao et al. (2021b) except for the data augmentation module, our improved results confirm that the proposed data augmentation method is effective for contrastive learning.

[6] https://github.com/yym6472/ConSERT

4.3 Training with less labeled data

In this section, we show our method is more label-efficient than the state-of-the-art contrastive learning methods with $\text{BERT}_{\texttt{base}}$. The results are presented in Table 2. We find not all the previous contrastive learning methods discuss the semi-supervised setting. However, Yan et al. (2021) proposes several ways of adding supervised signal into their unsupervised contrastive learning, and these methods are also applicable to other unsupervised approaches. Therefore, we re-evaluate previous methods under the semi-supervised setting following (Yan et al., 2021), and produce stronger baselines. The details about the re-implementation can be found in Appendix C. We report the results of our semi-supervised method with auxiliary loss in stage 2. For previous supervised contrastive learning methods that do not discuss the low labeled data condition, we re-evaluate their methods with less labeled data.

To construct the low labeled data settings, we sub-sample 1% (∼2,756) or 10% (∼27,560) labeled examples from the full NLI dataset. For each size, we sample 5 sub-datasets and take the average over 3 different runs. Thus, we report the averaged results over 15 models under each low-data setting. Table 2 shows our approach significantly outperforms baselines under all settings. We also notice that the performance gap increases as the number of labeled examples decreases.

We suspect that the advantages of our proposed method in low labeled data situations come from the prefix-tuning in stage 1. Since the baseline methods all fine-tune the whole parameters of the language model, it is easy to overfit when the labeled data is scarce. However, for our method, the amount of added prefix parameters is much smaller (only 0.3% of the parameters in $\text{BERT}_{\texttt{base}}$). Therefore, the overfitting problem can be alleviated.

Method	Fraction of labeled data			
	0%	1%	10%	100%
Semi-supervised setting				
$\text{SimCSE}_{\text{joint}}$	76.25$^{\heartsuit}$	$76.45_{\pm.6}$	$77.46_{\pm.6}$	$77.73_{\pm.6}$
$\text{PromptBERT}_{\text{joint}}$	78.54$^{\spadesuit}$	$79.26_{\pm.3}$	$79.65_{\pm.3}$	$79.81_{\pm.1}$
DiffAug (*ours*)	-	$\mathbf{80.42}_{\pm.1}$	$\mathbf{80.58}_{\pm.1}$	$\mathbf{80.61}_{\pm.2}$
Supervised setting				
SimCSE	-	$75.26_{\pm.4}$	$79.04_{\pm.2}$	$81.58_{\pm.1}$
PromptBERT	-	$78.33_{\pm.2}$	$80.01_{\pm.1}$	$81.88_{\pm.1}$
DiffAug (*ours*)	-	$\mathbf{80.18}_{\pm.6}$	$\mathbf{81.73}_{\pm.2}$	$\mathbf{82.42}_{\pm.0}$

Table 2: Averaged sentence representation performance on the standard seven STS tasks with different sizes of labeled data. $^{\heartsuit}$ and $^{\spadesuit}$ are the results of unsupervised SimCSE and PromptBERT respectively. We report these two results from their original papers. All other baseline results are re-evaluated by us.

4.4 Ablation study

In this section, we compare several variants of our proposed prefix-based method, and investigate the impact of different parameter-efficient methods for data augmentation and prefix length. More ablation studies (auxiliary loss and stage-1 training steps) are provided in Appendix E. All results are based on the STS-B development set.

Variants of applying prefix. In this work, we propose to use prefix-tuning (Li and Liang, 2021) for data augmentation. Now we investigate the performance of several variants of the proposed method. The results are presented in Table 3. The amount of added parameters relative to that in $\text{BERT}_{\texttt{base}}$ is also reported. We select two unsupervised data augmentation methods, i.e., synonym replacement and dropout, as baselines since they are simple yet effective for sentence representation learning.

Our method trains two *different* prefix modules in stage 1 and *jointly* tunes both prefixes and language model in stage 2. Here we first consider two variants of applying prefixes: using two identical prefixes (same) and fixing the prefixes during stage 2 (fix). The performance gap between the first variant and our method confirms that contrastive learning performs better when the positives are meaningfully different. The performance drop caused by the second variant validates the necessity of tuning prefix and language model together in stage 2.

To investigate the importance of fine-tuning language model with prefix in stage 2, we consider another prefix variant, i.e., prefix-tuning, which means that only prefix modules will be tuned for both stage 1 and stage 2. The performance gap between our proposed method and the "prefix-tuning" method indicates that optimizing the language model is necessary for obtaining good results on sentence representation learning tasks.

Different parameter-efficient methods as data augmentation module. We also consider other parameter-efficient (PE) methods for data augmentation. Previous work (He et al., 2021) has unified

Method	# added params (rel. $\text{BERT}_{\texttt{base}}$)	STS-B dev.
Unsupervised methods		
Synonym replacement$^{\heartsuit}$	0	77.4
Dropout$^{\heartsuit}$	0	82.5
Prefix variants ($l = 8$)		
Prefix (same)	0.3%	85.7
Prefix (fix)	0.3%	85.4
Prefix (different, *ours*)	0.3%	**86.2**
Prefix-tuning (different)	0.3%	84.8
Other parameter-efficient methods		
Adapter (different, $r = 4$)	0.3%	85.6
Adapter (different, $r = 64$)	4.8%	85.1
LoRA (different, $r = 4$)	0.3%	85.4
LoRA (different, $r = 64$)	4.8%	86.1

Table 3: Comparison between different methods for data augmentation under the semi-supervised setting. l and r are the prefix length and bottleneck dimension respectively. $^{\heartsuit}$: results are from Gao et al. (2021b).

current PE methods under one framework. Hence, it is theoretically possible to transfer the success of our method with prefix-tuning to other PE methods. In this section, we additionally tried two most common and effective methods, i.e., Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021) with different bottleneck dimensions. The description of these two methods can be found in Appendix F.

As shown in Table 3, with a small amount of additional parameters (0.3%), data augmentation with distinct prefix modules obtains the highest score on the STS-B development set. Such a phenomena of prefix's effectiveness with small budget of additional parameters has also been observed by both He et al. (2021) and Li and Liang (2021) on textual generation tasks. We believe this is because prefix directly modifies each attention head on the multi-head attention layers, which makes it more expressive than other methods when parameter budget is limited.

We then study whether this conclusion changes if more tunable parameters are added. Table 3 shows that the performance of Adapter drops after having more tunable parameters ($0.3\% \rightarrow 4.8\%$), but LoRA performs better (STS-B dev. score improves by 0.7 points) when the bottleneck dimension increases from 4 to 64, though it is still slightly lower than our proposed prefix-based method.

From the above results, we conclude that applying prefixes for data augmentation is the most effective and space-efficient method for sentence representation learning tasks.

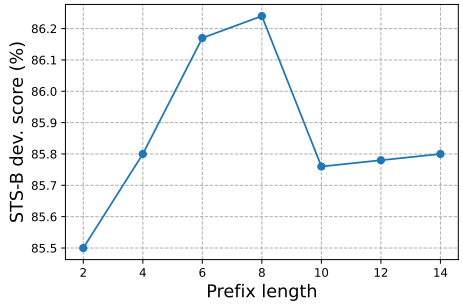


Figure 4: The relationship between prefix length and STS-B development set performance under the semi-supervised setting using $\text{BERT}_{\texttt{base}}$.

Prefix length. Prefix length is a critical hyperparameter in our experiments. The longer the prefix, the more tunable parameters, therefore the more expressive the data augmentation module is. However, it does not mean a longer prefix will definitely lead to a better performance. According to the InfoMin principle (Tian et al., 2020), a longer prefix may bring unnecessary noise, thus hurting the performance of contrastive learning. Figure 4 shows that the optimal prefix length for $\text{BERT}_{\texttt{base}}$ under the semi-supervised setting is around 8.

5 Related work

Learning sentence embeddings as a fundamental NLP problem has been extensively studied. Many works (Reimers and Gurevych, 2019; Li et al., 2020; Yan et al., 2021; Gao et al., 2021b; Jiang et al., 2022) achieve good results based on pre-trained language models. Among these works, contrastive learning methods achieve the state-of-the-art results. In this work, we improve the data augmentation module of the contrastive learning framework by a prompting method.

Contrastive sentence representation learning. Contrastive learning aims to learn effective representations by pulling together semantically close neighbors and pushing apart non-neighbors. A crucial problem in contrastive learning is how to generate positive instances. The commonly-used data augmentation methods in NLP include word deletion, reordering, and substitution (Xie et al., 2020; Yan et al., 2021). However, data augmentation in NLP is inherently difficult because of the discrete nature of language. (Gao et al., 2021b) shows that data augmentation via dropout is consistently better than previous discrete transformations.

Prompting. Prompting means querying a pre-trained language model by adding natural language tokens into the input sentences (Brown et al., 2020). This method alleviates the discrepancy between a language model's pre-training stage and fine-tuning stage, and is shown to be useful when the labeled data for downstream tasks is scarce (Gao et al., 2021a). Although adding natural language tokens (i.e., hard prompt) is effective for certain tasks, it takes efforts to design good prompts. Motivated by this problem, soft prompt methods (Qin and Eisner, 2021; Zhong et al., 2021) are proposed. Unlike hard prompt, soft prompt tokens can be fine-tuned continuously. Therefore, it is more expressive. Some recent works further improve the soft prompt method by adding deep prompt modules on language models (Houlsby et al., 2019; Li and Liang, 2021; Liu et al., 2022), and they are called parameter-efficient methods. With more tunable parameters, the parameter-efficient methods can handle complex natural language tasks, e.g., language understanding (Houlsby et al., 2019), text generation (Li and Liang, 2021), and structured prediction (Liu et al., 2022). Such methods usually assume the language model is fixed during fine-tuning, and only the newly-added prompting modules can be tuned.

6 Conclusion

In this paper, we propose a differentiable data augmentation method for contrastive sentence representation learning. Unlike previous discrete transformations for sentences (e.g., token shuffling and synonyms replacement) and continuous methods (e.g., dropout (Gao et al., 2021b)), the proposed method learns how to do data augmentation implicitly based on supervised signals from the natural language inference (NLI) tasks. In this way, our method produces *hard positives* with meaningful differences for contrastive learning. We demonstrate the effectiveness of our method on several semantic textual similarity tasks, and our method achieves new state-of-the-art performance in both semi-supervised and supervised settings on average. We also conduct experiments where only limited labeled data is available. The results demonstrate the proposed method is robust when is labeled data is scarce.

To the best of our knowledge, our method is the first work that provides a successful solution for combining the conventional pre-trained language model fine-tuning with parameter-efficient methods via a two-stage tuning process. The approach of combining the two yields better results than using any of them alone on the sentence representation learning tasks. One potential future research direction is to generalize this method to other NLP research areas, e.g., domain adaptation and low-resource tasks.

Limitations

Since our method incorporates a learnable module for data augmentation, it requires additional training time and GPU memory compared with previous contrastive learning methods. The demand for more computation resources is a clear limitation of our work. We compare the training overheads with two previous contrastive learning methods in Table 4. We have mentioned in the previous section that our stage-1 tuning is necessary, and it is not a trivial task to reach a module-compatible state. Therefore, reducing computation resources during training for our method is difficult. However, given the improvements that are brought by our methods on sentence embedding learning tasks, the additional computation cost is worthy.

Method	Training time	GPU memory usage
SimCSE	36 min.	10 GB
PromptBERT	48 min.	13 GB
DiffAug (*ours*)	65 min.	15 GB

Table 4: Comparison of training time and GPU memory usage of our method and previous contrastive learning methods. We use same batch size (128) and same number of epochs (3) on NLI dataset with $\text{BERT}_{\texttt{base}}$ for all three methods for a fair comparison. All results in this table are obtained from a single Nvidia Quadro RTX8000 GPU.

Acknowledgements

We would like to thank the anonymous reviewers and (senior) area chairs for their insightful comments and help with this work. We would also like to thank members of StatNLP research group for helpful discussions. This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISGRP-2019-012). Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not reflect the views of National Research Foundation, Singapore and AI Singapore.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. In *Proceedings of SemEval*.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. SemEval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of SemEval*.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of SemEval*.

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of *SEM*.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Proceedings of *SEM*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of EMNLP*.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris 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 *Proceedings of NeurIPS*.

Daniel Cer, Mona Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. SemEval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of SemEval*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020a. A simple framework for contrastive learning of visual representations. In *Proceedings of ICML*.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. 2020b. Big self-supervised models are strong semi-supervised learners. In *Proceedings of NeurIPS*.

Yung-Sung Chuang, Rumen Dangovski, Hongyin Luo, Yang Zhang, Shiyu Chang, Marin Soljacic, Shang-Wen Li, Scott Yih, Yoon Kim, and James Glass. 2022. DiffCSE: Difference-based contrastive learning for sentence embeddings. In *Proceedings of ACL*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of EMNLP*.

Rumen Dangovski, Li Jing, Charlotte Loh, Seungwook Han, Akash Srivastava, Brian Cheung, Pulkit Agrawal, and Marin Soljačić. 2021. Equivariant contrastive learning. *arXiv preprint arXiv:2111.00899*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL*.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021a. Making pre-trained language models better few-shot learners. In *Proceedings of ACL-IJCNLP*.

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

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*.

Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *Proceedings of CVPR*.

Junxian He, Chunting Zhou, Xuezhe Ma, Taylor Berg-Kirkpatrick, and Graham Neubig. 2021. Towards a unified view of parameter-efficient transfer learning. In *Proceedings of ICLR*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of ICCV*.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for nlp. In *Proceedings of ICML*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *Proceedings of ICLR*.

Ting Jiang, Shaohan Huang, Zihan Zhang, Deqing Wang, Fuzhen Zhuang, Furu Wei, Haizhen Huang, Liangjie Zhang, and Qi Zhang. 2022. Promptbert: Improving bert sentence embeddings with prompts. *arXiv preprint arXiv:2201.04337*.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of EMNLP*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of ACL-IJCNLP*.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of ACL*.

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*.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A SICK cure for the evaluation of compositional distributional semantic models. In *Proceedings of LREC*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NeurIPS*.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of EMNLP*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying lms with mixtures of soft prompts. In *Proceedings of NAACL*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of EMNLP-IJCNLP*.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *arXiv preprint arXiv:2103.15316*.

Chaofan Tao, Lu Hou, Wei Zhang, Lifeng Shang, Xin Jiang, Qun Liu, Ping Luo, and Ngai Wong. 2022. Compression of generative pre-trained language models via quantization. In *Proceedings of ACL*.

Yonglong Tian, Chen Sun, Ben Poole, Dilip Krishnan, Cordelia Schmid, and Phillip Isola. 2020. What makes for good views for contrastive learning? In *Proceedings of NeurIPS*.

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 *Proceedings of NeurIPS*.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of EMNLP*.

Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. 2020. Unsupervised data augmentation for consistency training. In *Proceedings of NeurIPS*.

Yuanmeng Yan, Rumei Li, Sirui Wang, Fuzheng Zhang, Wei Wu, and Weiran Xu. 2021. Consert: A contrastive framework for self-supervised sentence representation transfer. In *Proceedings of ACL*.

Zexuan Zhong, Dan Friedman, and Danqi Chen. 2021. Factual probing is [MASK]: Learning vs. learning to recall. In *Proceedings of NAACL*.

A Implementation details

We implement our method based on Hugging Face's Transformer (Wolf et al., 2020) and PyTorch libraries. The backbone of our code is from (Gao et al., 2021b) with some prompt implementation details from (Jiang et al., 2022). For both semi-supervised and supervised settings, we evaluate the model on the STS-B development set every 50 training steps, and only save the checkpoints with the best performance on development set for the final evaluation on test set.

For all our settings, we apply hard prompts on the input sentences and obtain the sentence representations from the hidden embeddings of `[MASK]` tokens. The hard prompt templates we use in our method are from (Jiang et al., 2022). We also apply the template denoising for contrastive learning since this technique brings significant improvements from only adding hard prompts according to (Jiang et al., 2022).

The core component of our method is the use of two different prefix modules for data augmentation. During training, we use them to construct hard positive pairs with meaningful differences for contrastive learning. During inference, we prepend both prefix modules to the languge model to obtain the sentence representations. In practice, we follow Li and Liang (2021) and reparametrize the key and value matrices P_k, P_v in the prefix by a smaller matrix P' composed with a large multi-layer perceptron to make the training stable. Our training hyperparameters under different settings for $\text{BERT}_{\texttt{base}}$ are in Table 5.

		Semi-supervised w/o aux	Semi-supervised w/ aux	Supervised w/o aux
Stage 1	learning rate	1e-3	1e-3	1e-3
	batch size	128	128	128
	training steps	2000	1500	1250
Stage 2	learning rate	1e-5	1e-5	5e-5
	batch size	256	256	128
	epochs	1	1	3

Table 5: Hyperparameters for each setting.

B Analysis of weight convergence

In the previous section, we mentioned that the module-compatible state in our case is closely related to the stage-1 training steps, and the optimal number of training steps is determined by two countering metrics, i.e., representation divergence and weight convergence. Previous works (Chen et al., 2020a; Tian et al., 2020) have demonstrated the importance of representation divergence in contrastive learning. In this section, we show the necessity of weight convergence in our stage-2 training. Our method is based on the prefix-tuning (Li and Liang, 2021) that affects the original language model by adding new key-value pairs on each self-attention layer. The added key and value matrices influence the self-attention computation differently. Specifically, key matrix (K) is used for attention matrix calculation, i.e., $A = \text{softmax}(\frac{Q^T K}{\sqrt{d}})$ while value matrix (V) is used for weighted averaged output generation, i.e., $H = AV$, we analyze the impact of weight convergence on key and value separately.

B.1 The impact of weight convergence on attention matrix calculation

The computation of attention matrix includes a softmax operation $S(\cdot) : \mathbb{R}^N \to \mathbb{R}^N$ that can be defined as:

$$S(\mathbf{a}) : \begin{pmatrix} a_1 \\ \vdots \\ a_N \end{pmatrix} \to \begin{pmatrix} s_1 \\ \vdots \\ s_N \end{pmatrix} \tag{9}$$

where $s_i = \frac{\exp(a_i)}{\sum_{j=1}^{N} \exp(a_j)}$, and N is the length of input and output vectors. Therefore, the derivative of the softmax function $S(\cdot)$, i.e., the Jacobian matrix, can be derived as:

$$J_{\text{softmax}} = \begin{pmatrix} \frac{\partial s_1}{\partial a_1} & \frac{\partial s_1}{\partial a_2} & \cdots & \frac{\partial s_1}{\partial a_N} \\ \frac{\partial s_2}{\partial a_1} & \frac{\partial s_2}{\partial a_2} & \cdots & \frac{\partial s_2}{\partial a_N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial s_N}{\partial a_1} & \frac{\partial s_N}{\partial a_2} & \cdots & \frac{\partial s_N}{\partial a_N} \end{pmatrix} \tag{10}$$

where

$$\frac{\partial s_i}{\partial a_j} = \begin{cases} s_i \cdot (1 - s_j), \text{if } i = j \\ -s_i \cdot s_j, \text{if } i \neq j \end{cases} \tag{11}$$

From the above derivation, either a too large or a too small s_i will make the derivative approach zero. Notice that in Section 3.2, we mentioned the averaged norm of the key vectors from randomly initialized prefix module is much smaller than those in the original language model. Therefore, if we tune both prefix and language model jointly without stage-1 training, the gradient back propagation of the self-attention layer will be blocked due to the vanishing gradients.

B.2 The impact of weight convergence on weighted averaged output generation

In the self-attention layer of the Transformer, the embeddings of each token are generated via the weighted average of the value vectors. Previous works (Glorot and Bengio, 2010; He et al., 2015) have shown the importance of keeping the same the variance of input of each layer in a deep neural network during forward propagation. Specifically, if we write the input matrix of layer i as Z^i, to ensure the information flow, we hope

$$\forall (i, i'),\ \text{Var}[Z^i] = \text{Var}[Z^{i'}] \tag{12}$$

For the Transformer's i-th self-attention layer, we use X^i, $H^i \in \mathbb{R}^{L \times d}$ to represent the input and output respectively, where L is the input length, and d is the hidden dimension. The importance of weight convergence in attention matrix has been discussed in the last section. Now, for simplicity, we write the attention function as $H^i = A^i[P_v^i;\ V^i]$, where $A^i \in \mathbb{R}^{L \times (L+l)}$ is the attention matrix after softmax, $P_v^i \in \mathbb{R}^{l \times d}$ is the added value matrix from the prefix module, and $V^i \in \mathbb{R}^{L \times d}$[7] is the value matrix of the original language model.

We have shown in Section 3.2 the norm of P_v^i is significantly less than that of V^i at the beginning. Therefore, if the training steps of stage 1 is not long enough, the difference between the variance of P_v^i and V^i could be so large that makes the condition in the Equation 12 invalid, which will cause the stage-2 joint training unstable.

[7] $V^i = X^i W_v^i$, where $W_v^i \in \mathbb{R}^{d \times d}$ is the projection matrix of value.

Method	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Avg.
ConSERT$_{\text{joint}}$	**72.60**$_{\pm.4}$	82.40$_{\pm.8}$	**77.64**$_{\pm.5}$	82.82$_{\pm.1}$	**78.37**$_{\pm.1}$	80.03$_{\pm.4}$	**78.20**$_{\pm.1}$	**78.87**$_{\pm.3}$
ConSERT$_{\text{sup-unsup}}$	70.11$_{\pm.2}$	82.07$_{\pm.5}$	74.97$_{\pm.4}$	**83.21**$_{\pm.2}$	76.91$_{\pm.3}$	78.89$_{\pm.3}$	76.92$_{\pm.5}$	77.58$_{\pm.3}$
ConSERT$_{\text{sup-joint}}$	71.96$_{\pm.3}$	**82.70**$_{\pm.4}$	75.85$_{\pm1.1}$	82.70$_{\pm.5}$	77.16$_{\pm.8}$	**80.78**$_{\pm.7}$	75.12$_{\pm.4}$	78.04$_{\pm.5}$
SimCSE$_{\text{joint}}$	70.73$_{\pm1.8}$	**83.22**$_{\pm.3}$	**75.75**$_{\pm.5}$	**82.93**$_{\pm.6}$	**77.90**$_{\pm1.0}$	**78.69**$_{\pm.3}$	**74.86**$_{\pm.5}$	**77.73**$_{\pm.6}$
SimCSE$_{\text{sup-unsup}}$	69.02$_{\pm2.4}$	82.21$_{\pm1.3}$	74.51$_{\pm.9}$	82.27$_{\pm.6}$	77.20$_{\pm.6}$	78.55$_{\pm1.3}$	74.06$_{\pm1.8}$	76.83$_{\pm1.3}$
SimCSE$_{\text{sup-joint}}$	**70.80**$_{\pm1.2}$	81.27$_{\pm.2}$	74.68$_{\pm.5}$	81.83$_{\pm1.0}$	77.58$_{\pm1.2}$	78.55$_{\pm1.5}$	73.58$_{\pm1.1}$	76.90$_{\pm.8}$
PromptBERT$_{\text{joint}}$	**71.71**$_{\pm.3}$	**84.99**$_{\pm.1}$	**77.25**$_{\pm.2}$	**85.01**$_{\pm.1}$	**80.66**$_{\pm.1}$	**81.86**$_{\pm.1}$	**75.50**$_{\pm.3}$	**79.81**$_{\pm.1}$
PromptBERT$_{\text{sup-unsup}}$	71.23$_{\pm.2}$	85.16$_{\pm.1}$	77.88$_{\pm.2}$	85.15$_{\pm.2}$	81.23$_{\pm.2}$	82.06$_{\pm.6}$	75.22$_{\pm.8}$	78.15$_{\pm.2}$
PromptBERT$_{\text{sup-joint}}$	71.34$_{\pm.7}$	81.61$_{\pm1.0}$	76.23$_{\pm.6}$	82.92$_{\pm.5}$	78.45$_{\pm.7}$	79.76$_{\pm1.0}$	74.63$_{\pm.7}$	77.85$_{\pm.7}$

Table 6: Semi-supervised baselines re-implemented by us over three different settings.

C Baseline methods

In this section, we describe how we re-evaluate the baseline models when their original published results are not directly comparable with our methods. For unsupervised ConSERT (Yan et al., 2021), we re-evaluate it on Wiki1M dataset.

For methods that do not implement with semi-supervised setting, we find that simply adding an auxiliary cross entropy loss with proper weight can boost the performance from unsupervised results. To make a comprehensive evaluation, we try the following three methods of adding supervised signals from NLI data inspired by (Yan et al., 2021):

- **Joint training (joint).** We combine the contrastive learning objective with the cross entropy loss, which can be written as

$$\mathcal{L}_{\text{joint}} = \mathcal{L}_{\text{cl}} + \alpha \cdot \mathcal{L}_{\text{ce}} \tag{13}$$

  where the contrastive loss is computed with unlabeled Wiki1M data while the cross entropy loss is computed with labeled NLI data. For this setting, we try different values of α for each method and select the best one to compare with our method.

- **Supervised training then unsupervised contrastive training (sup-unsup).** We split the whole training process into 2 stages: first train the model with cross entropy loss on NLI data, then train the model with contrastive loss on Wiki1M data.

- **Supervised training then joint training (sup-joint).** The training process is still split into 2 stages: first, the model is trained with cross entropy loss on NLI data; next, the model is trained with the joint loss $\mathcal{L}_{\text{joint}}$.

The results of each setting are shown in Table 6. We select the setting with the best performance (i.e., joint training setting) and use it to compare with our method in Table 2.

Method	Fraction of labeled data			
	0%	1%	10%	100%
Semi-supervised setting				
SimCSE$_{\text{joint}}$	76.57$^{\heartsuit}$	77.32$_{\pm.8}$	77.66$_{\pm.5}$	77.85$_{\pm.5}$
PromptRoBERTa$_{\text{joint}}$	79.15$^{\spadesuit}$	79.89$_{\pm.5}$	80.12$_{\pm.5}$	80.45$_{\pm.5}$
DiffAug (*ours*)	-	**80.56**$_{\pm.3}$	**80.71**$_{\pm.3}$	**80.93**$_{\pm.3}$
Supervised setting				
SimCSE	-	76.83$_{\pm.6}$	80.18$_{\pm.2}$	82.43$_{\pm.2}$
PromptRoBERTa	-	79.15$_{\pm.4}$	81.22$_{\pm.3}$	82.50$_{\pm.2}$
DiffAug (*ours*)	-	**79.50**$_{\pm.7}$	**81.89**$_{\pm.2}$	**82.76**$_{\pm.2}$

Table 7: Averaged sentence representation performance on STS tasks with different fractions of labeled data using RoBERTa$_{\texttt{base}}$. $^{\heartsuit}$ and $^{\spadesuit}$ are the results of unsupervised SimCSE and PromptBERT respectively. We report these two results from their original papers, and re-evaluate other baseline results.

D RoBERTa results

We compare our approach on RoBERTa$_{\texttt{base}}$ with the state-of-the-art contrastive learning methods (Gao et al., 2021b; Jiang et al., 2022) under different fractions of labeled data in Table 7. We use a similar approach to evaluate our method and baselines on RoBERTa$_{\texttt{base}}$ that we described in Section 4.3. The results show that the proposed method also outperforms the baselines on RoBERTa$_{\texttt{base}}$.

E Ablation studies

Auxiliary objective in stage 2. We now investigate the impact of cross entropy auxiliary objective in stage-2 training. When the auxiliary loss is added, the overall objective for stage 2 has a form of $\mathcal{L} = \mathcal{L}_{cl} + \alpha \cdot \mathcal{L}_{ce}$, where α is a scalar that balances two losses. Table 8 shows the impact of auxiliary loss in stage 2. We find that for

Add aux loss	α	STS-B dev.
False	-	86.2
True	1×10^{-4}	86.0
	1×10^{-3}	**86.3**
	1×10^{-2}	84.8
	1×10^{-1}	83.3

Table 8: Ablation studies of the impact of auxiliary objective during stage 2 for semi-supervised settings.

semi-supervised setting, adding auxiliary loss with $\alpha = 1 \times 10^{-3}$ works the best. We suspect NLI data contains valuable information that benefits sentence representation learning. However, for our semi-supervised method, such information gained from stage 1 can be gradually forgotten in stage 2 if no auxiliary loss is added. Hence, combining contrastive loss with an auxiliary cross entropy loss in stage 2 will benefit the overall training process.

Training steps of stage 1. Another hyperparameter we find crucial for the overall performance is the training steps of stage 1. The importance of this hyperparameter has been elaborated in Section 3.2. Here we conduct further analysis using $\text{BERT}_{\texttt{base}}$ under the semi-supervised setting by visualizing the relationship between stage-1 training steps and cross entropy loss on the training data. We observe that a lower stage-1 cross entropy loss does not always lead to a better overall performance. From Figure 5, we can see that the model reaches the highest development score when the number of training steps is around 2,000, but the training loss can still be further reduced.

As we mentioned in Section 3.1, one of the important roles that stage-1 training performs is to make sure the prefix modules are compatible with the language model during the contrastive learning in stage 2. Therefore, for our method, the compatibility between prefix and language model is more important than prefix's capability on capturing sentence relationships.

F Overview of the existing parameter-efficient methods

In this section, we introduce two existing parameter-efficient methods that are mentioned in Section 4.4, i.e., Adapter (Houlsby et al., 2019) and LoRA (Hu et al., 2021).

Adapter. This method proposes to insert modules with small amount of trainable parameters on

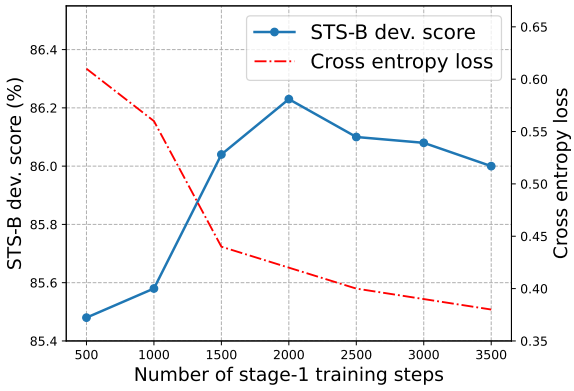


Figure 5: Number of stage-1 training steps V.S. STS-B development performance V.S. cross entropy loss on the training data.

each layer of the Transformer-based pre-trained language model. Specifically, each module has three main components: a down-projection linear layer with parameters $W_{\text{down}} \in \mathbb{R}^{d \times r}$, a up-projection linear layer with parameters $W_{\text{up}} \in \mathbb{R}^{r \times d}$, and a non-linear activation function $f(\cdot)$, where d and r are the hidden states and bottleneck dimensions respectively. The adpater module modifies the hidden state $\boldsymbol{h}$ in the following way:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + f(\boldsymbol{h} W_{\text{down}}) W_{\text{up}} \tag{14}$$

In the original implementation (Houlsby et al., 2019), the adapter modules are inserted in two places on each transformer layer, i.e., one is after the self-attention layer and the other is after the feed-forward network layer.

LoRA. Similar to Adapter, LoRA also adds small modules, called trainable rank decomposition matrices, on each layer of the pre-trained language model. Each LoRA module is composed of two tunable weight matrices $W_{\text{down}} \in \mathbb{R}^{d \times r}$ and $W_{\text{up}} \in \mathbb{R}^{r \times d}$. Unlike Adapter, LoRA module does not contain any non-linear activation functions, and the added weight matrices are parallel to the original weight matrices. Hence, it works in the following way:

$$\boldsymbol{h} \leftarrow \boldsymbol{h} + s \cdot \boldsymbol{x} W_{\text{down}} W_{\text{up}} \tag{15}$$

where $s \geq 1$ is a scaling factor and $\boldsymbol{x}$ represents the input. Another difference between LoRA and Adapter is that LoRA modules are only applied on the query and value projection matrices Q and V in the self-attention sub-layers, while the Adapter modules are attached on both the self-attention and feed-forward network sub-layers.