Prompt-Augmented Linear Probing: 
Scaling beyond the Limit of Few-Shot In-Context Learners

Hyunsoo Cho¹, Hyuhng Joon Kim¹, Junyeob Kim¹, Sang-Woo Lee²,³, Sang-goo Lee¹, Kang Min Yoo¹,²,* Taeuk Kim⁴,*
¹ Seoul National University
² NAVER Cloud
³ KAIST
⁴ Hanyang University
{johyunsoo, heyjoonkim, juny116, sglee}@europa.snu.ac.kr
{kangmin.yoo, sang.woo.lee}@navercorp.com, kimtaeuk@hanyang.ac.kr

Abstract
Through in-context learning (ICL), large-scale language models are effective few-shot learners without additional model fine-tuning. However, the ICL performance does not scale well with the number of available training samples as it is limited by the inherent input length constraint of the underlying language model. Meanwhile, many studies have revealed that language models are also powerful feature extractors, allowing them to be utilized in a black-box manner and enabling the linear probing paradigm, where lightweight discriminators are trained on top of the pre-extracted input representations. This paper proposes prompt-augmented linear probing (PALP), a hybrid of linear probing and ICL, which leverages the best of both worlds. PALP inherits the scalability of linear probing and the capability of enforcing language models to derive more meaningful representations via tailoring input into a more conceivable form. Throughout in-depth investigations on various datasets, we verified that PALP significantly enhances the input representations closing the gap between ICL in the data-hungry scenario and fine-tuning in the data-abundant scenario with little training overhead, potentially making PALP a strong alternative in a black-box scenario.

Introduction
Since the emergence of Transformer-based (Vaswani et al. 2017) language models, we have witnessed notable improvements in the natural language processing literature, even attaining human-level performance on several benchmarks. In addition, as it becomes evident that scaling laws work for such models (Kaplan et al. 2020), there has been a significant amount of investment in the field to enhance them in terms of the number of their parameters—from millions to billions—and the volume of the data they consume during training (Brown et al. 2020; Chowdhery et al. 2022; Fedus, Zoph, and Shazeer 2022; Hoffmann et al. 2022). As a result, some cutting-edge language models become possible to obtain intriguing extra functionalities, such as the ability to capture world knowledge (Petroni et al. 2019), generate codes (Poesia et al. 2022), or solve mathematical problems (Henighan et al. 2020), in addition to being proficient in recognizing linguistic patterns. These anecdotes, which demonstrate the general power of large language models, naturally raise researchers’ expectations that language models can act as a universal, off-the-shelf solution for a range of downstream tasks while minimizing the cost required for adapting them to a specific job at the same time.

However, there is no free lunch; the effectiveness and generalizability of large language models achieved by scaling come at the cost of physically serving such gigantic neural architectures. Thus, the institutions that distribute large models such as GPT-3 (Brown et al. 2020) usually pursue the strategy of providing commercial APIs which only allow limited access to the models. In other words, it is often the case that users cannot receive information about the inner workings of the models, such as gradients concerning the models’ parameters which are crucial for fine-tuning the models for a particular purpose. Therefore, there has been

*Corresponding authors.
Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
a growing interest in adapting language models in this restricted setting, dubbed as \textit{black-box tuning} (Sun et al. 2022; Diao et al. 2022).

In this scenario, especially focusing on classification, the two mainstream paradigms are linear models and in-context learning (ICL). The former derives the final representation from a language model and trains a simple classifier, e.g., Support Vector Machine (SVM), Single-Layer Perceptron (SLP), and \(k\) Nearest Neighbors (\(k\)-NN), on top of the extracted features. On the other hand, ICL is a training-free mechanism that makes the most use of the nature of language models. Specifically, ICL promotes a language model to generate the desired output by guiding the model with a few examples of the target task (i.e., demonstrations) plus a set of templates tailored for the task, where many works have suggested that exploiting such a mechanism improved various NLP tasks (Shwartz et al. 2020; Ben-David, Oved, and Reichart 2022).

Yet, it is worth noting that according to the number of data instances available for training, the two aforementioned approaches have clear strengths and shortcomings. Namely, ICL shows a remarkable generalization ability in a few-shot setting. However, the ICL performance does not scale well with the number of available training samples (See Figure 1 for details). This is because ICL relies on the innate capability of language models, i.e., being able to predict the next word given context, and a verbalizer, which is a function that maps the probabilities of some pre-defined tokens to the probability of each label rather than constructing a task-specific decision maker. By doing so, ICL is equipped with the invaluable benefit of being free from explicit fine-tuning, but it is also a double-edged sword that induces a scalability issue. In particular, an explicit upper bound exists on the performance of ICL, which is determined by the maximum length of the utilized language model.

In this paper, we show that the combination of linear classifiers and the techniques introduced for ICL can cover the weaknesses of each method. We train diverse linear classifiers whose representations are extracted from input pre-processing strategies invented for facilitating ICL. Specifically, we augment training data instances with the templates or prepend additional demonstrations in front of the input of interest for better contextualization.

We validate our method with various datasets, demonstrating that it is consistently superior to baselines in both the low-data and full-data settings. From empirical experiments, we observe that exploiting templates that provide hints about the target task or concatenating demonstrations can significantly enhance the extracted representations from PLM, improving the classifiers’ performance in various scenarios and reducing the gap between ICL and fine-tuning. Intriguingly, we also discover that importing the techniques directly from ICL without care may cause the inheritance of the disadvantages of ICL, such as a substantial performance variance depending on the appended demonstrations or high sensitivity to the format of templates.

---

**Preliminary**

**Problem Formulation**

In this work, we consider classifying input sequences based on \textit{black-box tuning} (Sun et al. 2022; Diao et al. 2022), where the parameters of pre-trained language models (PLMs) are inaccessible. That is, PLMs only serve as an encoder function \(e\) that delivers \(n\)-dimensional latent features \(h\) from input \(x\). Formally, for input \(x \in X\), let the \(n\)-dimensional continuous latent features extracted from PLMs be \(h = e(x)\), where \(h \in \mathcal{H}\). In addition, let \(Y = \{0, 1, \cdots, \lvert C \rvert\}\) be a label space, where \(\lvert C \rvert\) is the cardinality of the space. Then, a classifier \(p(y|h; \theta) : \mathcal{H} \rightarrow \mathcal{Y}\) maps \(h\) to a class label \(y \in \mathcal{Y}\), estimating the probability of input \(x\) belonging to a certain label.

**Linear Classifiers**

Linear models, such as single layer perceptron (SLP), Support Vector Machine (SVM), or logistic regression (LR), are trained by solving optimization problems concerning their parameters. First-order methods such as gradient descent shown below are a general choice for parameter estimation:

\[
\theta_{t+1} = \theta_t - \eta \nabla \mathcal{L}(\theta_t), \mathcal{H}_{batch}\),
\]

where \(\mathcal{L}, \eta, \mathcal{H}_{batch}\) refer to a loss function (e.g., the cross-entropy loss, hinge loss in SVM, and MSE loss in regression), learning rate, and a mini-batch sampled from the training dataset. In this paper, we evaluate 5 different linear probing classification methods: \(k\)-NN, LR, SVM, gaussian discriminative analysis (GDA), and SLP. The details of the mentioned approaches are presented in the Appendix.

**In-Context Learning**

In-context learning (ICL) is a brand new, training-free paradigm that attempts to make the most use of the nature of language models to conduct a target task. ICL promotes a language model to generate the desired output by guiding the model with a few examples of the target task (i.e., demonstrations) plus a set of templates tailored for the task.

In detail, ICL consists of two steps (Liu et al. 2022b): First, the \textit{input pre-processing} step combines the input of interest \(x\) with \(k\)-shot samples (i.e., demonstrations) \((x_i,y_i)_{i=1}^k\) from the training set. Then, a template function \(f_{\text{template}}(\cdot)\) attach pre-defined descriptions to the input \(f_{\text{template}}(x)\) or additionally attach the corresponding natural language label to the \textit{templified} input \(f_{\text{template}}(x, y)\), commonly referred to as a demonstration. (See Figure 2 for a graphical explanation.)

For instance, the function attaches a prefix or postfix to the original \(x\), or it transforms \(y_i\) into the form of natural language rather than numeric numbers. In consequence, the final input for ICL, denoted as \(\hat{x}\), becomes a concatenation of all the pre-processed \(x\) and \((x_i,y_i)_{i=1}^k\):

\[
\hat{x} = D_1 \oplus D_2 \oplus \cdots \oplus D_k \oplus f_{\text{template}}(x),
\]

where \(D_i = f_{\text{template}}(x_i, y_i)\) and \(\oplus\) refers to the concatenation operation.

Second, in the \textit{prediction} phase, ICL leverages PLMs to compute the feature \(\hat{h} = e(\hat{x})\), followed by a verbalizer
\[ \mathcal{V}: \mathcal{H} \to \mathcal{Y} \] that is a reformulation of the language model head for task-specific adaptation. It is often assumed that the verbalizer only considers single tokens as its output candidates, which correspond to each item in the label space \( \mathcal{Y} \).

**Prompt-Augmented Linear Probing (PALP)**

**Motivation**

The primary intuition behind our method borrows from the in-context learning ability exhibited by language models. Specifically, in-context learners benefit from more elaborate and longer prompts (Reynolds and McDonell 2021), allowing them to carry out deeper reasoning through longer input sequences and corresponding hidden layers.

We posit that providing appropriate guidance to the language model via prompts (input pre-processing step in ICL) benefits not only the usual causal language modeling (i.e., predicting the next token) ability but also enhances the quality of the representation for the input text. The primary goal of our method is to extract a more distinctive representation from PLMs via crafting a raw dataset into a more understandable form and training linear probers on the top of the extracted representations. Specifically, we transform the dataset in two ways:

1. We utilize a simple template that gives a brief description of input and the objective of the task.
2. On top of templified dataset, we concatenate a single class-wise demonstration to an inferring input to give important cues, i.e., input-label mapping, label space, or distribution of the input text (Min et al. 2022b) regarding the target task.

In the following subsection, we explain the dataset reconstruction strategies for each method.

**PALP-Template (PALP-T)**

Applying a template to the input is the most straightforward and intuitive way to enforce PLM to follow user requirements. Accordingly, we attempt to extract more task-specific features by attaching a fixed prefix, infix (for sentence pair tasks), or postfix, which describes a raw input into a more understandable form.

For instance, we transform sentiment analysis instance ‘very interesting,’ into ‘Review: very interesting. Sentence: ’ to provide the language model additional indications that the input is the form of review, and the user expects sentiment as an answer. Formally, for the training dataset \( D_{\text{train}} = \{(x_i, y_i) | i \in m \} \), we convert this into \( D^\text{template}_{\text{train}} = \{(f_{\text{template}}(x_i), y_i) | i \in m \} \), where \( f_{\text{template}}(\cdot) \) is a template function. Then we train linear classifiers with transformed dataset \( D^\text{template}_{\text{train}} \) in the exact same way as in Eq. 1.

In the inference time, we also have to apply the same template function to inferring input \( f_{\text{template}}(x_{\text{test}}) \) in order to match the format.

**PALP-Demonstration (PALP-D)**

On top of the previous templified dataset (PALP-T), PALP-D additionally concatenates some demonstrations to the front to maximize the capability of PLM to learn in-context. However, unlike ICL, which attaches all available samples, our method extracts a single representative demonstration per class (i.e., total \( |C| \) demonstrations) and leverages them as demonstrations. Formally, let there exist \( k \) accessible training samples per each class label:

\[ D_{\text{train}} = \{(x^j_i, y^j_i) | j \in k, i \in |C| \}, \] (3)

Figure 2: Overall illustration of ICL. ICL utilizes a template function to convert available few-shot samples into demonstrations \( f_{\text{template}}(x, y) \), and appends the generated demonstrations to the front of the templified current input \( f_{\text{template}}(x) \). Then, ICL uses LM’s prediction ability from the pre-training step to infer the correct answer.
Figure 3: Illustration of selecting demonstrations in PALP-D in the binary classification task. We estimate the normal distribution for each class from available training samples and select the closest sample, respectively.

where $|C|$ is the cardinality. Then, we choose one instance from each class and concatenate them into one prefix $\tau$:

$$\hat{x} = \tau \oplus f_{\text{template}}(x), \text{ where }$$

$$\tau = D_1 \oplus \cdots \oplus D_{|C|}. \quad (4)$$

In this way, we can avoid atrocity lengthy input, minimizing the computational cost and enhancing the method’s scalability. Specifically, while the length of the input in ICL is normally proportional to the total available number of samples, PALP-D is proportional to the number of labels of the task.

Given that PALP-D does not inject complete available training samples, the core is selecting a single meaningful demonstration that can distill PLM sufficient knowledge to comprehend essential signs of the task, such as input-label mapping, label space, or distribution of the input text. We hypothesize that the inputs closest to the centroid of each class label can capture the most representative information for respective classes and extract them as a set of demonstrations $D = \{D_1, D_2, \cdots, D_{|C|}\}$.

In order to do so, we first estimate the normal distribution for each class label $N(\mu_i \in \mathcal{C}_i, \Sigma_i \in \mathcal{C}_i)$ from the available training inputs and measure the distance between entire training samples and the estimated centroid through Mahalanobis distance:

$$\text{dist}_{\text{Mah}}(x, \mu_i; \Sigma_i) = (x - \mu_i)^\top \Sigma_i^{-1} (x - \mu_i). \quad (6)$$

Finally, we select samples closest to each class centroid and utilize them as demonstrations $D_i = \min_{j \in k} (x_j^i, u_i; \Sigma_i)$. (Figure 3 illustrates the selection of demonstration from available training samples.)

In a few-shot scenario, we randomly permuted the order of available demonstrations $P$ to construct multiple prefixes $\hat{\tau}$ in the training phase, where $\hat{\tau} \sim \sigma(P)$ and $\sigma(\cdot)$ refers to a random permutation operator. By doing so, we can generate $|C|!$ different prefixes that can give the effect of data augmentation and alleviate the data-scarcity problem. And in the inference phase or data-abundant setting, we attach a unified prefix $\tau$ (without random permutation) to the front of the test input to match the format with the training samples.

Table 1: Statistics of 15 different datasets used in our experiments.

<table>
<thead>
<tr>
<th>Task type</th>
<th>Dataset</th>
<th># Class</th>
<th># Train</th>
<th># Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Sentence</td>
<td>SST2</td>
<td>2</td>
<td>67,349</td>
<td>872</td>
</tr>
<tr>
<td></td>
<td>Rotten_tomatoes</td>
<td>2</td>
<td>8,530</td>
<td>1,066</td>
</tr>
<tr>
<td></td>
<td>Offensive</td>
<td>2</td>
<td>19,916</td>
<td>860</td>
</tr>
<tr>
<td></td>
<td>ColA</td>
<td>2</td>
<td>8,551</td>
<td>1,043</td>
</tr>
<tr>
<td></td>
<td>Stance_atheism</td>
<td>3</td>
<td>461</td>
<td>220</td>
</tr>
<tr>
<td></td>
<td>Emotion</td>
<td>4</td>
<td>3,257</td>
<td>1,421</td>
</tr>
<tr>
<td></td>
<td>AG_news</td>
<td>4</td>
<td>120,000</td>
<td>7,600</td>
</tr>
<tr>
<td></td>
<td>TREC</td>
<td>6</td>
<td>5,452</td>
<td>500</td>
</tr>
<tr>
<td></td>
<td>Banking77</td>
<td>77</td>
<td>10,003</td>
<td>3,080</td>
</tr>
<tr>
<td></td>
<td>CLINC</td>
<td>150</td>
<td>15,000</td>
<td>4,500</td>
</tr>
</tbody>
</table>

| Sentence pair | MNLI        | 3       | 392,702 | 9,832 |
|               | MRPC        | 2       | 3,668   | 408   |
|               | RTE         | 2       | 2,490   | 277   |
|               | BoolQ       | 2       | 9,427   | 3,270 |
|               | CB          | 3       | 250     | 56    |

Experiments

Experimental Setup

**Backbone.** In the following experiments, we adopt GPT-J (Wang and Komatsuzaki 2021) as the main backbone of our experiments, with additional experiments using GPT-2 (Radford et al. 2019).

**Datasets.** To investigate the performance of each method in many different scenarios, we select 15 datasets, as stipulated in Table 1. The selected dataset covers single sentence task to sentence pair tasks, binary to 150 classes (various numbers of class labels), in diverse domains. The detailed list of each dataset and references are covered in Appendix.

**Setting & Reporting Methods.** Our experiments cover both a few-shot setting and full training data setting. As a reporting method, we select currently prevailing linear probing methodologies (i.e., SVM, SLP, LR, GDA, and k-NN) and their application of PALP. In a few-shot setting, we take a closer look at how the performance of each method changes when the training input is converted into ICL style and compare them with ICL, which is known to yield superior per-
performance in the data-hungry setting. In the full-shot setting, we investigate the performance gap between our method and the white-box training method (i.e., accessible to model parameters), such as Adapter (Houlsby et al. 2019) or fine-tuning, which can be considered upper bound.

Other details. We optimized the hyper-parameters of each classification method on SST2 dataset with 4 Tesla V100 SXM2 32GB GPUs and universally utilized them in different settings. (Detailed hyper-parameters and implementations are in the Appendix.) Additionally, we found a manual template for each task where ICL exhibited sound performance and utilized them universally in our methods. (All templates for each task are stipulated in the Appendix.) For stable evaluation, we report the average of 5 different seeds (13, 27, 250, 583, 915) as a model performance and report standard deviations for each task in the Appendix.

### Few-shot Results
In the few-shot setting, we experimented based on the number of accessible samples for each task class. For instance, a 4-shot setting in the Sentiment Analysis task with 2 classes (positive and negative) means that a total of 8 samples are accessible, which is analogous to a balanced 8-shot setting in ICL. Table 2 summarizes the performance of ICL, baseline linear probing methods, and their application of PALP (T and D) in the 4,8-shot setting. Baseline refers to utilizing raw input without modification, which is a conventional supervised learning paradigm. Template (PALP-T) and demonstration (PALP-D) refer to template-based training samples and demonstration-based training samples from our method individually. We now refer to a T for PALP-T and D for PALP-D for short. (Additional results with 16-shot is in the Appendix.)

<table>
<thead>
<tr>
<th>Method</th>
<th>AG</th>
<th>SST2</th>
<th>Rotten</th>
<th>Stance</th>
<th>Emotion</th>
<th>TREC</th>
<th>CoLA</th>
<th>Offensive</th>
<th>MNLI</th>
<th>RTE</th>
<th>MRPC</th>
<th>AVG</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----</td>
</tr>
<tr>
<td>k-NN</td>
<td>54.66</td>
<td>51.33</td>
<td>58.67</td>
<td>54.73</td>
<td>33.92</td>
<td>48.84</td>
<td>42.36</td>
<td>49.37</td>
<td>35.09</td>
<td>48.74</td>
<td>64.66</td>
<td>49.31</td>
</tr>
<tr>
<td>LR</td>
<td>66.48</td>
<td>50.25</td>
<td>65.08</td>
<td>58.64</td>
<td>36.89</td>
<td>64.48</td>
<td>50.68</td>
<td>51.72</td>
<td>36.46</td>
<td>49.39</td>
<td>59.02</td>
<td>53.55</td>
</tr>
<tr>
<td>SVM</td>
<td>66.20</td>
<td>50.41</td>
<td>65.91</td>
<td>59.27</td>
<td>36.59</td>
<td>65.92</td>
<td>48.90</td>
<td>48.72</td>
<td>36.02</td>
<td>49.31</td>
<td>61.96</td>
<td>53.56</td>
</tr>
<tr>
<td>SLP</td>
<td>67.88</td>
<td>50.23</td>
<td>65.68</td>
<td>59.45</td>
<td>37.13</td>
<td>66.80</td>
<td>49.13</td>
<td>49.95</td>
<td>36.36</td>
<td>49.24</td>
<td>61.76</td>
<td>53.96</td>
</tr>
<tr>
<td>GDA</td>
<td>66.32</td>
<td>50.41</td>
<td>65.93</td>
<td>58.64</td>
<td>36.88</td>
<td>64.44</td>
<td>48.88</td>
<td>48.72</td>
<td>36.03</td>
<td>49.24</td>
<td>61.96</td>
<td>53.59</td>
</tr>
<tr>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----</td>
</tr>
<tr>
<td>k-NN</td>
<td>61.88</td>
<td>58.30</td>
<td>65.10</td>
<td>35.27</td>
<td>45.18</td>
<td>51.04</td>
<td>53.02</td>
<td>59.98</td>
<td>36.40</td>
<td>53.14</td>
<td>61.47</td>
<td>52.80</td>
</tr>
<tr>
<td>LR</td>
<td>71.84</td>
<td>62.00</td>
<td>71.73</td>
<td>58.09</td>
<td>50.56</td>
<td>65.56</td>
<td>49.86</td>
<td>59.00</td>
<td>38.73</td>
<td>50.54</td>
<td>60.44</td>
<td>58.03</td>
</tr>
<tr>
<td>SVM</td>
<td>72.08</td>
<td>63.67</td>
<td>71.33</td>
<td>56.27</td>
<td>50.27</td>
<td>66.36</td>
<td>53.08</td>
<td>62.02</td>
<td>37.93</td>
<td>49.60</td>
<td>61.18</td>
<td>58.53</td>
</tr>
<tr>
<td>SLP</td>
<td>73.79</td>
<td>63.58</td>
<td>72.18</td>
<td>57.45</td>
<td>52.71</td>
<td>68.44</td>
<td>58.81</td>
<td>63.07</td>
<td>38.25</td>
<td>49.96</td>
<td>61.57</td>
<td>59.53</td>
</tr>
<tr>
<td>GDA</td>
<td>72.82</td>
<td>64.20</td>
<td>71.33</td>
<td>56.27</td>
<td>52.86</td>
<td>66.00</td>
<td>53.08</td>
<td>62.02</td>
<td>38.13</td>
<td>49.60</td>
<td>61.18</td>
<td>58.86</td>
</tr>
<tr>
<td>D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-----</td>
</tr>
<tr>
<td>k-NN</td>
<td>75.99</td>
<td>70.16</td>
<td>71.01</td>
<td>49.45</td>
<td>56.95</td>
<td>46.80</td>
<td>55.67</td>
<td>54.56</td>
<td>38.85</td>
<td>51.09</td>
<td>63.86</td>
<td>57.67</td>
</tr>
<tr>
<td>LR</td>
<td>75.62</td>
<td>67.82</td>
<td>70.58</td>
<td>61.27</td>
<td>68.97</td>
<td>65.96</td>
<td>53.83</td>
<td>67.63</td>
<td>40.30</td>
<td>49.00</td>
<td>62.27</td>
<td>63.49</td>
</tr>
<tr>
<td>SVM</td>
<td>75.52</td>
<td>75.46</td>
<td>79.42</td>
<td>55.82</td>
<td>68.91</td>
<td>66.24</td>
<td>55.30</td>
<td>63.75</td>
<td>40.12</td>
<td>51.54</td>
<td>64.76</td>
<td>63.57</td>
</tr>
<tr>
<td>SLP</td>
<td>80.69</td>
<td>77.41</td>
<td>81.01</td>
<td>71.36</td>
<td>70.38</td>
<td>71.92</td>
<td>69.13</td>
<td>72.35</td>
<td>39.41</td>
<td>54.66</td>
<td>71.73</td>
<td>69.10</td>
</tr>
<tr>
<td>GDA</td>
<td>76.08</td>
<td>70.16</td>
<td>68.05</td>
<td>55.91</td>
<td>65.66</td>
<td>63.76</td>
<td>54.42</td>
<td>69.18</td>
<td>39.32</td>
<td>50.53</td>
<td>64.15</td>
<td>61.57</td>
</tr>
</tbody>
</table>

| ICL    | 81.74 | 91.77 | 90.45 | 23.09  | 72.76  | 64.00 | 37.89 | 73.19     | 36.04 | 55.60 | 68.38 | 63.17 |

Table 2: Experimental results of 11 different datasets on GPT-J in 4-shot / 8-shot per class settings. B, T, and D refers to a baseline, PALP-Template, and PALP-Demonstration respectively. For each dataset, the best method is in bold and the second best method and is underlined.
While ICL displays sound performance in various tasks, the baseline linear probing method exhibits poor performance compared to ICL. In sentiment analysis tasks (SST2, rotten tomatoes), the performance of ICL is above 90% only with 4-shot samples per class, whereas the baseline linear probing methodology almost makes arbitrary random decisions in the same environment (around 50% ~ 60%). However, the performance of ICL quickly saturates and scales poorly with the number of available training samples. And even in some cases, ICL performs worse than arbitrary decisions without understanding the target task at all (e.g., stance, CoLA). Furthermore, if the input length exceeds a certain level, it is infeasible to utilize ICL in the usual way. (See the 16, 32-shot results in the Appendix.) On the other hand, linear probing methods are much more scalable with the number of available samples, revealing stable performance regardless of the dataset. Moreover, their application of PALP boosts performance by a substantial margin minimizing the gap between ICL, especially in most single-sentence tasks. We can obtain around 5% improvement in average from each ablation (appending template and demonstrations) in the 4-shot setting, and some linear probing methods outperform ICL. The most high-performance results were obtained from the SLP among other linear probing methodologies.

### Full-data Results

Table 3, 4 display the performance of different methodologies when the training data is fully available. Appending a simple template also displays a significant advantage even in a data-rich scenario, obtaining considerable improvements in accuracy regardless of the methods or the dataset. Notably, the performance of k-NN increases dramatically with the application of the template (PALP-T), which improved accuracy by 16% on the Emotion dataset.

However, the method of appending demonstration (PALP-D) has more cons than pros in a data-abundant scenario. First, while PALP-D often performs similarly to or better than PALP-T, they also sometimes yield worse scores than PALP-T, leading to a similar performance on average. Speaking otherwise, PALP-D only makes the input more lengthy and entails much higher inference costs compared to PALP-T.
Figure 4: t-SNE visualization of SST2 representation from GPT-J. Adding understandable prompts to the input can reshape the representation into a more task-specialy clustered form without any supervision. Demonstration-best is the representation obtained by attaching the demonstration that showed the best performance, and Demonstration-worst is the opposite.

(a) Baseline  (b) Template  (c) Demonstration-best  (d) Demonstration-worst

<table>
<thead>
<tr>
<th>Acc</th>
<th>Prefix (τ)</th>
<th>Visualization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max (84.86)</td>
<td>Sentence 1: awful movie. \Sentiment: negative</td>
<td>Fig. 4c</td>
</tr>
<tr>
<td></td>
<td>Sentence 1: soulful, scathing and joyous.  \Sentiment: positive</td>
<td></td>
</tr>
<tr>
<td>Min (54.62)</td>
<td>Sentence 1: without any passion. \Sentiment: negative</td>
<td>Fig. 4d</td>
</tr>
<tr>
<td></td>
<td>Sentence 1: , incoherence and sub-sophomoric.  \Sentiment: positive</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Best and worst performing example prefixes from SST2.

to PALP-T in a data-abundant scenario. Moreover, PALP-D is infeasible to be applied to some tasks inheriting the limitations of ICL, as can be seen in Table 4: tasks with a large number of classes (i.e., CLINC, Banking), or tasks with long inputs (i.e., CB, BoolQ).

Nevertheless, PALP greatly minimizes the performance gap between white-box tuning methods, such as Adapter or full fine-tuning, and black-box tuning methods, which is around 7% with baseline linear probing methods, while our approach narrows this gap to nearly 4%. In particular, our method outperforms or reaches statistically equivalent performance to white-box tuning methods in some tasks, such as rotten tomatoes, TREC, and CLINC.

**Analysis & Ablations**

**Application to Small PLM**

In this subsection, we examine our method for relatively small PLM to verify whether our approach is transferrable to small language models. Namely, we report the performance of 3 different tasks (sentiment analysis, natural language inference, and multiclass classification) on GPT-2 large (Radford et al. 2019) in a 4-shot per class setting. Table 6 summarizes the performance. Similar to previous experiments, our method mainly shows considerable performance gain on single sentence tasks and relatively small improvement on challenging tasks like sentence pair tasks. To summarize, PALP also benefits smaller language models, unlike ICL, which is known to yield poor performance or be unable to apply them to relatively small language models. However, the performance improvement is less significant with smaller PLMs since our methodology depends solely on the capability of the language model.

**Dataset Visualization**

In this experiment, we visualize the representation space when the differing input pre-processing method is applied to the input. Figure 4 is the result of the t-SNE visualized representation of the SST2 task on GPT-J. Demonstration-best
is the representation obtained by attaching the demonstration that showed the best performance, and Demonstration-worst is the opposite. Table 5 summarizes actual examples of demonstrations and accuracy of the aforementioned best and worst cases.

Consistent with the experimental results, we confirmed that PLM could extract more distinctive representations when appropriate templates or demonstrations are concatenated to the input of interest. The fact that a more meaningful representation can be drawn by applying a template is understandable and quite intuitive, as research has already shown that using a template can benefit fine-tuning performance (Liu et al. 2021; Schick and Schütze 2021a,b). What is even more intriguing is that adding demonstrations to the front can promote language models to derive a more task-specific and form a more distinguishable representation cluster. While the degree of improvement varies significantly, depending on the selected demonstrations, even concatenating the poorest performing demonstration yields a more clustered representation than the baseline result indicating the language model’s capability to learn from the context of the input (Min et al. 2022b). Although we did not specifically identify which demonstrations were more helpful and which were not, we found that mislabeled demonstrations can hurt the overall quality of the representations. As can be seen in Table 5, the second demonstration of the worst demonstrations is wrongly labeled, where we conjecture is the cause of the poor performance as advocated in a recent study Yoo et al. (2022) that the mapping of input sentence and ground-truth label space can be crucial.

**Related Work**

Large language models such as GPT-3 (Brown et al. 2020) and ERNIE 3.0 (Sun et al. 2021) are often released as black-box APIs due to commercial considerations and the potential risk of misuse. Thus, users are unable to train those pre-trained models with the traditional transferring paradigm (i.e., fine-tuning). Even in some cases where the weights of the pre-trained model are accessible (Zhang et al. 2022; Scao et al. 2022), it may not be possible for many researchers to fine-tune those models due to the enormous resource they require. Several studies were proposed to circumvent the problems as mentioned earlier:

**Black-box Tuning**

Black-box tuning is a methodology that makes use of the target model without internal model access (e.g., gradient, middle layer representations, or weights). As such, black-box tuning methods usually train a lightweight discriminator on top of pre-trained models to optimize input prompt in a derivative-free fashion since the gradient of the pre-trained language models is unavailable. Specifically, Diao et al. (2022); Sun et al. (2022) attempts to find optimal prompt input without utilizes the natural evolution strategy (NES) to find better prompts for black-box models instead of using natural NES to fool the model as in black-box adversarial attacks. Sun et al. (2022) adopted a covariance matrix adaptation evolution strategy to perform optimization in a randomly generated small subspace, which is effective due to the low intrinsic dimensionality of large language models (Aghajanyan, Gupta, and Zettlemoyer 2021; Qin et al. 2021).

**In-Context Learning**

ICL (Brown et al. 2020) is an alternative to the gradient-based tuning method. It is a novel transferring method that derives answers via conditioning appropriate prompts or often concatenating the training data. ICL is drawing explosive interest in the field of NLP due to their strong generalizability among many different tasks, from traditional natural language understanding tasks, including sentiment analysis and natural language inference (Wang et al. 2019), to extreme ones, such as code generations (Poiesia et al. 2022) or mathematical problems (Henighan et al. 2020).

As the underlying mechanism of ICL astonished the NLP field and has reminisced the capability of PLMs, a plethora of works has been proposed to utilize and understand ICL better. Studies include advanced ICL methods maximizing the downstream performance (Zhao et al. 2021; Min et al. 2022a; Holtzman et al. 2021), advanced methods of choosing example data (Liu et al. 2022a; Lu et al. 2022; Rubin, Herzig, and Berant 2022), understanding the limitation of ICL (Liu et al. 2022a; Lu et al. 2022), and understanding the underlying mechanism of ICL (Xie et al. 2022; Reynolds and McDonell 2021; Min et al. 2022b; Razeghi et al. 2022; Yoo et al. 2022).

**Conclusion**

In this paper, we showed that providing task descriptions or demonstrations can enforce PLM to yield more robust representations without additional adaptation of the model weights, allowing them to be used for lightweight linear probing as an alternative to in-context learning. In light of this finding, we proposed prompt-augmented linear probing, where we augmented data representations with ICL-style crafted inputs. Our integrated approach is scalable with the available training data and the size of the language model. PALP obtains comparable results to ICL in the data-hungry scenario and comparable results to fine-tuning in the data-abundant scenario with little training overhead, potentially making PALP a strong alternative in various situations.

In our follow-up study, we will analyze how the additional prompt tokens (e.g., demonstrations or templates) affect the representation quality of the encapsulating input text. We are also interested in the effect of adopting self-supervised learning objectives, such as contrastive learning (Gao, Yao, and Chen 2021), to the shallow layers on top of the language model backbone, which might improve our method further.

**Acknowledgements**

This work was mainly supported by SNU-NAVER Hyper-scale AI Center and partially by the Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korean government (MSIT) No.2020-0-01373, Artificial Intelligence Graduate School
References


Xie, S. M.; Raghunathan, A.; Liang, P.; and Ma, T. 2022. An Explanation of In-context Learning as Implicit Bayesian Inference. In The Tenth International Conference on Learning Representations, ICLR.
