Bayesian Principles Improve Prompt Learning In Vision-Language Models

Mingyu Kim[*1] **Jongwoo Ko**[*2] **Mijung Park**[†1]
[1]UBC CS [2]KAIST AI
{mgyu.kim, mijung.park}@ubc.ca jongwoo.ko@kaist.ac.kr

Abstract

Prompt learning is a popular fine-tuning method for vision-language models due to its efficiency. It requires a small number of additional learnable parameters while significantly enhancing performance on target tasks. However, most existing methods suffer from overfitting to fine-tuning data, yielding poor generalizability. To address this, we propose a new training objective function based on a Bayesian learning principle to balance adaptability and generalizability. We derive a prior over the logits, where the mean function is parameterized by the pre-trained model, while the posterior corresponds to the fine-tuned model. This objective establishes a balance by allowing the fine-tuned model to adapt to downstream tasks while remaining close to the pre-trained model. To avoid the overfitting issues of the standard softmax function, we adopt the one-vs-each softmax approximation along with its Pólya-Gamma augmentation (OVE-PG). We evaluate our method on several benchmark datasets and demonstrate that using the Bayesian principle for prompt learning is indeed a sensible choice. Code is available at the official repository.

1 Introduction

Vision-Language models (VLMs), such as CLIP (Radford et al., 2021) and ALIGN (Jia et al., 2021), have been in the spotlight for their ability to generalize across unseen datasets, leveraging their universal knowledge. A key characteristic of these VLMs, particularly in image classification, is their capacity to classify novel classes without prior exposure by using simple class-specific text prompt (Radford et al., 2021; Jia et al., 2021). To enhance this ability, recent approaches have explored efficient transfer learning by employing trainable continuous prompt embeddings, which are fine-tuned using a small set of data points. These methods allow VLMs to adapt to new tasks while avoiding the exhausted computation costs typically associated with full fine-tuning (Zhou et al., 2022b,a). This approach extends their generalization in a modal-agnostic manner, which has been proven in the classification of natural language models and noisy label situations (Liu et al., 2023; Wu et al., 2023).

However, one issue that arises in prompt learning is still the risk of overfitting. It degrades performance on unseen datasets, as fine-tuning often leads to the forgetting of pre-trained knowledge (Khattak et al., 2023b). Initial attempts to mitigate have introduced knowledge distillation (KD) with pre-trained models (Khattak et al., 2023b; Yao et al., 2023; Oh et al., 2023). The KD regularization aims to maintain global knowledge while incorporating task-specific information. In parallel, Bayesian approaches to prompt learning have emerged as a way to handle diverse representations (Lu et al., 2022; Chen et al., 2023; Derakhshani et al., 2023a; Cho et al., 2024). These approaches treat multiple prompts as a population and model their distribution during fine-tuning. Each method designates its own form of regularization to account for the different representations that each prompt encodes.

Despite many appealing approaches, a significant limitation is that previous approaches have developed in isolation without integrating two strengths. Specifically, KD approaches and Bayesian methods for prompt learning have been explored independently. As a result, individual method remains incompatible with one another, limiting their potential when used together.

To address this gap, we propose a Bayesian princi-

[*]Equal contribution [†]Corresponding author

Proceedings of the 28[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2025, Mai Khao, Thailand. PMLR: Volume 258.

ple for prompt learning in VLMs that seamlessly integrates both distributional learning and knowledge distillation. First, the proposed method employs Pólya-gamma augmentation for composite likelihood, which helps accommodate the stochastic nature of the sigmoid function, thereby improving its distributional approximation for the softmax function. The standard softmax function is prone to overfitting specific one-hot target label information (Veličković et al., 2024), whereas the one-vs-each softmax approximation along with its pólya-gamma augmentation (OVE-PG) technique introduces advantageous noises and adjusts the logit functions using a prior mechanism during training (Snell and Zemel, 2021). Second, the proposed method aligns the KD regularization with the probabilistic nature. By representing all components, the proposed method synergizes with distributional logit function, leading to enhancing generalization. Notably, the proposed method remains fully compatible with previous approaches without requiring any modifications.

Our theoretical and experimental findings indicate that this method effectively harnesses global information. More specifically, it achieves an effective balancing between task-specific learning and global information, resulting to prompt representations suited for adaptation and generalization. The key contribution of this paper includes following:

1. We introduce a novel Bayesian principle, combining Polýa-Gamma augmentation and a one-vs-each softmax approximation, to mitigate overfitting in the prompt learning of VLMs. Our approach is simple and easily compatible with existing methods. We also demonstrate the efficacy of our method using 1D examples to provide clear intuition. **(Section 3.1 – 3.3)**
2. We also observe that our probabilistic framework can naturally be combined with KD regularization between the prior and posterior distributions of logit function, which is the inner product of image and text embeddings. Additionally, we provide insightful empirical validation showing that controlling the strength of this regularization enhances the effectiveness of our approach. **(Section 3.4)**
3. We show that OVE-PG significantly improves generalization for prompt learning in VLMs, despite its simplicity, and may alleviate overfitting better than complex regularization terms or additional network parameters. **(Section 5)**

2 Background

In this section, we provide background knowledge on prompt learning. Below, we start by describing the CLIP model and various prompt learning approaches.

2.1 Contrastive Language-Image Pretraining

The base model we consider in this paper is the CLIP model (Radford et al., 2021), which consists of an image encoder and a text encoder. Given an image input denoted by x and a corresponding text prompt denoted by $\boldsymbol{p}$, the image encoder denoted by $I(\boldsymbol{x})$ outputs the d-dimensional image embedding, while the text encoder denoted by $T(\boldsymbol{p})$ outputs the text embedding where the size of the word embedding is the text length and embedding dimension. Once the model is trained with the contrastive loss, the text embedding can serve as a linear classifier for novel datasets. That is, the logit function can be defined by the inner product between the image embedding and text embedding, where the text embedding plays the role of linear classifier weight. Specifically, consider a deterministic[1] function given an input $\boldsymbol{x}$ and a label c, $f^c(x) := I(\boldsymbol{x})^\top \boldsymbol{w}^c$. Here, w^c is a classifier weight for class c and the classifier weight is simply the text embedding of the c-th class prompt $\boldsymbol{w}^c = T(\boldsymbol{p}^c)$. Using this function, the predictive probability can then be represented by $p(y|x)=\frac{\exp[f^y(x)]}{\sum_c^C \exp[f^c(x)]}$.

2.2 Prompt Learning

While the classifiers based on CLIP showed unprecedented zero-shot classification accuracy, their manual prompting for each example is a limiting factor. Zhou et al. (2022b) and Zhou et al. (2022a) proposed the new line of work, so-called *prompt learning*, in which a small number of trainable parameters are defined and fine-tuned for target downstream tasks. In Context Optimization (CoOp) (Zhou et al., 2022b), learning the embedding matrix $\boldsymbol{p}_{\boldsymbol{\theta}}$ with parameters $\boldsymbol{\theta}$ was proposed. In this case, the logit function can be written as $f^c_{\boldsymbol{\theta}}(\boldsymbol{x}) := I(\boldsymbol{x})^\top T(\boldsymbol{p}^c_{\boldsymbol{\theta}})$. Here I and T are fixed. On the other hand, in Conditional Prompt Learning (CoCoOp) (Zhou et al., 2022a), both the tokens $\boldsymbol{p}_{\theta_1}$ identical to CoOp and an additional set of residual prompts $\boldsymbol{r}_{\theta_2}$ are learnable. This adjustment is designed to effectively tackle the domain shift by leveraging image-based features. In this approach, the logit function can be written as $f^c_{\boldsymbol{\theta}}(\boldsymbol{x}) := I(\boldsymbol{x})^\top T(\boldsymbol{p}^c_{\boldsymbol{\theta}_1} + \boldsymbol{r}_{\boldsymbol{\theta}_2}(\boldsymbol{x}))$. More recent work considers learning the distribution over the residual prompts and regularizing the prompt space to improve the prompt generalization on unseen prompts, e.g., (Lu et al., 2022; Derakhshani et al., 2023b). The logit function can take either of the forms in CoOp or CoCoOp: $f^c_{\boldsymbol{\theta}}(\boldsymbol{x}) = I(\boldsymbol{x})^\top T(\boldsymbol{p}^c_{\boldsymbol{\theta}_1} + \boldsymbol{r}_{\boldsymbol{\theta}_2}(\boldsymbol{x}))$.

[1] This is a deterministic function because once training is done, the image embedding is fixed, and the linear classifier weight is also fixed. Hence, given an arbitrary input, the output is deterministic.

What distinguishes our work from existing work is that we consider an approximated softmax function. This softmax function is bounded by one-vs-each approximation, which involves applying the sigmoid function to each pair-wise comparison with logit values. More precisely, a sigmoid function is applied to the difference between class-wise logit values. Additionally, we utilize the Pólya-gamma distribution to manage each sigmoid function effectively. We cover this in the following subsection.

2.3 Pólya-Gamma Data Augmentation

Pólya-Gamma (PG) data augmentation introduces an auxiliary variable per datapoint that is Pólya-Gamma distributed, such that the log-odds can be written as a mixture of Gaussians with respect to a Pólya-Gamma distribution (Nicholas G. Polson and Windle, 2013):

$$\frac{(e^{\psi})^a}{(1+e^{\psi})^b} = 2^{-b} e^{\kappa\psi} \int_0^\infty e^{-\omega\psi^2/2} p(\omega)\, d\omega, \quad (1)$$

where $\kappa = a - b/2$ and $\omega \sim \mathrm{PG}(b, 0)$. A particular interest is when $a = y$ and $b = 1$ and $\psi = \beta^\top x$, we recover an individual term of the logistic likelihood:

$$p(y|\psi) = \frac{(e^{\psi})^y}{1+e^{\psi}} = \frac{1}{2} e^{\kappa\psi} \int_0^\infty e^{-\omega\psi^2/2} p(\omega)\, d\omega,$$

where $\kappa = y - 1/2$ and $\omega \sim \mathrm{PG}(1, 0)$. Given a dataset of N datapoints, we can now write down the likelihood as Gaussian conditioning on ω:

$$\begin{aligned} p(\boldsymbol{y}|\boldsymbol{\psi}, \boldsymbol{\omega}) &\propto \prod_{i=1}^{N} e^{-\omega_i \psi_i^2/2} e^{\kappa_i \psi_i}, \\ &\propto \mathcal{N}(\Omega^{-1}\boldsymbol{\kappa}|\psi, \Omega^{-1}), \end{aligned} \quad (2)$$

where $\kappa_i = y_i - 1/2$ and $\Omega = \mathrm{diag}(\boldsymbol{\omega})$. Since we can write down the likelihood term as Gaussian when combined with the Gaussian prior over $\boldsymbol{\psi} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$, the posterior also becomes Gaussian conditioning on the PG variables $\boldsymbol{\omega}$:

$$\begin{aligned} p(\boldsymbol{\psi}|\boldsymbol{y}, \boldsymbol{\omega}) &\propto p(\boldsymbol{y}|\boldsymbol{\psi}, \boldsymbol{\omega}) p(\boldsymbol{\psi}) \\ &\propto \mathcal{N}(\boldsymbol{\psi}|\tilde{\Sigma}(\Sigma^{-1}\boldsymbol{\mu} + \boldsymbol{\kappa})\tilde{\Sigma}) \end{aligned} \quad (3)$$

where $\tilde{\Sigma} = (\Sigma^{-1} + \Omega)^{-1}$. The posterior inference is now analytically tractable given $\boldsymbol{\omega}$. To obtain the posterior distribution the PG variables $\boldsymbol{\omega}$, we can also employ the closed-form update given by (Nicholas G. Polson and Windle, 2013):

$$\begin{aligned} p(\boldsymbol{\omega}|\boldsymbol{y}, \boldsymbol{\psi}) &\propto \mathrm{PG}(\boldsymbol{\omega}|1, 0) e^{-\boldsymbol{\omega}\boldsymbol{\psi}^2/2} \\ &\propto \mathrm{PG}(\boldsymbol{\omega}|1, \boldsymbol{\psi}). \end{aligned} \quad (4)$$

The Gibbs sampling procedure that alternates the two updates for $\boldsymbol{\psi}$ in Equation 3 and $\boldsymbol{\omega}$ in Equation 4 yields the joint posterior distributions over the function $\boldsymbol{\psi}$ and the PG variables $\boldsymbol{\omega}$.

Algorithm 1 One-vs-Each SoftMax Approximation with Pólya-Gamma Auxiliary Variables for Prompt Learning

Input: Objective $\mathcal{L}_{\text{elbo}} = \{\mathcal{L}_{\text{nll}}, \mathcal{L}_{\text{KLD}}\}$, Class prompts $\mathcal{C} \in \{c_1, \cdots, c_C\}$, The learnable prompts $\mathbf{p}_{\boldsymbol{\theta}_1}$, and residual prompts $\boldsymbol{r}_{\boldsymbol{\theta}_2}$, The predefined prompts for the original CLIP $\mathbf{t} = \{t_1, \cdots, t_C\}$, Image-Encoder I and Text-encoder T, The number of parallel Gibbs chains M, Total number of classes C, Learning rate η, Weight for KLD β, α prior precision of $\boldsymbol{f}$, Einstein summation convention, denoted by $\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f}$.

Initialize: Parameters $\boldsymbol{\theta} = \{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2\}$ randomly.

repeat

Sample a mini-batch $(\boldsymbol{x}, \boldsymbol{y})$ from a training data $\mathcal{D}$, where $\boldsymbol{x} \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \{0, 1\}^{n \times C}$

$\boldsymbol{A} \leftarrow$ OVE-MATRIX$(\boldsymbol{y}) \in \mathbb{R}^{C \times C \times C}$ in eq.(11)

$\boldsymbol{\mu}_{\boldsymbol{\theta}} = I(\boldsymbol{x})^T T(\boldsymbol{p}^c_{\boldsymbol{\theta}_1} + \boldsymbol{r}^c_{\boldsymbol{\theta}_2}(\boldsymbol{x})) \in \mathbb{R}^{n \times C}$

$\boldsymbol{\mu} = I(\boldsymbol{x})^T T(\boldsymbol{p}^c) \in \mathbb{R}^{n \times C}$

$\boldsymbol{\psi}_{\boldsymbol{\theta}, nic} \leftarrow \sum_j \boldsymbol{A}_{ijc} \boldsymbol{\mu}_{\boldsymbol{\theta}, ni}, \quad \boldsymbol{\psi}_{\boldsymbol{\theta}} \in \mathbb{R}^{N \times C \times C}$

$\boldsymbol{\psi}_{nic} \leftarrow \sum_j \boldsymbol{A}_{ijc} \boldsymbol{\mu}_{ni}, \quad \boldsymbol{\psi} \in \mathbb{R}^{n \times C \times C}$

$\boldsymbol{\kappa}_{nic} = \sum_j \boldsymbol{A}_{ijc} \big(\boldsymbol{y}_{ni} - 1/2\big), \quad \boldsymbol{\kappa} \in \mathbb{R}^{n \times C \times C}$

for $m = 1$ **to** M **do**

$\boldsymbol{\omega}^{(m)} \sim \mathrm{PG}(1, \boldsymbol{\psi}) \in \mathbb{R}^{n \times C \times C}$ given in eq.(14)

$\boldsymbol{\psi}^{(m)}_{\boldsymbol{\theta}} \sim p(\boldsymbol{\psi}^{(m)} | \boldsymbol{\omega}^{(m)}, \boldsymbol{\psi}_{\boldsymbol{\theta}}, \boldsymbol{\kappa}, \alpha)$ given in eq(.3)

$\boldsymbol{f}^{(m)}_{\boldsymbol{\theta}} \leftarrow \sum_C \boldsymbol{\psi}^{(m)}_{\boldsymbol{\theta}} \in \mathbb{R}^{n \times C}$ due to $\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f}_{\boldsymbol{\theta}}$

$\mathcal{L}^{(m)}_{\text{nll}} \leftarrow -\sum_{i=1}^{N} \sum_{c=1}^{C} y_{i,c} \log f_{i,c}$ given in eq.(9)

$\mathcal{L}^{(m)}_{\text{KLD}} \leftarrow \beta \|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}\|_2^2$

end for

$\theta \leftarrow \theta - \frac{\eta}{M} \sum_{m=1}^{M} \nabla_\theta \big(\mathcal{L}^m_{\text{nll}} + \mathcal{L}^m_{\text{KLD}}\big)$

until convergence

3 Method

Our method involves specifying the key components: (1) the likelihood, (2) the prior, and (3) the posterior distributions. We then explain how to perform posterior inference and parameter estimation. The complete algorithm is presented in algorithm 1. The subsequent sections provide a detailed explanation of each component.

3.1 Prior over the Logit Function

As mentioned earlier, we consider the logit function in the following form:

$$f^c_{\boldsymbol{\theta}}(\boldsymbol{x}) = I(\boldsymbol{x})^\top T(\boldsymbol{p}^c_{\boldsymbol{\theta}} + \boldsymbol{r}^c_{\boldsymbol{\theta}}(\boldsymbol{x})) \quad (5)$$

with frozen I and T. In the following for the cleanness of the notation, we omit the f's dependency on $\boldsymbol{\theta}$, and simply write f for $f_{\boldsymbol{\theta}}$.

Given training data $\mathcal{D} = \{\boldsymbol{X}, \mathbf{Y}\}$ which consists of input points $\mathbf{X} \in \mathbb{R}^{N \times D}$ and corresponding one-hot

labels $\mathbf{Y} \in \{0,1\}^{N\times C}$, where C is the number of classes, we denote the logit function evaluated at those points as a single vector of length N

$$\mathbf{f} := (f_1^1, \ldots, f_N^1, f_1^2, \ldots, f_N^2, \ldots, f_1^C, \ldots, f_N^C)^\top \quad (6)$$

and place an independent GP prior on the logits for each class: $f_i^c(\mathbf{x}) \sim \mathcal{GP}(\mu(\mathbf{x}_i), k(\mathbf{x}_i, \mathbf{x}_j))$. The marginals on those points $\boldsymbol{X}$ are jointly Gaussian $p(\boldsymbol{f}|\mathcal{D}) \propto \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \boldsymbol{K})$ where $\mu_i^c = \mu(\mathbf{x}_i)$ and $\mathbf{K}$ is block diagonal with $K_{ij}^c = k(\mathbf{x}_i, \mathbf{x}_j)$ for each block $\mathbf{K}^c$.

To encode the information on the diverse sets of pre-training data into the model, we set $\boldsymbol{\mu}$ to the logit function of the pre-trained model, $\mu_i(\boldsymbol{x}) = I(\boldsymbol{x}_i)^\top T(\boldsymbol{p}^c)$ and $\boldsymbol{K}$ to an identity matrix with a small variance. This reflects that *a priori* we believe that the best function value the model can take is the logit function from the pretrained model (because that model already saw so much of the world) with high confidence, where the prior precision is measured by a hyperparameter α:

$$\text{Prior: } p(\boldsymbol{f}|\mathcal{D}) \propto \mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \alpha^{-1}\boldsymbol{I}) \quad (7)$$

3.2 Likelihood with the Logit Function

The softmax likelihood under the multi-class classification setting with a function f is given by

$$p(\boldsymbol{y}_i|\boldsymbol{f}(\boldsymbol{x}_i)) = \frac{\exp[\boldsymbol{f}^{\boldsymbol{y}_i}(\boldsymbol{x}_i)]}{\sum_c^C \exp[\boldsymbol{f}^c(\boldsymbol{x}_i)]}. \quad (8)$$

Here we make an *one-vs-each (OVE)* approximation (Titsias RC AUEB, 2016) to the likelihood is given by

$$\mathcal{L}_{\text{OVE}}(\mathcal{D}|\boldsymbol{f}) = \prod_i \left(\prod_{c\neq c'} \sigma(\boldsymbol{f}_i^c - \boldsymbol{f}_i^{c'}) \right)^{\boldsymbol{y}_i} \quad (9)$$

See Supplementary material for detailed explanation on the OVE likelihood. Equipped with the OVE likelihood, we further add Pólya-Gamma auxiliary variables $\boldsymbol{\omega} \in \mathbb{R}^N$ to the model. Due to Equation 2, we can represent the conditional likelihood of $\boldsymbol{\psi}$ (conditioned on $\boldsymbol{\omega}$) as Gaussian:

$$\begin{aligned} \text{Likelihood: } \mathcal{L}_{OVE}(\boldsymbol{\psi}|\boldsymbol{\omega}, \mathcal{D}) &\propto \prod_{j=1}^{NC} e^{-\boldsymbol{\omega}_j \boldsymbol{\psi}_j^2/2} \boldsymbol{\kappa}_j^{\psi_j} \\ &\propto \mathcal{N}(\boldsymbol{\Omega}^{-1/2}\boldsymbol{\kappa}|\boldsymbol{\psi}, \boldsymbol{\Omega}^{-1}) \end{aligned} \quad (10)$$

where $\kappa_j = 1/2$, $\boldsymbol{\Omega} = \text{diag}(\boldsymbol{\omega})$ and $\boldsymbol{\psi}_{nic} = \sum_j \boldsymbol{A}_{ijc}\boldsymbol{f}_{ni}$ following the Einstein summation convention. For simplicity, we use the Einstein summation convention, denoted by $\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f}$ between tensors and matrices unless otherwise specified. Here, an OVE-matrix $\boldsymbol{A}$ is a $C \times C \times C$ sparse block matrix. $\boldsymbol{A}_c$ corresponds to a square matrix of dimension of C. Each block $\boldsymbol{A}_c$ matrix defined as follows:

$$\boldsymbol{A}_c := \mathbf{1}_C \mathbf{e}_c^\top - \mathbf{I} \quad (11)$$

where, $\mathbf{1}_C$ is a column vector of dimension C with all components equal to 1 and the standard basis vector $\mathbf{e}_c = (0, 0, \ldots, 0, \underbrace{1}_{c\text{-th}}, 0, \ldots, 0)^\top$. A simple example of computation $\boldsymbol{A}$ and $\boldsymbol{f}$ is provided in Appendix.

3.3 Posterior

The exact posterior over $\boldsymbol{\psi}$ with the OVE likelihood together with the Pólya-Gamma data augmentation is given in Equation 3. This translates to the posterior over $\boldsymbol{f}$ given by (Snell and Zemel, 2021)

$$\begin{aligned} p(\boldsymbol{f}|\boldsymbol{\omega}, \mathcal{D}) &\propto \mathcal{L}_{\text{OVE}}(\boldsymbol{\psi}|\boldsymbol{\omega}, \mathcal{D})\mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \alpha^{-1}\boldsymbol{I}), \\ &\propto \mathcal{N}(\boldsymbol{\Omega}^{-1/2}\boldsymbol{\kappa}|\boldsymbol{A}\boldsymbol{f}, \boldsymbol{\Omega}^{-1})\mathcal{N}(\boldsymbol{f}|\boldsymbol{\mu}, \alpha^{-1}\boldsymbol{I}), \\ &\propto \mathcal{N}(\boldsymbol{f}|\boldsymbol{\Sigma}(\alpha\boldsymbol{\mu} + \boldsymbol{A}^\top\boldsymbol{\kappa}), \boldsymbol{\Sigma}) \end{aligned} \quad (12)$$

where $\boldsymbol{\Sigma} = (\alpha I + \boldsymbol{A}^\top\boldsymbol{\Omega}\boldsymbol{A})^{-1}$. We modify this to an approximate posterior over f:

$$\text{Posterior: } p(\boldsymbol{f}|\boldsymbol{\omega}, \mathcal{D}) \approx \mathcal{N}(\boldsymbol{f}_\theta|\boldsymbol{\mu}_\theta, \boldsymbol{\Lambda}) \quad (13)$$

where we set the mean function to the logit function of the prompt tuning model, i.e., $\boldsymbol{\mu}_\theta(\boldsymbol{x}) = I(\boldsymbol{x})^\top T(\boldsymbol{p}_{\boldsymbol{\theta}} + \boldsymbol{r}_{\boldsymbol{\theta}}(\boldsymbol{x}))$, and the covariance to $\Lambda = (\alpha\mathbf{I} + \boldsymbol{\Omega})^{-1}$. Notice that the mean function is the only quantity depending on the parameters of the prompt tuning model. Intuitively, sampling the posterior is guided by the dataset, adjusting logit values according to label information. Correct labels typically result in high logit values, reflecting higher confidence. Meanwhile, incorrect labels have lower logit values, indicating less confidence. We assume that the model learns to follow this behaviour during training. The PG distribution over the auxiliary variables is a conjugate prior for the given likelihood, so the posterior over the auxiliary variables is also PG distributed:

$$\begin{aligned} \text{Prior: } p(\boldsymbol{\omega}) &\propto \prod_{i=1}^{N} \text{PG}(\boldsymbol{\omega}_i|1, 0) \\ \text{Posterior: } p(\boldsymbol{\omega}|\boldsymbol{f}, \mathcal{D}) &\propto \text{PG}(\boldsymbol{\omega}|1, \boldsymbol{A}\boldsymbol{f}) \end{aligned} \quad (14)$$

This posterior has to be updated whenever we have a new value for $\boldsymbol{f}$. However, in our implementation, we simply set $\boldsymbol{f}$ to the prior mean $\boldsymbol{\mu}$, where we set the prior mean $\boldsymbol{\mu}$ to the logit function of the pre-trained CLIP model.

3.4 Parameter Estimation

The ultimate goal is maximizing the likelihood of data after integrating out the unknowns $\boldsymbol{f}$ and $\boldsymbol{\omega}$. We turn our attention to the variational lower bound using an

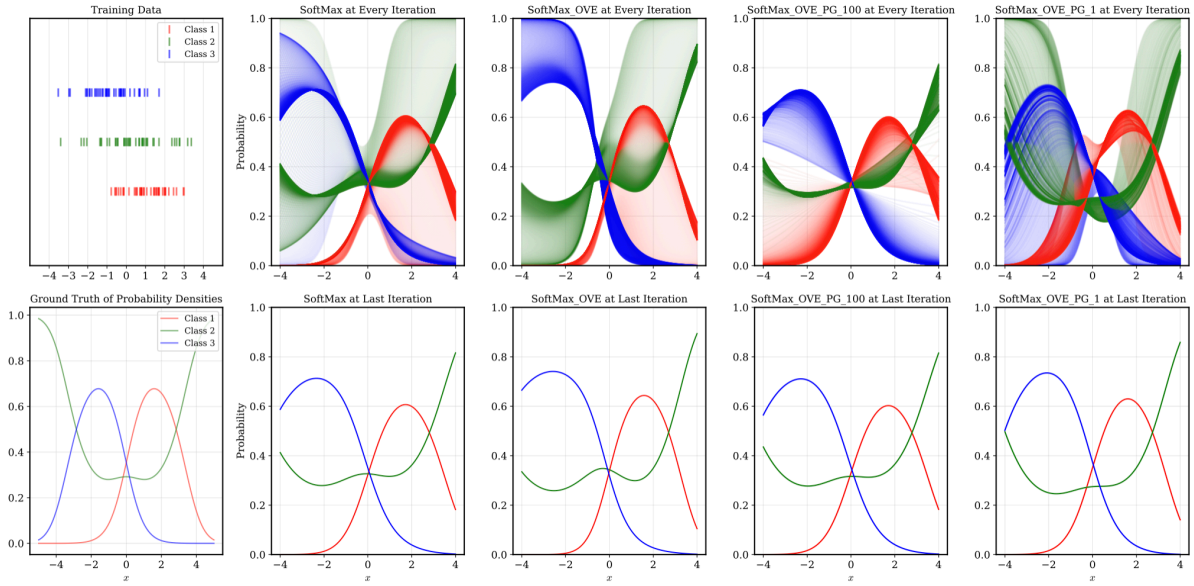

Figure 1: 1D synthetic classification comparison between SoftMax, SoftMax_OVE and SoftMAX_OVE_PG based on the gradient-descent. **[Column 1]** illustrates the arrangement of training samples **(top)** and the underlying ground-truth distributions **(bottom)**. **[Columns 2-5]** show the optimized functions from SoftMax, SoftMax_OVE, and SoftMax_OVE_PG with the precision α values of 1 and 100, respectively. In these columns, the top row displays the class distributions at every iteration, while the bottom row shows the class distribution at the last iteration.

approximating joint posterior over $\boldsymbol{f}, \boldsymbol{\omega}$ denoted by $q(\boldsymbol{f}, \boldsymbol{\omega})$:

$$
\begin{aligned}
&\log p(\mathcal{D}) \\
&= \log \int_{\boldsymbol{\omega}} \int_{\boldsymbol{f}} p(\mathcal{D}|\boldsymbol{f}, \boldsymbol{\omega}) p(\boldsymbol{f}, \boldsymbol{\omega}) d\boldsymbol{\omega} d\boldsymbol{f}, \\
&= \log \int_{\boldsymbol{\omega}} \int_{\boldsymbol{f}} q(\boldsymbol{f}, \boldsymbol{\omega}) p(\mathcal{D}|\boldsymbol{f}, \boldsymbol{\omega}) \frac{p(\boldsymbol{f}, \boldsymbol{\omega})}{q(\boldsymbol{f}, \boldsymbol{\omega})} d\boldsymbol{\omega} d\boldsymbol{f}, \\
&\geq \int_{\boldsymbol{\omega}} \int_{\boldsymbol{f}} q(\boldsymbol{f}, \boldsymbol{\omega}) \log \left(p(\mathcal{D}|\boldsymbol{f}, \boldsymbol{\omega}) \frac{p(\boldsymbol{f}, \boldsymbol{\omega})}{q(\boldsymbol{f}, \boldsymbol{\omega})} \right) d\boldsymbol{\omega} d\boldsymbol{f}, \\
&\text{Due to Jensen's inequality} \\
&= \int_{\boldsymbol{\omega}} \int_{\boldsymbol{f}} q(\boldsymbol{f}, \boldsymbol{\omega}) \log p(\mathcal{D}|\boldsymbol{f}, \boldsymbol{\omega}) d\boldsymbol{\omega} d\boldsymbol{f} \\
&\quad - D_{KL}[q(\boldsymbol{f}, \boldsymbol{\omega}) || p(\boldsymbol{f}, \boldsymbol{\omega})]. \qquad (15)
\end{aligned}
$$

Now we discuss each of these terms in Equation 15. The first term, expected log-likelihood of data, can be approximated via Monte Carlo integration using the samples of $\boldsymbol{f}$ and $\boldsymbol{\omega}$:

$$
\begin{aligned}
&\log \int_{\boldsymbol{\omega}} \int_{\boldsymbol{f}} p(\mathcal{D}|\boldsymbol{f}, \boldsymbol{\omega}) q(\boldsymbol{f}, \boldsymbol{\omega}) d\boldsymbol{\omega} d\boldsymbol{f} \\
&\approx \log \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_{\text{OVE}}(\boldsymbol{f}_{\boldsymbol{\theta}_i} \boldsymbol{A} \mid \boldsymbol{\omega}_i, \mathcal{D}), \qquad (16)
\end{aligned}
$$

where the samples can be drawn from the posteriors given in Equation 13 and Equation 14. For speedy training, we can fix $\boldsymbol{\omega}$ to its posterior mean using Equation 14, and sample only for $\boldsymbol{A}\boldsymbol{f}^s_{\boldsymbol{\theta}}$. Notice that $\boldsymbol{f}^s_{\boldsymbol{\theta}}$ is the only quantity that depends on the parameters of prompt learning, through $\boldsymbol{f}^s_{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \boldsymbol{\Lambda})$.

For the second term, we first write down the joint distribution as $q(\boldsymbol{f}, \boldsymbol{\omega}) = q(\boldsymbol{f}|\boldsymbol{\omega}) q(\boldsymbol{\omega})$ and $p(\boldsymbol{f}, \boldsymbol{\omega}) = p(\boldsymbol{f}|\boldsymbol{\omega}) p(\boldsymbol{\omega})$. In our prior formulation in Equation 7, the GP function $\boldsymbol{f}$ is independent of the PG variable $\boldsymbol{\omega}$, yielding $p(\boldsymbol{f}|\boldsymbol{\omega}) = p(\boldsymbol{f})$. If we fix $\boldsymbol{\omega}$ to some value $\hat{\boldsymbol{\omega}}$, e.g., its posterior mean with Equation 14, i.e., $\hat{\omega} = \mathbb{E}_{p(\boldsymbol{\omega}|\boldsymbol{f}, \mathcal{D})}[\boldsymbol{\omega}]$ and set $q(\boldsymbol{f}|\hat{\boldsymbol{\omega}})$ to the Gaussian posterior given in Equation 13 and $p(\boldsymbol{f})$ to the Gaussian prior given in Equation 7, the second term becomes the KL divergence between posterior and prior distributions over $\boldsymbol{f}$ which is in closed form:

$$
\begin{aligned}
D_{KL}[q(\boldsymbol{f}, \boldsymbol{\omega}) || p(\boldsymbol{f}, \boldsymbol{\omega})] &\approx D_{KL}[q(\boldsymbol{f}|\hat{\boldsymbol{\omega}}) || p(\boldsymbol{f})], \\
&\approx \|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}\|_2^2, \qquad (17)
\end{aligned}
$$

where in the last line, we further approximate the KL divergence between the two Gaussians by writing down only the term that depends on the parameters in prompt learning. The details of approximation can be found in subsection C.4. To be able to control the strength of this regularization, we add a hyperparameter β. Our algorithm is summarized in algorithm 1.

We construct the sampling procedure in the space of ψ for computational efficiency. We assume no correlation between $\boldsymbol{f}^c_i$ and $\boldsymbol{f}^{c'}_i$. Given that $\boldsymbol{A}$ is a sparse matrix and $\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f}$ is a deterministic projection, $\boldsymbol{A}\boldsymbol{f}$

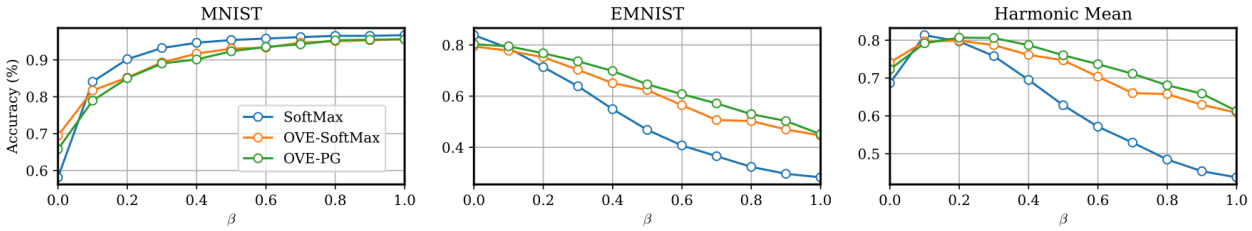

Figure 2: Effect of OVE-PG on ViT using the EMNIST and MNIST datasets. Our proposed OVE-PG (green line) achieves better generalization compared to Softmax (blue line) and OVE-Softmax (orange line) on unseen datasets (i.e., EMNIST), while slightly compromising performance on seen datasets (i.e., MNIST), resulting in higher overall performance.

represents the subtraction of two logit values, leading to doubling their variance. Additionally, when considering $\boldsymbol{A}^T\boldsymbol{\kappa}$ in Equation 12, it reverts the one-hot label matrix and extend it along another axis by duplicating. For $\boldsymbol{A}^T\boldsymbol{\Omega}\boldsymbol{A}$, it influences the variance of the posterior distribution of $\boldsymbol{f}$. Since $\boldsymbol{A}$ is a sparse matrix with each row containing only two components, 1 or -1, we approximate $\boldsymbol{A} \approx \boldsymbol{I}$. In this context, the components remain uncorrelated. Without additional modification, sampling in the $\boldsymbol{\psi}$ space improves computational efficiency.

Empirical Validation with Synthetic Examples. We compare OVE_PG (algorithm 1) with SoftMax (Equation 8) and its OVE approximation (SoftMax_OVE; Equation 9) in a synthetic 1D classification task. As shown in the second row of Figure 1, all methods converge to the same result. However, their dynamic behaviour during training differs, as seen in the first row. Initially, SoftMax and its OVE approximation appear identical, but the Pólya-Gamma variable and prior precision in OVE result in the extent of exploration. Specifically, when the precision $\alpha = 100$, the prior for each sigmoid function has a small variance, while $\alpha = 1$ results in a larger variance, promoting greater exploration during training. This exploration helps avoid overfitting and improves performance. The detailed procedure is provided in Appendix.

Empirical Validation with ViT. We also evaluate our proposed method on image classification using ViT (Dosovitskiy et al., 2021). The ViT model is first trained on MNIST with SoftMax and then fine-tuned on EMNIST with SoftMax, OVE, and OVE-PG, respectively. We tested on MNIST, EMNIST and the harmonic mean of both datasets. A sensitivity analysis on the hyperparameter β illustrates how performance changes with different values. Details of the experimental setup are in the Appendix. As shown in Figure 2, OVE-PG consistently outperforms across most β values, while softmax shows a notable performance drop on EMNIST and the harmonic mean as β increases.

4 Related Work

In this section, we review relevant literature related to our work, including vision-language models, prompt learning for such models, and Bayesian inference for multi-class.

Vision-Language Models (VLMs). VLMs address the shortcomings of vision-only supervised learning by introducing robustness and flexibility in zero-shot inference through natural language supervision. A pioneering work in this field is CLIP (Radford et al., 2021), which employs contrastive learning on a massive dataset of 400 million image-text pairs. Building on this foundation, ALIGN (Jia et al., 2021) further scales up the dataset with an even larger collection of noisy image-text pairs to enhance performance. CoCa (Yu et al., 2022) integrates both captioning and contrastive losses, effectively combining the strengths of contrastive methods like CLIP with generative approaches. EVA-CLIP (Sun et al., 2023) applies various training strategies, including alternative attention mechanisms and optimizers, to further boost CLIP's performance. Additionally, SigLIP (Zhai et al., 2023) replaces the traditional softmax loss with a sigmoid loss, enabling more efficient pretraining with smaller batch sizes.

Prompt Learning in VLMs. Parameter-efficient fine-tuning with soft prompts, originally from natural language processing, has gained attention (Lester et al., 2021). This technique has been applied in vision-language models to adapt to downstream tasks. CoOp Zhou et al. (2022b) first introduced learnable prompts for CLIP, replacing manual ones. CoCoOp conditions text prompts on images to prevent overfitting Zhou et al. (2022a), while KgCoOp minimizes the gap between learned and manual prompts (Yao et al., 2023). ProGrad aligns gradient directions to retain general knowledge (Zhu et al., 2022), and PromptSRC uses

	CoOp		CoCoOp		MaPLe		APEX	
	SoftMax	OVE-PG	Softmax	OVE-PG	SoftMax	OVE-PG	SoftMax	OVE-PG
Caltech101	94.43	**94.93**	94.03	**94.13**	94.00	**94.55**	**95.00**	94.77
DTD	54.97	**55.67**	55.07	**56.17**	50.97	**57.33**	58.67	**61.13**
EuroSAT	55.20	**71.23**	66.63	**68.03**	60.70	**63.43**	68.23	**71.37**
FGVC Aircraft	22.93	**36.23**	**34.13**	33.60	34.17	**36.97**	34.83	**35.13**
Food101	91.70	**91.80**	**91.73**	91.70	91.40	**91.60**	**91.80**	91.73
Flowers102	72.47	**73.83**	74.07	**74.33**	73.13	**75.60**	73.37	**74.67**
Oxford Pet	**97.77**	97.70	**97.63**	96.47	**97.87**	97.63	**96.83**	96.80
Stanford Cars	72.73	**75.23**	74.70	**75.50**	72.40	**74.07**	**74.50**	74.40
SUN397	75.90	**77.53**	76.87	**77.40**	76.77	**78.40**	**76.97**	76.87
UCF101	72.43	**76.20**	72.63	**73.27**	76.40	**78.47**	74.77	**75.23**
Average (Unseen)	71.05	**75.04**	73.75	**74.06**	72.78	**74.91**	74.50	**75.21**

Table 1: Comparison of accuracy (%) on unseen classes between Softmax and OVE-PG (Ours) for CoOP, CoCoOp, MaPLe, and APEX. The best performance within each shared base algorithm is highlighted in **bold**.

multiple regularization losses with Gaussian aggregation of weights to prevent forgetting (Khattak et al., 2023b). Recent studies reported that visual learnable prompts significantly enhance generalization capability (Khattak et al., 2023a; Yang et al., 2024). Derakhshani et al. (2023b) proposed a Bayesian approach to reduce overfitting by introducing an additional network. In contrast, our method shows that a simple Bayesian approach can effectively mitigate overfitting without requiring extra regularization or network parameters.

Bayesian Inference for Multi-Class. The logistic function is robust in handling outliers for classification tasks, but maintaining conjugacy between prior and posterior distributions is challenging. Initially, for binary classification, the sigmoid function was modeled as a Gaussian distribution conditioned on a variable from the Pólya-Gamma distribution (Nicholas G. Polson and Windle, 2013). However, this approach doesn't naturally extend to the softmax function. Inspired by Dirichlet processes, the stick-breaking concept has been applied to transform multi-class classification into a series of binary classifications, distinguishing each class from the others (Linderman et al., 2015). Since this method builds on sequence dependency, changing the order leads a different outcome. To address this, the softmax function is decomposed into several sigmoid functions based on the composite likelihood principle (Titsias RC AUEB, 2016), each retaining conjugacy with Pólya-Gamma variables (Snell and Zemel, 2021). While this technique showed theoretical and empirical results in few-shot classification tasks, its reliance on Gaussian processes is a limitation. Specifically, it introduces significant computational burdens as the number of classes and data points increases. Due to this complexity, this approach typically assumes a maximum of five classes and data points to maintain feasibility. In contrast, the proposed method builds on this framework but leverages prior knowledge from a pre-trained VLM. By sampling an appropriate logit function with an approximated softmax, it reduces computational complexity while retaining the original behaviour.

5 Experiments

Here, we present the experimental results that verify the effectiveness of our proposed method. We begin by introducing the experimental setup, followed by the relevant results for image recognition using CLIP.

5.1 Experimental Setup

We evaluate our proposed method on the unseen prompts generalization task, focusing on 10 image recognition datasets, following Zhou et al. (2022a). These datasets cover multiple recognition tasks, including Caltech101 (Fei-Fei et al., 2004a) which consists of generic objects; Oxford Pets (Parkhi et al., 2012), FGVC Aircraft (Maji et al., 2013), Flowers102 (Nilsback and Zisserman, 2008), Food101 (Bossard et al., 2014a), and Stanford Cars (Krause et al., 2013) for fine-grained classification, UCF101 (Soomro et al., 2012a) for action recognition, SUN397 (Xiao et al., 2010a) for scene recognition, UCF101 (Soomro et al., 2012b) for action recognition, DTD (Cimpoi et al., 2014) for texture classification, and EuroSAT (Helber et al., 2019), which consists of satellite images.

We integrate our proposed method into CoOp (Zhou et al., 2022b), CoCoOp (Zhou et al., 2022a), MaPLe (Khattak et al., 2023a) and APEX (Yang et al., 2024), as well as compare them to the baseline versions with the vanilla softmax function. In this experiment, we pay attention to how effectively the learned prompts generalize to unseen classes or domains. We utilize the ViT-B/16 model as the CLIP image encoder and

a standard GPT2-like structure with an End-Of-Text (EOT) token as the classification token for the text encoder. For training, we follow the approach outlined in Derakhshani et al. (2023b), except for using a batch size of 4. We use the SGD optimizer with a learning rate of 0.002 and a cosine learning rate scheduler. Also, all evaluations are conducted for three different seeds. The detailed experimental setup can be found in Appendix.

5.2 Main Results

Seen-to-Unseen Generalization. We evaluate the generalization performance of our proposed method, OVE-PG, on 10 datasets, each tested with multiple random seeds. In each dataset, classes are split into seen and unseen subsets. Our model is trained on the seen classes and evaluated on the unseen classes. As shown in the Table 1, OVE-PG consistently outperforms the softmax baseline for all existing approaches. Notably, OVE-PG with CoOp achieves the highest gap (+3.99%) among all cases, which indicates that our regularization effect sufficiently alleviates overfitting on seen classes. Also, OVE-PG provide consistent performance improvement across all baselines, including CoCoOp (+0.31%), MaPLe (+2.13%), and APEX (+0.71%), reinforcing its effectiveness in generalizing to unseen classes. Additionally, we report the performance on seen categories and the harmonic mean between seen and unseen categories provided in Appendix. Consequently, while OVE-PG introduces slight performance degradation in seen classes, it achieves higher or comparable performance from the perspective of the harmonic mean, which considers both seen-and-unseen generalization.

Cross-dataset Evaluation. To enable the model to generalize across different domains, we employed a cross-dataset evaluation task. Specifically, we first trained the model on ImageNet (Deng et al., 2009) and then transferred it to six other datasets. As shown in Table 2, OVE-PG achieves higher average performances compared to Softmax baselines. Notably, OVE-PG outperforms in four out of six datasets for both CoOp and CoCoOp cases, while only slightly compromising or even improving performance on the source dataset. This demonstrates the effectiveness of our proposed method, especially in challenging situations where both the task and domain are unseen.

5.3 Analysis

Component Analysis. To demonstrate the effectiveness of our PG augmentation, we compare OVE-PG to OVE without PG augmentation, as reported in Table 3. This indicates our performance improvements are primarily due to the novel PG augmentation, rather than the OVE method itself. Notably, while

		CoOp		CoCoOp	
		SoftMax	OVE-PG	SoftMax	OVE-PG
Source	ImageNet	**76.5**	74.5	75.9	**76.1**
Target	DTD	51.7	**57.4**	54.5	**55.4**
	EuroSAT	51.3	**61.7**	52.5	**55.6**
	Aircraft	**26.7**	23.8	24.2	**26.8**
	Flower102	**72.6**	71.5	**73.8**	71.5
	Cars	62.8	**63.6**	**65.2**	64.9
	UCF101	67.2	**69.9**	69.2	**70.6**
Average		58.40	**60.34**	59.56	**60.13**

Table 2: Comparison of accuracy (%) on cross-dataset tasks between Softmax and OVE-PG (Ours) for CoOP and CoCoOp. The best performance within each shared base algorithm is highlighted in **bold**.

	DTD	EuroSAT	FGVC Aircraft	UCF101
SoftMax	54.97	55.20	22.93	72.43
OVE	50.93 (-4.04%)	54.93 (-0.27%)	16.97 (-5.96%)	76.03 (+3.60%)
OVE-PG	**55.67** (+0.70%)	**71.23** (+16.03%)	**36.23** (+13.30%)	**76.20** (+3.77%)

Table 3: Comparison of accuracy (%) on unseen classes between OVE and OVE-PG (Ours) for CoOP. The results indicate that our PG augmentation significantly improve the performance of VLMs.

OVE performed worse on datasets where CoOp already struggled (e.g., DTD, EuroSAT, and FGVC Aircraft), OVE-PG significantly outperforms the baselines. Additionally, OVE-PG surpasses OVE even in datasets where OVE performs well, such as UCF101.

Sensitivity Analysis. The main hyperparameter for our proposed method is β, which controls the strength of the KL divergence between the pre-trained and fine-tuned distributions. To understand the effect of this hyperparameter, we conducted a sensitivity analysis on β using CoOp. Figure 3 summarizes the results for controlling different levels of KD regularization on four datasets: EuroSAT, FGVC Aircraft, Flowers102, UCF101, as well as the average results across these datasets. Our OVE-PG consistently outperforms the baseline by large margins across all values of β, ranging from 0.1 to 0.7. Notably, within the mild range of β (i.e., 0.2–0.5), OVE-PG shows robustness regardless of the specific choice of β, which may alleviate the effort required for tuning this hyperparameter. Extended results can be found in Appendix.

6 Conclusion and Discussion

In this paper, we have proposed a Bayesian framework, coined as OVE-PG, that integrates Pólya-Gamma augmentation with a one-vs-each softmax approximation to mitigate overfitting in prompt learning for VLMs.

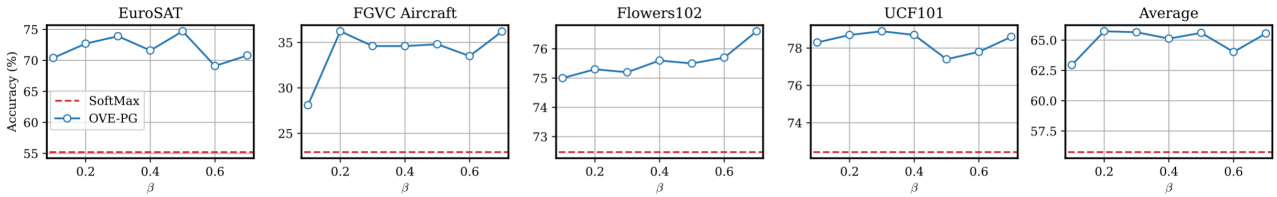

Figure 3: Sensitivity analysis of the hyperparameter β for controlling KLD regularization in OVE-PG (blue solid line), compared to the softmax baseline for CoOp (red dotted line). The results demonstrate robustness within a moderate range of β values (0.2–0.5) across all datasets.

OVE-PG outperforms existing methods by balancing task-specific learning with global knowledge retention. Notably, it does so without requiring additional regularization terms or network parameters, maintaining simplicity and compatibility with existing models. Our theoretical and experimental results demonstrate the robustness of this approach across various datasets. Additionally, we verify that our method is effective not only for VLMs but also in simple 1D examples and image classification with ViT. We believe that our approach provides a promising direction for improving prompt learning in VLMs, and hope that our framework can be expanded in a wide range of other domains.

Acknowledgements

M. Kim and M. Park were supported in part by the Natural Sciences and Engineering Research Council of Canada (NSERC) and the Canada CIFAR AI Chairs program at AMII (Alberta Machine Intelligence Institute). We also thank our anonymous reviewers for their constructive feedback, which has helped improve our manuscript.

References

Bossard, L., Guillaumin, M., and Gool, L. V. (2014a). Food-101 - mining discriminative components with random forests. In *European Conference on Computer Vision*.

Bossard, L., Guillaumin, M., and Gool, L. V. (2014b). Food-101–mining discriminative components with random forests. In *European Conference on Computer Vision*.

Chen, G., Yao, W., Song, X., Li, X., Rao, Y., and Zhang, K. (2023). PLOT: Prompt learning with optimal transport for vision-language models. In *The Eleventh International Conference on Learning Representations*.

Cho, Y., Bae, H., Shin, S., Youn, Y. D., Joo, W., and Moon, I.-C. (2024). Make prompts adaptable: Bayesian modeling for vision-language prompt learning with data-dependent prior. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 11552–11560.

Cimpoi, M., Maji, S., Kokkinos, I., Mohamed, S., and Vedaldi, A. (2014). Describing textures in the wild. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Derakhshani, M. M., Sanchez, E., Bulat, A., da Costa, V. G. T., Snoek, C. G., Tzimiropoulos, G., and Martinez, B. (2023a). Bayesian prompt learning for image-language model generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15237–15246.

Derakhshani, M. M., Sanchez, E., Bulat, A., da Costa, V. G. T., Snoek, C. G., Tzimiropoulos, G., and Martinez, B. (2023b). Bayesian prompt learning for image-language model generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15237–15246.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*.

Fei-Fei, L., Fergus, R., and Perona, P. (2004a). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *2004 Conference on Computer Vision and Pattern Recognition Workshop*, pages 178–178.

Fei-Fei, L., Fergus, R., and Perona, P. (2004b). Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. In *IEEE Conference on Computer Vision and Pattern Recognition - Workshops*.

Helber, P., Bischke, B., Dengel, A., and Borth, D. (2019). 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*.

Jia, C., Yang, Y., Xia, Y., Chen, Y.-T., Parekh, Z., Pham, H., Le, Q., Sung, Y.-H., Li, Z., and Duerig, T. (2021). Scaling up visual and vision-language representation learning with noisy text supervision. In *International conference on machine learning*, pages 4904–4916. PMLR.

Khattak, M. U., Rasheed, H., Maaz, M., Khan, S., and Khan, F. S. (2023a). Maple: Multi-modal prompt learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Khattak, M. U., Wasim, S. T., Naseer, M., Khan, S., Yang, M.-H., and Khan, F. S. (2023b). Self-regulating prompts: Foundational model adaptation without forgetting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15190–15200.

Krause, J., Stark, M., Deng, J., and Fei-Fei, L. (2013). 3D object representations for fine-grained categorization. In *IEEE Conference on Computer Vision and Pattern Recognition - Workshops*.

Lester, B., Al-Rfou, R., and Constant, N. (2021). The power of scale for parameter-efficient prompt tuning. In *Conference on Empirical Methods in Natural Language Processing*.

Linderman, S., Johnson, M. J., and Adams, R. P. (2015). Dependent multinomial models made easy: Stick-breaking with the polya-gamma augmentation. In Cortes, C., Lawrence, N., Lee, D., Sugiyama, M., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. (2023). Gpt understands, too. *AI Open*.

Lu, Y., Liu, J., Zhang, Y., Liu, Y., and Tian, X. (2022). Prompt distribution learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5206–5215.

Maji, S., Rahtu, E., Kannala, J., Blaschko, M., and Vedaldi, A. (2013). Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Nicholas G. Polson, J. G. S. and Windle, J. (2013). Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American Statistical Association*, 108(504):1339–1349.

Nilsback, M.-E. and Zisserman, A. (2008). Automated flower classification over a large number of classes. In *Indian Conference on Computer Vision, Graphics & Image Processing*.

Oh, C., Kim, M., Lim, H., Park, J., Jeong, E., Cheng, Z.-Q., and Song, K. (2023). Towards calibrated robust fine-tuning of vision-language models. *arXiv preprint arXiv:2311.01723*.

Parkhi, O. M., Vedaldi, A., Zisserman, A., and Jawahar, C. (2012). Cats and dogs. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. (2021). Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Snell, J. and Zemel, R. (2021). Bayesian few-shot classification with one-vs-each pólya-gamma augmented gaussian processes. In *International Conference on Learning Representations*.

Soomro, K., Zamir, A. R., and Shah, M. (2012a). A dataset of 101 human action classes from videos in the wild. *Center for Research in Computer Vision*.

Soomro, K., Zamir, A. R., and Shah, M. (2012b). Ucf101: A dataset of 101 human actions classes from videos in the wild. *ArXiv*, abs/1212.0402.

Sun, Q., Fang, Y., Wu, L., Wang, X., and Cao, Y. (2023). Eva-clip: Improved training techniques for clip at scale. *arXiv preprint arXiv:2303.15389*.

Titsias RC AUEB, M. (2016). One-vs-each approximation to softmax for scalable estimation of probabilities. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc.

Veličković, P., Perivolaropoulos, C., Barbero, F., and Pascanu, R. (2024). softmax is not enough (for sharp out-of-distribution).

Wu, C.-E., Tian, Y., Yu, H., Wang, H., Morgado, P., Hu, Y. H., and Yang, L. (2023). Why is prompt tuning for vision-language models robust to noisy labels? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15488–15497.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010a). Sun database: Large-scale scene recognition from abbey to zoo. *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492.

Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010b). Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Yang, Y., Ko, J., and Yun, S.-Y. (2024). Towards difficulty-agnostic efficient transfer learning for vision-language models. In Al-Onaizan, Y., Bansal,

M., and Chen, Y.-N., editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2066–2085, Miami, Florida, USA. Association for Computational Linguistics.

Yao, H., Zhang, R., and Xu, C. (2023). Visual-language prompt tuning with knowledge-guided context optimization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6757–6767.

Yu, J., Wang, Z., Vasudevan, V., Yeung, L., Seyedhosseini, M., and Wu, Y. (2022). Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*.

Zhai, X., Mustafa, B., Kolesnikov, A., and Beyer, L. (2023). Sigmoid loss for language image pre-training. *arXiv preprint arXiv:2303.15343*.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. (2022a). Conditional prompt learning for vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16816–16825.

Zhou, K., Yang, J., Loy, C. C., and Liu, Z. (2022b). Learning to prompt for vision-language models. *International Journal of Computer Vision*, 130(9):2337–2348.

Zhu, B., Niu, Y., Han, Y., Wu, Y., and Zhang, H. (2022). Prompt-aligned gradient for prompt tuning. *arXiv preprint arXiv:2205.14865*.

Supplementary Material: Bayesian Principles Improve Prompt Learning In VLMs

A One-vs-Each SoftMax Approximation

We provide a detailed proof of one-vs-each softmax approximation (Titsias RC AUEB, 2016). This serves as a lower bound to the softmax likelihood. In this section, we assume there are n data points and define $\boldsymbol{f} \in \mathbb{R}^{n \times \mathcal{C}}$ and $\boldsymbol{y} = \mathbb{R}^{n \times \mathcal{C}}$. For simplicity, we focus on a single data point, denoted as $\boldsymbol{y}_i = \{y_i^1, \cdots, y_i^{\mathcal{C}}\}$ and $\boldsymbol{f}_i = \{f_i^1, \cdots, f_i^{\mathcal{C}}\}$. Starting with the softmax likelihood, we have,

$$p(\boldsymbol{y}_i = c \mid \boldsymbol{f}_i) = \frac{e^{f_i^c}}{\sum_{c'} e^{f_i^{c'}}} = \frac{1}{1 + \sum_{c' \neq c} e^{-(f_i^c - f_i^{c'})}} \tag{A.1}$$

To derive the OVE approximation, we manipulate the denominator by separating the term e^{f_i} from the sum as follows.

$$\sum_{c=1}^{\mathcal{C}} e^{f_i^c} = e^{f_i^c} + \sum_{c' \neq c} e^{f_i^{c'}} \tag{A.2}$$

Dividing both the numerator and the denominator by e^{f_i}, the softmax probability becomes,

$$\begin{aligned} p(\boldsymbol{y}_i = c \mid \boldsymbol{f}_i) &= \frac{e^{f_i^c}}{e^{f_i^c} + \sum_{c' \neq c} e^{f_i^{c'}}} \\ &= \frac{1}{1 + \sum_{c' \neq c} e^{f_i^{c'} - f_i^c}} \\ &= \frac{1}{1 + \sum_{c' \neq c} e^{-(f_i^c - f_i^{c'})}} \end{aligned} \tag{A.3}$$

This expression highlights the dependence on the differences $f_i^c - f_i^{c'}$. To establish a lower bound, we utilize the following general inequality, which holds for any set of non-negative real numbers $\{\alpha_k\}$,

$$1 + \sum_k \alpha_k \leq \prod_k (1 + \alpha_k) \quad \text{such that} \quad \alpha_k \geq 0 \tag{A.4}$$

Applying this inequality to the denominator in Equation A.3, with $\alpha_{c'} = e^{-(f_i^c - f_i^{c'})} \geq 0$, we obtain

$$1 + \sum_{c' \neq c} e^{-(f_i^c - f_i^{c'})} \leq \prod_{c' \neq c} \left(1 + e^{-(f_i^c - f_i^{c'})}\right) \tag{A.5}$$

Since all terms are positive, by taking reciprocals on both sides, we have

$$\frac{1}{1 + \sum_{c' \neq c} e^{-(f_i^c - f_i^{c'})}} \geq \frac{1}{\prod_{c' \neq c} \left(1 + e^{-(f_i^c - f_i^{c'})}\right)} \tag{A.6}$$

Recognizing that $1/1+e^{-x}$ is the sigmoid function $\sigma(x)$, we can express each term in the product as:

$$\frac{1}{1+e^{-(f_i^c - f_i^{c'})}} = \sigma(f_i^c - f_i^{c'}) \tag{A.7}$$

Therefore, the product in the denominator becomes,

$$\prod_{c' \neq c} \frac{1}{1+e^{-(f_i^c - f_i^{c'})}} = \prod_{c' \neq c} \sigma(f_i^c - f_i^{c'}) \tag{A.8}$$

Combining Equation A.6 and Equation A.8, we establish the following inequality,

$$p(\boldsymbol{y}_i = c \mid \boldsymbol{f}_i) \geq \prod_{c' \neq c} \sigma(f_i^c - f_i^{c'}) \tag{A.9}$$

B Polya-Gamma Augmentation for Sigmoid Function

Nicholas G. Polson and Windle (2013) introduced the Polya-Gamma augmentation for the sigmoid function to facilitate Bayesian inference in logistic linear regression for binary classification. While previous methods approximate the likelihood function using Gaussian distributions, this approach enables a natural derivation of the likelihood as a Gaussian distribution by introducing augmented variables drawn from the Polya-Gamma distribution. The detailed procedure is provided as follows.

B.1 Sigmoid Function

We denote input $\boldsymbol{x}_i \in \mathbb{R}^D$ and $y_i \in \{0, 1\}$. The number of data points is N. The sigmoid function is denoted by $\sigma(f_i) = \frac{1}{1+\exp(-f_i)}$ and $f_i = \boldsymbol{\beta}^T \boldsymbol{x}_i$. We assume a linear model parameterized by $\boldsymbol{\beta}$.

$$\begin{aligned}
p(\mathbf{y}|f) &= \prod_{i=1}^{N} \sigma(f_i)^{y_i} (1 - \sigma(f_i))^{1-y_i} \\
&= \prod_{i=1}^{N} \left(\frac{1}{1+e^{-f_i}} \right)^{y_i} \left(\frac{e^{-f_i}}{1+e^{-f_i}} \right)^{1-y_i} \\
&= \prod_{i=1}^{N} \frac{1}{(1+e^{-f_i})^{y_i}} \frac{e^{-f_i(1-y_i)}}{(1+e^{-f_i})^{1-y_i}} \\
&= \prod_{i=1}^{N} \frac{e^{-f_i(1-y_i)}}{1+e^{-f_i}} = \prod_{i=1}^{N} \frac{(e^{-(\boldsymbol{\beta}^T \boldsymbol{x}_i)})^{(1-y_i)}}{1+e^{-\boldsymbol{\beta}^T \boldsymbol{x}_i}}
\end{aligned} \tag{B.10}$$

B.2 Understanding $\cosh(x)$

To demonstrate the relationship between $\cosh(x)$ and $\prod_{k=1}^{\infty} \left(1 + \frac{t}{2\pi^2(k-\frac{1}{2})^2}\right)$, we first explore the structure of infinite product representation of $\sin(\pi x)$ and $\cos(\pi x)$. A key step in this process is to examine the zeros of hypoerbolic cosine function $\cosh(x) = 0$. We begin by considering the following equality:

$$\cosh(x) = 0 \rightarrow \frac{e^x + e^{-x}}{2} = 0 \rightarrow e^{2x} + 1 = 0 \quad \rightarrow e^{-2x} = -1 \tag{B.11}$$

The solution of Equation B.11 lies in the imagenary domains because of $e^{-2x} = -1$. To explore this, we substitute $x = iy$, where y is real-value. Under this substitution, $\cosh(iy) = \frac{e^{iy}+e^{-iy}}{2} = \cos(y)$, therefore $\cosh(x) = 0$ when $\cos(y) = 0$. The zeros of cosin function occur at $y = (n + \frac{1}{2})\pi$ for $n \in \mathbb{Z}$ because $\cos(\frac{1}{2}\pi) = 0, \cos(\frac{3}{2}\pi) = 0, \cdots$. So, we rewrite the zeros of $\cosh(x)$ as $x = i\pi(n + \frac{1}{2})$ for $n \in \mathbb{Z}$.

Next, we consider the infinite product representation of $\cosh(x)$. According to the definition of entrie function and Weierstrass factorization theorem, any entire function $f(z)$ can be represented as an infinite product of its zeros z_k. In details, it is described as follows.

$$f(z) = e^{g(z)} \prod_{n=1}^{\infty} \left(1 - \frac{z}{z_n}\right) e^{\frac{z}{z_n} + \frac{1}{2}\left(\frac{z}{z_n}\right)^2 + \cdots + \frac{1}{m_n}\left(\frac{z}{z_n}\right)^{m_n}} \tag{B.12}$$

By entire function of order 1 and symetric property of $\cosh(x)$ about the origin (zero), the product simplifies and the exponential factors cancel out. We can write $\cosh(x) = C\prod_{n=1}^{\infty}\left(1 - \frac{x^2}{x_n^2}\right)$ where, C is constant. To simplify the infinite product, we substitute $x_n = i\pi(n + \frac{1}{2})$ into $x_n^2 = -\left(n + \frac{1}{2}\right)^2\pi^2$. As a result, we have $1 - \frac{x^2}{x_n^2} = 1 - \frac{x^2}{-\left(n+\frac{1}{2}\right)^2\pi^2} = 1 + \frac{x^2}{\left(n+\frac{1}{2}\right)^2\pi^2}$. Therefore, the product becomes $\cosh(x) = C\prod_{n=0}^{\infty}\left(1 + \frac{x^2}{\pi^2\left(n-\frac{1}{2}\right)^2}\right)$. In this context, assuming $C = 1$ simplifies the expression further:

$$\cosh(x) = \prod_{k=1}^{\infty}\left(1 + \frac{x^2}{\pi^2(k - \frac{1}{2})}\right) \tag{B.13}$$

B.3 Laplace Transformation and Moment Generating Function

The Laplace transformation $\mathcal{L}$ is defined as follows:

$$\mathcal{L}\{f(\omega)\}(t) = \int_0^{\infty} e^{-\omega t} f(\omega) d\omega \tag{B.14}$$

Suppose that the moment generating function (MGF) of x is given by $\mathcal{M}_x(t) = \mathbb{E}\left[e^{tx}\right]$. For instance, the MGF of Bernoulli distribution is given by $M_x(t) = 1 - p + pe^t$. Assuming that $f(\omega)$ is a probability density function, the Laplace transformation of $f(\omega)$ is the MGF of x.

$$\mathcal{L}\{f(\omega)\}(t) = M_{\omega}(-t) = \mathbb{E}\left[e^{-t\omega}\right] \tag{B.15}$$

For instance, the moment generating function of gamma distribution with the shape parameter b and scale parameter 1 is known as $\mathcal{M}_{\gamma}(t) = \mathbb{E}_{\gamma}\left[e^{t\gamma}\right] = (1-t)^{-b}$ and its laplace transformation is given by $\mathcal{L}_{p(\gamma)}(t) = (1+t)^{-b}$.

Here is the derivation of the MGF of gamma distribution with shape parameter b and scale parameter 1. By definition of MGF and gamma distribution, we have:

$$M_x(t) = \int_0^{\infty} e^{tx}\frac{x^{b-1}e^{-x}}{\Gamma(b)}\,dx = \frac{1}{\Gamma(b)}\int_0^{\infty} x^{b-1}e^{-(1-t)x}\,dx \tag{B.16}$$

Note that the interal in Equation B.16 converges only if $(1-t) > 0$, i.e., $t < 1$. Next, recall the definition of the gamma function $\Gamma(\alpha) = \int_0^{\infty} x^{\alpha-1}e^{-x}dx$, our integral in Equation B.16 has a similar form but with $e^{-(1-t)x}$ instead of e^{-x}. To align it with the Gamma function, we introduce the substitution $\lambda = 1 - t$ for Equation B.16, where $\lambda > 0$ when $t < 1$. Then, we have $M_x(t) = \frac{1}{\Gamma(b)}\int_0^{\infty} x^{b-1}e^{-\lambda x}dx$. This integral is the definition of the Gamma function scaled by λ:

$$\int_0^{\infty} x^{b-1}e^{-\lambda x}\,dx = \frac{\Gamma(b)}{\lambda^b} \tag{B.17}$$

Finally, we organize the results and substitute back to the MGF. Consider $\int_0^{\infty} x^{b-1}e^{-\lambda x}\,dx = \lambda^{-b}\int_0^{\infty}(\lambda x)^{b-1}e^{-\lambda x}\,d(\lambda x)\cdot\frac{1}{\lambda}$, To write this in a form of gamma distribution, we have substitution $d(\lambda x) = \lambda dx$ then we obtain:

$$\lambda^{-b}\int_0^{\infty} y^{b-1}e^{-y}dy = \lambda^{-b}\Gamma(b) \tag{B.18}$$

Therefore,

$$\int_0^{\infty} x^{b-1}e^{-\lambda x}dx = \frac{\Gamma(b)}{\lambda^b} \tag{B.19}$$

Now substitute back:

$$\begin{aligned} M_x(t) &= \frac{1}{\Gamma(b)} \cdot \frac{\Gamma(b)}{(1-t)^b} = \frac{1}{(1-t)^b} \quad t < 1 \\ \mathcal{L}\{p(\gamma)\}(t) &= (1+t)^{-b} \end{aligned} \tag{B.20}$$

B.4 Understanding Polya-Gamma distribution

The Polya-Gamma distribution is a class of distributions that are used to sample from the posterior distribution of the logistic regression model. Specifically, the random variables w drawn from the Polya-Gamma distribution is equivalent to the infinite sum of random variables drawn from gamma distribution with shape parameter b and scale parameter 1. In more detail, the Polya-Gamma distribution is defined as follows:

$$\begin{aligned} w \sim \mathrm{PG}(b,c) \leftarrow w &= \frac{1}{2\pi^2} \sum_{k=1}^{\infty} \frac{\gamma_k}{\left(k-\frac{1}{2}\right)^2 + \left(\frac{c}{4\pi^2}\right)^2} \\ \text{where} \quad \gamma_k \sim \mathrm{Gamma}(b,1), &\quad p(\gamma_k) = \frac{1}{\Gamma(b)} \gamma^{b-1} e^{-\gamma} \end{aligned} \tag{B.21}$$

We consider the laplace transform of $\omega \sim$ polya-gamma distribution (b, 0), we have:

$$\begin{aligned} \mathbb{E}\left[e^{-t\omega}\right] &= \mathbb{E}\left[e^{-t\frac{1}{2\pi^2}\sum_{k=1}^{\infty}\frac{\gamma_k}{\left(k-\frac{1}{2}\right)^2}}\right] \\ &= \prod_{k=1}^{\infty} \mathbb{E}\left[e^{\frac{-t}{2\pi^2}\frac{\gamma_k}{\left(k-\frac{1}{2}\right)^2}}\right] \end{aligned} \tag{B.22}$$

where, $\gamma_k \sim \mathrm{Gamma}(b,1)$. We can simplify the expectation of the exponential term as follows $\mathbb{E}\left[e^{\frac{-t}{2\pi^2}\frac{\gamma_k}{\left(k-\frac{1}{2}\right)^2}}\right] = (1+\frac{t}{2\pi^2}\frac{1}{\left(k-\frac{1}{2}\right)^2})^{-b}$ by $\mathbb{E}\left[e^{-t\gamma_k}\right] = (1+t)^{-b}$. By organizing the results, we substitute back to the Laplace transform of Polya-Gamma distribution:

$$\mathbb{E}\left[e^{-t\omega}\right] = \prod_{k=1}^{\infty}\left(1 + \frac{t}{2\pi^2\left(k-\frac{1}{2}\right)^2}\right)^{-b} \tag{B.23}$$

Next, we consider the $\cosh(x)$ from Equation B.13. It is directly related to the Laplace transform of Polya-Gamma distribution. Specifically, the equation $\cosh(\frac{\sqrt{t}}{2}) = \prod_{k=1}^{\infty}\left(1 + \frac{t}{2\pi^2\left(k-\frac{1}{2}\right)^2}\right)$ allows to rewrite the equation as follows.

$$\mathbb{E}\left[e^{-t\omega}\right] = \left(\cosh(\frac{\sqrt{t}}{2})\right)^{-b} \tag{B.24}$$

Hence, we obseve that the product form of $\cosh(x)$ leads to the expression of the Laplace transform of Polya-Gamma distribution.

B.5 Revisiting Sigmoid Function

We consider the general form of the sigmoid function Equation B.10 as $\frac{\left(e^f\right)^\alpha}{\left(1+e^f\right)^b}$. First, we dervie $\left(1+e^f\right)^{-b} = 2e^{\frac{f}{2}}\cosh\left(\frac{f}{2}\right)^{-b}$. We know $\cosh\left(\frac{x}{2}\right) = \frac{e^{x/2}-e^{-x/2}}{2}$ by Equation B.11. $(1+e^f) = e^{f/2}\left(e^{f/2}+e^{-f/2}\right)$ allows to rewrite as follow:

$$\begin{aligned} \left(1+e^f\right) &= e^{f/2}\left(e^{f/2}+e^{-f/2}\right) = 2e^{f/2}\cosh\left(\frac{f}{2}\right) \\ \left(1+e^f\right)^{-b} &= 2^{-b}e^{-bf/2}\left(\cosh\left(\frac{f}{2}\right)\right)^{-b} \end{aligned} \tag{B.25}$$

We observe that $\cosh\left(\frac{\sqrt{t}}{2}\right)$ is equivalent to MGF of Polya-Gamma distribution $\mathbb{E}_\omega\left[e^{-t\omega}\right]$. By substituting $t = \phi^2/2$, we can write the following relationship:

$$\begin{aligned}\left(\cosh\left(\frac{\sqrt{t}}{2}\right)\right)^{-b} &= \int_0^\infty e^{-t\omega}p(\omega)d\omega \quad \text{where,} \quad p(\omega) \sim (PG)(b,0)\\ \left(\cosh\left(\frac{\phi}{2}\right)\right)^{-b} &= \int_0^\infty e^{-\frac{\phi^2}{2}\cdot\omega}p(\omega)d\omega\end{aligned} \tag{B.26}$$

This equation $(1+e^f)^{-b} = 2^{-b}e^{-bf/2}\int_0^\infty e^{-\phi^2/2\cdot\omega}p(\omega)d\omega$ allows us to rewrite the sigmoid function in more general way. Specifically, we have

$$\begin{aligned}\frac{(e^f)^\alpha}{(1+e^f)^b} &= 2^{-b}e^{f\alpha}e^{-bf/2}\int_0^\infty e^{-\phi^2/2\cdot\omega}p(\omega)d\omega\\ \frac{(e^f)^\alpha}{(1+e^f)^b} &= 2^{-b}e^{f(\alpha-b/2)}\int_0^\infty e^{-\phi^2/2\cdot\omega}p(\omega)d\omega\end{aligned} \tag{B.27}$$

We substitute $\kappa = \alpha - b/2$ and allow as follows.

$$\frac{(e^f)^\alpha}{(1+e^f)^b} = 2^{-b}e^{f\kappa}\int_0^\infty e^{-\phi^2/2\cdot\omega}p(\omega)d\omega \tag{B.28}$$

Suppose to have $\alpha_i = y_i, b = 1, \kappa_i = y_i - 1/2$ and $f_i = \boldsymbol{\beta}^T\boldsymbol{x}_i$, we can rewrite the sigmoid function $\sigma(f) = \frac{1}{1+e^{-f_i}}$ as follows:

$$\begin{aligned}p(y_i|f_i) = \sigma(f_i)^{y_i}(1-\sigma(f_i))^{1-y_i} &= \left(\frac{e^{f_i}}{1+e^{f_i}}\right)^{y_i}\left(\frac{1}{1+e^{f_i}}\right)^{1-y_i}\\ = \frac{e^{f_i{}^{y_i}}}{1+e^{f_i}} &= \frac{1}{2}e^{\kappa_i f_i}\int_0^\infty e^{-\frac{\omega_i f_i^2}{2}}p(\omega_i)d\omega_i\end{aligned} \tag{B.29}$$

In this context, we view the likelihood $p(y_i|f_i) = \frac{e^{f_i{}^{y_i}}}{1+e^{f_i}} = \frac{1}{2}e^{\kappa_i f_i}\int_0^\infty e^{-\frac{\omega_i f_i^2}{2}}p(\omega_i)d\omega_i$ as an integral over the polya-gamma distribution, where we interpret this expression as $\int p(y_i,\omega_i|f_i)d\omega_i$. By this interpretation, the joint distribution $p(y_i,\omega_i|f_i)$ enables us to find the conditional distributions $p(y_i|\omega_i,f_i) = \frac{p(y_i,\omega_i|f_i)}{p(\omega_i|f_i)} = \frac{p(y_i,\omega_i|f_i)}{p(\omega_i)} \propto p(y_i,\omega_i|f_i)$.

When we consider the relationship $p(y_i|w_i,f_i) \propto \frac{e^{f_i{}^{y_i}}}{1+e^{f_i}} = \frac{1}{2}e^{\kappa_i f_i}\int_0^\infty e^{-\frac{\omega_i f_i^2}{2}}p(\omega_i)d\omega_i$, it implies that $p(y_i|\omega_i,f_i)$ is a gaussian distribution over f. Given that ω_i is drawn from polya-gamma distribution as shown in Equation B.21, the conditional distribution of y_i given f_i and ω_i simplifies to $p(y_i|f_i,\omega_i) \propto e^{-\frac{\omega_i f_i{}^2}{2}}e^{\kappa_i f_i}$. The detail derivation proceeds as follows.

$$\begin{aligned}p(y_i \mid f_i,\omega_i) &\propto e^{-\frac{\omega_i f_i{}^2}{2}}e^{\kappa_i f_i} = e^{-\frac{\omega_i f_i{}^2}{2}+\kappa_i f_i}\\ &= e^{-\frac{\omega_i}{2}\left(f_i^2 - \frac{2\kappa_i}{\omega_i}f_i + \frac{\kappa_i^2}{\omega_i^2} - \frac{\kappa_i^2}{\omega_i^2}\right)}\\ &= e^{-\frac{\omega_i}{2}\left(f_i - \frac{\kappa_i}{\omega_i}\right)^2 + \frac{\omega_i}{2}\frac{\kappa_i^2}{\omega_i^2}}\\ &= e^{-\frac{\omega_i}{2}\left(f_i - \frac{\kappa_i}{\omega_i}\right)^2}e^{\frac{\kappa_i^2}{2\omega_i}}\\ p(y_i \mid f_i,\omega_i) &\propto e^{-\frac{\omega_i}{2}\left(f_i - \frac{\kappa_i}{\omega_i}\right)^2}\end{aligned} \tag{B.30}$$

It implies that the conditional distribution of y_i given f_i and ω_i is a gaussian distribution $\mathcal{N}(\frac{\kappa_i}{\omega_i}; f_i, \omega_i^{-1})$. For convenience, let us introduce a new variable z_i such that $\frac{\kappa_i}{\omega_i}$ and it is related to the label y_i. The multivariate cases describe the conditional distribution $\mathcal{N}(\boldsymbol{\Omega}^{-1}\boldsymbol{\kappa}_i; \mathbf{f}_i, \boldsymbol{\Omega}^{-1})$ where $\boldsymbol{\kappa}_i, \boldsymbol{f}_i \in \mathbb{R}^{\mathcal{C}}$ and $\boldsymbol{\Omega} = \omega_i\boldsymbol{I}$.

B.6 Construction of Conjugate Relationship for Posterior Distribution f_i and ω_i.

Prior to derviation of posterior distribution, we cover the construction of conjugate relationship for gaussian distribution.

Definition 1. *Given a margnial gaussian distribution of* $\boldsymbol{b}$ *and a conditional gaussian distribution for* $\boldsymbol{a}$ *given* $\boldsymbol{b}$*, We define these gaussian distributions as follows:*

$$\begin{aligned} p(\boldsymbol{b}) &= \mathcal{N}(\boldsymbol{b}; \boldsymbol{m}, \boldsymbol{S}^{-1}) \\ p(\boldsymbol{a} \mid \boldsymbol{b}) &= \mathcal{N}(\boldsymbol{a}; \boldsymbol{W}\boldsymbol{b}, \boldsymbol{\Phi}^{-1}) \end{aligned} \tag{B.31}$$

Lemma 1. *The marginal distribution of* $\boldsymbol{a}$ *and the conditional distribution of* $\boldsymbol{b}$ *given* $\boldsymbol{a}$ *are given by*

$$\begin{aligned} p(\boldsymbol{b} \mid \boldsymbol{a}) &= \mathcal{N}(\boldsymbol{b}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ p(\boldsymbol{a}) &= \mathcal{N}(\boldsymbol{a}; \boldsymbol{W}\boldsymbol{m}, \boldsymbol{\Phi}^{-1} + \boldsymbol{W}\boldsymbol{S}^{-1}\boldsymbol{W}^T) \end{aligned} \tag{B.32}$$

where $\boldsymbol{\Sigma} = (\boldsymbol{S} + \boldsymbol{W}^T\boldsymbol{\Phi}\boldsymbol{W})^{-1}$ *and* $\boldsymbol{\mu} = \boldsymbol{\Sigma}(\boldsymbol{W}^T\boldsymbol{\Phi}\boldsymbol{a} + \boldsymbol{S}\boldsymbol{m})$.

B.6.1 Posterior Distribution f_i

Suppose that $f_i = \boldsymbol{\beta}^T\boldsymbol{x}_i$ in Equation B.10 is parameterized by $\boldsymbol{\beta}$, we can define the prior and posterior distribution for $\boldsymbol{\beta}$ instead of f_i. In this case, we consider the prior distribution $p(\boldsymbol{\beta})$ as $\boldsymbol{\beta} \sim \mathcal{N}(\boldsymbol{\beta}; \boldsymbol{m}, \boldsymbol{S}^{-1})$. The posterior distribution $p(\boldsymbol{\beta}|y_i, \omega_i)$ can be defined by Equation B.32 with $p(y_i, |f_i, \omega_i)$ and $p(\boldsymbol{\beta})$. It can be described as $p(\boldsymbol{\beta}|y_i, \omega_i) = \mathcal{N}(\boldsymbol{\beta}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\boldsymbol{\Sigma} = (\mathbf{S} + \boldsymbol{\Omega})^{-1}$ and $\boldsymbol{\mu} = \boldsymbol{\Sigma}(\boldsymbol{\kappa} + \mathbf{S}\mathbf{m})$.

$$\begin{aligned} p(f_i \mid y_i, \omega_i) &= p(\boldsymbol{\beta} \mid y_i, \omega_i) = \mathcal{N}(\boldsymbol{\beta}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \\ \text{where} \quad \boldsymbol{\mu} &= \boldsymbol{\Sigma}(\boldsymbol{\kappa} + \mathbf{S}\mathbf{m}) \\ \boldsymbol{\Sigma} &= (\mathbf{S} + \boldsymbol{\Omega})^{-1} \end{aligned} \tag{B.33}$$

Next, we derive the posterior distribution of ω_i given y_i and f_i. We start by considering the joint distribution $p(y_i, \omega_i|f_i)$. We substitue $p(\omega_i|y_i, f_i)$ with $\frac{p(y_i,\omega_i|f_i)}{p(y_i|f_i)} = \frac{p(y_i|\omega_i,f_i)p(\omega_i|f_i)}{p(y_i|f_i)} = \frac{p(y_i|\omega_i,f_i)p(\omega_i)}{p(y_i|f_i)}$ due to the independence of f_i and ω_i. Accordingly, we have $p(\omega_i|y_i, f_i) \propto p(y_i|\omega_i, f_i)p(\omega_i)$. By Equation B.30, $p(y_i|\omega_i, f_i) = e^{-\frac{\omega_i {f_i}^2}{2}} e^{\kappa_i f_i}$ and $p(\omega_i) = p(\omega_i) \sim \mathrm{PG}(1, 0)$ by definition.

C Detailed Illustration of Proposed Method

The procedure of the proposed method is outlined through its mathematical framework, beginning with tensor operations related to A and f. The next part explains the One-vs-Each SoftMax approximation. The final subsection describes the conjugacy of the One-vs-Each SoftMax approximation. Each component of the sigmoid function in this approximation shows conjugacy with Polya-Gamma variables, which are obtained from pre-trained models. Additionally, the label information from the training data helps define the posterior distribution, allowing the fine-tuned model to learn as expected.

C.1 Einstein Notation and Tensor Operations

We define the tensor A using Kronecker delta functions. Here, $A \in \mathbb{R}^{C\times C\times C}$ captures the differences between identity matrices along specified dimensions.

$$A_{ijk} = \delta_{ik} - \delta_{jk}, \quad \text{for } i, j, k = 1, \ldots, C \tag{C.34}$$

Next, we consider a matrix f of size $N \times C$. Here, N represents the number of samples, and C is the number of classes.

$$f \in \mathbb{R}^{N\times C} = \begin{bmatrix} f_{11} & \cdots & f_{1C} \\ f_{21} & \cdots & f_{2C} \\ \vdots & \ddots & \vdots \\ f_{N1} & \cdots & f_{NC} \end{bmatrix} \tag{C.35}$$

We aim to compute the tensor product Af, defined as:

$$\psi_{nij} = (Af)_{nij} = f_{ni} - f_{nj}, \quad \text{for} \quad n = 1, \ldots, N;\ i, j = 1, \ldots, C \tag{C.36}$$

This operation results in a tensor $Af \in \mathbb{R}^{N \times C \times C}$, where each element represents the difference between elements of f across different classes for each sample. To illustrate this explicitly, we can write out the tensor Af for each sample n:

$$\psi = Af = \left[\begin{bmatrix} 0 & f_{n1} - f_{n2} & \cdots & f_{n1} - f_{nC} \\ f_{n2} - f_{n1} & 0 & \cdots & f_{n2} - f_{nC} \\ \vdots & \vdots & \ddots & \vdots \\ f_{nC} - f_{n1} & f_{nC} - f_{n2} & \cdots & 0 \end{bmatrix} \Bigg|_{n=1}^{N} \right] \tag{C.37}$$

This explicit form shows that for each sample n, the tensor Af contains all pairwise differences of the elements of f for that sample. By representing the operation in this way, we clarify how the tensor product operates between A and f. This formulation can be efficiently executed using tensor manipulation functions such as `torch.einsum` in PyTorch, which is well-suited for handling complex tensor operations. To further illustrate this, the following subsection provides a straightforward example involving three-class classification, showcasing the practical application of this concept.

C.2 Simple Example: One-vs-Each SoftMax Approximation on Three-Class Classification

Let us now provide the tensor product based on Einstein summation convention $\boldsymbol{Af}$ in a practical implementation. Consider a classification task with three classes, where $C = 3$. The tensor $A \in \mathbb{R}^{3 \times 3 \times 3}$ is defined as

$$A \in \mathbb{R}^{C \times C \times C} = \left[\begin{bmatrix} 0 & 0 & 0 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 1 \\ 0 & -1 & 1 \\ 0 & 0 & 0 \end{bmatrix} \right] \tag{C.38}$$

The original logit matrix $\boldsymbol{f} \in \mathbb{R}^{N \times 3}$ contains the logit values for N samples and 3 classes

$$\boldsymbol{f} \in \mathbb{R}^{N \times C} = \begin{bmatrix} \boldsymbol{f}_1 \\ \boldsymbol{f}_2 \\ \vdots \\ \boldsymbol{f}_N \end{bmatrix} = \begin{bmatrix} f_{11} & \cdots & f_{1C} \\ f_{21} & \cdots & f_{2C} \\ \vdots & \ddots & \vdots \\ f_{N1} & \cdots & f_{NC} \end{bmatrix} \tag{C.39}$$

The Einstein summation notation $(\boldsymbol{Af})_{nic} = \sum_j \boldsymbol{A}_{ijc} \boldsymbol{f}_{ni}$, denoted by $\boldsymbol{\psi}$, computes the pairwise differences between the feature values across classes for each sample

$$\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f} = \begin{bmatrix} \begin{bmatrix} 0 & f_{11}-f_{12} & f_{11}-f_{13} \\ f_{12}-f_{11} & 0 & f_{12}-f_{13} \\ f_{13}-f_{11} & f_{13}-f_{12} & 0 \end{bmatrix} & (n=1) \\ \begin{bmatrix} 0 & f_{21}-f_{22} & f_{21}-f_{23} \\ f_{22}-f_{21} & 0 & f_{22}-f_{23} \\ f_{23}-f_{21} & f_{23}-f_{22} & 0 \end{bmatrix} & (n=2) \\ \vdots & \\ \begin{bmatrix} 0 & f_{n1}-f_{n2} & f_{n1}-f_{n3} \\ f_{n2}-f_{n1} & 0 & f_{n2}-f_{n3} \\ f_{n3}-f_{n1} & f_{n3}-f_{n2} & 0 \end{bmatrix} & (n=N) \end{bmatrix} \tag{C.40}$$

This example makes it clear how the tensor operations introduced in the previous subsection apply to the implementation scenario. Each matrix ψ_n for sample n contains the pairwise differences between the class logit values. Next, we compute the One-vs-Each SoftMax approximation using a sigmoid function $\sigma(\psi_{nij})$ and apply it for classification.

$$\begin{aligned} \prod_{i=1}^{3} \sigma\left(\psi_{nic}\right) &= \prod_{j=1}^{3} \sigma\left(\left(\boldsymbol{A}\boldsymbol{f}\right)_{nic}\right) \in \mathbb{R}^{n\times C} \\ &= \begin{pmatrix} \sigma(0)\sigma(\psi_{112})\sigma(\psi_{113}) & \sigma(\psi_{121})\sigma(0)\sigma(\psi_{123}) & \sigma(\psi_{131})\sigma(\psi_{132})\sigma(0) \\ \sigma(0)\sigma(\psi_{212})\sigma(\psi_{213}) & \sigma(\psi_{221})\sigma(0)\sigma(\psi_{223}) & \sigma(\psi_{231})\sigma(\psi_{232})\sigma(0) \\ \vdots & \vdots & \vdots \\ \sigma(0)\sigma(\psi_{n12})\sigma(\psi_{n13}) & \sigma(\psi_{n21})\sigma(0)\sigma(\psi_{n23}) & \sigma(\psi_{n31})\sigma(\psi_{n32})\sigma(0) \end{pmatrix} \\ &= \begin{pmatrix} \sigma(0)\sigma(f_{11}-f_{12})\sigma(f_{11}-f_{13}) & \sigma(f_{12}-f_{11})\sigma(0)\sigma(f_{12}-f_{13}) & \sigma(f_{13}-f_{11})\sigma(f_{13}-f_{12})\sigma(0) \\ \sigma(0)\sigma(f_{21}-f_{22})\sigma(f_{21}-f_{23}) & \sigma(f_{22}-f_{21})\sigma(0)\sigma(f_{22}-f_{23}) & \sigma(f_{23}-f_{21})\sigma(f_{23}-f_{22})\sigma(0) \\ \vdots & \vdots & \vdots \\ \sigma(0)\sigma(f_{n1}-f_{n2})\sigma(f_{n1}-f_{n3}) & \sigma(f_{n2}-f_{n1})\sigma(0)\sigma(f_{n2}-f_{n3}) & \sigma(f_{n3}-f_{n1})\sigma(f_{n3}-f_{n2})\sigma(0) \end{pmatrix} \end{aligned} \tag{C.41}$$

The final output is denoted as $\boldsymbol{f}_\theta^m$ in Algorithm 1, which is used in the loss function

$$\begin{aligned} \boldsymbol{f}_\theta^m &= \prod_{i=1}^{3} \sigma\left(\psi_{nic}\right) \in \mathbb{R}^{n\times C} \\ &= \begin{pmatrix} \sigma(0)\sigma(\psi_{112})\sigma(\psi_{113}) & \sigma(\psi_{121})\sigma(0)\sigma(\psi_{123}) & \sigma(\psi_{131})\sigma(\psi_{132})\sigma(0) \\ \sigma(0)\sigma(\psi_{212})\sigma(\psi_{213}) & \sigma(\psi_{221})\sigma(0)\sigma(\psi_{223}) & \sigma(\psi_{231})\sigma(\psi_{232})\sigma(0) \\ \vdots & \vdots & \vdots \\ \sigma(0)\sigma(\psi_{n12})\sigma(\psi_{n13}) & \sigma(\psi_{n21})\sigma(0)\sigma(\psi_{n23}) & \sigma(\psi_{n31})\sigma(\psi_{n32})\sigma(0) \end{pmatrix} \end{aligned} \tag{C.42}$$

In this example, we use the negative log-likelihood loss $\mathcal{L}_{\text{nll}}^m = \sum_n \sum_c y_n^c \log\left(f_{\theta}{}_n^c\right)^m$ for classification. This example serves as a practical illustration of how the tensor operations introduced in the previous subsection can be used in implementation.

C.3 One-vs-Each SoftMax Approximation with Polya-Gamma Augmentation

Following the previous sections Appendix B and subsection C.2, we now extend the One-vs-Each SoftMax approximation by introducing Polya-Gamma augmentation. Prior to explanation of this subsection, we implemente sampling procedure of posterior distribution in the ψ space rather than $\boldsymbol{f}$ space, because the matrix A must be

considered when working in the $\boldsymbol{f}$ space shown in the main manuscript. For computational efficiency, sampling on the space of ψ can minimize the computation burden based on our assumption. It makes the proposed method more feasible in terms of implementation. To maintain consistency with notation used in Algorithm 1, we substitute f from the previous subsection with μ. Specifically, $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}, \frac{1}{\alpha}\mathbf{I})$ and $\boldsymbol{f}_\theta \sim \mathcal{N}(\boldsymbol{\mu}_\theta, \frac{1}{\alpha}\mathbf{I})$.

This extension demonstrates that each sigmoid function in the tensor $\boldsymbol{\psi}_\theta = \boldsymbol{A}\boldsymbol{\mu}_\theta \in \mathbb{R}^{n\times C\times C}$ (defined in Equation C.40) has its own conjugacy properties. Specifically, $\boldsymbol{\psi}_\theta \sim \mathcal{N}(\boldsymbol{A}\boldsymbol{\mu}_\theta, {}^2\!/_\alpha\mathbf{I})$. This arises because A represents the subtract between f_i and f_j and since these terms are independent, the variances for ψ_θ are effectively doubled, leading to the simplified covaraince structure. First, we define the conjugacy of the Polya-Gamma variables. The variable $\boldsymbol{\kappa} \in \mathbb{R}^{n\times C\times C}$ is derived from the one-hot encoded label information $\boldsymbol{y} \in \mathbb{R}^{n\times C}$, along with the tensor A described in subsection C.1. These variables form the basis of the posterior distribution. The Polya-Gamma variable $\boldsymbol{\omega} \in \mathbb{R}^{n\times C\times C}$ is sampled using $\boldsymbol{\psi}$ constructed by the pre-trained model $\boldsymbol{\mu}$ and tensor A. More Specifically, $\boldsymbol{\psi}_\theta$ represents the fine-tuned model, while $\boldsymbol{\psi}$ denotes the pre-trained model, both utilizing the same A. We assume that each dimension of $\boldsymbol{\psi}$, $\boldsymbol{\kappa}$, and $\boldsymbol{\omega}$ is independent, which simplifies the covariance structure for the posterior distribution. Under this assumption, the posterior covariance becomes a diagonal matrix, significantly reducing computational complexity. The Polya-Gamma variable $\boldsymbol{\omega}$ is drawn as follows:

$$\boldsymbol{\omega} \sim \mathrm{PG}(1, \boldsymbol{\psi}) \in \mathbb{R}^{n\times C\times C} \tag{C.43}$$

Using this formulation, we can update $\boldsymbol{\psi}_\theta$ based on $\boldsymbol{\kappa}$ and $\boldsymbol{\psi}_\theta = A\boldsymbol{\mu}_\theta$. Note that $\boldsymbol{\psi}_\theta$ and $\boldsymbol{\mu}_\theta$ are distinct, as $\boldsymbol{\mu}_\theta$ indicates the original logit value induced by the fine-tuning model. The posterior distribution for $\boldsymbol{\psi}_\theta$ can be expressed as:

$$\begin{aligned} p(\boldsymbol{\psi}_\theta^{(m)}|\boldsymbol{y}, \boldsymbol{\omega}) &\propto p(\boldsymbol{y}|\boldsymbol{\psi}_\theta, \boldsymbol{\omega})p(\boldsymbol{\psi}_\theta), \\ &\propto \mathcal{N}(\boldsymbol{\psi}_\theta|\tilde{\boldsymbol{\Sigma}}(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_\theta\boldsymbol{A} + \boldsymbol{\kappa}), \tilde{\boldsymbol{\Sigma}}) \end{aligned}$$

Here, $\tilde{\Sigma}$ is the posterior covariance, defined as:

$$\tilde{\boldsymbol{\Sigma}} = (\frac{\alpha}{2}\mathbf{I} + \boldsymbol{\Omega})^{-1} \tag{C.44}$$

where ${}^\alpha\!/_2\mathbf{I} \in \mathbb{R}^{(n\times C\times C)\times(n\times C\times C)}$ represents the identity matrix scaled by α, and $\boldsymbol{\Omega} \in \mathbb{R}^{(n\times C\times C)\times(n\times C\times C)}$ is the Polya-Gamma covariance matrix. Since we assume that all components of the matrices are diagonal, computing the inverse of $\tilde{\boldsymbol{\Sigma}}$ is straightforward, as it reduces to taking the reciprocal of the diagonal components. The drawn samples $\boldsymbol{\psi}_\theta^{(m)} \in \mathbb{R}^{n\times C\times C}$ are fed into the sigmoid function as described in Equation C.41, ensuring One-vs-Each SoftMax approximation with polya-gamma augmentation.

The interpretation of this procedure is that the parameter $\boldsymbol{\omega}$, influenced by the pre-trained models, regulates the degree of confidence each sigmoid function derives from the pre-trained models. Additionally, the label information $\boldsymbol{\kappa}$ contributes to the adjustment of the posterior distribution, although it does not entirely determine the final logit values. This process ensures that the fine-tuned models incorporate both pre-trained knowledge and label information, rather than relying exclusively on label information. When updating the loss function $\mathcal{L}_{\mathrm{nll}}$ with $\boldsymbol{\psi}_\theta$, the parameter $\boldsymbol{\psi}_\theta^{(m)}$ is influenced by both the label information and the pre-trained model. However, no single component completely dominates the process. This balance helps prevent overfitting, allowing the fine-tuned model ψ_θ to capture the general patterns of the training datasets without being overly influenced by individual data points.

C.4 More Details on algorithm 1

Here we would like to provide more details on our method.

C.4.1 Assumption on $\mathbf{A} \approx \mathbf{I}$

As described in subsection 3.4, we assume $\boldsymbol{A} \approx \boldsymbol{I}$. This means we can write down the posterior over ψ for each class separately, and they coincide. Hence, the posterior over $\boldsymbol{\psi}$ for class c is given by

$$p(\boldsymbol{\psi}^c|\boldsymbol{\omega}^c, \mathcal{D}) \propto \mathcal{N}(\boldsymbol{\psi}^c|\boldsymbol{\Sigma}^c(\frac{\alpha}{2}\boldsymbol{A}^c\boldsymbol{\mu}^c + \boldsymbol{\kappa}^c), \boldsymbol{\Sigma}^c) \tag{C.45}$$

where $\boldsymbol{\Sigma}^c = (\frac{\alpha}{2}\boldsymbol{I}_n + \boldsymbol{\Omega}^c)^{-1}$. This construction is motivated by purely computational reasons. By assuming the functions across classes are independent, we avoid computing the large covariance matrix of $(n\times C\times C)\times(n\times C\times C)$,

which is prohibitive when C is large. The size of the covariances in each class is $(n \times C) \times (n \times C)$, which is manageable when n is small, as in our case.

C.4.2 Two Different Priors

In practice, we use $\boldsymbol{f} \sim \mathcal{N}(\boldsymbol{\mu}, \alpha^{-1}\boldsymbol{I})$ as a prior when we do the posterior inference on the PG variables $\boldsymbol{\omega}$. On the other hand, we use $\boldsymbol{f}_{\boldsymbol{\theta}} \sim \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{\theta}}, \alpha^{-1}\boldsymbol{I})$ as a prior when we do the posterior inference on the function $\boldsymbol{f}_{\boldsymbol{\theta}}$ (Sampling of $\boldsymbol{\psi}_{\boldsymbol{\theta}}^{(m)}$ in algorithm 1 indicates that).

We acknowledge that this formulation deviates from the usual Bayesian construction. Our reasoning for this construction is that this simplifies the posterior updates over $\boldsymbol{\omega}$ and $\boldsymbol{f}_{\boldsymbol{\theta}}$. In the usual case where there are no parameters to optimize, we employ the Gibs steps, which involve alternating the updates for $\boldsymbol{\omega}_{\boldsymbol{\theta}}$ given $\boldsymbol{f}_{\boldsymbol{\theta}}$ and $\boldsymbol{f}_{\boldsymbol{\theta}}$ given $\boldsymbol{\omega}_{\boldsymbol{\theta}}$. This is computationally costly because whenever we step toward new values of $\boldsymbol{\theta}$ in optimization, we need to alternate these two steps. By setting $\boldsymbol{\omega}$ as a function of the prior $\boldsymbol{f}$ (i.e., parameter-independent), we do not update $\boldsymbol{\omega}$, resulting in speedy optimization.

C.4.3 Rigorous Definition of the Posterior over f

We approximate the posterior over $\boldsymbol{f}_{\boldsymbol{\theta}}$ defined as Equation 13 using Equation 12. In this case, the rigorous KL term yields $\|(\alpha\boldsymbol{I} + \boldsymbol{A}^\top\Omega\boldsymbol{A})^{-1}(\alpha\boldsymbol{\mu}_{\boldsymbol{\theta}} + \boldsymbol{A}\boldsymbol{\kappa}) - \boldsymbol{\mu}\|_2^2$ which depends on the parameters $\boldsymbol{\theta}$. We approximately reduce this to $\|c\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}\|_2^2$, where c is some constant with respect to the parameters $\boldsymbol{\theta}$. So, we still maintain the same regularization term given in the manuscript.

C.5 Algorithm of Proposed Method for CoOp_OVE_PG

In Algorithm 1, this explanation focuses on the CoOp version. In this context, we compare the proposed method with the CoOp version. The primary distinction between CoOp and CoCoOp lies in their approach to prompt learning. Specifically, CoOp employs the expression $I(X)^T T(\boldsymbol{p}^c_{\boldsymbol{\theta}_1})$, whereas CoCoOp utilizes $I(X)^T T(\boldsymbol{p}^c_{\boldsymbol{\theta}_1} + \boldsymbol{r}^c_{\boldsymbol{\theta}_2}(\boldsymbol{x}))$. The detailed procedure for this is outlined below.

Algorithm 2 One-vs-Each SoftMax Approximation with Pólya-Gamma Auxiliary Variables for Prompt Learning (CoOp based)

Input: Objective $\mathcal{L}_{\text{elbo}} = \{\mathcal{L}_{\text{nll}}, \mathcal{L}_{\text{KLD}}\}$, Class prompts $\mathcal{C} \in \{c_1, \cdots, c_C\}$, The learnable prompts $\mathbf{p}_{\boldsymbol{\theta}}$, The predefined prompts for the original CLIP $\mathbf{t} = \{t_1, \cdots, t_C\}$, Image-Encoder I and Text-encoder T, The number of parallel Gibbs chains M, Total number of classes C. Learning rate η, Weight for KLD β, α prior precision of $\boldsymbol{f}$, Einstein summation convention, denoted by $\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f}$.

Initialize: Parameters $\boldsymbol{\theta}$ randomly.

repeat

Sample a mini-batch $(\boldsymbol{x}, \boldsymbol{y})$ from a training data $\mathcal{D}$, where $\boldsymbol{x} \in \mathbb{R}^{n \times d}, \boldsymbol{y} \in \{0,1\}^{n \times C}$

$\boldsymbol{A} \leftarrow$ OVE-MATRIX$(\mathbf{y}) \in \mathbb{R}^{\mathbf{C}\times\mathbf{C}\times\mathbf{C}}$ in Equation 11

$\boldsymbol{\mu}_{\boldsymbol{\theta}} = I(\boldsymbol{x})^T T(\boldsymbol{p}^c_{\boldsymbol{\theta}}) \in \mathbb{R}^{n \times C}$

$\boldsymbol{\mu} = I(\boldsymbol{x})^T T(\boldsymbol{p}^c) \in \mathbb{R}^{n \times C}$

$\boldsymbol{\psi}_{\boldsymbol{\theta},nic} \leftarrow \sum_j \boldsymbol{A}_{ijc}\boldsymbol{\mu}_{\boldsymbol{\theta},ni}, \quad \boldsymbol{\psi}_{\boldsymbol{\theta}} \in \mathbb{R}^{N \times C \times C}$

$\boldsymbol{\psi}_{nic} \leftarrow \sum_j \boldsymbol{A}_{ijc}\boldsymbol{\mu}_{ni}, \quad \boldsymbol{\psi} \in \mathbb{R}^{n \times C \times C}$

$\boldsymbol{\kappa}_{nic} = \sum_j \boldsymbol{A}_{ijc}\big(\boldsymbol{y}_{ni} - 1/2\big), \quad \boldsymbol{\kappa} \in \mathbb{R}^{n \times C \times C}$

for $m = 1$ **to** M **do**

$\boldsymbol{\omega}^{(m)} \sim \text{PG}(1, \boldsymbol{\psi})$ — given in Equation 14

$\boldsymbol{\psi}_{\boldsymbol{\theta}}^{(m)} \sim p(\boldsymbol{\psi}^{(m)} | \boldsymbol{\omega}^{(m)}, \boldsymbol{\psi}_{\boldsymbol{\theta}}, \boldsymbol{\kappa}, \alpha)$ — given in Equation 3

$\boldsymbol{f}_{\boldsymbol{\theta}}^{(m)} \leftarrow \sum_C \boldsymbol{\psi}_{\boldsymbol{\theta}}^{(m)} \in \mathbb{R}^{n \times C}$ — due to $\boldsymbol{\psi} = \boldsymbol{A}\boldsymbol{f}_{\boldsymbol{\theta}}$

$\mathcal{L}_{\text{nll}}^{(m)} \leftarrow -\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c} \log f_{i,c}$ — given in Equation 9

$\mathcal{L}_{\text{KLD}}^{(m)} \leftarrow \beta\|\boldsymbol{\mu}_{\boldsymbol{\theta}} - \boldsymbol{\mu}\|_2^2$

end for

$\theta \leftarrow \theta - \frac{\eta}{M}\sum_{m=1}^{M} \nabla_\theta\big(\mathcal{L}_{\text{nll}}^m + \mathcal{L}_{\text{KLD}}^m\big)$

until convergence

D Experiments for 1D Multi-Class Classification

This section illustrates the experimental results of 3rd-order polynomial multi-class classification using SoftMax, OVE SoftMax approximation and the proposed method. The ground truth data is created by predefined Gaussian distributions. Three Gaussian distributions are used to represent the ground-truth classes: $p(x) \sim \mathcal{N}(1.0, 1.0^2)$, $q(x) \sim \mathcal{N}(0.0, 2.0^2)$ and $r(x) \sim \mathcal{N}(-1.0, 1.0^2)$. These distributions have different means and variances to create distinguishable classes. The probabilistic densities of ground truth and the samples from each distribution are described in the first column of Figure 1. The sample X is transformed using a polynomial basis function $\Phi(X)$, which is of degree 3 in this case. The learnable parameters β are applied to this transformed feature space. More specifically, $X \in \mathbb{R}^{N\times 1}$ where N is the number of data points, and 1 is the dimensionality of the input data. $\Phi(X) \in \mathbb{R}^{N\times 3}$ where applying the basis function of degree 3.

For this multi-class classification problem, which involves three classes, there are three sets of weights, each of size 3, represented by $\beta = (\beta_1, \beta_2, \beta_3) \in \mathbb{R}^{3\times 3}$. The second column of Figure 1 displays the results where β is trained using the standard SoftMax function. The third column shows the results when β is trained using the OVE SoftMax approximation. The fourth and fifth columns present the results of the proposed method, where β is trained with different values of the hyperparameter α, specifically $\alpha = 1$ and $\alpha = 1000$, respectively. The source code for these experiments is provided in the supplementary materials.

E Experiments for Fine-tuning on E-MNIST with Pre-trained ViT on MNIST

This section presents the experimental results of fine-tuning a pre-trained Vision Transformer (ViT) on the E-MNIST dataset, where the model was initially pre-trained on MNIST. To demonstrate the generalization capabilities of the proposed method across different tasks, we provide a simple example. First, we describe the ViT architecture, where its parameters are adapted for both the MNIST and E-MNIST datasets. Specifically, we use the ViT-3/7 structure with $D = 64$ (embedding dimension) and $H = 4$ (number of heads). During the pre-training phase, the model is trained on the full MNIST dataset. For the fine-tuning phase, we use a subset of the original E-MNIST Digit dataset, sampling only the first 100 data points from each class to simulate a typical fine-tuning scenario. Both MNIST and E-MNIST consist of 28×28 grayscale images, and the class labels between the two datasets align. For evaluation, we test the fine-tuned model on the E-MNIST Digit test dataset, which contains data points the model has not encountered during training. This setup ensures a rigorous evaluation of the model's ability to generalize from the pre-trained task (MNIST) to the fine-tuning task (E-MNIST). The source code for these experiments is included in the supplementary material.

F Extended Experimental Results

Our implementation is based on the code from Derakhshani et al. (2023a)[2]. All experiments were conducted on a single NVIDIA RTX 4090. Due to our limited computational resources, we conducted the experiments with a batch size of 4 for CoOp, MaPLe, APEX and 1 for CoCoOp. We followed the hyper-parameter setup from Derakhshani et al. (2023a).

For the seen-to-unseen generalization experiments, the seen and unseen categories were split according to the original CoOp work (Zhou et al., 2022b), which is a common setup for seen and unseen tasks across a wide range of studies (Zhou et al., 2022a; Khattak et al., 2023a; Yang et al., 2024). While we reported the results for the unseen categories in our main manuscript, we further provide results for both seen and unseen categories, along with their harmonic means. All experiments were conducted with 3 random seeds, and we report the standard deviations of these trials in Table F.2 and Table F.3.

Extended Results for Table 1. Here, we provide the performance comparison for seen classes and harmonic mean between seen and unseen categories in Table F.1. While OVE-PG introduces slight performance degradation in seen classes, it achieves higher or comparable performance from the perspective of the harmonic mean. Also, Table F.2 and Table F.3 demonstrates the extended results for Table 1 on six datasets, showing classification accuracy on seen and unseen categories and their harmonic mean, respectively. We also report the averaged harmonic mean across six datasets and the number of learnable parameters for each method in Table F.4. Notably,

[2] `https://github.com/saic-fi/Bayesian-Prompt-Learning`

	CoOp		CoCoOp		MaPLe		APEX	
	SoftMax	OVE-PG	Softmax	OVE-PG	SoftMax	OVE-PG	SoftMax	OVE-PG
Caltech101	97.70	97.87	97.33	97.33	97.85	97.67	97.77	97.53
DTD	79.37	77.43	74.30	75.17	80.13	81.97	78.73	81.37
EuroSAT	87.77	85.27	84.83	84.73	89.67	90.43	90.03	84.63
FGVC Aircraft	33.60	32.73	35.83	35.79	38.67	36.60	36.77	35.20
Food101	90.43	90.60	90.67	90.43	90.57	90.37	90.70	90.40
Flowers102	95.27	92.60	92.53	92.20	96.00	95.70	95.10	94.73
Oxford Pet	95.00	95.13	95.33	94.03	95.60	95.30	95.40	95.13
Stanford Cars	72.30	67.37	69.80	69.83	74.00	70.17	73.17	70.93
SUN397	80.90	78.27	79.33	76.40	81.20	79.33	81.57	79.83
UCF101	83.17	80.37	81.07	80.43	83.30	83.07	83.40	83.63
Average (Seen)	81.55	79.76	80.10	79.63	82.70	82.06	82.26	81.34
Average (H.M)	75.94	77.33	76.79	76.74	77.42	78.32	78.19	78.15

Table F.1: Comparison of accuracy (%) on unseen classes between Softmax and OVE-PG (Ours) for CoOP, CoCoOp, MaPLe, and APEX.

	DTD			**EuroSAT**			**FGVC Aircraft**		
	Seen	Unseen	HM	Seen	Unseen	HM	Seen	Unseen	HM
SoftMax	79.37 (2.31)	54.97 (1.36)	64.95 (1.71)	87.77 (1.68)	55.20 (3.55)	67.78 (2.28)	33.60 (4.89)	22.93 (17.46)	27.26 (7.64)
VPT	48.50 (1.30)	43.77 (2.04)	46.01 (1.59)	60.83 (5.11)	65.90 (3.04)	63.26 (3.81)	26.13 (1.93)	29.30 (0.53)	27.62 (0.83)
OVE-PG	77.43 (0.81)	55.67 (3.25)	64.77 (1.30)	85.27 (0.32)	71.23 (3.45)	77.62 (0.59)	32.73 (0.16)	36.23 (0.86)	34.39 (0.27)
	Flowers102			**Stanford Cars**			**UCF101**		
	Seen	Unseen	HM	Seen	Unseen	HM	Seen	Unseen	HM
SoftMax	95.27 (0.61)	72.47 (1.04)	82.32 (0.77)	72.30 (1.00)	72.73 (1.78)	72.51 (1.28)	83.17 (1.06)	72.43 (4.41)	77.43 (1.71)
VPT	66.17 (0.64)	76.40 (0.87)	70.92 (0.74)	62.37 (0.49)	73.60 (0.70)	67.52 (0.58)	69.47 (0.47)	72.60 (2.20)	71.00 (0.77)
OVE-PG	92.60 (0.35)	73.83 (0.81)	82.16 (0.48)	67.37 (1.05)	75.23 (0.57)	71.08 (0.74)	80.37 (1.61)	76.20 (0.32)	78.23 (0.53)

Table F.2: Comparison of accuracy (%) on seen-to-unseen generalization among Softmax, Bayesian VPT (Derakhshani et al., 2023a) and OVE-PG (Ours) for CoOp.

	DTD			**EuroSAT**			**FGVC Aircraft**		
	Seen	Unseen	HM	Seen	Unseen	HM	Seen	Unseen	HM
SoftMax	74.30 (2.25)	55.07 (2.60)	63.26 (2.41)	84.83 (0.92)	66.63 (10.26)	74.64 (1.69)	35.83 (0.25)	34.13 (1.27)	34.96 (0.42)
VPT†	75.60 (0.79)	59.17 (0.98)	66.38 (0.87)	64.20 (0.46)	69.10 (0.95)	66.56 (0.62)	34.67 (0.21)	34.80 (0.17)	34.73 (0.19)
OVE-PG	75.17 (0.97)	56.17 (3.22)	64.30 (1.49)	84.73 (1.66)	68.03 (3.99)	75.47 (2.34)	35.79 (0.87)	33.60 (0.46)	34.66 (0.60)
	Flowers102			**Stanford Cars**			**UCF101**		
	Seen	Unseen	HM	Seen	Unseen	HM	Seen	Unseen	HM
SoftMax	92.53 (0.55)	74.07 (0.46)	82.28 (0.50)	69.80 (0.62)	74.70 (0.98)	72.17 (0.76)	81.07 (0.59)	72.63 (1.08)	76.62 (0.76)
VPT†	93.20 (0.30)	73.77 (1.47)	82.35 (0.50)	69.00 (0.46)	75.30 (0.26)	72.01 (0.33)	80.13 (0.21)	78.17 (1.03)	79.14 (0.35)
OVE-PG	92.20 (0.71)	74.33 (0.93)	82.31 (0.81)	69.83 (0.61)	75.50 (0.36)	72.55 (0.45)	80.43 (0.86)	73.27 (0.96)	76.68 (0.91)

Table F.3: Comparison of accuracy (%) on seen-to-unseen generalization among Softmax, Bayesian VPT (Derakhshani et al., 2023a) and OVE-PG (Ours) for CoCoOp. † indicates that we conducted the experiments based on our implementation for CoCoOp, as the original work was only conducted on CoOp.

OVE-PG on CoOp achieves the best average performance in terms of harmonic mean among the six approaches (i.e., SoftMax, Bayesian VPT, and OVE-PG for both CoOp and CoCoOp), indicating that our OVE-PG effectively regularizes the model for fitting both seen and unseen categories. While our method requires the smallest training cost, except for vanilla CoOp, it can significantly improve the overall performance of prompt learning in VLMs.

Extended Results for Table 2. We also provide the extended results for Table 2 in Table F.5. We evaluate the trained models on additional evaluation datasets including Caltech101 (Fei-Fei et al., 2004b), Food101 (Bossard et al., 2014b), Oxford Pets (Parkhi et al., 2012), SUN397 (Xiao et al., 2010b). For both CoOp and CoCoOp, our proposed method showed competitive performance on these additional larger datasets consistently outperforms

CoOp						CoCoOp					
SoftMax		VPT		OVE-PG		SoftMax		VPT		OVE-PG	
AHM	# Param (K)	AHM	# Param (K)	AHM	# Param (K)	AHM	# Param (K)	AHM	# Param (K)	AHM	# Param (K)
65.38	8.19	57.72	3.07	**68.04**	8.19	67.32	35.36	66.86	659.46	67.66	35.36

Table F.4: Comparison of the averaged harmonic mean (AHM) across six different datasets and the number of learnable parameters (# Param). Higher AHM and fewer parameters are better. The best values are highlighted in **bold**.

		Source	**Target**										**AVG.**
		ImageNet	Caltech101	DTD	EuroSAT	Aircraft	Food101	Flowers102	Pets	Cars	SUN397	UCF101	
CoOp	SoftMax	76.5	95.9	51.7	51.3	26.7	90.0	72.6	90.7	62.8	71.8	67.2	68.84
	OVE-PG	74.5	95.9	57.4	61.7	23.8	89.3	71.6	89.8	63.6	71.0	69.9	69.86
CoCoOp	SoftMax	75.9	95.7	54.5	52.5	24.2	89.7	73.8	90.7	65.2	73.2	69.2	69.51
	OVE-PG	76.1	96.2	55.4	55.6	26.8	89.3	71.5	90.6	64.9	71.5	70.6	69.86

Table F.5: Extended results for Table 2. We further provide the evaluation results on Caltech101, Food101, Oxford Pets, SUN397 datasets.

	DTD							**EuroSAT**						
β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Seen	73.8	77.4	74.8	76.3	75.8	76.9	75.1	85.1	85.3	84.1	83.0	83.7	78.8	80.0
Unseen	54.2	58.6	53.4	56.3	55.2	55.7	52.9	70.4	72.7	73.9	71.6	74.7	69.1	70.8
HM	62.50	66.70	62.31	64.79	63.88	64.61	62.07	77.06	78.50	78.67	76.88	78.94	73.63	75.12
	FGVC Aircraft							**Flowers102**						
β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Seen	32.0	32.7	32.8	32.4	33.6	31.5	32.7	91.5	91.2	91.3	92.7	90.0	90.7	87.8
Unseen	28.1	36.2	34.6	34.6	34.8	33.5	36.2	75.0	73.8	75.2	75.6	75.5	75.7	76.6
HM	29.92	34.36	33.68	33.46	34.19	32.47	34.36	82.43	81.58	82.47	83.28	82.11	82.52	81.82
	Stanford Cars							**EuroSAT**						
β	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.1	0.2	0.3	0.4	0.5	0.6	0.7
Seen	67.0	67.4	67.2	68.0	67.8	67.5	67.1	79.6	80.4	80.5	79.8	78.7	79.5	79.0
Unseen	74.9	76.6	74.2	75.0	75.6	75.3	75.6	78.3	78.7	78.9	78.7	77.4	77.8	78.6
HM	29.92	34.36	33.68	33.46	34.19	32.47	34.36	82.43	81.58	82.47	83.28	82.11	82.52	81.82

Table F.6: Extended results for Figure 3. We further provide the results for sensitivity analyses on unseen category and harmonic mean (HM) of seen and unseen categories.

the SoftMax baseline on average.

Extended Results for and Figure 3. We further report the numerical values for Figure 3 in Table F.6. The values correspond to the results using a seed number of 1. Also, we further provide the results across different β both seen and unseen, and their harmonic mean, unlike Figure 3 in main manuscript which is restricted from the number of page limitation. In overall, we can conclude that similar observation also holds for seen and harmonic mean as well as for unseen in Figure 3 that within the midle range of β (i.e., $0.2 - 0.5$), OVE-PG shows robustness regardless of the specific choice of β.