Differentiable Prompt Learning for Vision Language Models

Zhenhan Huang[1] , **Tejaswini Pedapati**[2] , **Pin-Yu Chen**[2] and **Jianxi Gao**[1]
[1]Rensselaer Polytechnic Institute
[2]IBM Research
huangz12@rpi.edu, tejaswinip@us.ibm.com, Pin-Yu.Chen@ibm.com, gaoj8@rpi.edu

Abstract

Prompt learning is an effective way to exploit the potential of large-scale pre-trained foundational models. Continuous prompts parameterize context tokens in prompts by turning them into differentiable vectors. Deep continuous prompts insert prompts not only in the input but also in the intermediate hidden representations. Manually designed deep continuous prompts exhibit a remarkable improvement compared to the zero-shot pre-trained model on downstream tasks. How to automate the continuous prompt design is an underexplored area, and a fundamental question arises, *is manually designed deep prompt strategy optimal?* To answer this question, we propose a method dubbed differentiable prompt learning (DPL). The DPL method is formulated as an optimization problem to automatically determine the optimal context length of the prompt to be added to each layer, where the objective is to maximize the performance. We test the DPL method on the pre-trained CLIP. We empirically find that by using only limited data, our DPL method can find deep continuous prompt configuration with high confidence. The performance on the downstream tasks exhibits the superiority of the automatic design: our method boosts the average test accuracy by **2.60 %** on 11 datasets compared to baseline methods. Besides, our method focuses only on the prompt configuration (*i.e.* context length for each layer), which means that our method is compatible with the baseline methods that have sophisticated designs to boost the performance. We release our code in https://github.com/Zhenhan-Huang/Differentiable-Prompt-Learn.

1 Introduction

Parameter-Efficient fine-tuning (PEFT) offers a paradigm to adapt pretrained model to downstream tasks using a small set of trainable parameters. Among the PEFT methods, the prompt learning can perform few-shot learning or even zero-shot learning to adapt the pre-trained models to new scenarios. Using appropriate prompts, both text prompts and visual prompts, to query foundational models have shown great potential. At the same time, the selection of context tokens in prompts has a pronounced effect on the performance of the prompt learning methods [Zhou *et al.*, 2022b; Schick and Schütze, 2021]. Continuous prompts use differentiable vectors that are in the embedding space of the foundation model. Continuous prompts use trainable parameters that are learned during the fine-tuning process. It avoids the time-consuming trial-and-error process using discrete context tokens. Besides, the continuous prompts are not constrained to be the embeddings of word tokens in the vocabulary. The selection for the embeddings of context tokens becomes continuous.

Deep continuous prompts add prompts to multiple layers. This method has shown superior performance compared to only adding continuous prompts to the input in both vision and language fields. There are two hyperparameters: the context length of continuous prompts c_p and prompt depth ℓ_p. Let the hidden dimension be d, continuous prompts $\mathbf{E} \in \mathbb{R}^{c_p \times d}$ are inserted to the input to the neural network layers up to ℓ_p layers. Since $\mathbf{E}$ are automatically determined during the fine-tuning process, a natural question is: *can we automatically determine c_p and ℓ_p?* Recently, it has been found that training merely part of deep neural network layers can achieve a performance comparable to or even better than training all layers. This line of work indicates that there is a subset of layers in the pre-trained model, depending on the distribution shift between the pre-training dataset and fine-tuning dataset, whose parameters might be close to a minima for downstream tasks [Lee *et al.*, 2022; Lodha *et al.*, 2023].

In the prompt learning, we postulate that there might be some layers that do not need deep continuous prompts or only need continuous prompts with shorter context length compared to rest of layers. In the existing prompt learning works, however, a fixed number c_p is used for each layer. To examine our postulation, we design our method to insert continuous prompts with different context lengths. The context lengths of continuous prompts are determined using a differentiable formulation. If a layer has the best context length found to be 0, we do not add continuous prompts to this layer. We name the proposed method *differentiable prompt learning* (DPL). The few-shot learning experiments show that the DPL method can find continuous prompt configurations, i.e., the context length and depth of continuous prompts inserted to the input of each layer. The performance of downstream fine-tuning over 11 datasets shows the superiority of the proposed method.

In summary, our DPL method has the following contributions:

- The DPL method automates the continuous prompt learning process. It automatically determines the continuous prompts and associated hyperparameters including the context length of the continuous prompt inserted for each neural network layer and prompt depth.
- The DPL method removes the constraint in the manually designed continuous prompt methods that the context length for each layer is fixed. The heterogeneous design of inserting continuous prompts enables a more flexible design for the prompt learning. The method is simple and focuses only on continuous prompts, which means it can be combined with sophisticated designs in existing prompt learning methods.
- We find the optimal deep continuous prompt configuration is dataset-dependent. By tailoring deep continuous prompts for each dataset, the DPL method can achieve the performance better than manually designed prompt learning methods.

2 Background and Related Work

Prompt Learning. Prompt learning offers an efficient way to adapt pre-trained foundational models to downstream tasks. This technique is of great interest in both vision and natural language. CoOp [Zhou *et al.*, 2022b] inserts continuous prompts to the input of the vision-language model. The visual prompt tuning [Jia *et al.*, 2022] proposes the deep continuous prompts to vision transformer [Dosovitskiy *et al.*, 2020]. MaPLe [Khattak *et al.*, 2023] combines the ideas from these two works by using deep continuous prompts in both the text branch and the image branch. Similar to CoOp, CoCoOp, PLOT and ProGrad insert continuous prompts only in the input. CoCoOp [Zhou *et al.*, 2022a] inserts the continuous prompts conditioning on the input images. PLOT [Chen *et al.*, 2022] uses the computationally costly iterative algorithm to compute the transport plan for aligning word tokens and image patches. ProGrad [Zhu *et al.*, 2023] uses the gradient-aligned knowledge distillation to avoid the overfitting problem. All of the above-mentioned methods use fixed context length as opposed to our proposed DPL method which adapts the context length for each layer dynamically based on the dataset.

Neural Architecture Search. Neural architecture search (NAS) automatically determines the architecture by minimizing the objective function. One-shot NAS is an efficient way to search neural architectures. In the one-shot NAS, a supernet [Saxena and Verbeek, 2016; Pham *et al.*, 2018; Liu *et al.*, 2018] is introduced comprising all possible architecture in the search space. After training the supernet, a subnet is uniquely determined as the neural architecture determined by NAS.

2.1 Revisiting Differentiable NAS

Differentiable NAS [Liu *et al.*, 2018; Yan *et al.*, 2021] uses a cell-based search space. A cell is represented by a directed acyclic graph (DAG) $\mathcal{G}(\mathcal{V}, \mathcal{E})$. Each node is a hidden representation and each directed edge is an operation transforming the hidden representation. The supernet incorporates all the operations. Let $\mathcal{O}$ be a set of candidate operations. The categorical choice of a particular operation is relaxed by the softmax over all possible operations:

$$\bar{o}^{(i,j)}(\mathbf{x}) = \sum_{o \in \mathcal{O}} \frac{\exp(\alpha_o^{(i,j)})}{\sum_{o'} \exp(\alpha_{o'}^{(i,j)})} o(\mathbf{x}) \ , \tag{1}$$

where $\alpha^{(i,j)}$ are trainable parameters that determine the searched operation between the latent representation i and j. The goal of the search process is to replace $\bar{o}^{(i,j)}$ with the most likely operation. The optimal neural architecture is obtained after the search process.

Inspired by differentiable NAS methods, we relax the categorical selection on the context lengths of continuous prompts to make the search space continuous. We use the search space to determine the optimal prompt configuration. The continuous prompts are trained from scratch. Existing vision-language prompt learning methods use the deep prompt method [Jia *et al.*, 2022] where continuous prompts are added to transformer blocks and removed after the self-attention [Vaswani *et al.*, 2017]:

$$[\mathbf{x}^{(l)}, \mathbf{E}^{(l)}] = f^{(l)}([\mathbf{x}^{(l-1)}, \mathbf{E}^{(l-1)}]) \ . \tag{2}$$

Here we denote l-th transformer block as $f^{(l)}$. Different options for adding continuous prompts cannot be mixed as the dimension of the input $[\mathbf{x}^{(l-1)}, \mathbf{E}^{(l-1)}]$ is different due to different context lengths. We solve this problem by using cross-attention. Details are reported in the following section.

3 Differentiable Prompt Learning

We use the pre-trained CLIP model [Radford *et al.*, 2021] in our experiments. The CLIP model is pre-trained on over 400 million image-text pairs. The DPL method has two stages: the searching stage and the training stage. The goal of the searching stage is to determine the context length of the continuous prompt to be added to each transformer block of the CLIP model. The best prompt configuration is used in the training stage. Figure 1 shows the proposed method.

3.1 Searching Stage

We prepend continuous prompts $\{\mathbf{E}^{(l)}\}$ to inputs to transformer blocks $\{\mathbf{x}^{(l)}\}$ in the text branch and image branch, where $1 \leq l \leq \ell, l \in \mathbb{N}^+$. The concatenated inputs are fed into transformer blocks $f(\cdot)$. *w.l.o.g*, for the l-th layer, there are t options for adding continuous prompts of different context length $\mathbf{E}_i^{(l)} \in \mathbb{R}^{c_i \times d}$, where $1 \leq i \leq t, i \in \mathbb{N}^+$, to the input $\mathbf{x}^{(l)} \in \mathbb{R}^{c_l \times d}$. c_i is the context length for i-th option, d is the hidden dimension. For the text branch, the hidden dimension is $d = d_{\text{txt}}$. For the image branch, the hidden dimension is $d = d_{\text{img}}$. Continuous prompts in the text branch

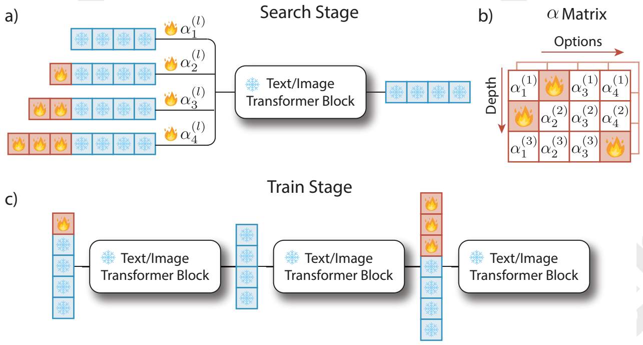

a)
Search Stage
$\alpha_1^{(l)}$
$\alpha_2^{(l)}$
$\alpha_3^{(l)}$
$\alpha_4^{(l)}$
Text/Image Transformer Block
b)
α Matrix
Options
Depth
$\alpha_1^{(1)}$ $\alpha_3^{(1)}$ $\alpha_4^{(1)}$
$\alpha_2^{(2)}$ $\alpha_3^{(2)}$ $\alpha_4^{(2)}$
$\alpha_1^{(3)}$ $\alpha_2^{(3)}$ $\alpha_3^{(3)}$
c)
Train Stage
Text/Image Transformer Block
Text/Image Transformer Block
Text/Image Transformer Block

Figure 1: (a) In the searching stage, continuous prompts with different context lengths (shown in red color) are added to the original context tokens (shown in blue color) as the input to transformer blocks in the text branch or image branch. The outputs of transformer blocks are used as the original context tokens for the next transformer blocks. $\{\alpha_i^{(l)}\}$ are differentiable parameters to control the contribution of different prompt options. (b) After the searching stage, two α matrices are obtained to indicate the selection of the search algorithm for differentiable context tokens in the language branch and the image branch. (c) In the training stage, prompt learning is conducted using the differentiable context token setting determined by the search algorithm.

are independent of those in the image branch. The output $\mathbf{h}^{(l)}$ of the transformer block is:

$$\mathbf{h}_i^{(l)} = \begin{cases} f^{(l)}([\mathbf{x}^{(l-1)}])\ , & i=1 \\ f^{(l)}([\mathbf{E}_i^{(l)}, \mathbf{x}^{(l-1)}])\ . & 1 < i \le t \end{cases} \tag{3}$$

$i = 1$ accounts for the case where no continuous prompt is added. Different from the traditional transformer [Vaswani *et al.*, 2017] where self-attention mechanism is applied, we use the cross-attention mechanism in transformer blocks:

$$\mathbf{Q} = \mathbf{w}_q^T \mathbf{x},\ \mathbf{K} = \mathbf{w}_k^T [\mathbf{E}_i^{(l)}, \mathbf{x}^{(l-1)}],\ \mathbf{V} = \mathbf{w}_v^T [\mathbf{E}_i^{(l)}, \mathbf{x}^{(l-1)}]\ . \tag{4}$$

$$\text{Cross Attention} = \text{Softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}})\mathbf{V},$$
$$\text{where } \mathbf{Q} \in \mathbb{R}^{c_l \times d},\ \mathbf{K} \in \mathbb{R}^{(c_i+c_l)\times d},\ \mathbf{V} \in \mathbb{R}^{(c_i+c_l)\times d}\ . \tag{5}$$

We use parameters $\{\beta_i^{(l)}\}_{i=1}^t$ to control the contribution of different options to the output. In other words, the mixing weight for the hidden representation $\mathbf{h}^{(l)}$ is parameterized by $\beta_i^{(l)}$:

$$\mathbf{x}^{(l)} = \sum_{i=1}^{t} \beta_i^{(l)} \mathbf{h}_i^{(l)}\ . \tag{6}$$

The Equation 6 indicates that the output of each transformer block is a combination of all possible context lengths, *i.e.* $[0, t]$. $\{\beta_i^{(l)}\}_{i=1}^t$ is computed by differentiable parameters $\{\alpha_i^{(l)}\}_{i=1}^t$:

$$\beta_i^{(l)} = \frac{\exp(\alpha_i^{(l)})}{\sum_{m=1}^t \exp(\alpha_m^{(l)})}\ . \tag{7}$$

Figure 1 (a) shows the illustration of the searching stage. Same as conventional prompt learning methods, the parameters of the transformer blocks of the pre-trained model are frozen. During the searching process, updated parameters are continuous prompts and α parameters. The goal of updating continuous prompts is to minimize the training loss $\mathcal{L}_{\text{train}}$ while that of updating α parameters is to minimize the validation loss $\mathcal{L}_{\text{val}}$. This implies that the searching stage is a bilevel optimization problem [Anandalingam and Friesz, 1992; Colson *et al.*, 2007] with the upper level parameter α and the lower level parameter $\mathbf{E}$:

$$\min_{\alpha} \quad \mathcal{L}_{\text{val}}(\mathbf{E}^*(\alpha), \alpha) \tag{8}$$
$$\text{s.t.} \quad \mathbf{E}^*(\alpha) = \underset{\mathbf{E}}{\text{argmin}}\ \mathcal{L}_{\text{train}}(\mathbf{E}, \alpha). \tag{9}$$

Algorithm 1 shows the searching stage. The input is the pre-trained vision-language model and two α matrices (we call them search spaces). Two α matrices are randomly initialized before training. After convergence, the α matrix $\mathbf{A}^\alpha \in \mathbb{R}^{\ell \times t}$ is obtained as shown in Figure 1 (b). The row dimension is associated with the depth and the column dimension is related to t options. The column index of the highest A_{im}^α $(1 \le m \le t, m \in \mathbb{N}^+)$ for the i-th transformer block determines the context length of the continuous prompts added to that block.

Algorithm 1 Searching stage for vision-language models

```
Input: A pre-trained model and two $\alpha$ matrices $\mathbf{A}^\alpha \in \mathbb{R}^{\ell \times t}$ with randomly initialized weights.
while not converged do
    Update $\mathbf{A}^\alpha$ by descending $\nabla_{\mathbf{A}^\alpha} \mathcal{L}_{\text{val}}(\mathbf{E}, \mathbf{A}^\alpha)$.
    Update continuous prompts in both text branch and image branch by descending $\nabla_{\mathbf{E}} \mathcal{L}_{\text{train}}(\mathbf{E}, \mathbf{A}^\alpha)$.
end while
for $i = 1$ to $\ell$ do
    $A_{ik}^\alpha = \max_m A_{im}^\alpha$, $k$ determines the context length of continuous prompts for the $i$-th block in the best prompt configuration.
end for
Output: Prompt configuration for the image branch and the text branch.
```

We use the term *supprompt* to represent the combination of the continuous prompts added to the pre-trained model and an α matrix in the searching stage. For vision-language models, there are two sets of supprompts for the image and text branches. After the search, the best context length is identified for each layer and the resulting model is then trained from scratch. We use the term *subprompt* to denote the final best context length for each layer that is used in the final model. Terminologies are summarized in Appendix A.2.

Similar to the differentiable NAS where the supernet incorporates all candidate operations, our method creates a supprompt that contains all prompt configurations in the search space. Incorporation of all possibilities inevitably leads to a relatively large supprompt compared to the subprompt, which can increase the computational cost in the searching stage. After the optimal prompt configuration is determined, the computational complexity is small compared to the deep continuous prompt method such as MaPLe. The complete comparison is reported in the Appendix A.3.

Figure 2 shows the comparison between our method and prevalent prompting methods for vision-language models. After transformer blocks, the traditional methods have varying context lengths that depend on the number of inserted context tokens. Our method results in a fixed context length.

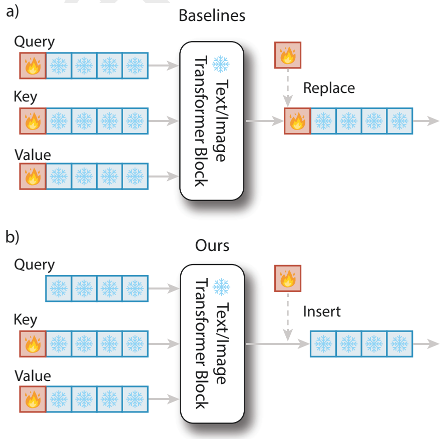


Figure 2: Illustration of the difference in inserting continuous prompts between our method and commonly used prompting methods for vision-language models.

3.2 Training Stage

We denote $\mathbf{A}^{\alpha}_{\text{txt}}$ for the α matrix of the text branch and $\mathbf{A}^{\alpha}_{\text{img}}$ for that of the image branch. The column index of the maximum value in each row of α matrix determines the context length of the inserted continuous prompts, i.e. $A^{\alpha}_{ik} = \max_m A^{\alpha}_{im}$. After the context length of continuous prompts is determined, we fine-tune the model on various downstream datasets. Figure 1 (c) shows the training stage. The fine-tuning process is the same as the conventional prompt learning method: model parameters are frozen and only continuous prompts are differentiable.

In the image branch, the input $\mathbf{x} \in \mathbb{R}^{C\times H\times W}$ is patchified and projected to produce patch token embeddings [Dosovitskiy *et al.*, 2020]. A learnable classification token embedding $\mathbf{u}_{\text{cls}} \in \mathbb{R}^{1\times d_{\text{txt}}}$ is added to the patch token embedding $\tilde{\mathbf{x}}_{\text{img}} = [\mathbf{u}_{\text{cls}}, \mathbf{u}_1, \ldots, \mathbf{u}_{n_{\text{img}}}]$. In the text branch, a text template such as `a photo of a [class]` is tokenized and embedded to generate text token embeddings that are formulated as $\tilde{\mathbf{x}}_{\text{txt}} = [\mathbf{v}_{\text{SOS}}, \mathbf{v}_1, \ldots, \mathbf{v}_{n_{\text{txt}}}, \mathbf{v}_k, \mathbf{v}_{\text{EOS}}]$, where $\mathbf{v}_{\text{SOS}}$ is the start token embedding, $\mathbf{v}_{\text{EOS}}$ is the end token embedding, and $\mathbf{v}_k$ is the embedding corresponding for the class name. $\tilde{\mathbf{x}}_{\text{txt}}$ and $\tilde{\mathbf{x}}_{\text{img}}$ are combined with continuous prompts and fed to transformer blocks in the text branch and image branch for obtaining text embedding $\tilde{\mathbf{g}}$ and image embedding $\tilde{\mathbf{f}}$, respective. Within each transformer block, we use the cross attention mechanism as shown in Equation 4 and 5. The output logits are based on the similarity between $\tilde{\mathbf{g}}$ and $\tilde{\mathbf{f}}$:

$$\hat{\mathbf{y}} = \frac{\exp(\text{sim}(\tilde{\mathbf{f}}, \tilde{\mathbf{g}})/\tau)}{\sum_{c\in\mathcal{C}} \exp(\text{sim}(\tilde{\mathbf{f}}, \tilde{\mathbf{g}}_c)/\tau)} , \tag{10}$$

where τ is the temperature parameter. The loss $\mathcal{L}_{\text{de}}$ computes the deviation of the prediction from the ground truth labels:

$$\mathcal{L}_{\text{de}} = \mathbb{E}_{(\mathbf{x},y)\sim\mathcal{D}}\ \mathcal{F}(\hat{\mathbf{y}}, y) , \tag{11}$$

where $\mathcal{F}(\cdot)$ is the loss function. We use the cross entropy loss for calculating $\mathcal{L}_{\text{de}}$. Continuous prompts are updated by descending the training loss $\nabla_{\mathbf{E}}\mathcal{L}_{\text{train}}(\mathbf{E})$.

In addition to the vanilla DPL, we add knowledge distillation in the training stage. Specifically, we use Kullback-Leibler (KL) divergence [Hinton *et al.*, 2015] between the model prediction $\hat{\mathbf{y}} = p(\mathbf{x})$ and the zero-shot pre-trained model prediction $p_{\text{zs}}(x)$:

$$\mathcal{L}_{\text{kl}} = -\sum_{\mathbf{x}\in\mathcal{X}} p_{\text{zs}}(\mathbf{x}) \log\left(\frac{p(\mathbf{x})}{p_{\text{zs}}(\mathbf{x})}\right) . \tag{12}$$

The total loss $\mathcal{L}_{\text{total}}$ is computed by:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{de}} + \lambda\mathcal{L}_{\text{kl}} , \tag{13}$$

where λ is a hypereparameter. The gradient descent of continuous prompts uses $\mathcal{L}_{\text{total}}$ to update parameters.

4 Experiments

4.1 Datasets and Experiment Setup

Datasets. We evaluate the DPL method on 11 datasets: Caltech101 [Fei-Fei *et al.*, 2004] and ImageNet [Deng *et al.*, 2009] for the generic object classification, DescribableTectures [Cimpoi *et al.*, 2014] for the texture classification, EuroSAT [Helber *et al.*, 2019] for the satellite image classification, FGVCAircraft [Maji *et al.*, 2013], Food101 [Bossard *et al.*, 2014], OxfordFlowers [Nilsback and Zisserman, 2008], OxfordPets [Parkhi *et al.*, 2012], and StanfordCars [Krause *et al.*, 2013] for the fine-grained image recognition, UCF101 [Soomro *et al.*, 2012] for the action classification, and SUN397 [Xiao *et al.*, 2010] for the scene recognition.

Baselines. We compare the proposed method with CoCoOp [Zhou *et al.*, 2022a], PLOT [Chen *et al.*, 2022], MaPLe [Khattak *et al.*, 2023] and ProGrad [Zhu *et al.*, 2023]. The original implementation of PLOT uses ResNet [He *et al.*, 2016] as the backbone model. For a fair comparison, we replace the backbone model of ResNet with the transformer model. In addition to the prompt learning methods, we examine the performance of the zero-shot CLIP (ZS CLIP) and that of training a linear classifier given image and text representations by CLIP (Linear probe) [Radford *et al.*, 2021].

Experiment Details. We use the pretrained ViT-B/16 CLIP model [Radford *et al.*, 2021] in the few-shot learning. In both training stage and searching stage, we use the same hyperparameters except for the number of epochs. The number of epochs in the searching stage is 60 while that for the training stage is 40. The batch size is 4 and we use stochastic gradient descent (SGD) to optimize continuous prompts. In the searching stage, two α matrices are optimized using SGD strategy. Learning rate is 3.5×10^{-3}. Experiments are conducted using a single NVIDIA A40 GPU.

4.2 Determining Context Lengths of Continuous Prompts

We use the few-shot learning setting in the searching stage. The number of shots is the same for the searching and training stages. The number of shots is 16. The results of using 8/4/2/1 shots are shown in Appendix A.5. The evolution of α matrices is visualized to examine the convergence process. The evolution of α matrices using 11 different datasets is reported in the Appendix A.1. We define special α matrices formally:

Definition 4.1 (single-dominant)**.** *An α matrix $\mathbf{A}^\alpha \in \mathbb{R}^{\ell \times t}$ is single-dominant if $\forall i \in \mathbb{N}^+, 1 \leq i \leq \ell$, $\exists j \in \mathbb{N}^+, 1 \leq j \leq t$ s.t. $A^\alpha_{ij} \gg A^\alpha_{ik}$, where $k \in \mathbb{N}^+, 1 \leq k \leq t, k \neq j$.*

The single-dominant α matrix means that the searching algorithm has a high confidence in the searched subprompts. This is beneficial to the training stage as the determined α matrix is robust against the fluctuation of the matrix in the updating process. We use the alpha value at each row with the largest value (i.e. $\operatorname{argmax}_j \alpha_{ij}$) to determine the context length of continuous prompts added in the training stage. Hence, the training stage does not require the α matrix to be single-dominant. When α matrix is not single-dominant, the decision on adding continuous prompts is not robust. For example, in the case where two α values are close:

$$\exists \epsilon \quad \text{s.t. } \epsilon > 0,\ \epsilon \ll |A^\alpha_{im}|,\ \epsilon \ll |A^\alpha_{in}|,\ |A^\alpha_{im} - A^\alpha_{in}| < \epsilon\ , \tag{14}$$

the fluctuation can change the order in the comparison of α values. At the epoch of 60, if we use a threshold value to filter out small α values, α matrices become essentially sparse matrices.

Figure 3 (a) and (b) shows α matrices at the epoch of 60 for all 11 datasets. Compared to the image branch, there are considerably more single-dominant α matrices in the text branch. Although we use 16-shots learning, the searching results indicate that the searching algorithm has pretty high confidence for the searched prompt configuration. α matrices at the last epoch are used in the training stage for the prompt learning. To quantify the evolution process of α matrices, we firstly define α difference:

$$\begin{aligned} \alpha \text{ Difference} &:= \sum_i \sum_j |\text{Softmax}(A^\alpha_{ij}) - \text{Softmax}(A^\alpha_{ik})|, \\ &\text{where } A^\alpha_{ik} = \max_m (A^\alpha_{im})\ . \end{aligned} \tag{15}$$

Apparently, single-dominant α matrix has a high α difference value. Figure 3 (c) shows the variation of the α difference. At the early training stage, the α difference increases drastically. there is a fluctuation when the α difference is close to the converged value. The fluctuation comes from two parts: the first part is related to the variation of the largest α value in each row. Even though the α matrix is single dominant, the largest α value is subjected to the variation in the gradient descent. Owing to the softmax operation, the contribution of the largest α value and that of the rest α values in a row is not independent, hence we call this *intra factor*. The second part is the fluctuation of α values except the largest ones. We call it *inter factor*.

The number of dominants describes the number of pairs between the largest α values and the rest of α values in the same row:

$$\begin{aligned} \#\ \text{Dominants} := \big|\{(A^\alpha_{ij}, A^\alpha_{ik}) \mid 1 \leq i \leq \ell, 1 \leq j \leq t, \\ j \neq k, A^\alpha_{ik} = \max_m (A^\alpha_{im}), A^\alpha_{ik} \gg A^\alpha_{ij}\}\big| \end{aligned} \tag{16}$$

The ideal converged α matrix has the number of dominants $\#$ Dominants $= \ell(t-1)$.

4.3 Prompt Learning Based on Alpha Matrix

After determining the context length of added continuous prompts, we add them to the pre-trained CLIP model and conduct the prompt learning in a supervised fashion. We use the few-shot learning and the number of shots is 16. The α matrix in the text branch might be different from that in the image branch as shown in Appendix Figure 5, and there is no coupling function [Khattak *et al.*, 2023; Zang *et al.*, 2022] between added continuous prompts in the two branches. Table 1 shows the performance comparison on downstream datasets. Note that the direct comparison of zero-sho CLIP with other methods is not fair as it does not require any training.

Our proposed method shows a pronounced advantage compared to baseline methods. It boosts the average accuracy by 2.60% test accuracy. The proposed method shows unfavorable performance on the Food101 dataset. When comparing the zero-shot CLIP method with the linear probe method which adds a linear classifier on top of pre-trained CLIP model, there is a remarkable performance drop ($77.30\% \rightarrow 70.13\%$). We postulate that the reason might be related to the forgetting issue [Chen *et al.*, 2019; Boschini *et al.*, 2022] in the fine-tuning process. ProGrad is designed to avoid the forgetting issue by aligning the gradient descent direction of matching predictions and ground-truth labels to that of knowledge distillation when there is a conflict. ProGrad exhibits a good performance on that dataset.

We find that the performance difference between the DPL method and baseline methods is largest on the EuroSAT and Aircraft datasets. There is a large distribution shift on EuroSAT and Aircraft datasets compared to the pre-train datasets of the CLIP model. On the generic dataset such as Caltech101, the distribution shift is small. Hence, the difference between the DPL method and baseline methods is minimal.

We visualize the attention map on the downstream datasets using different prompt learning methods as shown in Figure 4. The Grad-CAM visualization [Selvaraju *et al.*, 2017] indicates that our method is beneficial to focus on key elements in the foreground. In the image of amphibious aircraft, the attention of the model using the DPL method is localized on the float and the window. The model's attention focuses on the vehicle's logo in the car image. In the action classification, the model focuses on riders and horses. Those elements are important to differentiate the target class from the candidate classes. The PLOT method [Chen *et al.*, 2022] is designed to align the prompts with image features using the optimal transport theory [Peyré *et al.*, 2019]. It also exhibits a good

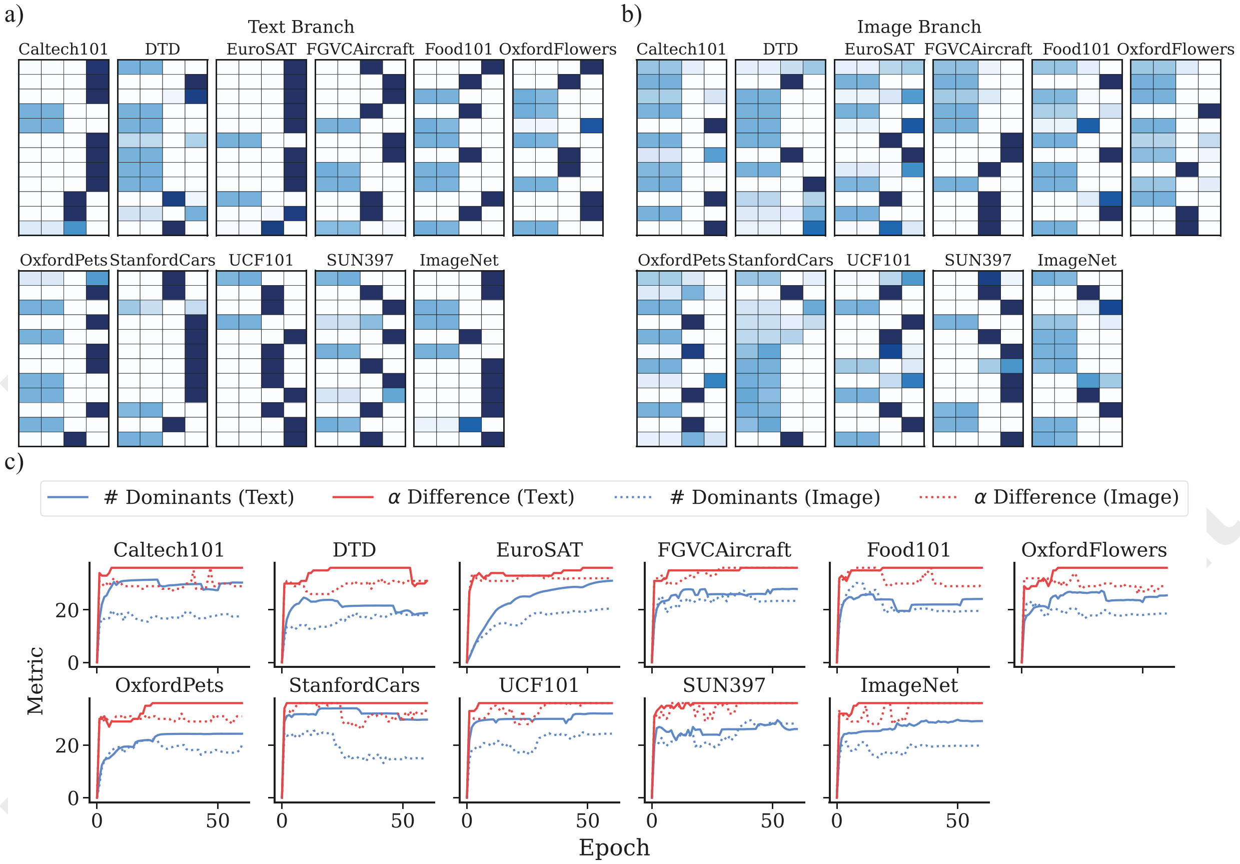


Figure 3: (a) α matrices for the text branch. (b) α matrices for the image branch. α matrices are obtained at the epoch of 60. The row dimension is related to the context length of added continuous prompts. The column dimension is related to model depth, i.e. the number of transformer blocks. (c) The evolution of the α difference and the number of dominants. As the number of training epochs increases, the α matrix gradually converges and the number of dominants increases to the converged value.

alignment. The DPL method only relies on the prompt configuration, which means it is compatible with using the optimal transport theory.

4.4 Ablation Studies

We examine the performance of the shallow prompt learning method: continuous prompts with a context length of 16 are added to the inputs of the text branch and the image branch respectively. The equivalent prompt depth is 1. The way of adding context prompts is illustrated in Equation 4 and Equation 5. The result is shown in the Table 2. Compared to the DPL method, the shallow prompt learning method has a remarkable performance drop. There is the largest performance degradation in the EuroSAT and Aircraft datasets. These two datasets also exhibit the advantage of the DPL method over baseline methods shown in the Table 1.

5 Discussion

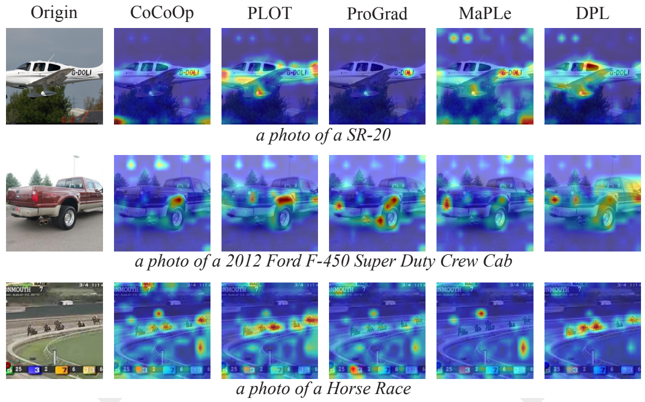
By using bi-level optimization, the DPL method is able to automatically determine the context length of the continuous prompt for each layer. It seems to indicate that the optimal prompt configuration might be dataset-dependent. The deep neural networks are found to be brittle to even small distribution shits between the pre-training and fine-tuning datasets [Recht *et al.*, 2019; Hendrycks and Dietterich, 2019]. A dataset-dependent training scheme provides the flexibility to the level of distribution shift. Our results show that the few-shot learning with DPL can boost the downstream accuracy.

Existing deep prompt learning works hardly consider a granular level of adding continuous prompts: each layer has the same context length and a hyperarameter of prompt depth is empirically determined. A prompt depth ℓ_p smaller than the model depth ℓ indicates that the last few layers of the pre-trained model might be close to a minima for the downstream dataset. In the transfer learning, not all layer parameters are responsible for the distribution shift. There are numerous researches on training different layers differently to mitigate the forgetting issue [Niu *et al.*, 2022; Toneva *et al.*, 2018] caused by the distribution shift: fine-tuning on the target domain using gradual unfreezing [Howard and Ruder, 2018], using different learning rate for different layers [Ro and Choi, 2021; Shen *et al.*, 2021], and training part of layers [Lee *et al.*, 2022; Vettoruzzo *et al.*, 2024].

If some layer parameters are close to optimal in downstream datasets, we believe there is no need to add continu-

Method	Test Accuracy ↑											
	Caltech101	DTD	EuroSAT	Aircraft	Food101	Flowers	Pets	Cars	UCF	SUN397	ImageNet	Average
ZS CLIP	87.20	42.34	37.57	17.29	77.30	66.18	85.79	55.63	61.45	58.73	58.77	58.93
Linear Probe	90.44	64.02	82.68	36.36	70.13	95.01	76.36	70.13	73.70	65.17	54.33	70.76
CoCoOp	95.10	63.63	74.10	33.67	87.37	89.97	93.53	72.30	76.97	72.67	71.17	75.50
PLOT	93.70	70.90	84.03	34.93	78.13	**97.27**	88.20	68.10	72.23	69.40	72.23	75.37
ProGrad	95.63	66.27	82.03	41.30	86.70	95.33	93.10	81.23	81.60	75.13	72.27	79.14
MaPLe	95.10	67.27	86.40	37.07	**87.43**	94.27	**93.63**	74.87	80.37	74.73	72.03	78.47
DPL	**95.80**	**71.50**	92.30	48.57	82.57	96.63	92.03	82.70	**84.10**	**79.83**	72.80	81.71
DPL + KD	95.73	70.90	**92.47**	**49.43**	82.80	96.80	91.83	**82.83**	83.77	79.63	**73.03**	**81.74**

Table 1: Test accuracies of DPL and baseline methods on 11 datasets using few-shot learning. Results are averaged over 3 runs. The DPL method shows a pronounced advantage compared to baseline methods. Combining the DPL method with knowledge distillation can further boost the performance.



a photo of a SR-20

a photo of a 2012 Ford F-450 Super Duty Crew Cab

a photo of a Horse Race

Figure 4: Original image and Grad-CAM visualization [Selvaraju *et al.*, 2017] for various methods on FGVCAircraft, StanfordCars and UCF101 datasets. Our method helps the pre-trained model focus on key elements in the foreground and avoid distraction from the background. The text template `a photo of a [class]` is used in the Grad-CAM calculation.

ous prompts to those layers. The deep prompt methods using $\ell_p < \ell$ avoid adding prompts to layers that are close to the optimal. If we refine this strategy, a natural question is *do all ℓ layers require the same context length*? Different layers might have a different level of deviation from the minima. The extreme case is that a layer is very close to the minima and no continuous prompts are needed, i.e. the context length is 0 in this layer. For layers with different levels of deviation, we might need to use different context lengths. This is consistent with the result of using the DPL method. Overall, the result reveals two things: (1) the optimal context length depends on the distribution shift. (2) different layers of the pre-trained model might require different context lengths.

6 Limitation

The searching stage of the DPL method, similar to NAS, has higher computational costs than conventional soft prompting methods due to the introduction of the supprompt. The computational cost is determined by the size of the search space. At each depth, DPL searches context lengths of $\{0, 2, 4, 6\}$. The size of the search space is 2.81×10^{15}. Decreasing the size of the search space can reduce computational costs. Using differentiable approach greatly accelerates the searching process compared to exhaustive search, *i.e.* $t_a \times |\mathcal{A}| \gg T_{\text{dpl}}$, where t_a is the training time for prompting methods and T_{dpl} is the total running time (training state + searching stage) for the DPL method. The training stage of the DPL method is lightweight as it relies completely on prompt configuration without advanced technologies such as coupling functions. Details regarding to the computational costs are discussed in Appendix A.3.

Method	Caltech101	DTD	EuroSAT	Aircraft	Food101	Flowers	Pets	Cars	UCF	SUN397	ImageNet	Average
Shallow Prompting	-0.37	-3.13	-7.27	-12.14	-0.80	-1.06	-1.16	-3.27	-3.03	-3.40	-1.73	-3.40

Table 2: Ablation study on prompt configuration, i.e. the context length and the depth of continuous prompts. The shallow prompting uses a prompt depth of 1 and the context length of 16. Test accuracy variation is reported compared to the optimal prompt configuration found by the DPL method. A negative value indicates the performance drop when using the shallow prompting method.

7 Conclusion

In this work, we automate the prompt learning design by relaxing the categorical selection of context lengths to obtain a continuous search space. Using limited data, our method is able to find a prompt learning setting that is better than the existing manually designed prompt learning methods. The method is simple yet effective: it only focuses on determine context lengths of continuous prompts. We empirically find that our searching algorithm has a higher confidence in the text branch compared to that in the image branch, and that the prompt configuration shows a data-dependent behavior. The data-dependent behavior and the asymmetric prompt insertion in two branches demonstrate the strength of the automatic prompt learning design. In summary, our work proposes a new paradigm of adding continuous prompts that removes the restriction of the manually designed context length fixed prompt learning method.

Acknowledgements

This work was supported by the USA National Science Foundation under grant #2047488, and IBM-RPI Future of Computing Research Collaboration.

References

[Anandalingam and Friesz, 1992] Gnana Anandalingam and Terry L Friesz. Hierarchical optimization: An introduction. *Annals of Operations Research*, 34:1–11, 1992.

[Boschini *et al.*, 2022] Matteo Boschini, Lorenzo Bonicelli, Angelo Porrello, Giovanni Bellitto, Matteo Pennisi, Simone Palazzo, Concetto Spampinato, and Simone Calderara. Transfer without forgetting. In *European Conference on Computer Vision*, pages 692–709. Springer, 2022.

[Bossard *et al.*, 2014] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer vision–ECCV 2014: 13th European conference, zurich, Switzerland, September 6-12, 2014, proceedings, part VI 13*, pages 446–461. Springer, 2014.

[Chen *et al.*, 2019] Xinyang Chen, Sinan Wang, Bo Fu, Mingsheng Long, and Jianmin Wang. Catastrophic forgetting meets negative transfer: Batch spectral shrinkage for safe transfer learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[Chen *et al.*, 2022] Guangyi Chen, Weiran Yao, Xiangchen Song, Xinyue Li, Yongming Rao, and Kun Zhang. Plot: Prompt learning with optimal transport for vision-language models. *arXiv preprint arXiv:2210.01253*, 2022.

[Cimpoi *et al.*, 2014] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[Colson *et al.*, 2007] Benoît Colson, Patrice Marcotte, and Gilles Savard. An overview of bilevel optimization. *Annals of operations research*, 153:235–256, 2007.

[Deng *et al.*, 2009] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[Dosovitskiy *et al.*, 2020] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[Fei-Fei *et al.*, 2004] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *2004 conference on computer vision and pattern recognition workshop*, pages 178–178. IEEE, 2004.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[Helber *et al.*, 2019] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 12(7):2217–2226, 2019.

[Hendrycks and Dietterich, 2019] Dan Hendrycks and Thomas Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*, 2019.

[Hinton *et al.*, 2015] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[Hirohashi *et al.*, 2024] Yuki Hirohashi, Tsubasa Hirakawa, Takayoshi Yamashita, and Hironobu Fujiyoshi. Prompt learning with one-shot setting based feature space analysis in vision-and-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7761–7770, 2024.

[Howard and Ruder, 2018] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[Jia *et al.*, 2022] Menglin Jia, Luming Tang, Bor-Chun Chen, Claire Cardie, Serge Belongie, Bharath Hariharan, and Ser-Nam Lim. Visual prompt tuning. In *European Conference on Computer Vision*, pages 709–727. Springer, 2022.

[Khattak *et al.*, 2023] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19113–19122, 2023.

[Krause *et al.*, 2013] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013.

[Lee *et al.*, 2022] Yoonho Lee, Annie S Chen, Fahim Tajwar, Ananya Kumar, Huaxiu Yao, Percy Liang, and Chelsea Finn. Surgical fine-tuning improves adaptation to distribution shifts. *arXiv preprint arXiv:2210.11466*, 2022.

[Liu *et al.*, 2018] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[Lodha *et al.*, 2023] Abhilasha Lodha, Gayatri Belapurkar, Saloni Chalkapurkar, Yuanming Tao, Reshmi Ghosh, Samyadeep Basu, Dmitrii Petrov, and Soundararajan Srinivasan. On surgical fine-tuning for language encoders. *arXiv preprint arXiv:2310.17041*, 2023.

[Maji *et al.*, 2013] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[Nilsback and Zisserman, 2008] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian conference on computer vision, graphics & image processing*, pages 722–729. IEEE, 2008.

[Niu *et al.*, 2022] Shuaicheng Niu, Jiaxiang Wu, Yifan Zhang, Yaofo Chen, Shijian Zheng, Peilin Zhao, and Mingkui Tan. Efficient test-time model adaptation without forgetting. In *International conference on machine learning*, pages 16888–16905. PMLR, 2022.

[Parkhi *et al.*, 2012] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3498–3505. IEEE, 2012.

[Peyré *et al.*, 2019] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport: With applications to data science. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[Pham *et al.*, 2018] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. Efficient neural architecture search via parameters sharing. In *International conference on machine learning*, pages 4095–4104. PMLR, 2018.

[Radford *et al.*, 2021] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[Recht *et al.*, 2019] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pages 5389–5400. PMLR, 2019.

[Ro and Choi, 2021] Youngmin Ro and Jin Young Choi. Autolr: Layer-wise pruning and auto-tuning of learning rates in fine-tuning of deep networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 2486–2494, 2021.

[Saxena and Verbeek, 2016] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. *Advances in neural information processing systems*, 29, 2016.

[Schick and Schütze, 2021] Timo Schick and Hinrich Schütze. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, 2021.

[Selvaraju *et al.*, 2017] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[Shen *et al.*, 2021] Zhiqiang Shen, Zechun Liu, Jie Qin, Marios Savvides, and Kwang-Ting Cheng. Partial is better than all: Revisiting fine-tuning strategy for few-shot learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 9594–9602, 2021.

[Soomro *et al.*, 2012] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[Toneva *et al.*, 2018] Mariya Toneva, Alessandro Sordoni, Remi Tachet des Combes, Adam Trischler, Yoshua Bengio, and Geoffrey J Gordon. An empirical study of example forgetting during deep neural network learning. *arXiv preprint arXiv:1812.05159*, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Vettoruzzo *et al.*, 2024] Anna Vettoruzzo, Mohamed-Rafik Bouguelia, Joaquin Vanschoren, Thorsteinn Rognvaldsson, and KC Santosh. Advances and challenges in meta-learning: A technical review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[Xiao *et al.*, 2010] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3485–3492. IEEE, 2010.

[Yan *et al.*, 2021] Caixia Yan, Xiaojun Chang, Zhihui Li, Weili Guan, Zongyuan Ge, Lei Zhu, and Qinghua Zheng. Zeronas: Differentiable generative adversarial networks search for zero-shot learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(12):9733–9740, 2021.

[Zang *et al.*, 2022] Yuhang Zang, Wei Li, Kaiyang Zhou, Chen Huang, and Chen Change Loy. Unified vision and language prompt learning. *arXiv preprint arXiv:2210.07225*, 2022.

[Zhou *et al.*, 2022a] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16816–16825, 2022.

[Zhou *et al.*, 2022b] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[Zhu *et al.*, 2023] Beier Zhu, Yulei Niu, Yucheng Han, Yue Wu, and Hanwang Zhang. Prompt-aligned gradient for prompt tuning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15659–15669, 2023.