Leveraging Self-Attention for Input-Dependent Soft Prompting in LLMs

Ananth Muppidi*
IIIT Hyderabad
India
ananth.muppidi21@gmail.com

Abhilash Nandy*
IIT Kharagpur
India
nandyabhilash@gmail.com

Sambaran Bandyopadhyay
Adobe Research
India
samb.bandyo@gmail.com

Abstract

The performance of large language models in domain-specific tasks necessitates fine-tuning, which is computationally expensive and technically challenging. This paper focuses on parameter-efficient fine-tuning using soft prompting, a promising approach that adapts pre-trained models to downstream tasks by learning a small set of parameters. We propose a novel Input Dependent Soft Prompting technique with a self-Attention Mechanism (*ID-SPAM*) that generates soft prompts based on the input tokens and attends different tokens with varying importance. Our method is simple and efficient, keeping the number of trainable parameters small. We show the merits of the proposed approach compared to state-of-the-art techniques on various tasks and show the improved zero shot domain transfer capability.

1 Introduction

Large language models (LLMs) have made significant advancements in natural language processing tasks, such as generation, translation and summarization (Yeo et al., 2023; Zhang et al., 2023a). Despite their success, LLMs' performance in domain-specific tasks is limited, and fine-tuning on task-oriented datasets is crucial. As models from BERT (Devlin et al., 2019) to GPT-3 (Brown et al., 2020) have millions to billions of parameters, fine-tuning becomes computationally expensive and challenging. Therefore, parameter efficient fine-tuning (Han et al., 2024) research aims to adapt pre-trained models to downstream tasks by fixing most parameters and only learning a small subset.

Soft prompting is a promising direction for fine-tuning large models. Without changing the core architecture of an LLM, soft prompt methods generally introduce a small trainable vector (known as a 'soft prompt') at the beginning of one or more transformer layers' inputs within the LLM. During fine tuning, only the soft prompt is trained to adapt to the downstream task keeping the parameters of the base LLM frozen. Lester et al. (2021) propose Prompt Tuning by prepending the trainable soft prompt vector before the embeddings of the text input, just after the embedding layer of the base LLM. On similar lines, Li and Liang (2021) introduce Prefix Tuning by prepending a soft prompt at every transformer layer and Liu et al. (2021) come up with P-tuning by interleaving learnable prompts with input embeddings. Contrary to text prompt engineering (Wei et al., 2022) or optimizing discrete token representations via in-context learning (Dai et al., 2023), Petrov et al. (2023) suggest that the continuous embedding space of soft prompts inherently possesses a greater amount of information.

Recent literature introduces several variants of soft prompt techniques such as removing the reparameterization module (Liu et al., 2022b), hierarchical structured pruning (Ma et al., 2022), introducing an adaptive gate mechanism to control the prefix importance in each transformer layer (Zhang et al., 2023b), diving the soft prompt into query, key and value prompts (Wang et al., 2023), learning multiple short soft prompts and a gating mechanism to route an input to a specific soft prompt (Choi et al., 2023), and decomposing the soft prompt into low rank matrices (Shi and Lipani, 2024).

Many of these methods keep the soft prompt independent of the actual input given to the LLM. However, this limits the soft prompt to adjust based on the actual input during the inference time. It is unlikely that a unified prompt would lead to a performance improvement across different input instances. It also makes the training difficult by increasing the convergence time. To address this, a few recent approaches leverage input dependent soft prompts. But they need to concatenate the soft prompts either at every transformer layer of the base LLM (Wu et al., 2022) or all the layers after an

*Equal contribution. Work done during the internship at Adobe Research.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 960–969
July 27 - August 1, 2025

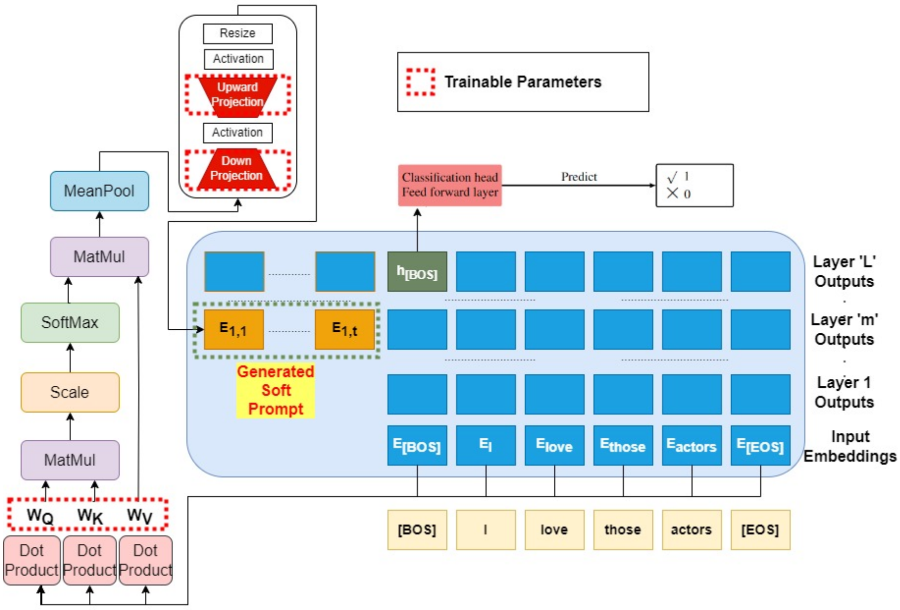

Figure 1: *ID-SPAM* Framework. Given an LM, the generated soft-prompt can be prepended to any transformer layer's inputs (the figure can be best seen in color)

intermediate layer (Liu et al., 2022a), or transform the soft prompt by using cross-attention with the input tokens without explicitly generating from them (Jin et al., 2023). These input dependent prompting techniques still have multiple limitations: (i) Many of them employ relatively complicated architecture by concatenating soft prompts in multiple internal transformer layers of the LLM; (ii) Since, a task may contain diverse samples with different types of words, it is important to attend different words of the input with different weights while generating the soft prompt; And (iii) Number of trainable parameters often increases significantly.

To address the above research gaps, we introduce an input dependent soft prompt technique where the soft prompt is generated by a trainable network that attends different tokens of the input with different importance by employing a self-attention mechanism. We prepend the soft prompt with the input to a single transformer layer of the base LLM, keeping the number of trainable parameters small and training smooth. Following are the *contributions* made in this work: **(i)** We propose *ID-SPAM*, a novel (Input Dependent Soft Prompting technique with a self-Attention Mechanism); Our method is simple and efficient to train. **(ii)** We show the merit of the proposed approach on six tasks from the GLUE benchmark (Wang et al., 2018); And **(iii)** Due to the use of trainable attention on the input tokens, our approach is more efficient in zero-shot domain transfer as shown in the experiment.

2 Proposed Solution

In this section, we introduce our proposed method *ID-SPAM* (see its framework in Figure 1).

Given a Task T having training data represented as $D_{train} = \{(x_i, y_i)\}_{i=1}^{K}$. Following Lester et al. (2021), we represent the input as $x_i = \mathbf{E}(\texttt{[SEP]}S_1\texttt{[SEP]}S_2\texttt{[EOS]})$ for a task with a pair of sentences S_1, S_2 as the input or $x_i = \mathbf{E}(\texttt{[SEP]}S_1\texttt{[EOS]})$ for for a task with a single sentence S_1 as the input, where $\mathbf{E}(\cdot)$ is the token embedding for the input sentence(s).

We introduce a learnable soft prompt such that the prompt not only varies with the task at hand, but is also generated based on the input in such a way that it primarily attends to those input tokens that are essential for the given task. To make the learning efficient, we freeze the parameters of the original LM M. Our proposed soft prompt for the task T can be defined as $\mathbf{S}_T \in \mathbb{R}^{n \times t}$, where t is the number of tokens in the prompt representation and n is the hidden dimension of the LM $\mathbf{M}$ under consideration. $\mathbf{S}_T$ is obtained by first applying a *learnable* attention layer (Vaswani et al., 2017) over the input embeddings $\mathbf{E}(\cdot)$ and averaging the outputs, providing a context-rich representation. The $n \times 1$ dimensional vector A so obtained is passed through a downward projection MLP Layer having learnable weights $\mathbf{W}_{down} \in \mathbb{R}^{n \times c}$ and bias $\mathbf{b}_{down} \in \mathbb{R}^{c}$, followed by a ReLU Activation Layer (Nair and Hinton, 2010), and then an upward projection MLP

Layer having learnable weights $\mathbf{W}_{up} \in \mathbb{R}^{c \times n.t}$ and bias $\mathbf{b}_{down} \in \mathbb{R}^{n.t}$, where $c < n$. The output so obtained is re-sized to get the learnable, input-dependent soft prompt $\mathbf{S}_T \in \mathbb{R}^{n \times t}$, which is either prepended to the token embeddings or to the input of any intermediate transformer layer of the LM **M**. We will show some analysis on the choice of intermediate layer in the experiments. Mathematically,

$$A = \text{mean}\left\{ \text{softmax}\left(\frac{(\mathbf{E}\mathbf{W}_Q)(\mathbf{E}\mathbf{W}_K)^\top}{\sqrt{d_k}} \right)(\mathbf{E}\mathbf{W}_V) \right\} \quad (1)$$

$$\mathbf{S}_T = \text{resize}(\sigma(W_{up}\sigma(W_{down}(A)))) \quad (2)$$

W_Q, W_K, and W_V are the query, key, and value parameter matrices respectively, and $\frac{1}{\sqrt{d_k}}$ is a scaling factor, as used in Vaswani et al. (2017). σ is a non-linear activation which we used ReLU here.

3 Experimental Evaluation

Here, we describe our experimental setup, evaluate *ID-SPAM* framework on GLUE and SuperGLUE benchmarks, and zero-shot domain transfer between tasks against several baselines, followed by a detailed analysis.

3.1 Experimental Setup

We compare *ID-SPAM* with the following baselines - (1) **Transformer fine-tuning:** Here, all parameters of LM are learned (2) **Parameter-Efficient Soft Prompt-based Methods -** (a) **Prompt Tuning:** We use standard prompt tuning (Lester et al., 2021), which learns soft prompts through backpropagation to condition frozen language models for specific tasks. (b) **P-tuning:** P-tuning (Liu et al., 2022b) is a variant of Deep Prompt Tuning (Li and Liang, 2021; Qin and Eisner, 2021) adapted for NLU (c) **Sparse Mixture of Prompts (SMoP):** SMoP (Choi et al., 2023) leverages multiple short soft prompts with a gating mechanism to train multiple prompts tailored in addressing different data subsets (d) **Late Prompt Tuning (LPT):** LPT (Liu et al., 2022a) injects a late prompt into an intermediate layer of the LM, rather than into the input layer or across all layers. (e) **Decomposed Prompt Tuning (DePT):** DePT (Shi and Lipani, 2024) employs a decomposition strategy for the soft prompt, breaking it down into a pair of low-rank matrices. These components are then optimized independently, each with its own specific learning rate. (3) **Parameter Efficient Fine-tuning using Low-Rank Adaptation (LoRA):** LoRA (Hu et al., 2022) addresses challenge of fine-tuning large language models by freezing pre-trained model's weights and introducing trainable low-rank matrices into each layer. Note that it does not use a soft prompt.

For all methods, we train upto 30 epochs (***Section E of Appendix*** shows convergence after 30 epochs) using Standard Cross-Entropy Loss and Adam Optimizer (Loshchilov and Hutter, 2018), and number of soft-prompt tokens $t = 10$. We perform hyperparameter tuning for *ID-SPAM*, as described ***in Section A of Appendix***. We use a NVIDIA A100 GPU with a VRAM of 80 GB for all experiments.

3.2 Evaluation on GLUE Benchmark

We evaluate *ID-SPAM* and baselines on the following 6 Natural Language Understanding (NLU) Tasks from GLUE Benchmark (Wang et al., 2018) - SST-2 (Socher et al., 2013), MRPC (Dolan and Brockett, 2005), MNLI (Williams et al., 2018), QNLI (Rajpurkar et al., 2016), RTE (Dagan et al., 2005), and QQP (Quora, 2017). These tasks cover various aspects of natural language understanding and inference, providing a comprehensive assessment of our approach's performance across different language processing challenges. All datasets were obtained from the Hugging Face library (Wolf et al., 2020; Lhoest et al., 2021). Further dataset statistics are shared in Table 1.

We report accuracy for {SST, MNLI, QNLI, RTE} and average of accuracy and macro F1-Score for {MRPC, QQP} using RoBERTa-BASE, RoBERTa-LARGE backbones (Liu et al., 2019) in Table 2.

We infer that *ID-SPAM* outperforms all Parameter-Efficient Soft Prompt-based baselines on 4 out of 6 GLUE tasks and w.r.t average task performance, and is a close second for 2 tasks, when using both RoBERTa-BASE and RoBERTa-LARGE backbones. This could be attributed to the attention layer followed by 2-layer MLP in *ID-SPAM*, which efficiently generates a context-rich soft prompt. Also, *ID-SPAM* is shown to be more or similarly efficient compared to well-performing LPT baseline ***in Section D of Appendix***.

Section B of Appendix shows - *ID-SPAM* performs better than Soft Prompt baselines - (1) on 2/4 and 3/4 **SuperGLUE** (Wang et al., 2019) tasks using RoBERTA-BASE and RoBERTA-LARGE backbones respectively, while giving best average score; (2) when using autoregressive GPT-2 backbone on 3/6 and 2/4 GLUE and SuperGLUE tasks

Category	Datasets	|Train|	|Dev|	|Labels|	Type	Labels
Single-sentence	SST-2	67349	872	2	sentiment	positive, negative
Sentence-pair	MNLI	392702	19647	3	NLI	entailment, neutral, contradiction
	MRPC	3668	408	2	paraphrase	equivalent, not equivalent
	QNLI	104743	5463	2	NLI	entailment, not entailment
	QQP	363846	40430	2	paraphrase	equivalent, not equivalent
	RTE	2490	277	2	NLI	entailment, not entailment

Table 1: Statistics of the datasets used in our experiments.

	MNLI	QNLI	SST-2	MRPC	RTE	QQP	Mean
Method	**GLUE (RoBERTa-BASE Backbone)**						
Fine-tuning	$87.4_{2.4}$	$91.3_{1.0}$	$92.3_{0.6}$	$92.7_{0.7}$	$82.5_{1.3}$	$90.9_{0.8}$	89.5
LoRA	$88.7_{0.4}$	$84.2_{2.1}$	$90.4_{0.3}$	$79.3_{0.5}$	$77.6_{1.1}$	$81.8_{0.2}$	83.7
Prompt Tuning	$78.3_{2.1}$	$81.4_{1.1}$	$89.3_{1.4}$	$74.4_{0.7}$	$57.9_{0.5}$	$77.8_{1.6}$	76.5
P-Tuning	$82_{2.2}$	$82.5_{0.3}$	$88.1_{0.5}$	$81.9_{1.7}$	$67.4_{0.9}$	$84.2_{0.1}$	81
SMoP	$80.7_{1.0}$	$82.9_{1.4}$	$89.8_{0.3}$	$78.1_{2.1}$	$71.7_{1.8}$	$83.7_{0.9}$	81.2
LPT	$81.7_{0.6}$	$83.2_{1.1}$	$91.8_{1.3}$	$\mathbf{84.3}_{0.2}$	$73.6_{0.7}$	$84.1_{0.5}$	83.1
DEPT	$81.5_{0.3}$	$\mathbf{87.9}_{1.2}$	$90.2_{1.2}$	$75.7_{0.6}$	$71.2_{1.0}$	$79.2_{0.3}$	81.0
ID-SPAM (ours)	$\mathbf{83.1}_{0.8}$	$86.4_{0.4}$	$\mathbf{92.7}_{1.2}$	$82.8_{0.3}$	$\mathbf{79.2}_{0.4}$	$\mathbf{84.6}_{0.5}$	**84.8**
Method	**GLUE (RoBERTa-LARGE Backbone)**						
Fine-tuning	$87.6_{1.7}$	$94.7_{2.3}$	$95.4_{1.3}$	$92.1_{1.2}$	$88.4_{0.3}$	$90.7_{0.2}$	91.48
LoRA	$89.1_{1.1}$	$87.9_{0.3}$	$95.1_{0.2}$	$86.5_{0.9}$	$78.7_{0.1}$	$88.4_{0.3}$	87.6
Prompt Tuning	$83.4_{1.1}$	$88.2_{0.2}$	$92.6_{0.5}$	$73.9_{1.4}$	$60.8_{0.6}$	$81.2_{0.6}$	80.0
P-Tuning	$86.4_{0.7}$	$88.7_{1.2}$	$\mathbf{95.8}_{0.8}$	$76.3_{1.1}$	$62.6_{0.5}$	$85.2_{1.3}$	82.5
SMoP	$86.7_{1.1}$	$88.4_{2.2}$	$\mathbf{95.8}_{1.4}$	$79.6_{0.8}$	$76.3_{1.4}$	$86.7_{0.3}$	85.6
LPT	$84.2_{1.1}$	$86.1_{0.5}$	$93.4_{1.4}$	$\mathbf{87.3}_{0.2}$	$74.2_{0.7}$	$85.3_{1.3}$	85.1
DEPT	$83.3_{1.2}$	$88.8_{1.3}$	$91.2_{1.8}$	$77.7_{0.3}$	$73.2_{0.8}$	$82.2_{0.7}$	82.7
ID-SPAM (ours)	$\mathbf{87.4}_{0.8}$	$\mathbf{91.1}_{0.4}$	$94.6_{1.2}$	$86.1_{0.3}$	$\mathbf{81.1}_{0.4}$	$\mathbf{88.4}_{0.5}$	**88.1**

Table 2: Test results on GLUE benchmark. We use RoBERTa-BASE, RoBERTa-LARGE Backbones for all methods. We report the score, along with stddev for 3 runs (in the subscript) for all tasks. The best performing Soft Prompt-based method's results are in bold

respectively, while giving better average score; (3) on average when using a GPT-2 Large Backbone.

Comparison with LoRA: *ID-SPAM* gives better average score compared to LoRA. Specifically, *ID-SPAM* outperforms LoRA in 5/6 and 3/6 tasks when using RoBERTa-BASE and RoBERTa-LARGE backbones respectively. Also, *ID-SPAM* is shown to be more efficient than LoRA based on the number of trainable parameters and training and inference times ***in Section D of Appendix***.

Ablation Analysis: We compare the results of *ID-SPAM* with just using mean-pooling directly using the RoBERTa-LARGE backbone on 3 GLUE Datasets in Table 3. *ID-SPAM* outperforms mean-pooling on all 3 tasks, giving an average improvement of 5.82%, thus highlighting the importance of the self-attention layer in *ID-SPAM*.

Method	MRPC	RTE	QQP
Mean-pooling	82.3	75.2	84.2
ID-SPAM	**86.1**	**81.1**	**88.4**

Table 3: Ablation Analysis on *ID-SPAM*

3.3 Evaluation on SuperGLUE Benchmark

We compare *ID-SPAM* with several Soft Prompt-Based Baselines on 4 SuperGLUE Datasets using RoBERTA-BASE and RoBERTA-LARGE backbones in Tables B and 5 respectively. We observe that *ID-SPAM* outperforms the baselines on 2/4 and 3/4 tasks using RoBERTA-BASE and RoBERTA-LARGE backbones respectively, while also giving the best average score.

	CB	COPA	MultiRC	BoolQ	Mean
Prompt Tuning	75.9	52.5	67.2	63.6	64.8
P-Tuning	76.3	54.7	67.9	63.7	65.6
SMoP	79.9	57.7	67.2	69.7	68.6
LPT	80.6	**59.2**	70.8	66.3	69.2
DEPT	78.6	52.9	67.1	**71.4**	67.5
ID-SPAM	**83.9**	57.8	**72.9**	69.9	**71.1**

Table 4: Test results on 4 SuperGLUE Datasets using RoBERTa-BASE Backbone. The best performing method is bold for each task.

	CB	COPA	MultiRC	BoolQ	Mean
Prompt Tuning	78	53	67.2	63.3	65.4
P-Tuning	76	55	68.1	64.0	65.8
SMoP	81.9	59	69.6	**71.1**	70.4
LPT	82	**60**	71.0	68.0	70.2
DEPT	79	54	69.0	71.0	68.2
ID-SPAM	**85**	**60**	**73.0**	70.0	**72.0**

Table 5: Test results on 4 SuperGLUE Datasets using RoBERTa-LARGE Backbone. The best performing method is bold for each task.

3.4 Zero-Shot Task, Domain Transfer

Table 6 shows Zero-Shot Transfer using RoBERTa-LARGE backbone, where a model is trained on training set of a dataset, evaluated on another dataset. We use (QQP, MRPC) and (SST-2, IMDB)[1] pairs for transfer across tasks and domains respectively similar to Lester et al. (2021). Table 6 shows *ID-SPAM* performs better than Soft Prompt-based baselines, showing *ID-SPAM* is generaliz-

[1]Task for SST-2 and IMDB is binary classification. SST-2 contains phrases, while IMDB contains full movie reviews

able across datasets. *ID-SPAM* even outperforms Fine-tuning in 3/4 pairs. Also, even though *ID-SPAM* has much less number of parameters compared to LoRA (***see Section D of Appendix***), *ID-SPAM* gives better/comparable performance. In addition, we show that *ID-SPAM* performs better/comparable to well-performing LPT baseline in Few-Shot Task Transfer ***in Section C of Appendix.***

Tuning Method	**QQP→ MRPC**	**MRPC→ QQP**	**SST-2→ IMDB**	**IMDB→ SST-2**
Fine-tuning	$64.0_{0.7}$	$68.3_{1.3}$	$87.1_{0.0}$	$88.8_{0.4}$
LoRA	$71.1_{0.1}$	$66.1_{0.4}$	$90.3_{0.2}$	$87.6_{1.1}$
Prompt Tuning	$54.1_{0.3}$	$54.6_{0.2}$	$68.7_{1.1}$	$63.5_{3.8}$
P-Tuning	$57.6_{1.2}$	$52.7_{1.1}$	$66.5_{0.0}$	$66.8_{1.3}$
SMoP	$67.9_{0.4}$	$64.1_{0.6}$	$84.5_{0.5}$	$83.3_{1.0}$
LPT	$66.7_{0.4}$	$64.5_{0.3}$	$67.1_{0.8}$	$71.1_{1.6}$
DEPT	$63.3_{1.8}$	$58.8_{0.5}$	$69.8_{0.1}$	$69.3_{0.9}$
ID-SPAM (ours)	$\mathbf{70.9}_{1.2}$	$\mathbf{69.2}_{0.4}$	$\mathbf{89.1}_{0.3}$	$\mathbf{86.0}_{0.8}$

Table 6: Mean, stddev of zero-shot task, domain transfer for different methods. 'Score' is average of Accuracy and macro F1-Score. The best performing Soft Prompt-based method's results are in bold.

3.5 Method Analysis

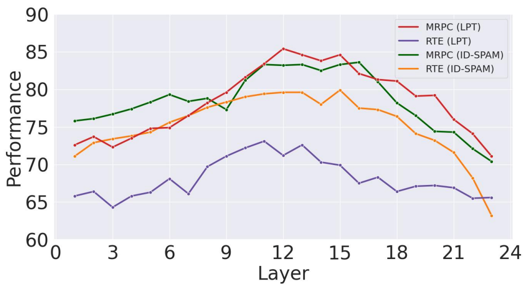

Figure 2: Effect of Variation in layer index (m) corresponding to which soft prompt is prepended on performance ($m = 0$ refers to input embeddings). Metrics are average of acc. and F1 for MRPC and acc. for RTE.

We analyze the effect of varying layer index where soft prompt is prepended (m in Figure 1) on performance of LPT and *ID-SPAM* on 2 GLUE datasets using RoBERTa-LARGE backbone in Figure 2. We infer that *ID-SPAM* and LPT perform better when soft prompt is prepended to inputs in middle layers of LM. Also, *ID-SPAM* significantly outperforms LPT corresponding to almost every layer index for RTE Dataset. Also, *ID-SPAM* performs better for earlier layers, as soft prompt is generated by using a single attention layer over input embeddings. Hence, prepending this prompt to an earlier layer's outputs performs better than later layer's outputs, as later layer's outputs are obtained after input embeddings are passed through several attention layers, reducing compatibility with the soft prompt. Also, if we prepend soft prompt to later layers, it passes through a small number of layers of LLM, thus showing a reduced performance.

4 Discussions and Conclusion

In this paper, we propose *ID-SPAM* which significantly improves parameter-efficient fine-tuning and zero-shot task and domain transfer performance on various NLU tasks compared to several SOTA parameter-efficient baselines. Notably, further analysis shows that *ID-SPAM* performs reasonably well when the generated soft prompt is prepended at any layer's inputs. Hence, *ID-SPAM* is an efficient, input-dependent soft prompt generation framework that could generalize well across several NLP tasks.

5 Limitations

We have shown that our proposed approach *ID-SPAM* improves the performance of two backbone LLMs (RoBERTa-BASE and RoBERTa-LARGE) on multiple NLP tasks. Our framework is generic and can be used with any open source LLMs as backbone. However, we could not use more recent very large scale pre-trained LLMs (like Llama-3.1-70B and Mixtral 8x22B) with tens of billions of parameters as backbone LMs in our experiments due to limited computational resources. We are interested to see the performance gain when we use our approach with those large scale state-of-the-art LLMs in some future work.

In the current work, we do not have an automated way to choose the layer of the LM where we input the soft prompt. The layer number is kept as a hyperparameter in the current work and its effect is shown in Section 3.5. In future, we want to automatically identify the optimal transformer layer, as proposed by Zhu and Tan (2023).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Joon-Young Choi, Junho Kim, Jun-Hyung Park, Wing-Lam Mok, and SangKeun Lee. 2023. Smop: Towards efficient and effective prompt tuning with sparse mixture-of-prompts. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In *Proceedings of the PASCAL Challenges Workshop on Recognising Textual Entailment*, volume 1.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 4005–4019.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

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

Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. 2023. Instance-aware prompt learning for language understanding and generation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22(7):1–18.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059.

Quentin Lhoest, Albert Villanova del Moral, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Timo Müller, Isabella Géron, Simon Brandeis, Sylvain Gugger, Théo Matussière, Abhishek Thakur, Philipp Schmid, Yacine Jernite, Jeff Boudier, Francesco Calefato, Clara Ma, Clement Delangue, Thibault Goehringer, Victor Sanh, Canwen Xu, Alexander M. Rush, and Thomas Wolf. 2021. Datasets: A community library for natural language processing. *Preprint*, arXiv:2109.02846.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597.

Xiangyang Liu, Tianxiang Sun, Xuan-Jing Huang, and Xipeng Qiu. 2022a. Late prompt tuning: A late prompt could be better than many prompts. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1325–1338.

Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. 2022b. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv:2103.10385*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *Preprint*, arXiv:1907.11692.

Ilya Loshchilov and Frank Hutter. 2018. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Fang Ma, Chen Zhang, Lei Ren, Jingang Wang, Qifan Wang, Wei Wu, Xiaojun Quan, and Dawei Song. 2022. Xprompt: Exploring the extreme of prompt tuning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11033–11047.

Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA. Omnipress.

Aleksandar Petrov, Philip Torr, and Adel Bibi. 2023. When do prompting and prefix-tuning work? a theory of capabilities and limitations. In *The Twelfth International Conference on Learning Representations*.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5203–5212, Online. Association for Computational Linguistics.

Quora. 2017. Quora question pairs.

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

Zhengxiang Shi and Aldo Lipani. 2024. DePT: Decomposed prompt tuning for parameter-efficient fine-tuning. In *The Twelfth International Conference on Learning Representations*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *Advances in neural information processing systems*, 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Qifan Wang, Yuning Mao, Jingang Wang, Hanchao Yu, Shaoliang Nie, Sinong Wang, Fuli Feng, Lifu Huang, Xiaojun Quan, Zenglin Xu, et al. 2023. Aprompt: Attention prompt tuning for efficient adaptation of pre-trained language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9147–9160.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, VG Vinod Vydiswaran, and Hao Ma. 2022. Idpg: An instance-dependent prompt generation method. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5507–5521.

Yee Hui Yeo, Jamil S Samaan, Wee Han Ng, Peng-Sheng Ting, Hirsh Trivedi, Aarshi Vipani, Walid Ayoub, Ju Dong Yang, Omer Liran, Brennan Spiegel, et al. 2023. Assessing the performance of chatgpt in answering questions regarding cirrhosis and hepatocellular carcinoma. *medRxiv*, pages 2023–02.

Haopeng Zhang, Xiao Liu, and Jiawei Zhang. 2023a. Summit: Iterative text summarization via chatgpt. *arXiv preprint arXiv:2305.14835*.

Zhen-Ru Zhang, Chuanqi Tan, Haiyang Xu, Chengyu Wang, Jun Huang, and Songfang Huang. 2023b. Towards adaptive prefix tuning for parameter-efficient language model fine-tuning. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1239–1248.

Wei Zhu and Ming Tan. 2023. Spt: Learning to selectively insert prompts for better prompt tuning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Appendix

A Experiment Settings

For our experiments, we use `roberta-base` and `roberta-large` implementations from HuggingFace. For all baselines, the number of appended prompt tokens (for Prompt Tuning, P-tuning, Late Prompt Tuning) are set to 10 tokens. For DEPT, we set the rank to 45. For P-Tuning, we set the encoder reparameterization type to MLP. For *ID-SPAM*, appended prompt tokens are set to 10 tokens. The search space for hyperparameters for tuning are shown in Table 7. For all experiments, standard CrossEntropyLoss was used. For all experiments, we train using a warm-up rate of 0.06, and AdamW optimizer with ϵ of 1×10^{-6}, β_1 of 0.9, β_2 of 0.98.

In Figure 2, we can see that layers 11-13 show optimal performance for both *ID-SPAM* and LPT. LPT (Liu et al., 2022a) shows that the 13th layer is optimal. This makes our method *ID-SPAM* comparable to LPT taking the layer number into account. Also, following the trend from other prior art on soft prompts (Lester et al., 2021; Liu et al., 2022a; Li and Liang, 2021; Choi et al., 2023), we used the best hyperparameter set for each of the baselines. Our experimental approach is also logical and consistent as the experimental settings (choice of backbone LMs, datasets) are same for baselines and our method *ID-SPAM*.

Hyperparameter	Values
Epochs	{1, 5, 10, 20, 30}
Batch Size	{16, 32, 64}
Learning Rates	{1e-3, 5e-4, 1e-4, 5e-3, 1e-5}
Dropout Rate	{0.1, 0.2, 0.3}
Weight Decay	{0, 0.01, 0.1}
Layer (RoBERTa-Large)	{1,2,3...23}
Layer (RoBERTa-Base)	{1,2,3...11}

Table 7: Hyperparameters used for tuning *ID-SPAM*.

B Evaluation using GPT-2 and GPT-2 Large Backbones

Using GPT-2 Backbone. We carry out experiments with decoder-only GPT-2 backbone on 6 GLUE Datasets - Table 8 shows that when using GPT-2 as backbone, *ID-SPAM* outperforms LPT on 3/6 tasks and gives an average performance improvement of 2.3%.

Next, we carry out experiments with decoder-only GPT-2 backbone on 4 SuperGLUE Datasets –

	MNLI	QNLI	SST-2	RTE	QQP	MRPC	AVG
LPT	69.5	**79.4**	**90.1**	62.8	80.3	**81.9**	77.3
ID-SPAM	**78.3**	77.1	85.1	**71.6**	**82.9**	79.5	**79.1**

Table 8: Test results on 6 GLUE Datasets using GPT-2 Backbone. The best performing PEFT method is bold for each task.

Table 9 shows that compared to Soft Prompt-Based baselines, *ID-SPAM* gives the best average score, and performs the best on 2 tasks, while performing the second best on one of them.

	CB	COPA	MultiRC	BoolQ	Mean
Prompt Tuning	71.7	57	61.7	64.1	63.6
P-Tuning	73.3	57.7	63.2	65.7	65
SMoP	81.4	61.2	68.4	69.4	70.1
LPT	82.1	61.3	72.1	**74.1**	72.4
DEPT	76.1	55.1	**73.5**	67.2	68
ID-SPAM	**88.1**	**63.1**	71.7	72.4	**73.8**

Table 9: Test results on 4 SuperGLUE Datasets using GPT-2 Backbone. The best performing method is bold for each task.

Using GPT-2 Large Backbone. We compare the performance of *ID-SPAM* with LoRA and LPT using a large generative model GPT-2 Large (around 0.8 Billion Parameters) as the backbone on 2 GLUE Datasets - RTE and MRPC, as shown in Table 10.

Method	RTE	MRPC	Average
LoRA	74.0	80.0	77.0
LPT	69.9	82.9	76.4
ID-SPAM	73.7	81.1	**77.4**

Table 10: Test results on 2 GLUE Datasets using GPT-2 Large Backbone.

ID-SPAM gives an average improvement of 0.5% and 1.3% compared to LoRA and LPT respectively across the 2 GLUE Datasets, showing that *ID-SPAM* is competitive even for a large, generative backbone LM.

C Few-Shot Task Transfer

Train	Eval (Few-shot, 100 samples)	Tuning	Score
MRPC	QQP	Fine-Tuning	81.7
MRPC	QQP	LPT	74.4
MRPC	QQP	*ID-SPAM*	73.1
QQP	MRPC	Fine-Tuning	79.7
QQP	MRPC	LPT	69.4
QQP	MRPC	*ID-SPAM*	72.5

Table 11: Few-shot task transfer for different methods using the RoBERTa-LARGE Backbone.

ID-SPAM and LPT (a well-performing baseline in Table 2) using the RoBERTa-LARGE Backbone are fine-tuned on the first dataset, and then further fine-tuned on 100 training samples from the second. This model is then evaluated on the second dataset.

From Table 11, we can see that *ID-SPAM* performs better than LPT on QQP->MRPC, while the performance is comparable for MRPC->QQP.

D Comparison of *ID-SPAM* with baselines w.r.t model size and training and inference times

Model	LPT	LoRA	*ID-SPAM*
RoBERTa-BASE	2,162,688	3,495,312	2,064,384
RoBERTa-LARGE	2,883,584	7,931,280	3,538,944

Table 12: number of trainable parameters of *ID-SPAM* and well-performing baselines LPT and LoRA (see Table 2).

Table 12 shows that the number of trainable parameters in *ID-SPAM* is lesser than that of LoRA for both backbones, and is lesser than that of LPT using RoBERTa-BASE backbone, while they are comparable in case of RoBERTa-LARGE backbone.

Backbone	No. of Parameters in Backbone LM	*ID-SPAM*	LoRA
GPT2	126.8	2.1	2.4 (1.1x)
GPT2-medium	361.1	3.5	6.3 (1.8x)
GPT2-large	785.8	5.1	11.8 (2.3x)
GPT2-xl	1577.3	8.3	19.7 (2.4x)
Gemma-2B (Team et al., 2024)	2525.8	13.4	19.6 (1.5x)
FLAN-T5-xl (Chung et al., 2024)	2823.6	13.4	35.5 (2.6x)

Table 13: Number of trainable parameters (in millions) of *ID-SPAM* compared to LoRA for several LM backbones of different sizes. The decrease in the number of trainable parameters of *ID-SPAM* compared to LoRA is written within brackets.

Table 13 shows that as the size of the backbone LM increases, efficiency in the number of trainable parameters of *ID-SPAM* compared to LoRA tends to increase. Hence, *ID-SPAM* is suitable even for massive LMs.

Dataset	Method	Training Time per sample (in secs)	Inference Time per sample (in secs)
BoolQ	LPT	0.669	0.236
BoolQ	LoRA	0.715	0.313
BoolQ	*ID-SPAM*	0.651	0.251
WiC	LPT	0.082	0.041
WiC	LoRA	0.113	0.067
WiC	*ID-SPAM*	0.084	0.035

Table 14: Training and inference times of ID-SPAM and well-performing baselines LPT and LoRA for 2 SuperGLUE Datasets.

Table 14 shows that *ID-SPAM* requires less time for training as well as for inference, in comparison to LoRA on both BoolQ (a yes/no QA dataset) and WiC (dataset for binary classification) Datasets (2 datasets from SuperGLUE). Also, *ID-SPAM* takes lesser time to train on BoolQ than LPT, while the times are comparable on WiC. In case of inference, *ID-SPAM* takes lesser time than LPT for WiC, while taking slightly more time than LPT for BoolQ. Hence, *ID-SPAM* has comparable training and inference times w.r.t LPT, while giving better performance on GLUE datasets (see Table 2).

	MNLI	QNLI	SST-2	RTE	QQP	MRPC
Fine Tuning	2887s	270s	224s	247s	1854s	87s
LPT	2013s	157s	161s	168s	1157s	59s
ID-SPAM	1902s	166s	171s	168s	1004s	51s

Table 15: Total training time cost before convergence (in seconds) of *ID-SPAM* compared to baselines

Table 15 shows the training convergence times (in seconds - lower the better) for LPT and our proposed *ID-SPAM* (By convergence, we mean the epoch where the validation error is the least) using RoBERTa-LARGE Backbone. We can see that *ID-SPAM* gives better/similar convergence time compared to LPT on 4 out of 6 GLUE Tasks. Also, LPT takes an average convergence of time of 619 s, while ID-SPAM takes 577 s, giving an improvement of 7.3% in average convergence time.

E Convergence of the LoRA Baseline

The training loss is tabulated every 5 epochs in Table 16 when training LoRA with the RoBERTa-BASE backbone on the MRPC and RTE Datasets from the GLUE Benchmark.

Epoch	**MRPC**	**RTE**
5	0.21	0.4
10	0.12	0.14
15	0.05	0.07
20	0.02	0.06
25	0.02	0.04
30	0.0001	0.02

Table 16: Training Loss across epochs when training LoRA with the RoBERTa-BASE backbone

We can see that the training loss continuously decreases with increasing epochs on both the MRPC and RTE Datasets. Also, the losses are considerably lowered after 30 epochs as can be seen in the table, thus showing convergence.