Bayesian Prompt Learning for Image-Language Model Generalization: A Report

Original paper by: Mohammad Mahdi Derakhshani, Enrique Sanchez, Adrian Bulat, Victor Guilherme Turrisi da Costa, Cees G. M. Snoek, Georgios Tzimiropoulos, Brais Martinez
Report by: Jonathan Hu, McGill University
jonathan.hu@mail.mcgill.ca

Abstract

This report reproduces and analyzes Bayesian Prompt Learning (BPL) [1], a variational formulation of prompt learning for CLIP [4] that learns a distribution over prompt residuals and averages predictions over samples at inference. We review the foundations of CLIP-based classification and deterministic prompt learning (CoOp/CoCoOp) [6, 7], then derive the BPL ELBO objective and its KL regularization effect on overfitting. Empirically, we evaluate unconditional BPL on the unseen prompt generalization setting using FGVCAircraft and reproduce key trends reported in the original work. In addition, we present ablations that diagnose (i) baseline overtraining effects for CoOp and (ii) Monte Carlo sample sensitivity for unconditional BPL in the low-sample regime.

1. Introduction

CLIP-style vision–language models (VLMs) enable zero-shot image classification by comparing image embeddings to text embeddings generated from class prompts [4]. In practice, downstream adaptation is often needed in the few-shot regime, where labels are scarce and full fine-tuning is undesirable. Prompt learning addresses this by optimizing a small set of continuous context tokens while keeping CLIP's encoders frozen [6, 7]. However, deterministic prompt optimization (e.g., CoOp/CoCoOp) can overfit, hurting robustness under distribution shift and on unseen classes [1]. Bayesian Prompt Learning (BPL) mitigates this by learning a variational distribution over prompt residuals and averaging predictions over samples at inference time, with a KL term that regularizes the prompt space [1].

Report Contributions:

- **Literature synthesis:** A compact review that ties together how CLIP enables zero-shot classification, how CoOp/CoCoOp adapt CLIP via learned context tokens, and why BPL reframes prompt learning as variational inference with KL regularization [1, 4, 6, 7].
- **Reproduction and ablations:** Unconditional BPL is reproduced on FGVCAircraft under the unseen prompt generalization setting, along with targeted ablations that diagnose baseline overtraining (epoch sensitivity for CoOp) and the effect of Monte Carlo sampling in unconditional BPL, especially in the low-sample regime [1].
- **Sequential Colab implementation:** The experimental pipeline is implemented in a single, readable Google Colab notebook to make the end-to-end flow easy to follow. Most components are written from scratch (zero-shot CLIP, CoOp, dataset split, training/evaluation), while the Bayesian prompt learner components (KL term and sampling) are reused/adapted from the original BPL repository.

2. Related Works

2.1. Contrastive Language-Image Pre-training

To understand prompt learning for vision-language models, it is important to first understand the approach behind Contrastive Language-Image Pre-training, which was first introduced in 2021 by Radford *et al*. at OpenAI [4]. In their seminal paper "Learning Transferable Visual Models from Natural Language Supervision" paper, Radford *et al*. demonstrated that Contrastive Language-Image Pre-training is an efficient method to learn image representations from natural language supervision and open-sourced the weights of CLIP, their pre-trained vision-language model [4]. It is worth noting that CLIP can refer to both the contrastive learning method to used to train the model and the open-source model itself. We will use CLIP to refer to the model, and contrastive learning to refer to the training method.

In terms of implementation, Radford *et al*. began with a simplified version of *ConVIRT* architecture and trained it on a large-scale dataset of image-text pairs [4]. For context,

ConVIRT was originally developed by Zhang *et al*. at Stanford in 2020 to learn visual representations of medical images and is a type of architecture that combines transformer-based language modeling with contrastive objectives [4, 5].

2.1.1 Contrastive Learning Objective

Given a dataset of N (image, text) pairs, CLIP jointly trains a text encoder and an image encoder to maximize the N cosine similarities of the correct image-text embedding pairs (i.e. the blue squares in (1) of Figure 4) while minimizing the cosine similarities of the $N^2 - N$ incorrect pairs [4]. The outcome of this training is a learned multi-modal embedding space derived from natural language supervision [4]. The intuition is that the training process updates the weights of the image and text encoders such that the correct image-text pairs are pushed closer together, and the incorrect pairs are pushed farther apart in the joint image-text embedding space.

2.1.2 Zero-Shot CLIP for Image Classification

To initialize the classifier in (2) of Figure 4, a set of C prompts associated with the C classes is passed through the text encoder to generate text embeddings T_1 through T_C [4]. At inference time, when a new image I_1 is given to CLIP, the cosine similarity is computed between the image and each of the T_1 through T_N text embeddings, and the predicted class is the class with the highest cosine similarity seen in (3) of Figure 4 [4].

In terms of accuracy and performance, zero-shot CLIP performs just as well as an ImageNet-trained ResNet50 linear classifier model across a wide variety of datasets [4]. However, CLIP was unable to match the performance of the state-of-the-art models specific to each of the benchmarks [4]. In fact, Radford *et al*. estimate that it would take approximately orders of magnitude more compute for zero-shot CLIP to reach the SOTA, which is intractable with the current hardware [4].

The key takeaway is that CLIP demonstrates a novel capability to adapt a vision-language model to a new classification task by simply generating descriptive textual prompts for each new class [4]. CLIP leverages the richness and contextual information contained in natural language to perform reasonably well at domain adaptation [4]. This is in contrast to supervised object classification models which require retraining on new labels to adapt it to a new classification task [4]. However, zero-shot CLIP needs to be finetuned to be useful in real world applications. One such method is prompt learning.

2.2. Prompt Learning in Visual-Language Models

2.2.1 Role of prompts

In the context of vision-language models, Zhou *et al*. describe prompting as the process by which "classification weights are synthesized from natural language describing classes of interest." [7]. This is not to be confused with prompt engineering in text-based large-language models (i.e. GPT-3) where the goal is to modify the input prompt to improve the model outputs. In the context of image classification and vision-language models, the input is an image, the output is a predicted class, and the prompts define the set of classes that the model compares the image against. The prompt with the highest cosine similarity with the image is the predicted class in zero-shot CLIP [4].

2.2.2 Handcrafted prompts

In CLIP, Radford *et al*. found that changing the prompts to "photo of a class" led to higher classification accuracy than simply "class", and they argue this is because the image-text pairs scraped from the internet tend to contain captions with multiple words [4]. The key takeaway is that handcrafting the textual context surrounding the class in the prompts can improve the performance of the CLIP model [4].

It is also interesting to consider Radford *et al*.'s observation that the prompt "photo of a {class}" performs better than the single-word prompt "{class}" [4]. Based on this observation, one would expect the prompt "a photo of a {class} texture" to perform better than "{class} texture" on the Describable Textures dataset. However, Zhou *et al*.'s [7] experiments show that "{class} texture" had 2% higher accuracy than "a photo of a {class} texture". Indeed, handcrafting prompts is a trial-and-error approach that introduces human bias, and so the patterns and tricks that work on one dataset will not necessarily generalize to new datasets and tasks.

2.2.3 Context Optimization (CoOp)

In 2022, in search of a optimization-based method that generalizes across datasets, Zhou *et al*. proposed a prompt learning approach called *Context Optimization* [7] to enable domain adaptation without modifying the weights of the CLIP image and text encoders. CoOp achieves this by encoding the contextual knowledge typically found in handcrafted prompts in a set of learned continuous vectors v_1 through v_M appended to the beginning of each prompt such that the prompt becomes "v_1 v_2 ... v_M {class}" [7]. CoOp leverages the fact that the contextual vectors are continuous, which enables them to be learned using backpropagation, but the drawback is that these vectors are not as easily interpretable as a handcrafted prompt [7]. In their

experiments, Zhou *et al*. find that few-shot CoOp, which requires a few labeled images per class, improves in-domain performance and does better at domain generalization than zero-shot CLIP, but its performance is still weak on unseen classes [1, 7].

2.2.4 Conditional Context Optimization (CoCoOp)

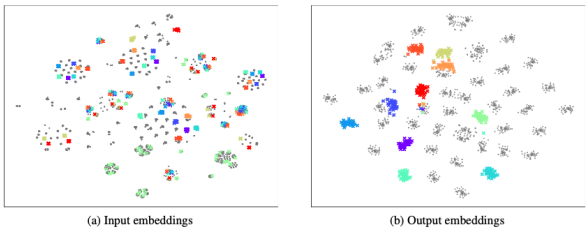
Attempting to address CoOp's limited ability to generalize well to "unseen classes within the same dataset", Zhou *et al*. published a follow-up paper called *Conditional Context Optimization (CoCoOp)* [6, 7]. Instead of a fixed prompt for all classes and all images, CoCoOp conditions the learnable prompt on the input image at test time [6]. At test time, after the input image is encoded into the joint image-text embedding space, the representation of the input image is fed to a neural network called the Meta-Net [6]. The Meta-Net then outputs a set of residuals that modifies the base prompt (Figure 2) [6]. While CoOpOp improves generalization of CoOp on unseen classes, it decreases the training efficiency because the prompts cannot encoded in parallel with the input image - the image must be encoded first to generate the contextual vectors using the Meta-Net, and then and only then can the prompt be encoded [6].

2.3. Statistical Approaches to Prompt Learning

2.3.1 Prompt Distribution Learning (ProDA)

In 2022, Lu *et al*. propose a novel probabilistic method called *PROmpt Distribution LeArning (ProDA)* for adapting pre-trained vision-language models to domain-specific tasks [2]. Two observations motivate their decision to use a probabilistic approach for prompt learning [2]. The first is that two images from the same class can vary slightly in the representation space due factors such as pose, deformation and lighting conditions; thus, a single prompt is ill-equipped to represent the variance within an entire class [2]. The second is that while prompts representing the same class might be far apart in the input embedding space, their representations in the output embedding space (i.e. after passing through the text encoder) are close together, as seen in Figure 1 [2]. This motivates the decision to model the distribution of prompt representations (i.e. their classification weights) in the output embedding space. Furthermore, given the clustering observed in the t-SNE visualization, they choose to model the distribution as a multivariate Gaussian, which allows them to derive a computable upper bound the loss function with Jensen's inequality [2].

In their experiments, Lu *et al*. show that ProDA yields an average accuracy improvement of 9.1% across 12 datasets compared to Radford *et al*.'s hand-crafted prompts and consistently outperforms previous approaches such as CoOp. [2, 4].



(a) Input embeddings (b) Output embeddings

Figure 1. t-SNE visualization of 50 random ImageNet classes [2]

2.3.2 Bayesian Prompt Learning (BPL)

In 2023, instead of modeling the prompts in the output embedding space, Derakhshani *et al*. propose to learn the distribution of prompt residuals in the input embedding space using variational inference [1]. The strength of their approach is their method can be adapted to both the conditional and unconditional prompt learning architectures; in contrast, ProDA is only compatible with the unconditional prompt learning architecture [1]. Another significant difference is at test time, BPL samples from the distribution and averages the predicted probabilities, whereas ProDA does not sample and uses the mean output embedding for each class from the learned multivariate Gaussian distribution because Lu *et al*. did not find a significant difference in accuracy [1, 2]. Furthermore, formulating prompt learning as a variational inference problem enables them to regularize the prompt space with a KL divergence term and reduce overfitting [1]. The next section will provide the theoretical foundations necessary to understand Bayesian Prompt Learning. As such, the theory behind ProDA will not be covered as it is not necessary for understanding BPL.

3. Methods

3.1. Preliminaries

The CLIP model is composed of a image consists of a image encoder $f(\cdot)$ and a text encoder $g(\cdot)$ [4]. Passing an input image $\mathbf{x}$ through the image encoder $f(\cdot)$, we obtain the normalized image embedding $\mathbf{z}$:

$$\mathbf{z} = \frac{f(\mathbf{x})}{\|f(\mathbf{x}))\|_2}$$

Passing input text $\mathbf{t}$ through the text encoder $g(\cdot)$, we obtain normalized text embedding $\mathbf{w}$:

$$\mathbf{w} = \frac{g(\mathbf{t})}{\|g(\mathbf{t})\|_2}$$

There are C classes to classify. We generate C category descriptions

$$\{\mathbf{t_c}\}_{c=1}^{C}$$

and pass them through the text encoder $g(\cdot)$ to generate classifier weights:

$$\mathbf{w}_{1:C} = [\mathbf{w}_1^T, \ldots, \mathbf{w}_C^T]^T$$

For the zero-shot case, $\mathbf{t_c}$ = "a photo of a $\{c\}$" and the probability that input image $\mathbf{x}$ is from class y is equal to:

$$p(y \mid \mathbf{x}) = \frac{e^{\mathbf{z}^T \mathbf{w}_y}}{\sum_{c=1}^{C} e^{\mathbf{z}^T \mathbf{w}_c}}$$

where $\mathbf{z}^T \mathbf{w}_c$ is the cosine similarity between the image embedding $\mathbf{z}$ and the text embedding of class $\mathbf{c}$.

Context Optimization [7] learns an optimal prompt $\mathbf{p}^*$ that minimizes the cross-entropy loss over the few-shot training set $\{\mathbf{x}_i, y_i\}_{i=1}^N$:

$$\mathbf{p}^* = \arg\min_{\mathbf{p}} \mathbb{E}_{\mathbf{x}_i, y_i} \big[-\log p(y_i \mid \mathbf{x}_i, \mathbf{p}) \big].$$

During training, $\mathbf{p}$ is randomly initialized and used to generate category descriptions and classifiers weights:

$$\mathbf{t_c}(\mathbf{p}) = \mathbf{p} + \{c\}$$

$$\mathbf{w}_{1:C} = [\mathbf{w}_1^T(\mathbf{p}), \ldots, \mathbf{w}_C^T(\mathbf{p})]^T, \text{ where } \mathbf{w}_c(\mathbf{p}) = g(\mathbf{t_c}(\mathbf{p}))$$

$\mathbf{p}$ is a learnable continuous prompt of dimensions $[L, e]$ where L is the context length (also known as the number of context tokens) and e is the embedding dimension [7]. For CLIP, the dimension of the joint image-text embedding space is 1024, but the dimension of the text embeddings outputted by the text encoder (i.e. a transformer) is 512 - this is also where the prompts operate so $e = 512$ [4]. To go from 512 to 1024 dimensions, the text embeddings are passed through an projection layer at the very end [4]. As discussed in Sec. 2.2.4, the conditional version of CoOp conditions the continuous prompt $\mathbf{p}^*$ on the image by adding residuals r which are generated by a Meta-Net neural network [6] seen in the bottom left of Figure 2.

With respect to performance, Zhou *et al.* find that a context length of 16 yields the highest average in-domain accuracy for CoOp, but it tends to learn prompts that are highly tuned to the training classes, hurting performance on unseen classes or shifted domains [7]. The reason is that training one fixed prompt to fit the labeled few-shot training set is equivalent to committing to one very specific solution ahead of time. Ideally, we would like to incorporate the fact that there is not single optimal prompt, but a family of optimal prompts. This is where a Bayesian view is advantageous, allowing us to model and find a distribution of optimal prompts.

3.2. Conditional Bayesian Prompt Learning

Instead of learning a single deterministic prompt, Bayesian Prompt Learning tries to learn the a conditional distribution of optimal prompt residuals in the input embedding space [1]. This is done by framing the residuals $\mathbf{r}$ as the latent variable and using variational inference to learn a Gaussian distribution with parameters $\mu(\mathbf{x})$ and $\Sigma(\mathbf{x})$ which are conditioned on the image $\mathbf{x}$ [1].

First, Derkashani *et al.* [1] begin by representing each prompt as composed of a base prompt $\mathbf{p} = [\mathbf{p_1}, \cdots, \mathbf{p_L}]$ plus a residual $\mathbf{r}$ sampled from the real posterior distribution $p_\gamma(\mathbf{x})$, which is not initially known. Recall that L is the context length of the prompt.

$$\mathbf{p}_\gamma(\mathbf{x}) = [\mathbf{p}_1 + \mathbf{r}_\gamma,\ \mathbf{p}_2 + \mathbf{r}_\gamma,\ \cdots,\ \mathbf{p}_L + \mathbf{r}_\gamma], \quad \mathbf{r}_\gamma \sim p_\gamma(\mathbf{x}).$$

Ideally, we would like to find the real posterior distribution $p_\gamma(\mathbf{x})$ of the residuals $\mathbf{r}$ that maximizes the marginal likelihood of class y given image $\mathbf{x}$:

$$p(y \mid \mathbf{x}) = \int_\gamma \frac{e^{f(\mathbf{x})^T g(t_{c,\gamma}(\mathbf{x}))}}{\sum_{c'} e^{f(\mathbf{x})^T g(t_{c',\gamma}(\mathbf{x}))}} p\big(\mathbf{p}_\gamma(\mathbf{x})\big)\, d\gamma.$$

Here, $g(t_{c,\gamma}(\mathbf{x}))$ is the classifier weight of class c generated by passing the $t_{c,\gamma}(\mathbf{x})) = \mathbf{p}_\gamma(\mathbf{x}) + \{\mathrm{c}\}$ into the text encoder $g(\cdot)$. However, it is intractable because we would need to know $p_\gamma(\mathbf{x})$, and we would need to compute the integral by either integrating over the entire 512-D embedding space of the residuals or finding a closed-form analytical solution, both of which are intractable.

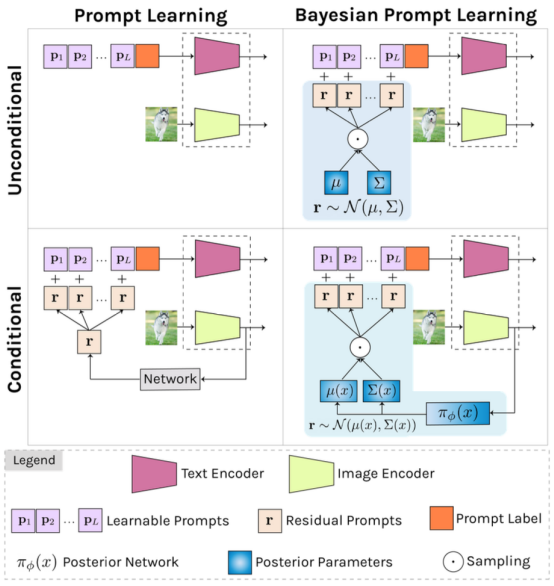


Figure 2. Differences between CoOp, CoCoOp and Bayesian Prompt Learning [1]

Instead, we introduce a surrogate posterior distribution $\pi_\phi(\mathbf{r} \mid \mathbf{x})$ over the residuals parametrized as a Gaussian with mean $\mu(\mathbf{x})$ and $\Sigma(\mathbf{x})$ which are estimated by conditioning on the image $\mathbf{x}$ using a neural network with two linear layers and two linear heads [1]. Then, we learn a $\pi_\phi(\mathbf{r} \mid \mathbf{x})$ that maximizes the marginal likelihood by maximizing the ELBO, which corresponds to the right hand side of the inequality below [1]:

$$\begin{aligned}\log p(y \mid \mathbf{x}) \geq \mathbb{E}_{\pi_\phi(\mathbf{r}|\mathbf{x})}\big[\log p(y \mid \mathbf{x}, \mathbf{r})\big] \\ - D_{\mathrm{KL}}\big[\pi_\phi(\mathbf{r} \mid \mathbf{x}) \,\big\|\, p_\gamma(\mathbf{r})\big]\end{aligned} \tag{1}$$

We can think of the expectation over $\pi_\phi(\mathbf{r}|\mathbf{x})$ of the log-likelihoods term as the data fit term and the KL divergence term as a regularization term that prevents the learned distribution from overfitting the training data. By defining the prior $p_\gamma(\mathbf{r})$ in the KL divergence term as a Gaussian with mean 0 and unit variance $\mathcal{N}(0, 1)$, we are able to regularize the prompt space [1]. The expectation over $\pi_\phi(\mathbf{r}|\mathbf{x})$ of the log-likelihoods is also intractable so we use Monte Carlo sampling to approximate that term [1]. Note that in theory, we would maximize the ELBO with gradient ascent, but in practice, we minimize the negative of the ELBO expression with gradient descent. During training, we use the reparameterization trick to first sample a residual from the surrogate distribution before then computing the loss and backpropagate.

In sum, during training, the CLIP encoders $f(\cdot)$ and $g(\cdot)$ are frozen and we jointly learn a base prompt $\mathbf{p} = [\mathbf{p_1}, \cdots, \mathbf{p_L}]$ and a neural network that estimates the mean and covariance of the surrogate distribution $\pi_\phi(\mathbf{r}|\mathbf{x})$ conditioning on the image $\mathbf{x}$ [1]. At inference, we sample K residuals from the surrogate posterior distribution $\pi_\phi(\mathbf{r}|\mathbf{x})$, which generates K different prompts, and we take the probability distribution generated by each prompt and average them [1].

3.3. Unconditional Bayesian Prompt Leaning

For the unconditional case, the formulation of ELBO becomes [1]:

$$\log p(y|\mathbf{x}) \geq \mathbb{E}_{\pi_\phi(\mathbf{r})}\big[\log p(y \mid \mathbf{x}, \mathbf{r})\big] - D_{\mathrm{KL}}\big[\pi_\phi(\mathbf{r}) \,\|\, p_\gamma(\mathbf{r})\big]$$

We still learn a base prompt $\mathbf{p} = [\mathbf{p_1}, \cdots, \mathbf{p_L}]$, but instead of learning a neural network to estimate of the mean and covariance of the surrogate posterior distribution $\pi_\phi(\mathbf{r}|\mathbf{x})$, we instead initialize two continuous vectors to learn the mean and covariance of a global latent distribution $\pi_\phi(\mathbf{r})$, which has no dependence on the input image $\mathbf{x}$. [1]. This is the version of Bayesian Prompt Learning that we will implement in our experiments for this report.

4. Experiments and Results

4.1. Implementation Scope

The primary objective of the experiments was to compare the generalization ability of the non-Bayesian prompt learning techniques such as CoOp and CoCoOp to the Bayesian Prompt Learning technique in the few-shot learning regime.

Due to the large number of dependencies in the original CoOp and BPL repositories, we reimplemented the experiments from scratch in Google Colab, which was more practical than adapting the existing codebases.

We favored a progressive approach to the implementation process, choosing to first start with the fundamentals. We started by replicating the zero-shot CLIP accuracy results on CIFAR10 and CIFAR100. This provided a sanity check and gave us an understanding of the dimensionality of the input and output embedding space.

Once that was complete, we implemented the traditional prompt learning approach (CoOp) from scratch because it would allow us to learn the fundamentals of learning a continuous prompt and the end-to-end pipelien from random initialization, to training, and then to inference. This is important because Bayesian Prompt Learning directly builds on top of Zhou *et al.*'s implementation of CoOp/CoCoOp, adding a module that learns a posterior distribution of residuals (conditional) or a global latent distribution (unconditional) with variational inference and samples from distribution for each forward pass [1,6,7]. This is also clear from Figure 2.

Given the significant amount of engineering that was necessary to implement the CoOp paper from scratch on Colab, we decided to use and adapt the statistical modules (i.e. KL divergence, sampling) in the Bayesian Prompt Learner class from Derakshani *et al.*'s BPL repo [1]. [1].

While there is a conditional and unconditional version of context optimization and Bayesian Prompt Learning, we decided to focus on the unconditional version for the implementation and for the experiments. Future work will extend it by training the neural network needed to enable the conditional version of BPL.

To recap, we implemented zero-shot CLIP, CoOp, the dataset splitting for the unseen prompt generalization task, the training loops and evaluation functions. However, we did not implement the Bayesian Prompt Learner, opting to use and adapt Derakshani *et al.*'s open-source code on GitHub [1].

4.2. Experimental Setup

We match our hyperparameters with those in CoOp and BPL, except for the warm-up schedule in the first epoch which we skip; the key hyperparameters are highlighted here and are held fixed across all experiments in this re-

port unless otherwise indicated in certain ablation studies [1, 6, 7]: ViT-B/16 visual backbone, 16 shots per class for training, learning rate of 0.002, cosine annealing decay schedule, SGD optimizer with 0.9 momentum, context length (L) of 4, random resized crop and horizontal flips for the image transforms. As in Derakshani *et al*. [1], the results of each experiment are averaged across three different seeds.

4.3. Unseen Prompt Generalization Task

We chose to reproduce the results for the unseen prompts generalization task proposed by Derakshani *et al*. [1] to quantitatively compare the generalization of the different models. Unseen prompt generalization, as defined in Derakshani *et al*. [1] consists of showing the model half of the dataset's classes during training and testing on the unseen half.

4.4. Dataset

We chose focus on the fine-grained aircraft classification dataset FGVCAircraft, consisting of 100 classes, and 100 images per class because the performance of CoOp, CoCoOp, ProDA and BPL was at its worst performance was poorest on this dataset for unseen generalization as seen in Figure 3 [1, 3]. The FGVCAircraft dataset consists of 30 plane manufacturers (i.e. Boeing), 70 families (i.e. Boeing 737s), and 100 variants (i.e. Boeing 737-200 vs. 737-800) corresponding to the 100 classes in this dataset [3]. Concretely, for the unseen prompt generalization task, the CoOp and unconditional BPL models were shown 16 images (16 shots) for each of the 50 aircraft variants during training and evaluated on a test set composed of images from the other 50 aircraft variants that it did not see during training.

4.5. Reproducibility

The results of our implementation can be reproduced by running the Google Colab notebook. The results for Zhou *et al*.'s CoOp and Derakshani *et al*.'s unconditional BPL were obtained by running their open-source code and experiments on our local machine [1, 7]. As long as the number of MC samples did not exceed 10 for BPL, our NVidia 3080's 10GB of GPU memory was sufficient for running the experiments. This was done to confirm reproducibility and to obtain results for ablations that were not conducted in their paper. For instance, Derakshani *et al*. only report results for the influence of Monte Carlo sampling on conditional BPL, but no results are reported for unconditional BPL, which is the approach implemented in this report [1]. Thus, we modified the config file in the BPL codebase to run the *unconditional* BPL MC-sampling ablation locally, reporting the results in Table 5.

4.6. Results

As a sanity check, we first compare the accuracy of our implementation of CoOp against Zhou *et al*.'s. We train our model for 200 epochs as in CoOp [7] on the entire FGVCAircraft training set with all 100 classes. This also provides a reference point for measuring how much performance degrades when we transition to the harder unseen-prompt generalization setting. Overall, we find that the accuracy of our implementation is in line with CoOp's across various context lengths [6, 7]. As reported by Zhou *et al*. in their paper, increasing context length increases performance [7].

Table 1. **CoOp in-domain accuracy**

Context Length	CoOp [6]	Ours
4	40.27%	39.67%
8	42.57%	41.90%
16	43.53%	43.78%

The results in the rest of this section will focus exclusively on the unseen prompt generalization setting where we train on half the classes and test on the unseen half.

We train our unconditional BPL model for 10 epochs and with 10 Monte Carlo samples for each forward pass to match BPL's default setup [1]. We vary the context lengths {4, 8,16}. As noted in Derakshani *et al*. [1], increasing the context length causes overfitting and decreases performance at the test time on unseen classes, and this is in line with the results from our implementation in Table 2. The intuition is that a prompt with more continuous tokens has more space to encode the contextual information specific and overfit to the data seen during training. Thus, longer prompts perform better when the test data is in-distribution as in Table 1, but tend to be less generalizable and robust to distribution shift and out-of-distribution examples [1]. This is the reason why Derakshani *et al*. found that a context length of 4 performs best for domain generalization [1].

Table 2. **Unconditional BPL: effect of context length on unseen prompt generalization**

Context Length	MC Samples	BPL [1]	Ours
4	10	34.17%	34.35%
8	10	32.53%	31.61%
16	10	31.10%	30.39%

Now, that we have established that a context length of 4 is optimal for domain generalization, we compare unconditional Bayesian Prompt Learning against CoOp and zero-shot CLIP in Table 3 and find that unconditional Bayesian Prompt Learning both methods on unseen prompt general-

ization with an accuracy of 34.45%. Furthermore, CoOp's in-domain accuracy, which was over 40% in Table 1 when trained and tested on 100 classes, dropped to 25.25% when faced with the unseen prompt generalization classification task.

Table 3. **Task I: unseen prompt generalization**

Method	Context Length	Original [1, 6]	Ours
Zero-shot	–	–	23.13%
CoOp	4	22.30%	**25.25%**
Unconditional BPL	4	33.80%	**34.45%**

5. Discussion

5.1. Weak CoOp Baseline

On unseen prompt generalization, Derakshani *et al*. [1] report an accuracy 22.30% for CoOp and 33.80% for unconditional BPL on the FGVCAircraft dataset in their paper's Table 4, which is in line with the results from our implementation in Table 3. Although Derakhshani *et al*. report that their Bayesian approach improves unseen prompt generalization by more than 10 percentage points relative to CoOp, our analysis indicates that the CoOp baseline used in their paper is overtrained and therefore weaker than a properly tuned CoOp model [1]. This is because Derakshani *et al*. naively transfer the number of training epochs (200) used in the CoOp paper for their model with a prompt of length 16 and apply it to their CoOp baseline with a prompt of length 4 [1]. The combination of 200 training epochs and a learnable prompt with 4 context tokens leads to the learnable prompt overfitting the training classes. We conjecture that a smaller number of epochs will lead to less overfitting. To confirm this, we perform an ablation on the number of training epochs for our CoOp baseline and find that reducing the training epochs from 200 to 25 increased the accuracy of the CoOp baseline from 25.25% to 30.41%. This oversight is understandable and does not take away from the fact unconditional Bayesian Prompt Learning still performs better than CoOp in the few-shot learning generalization regime. However, the improvement over the baseline is not as significant as initially reported.

Table 4. **Effect of training epochs on CoOp baseline**

	25	50	75	100	200
Ours	**30.41%**	28.33%	27.63%	28.33%	25.25%
CoOp [1]	–	–	–	–	22.30%

5.2. Key Modeling Assumptions

Global Residual: The BPL model architectures assumes a single vector $\mathbf{r}$ shifting all the prompt tokens captures the useful prompt variability [1].

Gaussian residual distribution: The decision to model the surrogate residual distribution π_ϕ generates the implicity assumption that residuals live roughly in a single Gaussian blob in the text-embedding space. This assumption might break if there multiple distinct "modes" of how a class if described (i.e. completely different textual description for the same object) or if the true prompt distribution is highly multi-modal.

Frozen CLIP encoders: The choice to use a frozen backbone assumes that your new dataset is close enough to the distribution that was seen during initial pre-training. Thus, if your new dataset is very far or was not seen during CLIP's pretraining distribution, then you may need to do more than just prompt learning to achieve a level of performance acceptable in the real-world and may actually need to finetune the backbone as well.

It is worth noting that although this situation is less and less likely for foundation models trained on massive datasets, it is still possible. In the CLIP paper, the task of classifying handwritten digits from the canonical MNIST dataset is discussed to illustrate the limitations of the CLIP model [4]. Although classifying MNIST digits is a straightforward task that a traditional supervised computer vision model can perform with high accuracy, CLIP achieved a subpar accuracy of 88% [4]. This was surprising given the relative straightforwardness of classifying handwritten digits and the fact that CLIP is a foundation model. To investigate, Radford *et al*. tried to find images in their training set that looked like the MNIST images by using a nearest-neighbor and nearest duplicate approach, but they found almost no look-alike images in their dataset [4]. This example illustrates that CLIP does not perform well on truly out of distribution data, and that foundation models handle this problem by trying to train on more data such that no data is out of distribution [4].

5.3. Ablation on Monte Carlo Sampling

In BPL, Derakshani *et al*. report that increasing the number of Monte Carlo samples marginally improves performance on their conditional BPL model, but starts to plateau at 20 samples [1]. We reproduce this ablation, but for the unconditional BPL model and find that the performance of unconditional BPL diverges significantly from conditional BPL in low sample regime. More specifically, when the number of MC samples is less than 5, the performance of the model collapses. One running hypothesis was that the number of training epochs is too low for the number of samples, so we increased the training epochs from 10 to 50, but we did not see a noticeable improving trend, so we did not

pursue that hypothesis further. It is more likely that the information gain from conditioning the residual distribution on the image prevents the conditional version of BPL from collapsing in the low MC sample regime.

Table 5. **Effect of Monte Carlo sampling** on Unconditional BPL

MC Samples	BPL [1]	Ours
1	3.80%	12.00%
2	10.93%	11.36%
5	33.93%	32.85%
10	34.17%	34.45%

5.4. Tradeoffs in Bayesian Prompt Learning

Implementing BPL requires learning the distribution of the prompt residuals with variational inference: this is conceptually simple and relatively easy to implement on top of CoOp [1, 7]. The other benefit is that the KL divergence term in the variational formulation of the loss function acts as a Bayesian regularizer in the prompt space, reducing overfitting and thereby improving generalization to unseen classes [1].

However, this comes at the cost of extra inference time due to sampling, though it is small for modest number of samples [1]. Furthermore, the Gaussian form of the surrogate distribution π_ϕ may not always accurately capture the distribution of optimal prompts. Finally, although BPL did demonstrate a significant improvement over CoOp and CoOp on domain generalization, the improvement in accuracy was not large with respect to the ProDA, the most comparable probabilistic approach at the time of publication [1].

5.5. Near Ill-Posedness in Fine-Grained Aircraft Variant Classification

Classifying aircraft variants still pose challenges for contrastive-learning based models such CLIP even when augmented with non-Bayesian and Bayesian prompt learning approaches. Although the aircraft classification is not technically ill-posed for a human, it is almost ill-posed. What makes this dataset challenging is that the differentiating characteristics between two aircraft from the same manufacturer are often subtle and spatial in nature. For instance, the major tells between a Bombardier Global 6500 and 7500 are the length of its wingspan, the number of windows, and the diameter of its jet engine intakes. When the photo of the plane is take from the front, it is not technically ill-posed because there is still always a way to identify it by the shape and relative size of the engine intakes, but it is almost ill-posed in the sense that you are missing most of the contextual information necessary to tell the difference (i.e. the number of windows is not immediately identifiable, etc.). While a plane enthusiast can discriminate between two aircraft variants in an almost ill-posed situation, VLMs struggle. This is in line with observations that zero-shot CLIP had difficulty counting objects in an image (i.e. spatial reasoning), raising the question about the nature of the representations learned by CLIP during pre-training [4].

5.6. Potential Improvements

When we consider that VLMs such as CLIP are trained with contrastive learning objectives, it would be reasonable to intuit that the learned representations are optimized to encode the differences between concepts and objects in the world (i.e. the model learns to represent what makes a photo of a cat different from a photo a plane) in the joint image-embedding space. Perhaps classifying aircraft variants is challenging because the subtle differences between two plane variants from the same family and manufacturer lead the image encoder $f(\cdot)$ to output image embeddings $\mathbf{z}$ that are closer in L-2 distance than the distance between the embedding of a photo of a car and plane. If you also consider the variation in pose, background and lighting for a given variant, this could very well make the "cluster" of one variant's representation overlap with another variant's in the joint image-text embedding space.

Further work could be done to investigate if the image representations of plane variants overlap or are closer in L-2 distance than the image representation of classes in other fine-grained classification dataset. If this is the case, the natural solution would be to find a method that adapts the frozen model by pushing the image representations of the plane variants further apart.

The limitation of prompt learning as defined in CoOp is that the model learns a prompt that tries to more closely align the text embeddings with the image embeddings in the joint image-text embedding space. It is possible that the space occupied by the image embeddings of plane variants is "crowded", and simply trying learn a prompt that aligns the text embeddings with the cluster of image embeddings corresponding to each variant will yield limited performance gains. Thus, one could imagine a conceptually similar method that learns to instead push the image embeddings further apart by learning variant (class) specific continuous tokens to append to the images before they are passed into the image encoder. Perhaps this begins to resemble fine-tuning the entire model itself, since learning class-specific image tokens effectively would push the image encoder to relearn parts of its representation space.

A second direction is to introduce hierarchical structure into the classification process. Instead of asking a single VLM to discriminate among 100 visually similar variants, one could decompose the task into stages: a first-level VLM that predicts the manufacturer, a second-level model that

identifies the aircraft family within that manufacturer, and a final model that distinguishes the specific variant.

6. Conclusion

Prompt learning for vision-language models has evolved from handcrafted prompts to learned context tokens and probabilistic variants. BPL [1] reframes prompt learning as variational inference and introduces KL regularization in prompt space to reduce overfitting. In our reproduction on FGVCAircraft under unseen prompt generalization, unconditional BPL consistently improves over standard CoOp, and we reproduce the trend that shorter context length generalizes better. However, we also find that the magnitude of this improvement is sensitive to baseline tuning: an overtrained CoOp baseline can inflate the apparent BPL gain. Finally, unconditional BPL is brittle in the very low Monte Carlo sample regime. Overall, Bayesian regularization improves robustness, but inference-time sampling and modeling assumptions (e.g., Gaussian residuals) remain important practical tradeoffs.

References

[1] Mohammad Mahdi Derakhshani, Enrique Sanchez, Adrian Bulat, Victor Guilherme Turrisi da Costa, Cees GM Snoek, Georgios Tzimiropoulos, and Brais Martinez. Bayesian prompt learning for image-language model generalization. *ICCV*, 2023. 1, 3, 4, 5, 6, 7, 8, 9, 10

[2] 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 (CVPR)*, pages 5206–5215, June 2022. 3

[3] S. Maji, J. Kannala, E. Rahtu, M. Blaschko, and A. Vedaldi. Fine-grained visual classification of aircraft. Technical report, University of Oxford, 2013. 6

[4] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021. 1, 2, 3, 4, 7, 8, 11

[5] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. Contrastive learning of medical visual representations from paired images and text. In Zachary Lipton, Rajesh Ranganath, Mark Sendak, Michael Sjoding, and Serena Yeung, editors, *Proceedings of the 7th Machine Learning for Healthcare Conference*, volume 182 of *Proceedings of Machine Learning Research*, pages 2–25. PMLR, 05–06 Aug 2022. 2

[6] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 3, 4, 5, 6, 7

[7] Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Learning to prompt for vision-language models. *International Journal of Computer Vision (IJCV)*, 2022. 1, 2, 3, 4, 5, 6, 8, 10

A. Choosing between implementing the experiments in Colab vs. locally

It is also worth noting that we decided to do our implementation in Google Colab even though the publicly available code repo for both CoOp/CoCoOp and BPL was designed to be run locally. The rationale was that the local machine available only had a 3080Ti GPU with 10GB of memory, and we found early on that running the BPL code often hit the GPU memory limits and would stop training. A workaround was developed to run in headless mode, but given this bottleneck was identified, we decided it would be safer to run on Google Colab. However, the tradeoff is that there is no persistent runtime, so you have to download and reload your data every single day and it is quite costly.

I plan to make the Colab public for future users who want to understand prompt learning. The contribution here is that the GitHub repos are designed to run locally and are efficient for generating experiments and ablations, but there are multiple dependencies that make it difficult to understand. For one, many of the hyperparameters are implicitly defined in another package called Dassl [7] where they define their learning schedules, and default optimizers values. There are also many scripts and layers, which makes it confusing and time-consuming to master and takes away from the principal aspects of learning the algorithms. A Colab notebook is sequential, so it is easier to follow at the expense of experimental efficiency.

B. Results from Bayesian Prompt Learning

	CoOp [55]	**CoCoOp** [54]	**ProDA** [32]	**Ours**
Caltech101	89.81	93.81	93.23	**94.93**$\pm$0.1
DTD	41.18	56.00	56.48	**60.80**$\pm$0.5
EuroSAT	54.74	60.04	66.00	**75.30**$\pm$0.7
FGVCAircraft	22.30	23.71	34.13	**35.00**$\pm$0.5
Flowers102	59.67	**71.75**	68.68	**70.40**$\pm$1.8
Food101	82.26	91.29	88.57	**92.13**$\pm$0.1
ImageNet	67.88	70.43	70.23	**70.93**$\pm$0.1
OxfordPets	95.29	97.69	97.83	**98.00**$\pm$0.1
StanfordCars	60.40	**73.59**	71.20	73.23$\pm$0.2
SUN397	65.89	76.86	76.93	**77.87**$\pm$0.5
UCF101	56.05	73.45	71.97	**75.77**$\pm$0.1
Average	63.22	71.69	72.30	**74.94**$\pm$0.2

Figure 3. Unseen prompts generalization [1]

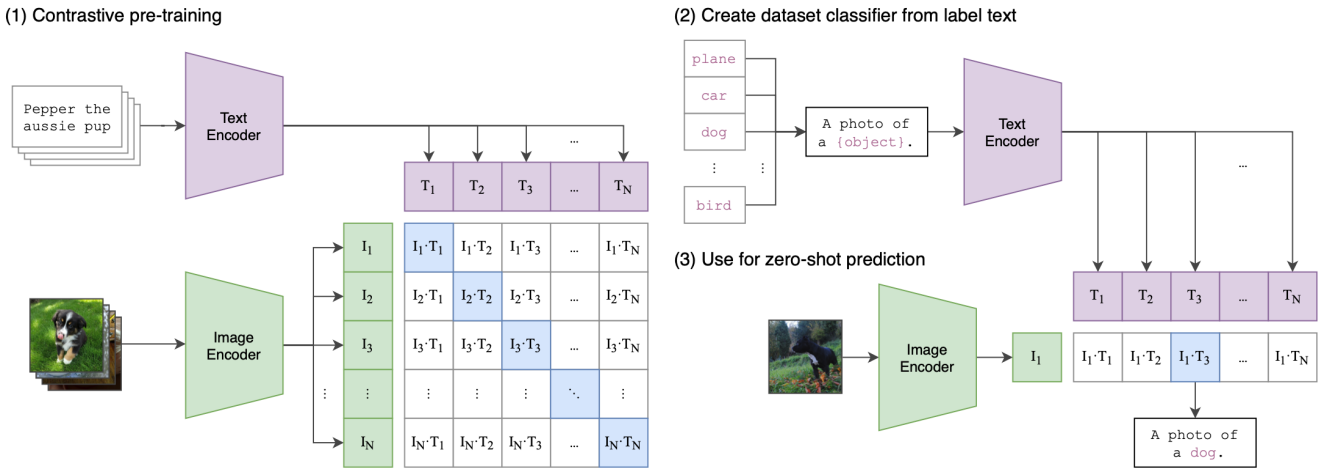


Figure 4. CLIP Vision-Language Model Architecture [4]