Large Language Models are Good Prompt Learners for Low-Shot Image Classification

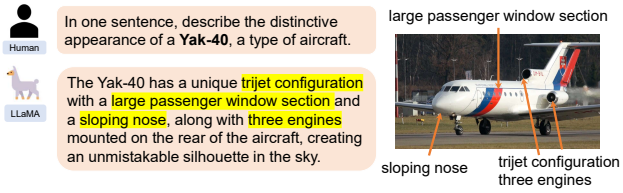
Zhaoheng Zheng Jingmin Wei Xuefeng Hu Haidong Zhu Ram Nevatia
Viterbi School of Engineering
University of Southern California
{zhaoheng.zheng, jingminw, xuefengh, haidongz, nevatia}@usc.edu

Abstract

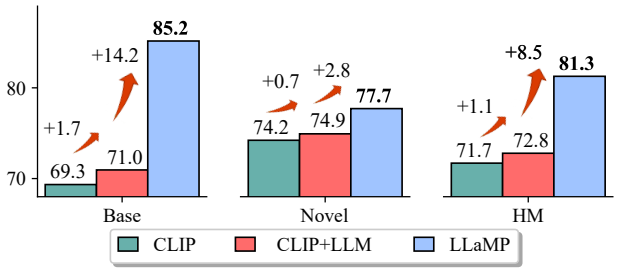
Low-shot image classification, where training images are limited or inaccessible, has benefited from recent progress on pre-trained vision-language (VL) models with strong generalizability, e.g. *CLIP. Prompt learning methods built with VL models generate text features from the class names that only have confined class-specific information. Large Language Models (LLMs), with their vast encyclopedic knowledge, emerge as the complement. Thus, in this paper, we discuss the integration of LLMs to enhance pre-trained VL models, specifically on low-shot classification. However, the domain gap between language and vision blocks the direct application of LLMs. Thus, we propose LLaMP, **L**arge **La**nguage **M**odels as **P**rompt learners, that produces adaptive prompts for the CLIP text encoder, establishing it as the connecting bridge. Experiments show that, compared with other state-of-the-art prompt learning methods, LLaMP yields better performance on both zero-shot generalization and few-shot image classification, over a spectrum of 11 datasets. Code will be made available at:* https://github.com/zhaohengz/LLaMP.

1. Introduction

Low-shot image classification tasks, including few-shot and zero-shot variants, are to learn from a set of class names along with a limited or null set of images. Such capacities are crucial for the extension and generalization of vision systems. Vision-Language (VL) models trained on large-scale web data, such as CLIP [32] and ALIGN [14], provide a new paradigm due to their generalization capabilities that includes zero-shot classification, and have been used in recent work [17–19, 23, 24, 48, 49]. Due to the scarcity of images for training, methods built for both tasks rely heavily on merely category names as the source of class-specific knowledge, resulting in a shortage of distinguishable descriptions. Meanwhile, Large Language Models (LLMs),



(a) Visual Descriptions from an LLM.



(b) LLMs' Knowledge Boosts the Performance.

Figure 1. Demonstration of LLaMP: (a) LLMs can provide visual descriptions for fine-grained object categories; (b) Zero-shot base-to-novel generalization benefits from the LLM knowledge.

e.g. GPT-4 [28] and LLaMA [38, 39], have demonstrated their encyclopedic knowledge and thus can provide linguistic visual descriptions for objects. Here, we investigate how to leverage LLMs for low-shot image classification.

The emergence of prompt learning has provided an efficient way to adapt large pre-trained models. Previous work has explored various strategies to prompt vision-language (VL) models, including vision-conditioned text prompt learning [48], joint VL prompt learning [18] and self-regulated VL prompts[19]. On the text side, regardless of the learning strategy, learned prompt vectors are shared across all categories. The only difference among text inputs is the class name. In low-shot scenarios where visual data is limited, the extraction of class-specific knowledge from textual inputs becomes essential. However, the current paradigm, which relies on the CLIP text encoder to distinguish between class names, faces challenges, partic-

ularly with fine-grained target categories. For example, in FGVCAircraft [25], the class name "Yak-40", can barely provide any information for recognizing the object.

Large Language Models, trained with large text corpora, are good candidates to serve as the complement. As in Fig. 1a, being queried about "Yak-40", the LLM generates a sentence detailing the visual appearance of the Yak-40 that can be further parsed into noun phrases and integrated into text prompts, providing richer information, compared with the ordinary prompt. We also show in Fig. 1b that by simply incorporating noun phrases extracted from a LLM's response, the performance of the ordinary CLIP models is improved by more than 1% without any training. Although recent prompt-learning based methods have shown notable improvements, it is non-trivial to apply them on textual visual descriptions generated by LLMs. Thus, instead of directly taking LLM generations as the textual input, we aim at producing class-specific representations by adapting LLMs to low-shot image classification.

One challenge of the adaption is the domain gap between vision and language. When trained exclusively with textual corpora, the latent feature space of a LLM significantly diverges from that of its visual counterpart. Even worse, the data scarcity under the low-shot scenario make it virtually impossible to align two spaces through plain contrastive loss. We argue that, the CLIP text encoder, which is trained to project features from the language domain into the joint VL domain, can serve as the bridge. Thus, we propose the **LLaMP** framework, **L**arge **La**nguage **M**odels as **P**rompt learners, which leverages LLMs to learn informative prompts for CLIP models. In LLaMP, we treat LLMs as the prompt learner of the CLIP text encoder. More specifically, for each object category, LLaMP extracts corresponding knowledge from the LLM and yields class-specific prompt vectors, which are further combined with class-agnostic prompt embeddings (as in previous approaches), and encoded by the CLIP text encoder. We design an efficient tuning pipeline to avoid fully fine-tuning the language model while performing effective adaptation.

Following the protocol in [48, 49], we evaluate LLaMP with two typical scenarios: zero-shot base-to-novel generalization [49] and few-shot image classification. For each scenario, we run LLaMP with 11 datasets covering a spectrum of tasks. On average, LLaMP achieves a 1.3% boost on the harmonic mean against the state-of-the-art PSRC [19], and 9.6% over the ordinary CLIP [32], on base-to-novel generalization. We also observe an average improvement of 0.94% on 16-shot image classification.

In summary, our approach makes use of Large Language Models to improve performance in low-shot image classification scenarios. The main contributions are: i) To the best of our knowledge, we are the first to investigate how to use the encyclopedic knowledge inherent in Large Language Models (LLMs) to enhance low-shot image classification; ii) We design a framework, LLaMP, to effectively adapt LLMs for image classification, without training the entire language model, and achieve state-of-the-art in both few-shot and zero-shot settings; iii) We conduct extensive analysis investigating the effectiveness of each components of LLaMP, and discuss the optimal setup for LLM-aided image classification.

2. Related Work

Large Language Models (LLMs). Recent years have witnessed remarkable progress in scaling up the size and capabilities of LLMs. Zhang *et al.* [47] first introduced a suite of transformers pre-trained at scale, followed by PaLM [6]. ChatGPT/GPT-4 [27, 28] emerged as a milestone conversational model, demonstrated impressive conversational abilities as a generalist model. Vicuna [5] further advanced by learning from ChatGPT, while LLaMA [38] demonstrates that larger scale training yields stronger foundation models. The subsequent LLaMA-2 [39] and PaLM-2 [2] achieved further gains in scale, efficiency and reasoning. Most recently, Almazrouei *et al.* [1] released Falcon, a 400B model.

Zero-Shot Learning (ZSL). ZSL stands in contrast to traditional fully-supervised paradigms. Instead of relying on direct visual training samples, it leverages side information that can be drawn from a multitude of non-visual domains, including attributes [22], word embeddings [36, 40], and descriptive texts [34]. Zhang *et al.* [46] designed an embedding model to bridge the gap between seen and unseen categories. Concurrently, studies like [4, 44, 50] have spotlighted that generative models can produce features for unseen categories. Moreover, Graph Convolution Networks (GCN) [20] has been explored in research such as [16, 40] for further generalization.

Prompt Learning. With the progress in large-scale vision-language models, such as CLIP [32] and ALIGN [14], which reveal their capacity in zero-shot transferability, prompt learning has emerged as an efficient learning scheme, where learnable prompts are appended to the input to fine-tune models. For low-shot image classification, CoOp [49] and CoCoOp [48], which modeled context words as learnable vectors to automate prompt engineering, have shown significant improvements over regular CLIP. MaPLe [18] further employed a hierarchical multi-modal prompting strategy across transformer blocks for progressive feature modeling. Kan *et al.* [17] incorporated external knowledge by designing knowledge-aware prompts and adaptation head for better generalization. Lee *et al.* [23] used masked attention to prevent internal representation shift for better generalization. Khattak *et al.* [19] further improved prompt learning by guiding prompts to balance task-specific and task-agnostic knowledge via mutual agreement maximization and prompt ensemble.

3. Approach

3.1. Preliminaries

Similar to previous CLIP-based learning approaches, we consider the classification problem as an image-text matching problem. We denote the image encoder and the text encoder, in CLIP-like models, as $\mathcal{F}$ and $\mathcal{G}$, parameterized by $\theta_{\mathcal{F}}$ and $\theta_{\mathcal{G}}$, respectively. An input image $\boldsymbol{x} \in \mathbb{R}^{C\times H\times W}$ is split into M equal-sized patches which are converted into a sequence of embeddings $\tilde{\boldsymbol{x}} = \{\boldsymbol{e}_{cls}, \boldsymbol{e}_1, \boldsymbol{e}_2, \ldots, \boldsymbol{e}_M\}$. The visual input sequence $\tilde{\boldsymbol{x}}$ is encoded by the image encoder, producing the image feature $\boldsymbol{f} = \mathcal{F}(\tilde{\boldsymbol{x}})$. On the text side, the text label y and the associated name is formatted as "`A photo of [STH]`" and tokenized into a sequence of tokens $\tilde{\boldsymbol{y}} = \{\boldsymbol{t}_{bos}, \boldsymbol{t}_1, \boldsymbol{t}_2, \ldots, \boldsymbol{t}_L, \boldsymbol{t}_{eos}\}$, where L is the length of input tokens. The input sequence is then encoded into $\boldsymbol{g} = \mathcal{G}(\tilde{\boldsymbol{y}})$. For image classification, target class labels $\{1, 2, \ldots, C\}$ are encoded into text features g_i. The classification is done by picking the class that has the highest similarity with the vision feature: $\hat{y} = \operatorname{argmax}_i \mathcal{C}(\boldsymbol{f}, \boldsymbol{g}_i)$, where $\mathcal{C}$ is the softmax cosine-similarity function $\mathcal{C}(\boldsymbol{f}, \boldsymbol{g}) = \frac{\exp(\boldsymbol{f}\cdot\boldsymbol{g}/\tau)}{\sum_{j=1}^{C}\exp(\boldsymbol{f}\cdot\boldsymbol{g}_j/\tau)}$ with temperature τ.

Multimodal Prompting Learning. Given the size of the CLIP model, fine-tuning the entire model becomes infeasible. As both image and text encoders are built with standard transformer architecture, prompt learning, which tunes the model by combining trainable prompts with hidden states has been applied on the text encoder [48, 49], the image encoder [15, 41, 42], or both [18, 19, 33]. Similar to [19, 33], we build our method following the vision-language prompting paradigm, with deep prompting [15, 19], which not only insert prompts to the input layer, but to later encoder layers.

More specifically, for each transformer layer that takes prompts, we define V learnable visual prompts $\boldsymbol{p}_v = \{p_v^1, p_v^2, \ldots, p_v^V\}$ and T learnable language prompts $\boldsymbol{p}_t = \{p_t^1, p_t^2, \ldots, p_t^T\}$. For the i-th vision encoder layer, visual prompts $\boldsymbol{p}_v^i$ are appended to input embeddings: $\tilde{\boldsymbol{x}}_{\boldsymbol{p}}^i = \{\boldsymbol{e}_{cls}^i, \boldsymbol{e}_1^i, \boldsymbol{e}_2^i, \ldots, \boldsymbol{e}_M^i, \boldsymbol{p}_v^i\}$. The prompt-augmented vision feature, $\boldsymbol{f}_{\boldsymbol{p}} = \mathcal{F}(\tilde{\boldsymbol{x}}_{\boldsymbol{p}})$, is produced by jointly encoding prompts and the image. As the ViT [9] architecture in CLIP adopts the bi-directional attention mechanism, the placement of $\boldsymbol{p}_v$ has no effect on $\boldsymbol{f}_{\boldsymbol{p}}$. On the language side, prompts are concatenated with the input of the i-th text encoder: $\tilde{\boldsymbol{y}}_{\boldsymbol{p}}^i = \{\boldsymbol{t}_{bos}^i, \boldsymbol{p}_t^i, \boldsymbol{t}_1^i, \boldsymbol{t}_2^i, \ldots, \boldsymbol{t}_L^i, \boldsymbol{t}_{eos}^i\}$. $\tilde{\boldsymbol{y}}_{\boldsymbol{p}}$ is further processed by the text encoder, resulting in the prompt-augmented language feature $\boldsymbol{g}_{\boldsymbol{p}} = \mathcal{G}(\tilde{\boldsymbol{y}}_{\boldsymbol{p}})$. More specifically, prompts to the first layer $\boldsymbol{p}_t^1$ are initialized with the embeddings of "A photo of a".

Low-Rank Adaptation [13] (LoRA). As a parameter-efficient tuning technique, LoRA is designed to adapt large transformer model without updating original model weights. The LoRA technique is, in particular, applied to linear projection layers. More specifically, for a linear layer with weight $W_0 \in \mathbb{R}^{d\times k}$, LoRA creates ΔW by learning two low rank matrices $B \in \mathbb{R}^{d\times r}$ and $A \in \mathbb{R}^{r\times k}$:

$$\boldsymbol{h} = (W_0 + \Delta W)\boldsymbol{x} = W_0\boldsymbol{x} + BA\boldsymbol{x}. \tag{1}$$

We adopt a hybrid tuning scheme on the vision encoder, which performs prompt learning on the first few layers and applies LoRA on the rest.

3.2. Adaptive Prompt Learning with LLMs

The goal of prompt tuning is to find a set of optimal prompts $\boldsymbol{p} = \{\boldsymbol{p}_v, \boldsymbol{p}_t\}$ which maximizes the log likelihood of $P(x, y|\theta_{\mathcal{F}}, \theta_{\mathcal{G}})$ over target downstream distribution $(\boldsymbol{x}, \boldsymbol{y}) \sim (\boldsymbol{X}, \boldsymbol{Y})$:

$$\boldsymbol{p} = \underset{\boldsymbol{p}}{\operatorname{argmax}}\, \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim(\boldsymbol{X},\boldsymbol{Y})} \log \mathcal{C}(\mathcal{F}(\boldsymbol{x}; \boldsymbol{p}_v), \mathcal{G}(\boldsymbol{y}; \boldsymbol{p}_t)) \tag{2}$$

However, the $\boldsymbol{p}$ optimized through Eqn. 2 has two issues. First, $\boldsymbol{p}$ is shared for all categories for the downstream task, while the optimal prompt for each category might be different. Second, in low-shot scenarios, $\boldsymbol{p}$ are usually empirically estimated from a limited training-set $\boldsymbol{X}^{train}$ with limited categories $\{1, 2, ..., C^{base}\}$, and therefore such $\boldsymbol{p}$ can often be over-fitted to the small training-set $\boldsymbol{X}^{train}$ and fail to generalize to novel categories outside $\{1, 2, ..., C^{base}\}$.

To overcome these problems, we propose to learn a meta function on the language side $\boldsymbol{p}_t = \Theta(y)$ which can adaptively estimate the optimal prompt for each category. An intuitive way to estimate proper prompts $\boldsymbol{p}$ for category name y is to take advantage of the knowledge of the pre-trained Large Language Models (LLM) $\mathcal{D}$ and extract discriminative descriptions of category y. For example, given the input text $\boldsymbol{z}$:"Describe {y}",

$$\boldsymbol{p}_t = \{p_1, p_2, ..., p_k\} = \mathcal{D}(\boldsymbol{z}). \tag{3}$$

while p_i being sequentially generated by $\mathcal{D}$ such that

$$\begin{aligned} p_i &= \mathcal{D}(\boldsymbol{z}, t_1, ..., t_{i-1}) = \mathcal{D}^{(i)}(\boldsymbol{z}) \\ t_i &= \mathcal{M}(p_i), \end{aligned} \tag{4}$$

where $\mathcal{D}^{(i)}$ is the i-th forward iteration of $\mathcal{D}$, and $\mathcal{M}$ maps continuous hidden states into discrete language tokens. To accelerate the process and to obtain p in one pass, we approximate the above process with K learnable prompts $\boldsymbol{p}_l = \{\theta_1, ..., \theta_K\}$ so that

$$\boldsymbol{p}_t = \Theta(y) = \mathcal{D}(\{\theta_1, ..., \theta_K\}|\boldsymbol{z}) \tag{5}$$

Discussion. While Large Language Models (LLMs) possess robust foundational knowledge within the linguistic domain, it is not feasible to directly substitute the text encoder of CLIP with an LLM. The reason lies in the inherent divergence between the LLM's latent space, which

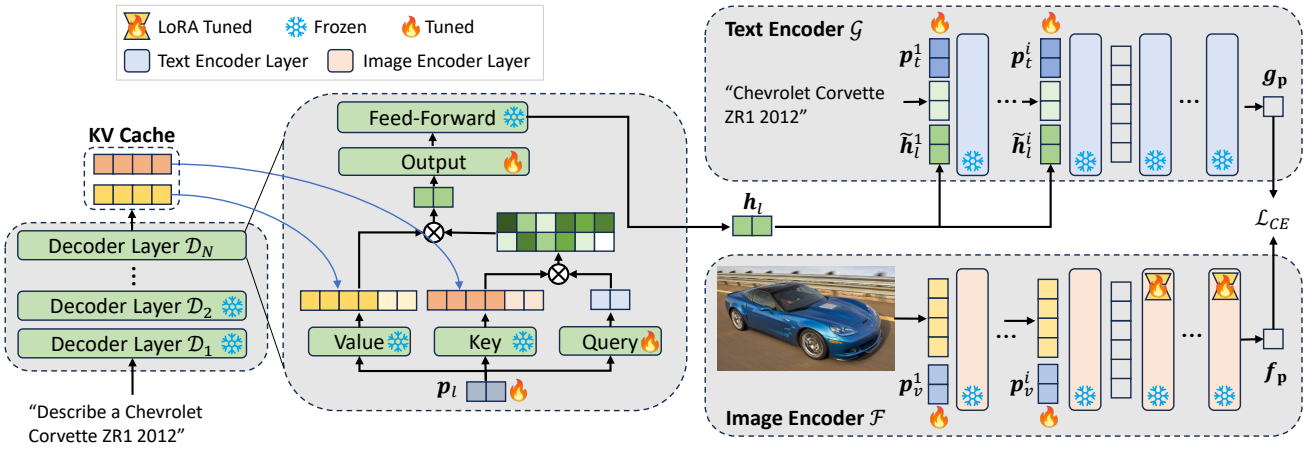


Figure 2. An Overview of the LLaMP Framework: We first generate the knowledge cache by passing the query prompt through the LLM $\mathcal{D}$ and use the knowledge cache to encode $\boldsymbol{p}_l$, resulting the adaptive prompts $\tilde{\boldsymbol{h}}_l^i = W\boldsymbol{h}_l^i + b^i$ for the CLIP text encoder. $\tilde{\boldsymbol{h}}_l$ is combined with regular learnable prompts of $\mathcal{G}$ to generate the final text feature vector $\boldsymbol{g_p}$. The image feature vector $\boldsymbol{f_p}$ is obtained through a hybrid-tuning strategy combining prompt learning and low-rank adaptation (LoRA).

is purely language-oriented, and the image-focused latent space of vision encoders. Attempting a direct alignment via contrastive learning would require an extensive dataset that is typically beyond the scope of low-shot learning. To bridge this gap, we introduce LLaMP—an adaptive prompt learning framework that leverages the LLM to craft class-specific prompt vectors, to reinforce the text encoder for low-shot image classification.

3.3. The LLaMP Framework

Fig. 2 shows an overview of the LLaMP framework. For convenience, we denote the decoder-only LLM as $\mathcal{D}$. The input to the decoder $\mathcal{D}$ consists of two components: textual prompts $\boldsymbol{y}$ in the form of sentences, tokenized as $\tilde{\boldsymbol{y}}$, and learnable prompts $\boldsymbol{p}_l$. We append prompt embeddings to the end of the input sequence and obtain the last hidden states of $\mathcal{D}$ as the feature $\boldsymbol{h}_l$:

$$\boldsymbol{h}_l = \mathcal{D}(\tilde{\boldsymbol{y}}, \boldsymbol{p}_l)[L+1 : L+K], L = \text{Length}(\tilde{\boldsymbol{y}}). \quad (6)$$

Hidden states of $\mathcal{D}$ are then mapped to the input space of the CLIP text encoder by the projection matrix $W \in \mathbb{R}^{d_1 \times d_2}$, where d_1 and d_2 are respectively the hidden sizes of the LLM $\mathcal{D}$ and the CLIP text encoder $\mathcal{G}$. A set of prompt-specific biases $b \in \mathbb{R}^{K \times d_2}$ are added:

$$\tilde{\boldsymbol{h}}_l = W\boldsymbol{h}_l + b \quad (7)$$

We combine $\tilde{\boldsymbol{h}}_l$ from LLM with regular learnable prompts, as in previous approaches [19], to construct the input for CLIP text encoder. Similar to deep prompting [15, 19], we create layer-specific prompts through different W matrices and b vectors. For the i-th layer, we let $\tilde{\boldsymbol{h}}_l^i = W^i\boldsymbol{h}_l + b^i$ and the entire sequence is constructed as

$$\tilde{\boldsymbol{y}}_l^i = \{\boldsymbol{t}_{bos}^i, \boldsymbol{p}_t^i, \boldsymbol{t}_1^i, \boldsymbol{t}_2^i, \ldots, \boldsymbol{t}_L^i, \tilde{\boldsymbol{h}}_l^i, \boldsymbol{t}_{eos}^i\} \quad (8)$$

LLM Knowledge Cache. A Large Language Model (LLM), as implied by its name, typically comprises billions of parameters. For example, the most compact LLaMA [38, 39] model has 7B parameters. Thus, even performing prompt learning on a LLM become impractical. The memory consumption to store gradients for back propagation can go beyond the limit of mainstream GPUs. Instead, the causal attention mechanism inherent in decoder-only LLMs, where the embedding of an input token only depends on the preceding tokens, facilitates a feasible workaround.

As previously mentioned, the prompt embeddings $\boldsymbol{p}_l$ are appended to the end of text tokens $\tilde{\boldsymbol{y}}$. According to the causal attention mechanism, $\tilde{\boldsymbol{y}}$ is encoded independently of $\boldsymbol{p}_l$. Thus, we design a two-stage process, where we create the LLM knowledge cache by passing $\tilde{\boldsymbol{y}}$ through $\mathcal{D}$ and leverage the cache to convert $\boldsymbol{p}_l$ into class-specific embeddings for the CLIP text encoder $\mathcal{G}$.

To compute the attention of a token, the only dependency is the *Key* and *Value* vectors from the preceding tokens. Thus, we adopt the KV-cache [31, 43], a technique used in inference acceleration of LLMs, to create the knowledge cache. At the first stage, we pass text tokens $\tilde{\boldsymbol{y}}$ through the language model $\mathcal{D}$ and save the *Keys* and *Values* as the knowledge cache for the second stage. Once computed, the knowledge cache remains fixed throughout the entire training process and bears the information that is needed for further computation. Thus, in LLaMP, we leverage the knowledge cache obtained at the first stage to generate class-specific prompt embeddings.

At the second stage, we create class-specific prompt embeddings from the pre-computed knowledge cache. As $\boldsymbol{p}_l$ is not initialized in the natural language domain, it need not pass through the entire LLM; instead, we insert those prompts $\boldsymbol{p}_l$ to the last layer of the LLM $\mathcal{D}_N$. It is achieved

by encoding them alongside the cache from $\tilde{\boldsymbol{y}}$, as in

$$\boldsymbol{H}_l = \mathcal{D}_N(\boldsymbol{K}_{\tilde{\boldsymbol{y}}}, \boldsymbol{V}_{\tilde{\boldsymbol{y}}}, \boldsymbol{p}_l), \tag{9}$$

where $\boldsymbol{K}_{\tilde{\boldsymbol{y}}}, \boldsymbol{V}_{\tilde{\boldsymbol{y}}}$ represent the knowledge cache. This design enables LLaMP to efficiently learn informative prompt embeddings for the CLIP encoder $\mathcal{G}$. It accomplishes this by incurring modest training costs, compared with training the entire LLM. Simultaneously, it maintains the essential knowledge inherent in the LLM decoder $\mathcal{D}$.

Training Targets of LLaMP. Although the training strategy in Eqn. 9 has reduced the number of learnable parameters, a full decoder inside a LLM still consists of an enormous number of parameters. For example, one layer in LLaMA-7B bears 200M parameters, making training of the entire layer costly. As the goal is to leverage the knowledge from LLM, altering a full layer can lead to the loss of knowledge. Thus, as shown in Fig. 2, a typical decoder layer has two major components: the self attention module, consisting *Query*, *Key*, *Value* and *Output* projection layers, and the Feed-Forward Network (FFN). LLaMP targets the *Query* and *Output* projection layer inside the self-attention module. By updating the *Query* layer, LLM prompts $\boldsymbol{p}_l$ are learned to distill pertinent information from the knowledge cache and the *Output* layer projects it to the latent space. We keep the *Key* and *Value* layers frozen to ensure the alignment between $\boldsymbol{p}_l$ and knowledge cache. We leave the FFN unchanged to preserve the knowledge. Further discussions regarding these choices are made in Sec. 4.3.

Textual Priors from Pre-Generated Responses. We extend the initial prompt, "`In one sentence, describe the distinctive appearance of [STH]`", by incorporating the response generated by the language model into the input sequence. This approach enriches the base content: the generated text provides a clear and explicit description of the object's appearance, acting as a valuable informative prior for language model adaptation. However, it is common for responses from an LLM to include filler words like "sure" for sentence structure coherence. To refine the input, we parse the noun phrases from the LLM's response through spaCy [12], an NLP engine, and merge them with the initial prompt, forming a more focused and informative language prior.

Textual Augmentations. Following the insights of Khattak et al. [19], which highlight the performance benefits of diverse textual inputs, we aim to further augment the text inputs used in the CLIP text encoder. Our approach, building upon the methods in [19, 48], incorporates handcrafted templates and expands their diversity through a two-step process: i) We introduce noun phrases into the existing templates for CLIP, for example, transforming "`A photo of [STH]`" to "`A photo of [STH] with [NP]`", thereby enriching the descriptive content; ii) We create a variety of new prompt templates for the LLM similar to "`In one sentence, describe the distinctive appearance of [STH]`" through GPT-4 [28], to further diversify the text input.

3.4. Training and Inference

Similar to PSRC [19], our objective function consists of three components: The main cross-entropy loss $\mathcal{L}_{CE}$, feature-level L1 regularization $\mathcal{L}_{l1}$, and soft distillation loss $\mathcal{L}_{dist}$. Given $\mathcal{C}$ training categories and $\mathcal{N}$ training samples, $\mathcal{L}_{CE}$ is defined as

$$\mathcal{L}_{CE} = -\frac{1}{\mathcal{N}} \sum_i \log \frac{\exp\left(\boldsymbol{f}_{\boldsymbol{p}}^i \cdot \boldsymbol{g}_{\boldsymbol{p}}^i / \tau\right)}{\sum \exp\left(\boldsymbol{f}_{\boldsymbol{p}}^i \cdot \boldsymbol{g}_{\boldsymbol{p}}^j / \tau\right)}. \tag{10}$$

The L1 regularization is computed between learned features $\boldsymbol{f}_{\boldsymbol{p}}, \boldsymbol{g}_{\boldsymbol{p}}$ and pre-trained CLIP features $\hat{\boldsymbol{f}}, \hat{\boldsymbol{g}}$:

$$\mathcal{L}_{l1} = \frac{1}{\mathcal{N}} \sum_i \lambda_v |\boldsymbol{f}_{\boldsymbol{p}}^i - \hat{\boldsymbol{f}}^i| + \frac{1}{C} \sum_i \lambda_t |\boldsymbol{g}_{\boldsymbol{p}}^i - \hat{\boldsymbol{g}}^i|, \tag{11}$$

where λ_v and λ_t are coefficients. The prediction of LLaMP is further bound by the KL-Divergence between predicted distributions of LLaMP and vanilla CLIP:

$$\mathcal{L}_{dist} = \lambda_{dist} D_{KL}(\boldsymbol{f}_{\boldsymbol{p}} \cdot \boldsymbol{g}_{\boldsymbol{p}}, \hat{\boldsymbol{f}} \cdot \hat{\boldsymbol{g}}). \tag{12}$$

We sum all three losses up as the final objecttive function: $\mathcal{L} = \mathcal{L}_{CE} + \mathcal{L}_{l1} + \mathcal{L}_{dist}$.

During training, we randomly sample one LLM template as the input of LLaMP for each batch. For inference, we compute the probability distribution predicted from each input template and average them.

4. Experiments

4.1. Experiment Setup

Datasets. Similar to previous work [18, 19, 48], in our study, we evaluate LLaMP performance over a spectrum of classification tasks with 11 datasets, including ImageNet [8] and Caltech101 [10] for generic image classification, OxfordPets [29], StanfordCars [21], Flowers102 [26], Food101 [3], and FGVCAircraft [25] for fine-grained classification, SUN397 [45] for scene recognition, UCF101 [37] for action recognition, DTD [7] for texture classification, and EuroSAT [11] for satellite image recognition.

Scenarios & Metrics. We evaluate LLaMP on two typical low-shot scenarios: zero-shot base-to-novel generalization and few-shot image classification. In zero-shot base-to-novel generalization, the base classes are seen during training, while the novel classes are unseen. We measure models performance through accuracies of base and novel classes, and the harmonic mean of the two. For few-shot classification, we assess the accuracy with 16 shots per class.

Implementation Details. We build LLaMP through the PyTorch [30] framework. All models are trained with

Method	*Average*			ImageNet [8]			Caltech101 [10]			OxfordPets [29]		
	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM
CLIP [32]	69.34	74.22	71.70	72.43	68.14	70.22	96.84	94.00	95.40	91.17	97.26	94.12
CoOp [49]	82.69	63.22	71.66	76.47	67.88	71.92	98.00	89.81	93.73	93.67	95.29	94.47
CoCoOp [48]	80.47	71.69	75.83	75.98	70.43	73.10	97.96	93.81	95.84	95.20	97.69	96.43
KAPT* [17]	78.41	70.52	74.26	71.10	65.20	68.02	97.10	93.53	95.28	93.13	96.53	94.80
ProDA [24]	81.56	72.30	76.65	76.66	70.54	73.47	97.74	94.36	96.02	95.43	97.76	96.58
MaPLe [18]	82.28	75.14	78.55	75.40	70.32	72.72	98.27	93.23	95.68	95.43	**97.83**	96.62
RPO [23]	81.13	75.00	77.78	76.60	**71.57**	74.00	97.97	94.37	96.03	94.63	97.50	96.05
PSRC [19]	84.26	76.10	79.97	77.60	70.73	74.01	98.10	94.03	96.02	95.33	97.30	96.30
LLaMP	**85.16**	**77.71**	**81.27**	**77.99**	71.27	**74.48**	**98.45**	**95.85**	**97.13**	**96.31**	97.74	**97.02**
Δ w.r.t. PSRC	+0.90	+1.61	+1.30	+0.39	+0.54	+0.47	+0.35	+1.82	+1.11	+0.98	+0.44	+0.72

Method	StanfordCars [21]			Flowers102[26]			Food101 [3]			FGVCAircraft [25]		
	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM
CLIP [32]	63.37	74.89	68.65	72.08	**77.80**	74.83	90.10	91.22	90.66	27.19	36.29	31.09
CoOp [49]	78.12	60.40	68.13	97.60	59.67	74.06	88.33	82.26	85.19	40.44	22.30	28.75
CoCoOp [48]	70.49	73.59	72.01	94.87	71.75	81.71	90.70	91.29	90.99	33.41	23.71	27.74
KAPT* [17]	69.47	66.20	67.79	95.00	71.20	81.40	86.13	87.06	86.59	29.67	28.73	29.19
ProDA [24]	72.94	74.00	73.47	95.92	72.46	82.56	90.71	**92.05**	91.38	37.44	35.61	36.50
MaPLe [18]	74.70	71.20	72.91	97.70	68.68	80.66	90.30	88.57	89.43	36.90	34.13	35.46
RPO [23]	73.87	**75.53**	74.69	94.13	76.67	84.50	90.33	90.83	90.58	37.33	34.20	35.70
PSRC [19]	78.27	74.97	76.58	**98.07**	76.50	85.95	90.67	91.53	91.10	42.73	**37.87**	40.15
LLaMP	**81.56**	74.54	**77.89**	97.82	77.40	**86.42**	**91.05**	91.93	**91.49**	**47.30**	37.61	**41.90**
Δ w.r.t. PSRC	+3.29	-0.43	+1.31	-0.25	+0.90	+0.47	+0.38	+0.40	+0.39	+4.57	-0.26	+1.75

Method	SUN397 [45]			DTD [7]			EuroSAT [11]			UCF101 [37]		
	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM	Base	Novel	HM
CLIP [32]	69.36	75.35	72.23	53.24	59.90	56.37	56.48	64.05	60.03	70.53	77.50	73.85
CoOp [49]	80.60	65.89	72.51	79.44	41.18	54.24	92.19	54.74	68.69	84.69	56.05	67.46
CoCoOp [48]	79.74	76.86	78.27	77.01	56.00	64.85	87.49	60.04	71.21	82.33	73.45	77.64
KAPT* [17]	79.40	74.33	76.78	75.97	58.30	65.97	84.80	67.57	75.21	80.83	67.10	73.33
ProDA [24]	80.82	78.70	79.75	80.36	59.18	68.16	**94.07**	73.23	82.35	83.00	78.66	80.77
MaPLe [18]	78.47	76.93	77.79	80.67	56.48	66.44	83.90	66.00	73.88	85.23	71.97	78.04
RPO [23]	80.60	77.80	79.18	76.70	62.13	68.61	86.63	68.97	76.79	83.67	75.43	79.34
PSRC [19]	82.67	78.47	80.52	83.37	62.97	71.75	92.90	73.90	82.32	87.10	78.80	82.74
LLaMP	**83.41**	**79.90**	**81.62**	**83.49**	**64.49**	**72.77**	91.93	**83.66**	**87.60**	**87.13**	**80.66**	**83.77**
Δ w.r.t. PSRC	+0.74	+1.43	+1.10	+0.12	+1.52	+1.02	-0.97	+9.76	+5.28	+0.03	+1.86	+1.03

Table 1. **Comparison with state-of-the-art methods on base-to-novel generalization**. LLaMP shows strong generalization results over previous approaches on 11 image classification tasks. Absolute gains over PSRC are indicated in blue. *KAPT is trained with ViT-B/32 image encoder instead of ViT-B/16.

2 NVIDIA A100 40GB GPUs. For LLaMP, we adopt LLaMA2-7B [39] as the language model $\mathcal{D}$, and ViT-B/16 [9] as the image encoder, following [18, 19, 48, 49]. On the text side, we set prompt learning depth to 9. To tune the vision encoder, we adopt the hybrid tuning scheme which performs deep prompt learning on the first 6 layers and LoRA on the rest. Similar to [13], LoRA is applied to the *Query* and *Value* projection layers inside attention modules. The number of $\boldsymbol{p}_l$ prompts, K, is set to 16. We set a global learning rate of 2E-4 with a batch size of 8. The learning rate of LoRA modules is set to 2E-5. λ_t, λ_v and λ_{dist} are set to 25, 10 and 2.5, respectively.

4.2. Quantitative Evaluation

Zero-Shot Base-to-Novel Generalization. LLaMP outperforms existing state-of-the-art prompt learning methods on most metrics of 11 classification datasets in the base-to-novel generalization benchmark. As shown in Tab. 1, compared to the latest model PSRC [19], LLaMP achieves average gains of 0.90% in base accuracy, 1.61% in novel

	16-Shot Classification											
	Average	ImageNet [8]	Caltech [10]	Pets [29]	Cars [21]	Flowers [26]	Food [3]	Aircraft [25]	SUN397 [45]	DTD [7]	EuroSAT [11]	UCF101 [37]
CLIP [32]	78.79 (65.02)	67.31	95.43	85.34	80.44	97.37	82.90	45.36	73.28	69.96	87.21	82.11
CoOp [49]	79.89 (73.82)	71.87	95.57	91.87	83.07	97.07	84.20	43.40	74.67	69.87	84.93	82.23
CoCoOp [48]	74.90 (70.70)	70.83	95.16	93.34	71.57	87.84	87.25	31.21	72.15	63.04	73.32	78.14
MaPLe [18]	81.79 (75.58)	72.33	96.00	92.83	83.57	97.00	85.33	48.40	75.53	71.33	92.33	85.03
PSRC [19]	82.87 (77.90)	73.17	96.07	93.67	83.83	97.60	87.50	50.83	**77.23**	72.73	**92.43**	86.47
LLaMP	**83.81** (**78.50**)	**73.49**	**97.08**	**94.21**	**86.07**	**98.06**	**87.62**	**56.07**	77.02	**74.17**	91.31	**86.84**

Table 2. Few shot classification results with 16 shots. Numbers in the bracket indicate the average performance over 1/2/4/8/16 shots.

accuracy, and 1.30% in harmonic mean on average. Moreover, LLaMP consistently achieves higher harmonic means (HM) compared to other models. These improvements indicate that our approach better balances performance on base and novel data, thus achieving stronger generalization compared to the prior prompt learning techniques.

In particular, LLaMP excels in fine-grained datasets requiring detailed analysis. On *FGVCAircraft*, LLaMP surpasses PSRC by 4.57% on base accuracy and 1.75% on HM, highlighting its strong understanding of detailed aircraft features. Furthermore, on *EuroSAT*, LLaMP achieves improvements of 9.76% and 5.28% on novel accuracy and HM, respectively. We also observe similar performance gains on *StanfordCars*, where LLaMP outperformns by 3.29% on base accuracy and 1.31% on HM. The information embedded in LLM enables LLaMP to capture and utilize the rich semantic information necessary for distinguishing between closely related categories.

Few-Shot Classification. LLaMP also achieves improvements across these classification datasets in few-shot classification tasks. As in Tab.2, with an average classification accuracy of 83.81%. Notably, on *FGVCAircraft* and *StanfordCars*, LLaMP shows a significant improvement over PSRC, further demonstrating that the knowledge from language models benefits the recognition of fine-grained object categories, which aligns with our observation on zero-shot base-to-novel generalization. Moreover, on `DTD`, where MaPLe and PSRC achieve around 72% accuracy, LLaMP achieves a higher accuracy of 74.17%, underscoring its ability to recognize textures.

4.3. Ablation Study

Is the knowledge from LLM helping? In Tab. 3, we show that the knowledge from LLM benefits in both ways: Without training, performance of ordinary CLIP model can be improved by introducing noun phrases; The LLaMP framework shows further improvement after training.

Noun phrases are parsed from the LLM's responses the prompt of "`Describe [STH]`". We then use the template, "`A photo of [STH] with [NP]`" to generate the NP-augmented text embedding for CLIP. We take the average of all augmented embeddings for classification. In Tab. 3 we show that even ordinary CLIP can benefit from incorporating LLMs' knowledge.

Method	LLM	Base	Novel	HM
CLIP		69.34	74.22	71.70
	✓	70.95	74.93	72.79
LLaMP		82.21	76.44	79.22
	✓	**85.16**	**77.71**	**81.27**

Table 3. Ablation study on the LLM Knowledge.

LP	QO	KV	FFN	%	Base	Novel	HM
✓				.03	85.00	77.29	80.96
✓	✓	✓		33	85.20	77.45	81.14
✓	✓		✓	83	85.05	**77.73**	81.22
✓	✓	✓	✓	100	**85.23**	77.56	81.21
✓	✓			17	85.16	77.71	**81.27**

Table 4. Ablation study on the Training Strategy. "%" indicates the ratio of parameters trained compared to fully tuning a layer.

Furthermore, the comparison between LLaMP and LLaMP without the LLM indicates that merely integrating LoRA [13] to the vision encoder is not beneficial. The "LLaMP without LLM" is essentially an ordinary prompting learning model plus LoRAs in the vision encoder. We show that the improved vision encoding capacity only benefits when the quality of text embeddings s are enhanced by incorporating LLMs' knowledge through LLaMP.

Decoder Training strategy. We categorize trainable parameters of $\mathcal{D}_N$ into four groups: learnable prompts (LP), *Query* and *Output* projections (QO), *Key* and *Value* projections (KV), and the feed-forward network (FFN). Tab. 4 indicates LLaMP can achieve desirable results by just learning the prompts of $\mathcal{D}$. One step further, adding QO into optimization achieves the best performance. Although other

Method	Priors	Base	Novel	HM
LLaMP	✗	84.90	77.59	81.08
	Plain	**85.26**	77.56	81.22
	NP	85.16	**77.71**	**81.27**

Table 5. Ablation Study on Pre-generated Text Priors. ✗ refers to "without textual priors" and NP stands for noun phrases.

Method	Base	Novel	HM
LLM Only	81.74	35.82	49.81
LLaMP	**85.16**	**77.71**	**81.27**

Table 6. The CLIP text encoder helps adaptation.

Scheme	Base	Novel	HM
Prompt × 9	84.67	77.28	80.81
LoRA × 12	84.89	77.27	80.90
Prompt ×6 + LoRA ×6	**85.16**	**77.71**	**81.27**

Table 7. Study on Vision Tuning Scheme. Our hybrid design achieves the best performance.

setups introduce much more trainable parameters, they can not surpass the "LP + QO" strategy.

Effect of Textual Priors. We study the effect of pre-generated textual priors on LLaMP. We compare three different approaches: without textual priors, using plain responses as the prior, and LLaMP, which takes parsed noun phrases. Tab. 5 shows that LLaMP can achieve over 81% on HM without pre-generated priors, while adding parsed noun phrases as textual priors further pushes the HM to 81.27%.

CLIP as the bridge. One may wonder if it is possible to replace CLIP text encoder with a large language model. Here, we study two setups: i) LLM as encoder, which treats the output of the language model, $\tilde{\boldsymbol{h}}_l$ as the text feature; ii) LLaMP, which treat $\tilde{\boldsymbol{h}}_l$ as part of the text input prompt. Tab. 6 reveals that relying solely on the LLM results in poor accuracy for novel categories. This supports our hypothesis that aligning LLMs with vision encoders generally requires a more extensive dataset. Furthermore, LLaMP's design significantly improve the novel accuracy by 40%.

Vision Training Strategy. As ViT-16/B has 12 transformer layers, we compare different vision training strategies within LLaMP in Tab. 7. Apart from the default hybrid scheme, we evaluate setups including prompt learning at first 9 layers (P9), a similar setup to PSRC [19], and LoRA [13] in all 12 layers. The results suggest that the scheme leverages the strengths of both prompt learning and LoRA, addressing potential bottlenecks in the vision encoder and enhancing overall performance in LLaMP.

Number of LLM Prompts. We vary the number of LLM prompts and study their effects on LLaMP. As in Fig. 3, using 16 prompts optimizes the LLM's capabilities, achieving the highest harmonic mean at 81.27%.

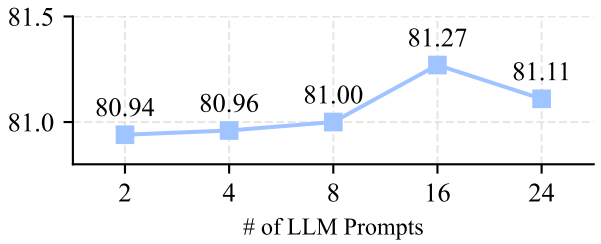


Figure 3. Effect of LLM Prompts on Harmonic Mean. 16 prompts achieve the most balanced performance.

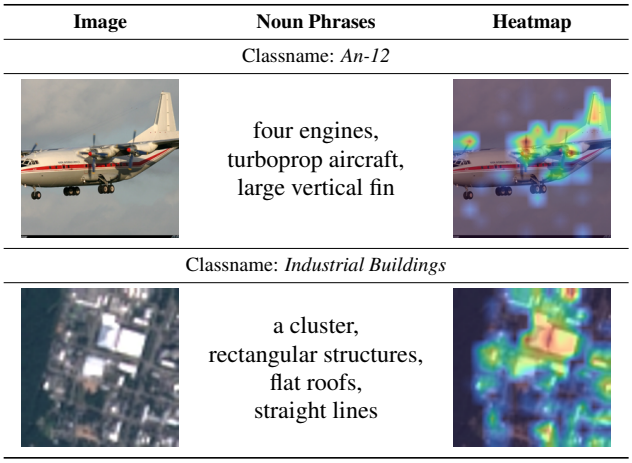


Figure 4. Visualization of LLaMP Predictions by GradCAM [35]

Visualizations. In Fig. 4, we visualize the gradient heatmap of input images from FGVCAircraft and EuroSAT, through GradCAM [35]. The figure shows that, LLaMP can capture distinctive features that matches LLM's description.

5. Conclusion

Our study shows that the encyclopedic knowledge from LLMs is beneficial for low-shot image classification as extra class-specific information. To leverage such knowledge, we propose LLaMP, a framework that adapts LLMs as prompt learners for the CLIP model. Over two common low-shot scenarios: zero-shot generalization and few-shot learning, LLaMP demonstrates notable improvements compared with previous state-of-the-arts on a spectrum of datasets.

Limitations. While LLaMP reveals an effective way in leveraging LLMs' knowledge, both modalities, vision and language, only interact at the finest feature level. Given the broader LLM-aided knowledge from the language side, the performance can be potentially further improved by introducing language priors at earlier vision encoding stages.

Acknowledgment

This research was supported, in part, by the Office of Naval Research under grant #N00014-21-1-2802.

References

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Merouane Debbah, Etienne Goffinet, Daniel Heslow, Julien Launay, Quentin Malartic, et al. Falcon-40b: an open large language model with state-of-the-art performance. Technical report, Technical report, Technology Innovation Institute, 2023. 2

[2] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023. 2

[3] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *ECCV*, pages 446–461. Springer, 2014. 5, 6, 7

[4] Shiming Chen, Wenjie Wang, Beihao Xia, Qinmu Peng, Xinge You, Feng Zheng, and Ling Shao. Free: Feature refinement for generalized zero-shot learning. In *ICCV*, pages 122–131, 2021. 2

[5] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, 2023. 2

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022. 2

[7] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *CVPR*, pages 3606–3613, 2014. 5, 6, 7

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255. Ieee, 2009. 5, 6, 7

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 3, 6

[10] 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 *CVPR workshop*, pages 178–178. IEEE, 2004. 5, 6, 7

[11] 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. 5, 6, 7

[12] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020. 5

[13] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021. 3, 6, 7, 8

[14] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, pages 4904–4916. PMLR, 2021. 1, 2

[15] 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. 3, 4

[16] Michael Kampffmeyer, Yinbo Chen, Xiaodan Liang, Hao Wang, Yujia Zhang, and Eric P Xing. Rethinking knowledge graph propagation for zero-shot learning. In *CVPR*, pages 11487–11496, 2019. 2

[17] Baoshuo Kan, Teng Wang, Wenpeng Lu, Xiantong Zhen, Weili Guan, and Feng Zheng. Knowledge-aware prompt tuning for generalizable vision-language models. In *ICCV*, pages 15670–15680, 2023. 1, 2, 6

[18] Muhammad Uzair Khattak, Hanoona Rasheed, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Maple: Multi-modal prompt learning. In *CVPR*, pages 19113–19122, 2023. 1, 2, 3, 5, 6, 7

[19] Muhammad Uzair Khattak, Syed Talal Wasim, Muzammal Naseer, Salman Khan, Ming-Hsuan Yang, and Fahad Shahbaz Khan. Self-regulating prompts: Foundational model adaptation without forgetting. In *ICCV*, pages 15190–15200, 2023. 1, 2, 3, 4, 5, 6, 7, 8

[20] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th ICLR, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. 2

[21] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV workshops*, pages 554–561, 2013. 5, 6, 7

[22] Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Attribute-based classification for zero-shot visual object categorization. *TPAMI*, 36(3):453–465, 2013. 2

[23] Dongjun Lee, Seokwon Song, Jihee Suh, Joonmyeong Choi, Sanghyeok Lee, and Hyunwoo J Kim. Read-only prompt optimization for vision-language few-shot learning. In *ICCV*, pages 1401–1411, 2023. 1, 2, 6

[24] Yuning Lu, Jianzhuang Liu, Yonggang Zhang, Yajing Liu, and Xinmei Tian. Prompt distribution learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5206–5215, 2022. 1, 6

[25] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013. 2, 5, 6, 7

[26] 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. 5, 6, 7

[27] OpenAI. Chatgpt, 2023. Available at https://openai.com/chatgpt. 2

[28] OpenAI. Gpt-4 technical report, 2023. 1, 2, 5

[29] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and CV Jawahar. Cats and dogs. In *CVPR*, pages 3498–3505. IEEE, 2012. 5, 6, 7

[30] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, pages 8026–8037, 2019. 5

[31] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023. 4

[32] 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 *ICML*, pages 8748–8763. PMLR, 2021. 1, 2, 6, 7

[33] Hanoona Rasheed, Muhammad Uzair Khattak, Muhammad Maaz, Salman Khan, and Fahad Shahbaz Khan. Fine-tuned clip models are efficient video learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6545–6554, 2023. 3

[34] Scott Reed, Zeynep Akata, Honglak Lee, and Bernt Schiele. Learning deep representations of fine-grained visual descriptions. In *CVPR*, pages 49–58, 2016. 2

[35] 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 *ICCV*, pages 618–626, 2017. 8

[36] Richard Socher, Milind Ganjoo, Hamsa Sridhar, Osbert Bastani, Christopher D Manning, and Andrew Y Ng. Zero-shot learning through cross-modal transfer. *arXiv preprint arXiv:1301.3666*, 2013. 2

[37] 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. 5, 6, 7

[38] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 1, 2, 4

[39] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. 1, 2, 4, 6

[40] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *CVPR*, pages 6857–6866, 2018. 2

[41] Zifeng Wang, Zizhao Zhang, Sayna Ebrahimi, Ruoxi Sun, Han Zhang, Chen-Yu Lee, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, et al. Dualprompt: Complementary prompting for rehearsal-free continual learning. In *European Conference on Computer Vision*, pages 631–648. Springer, 2022. 3

[42] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister. Learning to prompt for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 139–149, 2022. 3

[43] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. 4

[44] Yongqin Xian, Tobias Lorenz, Bernt Schiele, and Zeynep Akata. Feature generating networks for zero-shot learning. In *CVPR*, pages 5542–5551, 2018. 2

[45] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. Sun database: Exploring a large collection of scene categories. *IJCV*, 119(1):3–22, 2016. 5, 6, 7

[46] Li Zhang, Tao Xiang, and Shaogang Gong. Learning a deep embedding model for zero-shot learning. In *CVPR*, pages 2021–2030, 2017. 2

[47] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022. 2

[48] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, pages 16816–16825, 2022. 1, 2, 3, 5, 6, 7

[49] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *IJCV*, 130(9):2337–2348, 2022. 1, 2, 3, 6, 7

[50] Yizhe Zhu, Mohamed Elhoseiny, Bingchen Liu, Xi Peng, and Ahmed Elgammal. A generative adversarial approach for zero-shot learning from noisy texts. In *CVPR*, pages 1004–1013, 2018. 2