Zero/Few-shot Foundation Models

Naren Rachapalli

1 Introduction

Both vision and language have been the main pillars of the development of machine learning, each involving numerous important tasks. For vision, people are interested in image classification, image segmentation, image generation, and image completion or inpainting. For language, people are interested in machine translation, text classification, part-of-speech tagging, and nowadays the popular chatbots or ChatGPT.

Meanwhile, deep learning has been the prominent method of tackling these problems in machine learning. At the root of the fast-paced and increasing progress is the availability of many open-source pre-trained models. A pre-trained model is usually trained beforehand on an enormous-scale dataset, learning the high-level concepts in a general sense. Then the researchers can deploy the pre-trained model on smaller individual tasks with fine-tuning. For vision, there are popular CNN-based architectures AlexNet, ResNet, GoogleNet, InceptionNet, etc, and transformer-based architectures, ViT, Swin, etc. These models are usually trained on ImageNet [1] with millions of images. For language, popular pre-trained models like GPT and BERT are usually trained on Wikipedia or Brown corpus.

Although vision and language have been two separate research branches for a while, that is not how humans perceive the world. Humans often understand language through visualization and describe images through text. On the other hand, though unlabeled, vision and language data exist everywhere on the internet nowadays. Hence, we have the vision-language foundation model, which is trained on this large amount of unlabeled data and can deal with both numerous vision-related and language-related tasks.

Foundation models often are trained on an enormous amount of internet data that is only achievable in large cooperates like Meta or OpenAI. Due to its power, the introduction of foundation models has greatly influenced existing research topics. One of them is zero/few-shot learning. Zero/Few-shot learning has been a popular topic by limiting the data resources a model can lay hands on when fine-tuning on new domain. However, since the pre-trained weights of foundation models have been exposed to an enormous amount of information, these newer domains do not exactly feel like fresh queries to them. For example, CLIP-ResNet50 [2] can achieve about 60% of top-1 accuracy without any fine-tuning on the ImageNet, i.e. under the zero-shot setting. As a powerful foundation model, CLIP has been applied to many cross-modality vision-language tasks, which are tasks that involve the interaction between vision data and language data, such as image captioning [3, 4], visual question answering [4], and vision-language navigation [4].

In this project, we pick CLIP as our target foundation model and would like to dive deep into (1) how CLIP can train with such an amount of data and (2) how a little bit of tweaking can make CLIP a powerful few-shot learner. We will introduce and discuss the details of CLIP and other selected CLIP-based few-shot learning methods, including CoOp [5], CoCoOP [6], ClipAdapter [7], TipAdapter [8], and CaFo [9]. We hope this project will provide us with a better understanding of how CLIP is trained and how large its impact is on modern machine learning.

Specifically, our goals are listed as follows,

- To understand the development of vision-language foundation models.
- To understand the development of few-shot learning for CLIP.
- To conduct and reproduce experiments on multiple CLIP-based few-shot learning methods.

2 Background: CLIP

We have previously discussed that we will be utilizing vanilla Zero shot CLIP [2] as our baseline model. Further, the other foundation models being studied in this project: CoOp, CoCoOp, TipAdapter, ClipAdapter, and CaFo all borrow from CLIP's existing architecture. We believe it is therefore important to outline the working details of the zero-shot CLIP model to lay a foundation for the remaining project.

The underlying philosophy of the CLIP architecture is that associating an image with a simple class label, as is the case for many classification tasks is inefficient in multiple ways. Firstly, the information in an image when mapped to a simple label is a massive oversimplification. If the below image were to be associated just with the label "Cat", we ignore the additional information in the picture such as the activity of lying down, the color of the cat, and various other embedded representations like age, breed, activity, objects being interacted with, etc.

In lectures, we are familiar with classification and regression learning. For example, we can train classification tasks with cross-entropy loss; we can train regression tasks with mean-square-error loss. However, we cannot use these two losses for vision-language data. We are given a large amount of image-text pairs. Yet, the relation between an image and a text cannot be easily expressed by a one-to-one mapping or some numbers. An image cannot be labeled with a single text. For example, Fig. 1 can have the caption "There is a cat" or "A cat is sleeping". On the other hand, a text cannot be labeled with a single image. For example, "There is a cat" can either be referencing Fig. 1 or a figure of a white cat.

Figure 1: **Picture of a sleeping cat surrounded by yellow flowers.**

Secondly, manually trying to associate this information in an image by generating very specific labels becomes extremely laborious and not scalable for large datasets. Even if this we ignore the above problem, very specific class labels like "Picture of a young orange cat sleeping in a field" are not transferable to new images which may be a separate combination of cat age, size, and location.

To combat this, CLIP architecture utilizes language-supervised contrastive loss training to learn efficient and meaningfully separated (in representation space) representations of image information which can then be used for zero-shot predictions in downstream tasks.

In contrastive learning, as shown in Fig. 2, we instead maximize the similarity score between image text feature pair (I_1, T_1) in comparison to (I_1, T_2), (I_1, T_3), etc. This is because I_1 and T_1 are a correspondent pair while the others are not. We note that this is different than a classification loss since the loss is dependent on the text in the batch $\{I_1, I_2, \cdots I_N\}$, not the total text space.

2.1 Architecture

We already mentioned that CLIP employs language-supervised contrastive learning. This is achieved by using Language-Image pairs to learn parameters for vision and text models.

Vision Model As shown in Fig. 2, CLIP employs a vision model based on a deep convolutional neural network (CNN) architecture, often utilizing ResNet or similar structures. This model transforms input images into high-dimensional embeddings capturing visual features.

$$\text{Vision Embedding: } I_1 = VisionModel(Image\ \#1) \tag{1}$$

Text Model The text model is built on transformer-based architectures like the Vision Transformer (ViT). This model processes input text, capturing semantic information and generating text embeddings.

$$\text{Text Embedding: } T_1 = TextModel(Text\ \#1) \tag{2}$$

Here, embeddings I_1, T_1 are obtained by passing an image and text pair $Image\#1, Text\#1$ into their respective vision and text models $VisionModel, TextModel$.

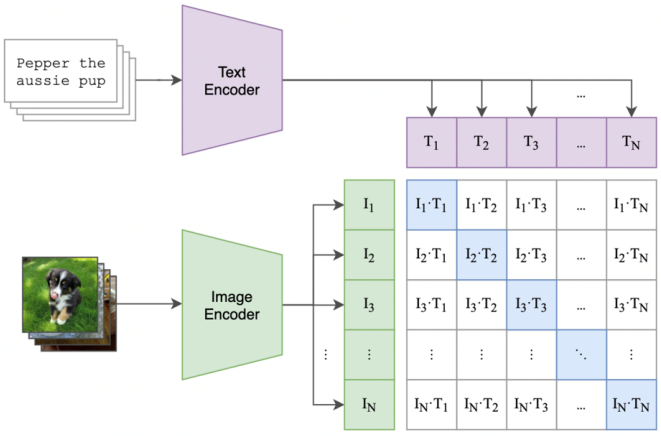

Figure 2: **Training architecture of CLIP model.** This figure is drawn by the authors of [2].

2.2 Contrastive Learning

The parameters θ, of both the encoders are found by minimizing a contrastive loss function.

This contrastive loss function encourages the model to assign higher similarity scores (or lower distances) to positive pairs and lower similarity scores (or higher distances) to negative pairs.

Mathematically speaking this can be expressed as:

$$L(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\log\left(\frac{\mathbb{E}(sim(v_i, t_i)/\tau)}{\sum_{j=1}^{N}\mathbb{E}(sim(v_i, t_j)/\tau)}\right) \tag{3}$$

In this equation:

- $L(\theta)$ represents the contrastive loss for a parameter collection θ
- N is the batch size
- $sim(v_i, t_j)$ denotes the similarity between the normalized embeddings from the image and text encoders, with $i = j$ representing the correctly associated image-text pair
- τ is a temperature parameter used to control the sharpness of the similarity scores

For CLIP, the similarity measure used is *cosine similarity*, defined as:

$$sim(u_i, v_j) = \frac{u_i^T v_j}{||u_i||\ ||v_j||}. \tag{4}$$

2.3 Dataset and training

The authors of the paper have not made CLIP's training dataset publicly available. CLIP has been trained on a custom image-text dataset of 400 million pairs collected from various sources on the internet by querying for words that are likely to appear in a real-world setting. This word collection is all words that appear at least 100 times across the entirety of Wikipedia- English. They call this dataset WIT for WebImageText.

While the dataset is not available, the model itself can be accessed via the 'clip' package in Python. This is all we need because our remaining foundation models do not alter the base model parameters of CLIP, but add their own architecture which is then tuned on relevant datasets.

2.4 Prediction

Once the parameters for text and vision encoders have been learned via the loss function outlined above, we can now use these weights to make predictions on new datasets. As shown in Fig. 3, given a new image $\mathcal{I}$ from a new dataset $\mathcal{D}$, we make a zero-shot prediction using CLIP as follows:

- Construct a set of potential class 'labels' $\mathcal{C}$
- Provide a prompt $p \in \mathcal{P}$, which embeds class labels $c \in \mathcal{C}$ resulting in the text $p(c)$, which will now serve as an input to the text encoder
- Return the specific class label $c \in \mathcal{C}$ that results in the highest similarity score between the embedded vector of $\mathcal{I}$ (say $v_{\mathcal{I}}$) and the embedded vector of $p(c)$ (say $t_{p(c)}$)

As we can see in the above procedure, the prompt template $p \in \mathcal{P}$ plays a major role by being an input to our text encoder which informs our label predictions. Mathematically, for a given prompt p:

$$\hat{y}_{\mathcal{I}} = argmax_c \; sim(v_{\mathcal{I}}, t_{p(c)}) \tag{5}$$

This means that the model's performance is going to vary based on the prompt provided. Therefore, care must be exercised to choose p which can result in better predictions.

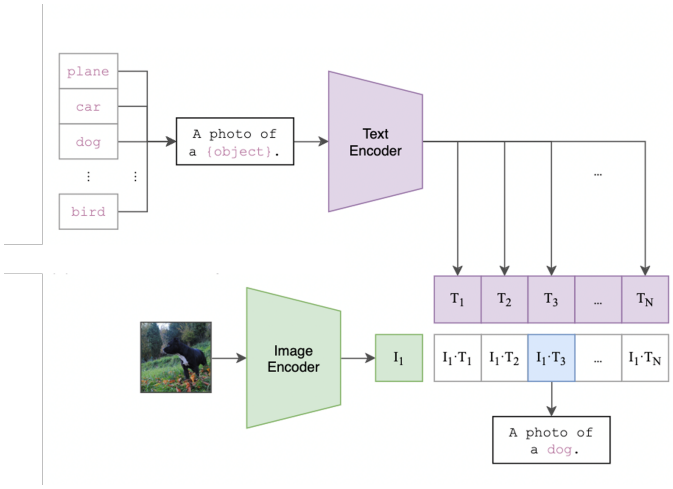

Figure 3: **Predicting architecture of CLIP model.** This figure is drawn by the authors of [2].

3 Few shot foundational models

Now that we have learned about the workings of the base CLIP model and how it is used for zero-shot learning, we will now proceed to look at other foundational models that operate in the realm of few-shot learning i.e. given example images about the task we are about to perform, make predictions on new, somewhat related images. In real-life scenarios, it is not given that we can obtain sufficient data. Few-shot learning mimics this scenario and limits the training data, or support set, to K samples for each class. This is called K-shot learning.

3.1 Linear Probe

The most simple method is linear probing, which is to introduce a linear classifier C on top of the image encoder. The prediction is made as follows,

$$\hat{y} = C(E_{\texttt{image}}(I)) \tag{6}$$

, where I is the input image and $E_{\texttt{image}}$ is the image encoder. Usually, we will fix the CLIP encoder and only train the classifier. The obvious con for this approach is that we are not using the text encoder, which gives up half of the advantage a vision-language model should bring.

3.2 CoOp

CoOp (Context Optimization) [5] is a framework for automatically generating prompts for vision-language models (VLMs) like CLIP. Instead of manually crafting prompts, CoOp learns to optimize the context words dynamically based on the given data.

This offers several advantages:

- **Reduced manual effort:** Eliminates the need for costly and time-consuming manual prompt design
- **Adaptivity:** CoOp can automatically adjust its prompts to different data distributions, leading to improved generalization performance
- **Improved performance:** CoOp has been shown to achieve state-of-the-art results on various vision-language tasks, exceeding the performance of manually crafted prompts

Framework The CLIP model serves as the backbone of CoOp. As we saw in the previous section, CLIP has already been trained on a massive dataset of 400 million image-text pairs, allowing it to learn general representations for both visual and textual information in a multi-modal representation space.

CoOp allows us to tweak this zero-shot model by introducing the concept of **context vectors**.

To perform few-shot learning with the new dataset, we "inject" relevant **context words** that we deem important for the task we are currently solving. The idea being that the addition of words that provide additional context, CLIP can be guided towards learning parameters more appropriate for our current dataset.

Each context word is associated with a continuous vector. These vectors are learned during training by minimizing the difference between CLIP's predictions and the true labels. The goal is to learn vectors that capture the semantic meaning of each word and its influence on CLIP's predictions.

The new training process looks as follows:

- **Data Preparation:** Prepare image-text pairs relevant to the desired task
- **Template Selection:** Choose a template that incorporates the context words meaningfully
- **Context Vector Initialization:** Initialize the context vectors randomly
- **Training Loop:**
  - For each training sample:
    - Encode the image using CLIP
    - Generate a prompt by inserting the context words into the template
    - Generate a prompt by inserting the context words into the template
    - Predict the label using the combined image and prompt encodings
    - Calculate the loss (cross-entropy) based on the difference between the predicted and true labels
    - Update the context vectors using gradient descent to minimize the loss
- **Fine-tuning:** Fine-tune the entire CLIP model (including the context vectors) on the specific task for further performance improvement

In summary, instead of using 'this is a picture of 'a" as a prompt, CoOp effectively uses a series of context vectors$\{v_1, v_2...v_m\}$, which are joined with the class label c_i to arrive to represent the text embedding for the $i-th$ image, t_i.

CoOp has been shown to achieve state-of-the-art results on various vision-language tasks with CLIP, surpassing the performance of manually crafted prompts.

This success is attributed to its ability to dynamically (as opposed to static) generate prompts based on the data distribution, leading to better performance on unseen data.

An added functionality is that by analyzing the learned context vectors, we can gain insights into how CLIP interprets the text and images.

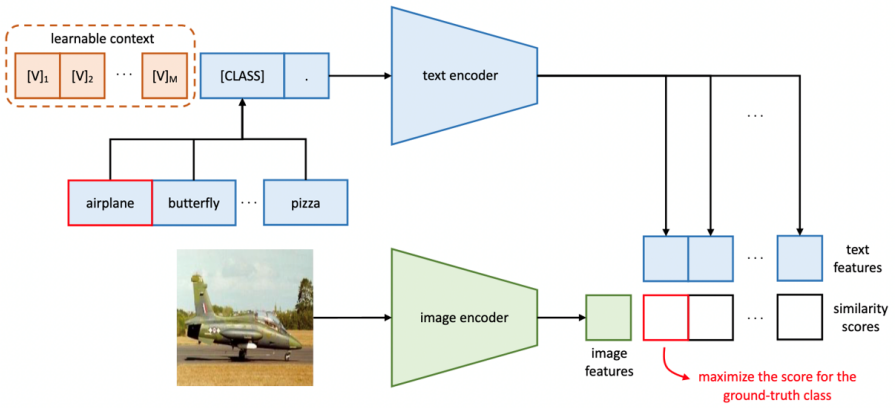

Figure 4: **CoOp training framework pipeline.** This figure is drawn by the author of [5].

3.3 CoCoOp

We previously observed that CoOp can learn context vectors to dynamically generate prompts for a given task which can then be used for better predictions. The authors note that although CoOp delivers improvements over manual prompting for a variety of learning tasks, it suffers from a lack of generalization. That is, once learned, the context vectors are not generalizable to unseen classes (not used for few-shot training) within the same task. Interestingly, zero-shot CLIP delivers better performance than few-shot CoOp models for unseen classes in the same dataset.

This makes sense because while the learned context vectors are specifically tuned over a few classes whereas generic manual prompts remain relatively more generalizable. To overcome this problem, a new architecture was proposed: CoCoOp, Conditional Context Learning [6].

Framework Broadly speaking, the philosophy is to be able to find a way to incorporate information from new image instances into the learnable context vector framework of CoOp to make the end product more generalizable.

This has been achieved as follows:

- Generate image embedding vector for an image instance
- Use this embedding as input to a lightweight neural network (called Meta-Net) with parameters θ
- Generate a new input-contextualized token (called meta token) using the neural network
- Combine input-contextualized token with existing context vectors to serve as input to the text encoder

In the above framework, let us assume that $h_\theta(.)$ gives us the output of Meta-Net for an input (image instance embedding from vision encoder).

Now, the new context vectors are generated using the neural network as $v_1(x) = v_1 + \pi$, $v_2(x) = v_2 + \pi$ and so on. Here,

- v_1, v_2 are the learned context vectors
- $\pi = h_\theta(x)$, where x is the image embedding obtained from the encoder
- $\{v_1(x), v_2(x), ..., v_M(x), c\}$ is now fed as prompt input $p_c(x)$ into the text encoder for an instance x and a specific class label c from our label set $\mathcal{C}$

During the learning process, the context vectors $\{v_1, v_2, ...\}$ and neural net parameter θ are simultaneously learned to minimize the task-specific loss: cross-entropy loss (in this case).

The prediction $\hat{y}$ for a given instance x and text encoder output function $t(.)$ and updates to:

$$\hat{y}_x = argmin_c \; sim(x, t(p_c(x))) \tag{7}$$

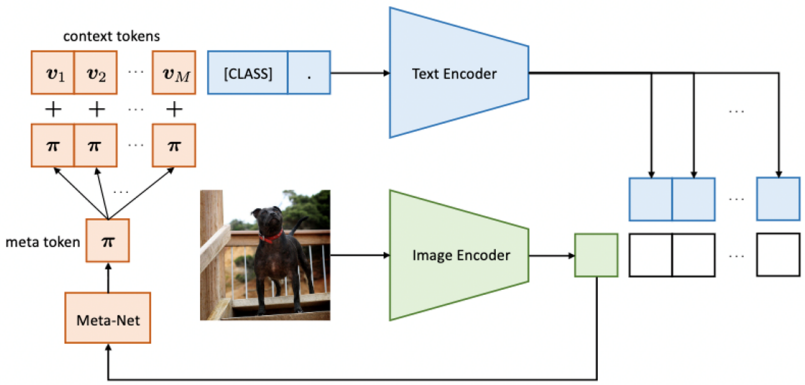

Figure 5: **CoCoOp training framework pipeline.** This figure is drawn by the authors of [6].

3.4 ClipAdapter

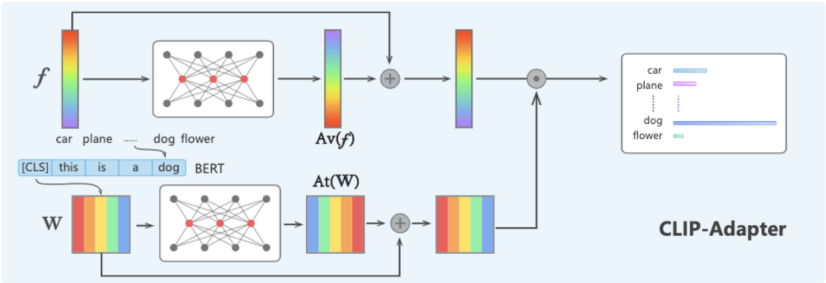

Figure 6: **Architecture of ClipAdapter** This figure is drawn by the authors of [7].

CLIP-adapter introduces two separate modules, one for the visual feature, and one for the text feature. Each can be trained specifically for its modality as follows,

$$A^v(f) = ReLU(f^T W_1^v) W_2^v \tag{8}$$

$$A^t(W) = ReLU(W^T W_1^t)) W_2^t \tag{9}$$

where A_v, A_t are the adapter modules for visual features and text features respectively, and $W_1^v, W_2^v, W_1^t, W_2^t$ are the weights to be trained. Both adapters learn the residuals to add to f or W, which speeds up the convergence.

3.5 TipAdapter

Tip-Adapter, on the other hand, focuses on improving the quality of support features. As shown in Fig. **??**, the support feature is stored in the cache model and can be finetuned easily. They also introduce test-time-augmented support images to further enrich its content. The final logits is the weighted sum from zero-shot logits and logits resulted from the cache model.

3.6 CaFo

Finally, CaFo, or cascade of foundation models, puts almost every foundation together. As shown in Fig. 8, they generate more prompts using GPT [10]; generate more synthetic images for pre-training using DALL-E [11]; and finally enhance the quality and richness of cache model using DINO [12].

4 Experiments

Our objective is to explore the ability of generalization of foundation models. First, we will examine the accuracy of the CLIP-based model on zero-shot benchmarks, including CIFAR100 [13], Ima-

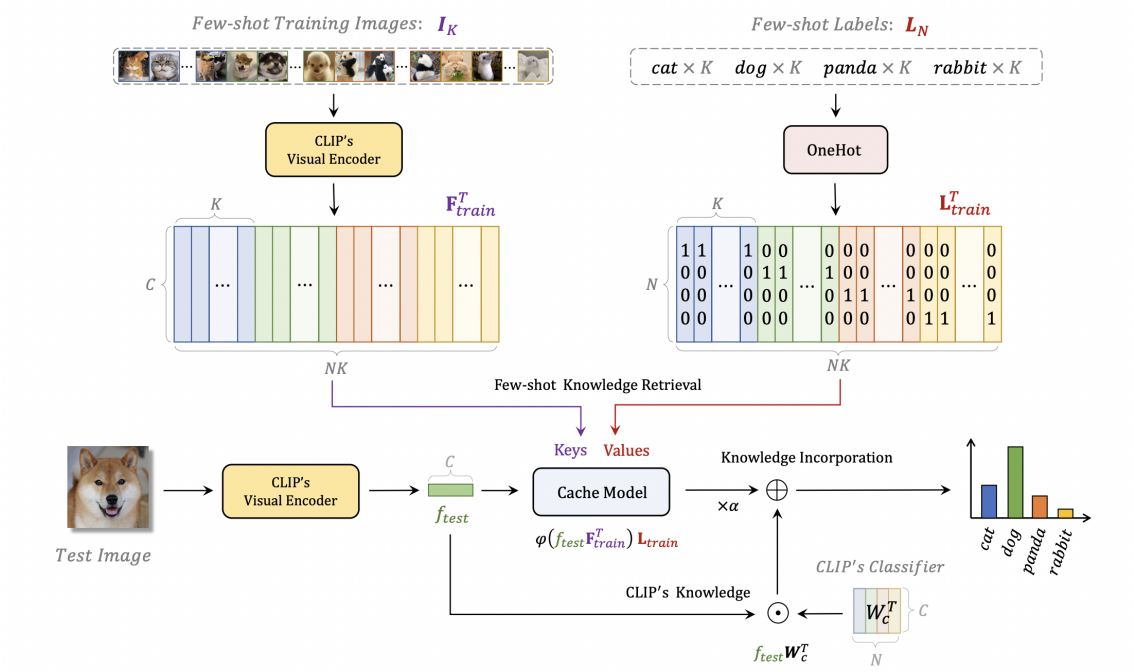

Figure 7: **Architecture of TipAdapter** This figure is drawn by the authors of [8].

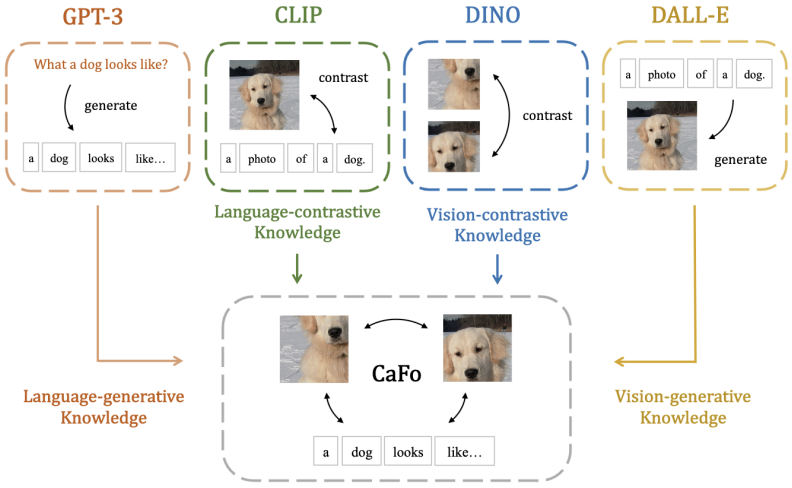

Figure 8: **Architecture of CaFo** This figure is drawn by the authors of [9].

geNet [1], and SUN397 [14]. Then we will implement some of the naive baselines for CLIP-based models to conduct few-shot learning like finetuning them with cross entropy loss. At last, we will implement and compare the selected few-shot learning methods on all the benchmarks.

For experiments, we compare the following baselines. We implement the zero-shot CLIP and linear probe method. We re-implement the few-shot learner CoOp and TipAdapter in our code while referencing their official GitHub. We also implement a very simple few-shot learner CE which trained the whole CLIP with no additional modules by Cross-Entropy loss.

In Tab. 1, 2, and 3, we report the top-1 accuracy on the ImageNet, CIFAR100, and SUN397 respectively , each with various K-shot settings among all the baselines. We found out that out of all the learners we implemented, TipAdapter and CoOp are the best ones, while CE is also quite

competitive. We also found out that linear probing is not a good few-shot learner on these datasets. In fact, zero-shot CLIP is significantly better than few-shot linear probing, even up to $K = 20$.

	0	1	2	5	10	15	20
ZS-Clip	60.33	–	–	–	–	–	–
Linear Probe	–	30.12	34.22	39.61	41.88	45.65	50.77
CE	–	61.31	61.62	62.46	63.04	63.95	64.54
CoOp	–	**61.50**	**61.98**	62.65	**63.69**	64.13	64.17
TipAdapter	–	61.42	61.84	**63.25**	63.35	**65.23**	**65.77**

Table 1: **Results on ImageNet.**

	0	5
ZS-Clip	50.72	–
Linear Probe	–	41.29
CE	–	56.78
CoOp	–	56.84
TipAdapter	–	**59.81**

Table 2: **Results on CIFAR100.**

	0	5
ZS-Clip	58.61	–
Linear Probe	–	55.53
CE	–	66.89
CoOp	–	67.28
TipAdapter	–	**68.68**

Table 3: **Results on SUN397.**

5 Conclusion

In this report, we have down extensive literature survey on the topic of vision-language foundation models and related few-shot learning methods. We have learned the development of CLIP. We have observed the astonishing ability of CLIP to generalize its knowledge to unseen or unfamiliar concepts. In the experiments, we conclude the importance of an instance-based mechanism in CLIP-based few-shot learning.

References

[1] Deng, J., W. Dong, R. Socher, et al. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[2] Radford, A., J. W. Kim, C. Hallacy, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[3] Mokady, R., A. Hertz, A. H. Bermano. Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*, 2021.

[4] Shen, S., L. H. Li, H. Tan, et al. How much can clip benefit vision-and-language tasks? *arXiv preprint arXiv:2107.06383*, 2021.

[5] Zhou, K., J. Yang, C. C. Loy, et al. Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348, 2022.

[6] —. Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825. 2022.

[7] Gao, P., S. Geng, R. Zhang, et al. Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision*, pages 1–15, 2023.

[8] Zhang, R., W. Zhang, R. Fang, et al. Tip-adapter: Training-free adaption of clip for few-shot classification. In *European Conference on Computer Vision*, pages 493–510. Springer, 2022.

[9] Zhang, R., X. Hu, B. Li, et al. Prompt, generate, then cache: Cascade of foundation models makes strong few-shot learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15211–15222. 2023.

[10] Brown, T., B. Mann, N. Ryder, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Ramesh, A., M. Pavlov, G. Goh, et al. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[12] Caron, M., H. Touvron, I. Misra, et al. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660. 2021.

[13] Krizhevsky, A., G. Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[14] Xiao, J., J. Hays, K. A. Ehinger, et al. Sun database: Large-scale scene recognition from abbey to zoo. In *Proc. CVPR*. 2010.