Contrastive Learning for Prompt-Based Few-Shot Language Learners

Yiren Jian
Dartmouth College
yiren.jian.gr@dartmouth.edu

Chongyang Gao
Northwestern University
cygao@u.northwestern.edu

Soroush Vosoughi
Dartmouth College
soroush@dartmouth.edu

Abstract

The impressive performance of GPT-3 using natural language prompts and in-context learning has inspired work on better fine-tuning of moderately-sized models under this paradigm. Following this line of work, we present a contrastive learning framework that clusters inputs from the same class for better generality of models trained with only limited examples. Specifically, we propose a supervised contrastive framework that clusters inputs from the same class under different augmented "views" and repel the ones from different classes. We create different "views" of an example by appending it with different language prompts and contextual demonstrations. Combining a contrastive loss with the standard masked language modeling (MLM) loss in prompt-based few-shot learners, the experimental results show that our method can improve over the state-of-the-art methods in a diverse set of 15 language tasks. Our framework makes minimal assumptions on the task or the base model, and can be applied to many recent methods with little modification. The code will be made available at: https://github.com/yiren-jian/LM-SupCon.

1 Introduction

The prompt-based fine-tuning method reduces the gap between pre-training and fine-tuning by forming the fine-tuning task into a masking language problem. A language prompt is a piece of text appended to the query input enabling the model to come up with better predictions (Schick and Schütze, 2021; Tam et al., 2021). For instance, by feeding a language model with *"The story is not worth reading, a truly __ one."*, the model assigns a higher probability for the blank to be filled with *"terrible"* than *"great"*. Here, *"a truly __ one."* is called the template of the prompt and *"terrible"* or *"great"* is the label word. Recently, LM-BFF (Gao et al., 2021) shows that appending demonstrations (*e.g."This is an amazing movie, a truly great one"*) to inputs can help the model to better understand the label word, leading to further improved results.

In this work, we show that Supervised Contrastive Learning (SupCon) (Khosla et al., 2020) at the feature space can be beneficial during the *fine-tuning* of prompt-based few-shot language learners, with proper data augmentation.

Data augmentation is the key component of SupCon. While there exists many augmentation techniques like Cutmix (Yun et al., 2019), Mixup (Zhang et al., 2018) in computer vision and EDA (Wei and Zou, 2019), AEDA (Karimi et al., 2021) for text, data augmentation remains challenging.

However, prompt-based few-shot learners with demonstrations actually provide us with a natural way to create multiple "views" (augmentations) of a single example, *i.e.*, for a fixed set of label words, we can sample different templates and different demonstrations to append to the input text (shown in Figure 1). This allows us to construct diverse input texts that are consistent and complete. By applying SupCon to cluster the above two example inputs with very different contents but the same label, our method is able to obtain an additional supervision at the feature space which is crucial if we are only given a few labeled examples.

The main contributions of our paper are:

- A Supervised Contrastive Learning framework for prompt-based few-shot learners.
- An effective data augmentation method using prompts for contrastive learning with prompt-based learners.

2 Related Work & Background

Few-shot Learning is often tackled by meta learning (Li and Zhang, 2021; Bansal et al., 2020; Sharaf et al., 2020; Jian et al., 2020; Jian and Gao, 2021), data augmentation (Jian et al., 2022; Jian and Torresani, 2022; Arthaud et al., 2021; Wei et al., 2021;

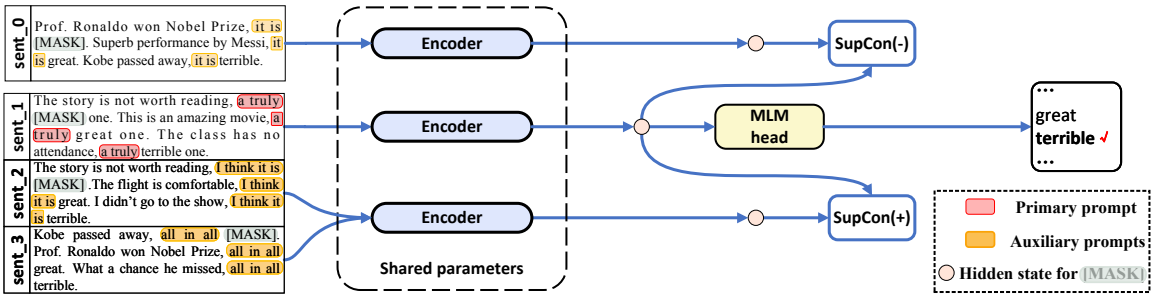

Figure 1: Overview of our proposed method. Besides the standard prompt-base MLM loss on label words "great" and "terrible", we introduce a SupCon loss on multi-views of input text. The positive pair is sentences (with sampled templates and/or demonstrations) in the same class, e.g. sent_1 and sent_3, or itself with a different template and demonstrations, e.g. sent_1 and sent_2. The negative sentence pair is input sentences (with sampled templates and/or demonstrations) in different classes, e.g. sent_1 and sent_0.

Kumar et al., 2019). Inspired by the in-context learning of GPT-3, prompt-based fine-tuning (Gao et al., 2021; Tam et al., 2021; Schick and Schütze, 2021) recently becomes dominant in NLP. Basu et al. (2021) applies contrastive learning in their few-shot semi-supervised intent classification, by using EDA (Wei and Zou, 2019) as augmentation method. Different from Basu et al. (2021), our method applies to prompt-based fine-tuning, and we show in experiments that our proposed augmentation outperforms EDA.

Supervised Contrastive Loss. SupCon is a special form of contrastive learning (Chen et al., 2020a,b; Tian et al., 2020a,b,c; Liu et al., 2021; Xiong et al., 2020) that clusters two augmented batches at the class level in the feature space. Let $\tilde{x}_{2k-1}, \tilde{x}_{2k}$ be two augmented views of an input batch x_k; and z_{2k}, z_{2k-1} to be the features of $\tilde{x}_{2k-1}, \tilde{x}_{2k}$. Then SupCon loss can be computed as

$$\mathcal{L}_{\text{SupCon}} = \text{SupCon}(z_{2k-1}, z_{2k}, y_k) \quad (1)$$

where y_k is the label for batch x_k. The details of SupCon can be found in Khosla et al. (2020).

3 Method

Problem formulation. Following the few-shot setting in LM-BFF, we assume to have access to a pre-trained language model $\mathcal{M}$, datasets $\mathcal{D}_{\text{train}}$ and $\mathcal{D}_{\text{test}}$ with label space $\mathcal{Y}$. There are only $K = 16$ examples per class in $\mathcal{D}_{\text{train}}$.

Fine-tuning with prompts and demonstrations. Prompt-based methods treat a classification problem as a masked language modeling (MLM) problem. They take as input a sentence (sent) and a masked template (temp) (i.e., $x_{\text{prompt}} = \text{sent}, \text{temp}([\text{mask}])$), and find the best token to fill in the [mask]. This leads to a MLM loss $\mathcal{L}_{\text{MLM}} = \text{MLM}(x_{\text{prompt}}, y)$, where y is the label word corresponding to x_{prompt}. LM-BFF (Gao et al., 2021) further appends demonstrations of label words to improve the results: $x_{\text{prompt+demo}} = \text{sent}_0, \text{temp}_0([\text{mask}]), \text{sent}_i, \text{temp}_0(\text{word}_i)$, where word_i is the label word for sent_i, and sent_i is sampled from the training set. Then the classification loss becomes:

$$\mathcal{L}_{\text{MLM}} = \text{MLM}(x_{\text{prompt+demo}}, y) \quad (2)$$

More mathematical formulation can be found in LM-BFF or our Appendix B.

Language-based Supervised Contrastive Loss. For applying SupCon on multi-views of an input text, we need to first obtain two views of a text:

$$x_1 = \text{sent}_0, \text{temp}_0([\text{mask}]), \text{sent}_i, \text{temp}_0(\text{word}_i)$$
$$x_2 = \text{sent}_0, \text{temp}_j([\text{mask}]), \text{sent}_k, \text{temp}_j(\text{word}_k)$$

where x_1 is identical to $x_{\text{prompt+demo}}$ in LM-BFF. We sample a new template (temp_j), demonstration (sent_k) and the corresponding label word (word_k) to replace those in x_1, to create a second view of input x_2. With x_1 and x_2, we can compute SupCon loss by Equation 1. The total loss is then

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{SupCon}} \quad (3)$$

See our Appendix C for more mathematical details.

Computational overhead. We show the algorithm of our method in Algorithm 1. In general, our method learns from $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{MLM}} + \mathcal{L}_{\text{SupCon}}$, whereas baseline LM-BFF learns with $\mathcal{L}_{\text{MLM}}$ only.

Learning from $\mathcal{L}_{\text{SupCon}}$ requires one additional forward and backward pass (highlighted in blue in Algorithm 1), leading to an increase of computational cost by $\times 1.5$.

Algorithm 1 Our method

$Max\_Step = 1000$,
LM: Language model,
$Train\_Set$: Training set,
$Sample$: Randomly sampling function,
$Concatenate$: The function to concatenate two strings,
CE: Cross Entropy loss,
$SupCon$: Supervised Contrastive loss.
for i in Max_Step **do**
    $sent, y = Sample(Train\_Set)$
    $demo_1 = Sample(Train\_Set)$
    $demo_2 = Sample(Train\_Set)$
    $input_1 = concatenate(sent, demo_1)$
    $input_2 = concatenate(sent, demo_2)$
    ▷ Learning from MLM Loss
    $output_1 = LM(input_1)$
    $L_{MLM} = CE(output_1, y)$
    $L_{MLM}.backward()$
    $optimizer.step()$
    ▷ Learning from SupCon Loss
    $output_2 = LM(input_2)$
    $L_{SupCon} = SupCon(output_1, output_2)$
    $L_{SupCon}.backward()$
    $optimizer.step()$
end for

4 Experiments

Evaluation datasets and protocol. We evaluate our method on 15 classification tasks studied in LM-BFF and follow the same setup as them to allow fair comparisons (see Appendix A for more training details). Contrastive learning algorithms benefit from large batch training. Thus, we report baselines with the same large batch size as ours.

Our method uses a single prompt/template (primary prompt) for the prediction of each task, and a set of prompts (auxiliary prompts) for generating multi-views of inputs for contrastive learning. The primary prompts we used are shown in Appendix D. The auxiliary prompts can be either manually designed or generated by a searching algorithm. In this work, we use the top-20 generated prompts from LM-BFF's project page and we randomly sample templates in these 20 prompts to produce second views of our inputs. Unless otherwise noted, we apply *both* random templates and random demonstrations to create second views of inputs for the contrastive learning.

Task	LM-BFF	LM-BFF + ours	PET	PET + ours
SST-2 (acc)	89.2 (1.3)	**90.6** (0.1)	88.4 (1.0)	**89.9** (0.6)
Subj (acc)	88.6 (3.3)	**90.4** (1.1)	89.2 (1.5)	**90.6** (1.6)
SST-5 (acc)	47.9 (0.8)	**49.5** (1.1)	46.0 (0.9)	**48.8** (1.2)
CoLA (Matt.)	6.1 (5.3)	**10.2** (5.8)	3.5 (3.4)	**5.9** (3.3)
TREC (acc)	82.8 (3.1)	**83.3** (1.5)	77.8 (9.1)	**82.3** (4.6)
MNLI (acc)	61.0 (2.1)	**64.0** (2.0)	58.2 (1.1)	**58.9** (3.1)
MNLI-mm (acc)	62.5 (2.1)	**65.5** (2.7)	59.8 (1.2)	**61.0** (3.3)
SNLI (acc)	66.9 (2.4)	**69.9** (2.4)	63.1 (2.5)	**65.7** (3.9)
QNLI (acc)	60.7 (1.7)	**66.4** (3.5)	61.5 (3.3)	**63.5** (3.7)
QQP (acc)	62.5 (2.6)	**68.8** (3.8)	61.9 (3.5)	**65.7** (4.3)
RTE (acc)	64.3 (2.7)	**65.1** (3.5)	60.9 (4.7)	**65.1** (3.5)
MRPC (F1)	75.5 (5.2)	**78.2** (3.1)	70.6 (6.0)	**75.7** (6.1)
MR (acc)	83.3 (1.4)	**85.8** (0.6)	85.0 (0.6)	**85.2** (0.9)
MPQA (acc)	83.6 (1.8)	**84.6** (1.5)	81.3 (2.6)	**81.8** (2.4)
CR (acc)	88.9 (1.0)	**89.4** (1.0)	89.3 (1.0)	**90.5** (0.5)

Table 1: Few-shot experiments of baseline methods and ours. LM-BFF is a prompt-based method with demonstrations of label words and PET is one without demonstrations. The experimental results show the means and standard deviations from 5 different train-test splits.

4.1 Main results on 15 tasks

We use RoBERTa-base (see Appendix E for RoBERTa-large). We compare ours with LM-BFF (a method w/ demonstrations) and PET (Schick and Schütze, 2021) (a method w/o demonstration).

Table 1 shows that our SupCon loss can consistently boost the performance of baseline prompt-based fine-tuning method LM-BFF. The introduction of SupCon loss has a maximum improvement of 6.3% in QQP and an average improvement of 2.5% across 15 tasks, likely due to the more generalized representations learned by SupCon. On average, the greater improvements by our model can be seen on the more difficult tasks (see Appendix 5 for more detail).

We want to emphasize that the input for baseline LM-BFF already appends different randomly sampled demonstrations at each tuning iteration. Thus, the improvement of our method can not be attributed to the diversity of inputs when learning from $\mathcal{L}_{\text{MLM}}$ of Equation 3, but to the $\mathcal{L}_{\text{SupCon}}$.

Table 1 also shows that our method works well even for prompt-based methods without demonstrations. PET, which is a method without demonstrations, works consistently worse than LM-BFF. However, with the additional SupCon loss, the few-shot performances of PET can be increased by an average of 2.3%. And the gap between having and not having demonstrations can be largely closed

(see LM-BFF vs. PET+ours in Table 1). In some tasks, *e.g.*, SST-2, SST-5, QNLI, QQP, RTE MRPC, MR, and CR, the contribution of our SupCon loss can be even larger than the sole use of the demonstrations for label words.

4.2 SupCon vs. other losses

Task	LM-BFF	LM-BFF +Dec	LM-BFF +Dec +Lab	LM-BFF +ConCal	LM-BFF +ours
SST-2	89.2 (1.3)	90.1 (0.6)	**90.6** (0.5)	88.5 (2.0)	**90.6** (0.1)
Subj	88.6 (3.3)	87.3 (3.6)	88.4 (4.9)	83.8 (7.3)	**90.4** (1.1)
SST-5	47.9 (0.8)	47.2 (1.0)	46.5 (0.7)	47.9 (1.1)	**49.5** (1.1)
CoLA	6.1 (5.3)	9.8 (6.5)	7.2 (5.2)	6.7 (4.6)	**10.2** (5.8)
TREC	82.8 (3.1)	81.9 (3.0)	82.3 (3.0)	71.1 (7.0)	**83.3** (1.5)
MNLI	61.0 (2.1)	61.3 (2.1)	59.4 (1.3)	61.0 (0.8)	**64.0** (2.0)
-mm	62.5 (2.1)	63.2 (2.1)	61.4 (1.6)	62.5 (0.8)	**65.5** (2.7)
SNLI	66.9 (2.4)	67.0 (3.1)	65.8 (2.1)	67.0 (2.9)	**69.9** (2.4)
QNLI	60.7 (1.7)	60.0 (2.5)	60.2 (2.0)	60.9 (2.0)	**66.4** (3.5)
QQP	62.5 (2.6)	**69.0** (1.7)	65.4 (1.2)	62.2 (2.7)	68.8 (3.8)
RTE	64.3 (2.7)	**65.6** (1.5)	65.3 (2.4)	60.2 (1.9)	65.1 (3.5)
MRPC	75.5 (5.2)	69.4 (7.0)	66.5 (7.0)	**78.3** (3.1)	$78.2^{\dagger}$ (3.1)
MR	83.3 (1.4)	85.0 (1.0)	84.6 (1.2)	84.0 (1.4)	**85.8** (0.6)
MPQA	83.6 (1.8)	82.3 (1.9)	84.3 (1.4)	72.3 (13.4)	**84.6** (1.5)
CR	88.9 (1.0)	89.3 (0.6)	**89.6** (0.7)	87.7 (1.1)	89.4 (1.0)

Table 2: Comparing our SupCon loss with Decoupling Label Loss (Dec), Label Condition Loss (Lab), and Contextual Calibration (ConCal). † We can achieve stronger performance 80.0 ± 1.8 by fixing templates/demonstrations when creating the second view of the input (see Section 6.2).

We further show that our method outperforms two latest methods that are designed to improve prompt-based language models. In ADAPET (Tam et al., 2021), the authors replace the traditional CrossEntropy loss with Decoupling Label Loss and Label Condition Loss in the prompt-based fine-tuning method PET, without demonstrations. Contextual Calibration (Zhao et al., 2021) calibrates the output probabilities by considering context-free inputs, *i.e.*, " " or "N/A". (Further see Appendix I)

From Table 2 we observe that on 12 tasks our $\mathcal{L}_{\text{SupCon}}$ outperforms the other losses, while performs on-par in other tasks. Contextual Calibration does not achieve good results overall. We speculate two reasons for this. First, Contextual Calibration is designed for large models without fine-tuning like GPT (zero-shot setting). Second, the form of in-context learning in Contextual Calibration is different from the demonstrations we study here.

4.3 Ensemble vs. our single model

Our method uses 20 generated templates (auxiliary prompts) to construct multi-views of input sentences. But only a single prompt (primary prompt) and one set of label words are used for main predictions. Thus, there is only a single model from our method. Here, we compare our model to an ensemble comprised of 20 models trained separately with the 20 prompts. From Table 3, we find that our method even outperforms the ensemble with 20× more number of parameters, showing that it is a more efficient way to make use of the generated prompts. We speculate that because of the over-fitting nature of few-shot learners, members in the ensemble fail to produce substantial diverse prediction distributions.

Task	LM-BFF +ours	LM-BFF ensemble
SST-5 (acc)	**49.5** (1.1)	48.0 (0.8)
CoLA (Matt.)	**10.2** (5.8)	7.5 (4,7)
MNLI (acc)	**63.3** (2.4)	62.2 (1.8)
MNLI-mm (acc)	**65.1** (2.4)	64.0 (1.8)
QNLI (acc)	**66.4** (3.5)	63.8 (2.7)
MR (acc)	**85.8** (0.6)	85.7 (0.7)

Table 3: Comparing our single model trained with SupCon loss to an ensemble of 20 models.

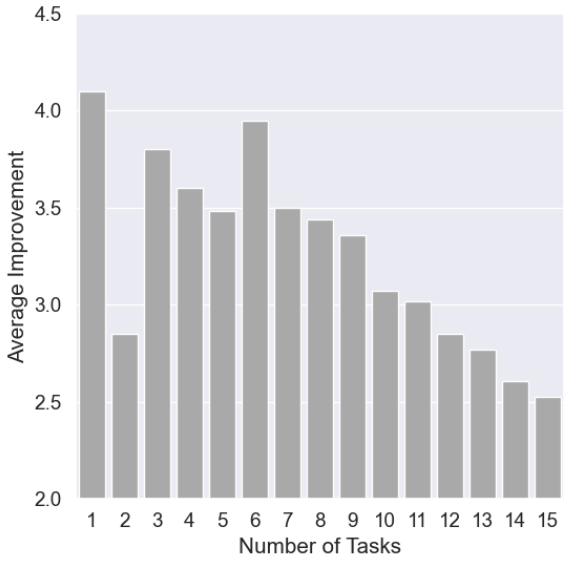

Figure 2: The average improvements achieved by our method on the top K hardest tasks, where K goes from 1 to 15.

5 Improvements vs. Task Difficulty

Here, we show that the improvements achieved by our method are greater for tasks with higher difficulty. To show this, we first sort the 15 tasks by base (LM-BFF) performance and use this ranking as a proxy for the difficulty of the task. Next, we report the average improvements achieved by our

method on the top K hardest tasks, where K goes from 1 to 15. Figure 2 shows these results. The first bar corresponds to the improvement achieved by our method on the hardest task, the second bar corresponds to the average improvement achieved by our method on the hardest and second-hardest tasks, and so on. The last bar corresponds to the average improvement on all 15 tasks.

6 Comparative Experiments

6.1 Input augmentation

The success of contrastive learning heavily relies on the data augmentation. Our method takes advantage of prompt-based language learners and naturally creates multi-views of a single input by appending it with different templates and/or demonstrations. Compared to EDA which includes synonym replacement (SR), random insertion (RI), random swap (RS) and random deletion (RD), our strategy for augmentation does not lead to incomplete and inconsistent sentences, while introducing adequate variations for effective learning.

Task	LM-BFF	SR	RI	RS	RD	EDA	ours
SST-2	89.2	90.7	**90.8**	90.7	90.7	90.5	90.6
Subj	88.6	90.6	90.8	**91.0**	90.5	89.1	90.4
SST-5	47.9	47.7	49.2	48.2	47.9	46.7	**49.5**
CoLA	6.1	5.8	6.5	4.9	4.0	3.9	**10.2**
TREC	82.8	78.1	80.7	79.0	80.7	80.6	**83.3**
MNLI	61.0	61.8	62.4	61.0	58.1	58.9	**64.0**
-mm	62.5	63.6	64.8	62.7	60.3	60.9	**65.5**
SNLI	66.9	63.1	66.4	67.2	65.2	62.2	**69.9**
QNLI	60.7	65.3	65.3	**67.4**	64.8	62.5	66.4†
QQP	62.5	64.5	65.8	68.0	63.2	61.0	**68.8**
RTE	64.3	61.4	61.4	61.3	62.1	61.1	**65.1**
MRPC	75.5	77.6	77.7	**79.3**	78.7	79.1	78.2†
MR	83.3	85.5	85.5	85.5	85.3	85.6	**85.8**
MPQA	83.6	82.2	84.4	84.4	83.9	82.8	**84.6**
CR	88.9	88.9	88.2	88.3	88.5	87.1	**89.4**

Table 4: Comparing our random templates/demonstrations as data augmentation to SR, RI, RS, RD and EDA. Numbers are average of 5 train-test splits. † We can achieve stronger performance by fixing templates/demonstrations when creating the second view of the input, see Section 6.2.

The results in Table 4 are obtained by applying SR, RI, RS, RD, EDA for 10% of input tokens (Results for 20% are in Appendix F). In contrast to ours, EDA, etc., for SupCon lead to worse performances than the baseline method in many tasks.

6.2 Variable templates, demonstrations

So far, we have shown the results by our method generating multi-views of inputs by appending *both* random templates and demonstrations. However, we find that in some tasks fixed templates with random demonstrations or random templates with fixed demonstration lead to even stronger performances (see Table 5). For example, sampling demonstrations with fixed templates for MRPC achieves a very strong result (80.0), outperforming all other methods in Table 4.

Task	LM-BFF	- demo + temp	+ demo - temp	+ demo + temp
SST-2 (acc)	89.2 (1.3)	**90.8** (0.3)	90.5 (0.4)	90.6 (0.1)
Subj (acc)	88.6 (3.3)	**90.8** (0.8)	90.6 (1.2)	90.4 (1.1)
SST-5 (acc)	47.9 (0.8)	49.3 (1.7)	48.9 (1.8)	**49.5** (1.1)
CoLA (Matt.)	6.1 (5.3)	9.9 (7.5)	8.5 (5.6)	**10.2** (5.8)
TREC (acc)	82.8 (3.1)	83.4 (0.5)	**86.7** (1.0)	83.3 (1.5)
MNLI (acc)	61.0 (2.1)	63.4 (3.3)	63.0 (3.2)	**64.0** (2.0)
MNLI-mm (acc)	62.5 (2.1)	65.5 (3.1)	64.9 (3.4)	**65.5** (2.7)
SNLI (acc)	66.9 (2.4)	69.8 (2.4)	68.5 (1.9)	**69.9** (2.4)
QNLI (acc)	60.7 (1.7)	65.4 (3.1)	**67.0** (3.6)	66.4 (3.5)
QQP (acc)	62.5 (2.6)	**68.9** (3.2)	67.8 (1.4)	68.8 (3.8)
RTE (acc)	64.3 (2.7)	64.9 (3.8)	62.6 (2.8)	**65.1** (3.5)
MRPC (F1)	75.5 (5.2)	79.0 (1.8)	**80.0** (1.8)	78.2 (3.1)
MR (acc)	83.3 (1.4)	85.8 (0.7)	85.4 (0.3)	**85.8** (0.6)
MPQA (acc)	83.6 (1.8)	84.0 (1.9)	84.1 (2.0)	**84.6** (1.5)
CR (acc)	88.9 (1.0)	88.6 (0.6)	88.2 (1.0)	**89.4** (1.0)

Table 5: Different strategies to construct multi-views of input sentences. Fixed demonstrations and sampling templates (- demo + temp), sampling demonstrations and fixed templates (+ demo - temp) and sampling both demonstrations and templates (+ demo + temp).

7 Limitations

Since SupCon clusters examples on class level, our framework applies only to classification tasks. Also, our framework requires large GPU memory, as SupCon is an in-batch contrastive loss that needs a large batch size.

8 Conclusion

We proposed a novel supervised contrastive learning framework and an effective augmentation method using prompts that can boost the performance of prompt-based language learners and outperform recent work on 15 few-shot tasks.

9 Ethical Considerations

As far as we are aware, our proposed work does not have any ethical considerations. However, our work relies on pre-trained language models, which have been shown to be biased in prior work (Liang et al., 2021). As such, users of such models should be aware of and if possible address such issues. The data and the code for this work will be made available to aid reproducibility.

References

Farid Arthaud, Rachel Bawden, and Alexandra Birch. 2021. Few-shot learning through contextual data augmentation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1049–1062, Online. Association for Computational Linguistics.

Trapit Bansal, Rishikesh Jha, Tsendsuren Munkhdalai, and Andrew McCallum. 2020. Self-supervised meta-learning for few-shot natural language classification tasks. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 522–534.

Samyadeep Basu, Karine lp Kiun Chong, Amr Sharaf, Alex Fischer, Vishal Rohra, Michael Amoake, Hazem El-Hammamy, Ehi Nosakhare, Vijay Ramani, and Benjamin Han. 2021. Semi-supervised few-shot intent classification and slot filling. *CoRR*, abs/2109.08754.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020a. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1597–1607. PMLR.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. 2020b. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 22243–22255. Curran Associates, Inc.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Association for Computational Linguistics (ACL)*.

Yiren Jian, Karim Ahmed, and Lorenzo Torresani. 2020. Task meta-transfer from limited parallel labels. *Meta-Learning workshop, NeurIPS 2020*.

Yiren Jian and Chongyang Gao. 2021. Metapix: Domain transfer for semantic segmentation by meta pixel weighting. *Image and Vision Computing*, 116:104334.

Yiren Jian, Chongyang Gao, and Soroush Vosoughi. 2022. Embedding hallucination for few-shot language learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics.

Yiren Jian and Lorenzo Torresani. 2022. Label hallucination for few-shot classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Akbar Karimi, Leonardo Rossi, and Andrea Prati. 2021. AEDA: an easier data augmentation technique for text classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 2748–2754. Association for Computational Linguistics.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. 2020. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 18661–18673. Curran Associates, Inc.

Varun Kumar, Hadrien Glaude, Cyprien de Lichy, and Wlliam Campbell. 2019. A closer look at feature space data augmentation for few-shot intent classification. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 1–10, Hong Kong, China. Association for Computational Linguistics.

Judith Yue Li and Jiong Zhang. 2021. Semi-supervised meta-learning for cross-domain few-shot intent classification. *MetaNLP 2021*, page 67.

Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. 2021. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR.

Shikun Liu, Shuaifeng Zhi, Edward Johns, and Andrew J Davison. 2021. Bootstrapping semantic segmentation with regional contrast. *arXiv preprint arXiv:2104.04465*.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *EACL*.

Amr Sharaf, Hany Hassan, and Hal Daumé III. 2020. Meta-learning for few-shot nmt adaptation. *ACL*, page 43.

Derek Tam, Rakesh R Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020a. Contrastive multiview coding. In *Computer Vision – ECCV 2020*, pages 776–794, Cham. Springer International Publishing.

Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020b. Contrastive representation distillation. In *International Conference on Learning Representations*.

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

Jason Wei, Chengyu Huang, Soroush Vosoughi, Yu Cheng, and Shiqi Xu. 2021. Few-shot text classification with triplet networks, data augmentation, and curriculum learning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5493–5500, Online. Association for Computational Linguistics.

Jason Wei and Kai Zou. 2019. EDA: Easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6383–6389, Hong Kong, China. Association for Computational Linguistics.

Yuwen Xiong, Mengye Ren, and Raquel Urtasun. 2020. Loco: Local contrastive representation learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 11142–11153. Curran Associates, Inc.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. 2018. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate Before Use: Improving Few-shot Performance of Language Models. In *International Conference on Machine Learning (ICML)*.

A Batch size and learning details

We use the same learning rate of $1e^{-5}$ for MLM loss as LM-BFF. To take full advantage of SupCon, we apply large batch sizes (16, 32, 40). We show the batch size and learning rate for SupCon in Table A.1. Note that for results of LM-BFF shown in the main paper, we use the same large batch size of our method to allow for fair comparisons.

We set the batch size to be dividable by the total number of examples in the task and small enough to fit into the GPU memory. The experiments with RoBERTa-base are carried out on one NVIDIA RTX-A6000 with 48 GB of memory. Experiments with RoBERTa-large require 4x NVIDIA RTX-8000 (or RTX-A6000) with 192 (4x 48) GB of momery.

Following LM-BFF, our fine-tuning runs a maximum of 1000 steps.

Task	Batch	LR
SST-2	16	$1e^{-6}$
Subj	16	$1e^{-5}$
SST-5	40	$1e^{-5}$
CoLA	16	$1e^{-5}$
TREC	32	$1e^{-5}$
MNLI	24	$1e^{-5}$
MNLI-mm	24	$1e^{-5}$
SNLI	32	$1e^{-5}$
QNLI	16	$1e^{-5}$
QQP	32	$1e^{-5}$
RTE	32	$1e^{-6}$
MRPC	16	$1e^{-5}$
MR	16	$1e^{-6}$
MPQA	16	$1e^{-5}$
CR	32	$1e^{-5}$

Table A.1: Batch size and learning rate (LR) for SupCon loss used for each task.

B Fine-tuning with prompts and demonstrations

We also consider LM-BFF as our baseline method due to its state-of-the-art performance in a wide range of few-shot tasks. The given masked language model $\mathcal{M}$ first encodes the input sentence x_{in} into a sequence of tokens $\tilde{x}_{\text{in}}$ and maps $\tilde{x}_{\text{in}}$ to a sequence of hidden states $\{\mathbf{h}_1, \mathbf{h}_2, ...\mathbf{h}_L\}$, where L is the length of the sequence and $\mathbf{h} \in \mathbb{R}^d$, where d is the dimension of the hidden states. For example, in prompt-base fine-tuning, for single sentence text x_{in} *e.g.*, *"The story is not worth reading."*), the input with the prompt (*e.g.*, *"a truly `[MASK]` one ."*) takes the form of

$$\begin{aligned} x_{\text{prompt}} &= \texttt{[CLS]}\, x_{\text{in}}, \text{a truly } \texttt{[MASK]} \text{ one.} \texttt{[SEP]} \\ &\equiv \mathcal{T}(x_{\text{in}}) \end{aligned}$$

Then, the model decides whether it is more likely to put the label word *"great"* or *"terrible"* at the `[MASK]` position. Fine-tuning with this fill-in-the-blank framework has been shown to be superior to standard fine-tuning (Schick and Schütze, 2021). By mapping the label space $\mathcal{Y}$ to the label words where $\mathcal{V}(y)$ denotes the label word for class y, the prediction of the model $\mathcal{M}$ for class $y \in \mathcal{Y}$ can be written as

$$p(y|x_{\text{in}}) = p(\texttt{[MASK]} = \mathcal{V}(y)|x_{\text{prompt}}) \quad (4)$$

$$= \frac{\exp(\mathbf{w}_{\mathcal{V}(y)} \cdot \mathbf{h}_{\texttt{[MASK]}})}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}_{\mathcal{V}(y')} \cdot \mathbf{h}_{\texttt{[MASK]}})} \quad (5)$$

where $\mathbf{w}$ is the weight vector of MLM head.

In LM-BFF, the authors further append demonstrations to the input x_{prompt} to help the model better understand what is *"great"* and *"terrible"*. Formally, $x_{\text{prompt}} \equiv \mathcal{T}(x_{\text{in}})$ and $\tilde{\mathcal{T}}(x^c_{\text{in}}, y^c)$ denote $\mathcal{T}(x^c_{\text{in}})$ with `[MASK]` replaced by the label word $\mathcal{V}(y^c)$. Then, the input to LM-BFF takes the form of

$$\mathcal{T}(x_{\text{in}}) \oplus \tilde{\mathcal{T}}(x^1_{\text{in}}, y^1) \oplus ... \oplus \tilde{\mathcal{T}}(x^{|\mathcal{Y}|}_{\text{in}}, y^{|\mathcal{Y}|}) \quad (6)$$

In this paper, we use random sampling for the demonstrations, *i.e.*, x^c_{in} is randomly chosen from the training set. The masked language modeling loss is then

$$\mathcal{L}_{\text{MLM}} = \sum_{(x_{\text{in}}, y) \in \mathcal{D}_{\text{train}}} -\log p(y|x_{\text{in}}) \quad (7)$$

C Language-based Supervised Contrastive Loss

Our method extends the loss $\mathcal{L}_{\text{MLM}}$ with an additional Supervised Contrastive Loss (SupCon). For applying SupCon on multi-views of an input text, we need to first obtain a second view of a text:

$$\tilde{x}_{2k} = Aug(\tilde{x}_{2k-1}) \quad (8)$$

As we show in ablations, traditional data augmentation for text does not work well in the contrastive framework. Thus, we propose obtaining a second

Task	Template	Label words
SST-2	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
SST-5	$<S_1>$ It was [MASK] .	v.positive: great, positive: good, neutral: okay, negative: bad, v.negative: terrible
MR	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
CR	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
MPQA	$<S_1>$ It was [MASK] .	positive: great, negative: terrible
Subj	$<S_1>$ This is [MASK] .	subjective: subjective, objective: objective
TREC	[MASK] : $<S_1>$	abbreviation: Expression, entity: Entity, description: Description human: Human, location: Location, numeric: Number
CoLA	$<S_1>$ This is [MASK] .	grammatical: correct, not_grammatical: incorrect
MNLI	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, netural: Maybe, contradiction: No
SNLI	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, netural: Maybe, contradiction: No
QNLI	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, not_entailment: No
RTE	$<S_1>$? [MASK] , $<S_2>$	entailment: Yes, not_entailment: No
MRPC	$<S_1>$ [MASK] , $<S_2>$	entailment: Yes, not_entailment: No
QQP	$<S_1>$ [MASK] , $<S_2>$	entailment: Yes, not_entailment: No

Table D.1: Primary templates and label words used in our experiments.

view by randomly changing the templates and/or demonstrations:

$$\tilde{x}_{2k-1} = \mathcal{T}_{t_0}(x_{\text{in}}) \oplus ... \oplus \tilde{\mathcal{T}}_{t_0}(x^c_{\text{in}}, y^c) \oplus ... \quad (9)$$

$$\tilde{x}_{2k} = \mathcal{T}_{t_j}(x_{\text{in}}) \oplus ... \oplus \tilde{\mathcal{T}}_{t_j}(\hat{x}^c_{\text{in}}, y^c) \oplus ... \quad (10)$$

where T denotes a set of pre-defined templates, $t_j \in T$ and $t_j \neq t_0$. $\hat{x}^c_{\text{in}}$ is another randomly sampled example as the demonstration text and $\hat{x}^c_{\text{in}} \neq x_{\text{in}}$. This strategy serves as a perfect form of augmentation for our purpose as it does not generate incomplete or inconsistent sentences, and since we do not edit the main input, the label for that input stays the same. Furthermore, $\tilde{x}_{2k}$ has a substantial variation from $\tilde{x}_{2k-1}$, which allows for effective contrastive learning.

D Manual primary prompts

Table D.1 shows the primary prompts we used for each task. Those prompts are manually chosen by LM-BFF (Gao et al., 2021).

E Experiments with RoBERTa-large

While we use RoBERTa-base to conduct extensive experiments in our main study and ablations, here we compare our framework to LM-BFF using RoBERTa-large. We also include results directly reported from LM-BFF (Gao et al., 2021) (henceforth referred to as LM-BFF†) , though the comparison between them could be unfair since the results reported in the original LM-BFF paper are:

- obtained with an additional sampling strategy to select similar demonstrations (section 6.2 of their paper), which put our results at a disadvantage.
- obtained from a set of batch sizes (2,4,8) and learning rates ($1e^{-5}, 2e^{-5}, 5e^{-5}$) and the best models are selected from the validation set. Whereas we show experimental results with models trained with a fixed batch size and a learning rate of $1e^{-5}$.

Nevertheless, we show the state-of-the-art results we achieved in Table E.1. We only marginally under-perform LM-BFF† in 2 tasks, possibly due to the reasons listed above.

Task	LM-BFF†	LM-BFF‡	ours
SST-2 (acc)	92.6 (0.5)	91.9 (1.3)	**94.2** (0.7)
Subj (acc)	92.3 (0.8)	91.0 (2.3)	**92.4** (0.6)
SST-5 (acc)	50.6 (1.4)	51.5 (1.4)	**54.0** (0.8)
CoLA (Matt.)	**18.7** (8.8)	11.6 (5.4)	18.1 (10.1)
TREC (acc)	87.5 (3.2)	85.4 (2.6)	**89.8** (1.8)
MNLI (acc)	70.7 (1.3)	70.0 (2.2)	**72.4** (2.0)
MNLI-mm (acc)	72.0 (1.2)	71.8 (2.0)	**74.2** (1.9)
SNLI (acc)	**79.7** (1.5)	79.5 (1.7)	79.6 (2.6)
QNLI (acc)	69.2 (1.9)	67.9 (3.4)	**71.1** (6.8)
QQP (acc)	N/A	71.1 (2.3)	**74.0** (2.5)
RTE (acc)	68.7 (2.3)	69.3 (1.1)	**71.8** (1.1)
MRPC (F1)	77.8 (2.0)	77.0 (5.7)	77.8 (4.6)
MR (acc)	86.6 (2.2)	85.6 (3.4)	**89.6** (0.8)
MPQA (acc)	**87.0** (1.1)	86.9 (1.3)	86.9 (1.1)
CR (acc)	90.2 (1.2)	90.4 (1.2)	**91.0** (1.4)

Table E.1: Comparison of our method using RoBERT-large to two version of LM-BFF: (1) † the results reported in LM-BFF (Gao et al., 2021). Note that we do *not* have their strategy of sampling similar demonstrations, which may put us at a disadvantage. The training details, hyper-parameters and number of GPUs are also different (2) ‡ The results of LM-BFF using the same batch size, learning rate, training steps, and number of GPUs. These results make the fairest comparison to ours.

Task	SR	RI	RS	R D	EDA	ours
SST-2 (acc)	90.6 (0.5)	90.8 (0.4)	**90.8** (0.4)	90.8 (0.4)	90.7 (0.6)	90.6 (0.1)
Subj (acc)	90.4 (1.4)	90.4 (1.3)	90.4 (2.5)	90.3 (1.4)	90.4 (1.1)	**90.4** (1.1)
SST-5 (acc)	47.6 (1.4)	47.0 (1.8)	46.5 (1.7)	46.9 (1.9)	45.2 (2.1)	**49.5** (1.1)
CoLA (Matt.)	6.0 (5.3)	6.0 (5.4)	7.2 (3.8)	5.6 (3.0)	5.6 (2.9)	**10.2** (5.8)
TREC (acc)	80.7 (2.2)	79.1 (3.8)	81.2 (2.2)	82.8 (2.1)	81.1 (4.3)	**83.3** (1.5)
MNLI (acc)	60.3 (2.2)	61.2 (2.1)	60.7 (2.6)	60.2 (2.1)	58.3 (2.5)	**64.0** (2.0)
MNLI-mm (acc)	62.2 (1.5)	63.3 (1.3)	63.0 (1.7)	62.9 (1.2)	60.2 (2.1)	**65.5** (2.7)
SNLI (acc)	63.4 (4.1)	63.4 (3.8)	62.3 (3.6)	62.0 (4.3)	63.0 (4.2)	**69.9** (2.4)
QNLI (acc)	63.2 (3.3)	64.5 (4.6)	64.0 (4.3)	64.8 (4.5)	60.8 (3.6)	**66.4** (3.5)
QQP (acc)	64.8 (2.8)	62.9 (2.2)	62.0 (3.5)	63.9 (2.1)	60.4 (5.8)	**68.8** (3.8)
RTE (acc)	61.2 (3.0)	62.2 (3.7)	47.9 (0.8)	47.9 (0.8)	64.3 (2.7)	**65.1** (3.5)
MRPC (F1)	77.0 (3.8)	**79.2** (4.6)	78.5 (2.1)	77.4 (3.2)	76.2 (6.0)	78.2 (3.1)
MR (acc)	83.3 (1.4)	85.5 (0.5)	85.6 (0.6)	85.2 (0.3)	85.7 (0.7)	**85.8** (0.6)
MPQA (acc)	82.6 (2.8)	82.7 (2.4)	83.4 (2.4)	84.3 (2.0)	83.1 (2.9)	**84.6** (1.5)
CR (acc)	87.8 (0.9)	88.1 (0.3)	87.5 (1.0)	88.5 (1.0)	87.9 (0.5)	**89.4** (1.0)

Table F.1: Comparing our random templates/demonstrations as data augmentation to synonym replacement (SR), random insertion (RI), random swapping (RS), random deletion (RD) and EDA (Wei and Zou, 2019) (with SR, RI, RS and RD all together) at 20% of input tokens. The results are means of 5 runs with different train-test splits.

F Augmentation with 20% of input tokens

In the main paper, we compare our augmentation strategy (random templates and random demonstrations) to standard augmentation techniques on 10% of input tokens for creating multi-view of inputs to apply the SupCon loss. Here, we show additional experimental results with synonym replacement (SR), random insertion (RI), random swapping (RS), random deletion (RD) and EDA (Wei and Zou, 2019) (with SR, RI, RS and RD all together) at 20% of input tokens. Same as before, the model under-performs when using standard augmentations. Results are shown in Table F.1.

G SimCLR vs. SupCon

Here, we compare our choice of contrastive loss SupCon (Khosla et al., 2020) to an unsupervised version SimCLR (Chen et al., 2020a). Unsupervised contrastive loss clusters examples at the instance level, *i.e.*, it only pulls the same instance under different views close to each other and push away all the others in a mini-batch. Whereas SupCon clusters examples at the class level.

As shown in Table G.1. SupCon is better than SimCLR in all tasks. In many cases, SimCLR even underperforms the baselines by large margins (see LM-BFF in Table 1), indicating that learning discriminative features at instance level only can hurt the fine-tuning process.

Task	SimCLR	SupCon
SST-2 (acc)	89.7 (0.8)	**90.6** (0.1)
Subj (acc)	86.4 (1.2)	**90.4** (1.1)
SST-5 (acc)	42.1 (0.8)	**49.5** (1.1)
CoLA (Matt.)	1.8 (3.6)	**10.2** (5.8)
TREC (acc)	60.0 (4.4)	**83.3** (1.5)
MNLI (acc)	52.5 (1.2)	**64.0** (2.0)
MNLI-mm (acc)	53.0 (1.2)	**65.5** (2.7)
SNLI (acc)	60.0 (4.5)	**69.9** (2.4)
QNLI (acc)	53.5 (0.6)	**66.4** (3.5)
QQP (acc)	56.4 (2.0)	**68.8** (3.8)
RTE (acc)	58.6 (2.8)	**65.1** (3.5)
MRPC (F1)	68.2 (6.3)	**78.2** (3.1)
MR (acc)	84.7 (1.1)	**85.8** (0.6)
MPQA (acc)	82.2 (1.6)	**84.6** (1.5)
CR (acc)	88.1 (2.1)	**89.4** (1.0)

Table G.1: Comparing SimCLR (Chen et al., 2020a) and SupCon (Khosla et al., 2020) as different forms of contrastive loss.

H `[CLS]` vs. `[MASK]`

SupCon takes the representations of inputs to perform contrastive learning. We use the hidden states at `[MASK]` tokens as the representations of sentences in the main experiments. Another common choice is to take the hidden states at `[CLS]` tokens.

Task	[CLS]	[MASK]
SST-2 (acc)	90.4 (0.5)	**90.6** (0.1)
Subj (acc)	90.0 (1.5)	**90.4** (1.1)
SST-5 (acc)	48.2 (0.6)	**49.5** (1.1)
CoLA (Matt.)	**10.7** (5.7)	10.2 (5.8)
TREC (acc)	81.1 (1.4)	**83.3** (1.5)
MNLI (acc)	59.9 (3.7)	**64.0** (2.0)
MNLI-mm (acc)	61.4 (4.0)	**65.5** (2.7)
SNLI (acc)	66.9 (2.5)	**69.9** (2.4)
QNLI (acc)	65.1 (3.1)	**66.4** (3.5)
QQP (acc)	66.0 (3.0)	**68.8** (3.8)
RTE (acc)	63.4 (3.1)	**65.1** (3.5)
MRPC (F1)	77.7 (1.8)	**78.2** (3.1)
MR (acc)	85.5 (0.8)	**85.8** (0.6)
MPQA (acc)	84.2 (1.6)	**84.6** (1.5)
CR (acc)	88.4 (1.2)	**89.4** (1.0)

Table H.1: Using hidden states at [CLS] tokens or [MASK] tokens as the representations of sentences to perform contrastive learning.

For example, in standard fine-tuning, the algorithm takes the representation of a sentence at [CLS] token and attaches a linear classifier on top of it.

Based on Table H.1, applying the contrastive loss at [MASK] tokens is generally better than applying it at [CLS]. This is fairly intuitive, as the final classifications are performed at [MASK] tokens and enforcing class-level discriminative representations explicitly at [MASK] tokens helps models generalize better after fine-tuning.

I Adapting ADAPET

For results in Table 2, we adapt the open-source code from ADAPET to our codebase. In original ADAPET, there are multiple label words corresponding to a label class (e.g positive class: *"great", "good", "nice"*). To make a fair comparison to LM-BFF and ours, we only apply one label word corresponding to a label class (e.g positive class: *"great"*). The original Label Condition Loss implemented in ADAPET has a hyper-parameter α to control the percentage of input tokens used for masked language modeling (see ADAPET (Tam et al., 2021) for more details). To match the training objective in LM-BFF with only one [MASK] token, we also set the Label Condition Loss to apply to one random token of inputs, i.e., $\alpha = \frac{1}{len(input)}$.