Revisit CLIP: Multi-Perspective Improvements for Vision-Language Model

Wenbo Hu
UCSD HDSI
w1hu@ucsd.edu

Johnny Liu
UCSD HDSI
jrl002@ucsd.edu

Abstract

*Large-scale contrastive vision-language pre-training has shown significant progress in visual representation learning. Unlike traditional visual systems trained by a fixed set of discrete labels, a new paradigm was introduced in CLIP to directly learn to align images with raw texts in an open-vocabulary setting. On downstream tasks, a carefully designed text prompt is employed to make zero-shot predictions. This task can be streamlined by using large language models such as GPT-3 [1] to generate text prompts rather than hand crafting prompts. To avoid non-trivial prompt engineering, context optimization [2] has also been proposed to learn continuous vectors as task-specific prompts with few-shot training examples. Instead of learning the input prompt token, an orthogonal way is learning the weight distributions of prompt [3], which is also very effective. An alternative path is fine-tuning with a light-weight feature adapter [4] on the visual branch The most recent work introduces multimodal prompt learning [5], which uses a synergy function to simultaneously adapt language and vision branches for improved generalization. In our work, we revisit recent improvements in CLIP from different perspectives and propose an optimal way of combining the model's architecture. We demonstrate that Data Augmentation (DA) and Test-Time Augmentation (TTA) are important for few-shot learning (FSL). We propose an end-to-end few-shot learning pipeline (DA + MaPLe + Adapters + TTA) that can be referenced for all downstream tasks. Compared with the state-of-the-art method ProDA [3] in FSL, our model achieves an absolute gain of **6.33%** on the 1-shot learning setting and **4.43%** on the 16-shot setting, averaged over 10 diverse image recognition datasets.*

1. Introduction

Typical computer vision systems suffer from a lack of flexibility as the standard model architecture is designed to predict labels on images from a fixed set of labels upon which the training dataset is built on. There is a fundamental limitation within this approach, namely the reliance of the model on the predetermined labels, and is unscalable when it comes to new categories unseen during training.

Recent progress in Vision-Language Models (VLMs) such as CLIP [6] attempts to solve this issue by learning visual concepts through natural language-described text. VLMs learn aligned embeddings of image and text via contrastive learning [7–9], encouraging the representations of an image and its language description to be similar.

The classification weights are generated by a parameterized text encoder (e.g., a Transformer [10]) through prompting [11]. For example, to differentiate pet images containing different breeds of dogs and cats, the prompt template, "a photo of a {class}, a type of pet," can be used as input to the text encoder, and by substituting real class names for the "{class}" token, class-specific weights for classification can be generated. Vision-language models utilized natural language as their source of supervision, enabling the exploration of open-set visual concepts and proved their effectiveness in acquiring transferable representations compared to discrete labels [6, 12]. However, the provided text should be adapted to the defined context of the task. Manually designing inevitably introduces artificial bias and may not be ideal for the targeted task. Thus, customizing suitable prompts for different recognition tasks relies on repetitive and time-consuming attempts by experts and requires a large validation set for prompt selection [6].

PointCLIP V2 [13] notes the transfer learning capability of CLIP to be lacking on 3D point clouds and utilizes large language models to describe more descriptive 3D-semantic prompts and address the weakness of handcrafted prompts. To automate prompt engineering, Zhou et al. [2] have recently explored the concept of prompt learning—a recent trend in NLP [14–19]—for adapting pre-trained vision-language models. Their approach, Context Optimization (CoOp) [2], turns context words in a prompt into a set of learnable vectors, taking advantage of the differentiable nature of neural networks. With only a few labeled images for learning, CoOp achieves huge improvements over intensively-tuned manual prompts across a wide range of image recognition datasets. To address the weak generalizability problem of CoOp, the follow-up work con-

ditional prompt learning (CoCoOp) [20] introduces the idea to make a prompt conditioned on each input instance (image) rather than fixed once learned. Another perspective is using variational prompt tuning [21] which uses a probabilistic modeling of the underlying distribution of prompts, allowing prompts within the support of an associated concept to be derived through stochastic sampling. Instead of learning the input prompt token, an orthogonal way is prompt distribution learning (ProDA) [3], which learns the weight distributions of the prompt output embedding. Recent work on multi-model prompt learning (MaPLe) [5] demonstrates the importance of using both vision and language branches to improve alignment between the vision and language representations. Their design promotes strong coupling between the vision-language prompts to ensure mutual synergy and discourages learning independent unimodal solution. As for utilizing the visual branch representations, CLIP-Adapter [4] used feature adapters on the visual branch, which learns new features and performs residual style feature blending with the original pre-trained features.

In our work, we attempt to streamline the process of text prompt generation. Our first approach leverages large language models such as GPT-3 and ChatGPT as suggested by PointCLIP V2 [13] to produce more semantically accurate text prompts for different datasets rather than the laborious process of manually suggesting prompts. In addition, we discovered that different perspectives of recent improvements in CLIP are orthogonal and thus can be combined for a better model. We choose to follow MaPLe's [5] architecture and added both text and visual adapters following the CLIP-Adapter's way [4]. We split data augmentation into three categories: weak augmentation, strong augmentation, and stronger augmentation. Following [7]'s way, we choose the optimal data augmentation for the few-shot learning task. Finally, we employed Test Time Augmentation (TTA) [22] for making predictions and demonstrated its strong performance. Since we summarized current state-of-the-art models and employed optimal data augmentation and test time augmentation, our model and workflow can be followed by other researchers to get the best performance on downstream tasks.

In summary, the main contributions of our work include:

- We implemented an automated system of prompt generation to produce competitive initial prompt approaches rather than manual prompt generation.
- We combined current state-of-the-art models from different perspectives and achieved better performance. We gained an average of 5.28% absolute improvement for [1,2,4,8,16] shots learning over 10 diverse image recognition datasets than the best baseline model.
- We employed optimal Data Augmentation and Test Time Augmentation (TTA) and demonstrates TTA's importance in few-shot learning and thus should be used as a convention in future few-shot learning tasks.

2. Related Work

2.1. Vision Language Model

Vision-language pre-training has emerged as a promising technique for building effective recognition models. By learning the connection between image content and language, these models are able to encode rich representations. Recent approaches have leveraged web-scale noisy image-text pairs for training [6, 12]. The success also contributed to the progress in contrastive representation learning-based methods [7–9]. A representative example CLIP [6] has shown impressive results in learning aligned representations of images and text. These VLMs, not only learn powerful visual representations but are also easily transferred to various downstream tasks. Despite their generalized representations, adapting these pre-trained models to specific tasks remains a challenging problem. Many works have demonstrated better performance on downstream tasks by using tailored methods to adapt V-L models for few-shot image recognition [4, 23], object detection [24–28] and segmentation [29–32]. A new technique that utilized multi-modal prompt learning [5] is proposed to effectively adapt pre-trained models for few-shot and zero-shot visual recognition tasks.

2.2. Prompts Learning

Prompt learning/engineering stems from recent advances in natural language processing (NLP). A novel prompt-based paradigm [14–19] for exploiting pre-trained language models has gradually replaced the traditional transfer approach of fine-tuning [33, 34] in NLP. While prompt learning receives considerable attention in NLP, it remains under-explored in computer vision. Pre-trained VLMs [6, 12] introduce hand-crafted prompts to perform zero-shot inference on the downstream tasks. The prompt can be either handcrafted for a downstream task or learned automatically during fine-tuning stage. Full fine-tuning and linear probing [4] are two typical approaches to adapt a V-L model (i.e. CLIP) to the downstream tasks. Since VLMs are purposefully designed to have high capacity, entailing that the model size would be enormous, typically with hundreds of millions of parameters or even billions, fine-tuning the entire model is impractical and degrades the well-learned representation space while linear probing limits the zero-shot capability of CLIP. Therefore, inspired by prompt learning in NLP, many works have proposed to adapt V-L models by learning the prompt tokens in an end-to-end fashion. CoOp [2] fine-tunes CLIP for few-shot image classification by optimizing a continuous set of prompt vectors. CoCoOp [20]

proposes an instance-conditional context that can generalize better to wider unseen classes within the same task. A concurrent work, ProDA [3] optimizes the weight distributions of the prompt output embedding. Bahng et al. [35] perform visual prompt tuning on CLIP by prompting on the vision branch. MaPLe [5] proposes to use multi-model prompt learning (i.e., in both language and vision branches) which is better suited to adapt CLIP and improves alignment between vision and language representations.

3. Methodology

3.1. Review of CLIP and CoCoOp

Contrastive Language-Image Pre-training known as CLIP has well demonstrated the potential of learning open-set visual concepts. CLIP is built using two encoders, one for image and the other for text. The image encoder can be either a ResNet [36] or a ViT [37], which is used to transform an image into a feature vector. The text encoder is a Transformer [10], which takes as input a sequence of word tokens and produces a vectorized representation. CLIP used hand-craft prompts for its text encoder. Specifically, it follows the format "a photo of {class}" where the class name is directly used in the hand-designed description. During training, CLIP adopts a contrastive loss to learn a joint embedding space for the two modalities. Specifically, for a mini-batch of image-text pairs, CLIP maximizes for each image the cosine similarity with the matched text.

Conditional Prompt Learning for Vision-Language Models known as CoCoOp [20] propose an instance-conditional context that can generalize better to wider unseen classes within the same task than its former version CoOp [2]. CoOP aims to overcome the inefficiency problem in prompt engineering for better adapting pre-trained vision-language models to downstream applications. The key idea in CoOp is to model each context token using a continuous vector that can be end-to-end learned from data. Concretely, instead of using "a photo of {class}" as the context, CoOp introduces M learnable context vectors, $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_M\}$, each having the same dimension with the word embeddings. In CoCoOp, on top of the M context vectors, they further learn a lightweight neural network, called Meta-Net, to generate for each input a conditional token (vector), which is then combined with the context vectors.

3.2. Review of CLIP-Adapter

CLIP-Adapter [4] proposes two learnable feature adapters, $A_v(\cdot)$ after the image encoder and $A_t(\cdot)$ after the text encoder. Each adapter which contains two layers of linear transformations is integrated to transform the encoded image output f and encoded text output $\mathbf{W}$. Then a residual connection is adopted for the feature adapter to avoid forgetting the original knowledge encoded by the pre-trained CLIP. Two constant values α and β are employed as the "residual ratio" to determine how much original knowledge is kept. In summary, the feature adapters can be written as

$$\begin{aligned} A_v(f) &= ReLU(f^T\mathbf{W}_1^v)\mathbf{W}_2^v \\ A_t(\mathbf{W}) &= ReLU(\mathbf{W}^T\mathbf{W}_1^t)\mathbf{W}_2^t \end{aligned} \tag{1}$$

The new knowledge captured via finetuning is added with the original features via residual connections:

$$\begin{aligned} f^* &= \alpha A_v(f)^T + (1-\alpha)f \\ \mathbf{W}^* &= \beta A_t(\mathbf{W})^T + (1-\beta)\mathbf{W} \end{aligned} \tag{2}$$

3.3. Our Approach

We propose to use optimal data augmentation and test time augmentation to make few-shot training more generalized to unseen classes. We followed MaPLe's way [5] to create a synergy between the text branch and the visual branch and used both learnable visual prompt and text prompt. Altough using only the visual feature adapter is experimented in CLIP-Adapter [4] to have a better performance, we choose to use both adapters $A_v(\cdot)$ and $A_t(\cdot)$. Our overall architecture is visualized in Figure 1.

Data Augmentation: We followed SimCLR's idea [7] but without the Gaussian blur since we discovered adding it degrades performance in few-shot learning. Stronger and mixing data augmentations such as Mixup and Cutmix improves the performance of one-shot training. However, when the k-shot i.e number of data per class is increased to [2,4,8,16], it degrades the performance. Unless otherwise specified, the default and optimal data augmentation we use is random horizontal flip, random crop and resize, and color distortions.

Text Prompting: The CLIP text encoder generates feature representations for text description by tokenizing the words and projecting them to word embeddings $W_0 = [w_0^1, w_0^2, \ldots, w_0^N] \in \mathbb{R}^{N \times d_l}$. At each stage, W_i is input to the $(i+1)^{th}$ transformer later of the text encoding branch ($\mathcal{L}_{i+1}$). For prompt learning, b learnable tokens $\{P^i \in \mathbb{R}^{d_l}\}_{i=1}^b$ are introduced. The input embeddings now follow the form $[P^1, P^2, \ldots, P^b, W]$. New learnable tokens are further introduced in each transformer block of the language encoder ($\mathcal{L}_i$) up to a specific depth J,

$$[\_, W_i] = \mathcal{L}_i([P_{i-1}, W_{i-1}]) \quad i = 1, 2, \ldots, J \tag{3}$$

Here $[\cdot, \cdot]$ acts as a concatenation operation. After J^{th} transformer layer, the subsequent layers process previous layer prompts and final text representation w is computed,

$$[P_j, W_j] = \mathcal{L}_j([P_{j-1}, W_{j-1}]) \quad j = J+1, \ldots, K \tag{4}$$

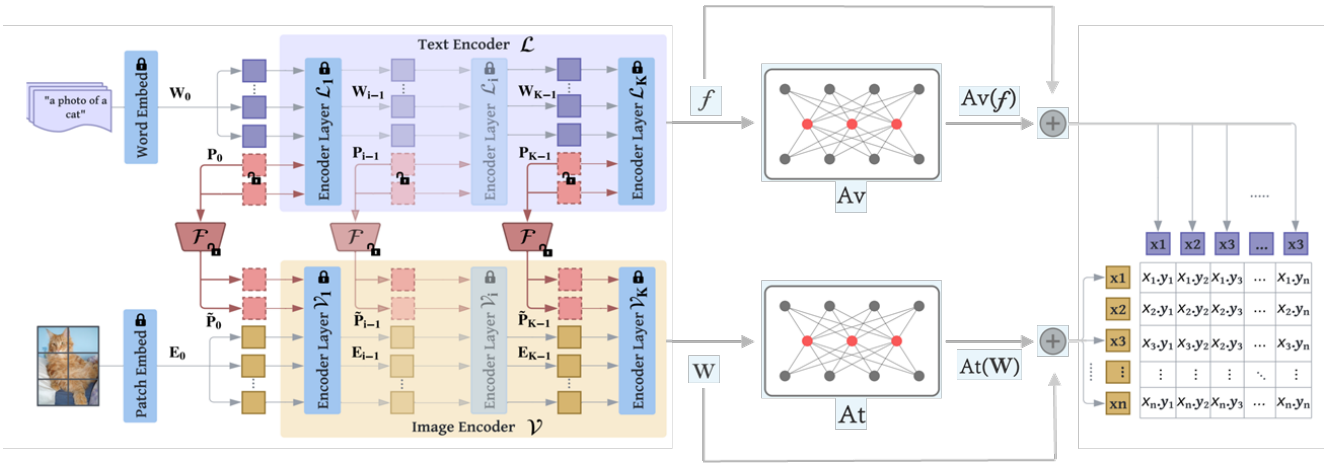

Figure 1. Overview of our proposed framework. The first part is MaPLe which tunes both vision and language branches where only the context prompts are learned, while the rest of the model is frozen. MaPLe conditions the vision prompts on language prompts via a V-L coupling function $\mathcal{F}$ to induce mutual synergy between the two modalities. The following A_v and A_t are each visual adapter and text adapter which perform residual style feature blending with the output embedding of visual branch and text branch separately.

The final text representation w is obtained by projecting the text embeddings corresponding to the last token of the last transformer block $\mathcal{L}_K$ to a common V-L latent embedding space via $TextProj(\cdot)$

$$w = TextProj(w_K^N) \quad w \in \mathbb{R}^{d_{vl}} \tag{5}$$

Viusal Prompting: The Image encoder $\mathcal{V}$ with K transformer layers $\{\mathcal{V}_i\}_{i=1}^{K}$, splits the image I into M fixed-size patches which are projected into patch embeddings $E_0 \in \mathbb{R}^{M \times d_v}$. Patch embeddings E_i are then input to the $(i+1)^{th}$ transformer block $(\mathcal{V}_{i+1})$ along with an appended learnable class token c_i and sequentially processed through K transformer blocks. Similar to text prompting, b learnable tokens $\{\tilde{P}^i \in \mathbb{R}^{d_v}\}_{i=1}^{b}$ are introduced. New learnable tokens are further introduced in deeper transformer layers of the image encoder $\mathcal{V}$ up to depth J.

$$\begin{aligned}
[c_i, E_i, _] &= \mathcal{V}_i([c_{i-1}, E_{i-1}, \tilde{P}_{i-1}]) \quad i = 1, 2, \ldots, J \\
[c_j, E_j, \tilde{P}_j] &= \mathcal{V}_j([c_{j-1}, E_{j-1}, \tilde{P}_{j-1}]) \quad j = J+1, \ldots, K
\end{aligned} \tag{6}$$

To obtain the final image representation x, the class token c_K of the last transformer layer $(\mathcal{V}_K)$ is projected to a common V-L latent embedding space via $ImageProj(\cdot)$,

$$x = ImageProj(c_K) \quad x \in \mathbb{R}^{d_{vl}} \tag{7}$$

Synergy function connecting V-L branches: To combine and interact the language prompts P and the vision prompts $\tilde{P}$, MaPLe [5] proposed a V-L coupling function $\mathcal{F}(\cdot)$ such that $\tilde{P}_k = \mathcal{F}_k(P_k)$. The coupling function is a linear layer that maps d_l dimensional inputs to d_v. The direction of mapping $(P \rightarrow \tilde{P})$ is chosen because of a lower information loss in such a design since $d_v > d_l$. The architecture is now:

$$\begin{aligned}
[c_i, E_i, _] &= \mathcal{V}_i([c_{i-1}, E_{i-1}, \mathcal{F}_{\mathbf{i-1}}(\tilde{\mathbf{P}}_{\mathbf{i-1}})]) \quad i = 1, 2, \ldots, J \\
[c_j, E_j, \tilde{P}_j] &= \mathcal{V}_j([c_{j-1}, E_{j-1}, \tilde{P}_{j-1}]) \quad j = J+1, \ldots, K \\
x &= ImageProj(c_K)
\end{aligned} \tag{8}$$

Test Time Augmentation (TTA): TTA is a technique used in computer vision community. Here, we emphasize its importance in improving performance during few-shot learning. For the image input I_0, we have k image transforms $\{T_1, T_2, \ldots, T_k\}$ to transform the image to $\{T_1(I_0), T_2(I_0), \ldots, T_k(I_0)\}$. Then we take the average of the output predictions of the model across the dimension k. Here is a pseudo-code for a single image input example (in practice, it's minibatch).

Algorithm 1 Test Time Augmentation (TTA)

for T in $\{T_1, T_2, \ldots, T_k\}$ **do**
  $I_0 = T(I_0)$
  $Pred = Model(I_0) \quad Pred \in \mathbb{R}^{num\_class}$
end for
$Preds = stack(Pred) \quad Preds \in \mathbb{R}^{k \times num\_class}$
$Pred = mean(Preds, dim = k) \quad Pred \in \mathbb{R}^{num\_class}$

In summary, our approach can be written as, let x be image features generated by the image encoder after using data augmentation or test time augmentation in algorithm 1 on the input image x_0. So $x = f_v(Aug(x_0))$, where f_v is the visual branch image encoder followed the Equation

(8). We then followed the Eqs (3), (4), (5) to get $\{\boldsymbol{w}_i\}_{i=1}^{K}$ which is a set of weight vectors produced by the text encoder, each representing a category (suppose there are K categories in total). We then applied both the visual feature adapter and text feature adapter following Eqs (1), (2). So x is denoted by $x = \alpha A_v(x)^T + (1-\alpha)x$ and w is denoted by $w = \alpha A_t(w)^T + (1-\alpha)w$
The prediction probability is then

$$p(y|x) = \frac{exp(sim(x, w_y)/\tau)}{\sum_{i=1}^{K} exp(sim(x, w_i)/\tau)} \tag{9}$$

where $sim(\cdot, \cdot)$ denotes cosine similarity, τ is a learned temperature parameter.

4. Experiments

Our approach is mainly evaluated in the Traditional few-shot training setting using $k = [1, 2, 4, 8, 16]$ (Section 4.1 4.2). However, due to constrained resources, we are unable to train Imagenet [38], it will be delayed to future experiments.

Datasets: Following CLIP [6] and CoOP [2], we evaluated our model on 11 downstream classification datasets, including general object recognition (ImageNet-1k [38], Caltech-101 [39]), fine-grained object recognition (Food-101 [40], Oxford-IIIT Pets [41], Stanford Cars [42], Oxford Flowers 102 [43], and FGVC Aircraft [44]), human interactions classification UCF-101 [45], large scale scene recognition dataset (SUN397 [46]), remote sensing recognition (EuroSAT [47]), and texture recognition (DTD [48]). Note here because of constrained resources, we are unable to train Imagenet [38], it will be delayed to future experiments.

4.1. Comparison of Textual Prompt Generation Methods

Datasets: For this section, we used a subset of 6 the 10 datasets defined above as some of the generated prompts had little to no difference from the human prompts listed by CLIP's [6] authors. To be exact, we used Caltech-101 [39]), Oxford-IIIT Pets [41], Stanford Cars [42], FGVC Aircraft [44]), EuroSAT [47], and DTD [48].

Implementation Details: We utilized a pre-trained ViT-B/16 CLIP model where d_l = 512, d_v = 768, and d_{vl} = 512 as the baseline model to start evaluation on prompts provided. The remaining architecture remains the same as CLIP's [6] initial implementation of zero-shot CLIP and few-shot learning CLIP results. We used an RTX 2080 GPU for inference and few-shot learning with batch sizes of 64 to to avoid memory issues between datasets as they vary greatly in size and class count.

Prompt Types: We compare results on zero-shot CLIP between three different prompt types: GPT-3, ChatGPT, and human suggested. For both large language models, we generated prompts by giving context to the vision-language implementation of CLIP and providing an example prompt, such as "this is a photo of {class}, a type of pet." In addition, we gave information about the content of the dataset to provide context to the generated prompt. We find that it becomes necessary to specify the model return a model with a single label input as both LLMs tend to end up just describing the dataset details with no class input. Human produced prompts were sourced from the initial CLIP paper by using the prompt templates the authors cited using. We do not implement ensemble prompts as is done in CLIP due to computational limitations, which may result in slight discrepancies compared to the reported results in CLIP.

Evalution Metric: We choose to use top-1 accuracy as our metric of choice as the classes tend to be fairly balanced in these procured datasets. Top-5 accuracy tends to be too high for some of the datasets with more limited label sets, such as EuroSAT, and thus we settle on reporting solely top-1 accuracy. For few-shot learning, we follow the precedent set by CLIP [6] on the number of labeled examples per class, which is 1, 2, 4, 8, and 16 labeled samples per class on each downstream task. Training examples are randomly sampled from each class of the dataset and then evaluated on the full dataset.

Performance Comparison & Results: Table 1 shows the comparison of prompts on 10 downstream tasks except ImageNet. Additionally, we show the results of improved prompts on few-shot learning for the previously stated examples per class. GPT-3 appears to generally perform a little better than ChatGPT suggested prompts, which tended to be because ChatGPT gave simpler, more conversational prompts while GPT-3 gave slightly more detailed responses. For majority of the datasets, LLM generated prompts provide a promising increase in performance for zero-shot performance. However, these results are not very promising when put in the context of few-shot learning. Since low amounts of few-shot learning, such as 1-shot and 2-shot still perform worse than zero-shot, we believe it requires a different approach to solve the problem of low performance few-shot learning. It is reassuring that the LLM generated prompts typically do not provide worse performance than the prompts that humans had to create through trial and error, but due to the inconsistent improvements with few-shot learning, we believe that text prompt changes is not sufficient to significantly upgrade the results of CLIP [6]. Thus, we implement other orthogonal

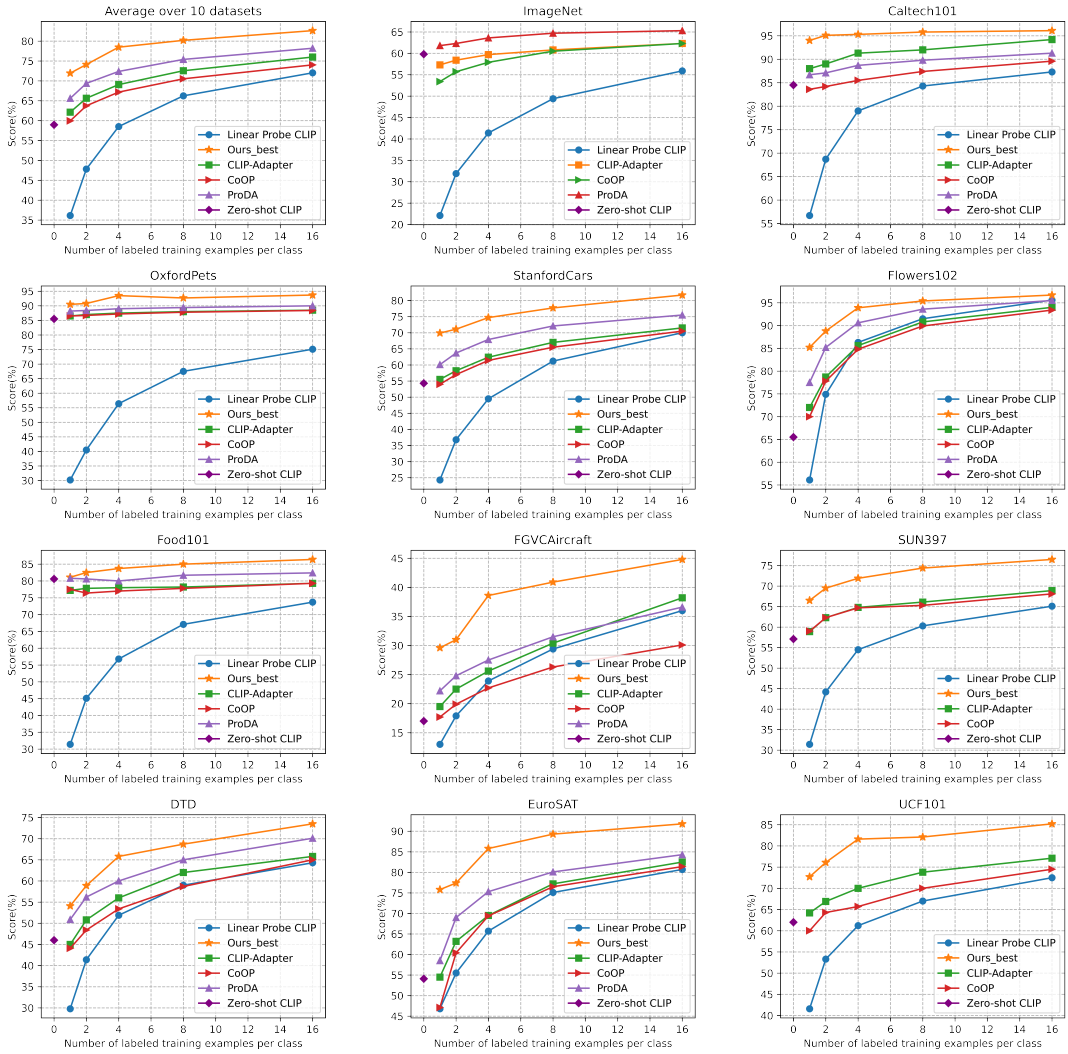

Figure 2. Main results of few-shot learning on 10 datasets. Not including ImageNet yet. Our method consistently and significantly shows better performance over previous baselines across different training shots. Here Ours-best is picking from the best performance between using TTA and not using TTA in our framework.

innovations to fundamentally change the text prompt aspect of CLIP away from manual input of prompts.

4.2. Few shot Learning

Implementation Details: We applied prompt tuning on a pre-trained ViT-B/16 CLIP model where d_l = 512, d_v = 768, and d_{vl} = 512. For MaPLe, we set prompt depth J to 9 and the language and vision prompt lengths to 2. All models are trained for 25 epochs with a batch-size of equal to the number of shots in few-shot training and a learning rate of 0.0035 via SGD optimizer on a single RTX 3080 GPU. For adapters, we choose α to be 0.4, that is

Dataset	Human	GPT-3	ChatGPT	1-shot	2-shot	4-shot	8-shot	16-shot
Caltech101	83.34	84.46	84.36	75.07	81.52	87.39	90.63	92.78
OxfordPets	86.28	87.73	87.06	42.08	60.54	75.06	84.15	89.68
StanfordCars	54.5	64.99	64.82	37.84	52.48	66.33	78.71	82.26
FGVCAircraft	18.27	24.57	25.32	22.47	32.30	40.37	53.87	71.66
EuroSAT	52.33	54.76	53.67	50.54	59.46	74.86	80.6	84.78
DTD	46.22	42.91	42.66	36.47	47.76	58.42	67.94	75.05

Table 1. Zero-shot performance results (top-1 accuracy) on 6 datasets using human, GPT-3, and ChatGPT suggested text prompts. Few-shot learning results were recorded using the best result between GPT-3 and ChatGPT prompts.

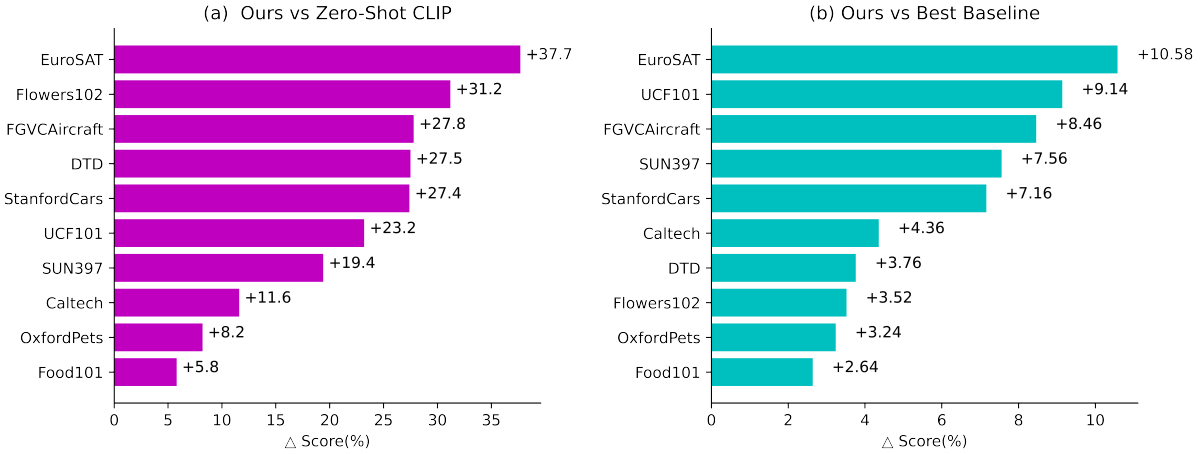

Figure 3. **Comparison with other baselines**. We show the absolute improvement of our approach compared to Zero shot CLIP [6] and state-of-the-art baselines on each downstream task. (a) We compare the hand-crafted prompts to Our method with 16 samples per class. (b) Our method is compared with the best performance among other baselines (including CoOp [2], CLIP-Adapter [4], Linear-probe CLIP [6], and ProDA [3]) by their average results over various shots (1, 2, 4, 8, and 16).

40% percent of knowledge is from the adapter and 60% of knowledge is from the output of encoders. We use the model of the last training epoch for evaluation.

Baseline Models: We compare our framework with five baseline models – the Zero-shot CLIP [6], Linear probe CLIP [6], CoOp [2], CLIP-Adapter [4], and ProDA [3]. The **Zero-shot CLIP** [6] uses hand-crafted prompts to generate the target classifier on the downstream task, as discussed in Sec.3.1. The prompt templates applied in each dataset are the same as CLIP [6]. For **Linear probe CLIP** [6], we trained a logistic regression classifier on the features of training images following the same binary search for hyperparameters. **CoOp** [2] learns a soft prompt by minimizing the classification loss, which is discussed in Sec.3.1. There are multiple candidate positions to place the class token in the prompt template, namely at the front, in the middle, or at the end. Here, we choose to compare with CoOp's best performance variant – placing the class token at the end of the 16-token soft prompt and sharing such a context among different classes. For **CLIP-Adapter** [4], we followed the implementation settings such as using the best α for different datasets and only use the visual branch adapter for best performance. Our implementation obtains slightly better results than those reported in the paper [4]. Since **ProDA** [3] doesn't have a published codebase and reproducing implementation by ourselves may be inconsistent with the authors' ideas, we choose to use the results directly from the paper.

Evalution Metric: We follow the few-shot transfer setting as CLIP [6], which learns with 1, 2, 4, 8, and 16 labeled samples per class on each downstream task. Training examples are sampled from the training set of each dataset. After training, each method is evaluated on the full test set of the downstream task with the corresponding metric. We report the results by fixing the seed as 1.

Performance Comparison & Results: Fig 2 shows the comparison with the baseline methods on 10 downstream tasks except ImageNet and the average results over all datasets. More detailed results are provided in the

supplementary materials. We also provide a summary of the absolute improvement of our approach compared to the best baseline model in Fig 3. Since MaPLe [5] directly concatenates prompts to the ViT model, here we adopt the ViT B/16 backbone from CLIP while other baselines are adopting the RN50 backbone from CLIP [6].

In comparison to the hand-crafted prompts (Zero-shot CLIP) [6], our approach substantially and consistently improves performance. Our method relatively improves the average results by 12.99% with 1 training sample per class and 23.69% with 16 training samples per class. In complex datasets such as EuroSAT, the relative improvements are more significant, which are 21.7% in the 1-shot setting and 37.7% in the 16-shot setting. For fine-grained object recognition such as Flower-102, our method achieve a substantial relative improvement, which is 19.7% in the 1-shot setting and more than 31.2% in the 16-shot setting.

The comparison with linear probe CLIP [6] demonstrates the benefit of using category names for recognition in the few-shot setting. In 1-shot, our approach has a relatively 35.81%higher average score than the linear probe CLIP. Natural language provides dense task-related information and is better than hand-designed prompts. Our results indicate that prompt learning is an efficient way to address vision tasks.

In addition, our approach consistently and significantly outperforms CoOp [2]. We have 11.99% relative average performance improvement in 1-shot and 8.61% in 16-shot compared with it. These results suggest the importance of concatenating learnable tokens to deep layers. By consistently updating weights for each layer until J, task-related information is captured and propagates to deep layers.

When compare with CLIP-Adapter [4], where they only used the visual branch feature adapter. Our method has 9.81% relative average performance improvement in 1-shot and 6.64% in 16-shot. It indicates prompt learning methods achieve a better performance than residual connections for prompt-fixed pre-trained vision-language models.

ProDA [3] demonstrates a strong result in few-shot learning by learning the distribution of weights. Part of our method, MaPLe [5] took an orthogonal way compared to ProDA and learned the input prompt tokens. There are no direct results indicating learning the input tokens to deep layers is better than learning the output prompt distribution since we also updated visual prompts and used feature adapters. Our method has 6.33% relative average performance improvement in 1-shot and 4.43% in 16-shot compared to ProDA.

Overall, our method (Data Augmentation + MaPLe + Adapters + TTA) substantially outperforms its prompt learning counterparts and residual feature adapter counterparts. These results demonstrate the effectiveness of our approach, which learns both visual and text prompts to deep layers and uses a coupling function to encourage the synergy between two branches. Two feature adapters are further used to capture the downstream task-related information by offering a controllable hyperparameter α to determine the proportion of information we want to keep specific to the downstream dataset. In the end, Test Time Augmentation is performed to capture a more robust data distribution in the real world by transforming the limited few shot data samples. Our end-to-end approach can be referred to as a standard workflow for few-shot learning as we combined the new breakthroughs in the field.

4.3. Ablation Study

Test Time Augmentation: Test Time Augmentation (TTA) is less valued in the few-shot learning community. However, due to the limited samples we can acquire in this task, data augmentation is a neglectable approach to get a more robust data distribution just like in the real world. We performed few-shot learning following our framework and then compared the performance between using TTA and not using TTA in Fig 4. We notice that using TTA degrades the performance for 1-shot and 2-shot settings. But it improves performance stably when we have more training samples per class. It indicates that with a little bit more samples, TTA is able to capture a better and more robust data distribution like the real world.

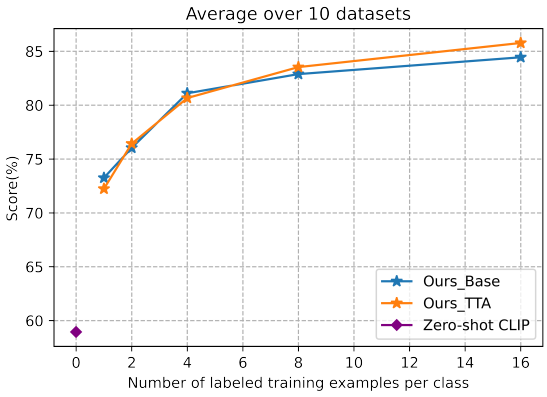

Figure 4. Compare the results of few-shot learning on 10 datasets between using TTA and not using TTA in our framework.

5. Conclusion

In this paper, we revisit current state-of-the-art methods in Contrastive Vision-Language Models from different perspectives. The first approach was to leverage large language models in hopes of being able to utilize the models to find semantically meaningful text prompts rather than blindly manually testing. This approach had strong effects on zero-shot learning accuracy but tended to struggle in the

realm of few-shot learning. Then, we combined orthogonal model architecture and propose optimal data augmentation and test-time augmentation to enhance the limited few-shot data to a more robust distribution. Our work shows that multimodal prompt learning i.e learning the context prompt from both visual branch and text branch is currently the best way to address downstream visual recognition tasks with a pre-trained VLM. According to the experimental results, our method outperforms competitive baselines on ten image classification datasets under different few-shot setups. Our work follows an end-to-end approach that can be referred to as a standard workflow for few-shot learning as we combined new breakthroughs in the field. In the future, we will focus more on multimodal prompt learning and parametrize it by a distribution from a bayesian perspective to improve the performance. We hope our approach will inspire future work.

References

[1] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. 1

[2] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Learning to prompt for vision-language models," *International Journal of Computer Vision*, vol. 130, no. 9, pp. 2337–2348, jul 2022. 1, 2, 3, 5, 7, 8

[3] Y. Lu, J. Liu, Y. Zhang, Y. Liu, and X. Tian, "Prompt distribution learning," 2022. 1, 2, 3, 7, 8

[4] P. Gao, S. Geng, R. Zhang, T. Ma, R. Fang, Y. Zhang, H. Li, and Y. Qiao, "Clip-adapter: Better vision-language models with feature adapters," *arXiv preprint arXiv:2110.04544*, 2021. 1, 2, 3, 7, 8

[5] M. U. khattak, H. Rasheed, M. Maaz, S. Khan, and F. S. Khan, "Maple: Multi-modal prompt learning," *arXiv:2210.03117*, 2022. 1, 2, 3, 4, 8

[6] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. 1, 2, 5, 7, 8

[7] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A simple framework for contrastive learning of visual representations," 2020. 1, 2, 3

[8] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, "Momentum contrast for unsupervised visual representation learning," 2019. 1, 2

[9] A. v. d. Oord, Y. Li, and O. Vinyals, "Representation learning with contrastive predictive coding," 2018. 1, 2

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017. 1, 3

[11] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig, "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing," 2021. 1

[12] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. V. Le, Y. Sung, Z. Li, and T. Duerig, "Scaling up visual and vision-language representation learning with noisy text supervision," 2021. 1, 2

[13] X. Zhu, R. Zhang, B. He, Z. Zeng, S. Zhang, and P. Gao, "Pointclip v2: Adapting clip for powerful 3d open-world learning," 2022. 1, 2

[14] T. Gao, A. Fisch, and D. Chen, "Making pre-trained language models better few-shot learners," 2020. 1, 2

[15] Z. Jiang, F. F. Xu, J. Araki, and G. Neubig, "How can we know what language models know?" 2019. 1, 2

[16] B. Lester, R. Al-Rfou, and N. Constant, "The power of scale for parameter-efficient prompt tuning," 2021. 1, 2

[17] X. L. Li and P. Liang, "Prefix-tuning: Optimizing continuous prompts for generation," 2021. 1, 2

[18] T. Shin, Y. Razeghi, R. L. Logan, E. Wallace, and S. Singh, "Autoprompt: Eliciting knowledge from language models with automatically generated prompts," 2020. 1, 2

[19] Z. Zhong, D. Friedman, and D. Chen, "Factual probing is [mask]: Learning vs. learning to recall," 2021. 1, 2

[20] K. Zhou, J. Yang, C. C. Loy, and Z. Liu, "Conditional prompt learning for vision-language models," 2022. 2, 3

[21] M. M. Derakhshani, E. Sanchez, A. Bulat, V. G. T. da Costa, C. G. M. Snoek, G. Tzimiropoulos, and B. Martinez, "Variational prompt tuning improves generalization of vision-language models," 2022. 2

[22] D. Shanmugam, D. Blalock, G. Balakrishnan, and J. Guttag, "Better aggregation in test-time augmentation," 2020. 2

[23] R. Zhang, R. Fang, W. Zhang, P. Gao, K. Li, J. Dai, Y. Qiao, and H. Li, "Tip-adapter: Training-free clip-adapter for better vision-language modeling," 2021. 2

[24] H. Rasheed, M. Maaz, M. U. Khattak, S. Khan, and F. S. Khan, "Bridging the gap between object and image-level representations for open-vocabulary detection," 2022. 2

[25] M. Maaz, H. Rasheed, S. Khan, F. S. Khan, R. M. Anwer, and M.-H. Yang, "Class-agnostic object detection with multi-modal transformer," 2021. 2

[26] X. Zhou, R. Girdhar, A. Joulin, P. Krähenbühl, and I. Misra, "Detecting twenty-thousand classes using image-level supervision," 2022. 2

[27] Y. Zang, W. Li, K. Zhou, C. Huang, and C. C. Loy, "Open-vocabulary DETR with conditional matching," in *Lecture Notes in Computer Science*. Springer Nature Switzerland, 2022, pp. 106–122. 2

[28] C. Feng, Y. Zhong, Z. Jie, X. Chu, H. Ren, X. Wei, W. Xie, and L. Ma, "Promptdet: Towards open-vocabulary detection using uncurated images," 2022. 2

[29] B. Li, K. Q. Weinberger, S. Belongie, V. Koltun, and R. Ranftl, "Language-driven semantic segmentation," 2022. 2

[30] Y. Rao, W. Zhao, G. Chen, Y. Tang, Z. Zhu, G. Huang, J. Zhou, and J. Lu, "Denseclip: Language-guided dense prediction with context-aware prompting," 2021. 2

[31] J. Ding, N. Xue, G.-S. Xia, and D. Dai, "Decoupling zero-shot semantic segmentation," 2021. 2

[32] T. Lüddecke and A. S. Ecker, "Image segmentation using text and image prompts," 2021. 2

[33] L. Dong, N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou, and H.-W. Hon, "Unified language model pre-training for natural language understanding and generation," 2019. 2

[34] T. S. Alec Radford, Karthik Narasimhan and I. Sutskever, "Improving language understanding by generative pre-training," 2018. 2

[35] H. Bahng, A. Jahanian, S. Sankaranarayanan, and P. Isola, "Exploring visual prompts for adapting large-scale models," 2022. 3

[36] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. 3

[37] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2020. 3

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009. 5

[39] F.-F. Li, M. Andreeto, M. Ranzato, and P. Perona, "Caltech 101," Apr 2022. 5

[40] L. Bossard, M. Guillaumin, and L. Van Gool, "Food-101 – mining discriminative components with random forests," in *European Conference on Computer Vision*, 2014. 5

[41] A. Z. Omkar M Parkhi, Andrea Vedaldi and C. V. Jawahar, "Cats and dogs," 2012. 5

[42] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013. 5

[43] E. Nilsback and A. Zisserman, "automated flower classification over a large number of classes," 2008. 5

[44] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi, "Fine-grained visual classification of aircraft," Tech. Rep., 2013. 5

[45] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," 2012. 5

[46] N. Dalal and B. Triggs, "Scalable recognition with a vocabulary tree," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, San Diego, CA, USA, 2005, pp. 216–223. 5

[47] P. Helber, B. Bischke, A. Dengel, and D. 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*, 2019. 5

[48] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, , and A. Vedaldi, "Describing textures in the wild," in *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014. 5