Ph.D. DISSERTATION

Bayesian Deep Meta-Learning via Variational Inference: Applications in Few-Shot and Federated Learning

변이 추론을 통한 베이지안 심층 메타 학습: 퓨샷 및 연합 학습에서의 응용

AUGUST 2023

DEPARTMENT OF ELECTRICAL ENGINEERING &
COMPUTER SCIENCE
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Insu Jeon

Bayesian Deep Meta-Learning via Variational Inference:
Applications in Few-Shot and Federated Learning

변이 추론을 통한 베이지안 심층 메타 학습: 퓨샷 및 연합 학습에서의 응용

지도교수 김 건 희

이 논문을 공학박사학위논문으로 제출함

2023 년 5 월

서울대학교 대학원

전기 컴퓨터 공학부

전 인 수

전 인 수의 공학박사 학위논문을 인준함

2023 년 8 월

위 원 장	장 병 탁	(인)
부위원장	김 건 희	(인)
위 원	김 선	(인)
위 원	서 진 욱	(인)
위 원	최 성 준	(인)

Abstract

Meta-learning is a subfield of machine learning that aims to develop an algorithm capable of rapid adaptation to new tasks. This quick adaptation of machines can be achieved by leveraging a meta-learner, which learns the learning process rather than focusing on learning individual tasks. Then, the meta-learner can be utilized to adapt machines for new tasks efficiently. Recently, diverse meta-learning approaches have been introduced in this field, including metric-based, optimization-based, and model-based methods, and applied to many applications such as few-shot regression, few-shot classification, active learning, and reinforcement learning. However, the conventional meta-learning approaches, specifically the meta-learner, still have limitations, such as computational demands, scalability, and model over-fitting.

This thesis introduces a new Bayesian meta-learning approach called a Meta-Variaitonal Dropout (MetaVD). MetaVD leverages a hyper-network to approximate conditional dropout rates for each neural network weight. This facilitates quick reconfiguration of global learning and sharing neural networks for new tasks while enabling data-efficient learning in the multi-task environment. Several novel techniques regarding this framework are discussed, including the low-rank approximation for memory-efficient mapping of dropout rates for the entire neural network weights and a new shared variational prior interpretation for regularizing the dropout posterior. MetaVD is a versatile approach that can be applied to a wide range of conventional deep neural network algorithms. The proposed methodology was tested and demonstrated excellent adaptation and generalization performance in various few-shot learning applications, including

1d regression, image inpainting, and classification.

Federated learning (FL) is a research field in machine learning that aims to train a global inference model from remotely distributed local clients, gaining popularity due to its benefit of improving data privacy. However, conventional FL approaches encounter many challenges in practical scenarios, including model overfitting and diverging local models due to the limited and non-i.i.d. data among clients' devices. To address these issues, MetaVD is extended and applied to the distributed environment. In the FL, the shared hypernetwork is kept in the server and is learning to predict client-dependent dropout rates. This allows an effect model personalization of FL algorithms in the limited non-i.i.d. data settings. In addition, the posterior aggregation based on conditional dropout posterior is also introduced. We performed extensive experiments on various sparse and non-i.i.d. FL datasets. MetaVD demonstrated outstanding classification accuracy and generalization performance, particularly for out-of-distribution (OOD) clients. In addition, MetaVD compresses the local model parameters needed for each client, reducing communication costs and improving the calibration of the model prediction.

Overall, we propose a novel Bayesian meta-learning approach that can address many challenges in few-shot learning and federated learning applications. The conditional dropout posterior enables efficient model personalization, uncertainty calibration, and outstanding predictive performance. Experimental results show the excellent performance of the proposed approach. This contributes to the development of meta-learning and application in real scenarios.

Keywords: Deep Learning, Meta-Learning, Bayesian Neural Network, Variational Dropout, Multi-task Learning, Few-shot Learning, Federated Learning
Student Number: 2012-23237

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The evolution of deep learning in machine learning has resulted in remarkable progress in numerous applications, ranging from image recognition to natural language processing [1, 2, 3, 4]. The heart of this progress lies in the core assumption that a large amount of (labeled or unlabeled) data is available for training robust and generalizable models. However, this assumption may not hold in many real-world situations (e.g., medical imaging, robotics, military AI, and federated learning) due to the different devices, personal and environmental conditions, and security issues. In addition, the collected data typically follow a long-tail distribution[1]. The traditional deep-learning methods could be less effective for the generalization in tasks with limited data environments [5].

On the other hand, humans often can quickly understand new tasks and solve problems even from a few examples [6, 7, 8, 9]. A child does not require

[1] A distribution with a small number of tasks of high frequency and a large number of tasks of low frequency. This brings many challenges: data imbalance, underfitting, and overfitting.

Figure 1.1: A person who can ride snow-board might be able to learn to ride skate-board or surfing-board quickly. This quick adaptation ability of humans is due to the efficient generalization ability of their cognitive system.

thousands of images to differentiate between a dog and a cat. Instead, they can quickly learn general concepts and then can infer new dogs and cats they have never seen before. This remarkable learning efficiency of the human cognitive system is referred to as *meta-learning* (or *learning-to-learn*) ability [7, 8]. Humans can adapt to new tasks quickly since they can utilize past experiences for new learning. Humans do not just learn a task; they learn how to learn better.

The inquiry to develop machines with human-like adaptation efficiency has long been an active research topic in machine learning [10, 8, 11, 12, 13]. Meta-learning aims to design such machines that can seamlessly adapt to new tasks and generalize successfully even with limited data. This pursuit has led to a variety of different methodologies, including metric-based [14, 12, 13, 15, 16], model-based [17, 18, 19, 20, 21], and optimization-based methods [22, 23, 24, 11, 25, 26]. A fundamental idea shared by them is the concept of a higher-level learning algorithm, called a *meta-learner*, which, by drawing upon experiences from multiple tasks, can generalize for new, unseen tasks. This mimics the cognitive systems of humans, which enhance their adaptation capabilities for new tasks based on accumulated knowledge and experiences.

While the conventional meta-learning approaches have shown considerable

progress in many few-shot[2] tasks. These methodologies have not yet entirely overcome the issue of model overfitting yet [27]. Deep learning generally involves a vast number of parameters [1, 3, 2]. The conventional meta-learning methods are based on a point estimation of them. Thus, the meta-learner proposed in the previous works are still prone to overfitting on unseen tasks; simultaneously learning an adaptation-efficient but also robust meta-learner is difficult.

In response to this challenge, a new field known as Bayesian meta-learning has recently emerged [27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. Bayesian deep learning, which reinterprets deep learning models using a probabilistic perspective [37, 38, 39, 40, 41, 42, 43, 44], excels in preventing overfitting and quantifying inherent uncertainties of the model. Thus, incorporating the uncertainty and regularization mechanism into the meta-learning framework can improve the generalization. The Bayesian perspective also provides a unifying view of various existing approaches, broadening our understanding of meta-learning.

Despite those recent achievements, the field of Bayesian meta-learning is still in the development stage and poses several challenges. The optimization-based Bayesian approach [27, 28, 29, 36] is a universal approach since the adaptation of meta-learner is based on stochastic gradient descent (SGD) algorithm. However, they could require a costly computation of second-order derivatives, Hessians, of parameters and additional memory space to keep the parameter particles. On the other hand, the model-based approach [31, 32, 33, 34, 35] is more computationally efficient during testing than the optimization-based approach since the adaptation process in amortized in a deep learning model. Nonetheless, the model-based meta-learner like Neural Processes (NP) [33, 34, 45] exhibit underfitting and posterior-collapsing issues. Also, a hyper-network [46, 47, 48, 49]

[2]The few-shot learning assumes only a few examples (e.g., 1 or 5) are available for each task, with the number of tasks being large [8].

type of meta-learner, Versatile Amortized inference (VERSA) [50], can be challenging to scale due to the complexity of the meta-learner's output, and they still tend to overfit due to the lacking of explicit regularization mechanism [35]. These issues of robust adaptation and scalability can limit the practical applicability of Bayesian meta-learning approaches in real-world scenarios. Motivated by these challenges, in this thesis, we propose a new model-based Bayesian meta-learning approach, leveraging the dropout [42, 44, 43, 51, 52] and Variational Inference (VI) [53, 1, 54, 2] techniques. This new approach offers a more efficient and robust but also versatile deep meta-learning framework.

1.2 Contribution

The key contributions in this work can be summarized as follows:

Development of a Variational Dropout-based Meta-learning Approach (Chapter 3) In this thesis, we propose a new model-based Bayesian meta-learning approach called Meta-Variational Dropout (MetaVD). MetaVD is a new type of meta-learner that is built on dropout techniques and variational inference. MetaVD utilizes a model-based meta-learner to predict the task-specific dropout rates for each weight of the agent neural network (NN). This enables a quick reconfiguration of neural networks only by selectively switching off a subset of the global NN's weight. MetaVD can modulate various different task functions into one NN while data-efficiently learning the weight from multiple different sources. Our approach is also computationally efficient relative to the other model-based approach because the meta-learner only approximates the dropout rates (or partial variance) instead of the full distributions of weight. MetaVD also suffers less from the model-collapsing problem of the existing approach because the dropout can be efficiently applied to all layer of the agent

NN model. To enable this advancement, several techniques, such as a low-rank approximation of dropout rates, a shared and reversed prior distribution, and a theoretical analysis of them, have been presented. In addition, MetaVD assumes a well-posed Bayesian prior distribution for the meta-learner regularization. This can not only improve the generalization of the model's prediction but also compress the size of the weight. Lastly, MetaVD is a versatile approach; it can be combined with any existing (optimization-based) meta-learning or regular deep-learning method without touching their necessary assumption.

Application of MetaVD in Few-Shot Learning (Chapter 4) We have tested the MetaVD in various few-shot learning tasks: 1D regression, image inpainting, classification, and active learning tasks. The key attribute of these experiments was to support that the MetaVD can simultaneously improve the adaptation by fitting the data while capturing the variability of task function as well via the conditional dropout. To show this, we have provided a training objective of the MetaVD for few-shot learning. We also derived a computationally efficient inference algorithm based on a local reparameterization trick and stochastic gradient descent (SGD). We call this algorithm Neural Variational Dropout Processes (NVDPs), satisfying the exchangeable properity of a stoachasitc process. The experimental results demonstrate our method achieved outstanding performance compared to other model-based meta-learning methods [32, 34, 33, 55] in terms of many metrics such as log-likelihood, reconstruction, predictive accuracy for unseen tasks, and adaption efficiency.

Application of MetaVD in Federated Learning (Chapter 5) We have extended the MetaVD to address non-i.i.d. and limited data issues in the Federated Learning (FL) domain, offering a novel Bayesian FL approach. MetaVD

was extended to approximate the dropout for effectively estimating the client's personal model. We also present an additional novel posterior aggregation strategy based on the client-specific dropout uncertainty, enabling a more principled Bayesian way to consolidate the distributed local models into a global one. When applied to the conventional FL algorithms, MetaVD facilitates flexible model personalization across diverse non-i.i.d. clients' data, achieving state-of-the-art results across a wide range of experimental scenarios, such as different participation rates and multi-domain environments. In addition, the hierarchical prior is also leveraged, enabling model compression and reducing communication costs of exchanging the model weights for the efficient FL.

1.3 Thesis Organization

This thesis is structured as follows. Chapter 2 provides a quick background for deep learning, Bayesian learning, and Variational Dropout (VD). We also discuss the basic concept and recent developments of Bayesian meta-learning and challenges in this field. In Chapter 3, We introduce the Variational Inference (VI) framework for meta-learning and dropout. Then, a new Bayesian meta-learning approach, Meta-Variational Dropout (MetaVD), is presented. In Chapter 4, we demonstrated the experimental results of our approach in the various few-shot learning tasks. In Chapter 5, we further assess our approach in the Federated Learning (FL) domain, illustrating the versatility of MetaVD that can be combined with various other existing algorithms. Finally, we conclude the thesis in Chapter 6. Here, we summarize our findings, the potential implications of our contributions to the field of machine learning, and the aim for the future research.

Chapter 2

Background

2.1 Neural Networks

Neural Networks (NNs) are a subset of the machine learning models built based on a simple activation function called perception. Perceptron is a mathematical model of how a neuron transmits an electrical signal in our brain [56]. NNs consist of several layers of perceptrons, each of which utilizes a set of weights to transform its input and then applies an activation function to the output.

Mathematically, a NN model can be defined as a function $f : \mathbb{R}^n \to \mathbb{R}^m$ that maps an input vector $x \in \mathbb{R}^n$ to an output vector $y \in \mathbb{R}^m$. For a NN with L layers, the output for each l-th layer can be written as:

$$h^{(l)} = f^{(l)}(\mathbf{W}^{(l)} h^{(l-1)} + \mathbf{b}^{(l)}) \tag{2.1}$$

where $h^{(l)}$ is the output of the l-th layer, $\mathbf{W}^{(l)}$ and $\mathbf{b}^{(l)}$ are the weight matrix and bias vector (or the model parameters) for each l-th layer respectively, and $h^{(0)} = x$. In the multilayer architecture, the output of one layer serves as a new set of inputs for the following layer [1]. The activation function $f^{(l)}$ in the

intermediate layer is typically set as a non-linear function (e.g., a sigmoid [3] or ReLU [57]), making the NN a universal approximator [58] that can learn any complex patterns among the data. The last activation function $f^{(L)}$ is often set to be a softmax or an identity function depending on the machine learning tasks (e.g., classification or regression). Then the NN function's output can be given as $\hat{y} = f(x) = (f^{(1)} \circ \cdots \circ f^{(L)})(x)$. An optimal set of NN parameters, the weights and biases, is typically obtained by minimizing a predefined error (or objective) function $O(\hat{y}, y)$ through backpropagation and gradient descent algorithms [1, 3, 59].

NN models are characterized by their hierarchical structure, which enables them to learn complex representations of input data. Initial layers typically capture low-level features, such as edges in an image, while deeper layers combine these to interpret higher-level concepts [60, 61]. The capacity to learn such high-level representations directly from raw data, without the need for manual feature engineering, is a key advantage of deep learning. Recent advancements in deep architectures based on NNs, have led to the development of models such as Convolutional Neural Networks (CNNs) [62], Recurrent Neural Networks (RNNs) [63], Long Short-Term Memory (LSTM) networks [64], and attention models [65, 66]. These deep learning models have demonstrated remarkable success across a range of machine learning tasks, from image generation to natural language processing [67, 68].

2.2 Maximum Likelihood Estimation of Neural Networks

Maximum Likelihood Estimation (MLE) is a statistical framework for optimizing a NN model. In machine learning, a conditional probability, or likelihood, is often constructed with the NN model [1, 2]. For example, a simple probabilistic

modeling of input x and output y can be described as follows:

$$p(y|x;\phi) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - f_\phi(x))^2}{2\sigma^2}\right) \tag{2.2}$$

Here, the likelihood model $p(y|x;\phi)$ is a Gaussian distribution. The output of the NN model, $f_\phi(x)$, acts as a conditional mean of the distribution over the output y. The parameters vectors of the NN model are represented by $\phi = \{(\mathbf{W}^{(l)}, \mathbf{b}^{(l)})\}_{l=1}^{L}$. The equation 2.2 indicates that a Gaussian noise is added to the observed output (e.g., $y = f_\phi(x) + \epsilon,$ where $\epsilon \sim N(0, \sigma^2)$) where σ^2 is the variance of random noise. Accounting for the noise in the data and interpreting the NN as a conditional likelihood is a standard approach in probabilistic machine learning[1] since most real-world label data are often distorted or corrupted by noise due to measurement or human errors.

Given a training dataset, $D = (x_i, y_i)_{i=1}^{N}$ with N independent and identically distributed (i.i.d.) pairs of input and output data. The core principle of MLE is to find the optimal model parameter ϕ that maximizes the likelihood model $p(y|x;\phi)$ under the dataset D. The mathematical expression for the MLE objective, $L(\phi)$, can be given as:

$$L(\phi) = p(D;\phi) = \prod_{i=1}^{N} p(y_i|x_i,\phi). \tag{2.3}$$

Each likelihood $p(y_i|x_i,\phi)$ represents the conditional probability of observing the output y_i given the input x_i and NN parameter ϕ. Thus, the product of the likelihoods for each individual data pair yields the probability of observing the entire dataset.

In practice, we minimize the negative log-likelihood, $-\ell(\phi) = -\log L(\phi)$ due to mathematical convenience. Since the logarithm is a strictly increasing function, minimizing the negative log-likelihood maximizes the likelihood. Thus,

[1]For a classification problem, we assume a Bernoulli or Multinomial distribution.

the optimal parameter under the MLE is typically expressed as:

$$\phi_{\text{MLE}} = \arg\min_{\phi} -\ell(\phi) = \arg\min_{\phi} \left(\sum_{i=1}^{N} -\log p(y_i|x_i;\phi) \right). \tag{2.4}$$

If we assume the Gaussian likelihood model $p_\phi(y|x)$ defined by equation 2.2 and a fixed noise variance $\sigma^2 = 1$, the negative log-likelihood can be reduced to a squared loss over the dataset, $-\ell(\phi) \simeq \frac{1}{2}\sum_{i=1}^{N}(y_i - f_\phi(x_i))^2$, which is a typical error function used for the NN training in the classical regression problem[2]. The minimization of the $-\ell(\phi)$ with respect to the parameter ϕ is non-trivial due to the complexity of the NN function. However, modern development in the optimization algorithms, such as backpropagation, gradient descent, and activation function such as ReLU, enables efficient optimal parameter search under the framework of MLE [69].

Although MLE is a standard methodology for optimizing the NN model, they do have limitations. As the number of training epochs in the gradient descent algorithm increases, the model becomes increasingly adept at fitting the training data. However, learning the training data too specific can lead to a failure in generalizing to unseen data. This phenomenon, known as overfitting [70, 71], occurs when the complexity of the model is high relative to the available training data. The optimization process with MLE does not include any regularization mechanism for the model complexity. As such, MLE typically requires a lot of training data to prevent the NN model from overfitting. Unfortunately, accessing such large datasets is not always possible. From a methodological perspective, the overfitting issue is further compounded by the fixed point estimation of NN parameters. This is because a single set of NN parameters cannot adequately learn the systemic uncertainty within the whole dataset.

[2] If $p_\phi(y|x)$ is a Bernoulli distribution, the $-\ell(\phi)$ is a binary entropy loss for classification.

2.3 Bayesian Neural Networks

Bayesian Neural Networks (BNNs) are alternatives to traditional neural network models, which employ Bayes' theorem for the robust parameter optimization [72, 73, 74]. Bayes' theorem is a concept in probability theory and statistics learning that describes the updating process of our prior belief of an event based on new observations. From the model optimization perspective, this means that we would like to learn a probability distribution over the NN parameters, $p(\phi)$, (for weights and biases) using the dataset rather than a fixed point estimation of them. The prior distribution also allows for incorporating previous knowledge about the model parameter.

Suppose a training dataset D and the likelihood model $p(y|x, \phi)$ is given as in the equation 2.2. Bayes' rule can be applied to compute the posterior distribution as follows:

$$p(\phi|\mathcal{D}) = \frac{p(\phi)p(\mathcal{D}|\phi)}{p(\mathcal{D})} = \frac{p(\phi)\prod_{i=1}^{N} p(y_i|x_i, \phi)}{\prod_{i=1}^{N}\int_{\phi} p(y_i|x_i, \phi)p(\phi)\,\mathrm{d}\phi}. \tag{2.5}$$

In this equation, the likelihood $p(\mathcal{D}|\phi)$ represents the probability of observing the data, while the prior $p(\phi)$ reflects our initial beliefs about the NN model parameters (e.g, the prior is often modeled as a Gaussian distribution). The posterior distribution $p(\phi|\mathcal{D})$ represents the updated probability of the parameters after incorporating the observed data. Essentially, computing the optimal posterior $p(\phi|\mathcal{D})$ is to goal in Bayesian learning. This posterior distribution over the parameters in BNNs can help to mitigate the model overfitting issues and also provides a means to quantify uncertainty in the model's predictions.

Posterior Inference Computing the posterior distribution in equation 2.5 requires the computation of the evidence $p(\mathcal{D})$: a normalization constant that ensures the posterior distribution is a valid probability distribution. However,

solving the internalization in the evidence is a difficult problem due to the complexity of the parameter structure. Although there could exist a conjugate prior relationship for a simple likelihood model, which allows a simple computation of the posterior distribution. In the case of the likelihood model with a NN, the direct computation of the exact posterior is computationally infeasible.

Therefore, various approximation methods for computing the posterior distribution of BNNs have been studied in the past. Buntine and Weigend [75] proposed maximum-a-posteriori (MAP) schemes for neural networks and introduced second-order derivatives in the prior distribution to encourage smoothness in the approximate posterior distribution. Hinton and Van Camp [76] later introduced variational methods that served as regularizers in NNs, with the amount of information in weights controlled by adding Gaussian noise. Hochreiter and Schmidhuber [64] incorporated an information theory perspective, utilizing a minimum description length (MDL) loss that penalized non-robust weights based on perturbations of the weights on the outputs. Denker and LeCun [77] and MacKay [74] investigated posterior probability distributions of neural networks using Laplace approximations. Neal [72] explored the use of hybrid Monte Carlo methods for training neural networks, although scalability remained a challenge for large networks in practical applications. More recently, Graves [78] derived a variational inference scheme for neural networks, and Blundell et al. [79] extended it with an unbiased update for the variance. Dropout and Gaussian Dropout have also been viewed as approximate variational inference schemes [42, 80, 42].

BNNs with Markov Chain Monte Carlo sampling One straightforward approach to approximating the posterior distribution in BNNs is the Markov chain Monte Carlo (MCMC) method [72, 81]. MCMC methods for BNNs state

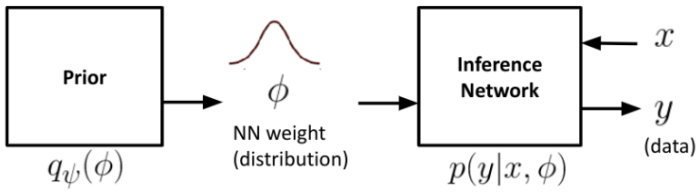


Figure 2.1: An illustration of Bayesian Neural Network framework. $q_\psi(\phi)$ is the prior distribution over the parameter ϕ of NN model $p(y|x,\phi)$. ψ is the parameter of prior. Bayesian learning is a prior distribution learning.

that we can employ a predefined posterior distribution $q_\psi(\phi)$ to approximate the exact posterior $p(\phi|\mathcal{D})$ via a simple sampling technique. This allows us to learn a probability distribution over the model parameters instead of a deterministic one. MC sampling technique for approximating the BNNs' posterior can be defined as follows:

$$L(\psi) = \frac{1}{S}\sum_{s=1}^{S} p(y|x,\phi_s) \;\text{ where }\; \phi_s \sim q_\psi(\phi) \tag{2.6}$$

Here, the approximate posterior predictive distribution is obtained by sampling from the approximate posterior distribution $q_\psi(\phi)$ and averaging the predictions over the samples. S represents the number of samples used in the approximation; the more samples are taken, the better the distribution is approximated. Here, the goal is to maximize a parameter ψ that defines the approximate posterior distribution $q_\psi(\phi)$ rather than the original NN parameters ϕ. MCMC approach updates the posterior distribution using the samples obtained from the partially updated posterior distribution. By iteratively updating the approximate posterior distribution on the subset of the dataset, it will reach the desired equilibrium states, which can approximate the true posterior $p(\phi|\mathcal{D})$. Stochastic Gradient Descent (SGD) [82] and parameterization trick [83] is commonly used to optimize the parameter of approximating posterior ψ via maximizing

the posterior distribution in the equation 2.6. Once the optimal approximate posterior distribution $q_\psi(\psi)$ is learned, we can also make predictions using the posterior predictive distribution. The posterior predictive distribution provides uncertainty quantification in predicting a new observation given the new input.

BNNs with Variational Inference Another popular approach to optimize the BNNs is Variational Inference (VI) [76, 78, 79, 80, 84]. The VI approach sets a variational posterior model $q_\psi(\phi)$ parameterized by a tractable parameter ψ to approximate the true posterior distribution $p(\phi|\mathcal{D})$. The learning objective in VI is the Kullback-Leibler divergence $\mathrm{KL}(q_\psi(\phi)|p(\phi|\mathcal{D}))$. In practice, the optimization problem for the tractable variational posterior $q_\psi(\phi)$ is defined as:

$$\log p(\mathcal{D}) \geq \mathbb{E}_{q_\psi(\phi)}[\log p_(\mathbf{y}|\mathbf{x};\phi)] - \mathrm{KL}(q_\psi(\phi)||p(\phi)). \tag{2.7}$$

The right side of equation 2.7, also referred to as the evidence lower bound (ELBO), embodies a trade-off between the expected log-likelihood on the training dataset $\mathcal{D}$ and the KL term with some pre-specified prior distribution $p(\phi)$. Maximizing the ELBO in equation 2.7 with respect to the tractable variational parameter ψ is equivalent to minimizing the divergence $\mathrm{KL}(q_\psi(\phi)||p(\phi|\mathcal{D}))$. Thus, the variational posterior $q_\psi(\phi)$ is closely optimized to the true posterior distribution. Unlike the MCMC sampling technique, the VI technique utilizes the explicit prior distribution $p(\phi)$ in the optimization to enable a stochastic regularization for the $q_\psi(\phi)$, which can prevent the degeneration of the model.

Posterior Predictive distribution Once we learned the variational posterior distribution $q_{\psi^*}(\phi)$ with an optimal variational parameter ψ^* via the MCMC sampling or VI, it can be utilized to approximate the posterior distribution $p(\phi|D)$. The learned posterior distribution provides valuable insights about the model parameter (e.g., the uncertainty and the structure of the

parameter). One primary use of the posterior distribution is to compute the posterior predictive distribution [1, 2, 85]. The posterior predictive distribution allows us to predict the probability of the next data point given newly observed data. Although computing the exact posterior predictive distribution for NNs is difficult due to the complex parameter, we can approximate it by utilizing the variational posterior with MCMC sampling technique [72, 81] defined as follows:

$$p(y_{\text{new}}|x_{\text{new}}, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y_{\text{new}}|x_{\text{new}}, \phi_s) \text{ where } \phi_s \sim q_{\psi^*}(\phi) \tag{2.8}$$

where x_{new} is the new input and y_{new} is the new data point we want to predict. The equation equation 2.8 states that, under Bayesian learning, we should aggregate all the possible parameters that we can get from the posterior distribution $p(\phi|D)$ to predict the new output, which also takes into account the uncertainty in the parameter estimates [80, 40]. This can provide the uncertainty quantification in predicting y_{new} given the new unseen input x_{new}.

2.4 Variational Dropout

Variational Dropout [83, 42, 86, 87, 88] is a set of approaches that models the variational posterior over the NN parameter $q_\psi(\phi)$ in equation 2.7 based on the dropout regularization technique [89, 90]. The dropout regularization randomly turns off some of the Neural Network (NN) parameters during training by multiplying discrete Bernoulli random noises to the parameters. This technique was initially popularized as an efficient way to prevent the NN model's over-fitting.

Consider a fully connected neural network (NN) with L layers. During training, dropout can be applied to the deterministic parameters θ of each l-th layer. This operation can be formalized as follows [91, 92]:

$$B = A(\Xi \circ \theta), \text{ with } \Xi \sim \text{Bernoulli}(1 - \mathbf{p}) \tag{2.9}$$

Here, A represents the $M \times K$ matrix of input features for a mini-batch of M data points, while B corresponds to the $M \times D$ output matrix before applying the activation function. θ is the $K \times D$ parameter matrix for each layer of the NN. The dropout operation introduces a $K \times D$ noise matrix Ξ whose elements $\Xi_{k,d}$ are independently sampled from a Bernoulli distribution with dropout rate $\mathbf{p}_{k,d}$. Each parameter of the matrix θ is then multiplied by the corresponding noise element in Ξ using the element-wise product operation, denoted by $\circ$. Then, we get the masked NN parameter $\phi = \Xi \circ \theta$. In essence, the variational dropout procedure perturbs the model parameters during training using the randomly sampled noise matrix, which helps prevent overfitting by effectively averaging over many different network configurations.

Gaussian Dropout Later, fast dropout [92, 93] proposes an alternative to the conventional Bernoulli dropout: it replaces the Bernoulli noise with continuous noise drawn from a Gaussian distribution, $\Xi \sim \mathcal{N}(1, \alpha)$, where α is reparameterized dropout parameter defined by $\alpha = (1 - \mathbf{p})/\mathbf{p}$. This Gaussian dropout approximates the Bernoulli dropout, as justified by the Central Limit Theorem [89, 92]. Thus the approximate posterior distribution can be defined as:

$$q_\psi(\phi) = \prod_{k=1}^{K} \prod_{d=1}^{D} q(\phi_{k,d}) = \prod_{k=1}^{K} \prod_{d=1}^{D} \mathcal{N}(\phi_{k,d} | \theta_{k,d}, \alpha_{k,d}\theta_{k,d}^2). \tag{2.10}$$

In the equation above, the mean and variance of each independent Gaussian distribution are dictated by the independent dropout rate $\alpha_{k,d}$ and the deterministic parameter $\theta_{k,d}$, respectively. Thus, in Gaussian dropout, the variational parameter of the posterior model is $\psi = \{\alpha, \theta\}$. The significant advantage of this approach lies in its ease of interpretation. Gaussian dropout yields a fully factorized posterior distribution over the independent parameters of the NN.

Unlike conventional dropout[89, 90], which uses a single fixed dropout rate for all parameters, VD assigns individual dropout rates to each parameter, effectively learning a unique regularization for every feature in the model [83, 87, 86, 42, 88]. The learned dropout rates can be interpreted as the variational parameters of the approximated posterior distribution in the Bayesian Neural Network. This aspect of VD has been shown to effectively capture the model's uncertainty from data, improving its generalization ability.

Prior The optimization of ELBO objective in equation 2.7 requires a prior distribution $p(\phi)$. VD approaches often employ sparse priors for regularization [44, 43, 51, 52], which facilitate the learning of independent dropout rates on the NN parameters. In the original VD literature [83], the posterior model equation 2.10 with a specially chosen log-uniform prior (e.g., $p(\log(|\phi|) \propto c$) is employed to optimize the ELBO in equation 2.7. The log-uniform prior in the VD was designed to satisfy that the analytical derivation of the $\mathrm{KL}(q_\psi(\phi)||p(\phi))$ in equation 2.7 does not depend on the θ. This allows the consent optimization of the ELBO w.r.t all the independent variational parameters (*i.e.* $\theta_{l,k,d}$ and $\mathbf{p}_{l,k,d}$ for all the $l = 1 \dots L$, $k = 1 \dots K$ and $d = 1 \dots D$) via the SGD algorithm. The learnable dropout rates for each independent parameter distinguish the VD from the conventional dropout approaches using the same fixed dropout rate for all the parameters. The VD approaches based on the dropout posterior have been proven effective for learning the uncertainty of the model from the data (i.e., $q_\psi(\phi) \approx p(\phi|\mathcal{D})$) and improving the model's generalization [83, 87, 88, 86, 42, 88].

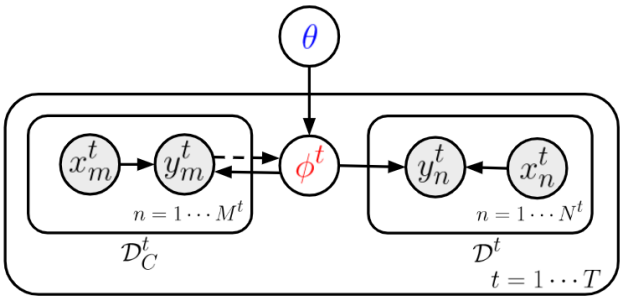


Figure 2.2: Probabilistic graphical model of Bayesian multi-task learning framework given a multiple dataset $\mathcal{D}^t$ for each task $t = 1, ..., T$. The θ is a global parameter shared across different tasks. ϕ^t is a task-specific parameter for each task. $\mathcal{D}_C^t$ is a small subset of data for each task (e.g., $M^t (<< N^t)$), which is used as a context set to approximate the ϕ^t.

2.5 Bayesian Meta-Learning

A goal of meta-learning is to construct a model that can quickly solve new tasks from small amounts of observed data [10, 8, 11, 12, 13]. A fundamental idea shared by them is the concept of a higher-level learning algorithm, called a *meta-learner*, which allows the learning process of learning itself rather than only learning a specific task. The premise is that if the meta-learner has encountered numerous similar tasks, it might accumulate enough knowledge to generalize across different tasks. However, the conventional meta-learning methods are based on a point estimation [17, 18, 19, 20, 21, 22, 23, 24, 11, 25, 26]. Thus, the meta-learner proposed in the previous approach was prone to overfitting.

Recently, various Bayesian meta-learning approaches have been introduced, interpreting the meta-learning as a posterior distribution approximation in Bayesian Learning [27, 28, 29, 30, 31, 32, 33, 34, 35, 36]. A multi-task learning environment is often assumed for training a robust meta-learner. Suppose a collection of T related tasks is given and each t-th task has the training data $\mathcal{D}^t$ containing N i.i.d. observed tuples $(\mathbf{x}^t, \mathbf{y}^t) = (x_i^t, y_i^t)_{i=1}^N$, Bayesian meta-learning

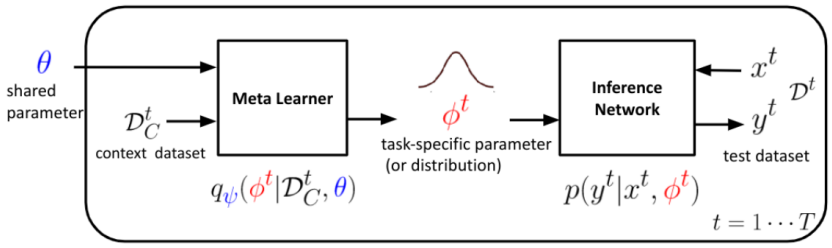


Figure 2.3: Bayesian meta-learning is a posterior predictive distribution learning across multiple tasks. $p(y^t|x^t, \phi^t)$ is a likelihood (or NN model) on the t-th training data and ϕ^t is a t-th task-specific variable. The $q_\psi(\phi^t|\mathcal{D}_C^t, \theta)$ is a conditional posterior distribution (or meta learner), learned to predict the task-specific parameter ϕ^t conditioned on a smll set $\mathcal{D}_C^t$. The θ is a global parameter shared across different tasks.

objective can be defined using MCMC sampling as follows:

$$L(\theta, \psi) = \frac{1}{T}\sum_{t=1}^{T} \log \frac{1}{S}\sum_{s=1}^{S} p(y^t|x^t, \phi_s^t) \text{ where } \phi_s^t \sim q_\psi(\phi^t|\mathcal{D}_C^t, \theta) \tag{2.11}$$

Here, $p(y^t|x^t, \phi^t)$ is a likelihood (or NN model) on the t-th training data and ϕ^t is a t-th task-specific variable (*i.e.* a latent representation or weights of NN). The θ is a global parameter shared across different tasks. This hierarchical structure enables data-efficient learning. $\mathcal{D}_C^t$ is an additional small sampled data (e.g., $M^t(<< N^t)$) for each task, which is used as a context set (e.g., an input to meta learner) in meta-learning. The $q_\psi(\phi^t|\mathcal{D}_C^t, \theta)$ is a conditional posterior distribution (or meta learner), learned to map the context set $\mathcal{D}_C^t$ to predict the distribution over the task-specific parameter ϕ^t. The objective corresponds to the approximation of log posterior predictive distribution over the T tasks (or datasets) while sharing the global parameter θ (and ψ) across tasks. Based on the modeling approach of the task-specific posterior $q_\psi(\phi^t|\mathcal{D}_C^t, \theta)$, many of the recent Bayesian meta-learning approaches can be roughly categorized into the optimization-based [27, 28, 29, 31, 30, 36] or the model-based posterior approximation approaches [32, 34, 33, 55, 35].

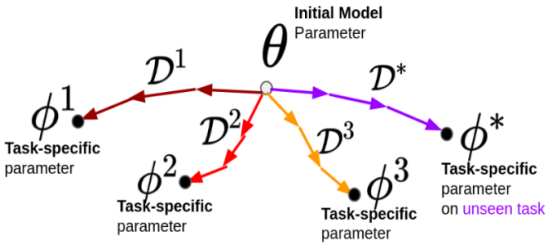


Figure 2.4: MAML aims to find a global parameter θ that can rapidly adapt to the related task parameter ϕ^t with only a few gradient update steps. The meta-learner in this context is an implicit gradient descent optimizer.

2.6 Model Agnostic Meta-Learning (MAML)

Model-Agnostic Meta-Learning (MAML) [25] is a meta-learning algorithm designed to quickly adapt to new tasks with a small amount of data. The core idea is to train an initial NN model parameter θ on a variety of tasks such that it can learn new tasks using only a small number of gradient updates. The learning objective for the initial parameter θ in MAML can be formalized as follows:

$$\theta^* = \arg\min_{\theta} \sum_{t=1}^{T} L_t(\theta - \eta \nabla_\theta L_t(\theta; \mathcal{D}_C^t); \mathcal{D}^t), \tag{2.12}$$

Where $\nabla_\theta L_t(\theta, \mathcal{D}_C^t)$ is the gradient of the loss computed on the small subset[3] of data $\mathcal{D}_C^t$ (that is also called meta-training set). η is the adaptation step size (or learning rate). By substrating the gradient from the initial parameter θ (formalizing one-step gradient update in this case), we get a task-specific parameter $\phi^t(\theta, \mathcal{D}_C^t) = \theta - \eta \nabla_\theta L_t(\theta; \mathcal{D}_C^t)$. In general, a few inner gradient updates on meta-training data $\mathcal{D}_C^t$ are performed to get the tasks specific parameter ϕ^t. Then, we minimize the loss $L_t(\phi^t(\theta, \mathcal{D}_C^t); \mathcal{D}^t)$ computed on the meta-test dataset $\mathcal{D}^t$ with respect to the initial parameter θ. Hence, MAML aims to find

[3]The subset $\mathcal{D}_C^t (\subseteq \mathcal{D}^t)$ is known as a context (or support) set for each task. A small S size of context set (e.g., 1-shot, 5-shot, or random) is often used in the few-shot learning tasks [8].

a parameter θ such that, after a small number of gradient steps on task t, the updated parameter ϕ^t performs well on task t.

The objective equation 2.12 can be optimized with the SGD algorithm. The crucial aspect of MAML lies in its ability to compute second-order derivatives (or the Hessian) with respect to the parameter θ. While calculating the Hessian is generally expensive, MAML avoids an explicit computation by using the automatic differentiation engine such as Pytorch or Tensorflow or by utilizing a first-order approximation [26].

Bayesian view of MAML In the optimization-based Bayesian meta-learning approaches, the task-specific variable ϕ^t can be seen as the adapted NN weights. For example, the posterior distribution (or meta-learner) of MAML [25] in equation 2.11 can be considered as a Dirac delta variational posterior modeling:

$$q(\phi^t|\mathcal{D}_C^t;\theta) \approx \delta(\phi^t - \mathrm{SGD}_j(\mathcal{D}_C^t,\theta)) \tag{2.13}$$

where the goal is to learn the shared global initialization NN's parameters θ such that a few j steps of SGD updates on the small subset $\mathcal{D}_C^t$ of the t-th dataset $\mathcal{D}^t$ provides a good approximation of the task-specific weights ϕ^t.

Other optimization-based approaches Many optimization-based meta-learning methods such as LLAMA [27], PLATIPIS [29], BMAML [28], and ABML [31] have incorporated the Gaussian type of posterior and prior models into the deterministic adaptation framework of MAML to improve the robustness of models. However, the adaptation cost of optimization-based meta-learning methods is often computationally expensive due to the inversion of the Hessian or kernel matrix; they may not be suitable for environments with limited computing resources at test.

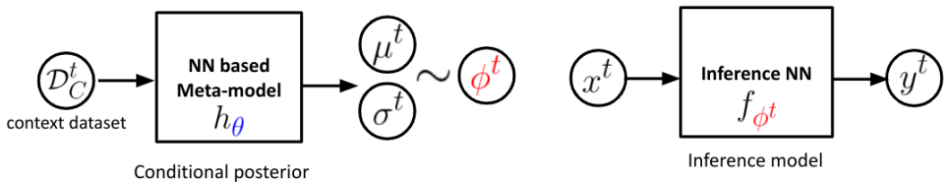


Figure 2.5: VERSA is a model-based Bayesian approach that extends the posterior distribution in BNNs as a meta-learner.

2.7 Versatile Amortized Inference (VERSA)

The model-based Bayesian meta-learning approaches such as VERSA [32] allow an instant estimation of the Bayesian predictive distribution at test time via NN-based conditional posterior modeling. VERSA employs an additional NN-based meta-model $g_\theta(\cdot)$ parameterized by θ to directly predict the Gaussian posterior of agent NN's task-specific weight ϕ^t from the small context set $\mathcal{D}_C^t$:

$$q(\phi^t|\mathcal{D}_C^t;\theta) = \mathcal{N}(\phi^t|(\mu,\sigma) = g_\theta(\mathcal{D}_C^t)). \tag{2.14}$$

In this case, the shared structure θ represents the meta-model's parameters. However, the direct approximation of agent NN weights in VERSA could limit their scalability due to the large dimensionality of the NN's weights; they only consider the task-specific weights for the single softmax output layer. In addition, VERSA did not utilize any task-specific prior $p(\phi^t)$, so the conditional posterior could collapse into a deterministic one while training with the Monte Carlo approximation [35].

2.8 Neural Processes (NPs)

Stochastic processes Stochastic processes in the regression problem treat the function f as a random variable and impose a distribution over the function as $p(f)$. Given that the n pairs of input and output data $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$

are independently sampled from the unknown distribution, the likelihood of stochastic processes can be defined as:

$$p(y_{1:n}|x_{1:n}) = \int p(y_{1:n}|f, x_{1:n})p(f)df = \int \prod_{i=1}^{n} \mathcal{N}(y_i|f(x_i), \sigma^2)p(f)df. \quad (2.15)$$

The σ^2 is an observation variance[4] and $f(x_i)$ is a predicted mean for the output y_i using the input x_i and a function f drawn from the distribution over function $p(f)$. The stochastic processes allow for reasoning about the uncertainty in the multiple underlying functions that might be presented in the data-generating process. Modeling $p(f) = \mathcal{GP}(\mathcal{N}(m(x_i), k(x_{1:n}, x_{1:n})))$ (as Gaussian Processes [94]) is a popular machine learning approach.

Neuarl Processes Recently, Garnelo et al. [33] proposed NN model-based stochastic processes method called Neural Processes (NPs), combining desirable proprieties of Gaussian Processes (GPs) [94] and the NN. NPs offer an implicit measure of the distribution over function that can be learned from data efficiently, avoiding the requirement for specifying a suitable type of kernel in GPs. The likelihood of NPs is defined as follows

$$p(y_{1:n}|x_{1:n}) = \int p(y_{1:n}|z, x_{1:n})p(z)dz = \int \prod_{i=1}^{n} \mathcal{N}(y_i|g_\theta(x_i, z), \sigma^2)p(z)dz. \quad (2.16)$$

Where the z is the latent variable, $p(z) = \mathcal{N}(z; 0, I)$ is a multivariate Normal prior, and $g_\theta(x, z)$ is a *decoder* NN parameterized by θ in NPs. the variable z is introduced for modeling the distribution over function $p(f)$ in equation 2.15. The randomness of the *decoder* $g_\theta(x_i, z)$ in NPs is arising due to the stochasticity in the representation z. Each sample of z would correspond to one realization of the stochastic process. Since the *decoder* is non-linear, amortized variational

[4] In the experiment, σ^2 is also learned as $\sigma^2(x_i)$. For clarity, we use a fixed variance notion.

inference (VI) is performed to learn the parameter θ. The evidence lower-bound (ELBO) objective of NPs can be given by:

$$\log p(y_{1:n}|x_{1:n}) \geq \mathbb{E}_{q(z|x_{1:n},y_{1:n})}[\log p_\theta(y_{1:n}|x_{1:n},z)] - D_{\mathrm{KL}}(q_\phi(z|x_{1:n},y_{1:n})||p(z)). \tag{2.17}$$

Here, the variational posterior $q_\phi(z|x_{1:n},y_{1:n})$ is modeled as a factorized Gaussian with mean $\mu_\phi(\cdot)$ and variance $\sigma_\phi(\cdot)$ NN parameterized by ϕ, analogous to the *encoder* in variational auto-encoders (VAE) [95], but the variational posterior in NPs is defined on a set of data $\mathcal{D} = \{x_i, y_i\}_{i=1}^n$ as:

$$q_\phi(z|x_{1:n},y_{1:n}) = \mathcal{N}(z; \mu_\phi(r)), \sigma_\phi(r))), \text{where } r = a(\{r_i\}_{i=1}^n), r_i = h_\psi(x_i,y_i), \tag{2.18}$$

where $h_\psi(\cdot)$ is a NN parameterized by ψ, an *aggregator* a is defined by the mean function *i.e.* , $a(\cdot) = \sum(\cdot)/n$, and r is a representation summarising the dataset [33, 96]. In other words, $\mu_\phi(\cdot)$ and $\sigma_\phi(\cdot)$ in equation 2.18 take encoded and aggregated input-output pairs as inputs and parameterize a normal distribution from which z is sampled. The reparameterization trick [95] applied to sample the z. The optimization of the parameter θ, ϕ, ψ of NPs can be performed via Stochastic Gradient Descent [33].

Posterior predictive inference Although the first ELBO formulation of NPs was given as equation 2.17. Carmelo et al. [33] introduced another formulation of NPs that better reflects the Bayesian posterior predictive propriety of GPs [94]. Consider the set of dataset $\mathcal{D}$ is randomly *split* into input-out pairs of the *context set* $\mathcal{D}_C = \{x_C, y_C\} = \{x_i, y_i\}_{i=1}^m$ of size m, and the *target set* $\mathcal{D}_T = \{x_T, y_T\} = \{x_i, y_i\}_{i=1}^n$ of size n at each training iteration ($\mathcal{D}_C \nsubseteq \mathcal{D}_T$ in general [33], $\mathcal{D}_C \subseteq \mathcal{D}_T$ in practice [34, 55]).

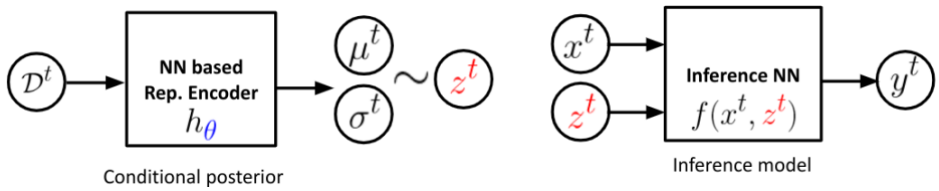


Figure 2.6: NPs are model-based Bayesian approaches that employ a representation-based posterior distribution as a meta-learner.

In the *split* dataset regime, another VI formulation of NPs [33] provides an ELBO objective to approximate the posterior predictive distribution as follows:

$$\log p(y_T|x_T, x_C, y_C) \geq \mathbb{E}_{q(z|x_T,y_T)}[\log p(y_T|x_T, z)] - D_{\mathrm{KL}}(q(z|x_T, y_T)||q(z|x_C, y_C)), \tag{2.19}$$

where the prior $p(z)$ in equation 2.17 is replaced with the variational posterior $q(z|x_C, y_C)$ in equation 2.19, approximating the unknown true posterior distribution $p(z|x_C, y_C)$. That is to say, the *decoder* of NPs trained with equation 2.19 learns to reconstruct targets, regularised by the KL term that encourages the posterior representation z given the target set to be not too far from the posterior representation z given the context set.

Bayesian view of NPs Although NPs [33, 97] were originally proposed as efficient NN-based stochastic processes. The meta-learner in NPs can be interpreted as a representation-based posterior distribution approximation in the context of Bayesian meta-learning of equation 2.5. The conditional posterior employing the latent representation z^t in NPs can be interpreted as:

$$q(z^t|\mathcal{D}^t; \theta) = \mathcal{N}(z^t|(\mu, \sigma) = g_\theta(\mathcal{D}^t)) \tag{2.20}$$

The task-specific latent representation z^t is approximated from the NN-based meta-learner $g_\theta(\mathcal{D}^t$, which approximates the mean, μ, and variance, σ, of the

latent representation, based on the t-th task dataset $mathcalD^t$. Regarding the z^t as a second input to the NN model $p(y^t|x^t, z^t)$ (similar to [95] or [98]) provides an efficient way to conditioning the agent NN [33].

Despite many desirable properties of NPs, one weakness is that the *decoder* NNs of NPs tends to underfit the context set [55, 99]. Kim et al. [55] hypothesis that the underfitting behavior of NPs is due to the mean aggregation step in the encoder of NPs acts as a bottleneck and introduce an attention mechanism [65, 100] to mitigate the problem. By letting the *decoder* of NP take an additional deterministic representation extracted by using the multi-head cross attention between input and context set, ANPs [55] could significantly resolve the underfitting issues in NPs.

NPs are a flexible approach for modeling stochastic processes since the distribution over function in NPs is based on NNs and can be learned from data directly. However, the high flexibility of the *decoder* NN could still cause the risk of overfitting when the size of training data is small [89, 101]. Allowing the deterministic paths to *decoder* might also cause the model tends to ignore the latent variable z when they have a deterministic path [102] since the posterior collapsing problem of the latent model, a powerful decoder such as NN tends to ignore the latent variable z, is well-known [103, 104, 105, 106, 107, 108, 109, 110]. We conjecture that this overfitting behavior in ANPs is involved with the original formulation of neural processes. Although the posterior representation z with targets is trained to be not too far from the posterior with the contexts by the KL term, the posterior conditioned on the small contexts set, $q(z|\mathcal{D}_C)$, at test time might not be sufficient to estimate the expected term for predicting new targets.

Chapter 3

Methodology: Meta-Variational Dropout

3.1 Variational Inference for Bayesian Meta-Learning

A goal of meta-learning is to construct a model that can quickly solve new tasks from small amounts of observed data. To achieve this, it is important to learn a general (or task-invariant) structure from multiple tasks that can be utilized for efficient model adaptation when necessary. Bayesian meta-learning methods [32, 33, 31] formulate this objective as an amortized variational inference (VI) of the posterior distribution in a multi-task environment.

Suppose a collection of T related tasks is given, and each t-th task has the training data $\mathcal{D}^t$ containing N i.i.d. observed tuples $(\mathbf{x}^t, \mathbf{y}^t) = (x_i^t, y_i^t)_{i=1}^N$. Then, the evidence lower-bound (ELBO) over the log-likelihood of the multi-

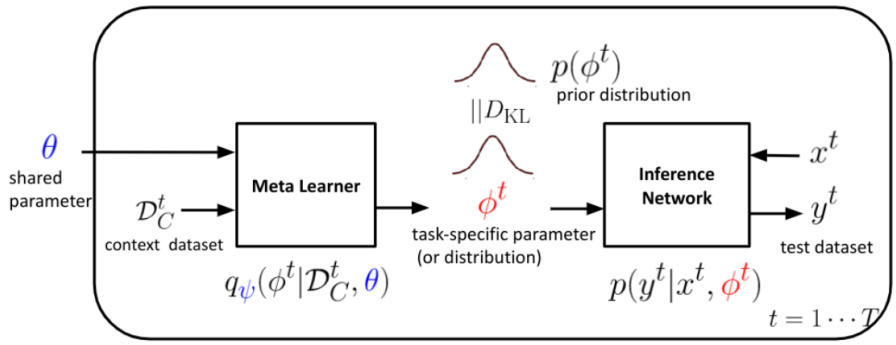
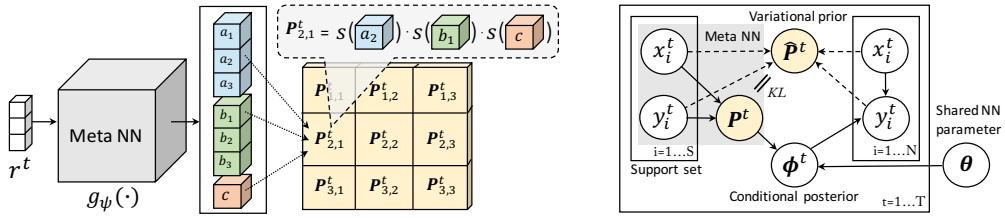


Figure 3.1: Illustration of VI of Bayesian meta-learning in a multi-task learning setting. $p(y^t|x^t, \phi^t)$ is a likelihood (or NN model) on the t-th training data and ϕ^t is a t-th task-specific variable. The $q_\psi(\phi^t|\mathcal{D}_C^t, \theta)$ is a conditional posterior distribution (or meta learner), learned to predict the task-specific parameter ϕ^t conditioned on a smll set $\mathcal{D}_C^t$. The θ is a global parameter shared across different tasks. In the VI framework, we can explicitly regularize the meta-learner with a prior distribution $p(\phi^t)$.

task dataset can be derived as:

$$\sum_{t=1}^{T} \log p(\mathcal{D}^t; \theta) \geq \sum_{t=1}^{T} \{\mathbb{E}_{q(\phi^t|\mathcal{D}^t)}[\log p(\mathbf{y}^t|\mathbf{x}^t, \phi^t)] - \mathrm{KL}(q(\phi^t|\mathcal{D}^t; \theta)||p(\phi^t))\}. \tag{3.1}$$

Here, $p(\mathbf{y}^t|\mathbf{x}^t, \phi^t)$ is a likelihood (or NN model) on the t-th training data and ϕ^t is a t-th task-specific variable (*i.e.* a latent representation or weights of NN) and $q(\phi^t|\mathcal{D}^t; \theta)$ is a tractable amortized variational posterior model utilized to approximate the true unknown posterior distribution over ϕ^t for each given t-th task data (i.e., $p(\phi^t|\mathcal{D}^t)$) [95, 32, 33, 31, 35]. The parameter θ represents the common structure that can be efficiently learned across multiple different tasks. The prespecified prior distribution $p(\phi^t)$ in the Kullback–Leibler (KL) divergence term provides a stochastic regularization that can help to capture the task-conditional uncertainty and prevent the collapsing of $q(\phi^t|\mathcal{D}^t; \theta)$. In fact, the maximization of the ELBO, right side of equation 3.1, with respect



(a) The low-rank product of Bernoulli experts meta-model

(b) The probabilistic graphical model of NVDP's

Figure 3.2: (a) The low-rank product of Bernoulli experts meta-model of conditional *dropout* posterior. (b) The probabilistic graphical model of Neural Variational Dropout Processes (NVDPs) with the variational prior. Where $\{x_i^t, y_i^t\}_{i=1}^N$ is the N i.i.d samples from the t-th training dataset $\mathcal{D}^t$ among T tasks. The context set $\mathcal{D}_C^t = \{x_i^t, y_i^t\}_{i=1}^S$ is a small subset of the t-th training dataset.

to the conditional variational posterior model is equivalent to the minimization of $\sum_{t=1}^T \mathrm{KL}(q(\phi^t|\mathcal{D}^t;\theta)||p(\phi^t|\mathcal{D}^t))$. Essentially, the goal in the amortized VI of Bayesian meta-learning is to learn the inference process of the true conditional posterior distribution via the variational model $q(\phi^t|\mathcal{D}^t;\theta)$ and the shared general structure θ across multiple tasks since this task-invariant knowledge can later be utilized for the efficient adaptation of the NN function on new unseen tasks. The approximation of the conditional posterior also enables ensemble modeling and uncertainty quantification.

3.2 Meta Variational Dropout

This section introduces a new model-based Bayesian meta-learning approach called Neural Variational Dropout Processes (NVDPs). Unlike the existing methods such as NPs or VERSA employing conditional latent representation or direct modeling of NN's weights, NVDPs extend the posterior modeling of the Variational Dropout (VD) in the context of meta-learning. We also introduce a new type of task-specific prior to optimizing the conditional *dropout* posterior in variational inference.

Conditional dropout posterior. We propose a new amortized variational posterior model that can be efficiently adapted for each given task. Suppose we train a fully connected NN of L layers, then a conditional *dropout* posterior[1] based on the task-specific dropout rates $\mathbf{P}^t$ over the $K \times D$ dimensional deterministic parameters θ of each l-th layer of the NN can be given as follows:

$$q(\phi^t|\mathcal{D}_C^t;\theta) = \prod_{k=1}^{K}\prod_{d=1}^{D} q(\phi_{k,d}^t|\mathcal{D}_C^t) \tag{3.2}$$

$$= \prod_{k=1}^{K}\prod_{d=1}^{D} \mathcal{N}(\phi_{k,d}^t|(1-\mathbf{P}_{k,d}^t)\theta_{k,d}, \mathbf{P}_{k,d}^t(1-\mathbf{P}_{k,d}^t)\theta_{k,d}^2). \tag{3.3}$$

The parameter[2] $\theta_{k,d}$ is shared across different tasks, representing the common task-invariant structure. In the equation 5.3, the task-specific NN parameter ϕ^t are fully described by the mean and variance of each independent Gaussian distribution via $\theta_{k,d}$ and $\mathbf{P}_{k,d}^t$. This, the $\theta_{k,d}$ can be seen as a shared module that operates on separate tasks selectively by the conditional dropout rates $\mathbf{P}_{k,d}^t$, Note that the variational posterior model is explicitly conditioned on the subset of the t-th training dataset $\mathcal{D}_C^t = \{\mathbf{x}_i^t, \mathbf{y}_i^t\}_{i=1}^S$ $(\subseteq \mathcal{D}^t)$ known as the t-th context set. The key idea of conditional posterior modeling in NVDPs is to employ an NN-based meta-model to predict the task-specific dropout rate $\mathbf{P}_{k,d}^t$ from the small context set $\mathcal{D}_C^t$. The meta-model to approximate $\mathbf{P}_{k,d}^t$ for each given task is simply defined as:

$$\mathbf{P}_{k,d}^t = s(\mathbf{a}_k)\cdot s(\mathbf{b}_d)\cdot s(\mathbf{c}), \text{ where } (\mathbf{a},\mathbf{b},\mathbf{c}) = g_\psi(r^t). \tag{3.4}$$

Here, the set representation r^t is defined as the mean of features obtained from each data in t-th context set $\mathcal{D}_C^t$ (*i.e.*, $r^t = \sum_{i=1}^S h_\omega(\mathbf{x}_i^t, \mathbf{y}_i^t)/S$, where h_ω is a

[1]The original Gaussian approximation in the VD is $q(\phi_{k,d}) = \mathcal{N}(\phi_{k,d}|\theta_{k,d}, \alpha\theta_{k,d}^2)$ with $\alpha = \mathbf{P}_{k,d}/(1-\mathbf{P}_{k,d})$. But, NVDPs extend the Bernoulli *dropout* model [92, 91].

[2]We omit the layer index l of the parameter $\theta_{l,k,d}$ for brevity.

feature extracting NN parameterized by ω), summarizing order invariant set information [98, 111]. The $g_\psi(\cdot)$ is meta NN model parameterized by ψ to predict a set of logit vectors (i.e., $\mathbf{a} \in \mathbb{R}^K, \mathbf{b} \in \mathbb{R}^D$ and $\mathbf{c} \in \mathbb{R}$). Then, the *sigmoid* function with a learnable temperature parameter τ (i.e., $s_\tau(\cdot) = 1/(1+\exp(-(\cdot)/\tau)$) is applied to them to get the low-rank components of task-specific dropout rates $\mathbf{P}^t_{k,d}$: the row-wise $s(\mathbf{a}_k)$, column-wise $s(\mathbf{b}_d)$, and layer-wise dropout rate $s(\mathbf{c})$. In other words, the task-specific dropout rate $\mathbf{P}^t_{k,d}$ is obtained by multiplying low-rank components of the conditionally approximated dropout rates from the NN-based meta-model $g_\psi(\mathcal{D}^t_C)$ (see Figure1 (a)).

The product of n Bernoulli random variables is also a Bernoulli variable [112]. By exploiting this property, we interpret the approximation of the task-specific dropout rates in terms of the low-rank product of Bernoulli experts. Unlike VERSA whose meta-model's complexity is $\mathcal{O}(LKD)$ to model the full NN weight posterior directly, the complexity in NVDPs is $\mathcal{O}(L(K+D+1))$. In addition, the meta-model's role is only to predict the low-rank components of the task-specific dropout rates. With the shared parameter θ, this can greatly reduce the complexity of the posterior distribution approximation of the high-dimensional task-specific NN's weights using only a few observed context examples. The product model tends to give sharp probability boundaries, which is often used for modeling the high dimensional data space [113].

In the Bayesian perspective, the permutation invariant representation r in MetaVD is particularly important due to the *exchangeability* By choosing the aggregator $a(\cdot)$ in r to be the mean function, MetaVD could model the *epistemic* distribution of function that is invariant to an arbitrary number and order of the observed contexts [114, 96]. In probability theory, *de Finetti's theorem* also states that the *exchangeability* guarantees the existence of a prior distribution on model's weight $p(w)$ [115, 116]. The amortized PoB VD can be regarded as

an application of *de Finetti's theorem*; The existence $(1-p)$ of each parameter in the *decoder* NN is governed by the (meta) NN conditioned on the permutation invariant representation r from which the posterior on weight is defined.

3.3 Variational Prior

To optimize the conditional *dropout* posterior in equation 5.3, a specification of the prior distribution $p(\phi^t)$ is necessary to get a tractable derivation of the KL regularization term of the ELBO in equation 3.1. The question is how we can define the effective task-specific prior. In fact, an important requirement in the choice of the prior distribution in the conventional VD framework [83] is that the analytical derivation of the KL term (*i.e.,* $\mathrm{KL}(q(\phi^t|\mathcal{D}^t_C;\theta)||p(\phi^t))$ in equation 3.1) should not depend on the deterministic NN parameter θ. This allows the constant optimization of the ELBO w.r.t all the independent variational parameters (*i.e.,* $\theta_{l,k,d}$ and $\mathbf{P}_{l,k,d}$ for all $l = 1 \dots L$, $k = 1 \dots K$ and $d = 1 \dots D$). One way of modeling the prior is to employ the log-uniform prior $p(\log(|\phi|) \propto c$ as in the conventional VD [83]. However, a recently known limitation is that the log-uniform prior is an improper prior (*e.g.,* the KL divergence between the posterior and the log-uniform prior is infinite). This could yield a degeneration of the *dropout* posterior model to a deterministic one [86, 87, 117, 88]. Besides, the conventional prior used in VD approaches does not support a task-dependent regularization.

We introduce a new task-specific prior modeling approach to optimize the proposed conditional posterior; it is approximated with the *variational prior* defined by the same *dropout* posterior model in equation 5.3 except that the prior is conditioned on the whole task data (i.e., $p(\phi^t) \approx q(\phi^t|\mathcal{D}^t)$). The KL divergence between the conditional *dropout* posterior (with the only context

set) and the *variational prior* (with the whole task data) can be derived as:

$$\mathrm{KL}(q(\phi^t|\mathcal{D}^t_C)||q(\phi^t|\mathcal{D}^t)) \tag{3.5}$$

$$= \sum_{k=1}^{K}\sum_{d=1}^{D}\{\frac{\mathbf{P}^t_{k,d}(1-\mathbf{P}^t_{k,d}) + (\widehat{\mathbf{P}}^t_{k,d} - \mathbf{P}^t_{k,d})^2}{2\widehat{\mathbf{P}}^t_{k,d}(1-\widehat{\mathbf{P}}^t_{k,d})} + \frac{1}{2}\log\frac{\widehat{\mathbf{P}}^t_{k,d}(1-\widehat{\mathbf{P}}^t_{k,d})}{\mathbf{P}^t_{k,d}(1-\mathbf{P}^t_{k,d})}\} \tag{3.6}$$

where both $\mathbf{P}_{k,d}$ and $\widehat{\mathbf{P}}_{k,d}$ are the dropout rates predicted from the meta-model but with different conditional set information: $\mathbf{P}_{k,d}$ is obtained from the small context set $\mathcal{D}^t_C$, while $\widehat{\mathbf{P}}_{k,d}$ is from the whole task set $\mathcal{D}^t$. Interestingly, the analytical derivation of the KL is independent of the shared parameter θ, thus this satisfies the necessary condition to be used as a prior in the VI optimization of the *dropout* posterior. Figure 1(b) depicts the variational prior is conditioned on the whole dataset.

The shared posterior model for the task-specific prior introduced here was inspired by recent Bayesian meta-learning approaches [33, 55, 35], but some critical differences are that: 1) we have developed the *variational prior* to regularize the task-specific dropout rates in the optimization of the conditional *dropout* posterior, 2) the denominator and the numerator of the KL divergence term in equation 3.5 are reversed compared with the existing approaches. In fact, some other recent studies of amortized VI inference [118, 119] analytically derived that the optimal prior maximizing VI objective is the variational posterior aggregated on the whole dataset: $p^*(\phi) = \int_{\mathcal{D}} q(\phi|\mathcal{D})p(\mathcal{D})$. Thus, we hypothesized the conditional posterior model that depends on the whole dataset should be used to approximate the optimal prior $p^*(\phi^t) \approx q(\phi^t|\mathcal{D}^t)$ since the variational model conditioned on the aggregated set representation with much larger context (e.g.,$\mathcal{D}^t \supseteq \mathcal{D}^t_C$) is likely to be much closer to the optimal task-specific prior than the model conditioned only on a subset. The experiments in Section 5.4 demonstrate that the proposed *variational prior* approach provides a reliable

regularization for the conditional *dropout* posterior and the similar formulation is also applicable to the latent variable-based conditional posterior models [33, 55].

3.4 Derivation of the ELBO

This section describes a detailed derivation of the evidence lower-bound (ELBO) of NVDPs in equation 5.4. Given the context data $\mathcal{D}_C^t = (\mathbf{x}_C^t, \mathbf{y}_C^t)$ and target data $\mathcal{D}^t = (\mathbf{x}^t, \mathbf{y}^t)$, the KL divergence between the true unknown posterior distribution over parameter $p(\phi^t|\mathbf{x}^t, \mathbf{y}^t)$ and the (conditional) variational posterior $q(\phi^t|\mathcal{D}_C^t)$ is given by:

$$\begin{aligned}
&\sum_{t=1}^{T} D_{\mathrm{KL}}(q(\phi^t|\mathcal{D}_C^t)||p(\phi^t|\mathcal{D}^t)) = \sum_{t=1}^{T} \int q(\phi^t|\mathcal{D}_C^t) \log \frac{q(\phi^t|\mathcal{D}_C^t)}{p(\phi^t|\mathbf{x}^t, \mathbf{y}^t)} d\phi^t \\
&= \sum_{t=1}^{T} \int q(\phi^t|\mathcal{D}_C^t) \log \frac{q(\phi^t|\mathcal{D}_C^t) p(\mathbf{y}^t|\mathbf{x}^t)}{p(\mathbf{y}^t|\mathbf{x}^t, \phi^t) p(\phi^t)} d\phi^t \qquad (3.7) \\
&= \sum_{t=1}^{T} \int q(\phi^t|\mathcal{D}_C^t) \Big\{ \log \frac{q(\phi^t|\mathcal{D}_C^t)}{p(\phi^t)} + \log p(\mathbf{y}^t|\mathbf{x}^t) - \log p(\mathbf{y}^t|\phi^t, \mathbf{x}^t) \Big\} d\phi^t \\
&= \sum_{t=1}^{T} D_{\mathrm{KL}}(q(\phi^t|\mathcal{D}_C^t)||p(\phi^t)) + \log p(\mathbf{y}^t|\mathbf{x}^t) - \mathbb{E}_{q(\phi^t|\mathcal{D}_C^t)}[\log p(\mathbf{y}^t|\phi^t, \mathbf{x}^t)]. \qquad (3.8)
\end{aligned}$$

The step (7) is due to Bayes rule of $p(\phi^t|\mathbf{x}^t, \mathbf{y}^t) = \frac{p(\mathbf{y}^t|\phi^t, \mathbf{x}^t) p(\phi^t)}{p(\mathbf{y}^t|\mathbf{x}^t)}$ where ϕ^t is often assumed to be independent of $\mathbf{x}^t$. By reordering (8), we get

$$\begin{aligned}
\sum_{t=1}^{T} \log p(\mathbf{y}^t|\mathbf{x}^t) &\geq \sum_{t=1}^{T} \mathbb{E}_{q(\phi^t|\mathcal{D}_C^t)}[\log p(\mathbf{y}^t|\mathbf{x}^t, \phi^t)] - D_{\mathrm{KL}}(q(\phi^t|\mathcal{D}_C^t)||p(\phi^t)) \qquad (3.9) \\
&\approx \sum_{t=1}^{T} \mathbb{E}_{q(\phi^t|\mathcal{D}_C^t)}[\log p(\mathbf{y}^t|\phi^t, \mathbf{x}^t, \mathcal{D}_C^t)] - D_{\mathrm{KL}}(q(\phi^t|\mathcal{D}_C^t)||q(\phi^t|\mathcal{D}^t)) \qquad (3.10)
\end{aligned}$$

The last step is due to the non-negativity of the $\sum_{t=1}^{T} D_{\mathrm{KL}}(q(\phi^t|\mathcal{D}_C^t)||p(\phi^t|\mathcal{D}^t))$. In the lower-bound of (9), the choice of $p(\phi^t)$ is often difficult since the biased

prior can lead to over-fitting or under-fitting of the model. However, some recent studies of the amortized VI inference [118, 119] analytically discussed that the optimal prior in the amortized variational inference is the aggregated conditional posterior model on whole dataset: $p^*(\phi) = \int_{\mathcal{D}} q(\phi|\mathcal{D})p(\mathcal{D})$. Usually, the aggregated posterior cannot be calculated in a closed form due to the expensive computation cost of the integral. However, the aggregation on the whole dataset in the model-based conditional posterior is constructed based on the set representation. This motivates us to define the conditional posterior given the whole task dataset as an empirical approximation of the optimal prior (i.e., $p^*(\phi^t) \approx q(\phi^t|\mathcal{D}^t)$). We call this a *variational prior*. Thus, the approximation of $p^t(\phi)$ in (9) with the *variational prior* yields the approximated lower bound of (10).

3.5 Derivation of the KL Divergence

Gaussian Approximation. An essential requirement in the choice of the prior is that the analytical derivation of the KL divergence term in equation (10) should not depend on the deterministic NN parameter θ [83, 86, 87, 117, 88]. This allows the constant optimization of the ELBO w.r.t all the independent variational parameters (*i.e.* $\theta_{l,k,d}$ and $\mathbf{P}_{l,k,d}$ for all $l = 1 \dots L$, $k = 1 \dots K$, and $d = 1 \dots D$). In fact, the conditional posterior defined in (2) is a Gaussian

distribution, thus $\mathrm{KL}(q(\phi^t|\mathcal{D}_C^t;\theta)||q(\phi^t|\mathcal{D}^t;\theta))$ is analytically defined as:

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(\phi^t|\mathcal{D}_C^t;\theta)||q(\phi^t|\mathcal{D}^t;\theta))) \\
&= \frac{\mathbf{P}^t(1-\mathbf{P}^t)\theta^2 + ((1-\mathbf{P}^t)\theta - (1-\widehat{\mathbf{P}}^t)\theta)^2}{2\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)\theta^2} + \log\frac{\sqrt{\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)\theta^2}}{\sqrt{\mathbf{P}^t(1-\mathbf{P}^t)\theta^2}} - \frac{1}{2} \qquad (3.11) \\
&= \frac{\mathbf{P}^t(1-\mathbf{P}^t)\cancel{\theta^2} + (\widehat{\mathbf{P}}^t-\mathbf{P}^t)^2\cancel{\theta^2}}{2\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)\cancel{\theta^2}} + \frac{1}{2}\log\frac{\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)\cancel{\theta^2}}{\mathbf{P}^t(1-\mathbf{P}^t)\cancel{\theta^2}} - \frac{1}{2} \qquad (3.12) \\
&= \frac{\mathbf{P}^t(1-\mathbf{P}^t) + (\widehat{\mathbf{P}}^t-\mathbf{P}^t)^2}{2\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)} + \frac{1}{2}\log\frac{\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)}{\mathbf{P}^t(1-\mathbf{P}^t)} - \frac{1}{2} \qquad (3.13)
\end{aligned}
$$

where both $\mathbf{P}$ and $\widehat{\mathbf{P}}$ are the dropout rates predicted from the meta-model via the equation (3) but with different conditional set information: $\mathbf{P}$ is obtained from the small context set $\mathcal{D}_C^t$, while $\widehat{\mathbf{P}}$ is from the whole task set $\mathcal{D}^t$. (11) is derived using the analytical formulation of KL divergence between two Gaussian distributions. (13) is equivalent to the KL term defined in (4) of the manuscript (except that the constant term 1/2 is omitted for brevity). Interestingly, the analytical derivation of the KL is independent of the shared parameter θ, thus this satisfies the necessary condition to be used as a prior in the VI optimization of the *dropout* posterior.

The KL divergence in (13) intuitively means that the dropout rates predicted from a small context set should be close to the dropout rates predicted from a much larger context set while training. The experiments validated that this surprisingly works well to induce a robust conditional functional uncertainty. However, one practical issue while training the dropout rate with the KL term (13) is that the dropout rate could converge to zeros during the early training period due to the larger gradients from the KL than from the likelihood [3]. In practice, we adopt the dropout rate clipping technique often used in other

[3] It also turns out to be a floating-point exception problem. We later set the minimum noise (i.e., *eps*) to $1e$ - 10 in the calculation of log, sqrt, and division function, and the collapsing problem did not occur again.

Variational Dropout approaches [83, 87, 88, 86, 42, 88]. We use the dropout rate range of (0.01, 0.99) for all experiments.

Bernoulli Approximation. If $I \in \{0, 1\}$ is a Bernoulli random variable, denoted $I \sim \textit{Bernoulli}(p)$, the probability mass function is defined as:

$$f(I; p) = \begin{cases} p & \text{if } I = 1, \\ 1 - p & \text{if } I = 0. \end{cases} \tag{3.14}$$

Bernoulli random variables and indicator variables are two aspects of the same concept. The random variable I is called an indicator variable for an event A if $I = 1$ when A occurs and $I = 0$ if A does not occur. $p(I = 1) = p(A)$ and $E[I] = P(A)$. Indicator random variables are Bernoulli random variables, with $p = p(A)$.

The (conditional) *Bernoulli* dropout (DropConnect) on the weight in the manuscript is defined[4] as:

$$w_{k,d} = \Xi_{k,d} \cdot \theta_{k,d}, \text{ where } \Xi_{k,d} \sim \textit{Bernoulli}(1 - p_{k,d}), \text{ and } \Xi_{k,d} \in \{0, 1\}. \tag{3.15}$$

If we regard the $\Xi_{k,d}$ as an indicator variable of the parameter $\theta_{k,d}$, then the (conditional) posterior on weight $w_{k,d}$ induced from the *Bernoulli* dropout can be interpreted as:

$$q(\Xi_{k,d} = 1 | \mathcal{D}_C) = q(w_{k,d} | \mathcal{D}_C) \simeq \textit{Bernoulli}(1 - p_{k,d}). \tag{3.16}$$

The p^C and p^T are the dropout rate in the variational posterior $q(w|\mathcal{D}_C)$ and the variational prior $q(w|\mathcal{D}_T)$. Therefore, the KL divergence between the two

[4]We found an error in the equation (7) of the manuscript, the correct description of *Bernoulli* dropout should be given as (20) (e.g., $\textit{Bernoulli}(1-p)$) in this document. Except for the error in (7) and line 185 of the manuscript, other equations or descriptions do not need to be changed.

Bernoulli distributions and can be represented as:

$$D_{\mathrm{KL}}(q(w|\mathcal{D}_C)||q(w|\mathcal{D}_T))) = \sum_{k=1}^{K}\sum_{d=1}^{D}\{(1-p_{k,d}^C)\log\frac{1-p_{k,d}^C}{1-p_{k,d}^T} + p_{k,d}^C\log\frac{p_{k,d}^C}{p_{k,d}^T}\} \tag{3.17}$$

Which is equivalent to the equation (12) in the manuscript, and proofs that the KL divergence is independent of the parameter θ.

In fact, the KL divergence between the *Bernoulli dropout posteriors* can be also approximated with the Gaussian approximation of used in (9) of the manuscript as follows:

$$D_{\mathrm{KL}}(q(w|\mathcal{D}_C)||q(w|\mathcal{D}_T))) \tag{3.18}$$

$$= \sum_{k=1}^{K}\sum_{d=1}^{D}\{\frac{p_{k,d}^C(1-p_{k,d}^C)+(p_{k,d}^T-p_{k,d}^C)^2}{2p_{k,d}^T(1-p_{k,d}^T)} + \frac{p_{k,d}^T(1-p_{k,d}^T)}{2p_{k,d}^C(1-p_{k,d}^C)} - \frac{1}{2}\} \tag{3.19}$$

The KL term is also independent of the parameter θ, so we have tested this KL term in our experiment. However, this Gaussian approximated version of the KL term was more unstable than the direct KL divergence defined in (22). Specifically, training NVDPs with (23) was also possible, but it was very sensitive to the parameter τ of the Gumbel-sigmoid trick we will introduce in the next section. On the other hand, the direct KL divergence in (22) was much more robust than (23) in terms of the training stability, performance, and sensitivity to the parameter τ.

Derivation of the KL Divergence with Hierarchical Prior With the hierarchical prior $p(w,\gamma) = p(w|\gamma)p(\gamma)$ proposed in variational bayesian dropout (VBD) [88] and the (conditional) variational posterior $q(w,\gamma|\mathcal{D}_C) = q(w|\mathcal{D}_C)q(\gamma)$

in NVDPs, we have

$$D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma|\mathcal{D}_T,\mathcal{D}_C)) = \int q(w,\gamma|\mathcal{D}_C)\log\frac{q(w,\gamma|\mathcal{D}_C)}{p(w,\gamma|x_T,y_T,\mathcal{D}_C)}dwd\gamma$$

$$= \int q(w,\gamma|\mathcal{D}_C)\log\frac{q(w,\gamma|\mathcal{D}_C)p(y_T|x_T,\mathcal{D}_C)}{p(y_T|w,x_T,\mathcal{D}_C)p(w,\gamma)}dwd\gamma \tag{3.20}$$

$$= \int q(w,\gamma|\mathcal{D}_C)\Big\{\log\frac{q(w,\gamma|\mathcal{D}_C)}{p(w,\gamma)} + \log p(y_T|x_T,\mathcal{D}_C) - \log p(y_T|w,x_T,\mathcal{D}_C)\Big\}dwd\gamma$$

$$= D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma)) + \log p(y_T|x_T,\mathcal{D}_C) - \mathbb{E}_{q(w,\gamma|\mathcal{D}_C)}[\log p(y_T|w,x_T,\mathcal{D}_C)]. \tag{3.21}$$

The step (6) is due to Bayes rule of $p(w,\gamma|\mathcal{D}_T,\mathcal{D}_C) = \frac{p(y_T|w,x_T,\mathcal{D}_C)p(w,\gamma)}{p(y_T|x_T,\mathcal{D}_C)}$ where the (hierarchical) joint prior on weight and variance $p(w,\gamma)$ is assumed to be independent of the x_T and $\mathcal{D}_C$. By reordering (7), we get

$$\log p(y_T|x_T,\mathcal{D}_C) \geq \mathbb{E}_{q(w,\gamma|\mathcal{D}_C)}[\log p(y_T|w,x_T,\mathcal{D}_C)] - D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma)) \tag{3.22}$$

$$= \mathbb{E}_{q(w|\mathcal{D}_C)}\mathbb{E}_{q(\gamma)}[\log p(y_T|w,x_T,\mathcal{D}_C)] - D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma))$$

$$= \mathbb{E}_{q(w|\mathcal{D}_C)}[\log p(y_T|w,x_T,\mathcal{D}_C)] - D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma)) \tag{3.23}$$

The lower-bound in (8) is due to the positivity of the $D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma|\mathcal{D}_T,\mathcal{D}_C))$. By replacing the $p(y_T|w,x_T,\mathcal{D}_C)$ of (9) with the (posterior predictive) likelihood of CNPs, we get

$$\log p(y_T|x_T,\mathcal{D}_C) \geq \mathbb{E}_{q_\phi(w|\mathcal{D}_C)}[\log\mathcal{N}(y_T|g_w(x_T,r),\sigma^2)] - D_{\mathrm{KL}}(q_\phi(w,\gamma|\mathcal{D}_C)||p(w,\gamma)). \tag{3.24}$$

which is equivalent to the ELBO objective of NVDPs with hierarchical prior described in the manuscript.

KL divergence. The KL divergence with hierarchical prior in (10) can be

further decomposed as:

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma)) = \int q(w,\gamma|\mathcal{D}_C)\log\frac{q(w,\gamma|\mathcal{D}_C)}{p(w,\gamma)}dwd\gamma \\
&= \int q(w|\mathcal{D}_C)q(\gamma)\log\frac{q(w|\mathcal{D}_C)q(\gamma)}{p(w|\gamma)p(\gamma)}dwdr \\
&= \int q(w|\mathcal{D}_C)q(\gamma)\log\frac{q(w|\mathcal{D}_C)}{p(w|\gamma)}dwd\gamma + \int q(w|\mathcal{D}_C)q(\gamma)\log\frac{q(\gamma)}{p(\gamma)}dwd\gamma \\
&= D_{\mathrm{KL}}(q(w|\mathcal{D}_C)||p(w|\gamma)) + D_{\mathrm{KL}}(q(\gamma)||p(\gamma)).
\end{aligned} \tag{3.25}
$$

Since both the (conditional) variational posterior $q(w_{k,d}|\mathcal{D}_C)$ and the hierarchical prior $p(w_{k,d}|\gamma)$ follows factorized Gaussian distribution as described in the manuscript, the first KL divergence term in (11) can be written as:

$$
\begin{aligned}
&D_{\mathrm{KL}}(q(w_{k,d}|\mathcal{D}_C)||p(w_{k,d}|\gamma_{k,d})) \\
&= 0.5\log(\frac{\gamma_{k,d}}{p_{k,d}(1-p_{k,d})\theta_{k,d}^2}) + \frac{p_{k,d}(1-p_{k,d})\theta^2 + (1-p)^2\theta_{k,d}^2}{2\gamma_{k,d}} - 0.5
\end{aligned} \tag{3.26}
$$

To find the optimal $\gamma_{k,d}$, referred as $\gamma_{k,d}^*$ in the optimization of (10), we can take a partial differential of (12) with respect to $\gamma_{k,d}$ to zero. Then, we have:

$$
\gamma_{k,d}^* = p_{k,d}(1-p_{k,d})\theta_{k,d}^2 + (1-p_{k,d})^2\theta_{k,d}^2 \tag{3.27}
$$

Replacing $r_{k,d}$ in (12) with $r_{k,d}^*$ in (13), we get:

$$
0.5\log(\frac{p_{k,d}(1-p_{k,d})\theta_{k,d}^2 + (1-p_{k,d})^2\theta_{k,d}^2}{p_{k,d}(1-p_{k,d})\theta_{k,d}^2}) \tag{3.28}
$$

$$
+ \frac{\cancel{p_{k,d}(1-p_{k,d})\theta^2} + (1-p)^2\theta_{k,d}^2}{2p_{k,d}(1-p_{k,d})\theta_{k,d}^2 + (1-p_{k,d})^2\theta_{k,d}^2} - 0.5
$$

$$
= 0.5\log(1+\frac{(1-p_{k,d})}{p_{k,d}}) = -0.5\log(p_{k,d}) \tag{3.29}
$$

In addition, with the mean-field approximation of $q(\gamma) = \prod_{k=1}^{K}\prod_{d=1}^{D} q(\gamma_{k,d})$ where $q(\gamma_{k,d})$ obeys a delta distribution and the uniform prior $q(\gamma_{k,d}) =$

$\prod_{k=1}^{K}\prod_{d=1}^{D}\mathcal{U}(\gamma_{k,d}|a,b)$ described in the manuscript, we can obtain the second KL terms in (11) as:

$$\begin{aligned} D_{\mathrm{KL}}(q(\gamma_{k,d})||p(\gamma_{k,d})) &= \int q(\gamma_{k,d})\log\frac{q(\gamma_{k,d})}{p(\gamma_{k,d})}d\gamma_{k,d} \\ &= \int q(\gamma_{k,d})\log q(\gamma_{k,d})d\gamma_{k,d} - \int q(\gamma_{k,d})\log p(\gamma_{k,d}) \\ &= -\mathcal{H}(\gamma_{k,d}) - \int q(\gamma_{k,d})\log p(\gamma_{k,d}), \end{aligned} \tag{3.30}$$

where the entropy of a random variable $\gamma_{k,d}$ is zero (e.g., $\mathcal{H}(\gamma_{k,d})=0$) because the delta distribution $q(\gamma_{k,d})$ do not provide any uncertainty [2]. As a result, we obtain:

$$D_{\mathrm{KL}}(q(\gamma_{k,d})||p(\gamma_{k,d})) = -\int q(\gamma_{k,d})\log p(\gamma_{k,d})d\gamma_{k,d} \tag{3.31}$$

To simplify the problem, assuming the dimension of $r_{k,d}$ is 1, e.g, $p(r_{k,d}) = 1/(b-a)$, thus we have:

$$\begin{aligned} &D_{\mathrm{KL}}(q(\gamma_{k,d})||p(\gamma_{k,d})) = -\int q(\gamma_{k,d})\log p(\gamma_{k,d})d\gamma_{k,d} \\ &= -\int_{-\infty}^{a} q(\gamma_{k,d})\log p(\gamma_{k,d}) - \int_{a}^{b} q(\gamma_{k,d})\log p(\gamma_{k,d}) - \int_{b}^{+\infty} q(\gamma_{k,d})\log p(\gamma_{k,d}) \\ &= -\log\beta\int_{-\infty}^{a} q(\gamma_{k,d}) - \log(1/(b-a))\int_{a}^{b} q(\gamma_{k,d}) - \log\beta\int_{b}^{+\infty} q(\gamma_{k,d}), \end{aligned} \tag{3.32}$$

Here, β is an infinitesimally small value, effectively zero. It represents the negligible probability of $\gamma_{k,d}$ being outside the interval [a, b]. Hence, if $q(\gamma_{k,d})$, modeled as a delta function, falls within the interval [a, b], the outer two integrals are zero, and the KL divergence is simply the log of the interval length (b - a). If the delta function lies outside this interval, the KL divergence is effectively infinite. However, to avoid such infinite KL divergence, it's common to consider [a, b] as a sufficiently large interval such that $q(\gamma_{k,d})$ falls within it. Thus, the KL divergence is treated as a constant $\log(b-a)$ and can be

disregarded in calculations. For a more detailed theoretical derivation of this part please refer to [88].

Hence, with all the aforementioned assumptions and equations, we get

$$D_{\mathrm{KL}}(q(w,\gamma|\mathcal{D}_C)||p(w,\gamma)) = D_{\mathrm{KL}}(q(w|\mathcal{D}_C)||p(w|\gamma)) + D_{\mathrm{KL}}(q(\gamma)||p(\gamma))$$
$$= \sum_{k=1}^{K}\sum_{d=1}^{D}\{-0.5\log(p_{k,d}) + \log(b-a)\} \tag{3.33}$$

(18) is equivalent to the KL divergence term defined in (11) of the manuscript. The KL divergence between amortized PoB dropout posterior with hierarchical prior is independent of the parameter θ, thus we can employ it to perform VI.

Advantages of the hierarchical prior. This brings two aspects of advantages. Firstly, two kinds of very simple distributions in hierarchical structure can produce much more complicated distribution, e.g., a hierarchical sparse prior [36], a zero-mean Gaussian distribution with variance depicted by a gamma distribution, the student-t prior, and the super-Gaussian scale mixture model. Thus the two-level structure increases the possible solution spaces for the proper and feasible prior to interpreting variational dropout. Secondly, the hierarchical structure enables the two-level prior separable in the involved Bayesian inference and thus is possible to simplify the Bayesian inference or make the intractable inference tractable.

Chapter 4

Application in Few-shot Learning

4.1 Introduction

In this section, We have evaluated NVDPs compared with other methods on various few-shot learning tasks and datasets. In the experiment, we first compared the NVDP with Neural Process (NP) family [33, 34, 55] on the few-shot 1D regression task with the Gaussian Process (GP) dataset. In the GP dataset, we additionally performed active learning experiment to see the model's fast adaptation ability. Then, the few-shot image completion task is performed on the MNIST and CelebA dataset. To see the generalization performance of models on completely new dataset, we also tested the models trained with MNIST on the Omniglot dataset in the image completion task. Finally, we tested the NVDP on the standard few-shot classification tasks such as the Omniglot and MiniImagenet dataset with other baseline such as VERSA [32], CNP [34], MAML [25], and others [12, 13]. The experiments show that NVDPs can circumvent the under-fitting and posterior collapsing and achieve outstanding

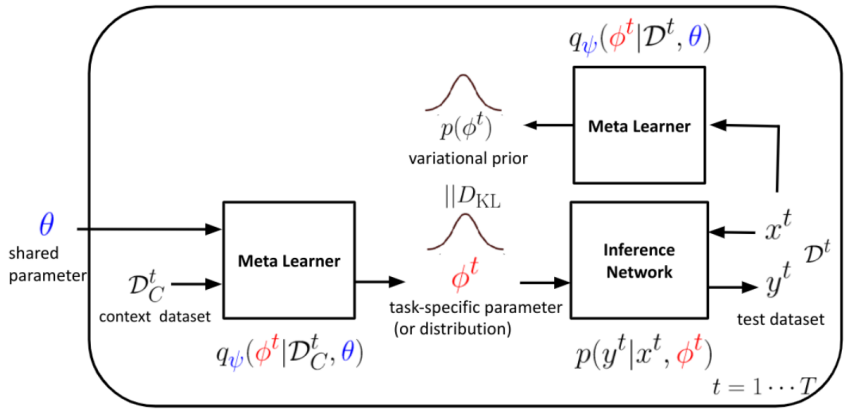


Figure 4.1: An overview of variational prior (or shared prior) in NVDP.

performances.

4.2 Neural Variational Dropout Processes for Few-shot Learning

With the conditional dropout posterior defined in equation 5.3 and the KL regularization term in equation 3.5, we can now fully describe the ELBO objective of NVDPs for the multi-task dataset as:

$$\sum_{t=1}^{T} \log p(\mathcal{D}^t) \geq \sum_{t=1}^{T} \{\mathbb{E}_{q(\phi^t|\mathcal{D}_C^t;\theta)}[\log p(\mathbf{y}^t|\mathbf{x}^t, \phi^t)] - \mathrm{KL}(q(\phi^t|\mathcal{D}_C^t;\theta)||q(\phi^t|\mathcal{D}^t;\theta))\}. \tag{4.1}$$

The goal is to maximize the ELBO equation 5.4 w.r.t. the variational parameters (*i.e.* $\theta, \psi, \omega,$ and τ) of the posterior $q(\phi_t|\mathcal{D}_C^t;\theta)$ defined in equation 5.3. The optimization of these parameters is done by using the *stochastic gradient variational Bayes* (SGVB) [95, 120]. The basic trick in the SGVB is to parameterize the random weights $\phi^t \sim q(\phi^t|\mathcal{D}^t)$ using a deterministic differentiable transformation $\phi^t = f(\epsilon, \mathcal{D}_C^t)$ with a non-parametric i.i.d. noise $\epsilon \sim p(\epsilon)$. Then, an unbiased differentiable minibatch-based Monte Carlo estimator, $\hat{\mathcal{L}}(\theta, \psi, \omega, \tau)$,

of the ELBO of NVDPs can be defined as:

$$\frac{1}{T'}\sum_{t=1}^{T'}\{\frac{1}{M}\sum_{i=1}^{M}\log p(\mathbf{y}_i^t|\mathbf{x}_i^t, f(\epsilon, \mathcal{D}_C^t)) - (\frac{\mathbf{P}^t(1-\mathbf{P}^t)+(\widehat{\mathbf{P}}^t-\mathbf{P}^t)^2}{2\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)} + \frac{1}{2}\log\frac{\widehat{\mathbf{P}}^t(1-\widehat{\mathbf{P}}^t)}{\mathbf{P}^t(1-\mathbf{P}^t)})\} \tag{4.2}$$

T' and M are the sizes of randomly sampled mini-batch of tasks and data points, respectively, per each epoch (*i.e.*,$\{\mathbf{x}_i^t, \mathbf{y}_i^t\}_{i=1}^M = \mathcal{D}^t \sim \{\mathcal{D}^t\}_{t=1}^{T'} \sim \mathcal{D}$). $\mathcal{D}_C^t$ is the subset set of $\mathcal{D}^t$ discussed in section 3.2 and 3.3. The transformation of the task-specific weights is given as $\phi_{k,d}^t = f(\epsilon_{k,d}, \mathcal{D}_C^t) = (1-\mathbf{P}_{k,d}^t)\theta_{k,d} + \sqrt{\mathbf{P}_{k,d}^t(1-\mathbf{P}_{k,d}^t)}\theta_{k,d}\epsilon_{k,d}$ with $\epsilon_{k,d} \sim N(0,1)$ from the equation 5.3. The intermediate weight $\phi_{k,d}^t$ is now differentiable with respect to $\theta_{k,d}$ and $\mathbf{P}_{k,d}^t$. Also, $\mathbf{P}_{k,d}^t$ (and $\widehat{\mathbf{P}}_{k,d}^t$) is deterministically computed by the meta-model parameterized by ψ, ω, and τ as in equation 3.4. Thus, the estimator $\hat{\mathcal{L}}(\theta, \psi, \omega, \tau)$ in equation 4.2 is differentiable with respect to all the variational parameters and can be optimized via the SGD algorithm. In practice, a local reparameterization trick[1] is further utilized to reduce the variance of gradient estimators on training [83].

Few-shot Posterior Adaptation. The learned variational model in equation 5.4 can be later leveraged for the agent model's few-shot adaptation on a new unseen task $\mathcal{D}^*$. For example, given a few observed examples $\mathcal{D}_s^* = \{\mathbf{x}_i^*, \mathbf{y}_i^*\}_{i=1}^S$ from the new task, a posterior predictive distribution for the unknown target y^* conditioned on the newly observed input $x^* \sim \mathcal{D}^*$ can be approximated with the Monte Carlo (MC) approach:

$$p(y^*|x^*, \mathcal{D}^*) \approx \frac{1}{C}\sum_{i=1}^{C} p_{\phi_{(i)}^*}(y^*|x^*), \tag{4.3}$$

[1] We can sample pre-activations $B_{m,d}$ for a mini-batch of size M directly using inputs $A_{m,k}$ ($B_{m,d} \sim \mathcal{N}(B_{m,d}|\sum_{k=1}^K A_{m,k}(1-\mathbf{P}_{k,d}^t)\theta_{k,d}, \sum_{k=1}^K A_{m,k}^2\mathbf{P}_{k,d}^t(1-\mathbf{P}_{k,d}^t)\theta_{k,d}^2)$).

where $\phi^*_{(i)} \sim q(\phi^*|\mathcal{D}^*_s;\theta)$. This enables approximation and uncertainty approximation for newly observed data.

4.3 Experiments

Metrics in regression. In the evaluation of the conditional NN models' adaptation, the newly observed task data $\mathcal{D}^*$ is split into the input-output pairs of the *context set* $\mathcal{D}^*_C = \{x_i, y_i\}_{i=1}^S$ and the *target set* $\mathcal{D}^*_T = \{x_i, y_i\}_{i=1}^N$ ($\mathcal{D}^*_C \not\subseteq \mathcal{D}^*_T$ in evaluation [33, 32]).

1. The log-likelihood (LL), $\frac{1}{N+S}\sum_{i\in\mathcal{D}^*_C\cup\mathcal{D}^*_T}\mathbb{E}_{q(\phi^*|\mathcal{D}^*_C)}[\log p(y_i|x_i,\phi^*)]$, measures the performance of the NN model over the whole dataset $\mathcal{D}^*$ conditioned on the *context set*.

2. The reconstructive log-likelihood (RLL), $\frac{1}{S}\sum_{i\in\mathcal{D}^*_C}\mathbb{E}_{q(\phi^*|\mathcal{D}^*_C)}[\log p(y_i|x_i,\phi^*)]$, measures how well the model reconstructs the data points in the *context set*. A low RLL is a sign of under-fitting.

3. The predictive log-likelihood (PLL), $\frac{1}{N}\sum_{i\in\mathcal{D}^*_T}\mathbb{E}_{q(\phi^*|\mathcal{D}^*_C)}[\log p(y_i|x_i,\phi^*)]$, measures the prediction on the data points in the *target set* (not in the *context set*). A low PLL is a sign of over-fitting.

4.3.1 1D few-shot Regression with GP samples.

The 1D regression task is to predict unknown functions given some observed context points; each function (or data point) is generated from a Gaussian Process (GP) with the random kernel, then the standard data split procedure is performed (i.e., $S \sim U(3, 97)$ and $N \sim U[S + 1, 100)$) at train time. For the baseline models, the Neural Process model (NP) and NP with additional deterministic representation (NP+CNP) described in [33, 34, 55] is compared. We

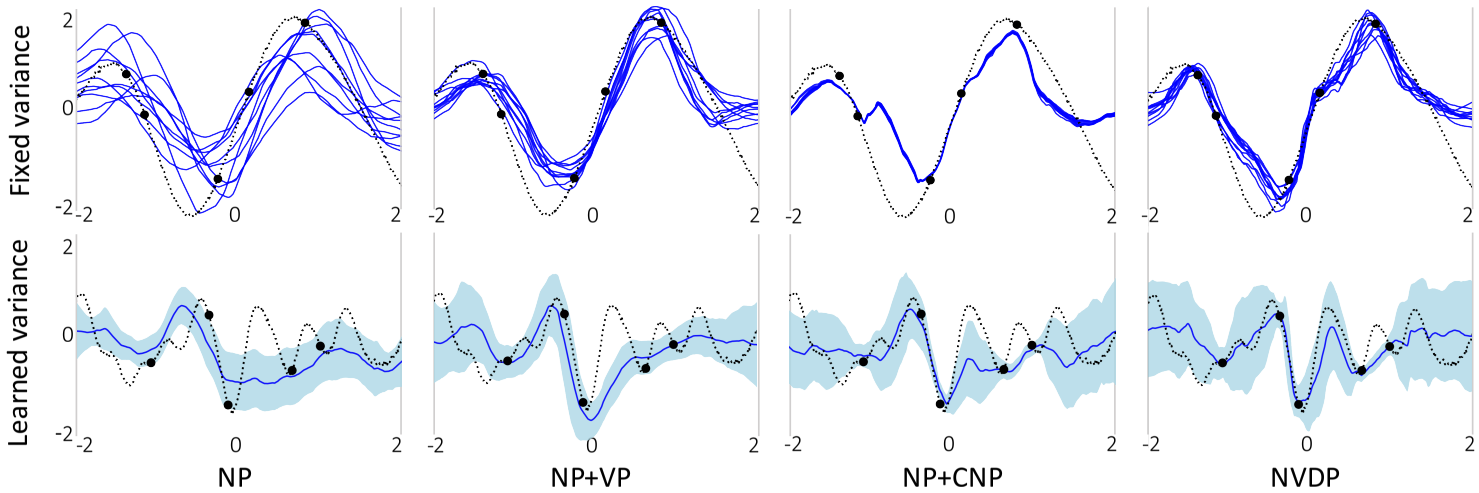


Figure 4.2: The 1D few-shot regression results of the models on GP dataset in *fixed variance* and *learned variance* settings. The black (dash-line) represents the true unknown task function. Black dots are a few context points ($S = 5$) given to the posteriors. The blue lines (and light blue area in *learned variance* settings) are mean values (and variance) predicted from the sampled NNs.

GP Dataset		NP	NP+VP	NP+CNP	NP+CNP+VP	NVDP
Fixed	LL	$-0.98(\pm0.08)$	$-0.96(\pm0.06)$	$-0.95(\pm0.05)$	$-0.94(\pm0.05)$	$\mathbf{-0.94(\pm0.04)}$
Variance	RLL	$-0.97(\pm0.06)$	$-0.95(\pm0.04)$	$-0.93(\pm0.02)$	$-0.93(\pm0.02)$	$\mathbf{-0.93(\pm0.01)}$
	PLL	$-0.98(\pm0.08)$	$-0.97(\pm0.06)$	$-0.95(\pm0.05)$	$-0.94(\pm0.05)$	$\mathbf{-0.94(\pm0.05)}$
Learned	LL	$0.19(\pm1.87)$	$0.48(\pm0.77)$	$0.70(\pm0.89)$	$0.70(\pm0.73)$	$\mathbf{0.83(\pm0.61)}$
Variance	RLL	$0.72(\pm0.57)$	$0.71(\pm0.55)$	$1.03(\pm0.39)$	$1.00(\pm0.41)$	$\mathbf{1.10(\pm0.34)}$
	PLL	$0.16(\pm1.90)$	$0.46(\pm0.78)$	$0.66(\pm0.92)$	$0.68(\pm0.75)$	$\mathbf{0.81(\pm0.62)}$

Table 4.1: The validation result of the 1D regression models on the GP dataset in *fixed variance* and *learned variance* settings. The higher LLs are the better. The models of NP, NP with variational prior (NP+VP), NP with deterministic path (NP+CNP), NP+CNP+VP and NVDP (ours) are compared.

also adopted the variational prior (VP) into the representation-based posterior of NP and its variants (NP+VP and NP+CNP+VP). For all models (including our NVDP), we used the same *fixed variance* and *learned variance* likelihood architecture depicted in [55]: the agent NN with 4 hidden layers of 128 units with LeLU activation [57] and an output layer of 1 unit for the mean (or an additional 1 unit for variance). The dimensions of the set representation r^t were fixed to 128. The meta NNs in the conditional *dropout* posterior in NVDP has 4 hidden layers of 128 units with LeakyReLU and an output layer of 257 units (i.e. $K+D+1$) for each layer of the agent model. All models were trained with Adam optimizer [121] with learning rate 5e-4 and 16 task-batches for 0.5 million iterations. On validation, 50000 random tasks (or functions) were sampled from the GP function generator and the split data of $S \sim U(3, 97)$ and $N = 400 - S$ were used to compute the log-likelihood (LL) and other evaluation metrics.

Table 1 summarizes the validation results of 1D regression with the GP dataset. NVDP achieves the best LL scores compared with all other baselines in both the *fixed* and *learned variance* likelihood model settings. The NVDPs record high RLL on the observed data points and excellent PLL scores on the unseen function space in the new task; this indicates that the proposed conditional *dropout* posterior approach can simultaneously mitigate the under-fitting and over-fitting of the agent model compared with the other baselines. When VP is applied to NP or NP+CNP, the PLL scores tend to increase by meaningful margins in all cases. This demonstrates that the proposed variational prior (VP) approach can also reduce the over-fitting of latent representation-based conditional posterior. Figure 2 visualizes the few-shot 1D function regression results in both model settings. In the *fixed variance* setting, the functions sampled from NPs show high variability but cannot fit the context data well. NP+CNP can fit the context data well but loses *epistemic* uncertainty due to

the collapsing of conditional posterior. On the other hand, the functions from NVDPs show a different behavior; they capture the function's variability well while also fitting the observed context points better. NVDPs also approximate the mean and variance of unknown functions well in *learned variance* setting.

4.3.2 Active Learning with Regression.

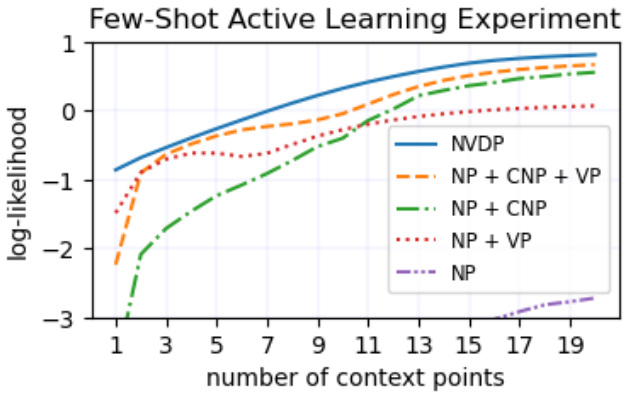


Figure 4.3: Active learning performance on regression after up to 19 selected data points. NVDPs can use its uncertainty estimation to quickly improve LLs, while other models are learning slowly.

To further compare the uncertainty modeling accuracy, we performed an additional active learning experiment on the GP dataset described above. The goal in active learning is to improve the log-likelihood of models with a minimal number of context points. To this end, each model chooses additional data points sequentially; the points with maximal variance across the sampled regressors were selected at each step in our experiment. The initial data point is randomly sampled within the input domain, followed by 19 additional points that are selected according to the variance estimates. As seen in Figure 3, NVDPs outperform the others due to their accurate variance estimation, especially with a small number of additional points, and show steady improvement

with less over-fitting.

4.3.3 Few-shot Image Completion Tasks

The image completion tasks are performed to validate the performance of the models in the more complex function spaces [33, 34, 55]. Here, we treat the image samples from MNIST [122] and CelebA [123] as unknown functions. The task is to predict a mapping from normalized 2D pixel coordinates x_i ($\in [0, 1]^2$) to pixel intensities y_i ($\in \mathbb{R}^1$ for greyscale, $\in \mathbb{R}^3$ for RGB) given some context points. We used the same *learned variance* baselines implemented in the GP data regression task (except the 2D input and 3D output for the agent NN and $r^t = 1024$ were used for CelebA). At each iteration, the images in the training set are split into S context and N target points (e.g., $S \sim U(3, 197)$, $n \sim U[N + 1, 200)$ at train and $S \sim U(3, 197)$, $N = 784 - S$ at validation). Adam optimizer with a learning rate 4e-4 and 16 task batches with 300 epochs were used for training. The validation is performed on the separated validation set. To see the generalization performance on a completely new dataset, we also tested the models trained on MNIST to the Omniglot validation set.

Table 2 summarizes the validation results of image completion tasks. NVDPs achieve the outperforming LLs compared with all other baselines. Interestingly, NVDPs (trained on the MNIST dataset) also achieve the best results on the Omniglot dataset. Figure 4 shows the image reconstruction results with a varying number of random context pixels. NP generated various image samples when the number of contexts was small (e.g., $m \leq 30$ or half), but those samples could not approximate the true unknown images well compared with the other models. The NP+CNP achieves crisp reconstruction results compared with NP and also shows a good generalization performance on the Omniglot dataset, but the sampled images (or functions) from NP+CNP had almost no variability

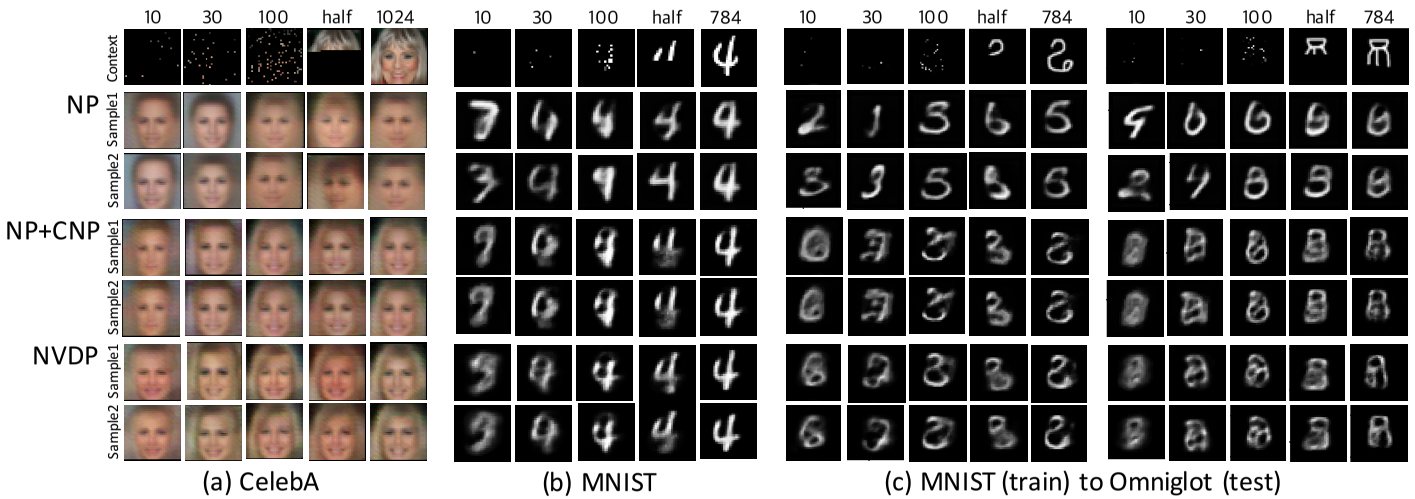


Figure 4.4: The results from the 2D image completion tasks on CelebA, MNIST, and Omniglot dataset. Given the observed context points (10, 30, 100, half, and full pixels), the mean values of two independently sampled functions from the models (i.e. NP, NP+CNP, and NVDP (ours)) are presented.

Image Dataset		NP	NP+VP	NP+CNP	NP+CNP+VP	NVDP
MNIST	LL	0.54(±0.51)	0.76(±0.14)	0.83(±0.21)	0.86(±0.16)	**0.90**(±**0.16**)
	RLL	0.94(±0.18)	0.90(±0.10)	1.12(±0.09)	1.12(±0.09)	**1.15**(±**0.05**)
	PLL	0.51(±0.51)	0.75(±0.14)	0.80(±0.20)	0.83(±0.15)	**0.88**(±**0.15**)
MNIST (*train*)	LL	0.35(±0.29)	0.56(±0.10)	0.64(±0.17)	0.68(±0.13)	**0.70**(±**0.13**)
to	RLL	0.72(±0.19)	0.73(±0.12)	0.95(±0.09)	0.98(±0.09)	**0.99**(±**0.07**)
Omniglot (*test*)	PLL	0.32(±0.28)	0.54(±0.11)	0.60(±0.16)	0.64(±0.13)	**0.66**(±**0.12**)
CelebA	LL	0.51(±0.27)	0.60(±0.12)	0.76(±0.15)	0.77(±0.15)	**0.83**(±**0.15**)
	RLL	0.63(±0.11)	0.68(±0.06)	0.91(±0.05)	0.92(±0.05)	**0.99**(±**0.04**)
	PLL	0.50(±0.27)	0.59(±0.12)	0.75(±0.15)	0.76(±0.15)	**0.82**(±**0.15**)

Table 4.2: The summary of 2D image completion tasks on the MNIST, CelebA, and Omniglot datasets.

due to its posterior collapsing behavior. On the other hand, the samples from NVDPs exhibit comparable reconstruction results while also showing a reasonable amount of variability. In addition, NVDPs also present an outstanding generalization performance on the unseen Omniglot dataset. This implies that NVDPs not only fit better in the complex function regression but can also capture more general knowledge that can be applied to new unseen tasks.

4.3.4 Few-shot Classification Tasks.

NVDPs can also be successfully applied for few-shot classification tasks; we have tested NVDPs on standard benchmark datasets such as Omniglot [8] and MiniImagenet [11] with other baselines: VERSA [32], CNP [34], Matching Nets [12], Prototypical Nets [13], MAML [25], Meta-SGD [124] and Meta-dropout [30]. For the classifier, we used one-layer NNs with hidden units of 512. For the meta-model, we used two-layer NNs with hidden units of 256 similar to VERSA's conditional posterior model. For an image input, the NN classifier outputs one logit value per each class; class-specific dropout rates for the NN classifier are computed with the image features $r^t \in \mathbb{R}^{256}$ aggregated by the same class in the few-shot context examples. The same deep (CONV5) feature extractor architecture is used as in VERSA [32]. For each class, 1 or 5 few-shot context samples (i.e., labeled images) are given. Among 5 or 20 classes, only the logit value related to the true label is maximized. We use the same batch sizes, learning rate, and epoch settings depicted in VERSA [32]. Table 3 summarizes the results. NVDPs achieve higher predictive accuracy than the model-based meta-learning approaches CNP and VERSA and are comparable with the state-of-the-art optimization-based meta-learning approaches such as MAML or Meta-Dropout on the Omniglot dataset. NVDPs also record good classification accuracy in the MiniImageNet dataset.

	Omniglot dataset				*MiniImageNet* dataset	
	5-way accuracy (%)		**20-way accuracy (%)**		**5-way accuracy (%)**	
Method	1-shot	5-shot	1-shot	5-shot	1-shot	5-shot
Matching Nets	98.1	98.9	93.8	98.5	46.6	60.0
Prototypical Nets	97.4	99.3	95.4	98.7	46.61 ± 0.78	65.77 ± 0.70
CNP	95.3	98.5	89.9	96.8	48.05 ± 2.85	62.71 ± 0.58
Meta-SGD (MSGD)	-	-	96.16 ± 0.14	98.54 ± 0.07	48.30 ± 0.64	65.55 ± 0.56
MSGD + Meta-dropout	-	-	97.02 ± 0.13	$\mathbf{99.05 \pm 0.05}$	50.87 ± 0.63	65.55 ± 0.57
MAML	98.70 ± 0.40	$\mathbf{99.90 \pm 0.10}$	95.80 ± 0.63	98.90 ± 0.20	48.70 ± 1.84	63.11 ± 0.92
VERSA	99.70 ± 0.20	99.75 ± 0.13	97.66 ± 0.29	98.77 ± 0.18	53.40 ± 1.82	67.37 ± 0.86
NVDP	$\mathbf{99.70 \pm 0.12}$	99.86 ± 0.28	$\mathbf{97.98 \pm 0.22}$	98.99 ± 0.22	$\mathbf{54.06 \pm 1.86}$	$\mathbf{68.12 \pm 1.04}$

Table 4.3: Few-shot classification results on Omniglot and MiniImageNet dataset. The baselines are Matching Nets, Prototypical Nets, MAML, Meta-SGD, Meta-dropout, CNP, VERSA, and NVDP (our). Each value corresponds to the classification accuracy (%) (and *std*) on the validation set.

4.3.5 Experiments on a Trigonometry Dataset

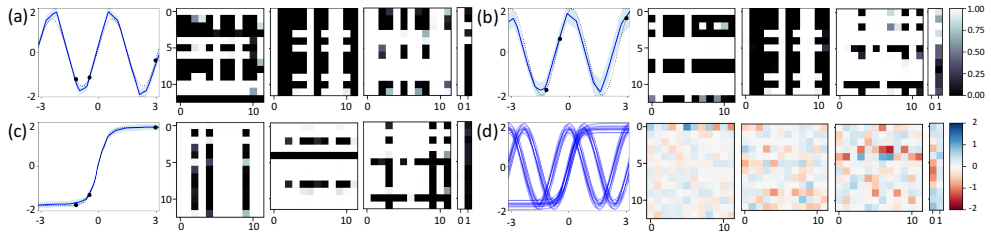


Figure 4.5: (a) sine, (b) cosine, (c) tanh function (left) with the probabilities of using parameters (1-$\mathbf{P}^t$) (right) predicted with small NVDPs (13-12-12-2) conditioned on 4-shot context points (black dots), and (d) the trigonometry dataset (left) and the deterministic shared NN parameters θ (right).

Setup. To further investigate how the proposed meta-model utilizes the common structures of the NN parameters, we trained a small NVDP model (of the size of 13-12-12-2) on a mixture of scaled trigonometric functions: x is sampled in a range of $[-\pi, \pi]$, and y is determined by $y = a * f(2 * x - b * \pi)$ where f is one of three functions sine, cosine, and tanh with the probability of one third, and $a \sim \mathcal{U}(1.5, 2)$ and $b \sim \mathcal{U}(-0.1, 0.1)$. We used the training procedures in the next section except the simple model architecture and learning rate $5e-4$.

Results. Figure 4.5 displays the trained NVDP on the trigonometry function dataset. The NVDP could capture the probability of dropout for each parameter θ of the agent, successfully predicting the task-specific trigonometric functions conditioned on the small ($S = 4$) context set. This shows that the task-specific dropout rates can transform a single conventional NN agent to express multiple functions. It is interesting to see that the contexts from sine and cosine functions yield similar dropout rates for the second layer. On the other hand, the contexts from the tanh function result in different dropout structures for all layers.

Figure 6 displays the result of applying the relaxed sigmoid trick with varying τ discussed in section 4 in this document. BNP (with hierarchical prior) tends to employ more sparse parameters than BNP+ (with variational prior).

4.4 Conclusion

This study presents a new model-based Bayesian meta-learning approach, Neural Variational Dropout Processes (NVDPs). A novel conditional *dropout* posterior is induced from a meta-model that predicts the task-specific dropout rates of each NN parameter conditioned on the observed context. This paper also introduces a new type of *variational prior* for optimizing the conditional posterior in the amortized variational inference. We have evaluated the proposed method compared with the existing approaches in various few-shot learning tasks, including 1D regression, image inpainting, and classification tasks. The experimental results demonstrate that NVDPs simultaneously improved the model's adaptation to the context data, functional variability, and generalization to new tasks. The proposed *variational prior* could also improve the variability of the representation-based posterior model [33].

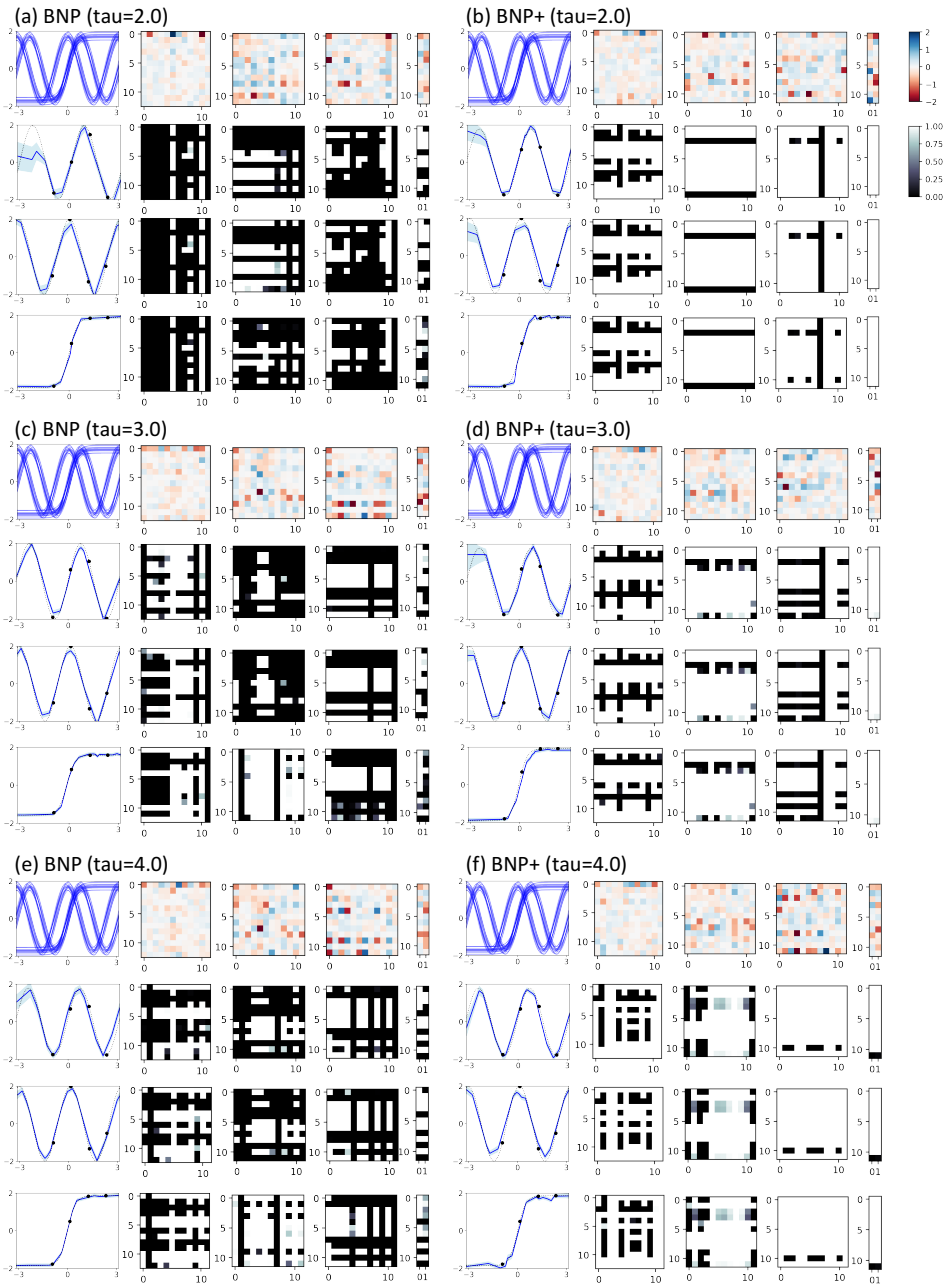


Figure 4.6: Exhibiting the result of experiment with the *relaxed sigmoid trick* (without Gumbel noise) mentioned in the section 4 in this document. The descriptions are similar to Figure 1.

Chapter 5

Application in Federated Learning

5.1 Introduction

Federated learning (FL) aims at training a global model from distributed clients without sharing or collecting their sensitive raw data. Thanks to its privacy-preserving aspect of FL [125], it is increasingly popular to be applied to various applications such as image classification [126, 127], object detection [128, 129], keyboard suggestion [130, 131], recommendation [132, 133], and healthcare [134, 135]. Conventional FL methods are effective with convergence guarantee when the data from different clients are independently and identically (i.i.d.) distributed [136, 137]. However, due to the differences in preferences, locations, and usage habits of clients, the private data in FL are usually non-i.i.d. When the data distributions of the clients vary, the local model learned from each client can diverge, and thus learning an optimal global model could fail [138, 139]. Furthermore, the client data scale may not be large enough to train a local model with many parameters, causing model overfitting and poor gen-

eralization [140, 141].

To overcome the challenge caused by non-i.i.d. data, personalized federated learning (PFL) has been studied [138, 139]. Each client is allowed to have its own personalized model trained on its local data while still participating in the global model training. There are many branches of PFL, such as those based on multitask learning [142, 143], meta-learning [144, 145, 146, 147, 46], and transfer learning [148, 149, 150]. Although these approaches enhance training convergence in non-i.i.d. data settings, they may still experience model overfitting with limited client data. Recently, the Bayesian learning paradigm was introduced to the FL to tackle overfitting by considering the uncertainty of the model parameters [151, 152, 153]. However, they could also struggle with diverging local models if the data from different clients exhibit significant statistical variability. Motivated by these challenges, our objective is to simultaneously address the issues of FL with the limited and non-i.i.d. client data.

5.2 Background

A standard approach to FL (*e.g.,* FedAvg [125]) iterates between the local training on the client devices and global optimization at the server. Assuming M clients each of which has a data set $\mathcal{D}^m = \{(x_i^m, y_i^m)\}_{i=1}^{|\mathcal{D}^m|}$ the FL problem can be formulated as follows:

$$\text{Server: } \min_{w} \ \mathcal{J}(w) = \sum_{m=1}^{M} g^m \mathcal{J}^m(w), \quad \text{Client: } \mathcal{J}^m(w) = \frac{1}{|\mathcal{D}^m|} \sum_{i} \ell(x_i^m, y_i^m; w). \tag{5.1}$$

w indicates the model parameter, and the global learning objective $\mathcal{J}(w)$ at the server is a weighted average of local objectives $\mathcal{J}^m(w)$ across M clients. The weight g^m is proportional to the size of the local dataset (*e.g.* $|\mathcal{D}^m|/|\mathcal{D}|$). The local loss is usually defined as the empirical negative log-likelihood on the m-th

client's dataset D^m (*i.e.* $\ell(\mathcal{D}^m; w) = -\log p(y^m|x^m, w)$).

The local training is carried out in parallel fully (or partly) in each client device, with multiple SGD epochs to update the local weight w^m. Then, the aggregation step computes the global weight (*e.g.* $\bar{w} \leftarrow \sum_{m=1}^{M} g^m w^m$) in the server by taking the weighted average of the local weights. The global weight $\bar{w}$ is subsequently set as the initial weight for each client in the next local training round. FL aims to train models on large distributed datasets by only exchanging model parameters (e.g., $\bar{w}$ and w^m) between server and local devices, thereby minimizing privacy leakage of clients' datasets.

Challenges in FL. There are many challenges for real-world FL applications. (1) *Heterogeneity of client data.* The original FL algorithm guarantees convergence well when clients' data are i.i.d. However, the data distributions of clients often have different characteristics (*e.g.* classes or tasks follow non-i.i.d.), w^m would drift away from each other, causing the $\bar{w}$ suboptimal [138, 139]. (2) *Sparse Connectivity.* In practice, the total number of clients M can be extremely large, while the communication between the server and clients may be spotty or unreliable. This creates a challenge of reliable and consistent training with a small subset of participating clients in each communication round. (3) *Poor generalization due to limited data.* When the training data available on each local device are few, the local model can easily overfit, resulting in poor generalization on novel clients [140, 141]. (4) *Communication cost.* FL optimization requires frequent communication of exchanging model parameters between local devices and the central server. This process is slow and could introduce additional privacy concerns. Thus, reducing the model size might be important as well.

Bayesian FL. While conventional FL methods employ a point estimate of weight as in 5.1, recent works [151, 152, 153] have incorporated probability dis-

Method	Personalization	Uncertainty	Compression
FedAvg [125]	✗	✗	✗
FedBE [151]	✗	✓	✗
FedPA [153]	✗	✓	✗
pFedHN [46]	✓	✗	✗
Reptile [145]	✓	✗	✗
PerFedAvg [146]	✓	✗	✗
MetaVD (ours)	✓	✓	✓

Table 5.1: Comparison of MetaVD with other PFL methods in terms of personalization, uncertainty, and compression ability.

tributions over the weights. Based on FedAvg, a Gaussian distribution is used to represent each parameter [152], and a posterior aggregation strategy is proposed using the MCMC technique [153]. FedBE [151] uses a Bayesian ensemble global model with Gaussian or Dirichlet local distributions. In these Bayesian FL approach, the client device first estimates local posterior using their local data, and then the server aggregates the partially updated local posteriors into a global posterior. This strategy improves prediction confidence and model convergence. However, probabilistic modeling typically requires additional parameters approximating its density, which may increase communication costs in FL. FedAG [152] and FedBE [151] use Gaussian global posterior distributions but only consider point estimates of local models. FedPA [153] assumes Gaussian posteriors both globally and locally but only maintains the global posterior mean on the server, which makes it challenging to track local uncertainties. Moreover, previous Bayesian FL methods are not developed for model personalization, and the performance can be degraded in non-i.i.d. client data.

5.3 Extended MetaVD Approach for Few-shot Learning

We propose a new Bayesian PFL approach, called Meta-Variational Dropout (MetaVD), which can simultaneously manage the local posterior personalization and model regularization in FL with the limited non-i.i.d. datasets. MetaVD is also a flexible approach that is compatible with conventional FL algorithms.

5.3.1 Variational Inference for FL

Instead of MCMC estimates of the local posterior in Bayesian FL, we can utilize the (amortized) Variational Inference (VI) framework of Bayesian meta-learning [50, 154, 155, 156] that is originally developed for few-shot multi-task learning [157, 158, 159, 160]. Considering each client in FL as an individual task, an evidence lower-bound (ELBO) $\mathcal{L}_{\text{ELBO}}$ over all the distributed M datasets is defined as

$$\max_{\phi} \mathcal{L}_{\text{ELBO}}(\phi) = \sum_{m=1}^{M} g^m \{\mathbb{E}_{q(w^m;\phi)}[\log p(y^m|x^m, w^m)] - \text{KL}(q(w^m;\phi)||p(w^m))\}. \tag{5.2}$$

Here, $p(y^m|x^m, w^m)$ represents a likelihood model constructed with a neural network (NN) [38, 40, 41, 161], and w^m is a client-specific NN weight. $q(w^m;\phi)$ is a variational posterior model parametrized by ϕ. g^m is the weight as defined in equation 5.1. Each local ELBO associated with the m-th client in equation 5.2 trade-offs between the expected log-likelihood on their local dataset $\mathcal{D}^m$ and the KL divergence with a prior $p(w^m)$. The prior can act as a regularizer for the variational posterior $q(w^m;\phi)$. In the VI, maximizing the $\mathcal{L}_{\text{ELBO}}(\phi)$ with respect to the variational parameter ϕ is equivalent to minimizing $\sum_{t=1}^{M} g^m \text{KL}(q(w^m;\phi)||p(w^m|\mathcal{D}^m))$. Thus, $q(w^m;\phi)$ can be trained to approximate the true unknown posterior over w^m (*i.e.* $p(w^m|\mathcal{D}^m) \approx q(w^m;\phi)$)

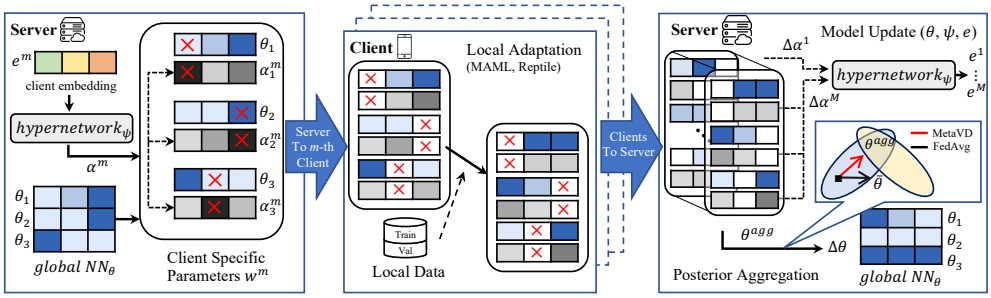


Figure 5.1: Overview of the Meta-Variational Dropout (MetaVD) algorithm. The server's hypernetwork predicts client-specific dropout rates from client embedding, e^m. Global parameters, θ, and dropout variables, α^m, are sent to m-th client. Following local posterior adaptation, the updated parameters are transmitted back to the server, which then updates its variational parameters θ, ψ, and e.

[38, 40, 41, 161].

A straightforward Bayesian PFL approach might be utilizing a separate variational model for all client devices (*i.e.* $\phi = \phi^1, \cdots, \phi^M$). However, only a sparse subset of clients can participate in each FL round due to the communication availability of edge devices. Moreover, the number of clients M can be extremely large in practice, and some clients might only have small datasets. Thus, learning the variational parameter ϕ^m independently for each device is difficult. As an alternative, we introduce a new hypernetwork-based [46, 47, 48, 49] conditional dropout posterior modeling approach that can be data-efficiently trained across multiple clients in FL.

5.3.2 Conditional Posterior Model

To promote efficient model personalization and reduce overfitting in Bayesian FL, we define the posterior model in 5.2 based on a Variational Dropout (VD) technique that multiplies continuous Gaussian noise to the NN parameters during training to prevent overfitting [44, 52, 42, 43, 51]. MetaVD extends the VD

posterior by employing a global hypernetwork learning to predict client-specific dropout rates (or personal model structure). In MetaVD, the variational posterior model for each m-th client's weight, $q(w^m;\phi)$ in equation 5.2, is defined as

$$q(w^m;\phi=(\theta,\psi,e^m))=\prod_{k=1}^{K}\mathcal{N}(w_k^m|\theta_k,\alpha_k^m\theta_k^2) \text{ where } \alpha^m=h_\psi(e^m). \tag{5.3}$$

The variational parameter ϕ is characterized by three distinct variational parameters (θ,ψ,e). θ is a global NN model parameter kept at the server (which has no client index m), and K is the dimension of the parameter. The posterior distribution over the m-th client's weight w^m is described as a product of Gaussian distributions. Essentially, equation 5.3 describes a form of conditional Gaussian noise multiplication to the global NN parameters (*e.g.* $w^m=\theta*\epsilon^m$ where $\epsilon^m\sim\mathcal{N}(\mathbf{1},\alpha^m)$). α_k^m represents the client-specific dropout variable[1] on each k-th NN parameter θ_k. The h_ψ is a hypernetwork [47, 48, 49, 49], parameterized by ψ, predicting the client-specific dropout rate α^m. e^m is a learnable client embedding used as an input to the h_ψ. Since learning one hypernetwork across multiple local clients is more efficient than learning all the local posteriors independently, this approach can mitigate the sparse client participation and limited data issues in FL.

5.3.3 Hierarchical Prior Model

To optimize the posterior model $q(w^m;\phi)$ in equation 5.2, we need to specify the KL divergence term and the prior model $p(w^m)$. The prior model in MetaVD has two criteria: (i) the KL divergence terms must be independent of the NN parameter θ to ensure the lower-bound assumption in VD [44, 52, 43, 51],

[1]The dropout rate is $p=\alpha/(1+\alpha)\in[0,1]$ [162, 44]. We refer to α as a dropout variable for simplicity.

and (ii) all clients must share the same prior model to support the multiplicative posterior aggregation rule in Bayesian FL. We utilize the hierarchical prior [163, 164, 52] discussed in [52] due to its straightforward analytic KL term derivation and proven efficacy in network sparsification. Under the hierarchical prior assumption, the KL divergence term in equation 5.2 simplifies to $\mathrm{KL}(q(w^m;\phi)||p(w^m)) = \sum_{k=1}^{K} 0.5\log(1+(\alpha_k^m)^{-1})$; please refer Appendix for more details. This KL term is independent of the global NN parameter θ and regularizes the dropout variable efficiently. The same hierarchical prior is applied across all the $1...M$ clients.

5.3.4 Client-side Optimization

Initially, the hypernetwork in the server approximates the dropout variable α^m for each m-th client. The global parameter θ and α^m are transmitted to each client device. Then, the client's posterior model of equation 5.3 is trained on local data using the following ELBO objective:

$$\max_{\theta,\alpha^m} \mathcal{L}_{\mathrm{ELBO}}^m(\theta,\alpha) = \frac{1}{|\mathcal{D}^m|}\sum_i \log p(y_i^m|x_i^m, f(\epsilon;\theta,\alpha^m)) - \sum_{k=1}^{K} 0.5\log(1+(\alpha_k^m)^{-1}), \tag{5.4}$$

which maximizes $\mathcal{L}_{\mathrm{ELBO}}^m$ on the client dataset $\mathcal{D}^m = \{(x_i^m, y_i^m)\}_{i=1}^{|\mathcal{D}^m|}$ with respect to the variational parameters (*i.e.* θ, α^m). The optimization can be done by the *stochastic gradient variational Bayes* (SGVB) [38, 40, 41], which reparameterizes the random weight variable w^m using a differentiable transformation as $w_k^m = f(\epsilon_k;\theta_k,\alpha_k^m) = \theta_k + \sqrt{\alpha_k^m}\theta_k\epsilon_k$ with a random i.i.d. noise $\epsilon_k \sim \mathcal{N}(0,1)$. The intermediate weight w_k^m is now differentiable with respect to θ_k and α_k^m and can be optimized via the SGD. The KL term act as regularization for α_k^m.

5.3.5 Server-side Optimization

Once a local posterior adaptation is done in each client device, the updated parameters θ^m and α^m are returned to the server, which then updates the variational parameter (*e.g.* θ, ψ, e^m) using the collected local parameters. To update the global NN parameter θ, we follow the posterior multiplication assumption: $p(w|\mathcal{D}) \propto \prod_{m=1}^{M} p(w^m|\mathcal{D}^m)$ [151, 152, 153, 165]. Since the local posteriors in equation 5.3 is a Gaussian (dropout) distribution, their product are also Gaussian: $\mathcal{N}(w|\theta^{\text{agg}}, \cdot) \approx \prod_{m=1}^{M} q(w^m; \theta^m, \alpha^m)$. This provides us with the aggregation rule for calculating the mean of the global posterior θ_{agg} as follows:

$$\theta_k^{\text{agg}} = \frac{1}{M} \sum_m r_k^m \theta_k^m \text{ where } r_k^m = \frac{g^m (\alpha_k^m (\theta_k^m)^2)^{-1}}{\sum_m g^m (\alpha_k^m (\theta_k^m)^2)^{-1}}. \tag{5.5}$$

Note that equation 5.5 has an intuitive interpretation that the aggregation weight r_k^m is inversely proportional to its corresponding dropout variable (or noise variance) α_k^m. Thus, parameters with high uncertainty have correspondingly less influence on the overall mean prediction. In this way, we can fully utilize the uncertainty in the model parameter for the FL. To update the global parameter θ, we follow the parameter update rule of FedAVG except that we use the mean aggregation in equation 5.5. For the parameter ψ, a more general update rule in [46] is employed; we compute the changes in the updated dropout for each client $\Delta\alpha^m$ as described in Algorithm 1. Then, the gradient for the hypernetwork parameter is computed using the chain rule as $\nabla_\psi \mathcal{L}_{\text{ELBO}}^m(\alpha^m) = (\nabla_\psi \alpha^m)^T \Delta\alpha^m$, where $\nabla_\psi \alpha^m$ is the gradient of a hypernetwork's output. The gradient for $\nabla_{e^m} \mathcal{L}_{\text{ELBO}}^m(\alpha^m)$ can be derived using the same chain rule. The detailed updating rules for each parameter are summarized in Algo.1.

Algorithm 1: MetaVD algorithm with MAML and Reptile variant for FL

Input: # of communication round R, # of client N, server learning rate η, client learning rate γ, inner learning rate l, local steps E, inner steps I, KL divergence parameter β.
Initialize: a global parameter θ, hypernetwork $h_\psi(\cdot)$ and e.
for $r = 1, ..., R$ **do**
  Sample M clients from $1, ..., N$ clients
  for $m = 1, ..., M$ **do**
    Set $\theta^m = \theta$ and $\alpha^m = h_\psi(e^m)$
    Send θ^m and α^m to the m-th client
    θ^m_*, $\alpha^m_* \leftarrow$ LocalAdaptation(θ^m, α^m)
  end for
  Compute global param θ^{agg} using θ^m_*, α^m_* and equation 5.5
  $\Delta\theta \leftarrow \theta^{\text{agg}} - \theta$
  $\Delta\alpha^m \leftarrow \alpha^m_* - \alpha^m$
  $\theta \leftarrow \theta + \eta\Delta\theta$
  $\psi \leftarrow \psi - \eta\frac{1}{M}\Sigma_m g_m((\nabla_\psi \alpha^m)^T \Delta\alpha^m)$
  for $m = 1, \cdots, M$ **do**
    $e^m \leftarrow e^m - \eta(\nabla_{e^m}\psi)^T(\nabla_\psi \alpha^m)^T \Delta\alpha^m$
  end for
end for

Procedure LocalAdaptation_MAML(θ, α)
  for Each local step i from 1 to E **do**
    Set $\theta'_i = \theta_i$ and $\alpha'_i = \alpha_i$
    Sample dataset D^m_{tr} and D^m_{val} from D^m
    for Each inner step j from 1 to I **do**
      $\theta'_i \leftarrow \theta'_i - l\nabla_{\theta'_i}\mathcal{L}^m_{ELBO}(\theta'_i, \alpha'_i; D^m_{\text{tr}})$
    end for
    $\theta_i \leftarrow \theta_i - \gamma\nabla_{\theta_i}\mathcal{L}^m_{ELBO}(\theta'_i, \alpha'_i; D^m_{\text{val}})$
    $\alpha_i \leftarrow \alpha_i - \gamma\nabla_{\alpha_i}\mathcal{L}^m_{ELBO}(\theta'_i, \alpha'_i; D^m_{\text{val}})$
  end for
  Set $\theta_* = \theta_i$ and $\alpha_* = \alpha_i$
  Send θ_* and α_* to the server

Procedure LocalAdaptation_Reptile(θ, α)
  for Each local step i from 1 to E **do**
    $\theta_i \leftarrow \theta_i - \gamma\nabla_{\theta_i}\mathcal{L}^m_{ELBO}(\theta_i, \alpha_i; D^m)$
    $\alpha_i \leftarrow \alpha_i - \gamma\nabla_{\alpha_i}\mathcal{L}^m_{ELBO}(\theta_i, \alpha_i; D^m)$
  end for
  Set $\theta_* = \theta_i$ and $\alpha_* = \alpha_i$
  Send θ_* and α_* to the server

5.3.6 Combination with Other Meta-learning Algorithms

Since the KL regularization in equation 5.4 is independent of the global (or initial) parameter θ, MetaVD can be combined with various existing meta-learning-based PFL algorithms (e.g., Reptile [145, 26], MAML [144, 160], PerFedAvg [146, 166]). For example, MAML [144, 160] requires a few inner update steps using a subsampled dataset to compute the second-order gradient for θ. Reptile [145, 26] only utilize a first-order gradient computation as described in Algorithm 1 that is similar to FedAvg except for the learning rate η. Each local adaptation step is performed using the $\mathcal{L}^m_{\text{ELBO}}$ in equation 5.4 and local dataset D^m. Unlike the conventional meta-learning-based PFL algorithms that only keep one global initialization parameter, MetaVD enables to alteration of the mode of the initial parameters. This approach allows effective utilization of the global parameter in the multi-domain FL experiment.

5.4 Experiments

To verify the MetaVD approach, we perform extensive experiments under various scenarios [141] (*i.e.* different degrees of non-i.i.d. and client participation rates), using multiple different FL datasets [167] such as CIFAR-10, CIFAR-100, FEMINIST, and CelebA. In order to evaluate the effectiveness of the hypernetwork, an ablation study comparing MetaVD to regular VD is performed. Furthermore, we evaluate the uncertainty calibration and model compression ability of MetaVD. Lastly, we test MetaVD with multi-domain datasets.

Baselines. We compare our method with standard FL methods (*e.g.* FedAvg [125] and FedProx [168]), meta-learning PFL algorithms (*e.g.* Reptile [145], MAML [144], and PerFedAvg [146]), and Bayesian FL methods (*i.e.* FedBE [151]). For all baselines, we use the widely adopted CNN model in

FL [127, 169, 46]. Additionally, "fine-tuning" (FT) indicates that a few local learning steps are performed on FedAvg before the evaluation. Please see the appendix for an overview of the baselines.

Implementation. For reproducibility, we conduct experiments in a containerized environment that simulates FL communication with clients only in a server. We test $T = 1000$ of total FL rounds following the conventions in [141]. All experiments are executed on a cluster of 32 NVIDIA GTX 1080 GPUs. The hypernetwork of MetaVD consists of an embedding layer of dimension $(1 + M/4)$, followed by three fully connected NNs with Reaky ReLU activation and one exponential activation for the dropout logit output. The predicted dropout variable is then applied to the global weight of other baselines. In our study, we apply MetaVD to just one fully connected layer before the output layer [170, 171, 172, 173], which yields significant performance improvements in all experiments. Please refer to Appendix for implementation details.

5.4.1 Generalization on Non-i.i.d. Settings

Datasets and training. To evaluate the generalization capabilities of the models under non-i.i.d. data conditions, we conduct tests on both CIFAR-10 and CIFAR-100 datasets with varying degrees of heterogeneity. We follow the similar evaluation protocols of pFL-Bench [141], leveraging Dirichlet Allocation to partition each dataset into 130 clients with varying Dirichlet parameters denoted as $\dot{\alpha} = [5, 0.5, 0.1]$. As shown in Figure 5.2, the class labels and the data size per client are heterogeneous across the clients. A smaller $\dot{\alpha}$ represents a higher degree of heterogeneity. To evaluate the Test accuracy and generalization performance of the baselines on new clients, we randomly select 30 out of 130 clients as out-of-distribution (OOD) clients who are not involved during the training phase. More details are in Appendix.

Hetrogenity	*CIFAR-100* dataset $\dot{\alpha} = 5.0$			*CIFAR-100* dataset $\dot{\alpha} = 0.5$			*CIFAR-10* dataset $\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	42.35	43.08	+0.73	41.92	41.96	+0.04	71.65	71.57	−0.08
FedAvg+FT [141]	41.49	42.45	+0.96	40.99	39.83	−1.16	69.62	68.38	−1.24
FedProx [168]	42.23	44.11	+1.88	42.03	40.51	−1.52	72.27	73.75	+1.48
FedBE [151]	45.17	45.43	+0.26	44.29	44.23	−0.06	70.23	69.19	−1.04
Reptile [145]	47.87	47.73	−0.14	46.13	45.94	−0.19	73.93	76.36	+2.43
MAML [144]	48.30	49.14	+0.84	46.33	46.65	**+0.32**	76.06	74.89	−1.17
PerFedAvg (HF-MAML) [146]	48.19	47.35	−0.84	46.22	46.36	+0.14	75.42	79.56	+4.14
FedAvg+MetaVD (ours)	47.82	50.26	**+2.44**	47.54	47.55	+0.01	76.87	76.25	−0.62
Reptile+MetaVD (ours)	**53.71**	**54.50**	+0.79	**52.06**	**51.50**	−0.56	76.51	**82.07**	+5.56
MAML+MetaVD (ours)	52.40	51.78	−0.62	50.21	49.75	−0.46	**77.27**	79.05	+1.78
PerFedAvg+MetaVD (ours)	51.67	51.70	+0.03	50.02	48.70	−1.32	76.06	81.77	**+5.71**

Table 5.2: Classification accuracies with different (non-i.i.d.) heterogeneity degrees of $\dot{\alpha} = [5.0, 0.5]$ in CIFAR-100 and $\dot{\alpha} = 0.1$ in CIFAR-10. The higher score, the better.

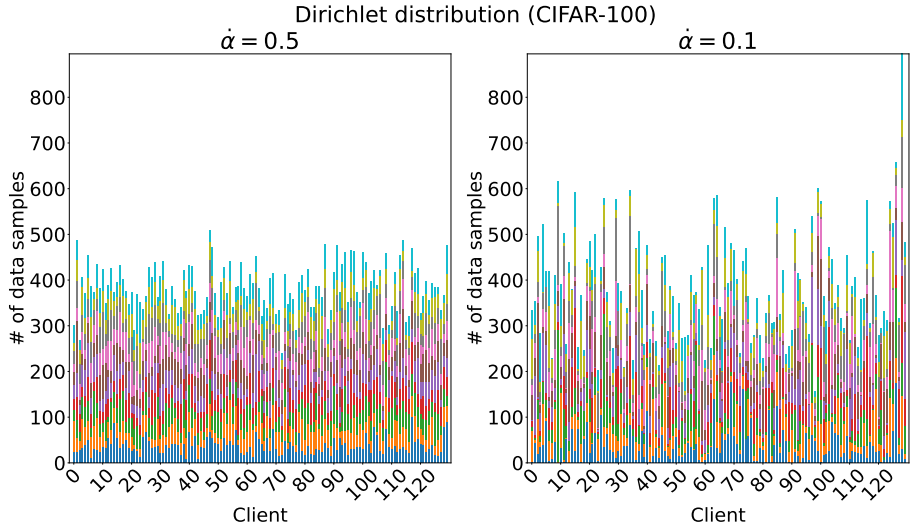


Figure 5.2: Visualization of client's data distribution in different non-i.i.d. degrees ($\dot{\alpha} = [0.5, 0.1]$).

Results. Table 5.2 presents the weighted average classification accuracy for participating (Test) and non-participating (OOD) clients on CIFAR-10 and CIFAR-100 datasets with varying non-i.i.d. degrees. The generalization gap, denoted by Δ, represents the difference between OOD and Test accuracy. As shown in Table 5.2, PFL methods such as Reptile, MAML, and PerFedAvg generally outperform non-PFL methods like FedAvg and FedProx. While the Bayesian ensemble approach, FedBE, improves upon FedAvg, it still lags behind PFL methods in OOD accuracy. When $\dot{\alpha}$ changes from 5.0 to 0.5, the Test and OOD accuracy of all models decrease since the degree of non-i.i.d. increases. When combined with MetaVD, all baselines show significant performance enhancements, regardless of whether they are FL or PFL algorithms (*e.g.* Reptile enhances from 47.87 to 53.71 adapting MetaVD). These results demonstrate the adaptability and efficacy of MetaVD in mitigating model overfitting and handling the non-i.i.d. client data in FL contexts.

CIFAR-100 dataset			
Method	Test (%)	OOD (%)	Δ
Reptile [145]	47.87	47.73	+0.14
Reptile+VD [44]	50.20	49.28	**+0.92**
Reptile+EnsembleVD [174]	52.49	52.36	+0.13
Reptile+MetaVD (ours)	**53.71**	**54.50**	−0.79

Table 5.3: MetaVD ablation study in CIFAR-100.

5.4.2 Ablation Study

Settings. To evaluate the capability of the hypernetwork in MetaVD, we conduct an ablation study by comparing MetaVD to naive VD [44] and Ensemble VD [174] approaches in the CIFAR-100 dataset. The naive (global) VD model keeps one global dropout parameter shared with all clients; the dropout parameter is treated as a global model parameter as in FedAvg. In EnsembleVD, M independent dropout parameters are kept for all clients. The client-specific dropout parameter can be stored in each client analogous to the partial FL [175, 129, 176]. In contrast, MetaVD utilizes a hypernetwork to learn the personal dropout rate across all clients. All models employ Bayesian posterior aggregation rules based on dropout rates to update the global model parameter [177, 178].

Results. Table 5.3 outlines the results of the ablation study; MetaVD's hypernetwork-based posterior modeling outperforms all other baselines. The dropout rates in baselines like VD or EnsembleVD are not fully optimized due to restricted client participation. On the other hand, both the hypernetwork and the global parameter converge well in MetaVD. This observation demonstrates that MetaVD's hypernetwork offers a more data-efficient approach to learn the client-specific model uncertainty compared to other baselines. More ablation

study performed on the FEMNIST dataset is shown in Appendix.

5.4.3 Uncertainty Calibration

CIFAR-100 dataset		
Method	ECE (%)	MCE (%)
FedAvg [125]	0.60	36.79
FedAvg+FT [141]	0.69	45.04
FedProx [168]	0.67	39.69
FedBE [151]	0.50	34.66
Reptile [145]	0.77	50.52
MAML [144]	0.75	46.57
PerFedAvg (HF-MAML) [146]	0.69	45.27
FedAvg+MetaVD (ours)	**0.39**	**25.27**
Reptile+MetaVD (ours)	0.57	42.40
MAML+MetaVD (ours)	0.52	37.26
PerFedAvg+MetaVD (ours)	0.43	30.20

Table 5.4: Uncertainty calibration scores (ECE and MCE) in CIFAR-100 ($\dot{\alpha} = 0.1$). The lower is the better.

Settings. Identifying any potential bias in the model's prediction is important to avoid serious consequences, especially when the model is used for important decision-making [179, 180, 181]. In the FL environment, where clients have limited non-i.i.d. data, it is even more critical to calibrate the prediction model appropriately. Hence, we explore whether the proposed MetaVD approach can also enhance calibration measures for FL baselines. The Expected Calibration Error (ECE) measures the expected deviation between a model's predicted probability and the actual positive class frequency, while the Maximum Calibration Error (MCE) measures the maximum difference. These calibration metrics are commonly used to evaluate the confidence of probabilistic predictions.

Results. Table 5.4 summarizes the ECE and MCE scores tested with the

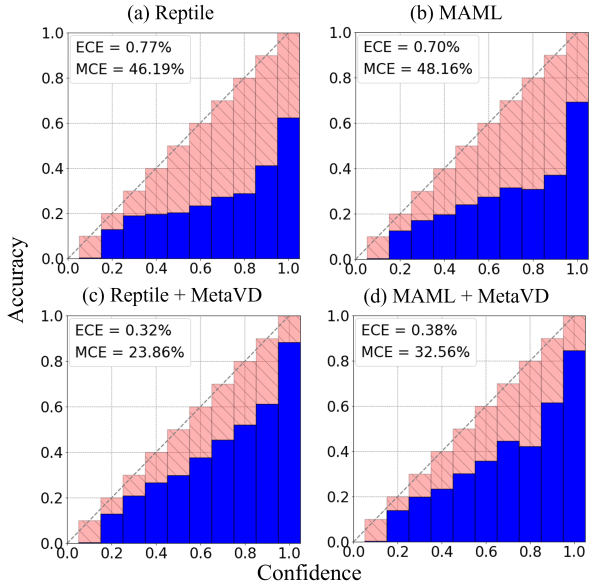


Figure 5.3: Reliability diagrams for (a) Reptile, (b) MAML, (c) Reptile+MetaVD, and (d) MAML+MetaVD in CIFAR-100.

OOD clients in the CIFAR-100 dataset, indicating that the meta-learning-based PFL algorithms (*e.g.* Reptile, MAML, and PerFedAvg) tend to have higher ECE and MCE scores than the conventional FL algorithms (*e.g.* FedAvg, FedProx, and FedBE). This means that PFL baselines achieve high classification accuracy, but their probability predictions are more likely to be biased. This may be due to a byproduct of the additional optimization-based adaptation steps of meta-learning with limited client local data. On the other hand, our MetaVD approach significantly reduces both ECE and MCE scores for all the meta-learning-based methods, indicating that MetaVD effectively mitigates overfitting and reduces the bias on the OOD clients. Figure 5.3 shows the reliability diagrams that visualize model calibration [179, 180, 181] in CIFAR-100 ($\dot{\alpha} = 0.5$). They plot the expected sample accuracy as a function of confidence. If the model is perfectly calibrated, then the diagram becomes the identity function.

Any deviation from a perfect diagonal represents miscalibration. While Reptile and MAML tend to be overconfident in their predictions, Reptile+MetaVD most closely recovers the desired diagonal function. Remarkably, in most instances, employing MetaVD leads to improved model calibration, drawing the reliability diagrams closer to the identity function.

5.4.4 Client Participation

Settings. Under real-world FL scenarios, such as intermittent connections between clients and servers or limited client device performance, numerous clients might not be able to participate in each FL round. This is important for cross-device FL processes with a large number of clients or resource-limited clients. In this experiment, we evaluate the performance of methods under different degrees of client participation rates in each FL round. We experiment with 200 clients in the FEMNIST dataset. For each FL round, we randomly selected 40, 20, and 10 clients for participation during training, with participant client rates s of 0.2, 0.1, and 0.05, respectively. In order to measure OOD accuracy, we exclude 40 pre-selected clients from being selected out of 200 clients so that they do not participate in the entire training.

Results. The overall classification results with different participant client rate s are summarized in Table 5.5. Reptile shows better performance than FedAvg. The effect of data heterogeneity on performance degradation becomes more severe as more clients participate in the training. The decline in Test accuracy when the participant client rate s gets smaller is due to having less training data, as fewer clients take part in each round. Meanwhile, Reptile+MetaVD outperforms other baselines. Interestingly, the performance decrease for Reptile+MetaVD is not as significant as it is for Reptile, showing that MetaVD can also adapt well to FL scenarios with smaller participation sizes.

Sparsity	*FEMNIST* dataset $s = 0.2$			$s = 0.1$			$s = 0.05$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	88.08	85.29	+2.79	88.13	84.70	**+3.43**	88.06	86.22	+1.84
FedAvg+FT [141]	88.33	86.37	+1.96	87.85	86.95	+0.90	87.66	87.11	+0.55
Reptile [145]	88.55	86.52	+2.03	88.39	87.2	+1.19	87.86	88.22	−0.36
FedAvg+MetaVD (ours)	88.81	85.28	**+3.53**	88.69	85.71	+2.98	88.66	86.21	**+2.45**
Reptile+MetaVD (ours)	**89.90**	**89.04**	+0.86	**89.86**	**88.63**	+1.23	**89.43**	**88.71**	+0.72

Table 5.5: Results of classification accuracies with different participant client rates of $s = 0.2$, $s = 0.1$, and $s = 0.05$ in the FEMNIST dataset. We report the results of participating clients (Test) and non-participating clients (OOD). The higher the better.

5.4.5 Muti-domain Datasets

Settings. Existing federated learning algorithms usually assume a single-domain approach, where only one dataset is used in the experiment. Multi-domain learning [182, 183, 184] aims to leverage all available training data across different domains to enhance the performance of the model. In this section, we further evaluate the performance of PFL algorithms on a multi-domain FL dataset, in which we assume each client can have data from different domains. We use three different FL datasets to construct the multi-domain task distributions: FEMNIST, CIFAR-100, and CelebA. To sample each client's local data, we utilize the Dirichlet sampling technique ($\dot{\alpha} = 0.5$) used in the §5.4.1.

Results. Table 5.6 illustrates the classification accuracies for Test and OOD clients when using a combination of two or three datasets. The meta-learning-based PFL algorithm consistently outperforms FedAvg in terms of classification accuracy. Moreover, when MetaVD is applied to FL or PFL algorithm, it significantly enhances prediction accuracy across all multi-domain settings. Notably, MetaVD exhibits larger improvements in OOD accuracy compared to the improvement in Test accuracy. Overall, these findings demonstrate the versatility of MetaVD in enhancing robustness and generalization even in multi-domain FL datasets, regardless of the specific multi-domain settings.

5.4.6 Model Compression

Settings. Federated learning optimization involves frequent communication of model parameters between devices and the central server, which can be slow and may raise privacy concerns. Thus, minimizing the communication cost by reducing the model parameters is an important issue in FL. To explore the compression capabilities of MetaVD, we performed an additional experiment on the CIFAR-10 dataset. The sign DP indicates that each model parameter

Method	2 Domains (a)		2 Domains (b)		2 Domains (c)		3 Domains (d)	
	Test (%)	OOD (%)	Test (%)	OOD (%)	Test (%)	OOD (%)	Test (%)	OOD (%)
FedAvg [125]	43.65	43.45	64.02	52.88	86.81	81.01	63.73	55.55
Reptile [145]	48.92	48.93	66.13	56.22	87.50	83.05	66.98	57.12
MAML [144]	47.39	48.51	66.56	55.72	88.57	84.52	67.08	58.15
PerFedAvg (HF-MAML) [146]	49.21	50.91	66.57	55.24	87.94	83.75	67.20	57.03
FedAvg+MetaVD (ours)	48.23	48.62	65.58	56.85	87.38	82.67	65.93	58.58
Reptile+MetaVD (ours)	**52.26**	**54.75**	**68.35**	59.07	**88.63**	85.03	68.78	61.59
MAML+MetaVD (ours)	51.34	52.82	68.24	61.21	88.59	85.02	**68.81**	**61.60**
PerFedAvg+MetaVD (ours)	51.18	53.06	67.93	**61.32**	88.27	**85.32**	68.05	61.17

Table 5.6: Classification accuracies with multi-domain datasets. (a) CelebA + CIFAR-100, (b) CIFAR-100 + FEMNIST, (c) CelebA + FEMNIST, (d) CelebA + CIFAR-100 + FEMNIST.

CIFAR-10 dataset ($\dot{\alpha} = 0.5$)			
Method	Test (%)	OOD (%)	Sparsity(%)
Reptile+MetaVD	**83.20**	**83.40**	0
MAML+MetaVD	81.32	81.81	0
PerFedAvg+MetaVD	81.06	81.47	0
Reptile+MetaVD+DP	81.40	80.98	**80.06**
MAML+MetaVD+DP	81.48	81.73	79.49
PerFedAvg+MetaVD+DP	82.43	82.19	78.20

Table 5.7: Results of model compression. MetaVD+DP does not communicate the model parameters whose dropout rates are larger than 0.8.

is dropped during the FL communication. we used the thresholding technique to drop the model parameter; any parameter with a dropout rate greater than 0.8 was dropped during the FL rounds.

Results. Table 5.7 demonstrates the results of the Test and OOD accuracy, as well as the sparsity (%) in the CIFAR-100 dataset. The sparsity represents the ratio of zero-valued model parameters in the personalized layer. A higher sparsity percentage indicates greater parameter pruning or elimination performed on the weight. In our experiment, MetaVD could prune about 80% of the weights in the personalized layer. In addition, when we dropped the communication of the parameters between the client and server using the dropout thresholding technique, MetaVD still showed relatively good performance. In the case of Reptile+MetaVD+DP, the performance decreased by approximately 2% while using only 20% of the weights. On the other hand, MAML+MetaVD+DP and PerFedAvg+MetaVD+DP show an improvement in the performance of around 1%. This demonstrates that MetaVD can compress the model parameters required in the personalized layer, reducing the communication cost in FL without sacrificing performance a lot. The appendix shows more experiments on model compression results in the CIFAR-100 dataset with

different non-i.i.d. settings.

5.5 Conclusion

In this chapter, we presented a novel Bayesian personalized federated learning (PFL) approach based on meta-variational dropout (MetaVD). MetaVD utilizes a hypernetwork that predicts the dropout rates for each independent model parameter, which allows an effective model personalization and adaptation in FL with the limited non-i.i.d. data environment. MetaVD can be combined with other conventional meta-learning-based PFL personalized algorithms to prevent overfitting of the model. The conditional dropout posterior enables a principled Bayesian aggregation strategy for consolidating local models into a global one. Additionally, MetaVD's ability to compress model parameters can also reduce communication costs during federated learning. One potential limitation of our approach is that it might increase the complexity of the model due to the introduction of an additional hypernetwork. However, the hypernetwork is kept on the server, and applying our approach to just one last layer before the output layer yields significant performance improvements in all experiments. To validate the performance of MetaVD, we performed extensive experiments on various FL datasets, including CIFAR-10, CIFAR-100, FEMINIST, and CelebA and multi-domain FL datasets. It achieves outstanding classification accuracy and uncertainty calibration, particularly for out-of-distribution (OOD) clients. Overall, the experimental results demonstrate that MetaVD is a highly versatile approach that excels in prediction, uncertainty calibration, generalization, and communication efficiency in the context of FL. These findings support the usefulness of MetaVD in real-world applications.

Chapter 6

Conclusion

6.1 Summary

The main contribution of this work is to design and introduce a new type of Neural Network (NN) based conditional posterior model for Bayesian meta-learning that can bypass the under-fitting and posterior collapsing of existing approaches [32, 34, 33, 35]. MetaVD effectively utilizes the capabilities of Variational Dropout's conditional posterior modeling [83, 42, 86, 87, 88], coupled with a hypernetwork, to generate task-specific dropout rates for each weight in the neural network. To create a robust approximation of task-specific dropout rates, this study employs several advanced techniques within the MetaVD framework, such as an amortized Variational Inference (VI) setup, a Bernoulli expert's meta-model offering a memory-efficient mapping of dropout rates, and an innovative prior, dependent on the entirety of the task data, which optimizes the conditional (dropout) posterior in the amortized VI. The primary strength of MetaVD lies in its ability to adapt a globally learned neural network to

new tasks rapidly. This makes it incredibly suitable for few-shot learning environments, where models quickly adapt to new tasks with limited data. We have evaluated MetaVD on various few-shot learning tasks and datasets. The experiments show that MetaVD can circumvent under-fitting and posterior collapsing and achieve outstanding performances in log-likelihood, active learning efficiency, prediction accuracy, and generalization.

In the context of Federated Learning (FL)—a privacy-enhancing method training a global model from distributed clients—MetaVD addresses common issues such as model overfitting and divergent local models caused by disparate client data. This is achieved through learning client-specific dropout rates via a shared hypernetwork, offering personalized models for FL algorithms. A notable feature is a weight given to conditional dropout uncertainty in consolidating a global model, seen through the lens of Bayesian FL's posterior aggregation. Experimental assessments demonstrate MetaVD's effectiveness in various FL and few-shot learning settings, with remarkable predictive accuracy and uncertainty calibration, especially for out-of-distribution clients. By compressing local model parameters, MetaVD curbs model overfitting and reduces communication costs. In essence, this thesis's novel insights, theoretical contributions, and empirical results advance our understanding of Bayesian Meta-Learning, opening up new possibilities for robust and efficient learning systems that can better adapt to varying tasks with limited data.

6.2 Future Work

There is still room for improvement in the MetaVD. In future work, we may consider improving our proposed methods in this thesis in the following ways:

Adapting the advanced set representations. Adapting advanced set representations could be an exciting avenue for future research. Set represen-

tations have shown promise in tasks that involve unordered or variable-length inputs, such as few-shot learning. The MetaVD approach could benefit from leveraging these advanced set representations, such as those proposed in studies [111, 45, 185]. These set representations could enhance the model's ability to capture complex relationships and dependencies between elements in a set, leading to improved performance in few-shot learning tasks.

Extention to more complex deep learning architectures. Exploring more complex architectures could also be a fruitful direction for future work. The cited papers [186, 187] propose various advanced architectures for meta-learning and few-shot learning tasks. These architectures incorporate innovative design choices such as hierarchical structures, convolutional neural networks, and attention mechanisms. Integrating the MetaVD framework with these complex architectures could potentially lead to even better performance by leveraging the strengths of both approaches. By combining the flexibility and adaptability of MetaVD with the expressiveness and representational power of more sophisticated architectures, it may be possible to achieve even stronger adaptation to context data, improved functional variability, and enhanced generalization to new tasks.

Application in language models. Language modeling tasks often require effectively handling sequential data and capturing the dependencies between words or phrases [66, 188, 68]. Extending the MetaVD framework to language models could improve the model's ability to adapt to different contexts, enhance the variability of representations, and generalize well to new language-related tasks. Exploring the application of MetaVD in tasks such as text generation, sentiment analysis, and natural language understanding would be an intriguing direction to advance the field of language modeling and improve the performance of language-related applications.

Bibliography

[1] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[2] K. P. Murphy, *Machine learning : a probabilistic perspective*. MIT Press, 2013.

[3] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[4] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[5] Z. Liu, Z. Miao, X. Zhan, J. Wang, B. Gong, and S. X. Yu, "Large-scale long-tailed recognition in an open world," in *Proceedings of the IEEE/ CVF conference on computer vision and pattern recognition*, pp. 2537–2546, 2019.

[6] D. K. Naik and R. J. Mammone, "Meta-neural networks that learn by learning," in *IJCNN*, vol. 1, pp. 437–442, IEEE, 1992.

[7] J. Schmidhuber, *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[8] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science,* vol. 350, no. 6266, pp. 1332–1338, 2015.

[9] M. Andrychowicz, M. Denil, S. Gomez, M. W. Hoffman, D. Pfau, T. Schaul, B. Shillingford, and N. De Freitas, "Learning to learn by gradient descent by gradient descent," in *NeurIPS*, 2016.

[10] M. Guillaumin, J. Verbeek, and C. Schmid, "Is that you? metric learning approaches for face identification," in *2009 IEEE 12th international conference on computer vision*, pp. 498–505, IEEE, 2009.

[11] S. Ravi and H. Larochelle, "Optimization as a model for few-shot learning," in *ICLR*, 2017.

[12] O. Vinyals, C. Blundell, T. Lillicrap, K. Kavukcuoglu, and D. Wierstra, "Matching networks for one shot learning," in *NeurIPS*, 2016.

[13] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," in *NeurIPS*, 2017.

[14] G. Koch, R. Zemel, R. Salakhutdinov, *et al.*, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, Lille, 2015.

[15] M. Ren, E. Triantafillou, S. Ravi, J. Snell, K. Swersky, J. B. Tenenbaum, H. Larochelle, and R. S. Zemel, "Meta-learning for semi-supervised few-shot classification," *arXiv preprint arXiv:1803.00676,* 2018.

[16] N. Dvornik, C. Schmid, and J. Mairal, "Diversity with cooperation: Ensemble methods for few-shot classification," in *Proceedings of the IEEE/ CVF International Conference on Computer Vision*, pp. 3723–3731, 2019.

[17] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv preprint arXiv:1605.06065,* 2016.

[18] Y. Duan, J. Schulman, X. Chen, P. L. Bartlett, I. Sutskever, and P. Abbeel, "Rl ^2: Fast reinforcement learning via slow reinforcement learning," *arXiv preprint arXiv:1611.02779,* 2016.

[19] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick, "Learning to reinforcement learn," *arXiv preprint arXiv:1611.05763,* 2016.

[20] T. Munkhdalai and H. Yu, "Meta networks," in *International conference on machine learning*, pp. 2554–2563, PMLR, 2017.

[21] N. Mishra, M. Rohaninejad, X. Chen, and P. Abbeel, "A simple neural attentive meta-learner," *arXiv preprint arXiv:1707.03141,* 2017.

[22] S. Bengio, Y. Bengio, J. Cloutier, and J. Gescei, "respectively," in *Optimality in Biological and Artificial Networks*, pp. 281–303, Routledge, 2013.

[23] M. Husken and C. Goerick, "Fast learning for problem classes using knowledge based network initialization," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 6, pp. 619–624, IEEE, 2000.

[24] K. Li and J. Malik, "Learning to optimize," *arXiv preprint arXiv:1606.01885,* 2016.

[25] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *PMLR*, pp. 1126–1135, 2017.

[26] A. Nichol, J. Achiam, and J. Schulman, "On First-Order Meta-Learning algorithms," *arXiv:1803.02999,* Mar. 2018.

[27] E. Grant, C. Finn, S. Levine, T. Darrell, and T. Griffiths, "Recasting gradient-based meta-learning as hierarchical bayes," in *ICLR*, 2018.

[28] J. Yoon, T. Kim, O. Dia, S. Kim, Y. Bengio, and S. Ahn, "Bayesian model-agnostic meta-learning," in *NeurIPS*, pp. 7343–7353, 2018.

[29] C. Finn, K. Xu, and S. Levine, "Probabilistic model-agnostic meta-learning," in *NeurIPS*, 2018.

[30] H. B. Lee, T. Nam, E. Yang, and S. J. Hwang, "Meta dropout: Learning to perturb latent features for generalization," in *ICLR*, 2019.

[31] S. Ravi and A. Beatson, "Amortized bayesian meta-learning.," in *ICLR*, 2019.

[32] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner, "Meta-learning probabilistic inference for prediction," in *ICLR*, 2018.

[33] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, D. J. Rezende, S. Eslami, and Y. W. Teh, "Neural processes," in *ICML Workshop on Theoretical Foundations and Applications of Deep Generative Models*, 2018.

[34] M. Garnelo, D. Rosenbaum, C. J. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. J. Rezende, and S. Eslami, "Conditional neural processes," in *ICML*, 2018.

[35] E. Iakovleva, J. Verbeek, and K. Alahari, "Meta-learning with shared amortized variational inference," in *ICML*, pp. 4572–4582, PMLR, 2020.

[36] C. Nguyen, T.-T. Do, and G. Carneiro, "Uncertainty in model-agnostic meta-learning using variational inference," in *WACV*, pp. 3090–3100, 2020.

[37] D. J. C. MacKay, "A practical bayesian framework for backpropagation networks," *Neural Comput.*, vol. 4, pp. 448–472, May 1992.

[38] R. M. Neal, *Bayesian Learning for Neural Networks*. Springer Science & Business Media, Dec. 2012.

[39] D. Barber and C. M. Bishop, "Ensemble learning in bayesian neural networks," *Nato ASI Series F Computer and Systems Sciences*, vol. 168, pp. 215–238, 1998.

[40] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," *NeurIPS*, 2017.

[41] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network," in *Proceedings of the 32nd International Conference on Machine Learning* (F. Bach and D. Blei, eds.), vol. 37 of *Proceedings of Machine Learning Research*, (Lille, France), pp. 1613–1622, PMLR, 2015.

[42] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *ICML*, pp. 1050–1059, 2016.

[43] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *Proceedings of the 34th International Con-*

ference on Machine Learning (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 2498–2507, PMLR, 2017.

[44] Kingma, Salimans, and others, "Variational dropout and the local reparameterization trick," *Adv. Neural Inf. Process. Syst.*, 2015.

[45] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," *ICLR*, 2019.

[46] A. Shamsian, A. Navon, E. Fetaya, and G. Chechik, "Personalized federated learning using hypernetworks," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 9489–9502, PMLR, 2021.

[47] D. Ha, A. Dai, and Q. V. Le, "HyperNetworks," *ICLR*, 2017.

[48] J. von Oswald, C. Henning, J. Sacramento, and B. F. Grewe, "Continual learning with hypernetworks," *ICLR*, 2020.

[49] D. Zhao, J. von Oswald, S. Kobayashi, J. Sacramento, and B. F. Grewe, "Meta-Learning via hypernetworks," in *NeurIPS*, IEEE, Dec. 2020.

[50] J. Gordon, J. Bronskill, M. Bauer, S. Nowozin, and R. E. Turner, "Meta-Learning probabilistic inference for prediction," *ICLR*, 2019.

[51] J. Hron, A. Matthews, and Z. Ghahramani, "Variational bayesian dropout: pitfalls and fixes," in *International Conference on Machine Learning*, pp. 2019–2028, 2018.

[52] Liu, Dong, Zhang, Gong, and others, "Variational bayesian dropout with a hierarchical prior," *Proc. Estonian Acad. Sci. Biol. Ecol.*, 2019.

[53] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *ML,* vol. 37, no. 2, pp. 183–233, 1999.

[54] M. J. Wainwright and M. I. Jordan, *Graphical Models, Exponential Families, and Variational Inference*. Now Publishers Inc., 2008.

[55] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, and Y. W. Teh, "Attentive neural processes," in *ICML*, 2019.

[56] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review,* vol. 65, no. 6, p. 386, 1958.

[57] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *ICML*, pp. 807–814, 2010.

[58] F. Scarselli and A. C. Tsoi, "Universal approximation using feedforward neural networks: A survey of some existing methods, and some new results," *Neural networks,* vol. 11, no. 1, pp. 15–37, 1998.

[59] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2014.

[60] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *PAMI,* vol. 35, no. 8, pp. 1798–1828, 2013.

[61] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pp. 818–833, Springer, 2014.

[62] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: analysis, applications, and prospects," *IEEE transactions on neural networks and learning systems,* 2021.

[63] L. R. Medsker and L. Jain, "Recurrent neural networks," *Design and Applications,* vol. 5, no. 64-67, p. 2, 2001.

[64] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural computation,* vol. 9, no. 8, pp. 1735–1780, 1997.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *NeurIPS*, pp. 5998–6008, 2017.

[66] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *NAACL-HLT*, 2019.

[67] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative Adversarial Nets," in *NeurIPS*, 2014.

[68] OpenAI, "Chatgpt: A large language model for medical education," *PLOS Digital Health,* 2023.

[69] K. Choudhary, B. DeCost, C. Chen, A. Jain, F. Tavazza, R. Cohn, C. W. Park, A. Choudhary, A. Agrawal, S. J. Billinge, *et al.*, "Recent advances and applications of deep learning methods in materials science," *npj Computational Materials,* vol. 8, no. 1, p. 59, 2022.

[70] R. Caruana, S. Lawrence, and C. Giles, "Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping," *Advances in neural information processing systems*, vol. 13, 2000.

[71] J. Lever, M. Krzywinski, and N. Altman, "Points of significance: model selection and overfitting," *Nature methods*, vol. 13, no. 9, pp. 703–705, 2016.

[72] R. M. Neal, *Bayesian learning for neural networks*, vol. 118. Springer Science & Business Media, 2012.

[73] D. Barber and C. M. Bishop, "Ensemble learning for multi-layer networks," *Advances in neural information processing systems*, pp. 395–401, 1998.

[74] D. J. C. MacKay, *A practical Bayesian framework for backpropagation networks*. MIT Press, 1992.

[75] W. L. Buntine and A. S. Weigend, "Bayesian back-propagation," *Complex systems*, vol. 5, no. 6, pp. 603–643, 1991.

[76] G. E. Hinton and D. Van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13, 1993.

[77] J. S. Denker and Y. LeCun, "Transforming neural-net output levels to probability distributions," in *NeurIPS*, pp. 853–859, 1991.

[78] A. Graves, "Practical variational inference for neural networks," in *Advances in neural information processing systems*, pp. 2348–2356, Citeseer, 2011.

[79] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *ICML*, 2015.

[80] Y. Gal, *Uncertainty in Deep Learning*. University of Cambridge, 2016.

[81] S. L. Scott, A. W. Blocker, F. V. Bonassi, H. A. Chipman, E. I. George, and R. E. McCulloch, "Bayes and big data: the consensus monte carlo algorithm," *International Journal of Management Science and Engineering Management*, vol. 11, pp. 78–88, Apr. 2016.

[82] L. Bottou and Y. Bengio, "Convergence Properties of the K-Means Algorithms," in *NeurIPS*, 1995.

[83] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *NeurIPS*, pp. 2575–2583, 2015.

[84] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?," in *NeurIPS*, pp. 5574–5584, 2017.

[85] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, pp. 452–459, 2015.

[86] D. Molchanov, A. Ashukha, and D. Vetrov, "Variational dropout sparsifies deep neural networks," in *ICML*, pp. 2498–2507, 2017.

[87] J. Hron, A. G. d. G. Matthews, and Z. Ghahramani, "Variational bayesian dropout: pitfalls and fixes," in *ICML*, 2018.

[88] Y. Liu, W. Dong, L. Zhang, D. Gong, and Q. Shi, "Variational bayesian dropout with a hierarchical prior," in *CVPR*, pp. 7124–7133, 2019.

[89] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *JMLR,* pp. 1929–1958, 2014.

[90] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580,* 2012.

[91] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *ICML*, pp. 1058–1066, 2013.

[92] S. Wang and C. Manning, "Fast dropout training," in *ICML*, pp. 118–126, 2013.

[93] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using DropConnect," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 1058–1066, PMLR, 2013.

[94] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning* (*Adaptive Computation and Machine Learning*). MIT Press, 2005.

[95] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *ICLR*, 2013.

[96] B. Bloem-Reddy and Y. W. Teh, "Probabilistic symmetry and invariant neural networks," *JMLR,* 2020.

[97] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, and S. M. A. Eslami, "Conditional

neural processes," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 1704–1713, PMLR, 2018.

[98] H. Edwards and A. Storkey, "Towards A Neural Statistician," in *ICLR*, 2017.

[99] A. Grover, D. Tran, R. Shu, B. Poole, and K. Murphy, "Probing uncertainty estimates of neural processes," in *NeurIPS Workshop on Bayesian Deep Learning*, 2019.

[100] N. Parmar, A. Vaswani, J. Uszkoreit, Ł. Kaiser, N. Shazeer, A. Ku, and D. Tran, "Image transformer," *arXiv preprint arXiv:1802.05751,* 2018.

[101] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *arXiv preprint arXiv:1611.03530,* 2016.

[102] T. A. Le, H. Kim, M. Garnelo, D. Rosenbaum, J. Schwarz, and Y. W. Teh, "Empirical evaluation of neural process objectives," in *NeurIPS Workshop on Bayesian Deep Learning*, 2018.

[103] M. D. Hoffman and M. J. Johnson, "Elbo surgery: yet another way to carve up the variational evidence lower bound," in *NeurIPS Workshop in Advances in Approximate Bayesian Inference*, vol. 1, p. 2, 2016.

[104] C. K. Sønderby, T. Raiko, L. Maaløe, S. K. Sønderby, and O. Winther, "How to train deep variational autoencoders and probabilistic ladder networks," in *ICML*, 2016.

[105] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *NeurIPS*, pp. 4743–4751, 2016.

[106] X. Chen, D. P. Kingma, T. Salimans, Y. Duan, P. Dhariwal, J. Schulman, I. Sutskever, and P. Abbeel, "Variational lossy autoencoder," in *ICLR*, 2017.

[107] S. Yeung, A. Kannan, Y. Dauphin, and L. Fei-Fei, "Tackling over-pruning in variational autoencoders," in *ICML*, 2017.

[108] A. A. Alemi, B. Poole, I. Fischer, J. V. Dillon, R. A. Saurous, and K. Murphy, "Fixing a broken elbo," in *PMLR*, 2018.

[109] J. Lucas, G. Tucker, R. B. Grosse, and M. Norouzi, "Don't blame the elbo! a linear vae perspective on posterior collapse," in *NeurIPS*, pp. 9403–9413, 2019.

[110] A. B. Dieng, Y. Kim, A. M. Rush, and D. M. Blei, "Avoiding latent variable collapse with generative skip models," in *AISTAT*, 2018.

[111] J. Lee, Y. Lee, J. Kim, A. R. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *ICML*, 2019.

[112] L. Leemis, "The product of n mutually independent bernoulli random variables is bernoulli.." `http://www.math.wm.edu/~leemis/chart/UDR/PDFs/BernoulliP.pdf`, 2020.

[113] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[114] Y. W. Teh and I. Lecture, "On statistical thinking in deep learning a blog post." https://imstat.org/wp-content/uploads/bulletin/ml-LONG_On_Statistical_Thinking_in_Deep_Learning.pdf, 2019.

[115] B. De Finetti, *Probability, induction, and statistics: The art of Guessing.* Widely, London, 1972.

[116] J. M. Bernardo, "The concept of exchangeability and its applications," *Far East Journal of Mathematical Sciences*, 1996.

[117] J. Hron, A. G. d. G. Matthews, and Z. Ghahramani, "Variational gaussian dropout is not bayesian," in *NeurIPS*, 2017.

[118] J. Tomczak and M. Welling, "Vae with a vampprior," in *AISTATS*, pp. 1214–1223, 2018.

[119] H. Takahashi, T. Iwata, Y. Yamanaka, M. Yamada, and S. Yagi, "Variational autoencoder with implicit optimal priors," in *AAAI*, pp. 5066–5073, 2019.

[120] D. P. Kingma, D. J. Rezende, S. Mohamed, and M. Welling, "Semi-supervised learning with deep generative models," in *NeurIPS*, 2014.

[121] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.

[122] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998.

[123] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *ICCV*, 2015.

[124] Z. Li, F. Zhou, F. Chen, and H. Li, "Meta-sgd: Learning to learn quickly for few-shot learning," *arXiv preprint arXiv:1707.09835,* 2017.

[125] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR, 2017.

[126] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *arXiv preprint arXiv:1806.00582.,* June 2018.

[127] J. Oh, S. Kim, and S.-Y. Yun, "FedBABU: Towards enhanced representation for federated image classification," *ICLR,* 2022.

[128] Y. Liu, A. Huang, Y. Luo, H. Huang, Y. Liu, Y. Chen, L. Feng, T. Chen, H. Yu, and Q. Yang, "FedVision: An online visual object detection platform powered by federated learning," *AAAI,* vol. 34, pp. 13172–13179, Apr. 2020.

[129] B. Sun, H. Huo, Y. I. Yang, and B. Bai, "PartialFed: Cross-domain personalized federated learning via partial initialization," *Adv. Neural Inf. Process. Syst.,* vol. 34, pp. 23309–23320, Dec. 2021.

[130] A. Hard, K. Rao, R. Mathews, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *ArXiv,* vol. abs/1811.03604, 2018.

[131] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," *arXiv:1812.02903,* Dec. 2018.

[132] K. Muhammad, Q. Wang, D. O'Reilly-Morgan, E. Tragos, B. Smyth, N. Hurley, J. Geraci, and A. Lawlor, "FedFast: Going beyond average for faster training of federated recommender systems," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, (New York, NY, USA), pp. 1234–1242, Association for Computing Machinery, Aug. 2020.

[133] C. Wu, F. Wu, Y. Cao, Y. Huang, and X. Xie, "FedGNN: Federated graph neural network for Privacy-Preserving recommendation," *arXiv:2102.04925*, Feb. 2021.

[134] Q. Yang, J. Zhang, W. Hao, G. P. Spell, and L. Carin, "FLOP: Federated learning on medical datasets using partial networks," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, (New York, NY, USA), pp. 3845–3853, Association for Computing Machinery, Aug. 2021.

[135] I. Dayan, H. R. Roth, A. Zhong, A. Harouni, A. Gentili, A. Z. Abidin, A. Liu, A. B. Costa, B. J. Wood, C.-S. Tsai, C.-H. Wang, C.-N. Hsu, C. K. Lee, P. Ruan, D. Xu, D. Wu, E. Huang, F. C. Kitamura, G. Lacey, G. C. de Antônio Corradi, G. Nino, H.-H. Shin, H. Obinata, H. Ren, J. C. Crane, J. Tetreault, J. Guan, J. W. Garrett, J. D. Kaggie, J. G. Park, K. Dreyer, K. Juluru, K. Kersten, M. A. B. C. Rockenbach, M. G. Linguraru, M. A. Haider, M. AbdelMaseeh, N. Rieke, P. F. Damasceno, P. M. C. E Silva, P. Wang, S. Xu, S. Kawano, S. Sriswasdi, S. Y. Park, T. M. Grist, V. Buch, W. Jantarabenjakul, W. Wang, W. Y. Tak, X. Li, X. Lin, Y. J. Kwon, A. Quraini, A. Feng, A. N. Priest, B. Turkbey, B. Glicksberg, B. Bizzo, B. S. Kim, C. Tor-Díez, C.-C. Lee, C.-J. Hsu, C. Lin, C.-L. Lai, C. P. Hess, C. Compas, D. Bhatia, E. K. Oermann,

E. Leibovitz, H. Sasaki, H. Mori, I. Yang, J. H. Sohn, K. N. K. Murthy, L.-C. Fu, M. R. F. de Mendonça, M. Fralick, M. K. Kang, M. Adil, N. Gangai, P. Vateekul, P. Elnajjar, S. Hickman, S. Majumdar, S. L. McLeod, S. Reed, S. Gräf, S. Harmon, T. Kodama, T. Puthanakit, T. Mazzulli, V. L. de Lavor, Y. Rakvongthai, Y. R. Lee, Y. Wen, F. J. Gilbert, M. G. Flores, and Q. Li, "Federated learning for predicting clinical outcomes in patients with COVID-19," *Nat. Med.*, vol. 27, pp. 1735–1743, Oct. 2021.

[136] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. Nitin Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. El Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konecný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Foundations and Trends® in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[137] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, July 2020.

[138] V. Kulkarni, M. Kulkarni, and A. Pant, "Survey of personalization techniques for federated learning," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability* (*WorldS4*), pp. 794–797, July 2020.

[139] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," 2022.

[140] H. Yuan, W. Morningstar, L. Ning, and K. Singhal, "What do we mean by generalization in federated learning?," *arXiv:2110.14216,* Oct. 2021.

[141] D. Chen, D. Gao, W. Kuang, Y. Li, and B. Ding, "pfl-bench: A comprehensive benchmark for personalized federated learning," *Advances in Neural Information Processing Systems,* vol. 35, pp. 9344–9360, 2022.

[142] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *Proceedings of the 38th International Conference on Machine Learning* (M. Meila and T. Zhang, eds.), vol. 139 of *Proceedings of Machine Learning Research*, pp. 2089–2099, PMLR, 2021.

[143] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *International Conference on Machine Learning*, pp. 6357–6368, PMLR, 2021.

[144] F. Chen, M. Luo, Z. Dong, Z. Li, and X. He, "Federated Meta-Learning with fast convergence and efficient communication," *arXiv:1802.07876,* Feb. 2018.

[145] Y. Jiang, J. Konečný, K. Rush, and S. Kannan, "Improving federated learning personalization via model agnostic meta learning," *arXiv:1909.12488,* Sept. 2019.

[146] Fallah, Mokhtari, and others, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Adv. Neural Inf. Process. Syst.,* 2020.

[147] C. T. Dinh, N. Tran, and J. Nguyen, "Personalized federated learning with moreau envelopes," *Adv. Neural Inf. Process. Syst.,* vol. 33, pp. 21394–21405, 2020.

[148] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.,* vol. 10, pp. 1–19, Jan. 2019.

[149] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. Iii and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 5132–5143, PMLR, 2020.

[150] Z. Zhu, J. Hong, and J. Zhou, "Data-Free knowledge distillation for heterogeneous federated learning," *Proc Mach Learn Res,* vol. 139, pp. 12878–12889, July 2021.

[151] H.-Y. Chen and W.-L. Chao, "FedBE: Making bayesian model ensemble applicable to federated learning," *arXiv:2009.01974,* Sept. 2020.

[152] A. T. Thorgeirsson and F. Gauterin, "Probabilistic predictions with federated learning," *Entropy,* vol. 23, Dec. 2020.

[153] M. Al-Shedivat, J. Gillenwater, E. Xing, and A. Rostamizadeh, "Federated learning via posterior averaging: A new perspective and practical algorithms," *arXiv:2010.05273,* Oct. 2020.

[154] M. Garnelo, J. Schwarz, D. Rosenbaum, F. Viola, and others, "Neural processes," *arXiv preprint arXiv,* 2018.

[155] E. Iakovleva, J. Verbeek, and K. Alahari, "Meta-Learning with shared amortized variational inference," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. Iii and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 4572–4582, PMLR, 2020.

[156] I. Jeon, Y. Park, and G. Kim, "Neural variational dropout processes," in *International Conference On Learning Representations*, 2022.

[157] S. Ravi and A. Beatson, "Amortized bayesian Meta-Learning," *ICLR,* Feb. 2019.

[158] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science,* vol. 350, pp. 1332–1338, Dec. 2015.

[159] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Adv. Neural Inf. Process. Syst.,* vol. 30, 2017.

[160] C. Finn, P. Abbeel, and S. Levine, "Model-Agnostic Meta-Learning for fast adaptation of deep networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1126–1135, PMLR, 2017.

[161] K. Neklyudov, D. Molchanov, A. Ashukha, and D. Vetrov, "Variance networks: When expectation does not meet your expectations," *arXiv:1803.03764,* Mar. 2018.

[162] S. Wang and C. Manning, "Fast dropout training," in *Proceedings of the 30th International Conference on Machine Learning* (S. Dasgupta and

D. McAllester, eds.), vol. 28 of *Proceedings of Machine Learning Research*, (Atlanta, Georgia, USA), pp. 118–126, PMLR, 2013.

[163] M. E. Tipping, "Sparse bayesian learning and the relevance vector machine." `https://www.jmlr.org/papers/volume1/tipping01a/tipping01a.pdf?ref=https://githubhelp.com`, 2001. Accessed: 2023-3-8.

[164] S. D. Babacan, R. Molina, M. N. Do, and A. K. Katsaggelos, "Bayesian blind deconvolution with general sparse image priors," in *Computer Vision – ECCV 2012*, pp. 341–355, Springer Berlin Heidelberg, 2012.

[165] W. Neiswanger, C. Wang, and E. Xing, "Asymptotically exact, embarrassingly parallel MCMC," *arXiv:1311.4780*, Nov. 2013.

[166] A. Fallah, A. Mokhtari, and A. Ozdaglar, "On the convergence theory of Gradient-Based Model-Agnostic Meta-Learning algorithms," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics* (S. Chiappa and R. Calandra, eds.), vol. 108 of *Proceedings of Machine Learning Research*, pp. 1082–1092, PMLR, 2020.

[167] S. Caldas, S. M. K. Duddu, P. Wu, T. Li, J. Konečný, H. Brendan McMahan, V. Smith, and A. Talwalkar, "LEAF: A benchmark for federated settings," *arXiv:1812.01097*, Dec. 2018.

[168] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429–450, Mar. 2020.

[169] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Fed-

erated learning with local and global representations," *arXiv:2001.01523,* Jan. 2020.

[170] Kristiadi, Hein, and Hennig, "Being bayesian, even just a bit, fixes overconfidence in relu networks," *ICML,* 2020.

[171] Weber, Starc, Mittal, and others, "Optimizing over a bayesian last layer," *Third workshop on Bayesian Deep Learning* (*NeurIPS*), 2018.

[172] J. Watson, J. Andreas Lin, P. Klink, J. Pajarinen, and J. Peters, "Latent derivative bayesian last layer networks," in *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics* (A. Banerjee and K. Fukumizu, eds.), vol. 130 of *Proceedings of Machine Learning Research*, pp. 1198–1206, PMLR, 2021.

[173] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *arXiv:1612.00410,* Dec. 2016.

[174] W. Du, X. Zeng, M. Yan, and M. Zhang, "Efficient federated learning via variational dropout," *not published* (*rejected from ICLR*), 2019.

[175] K. Pillutla, K. Malik, A. Mohamed, M. Rabbat, M. Sanjabi, and L. Xiao, "Federated learning with partial model personalization," *arXiv:2204.03809,* Apr. 2022.

[176] Singhal, Sidahmed, Garrett, and others, "Federated reconstruction: Partially local federated learning," *NeurIPS,* 2021.

[177] D. Wen, K.-J. Jeon, and K. Huang, "Federated dropout – a simple approach for enabling federated learning on resource constrained devices," *arXiv:2109.15258,* Sept. 2021.

[178] Cheng, Charles, Garrett, and others, "Does federated dropout actually work?," *CVPR,* 2022.

[179] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proceedings of the 34th International Conference on Machine Learning* (D. Precup and Y. W. Teh, eds.), vol. 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330, PMLR, 2017.

[180] A. Niculescu-Mizil and R. Caruana, "Predicting good probabilities with supervised learning," 2005.

[181] M.-H. Laves, S. Ihler, K.-P. Kortmann, and T. Ortmaier, "Calibration of model uncertainty for dropout variational inference," *arXiv:2006.11584,* June 2020.

[182] S. Li and C. Zong, "Multi-domain sentiment classification," in *Proceedings of ACL-08: HLT, Short Papers*, (Columbus, Ohio), pp. 257–260, Association for Computational Linguistics, June 2008.

[183] Y. Yang and T. M. Hospedales, "A unified perspective on Multi-Domain and Multi-Task learning," *arXiv:1412.7489,* Dec. 2014.

[184] H. Bilen and A. Vedaldi, "Universal representations:the missing link between faces, text, planktons, and cat breeds," *arXiv:1701.07275,* Jan. 2017.

[185] M. Volpp, F. Flürenbrock, L. Grossberger, C. Daniel, and G. Neumann, "Bayesian context aggregation for neural processes," in *ICLR*, 2020.

[186] J. Gordon, W. P. Bruinsma, A. Y. Foong, J. Requeima, Y. Dubois, and R. E. Turner, "Convolutional conditional neural processes," in *ICLR*, 2020.

[187] A. Y. Foong, W. P. Bruinsma, J. Gordon, Y. Dubois, J. Requeima, and R. E. Turner, "Meta-learning stationary stochastic process prediction with convolutional neural processes," in *NeurIPS*, 2020.

[188] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, *et al.*, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165,* 2020.

[189] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *arXiv preprint arXiv:1611.01144,* 2016.

[190] C. J. Maddison, A. Mnih, and Y. W. Teh, "The concrete distribution: A continuous relaxation of discrete random variables," *arXiv preprint arXiv:1611.00712,* 2016.

[191] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531,* 2015.

[192] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv preprint arXiv:2002.06440,* 2020.

[193] I. S. Jeon, D. Kang, and S. I. Yoo, "Blind image deconvolution using student's-t prior with overlapping group sparsity," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing* (*ICASSP*), pp. 1817–1821, Mar. 2017.

[194] A. Krizhevsky, "Learning multiple layers of features from tiny images." `http://www.cs.utoronto.ca/~kriz/learning-features-2009-TR.pdf`. Accessed: 2023-3-14.

[195] G. Cohen, S. Afshar, J. Tapson, and A. van Schaik, "EMNIST: Extending MNIST to handwritten letters," in *2017 International Joint Conference on Neural Networks* (*IJCNN*), pp. 2921–2926, May 2017.

[196] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of the IEEE international conference on computer vision*, pp. 3730–3738, 2015.

[197] N. Lee, T. Ajanthan, and P. H. S. Torr, "SNIP: Single-shot network pruning based on connection sensitivity," *arXiv:1810.02340,* Oct. 2018.

[198] Y. Jiang, S. Wang, V. Valls, B. J. Ko, W.-H. Lee, K. K. Leung, and L. Tassiulas, "Model pruning enables efficient federated learning on edge devices," 2022.

[199] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, (New York, NY, USA), pp. 2623–2631, Association for Computing Machinery, July 2019.

Appendix A

Application in Few-shot Learning

A.1 Additional Experiments on 1D Few-shot Regression

Setup. We explored the 1D function regression on the data generated from the synthetic GPs with varying kernels[1] in the previous work [33, 34, 55] that is suitable for measuring the uncertainty adaptation. For the baseline models, the Neural Process model (NP) and NP with additional deterministic representation (NP+CNP) described in [33, 34, 55] are compared. We also adopted the variational prior (VP) into the representation-based posterior of NP and its variants (NP+VP and NP+CNP+VP). For all models (including our NVDP), we used the same *fixed variance* and *learned variance* likelihood architecture depicted in [55]: the agent NN with 4 hidden layers of 128 units with LeLU activation [57] and an output layer of 1 unit for the mean (or an additional 1 unit for variance). The dimensions of the set representation r^t were fixed to 128. The meta NNs in the conditional *dropout* posterior in NVDPs have 4 hidden layers

[1] `https://github.com/deepmind/neural-processes`

of 128 units with LeakyReLU and an output layer of 257 units (i.e. $K+D+1$) for each layer of the agent model. All models were trained with Adam optimizer [121] with a learning rate 5e-4 for 0.5 million iterations. We draw 16 functions (a batch) from GPs at each iteration. Specifically, at every training step, we draw 16 functions $f(\cdot)$ from GP prior with the squared-exponential kernels, $k(x, x') = \sigma_f^2 \exp(-(x - x')^2/2l^2)$, generated with length-scale $l \sim U(0.1, 0.6)$ and function noise level $\sigma_f \sim U(0.1, 1.0)$. Then, x values was uniformly sampled from $[-2, 2]$, and corresponding y value was determined by the randomly drawn function (i.e. $y = f(x), f \sim \mathcal{GP}$). And the task data points are split into a disjoint sample of m contexts and n targets as $m \sim U(3, 97)$ and $n \sim U[m+1, 100)$, respectively. In the test or validation, the numbers of contexts and targets were chosen as $m \sim U(3, 97)$ and $n = 400 - m$, respectively. 50000 functions were sampled from GPs to compute the log-likelihood (LL) and other scores for validation.

Implementation. The architecture of NVDP model is as follows: **(Deterministic) Feature Encoder.** $r(\mathbf{x}_s^t, \mathbf{y}_s^t) : 2 \times |C| \underbrace{\xrightarrow{\text{lin+relu}} 128 \times |\mathcal{C}|}_{\text{6 times}} \xrightarrow{\text{mean}} 128$.

Decoder (Agent). $f(x_i) : 1 \underbrace{\xrightarrow{\text{lin+relu}} 128}_{\text{4 times}} \xrightarrow{\text{lin+relu}} 2 \xrightarrow{\text{split}} (\mu, \sigma)$ (where $\sigma = 0.1 + 0.9 \cdot \text{softplus(logstd)}$).

Meta Model. $g^{(l)}(r) : d_r \underbrace{\xrightarrow{\text{lin+leakyReLu}} 128}_{\text{4 times}} \xrightarrow{\text{lin+leakyReLu}} (K^{(l)}, D^{(l)}, 1) \xrightarrow{\text{split}} (\mathbf{a}, \mathbf{b}, \mathbf{c})$

where $g^{(l)}(r)$ is the meta NN for the l-th layer of the decoder, and $K^{(l)} \times D^{(l)}$ is the number of parameters in the l-th layer.

GP Dataset		CNP
Fixed	LL	$-0.94(\pm 0.05)$
Variance	RLL	$-0.93(\pm 0.01)$
	PLL	$-0.94(\pm 0.05)$
Learned	LL	$0.72(\pm 0.54)$
Variance	RLL	$1.03(\pm 0.38)$
	PLL	$0.69(\pm 0.53)$

Table A.1: An additional summary of the 1D regression with the GP with random kernel dataset. The deterministic baseline (CNP) is presented. We could observe that the performance of the CNP is close to or slightly better than the NP+CNP in Tables 1 of the manuscript. However, the CNP model could lose the functional variability as shown in Figure 2.

A.2 Additional Experiments on Few-shot Image Completion Tasks

Setup. The image completion tasks are performed to validate the performance of the models in more complex function spaces [33, 34, 55] Here, we treat the image samples from MNIST [122] and CelebA [123] as unknown functions. The task is to predict a mapping from normalized 2D pixel coordinates x_i ($\in [0,1]^2$) to pixel intensities y_i ($\in \mathbb{R}^1$ for greyscale, $\in \mathbb{R}^3$ for RGB) given some context points. For 2D regression experiments, we used the same *learned variance* baselines implemented in the GP data regression task, except the input and output of the decoder are changed according to dataset, e.g., $x_i \in [0,1]^2$, and $y_i \in \mathcal{R}^1$ for MNIST (or $\in \mathcal{R}^3$ and $r^t = 1024$ for CelebA). At each iteration, the images in the training set are split into S context and N target points (e.g., $S \sim U(3, 197)$, $n \sim U[N+1, 200)$ at train and $S \sim U(3, 197)$, $N = 784 - S$ at validation). Adam optimizer with a learning rate 4e-4 and 16 batches with 300 epochs were used for training. The validation is performed on the separated validation images set. To see the generalization performance on a completely

new dataset, we also tested the models trained on MNIST to the Omniglot validation set.

Image Dataset		CNP
MNIST	LL	0.87(±0.16)
	RLL	1.14(±0.08)
	PLL	0.83(±0.15)
MNIST	LL	0.68(±0.11)
to	RLL	0.99(±0.08)
Omniglot	PLL	0.64(±0.13)
CelebA	LL	0.78(±0.15)
	RLL	0.93(±0.05)
	PLL	0.77(±0.15)

Table A.2: An additional summary of the 2D image completion tasks on the MNIST, CelebA, and Omniglot dataset. The deterministic baseline (CNP) is presented. We could observe that the performance of the CNP is close to or slightly better than the NP+CNP in Tables 2 of the manuscript. However, the CNP model could lose the functional variability as shown in Figure 4.

A.3 Additional Details on a Trignomy Dataset

Implementation. For the small NVDP model on 1D function regression with a Trigonometry dataset, the following architecture was employed:

(Deterministic) Feature Encoder. $r(\mathbf{x}_s^t, \mathbf{y}_s^t) : 2 \times |C| \underbrace{\overset{\text{lin+relu}}{\longrightarrow} 12 \times |\mathcal{C}|}_{\text{6 times}} \overset{\text{mean}}{\longrightarrow}$ 12.

Decoder (Agent). $f(x_i, r) : (1 + d_r) \underbrace{\overset{\text{lin+relu}}{\longrightarrow} 12}_{\text{2 times}} \overset{\text{lin+relu}}{\longrightarrow} 2 \overset{\text{split}}{\longrightarrow} (\mu, \sigma)$ (where $\sigma = 0.1 + 0.9 \cdot$ softplus(logstd)).

Meta Model. $g^{(l)}(r) : d_r \underbrace{\overset{\text{lin+Mish}}{\longrightarrow} 12}_{\text{4 times}} \overset{\text{lin+Mish}}{\longrightarrow} (K^{(l)}, D^{(l)}, 1) \overset{\text{split}}{\longrightarrow} (\mathbf{a}, \mathbf{b}, \mathbf{c})$

where $g^{(l}(r)$ is the meta NN for the l-th layer of the decoder, and $K^{(l)} \times D^{(l)}$ is the number of parameters in the l-th layer.

A.4 Additional Implementation Details

Gumbel-Sigmoid. The Gumbel-Sigmoid function is derived from the Gumbel-Softmax distribution originally suggested for continuous relaxation for discrete random variables [189, 190]. Given two logits: a and 0, the Gumbel-Sigmoid function can be derived as follows:

$$\text{Gumbel-Sigmoid}(a;\tau) = \text{Gumbel-Softmax}((a,0);\tau)[0] \tag{A.1}$$

$$= \frac{\exp((a+g1)/\tau)}{\exp((a+g1)/\tau) + exp((0+g2)/\tau)} = \frac{1}{1+\exp(-(a+(g1-g2))/\tau)} \tag{A.2}$$

$$\text{where} \quad g1, g2 \sim -\log(-\log(\text{Uniform}(0,1)))$$

where $g1$ and $g2$ are two samples from Gumbel(0,1) distribution. For low temperatures (e.g., $\tau \to 0$), the expected value of a Gumbel-Softmax random variable approaches the expected value of a categorical random variable with the same logits. As the temperature increases (e.g., $\tau \to \infty$), the expected value converges to a uniform distribution.

While training MetaVD, adapting the Gumbel-Sigmoid function with the temperature (e.g., $\tau > 1$) in the calculation of the hierarchical prior (or variational prior) improved the training stability. As we mentioned in the manuscript, this makes the training of the meta NN much more flexible.

Softmax Relaxation. Although the Gumbel-Softmax distribution is originally suggested for relaxation for discrete random variables [189, 190], another way to view the Gumbel-Softmax is a smoothing the "softmax" function with the temperature τ. This can be more easily identifiable if we only consider the relaxed softmax function without the random Gumbel noises:

$$\text{Softmax}((a1,a2);\tau) = \Big[\frac{\exp(a1/\tau)}{\exp(a1/\tau) + exp(a2/\tau)}, \frac{\exp(a2/\tau)}{\exp(a1/\tau) + exp(a2/\tau)}\Big] \tag{A.3}$$

Using the temperature of τ, the softmax is applied on the scaled logits (i.e., logits/τ). When the temperature is 1, the softmax is directly applied on the unscaled logits. Thus, the temperature τ can control the softness of softmax in the prediction of the probability distribution over classes.

Relation to Knowledge Distillation Softmax relaxation is also well known as a technique for distillation [191]. Hinton et al. [191] adapted the relaxed softmax to train a distilled model from the knowledge in an ensemble model. Specifically, the soft class targets (or probabilities) produced by the ensemble model with the relaxed softmax are used to train the distilled model. Since the soft targets have high entropy, they tend to provide much more information than hard targets and much less variance in the gradient estimation. Similarly, the relaxed sigmoid with τ is adopted in the meta NN of NVDPs for reducing the variance in the estimation of the dropout rate p while training. In terms of distillation perspective, the KL divergence between the variational posterior and the (variational) prior in NVDPs can be seen as transferring knowledge of the dropout structure of the parameters. Also, the dropout rate predicted from the target set can act as a teacher network to the prediction of the dropout rate predicted from the context set.

In the experiment of NVDPs, we employed the Gumbel-Sigmoid since it allows stronger exploration than the relaxed sigmoid due to the random Gumbel noises. However, training NVDPs with the relaxed sigmoid without the Gumbel noises was also possible. For the empirical evidence, we provide an additional experimental result in Section 12 of this document. The relaxed sigmoid trick without sampling the noise could be more practical for training a large-scale BNP model.

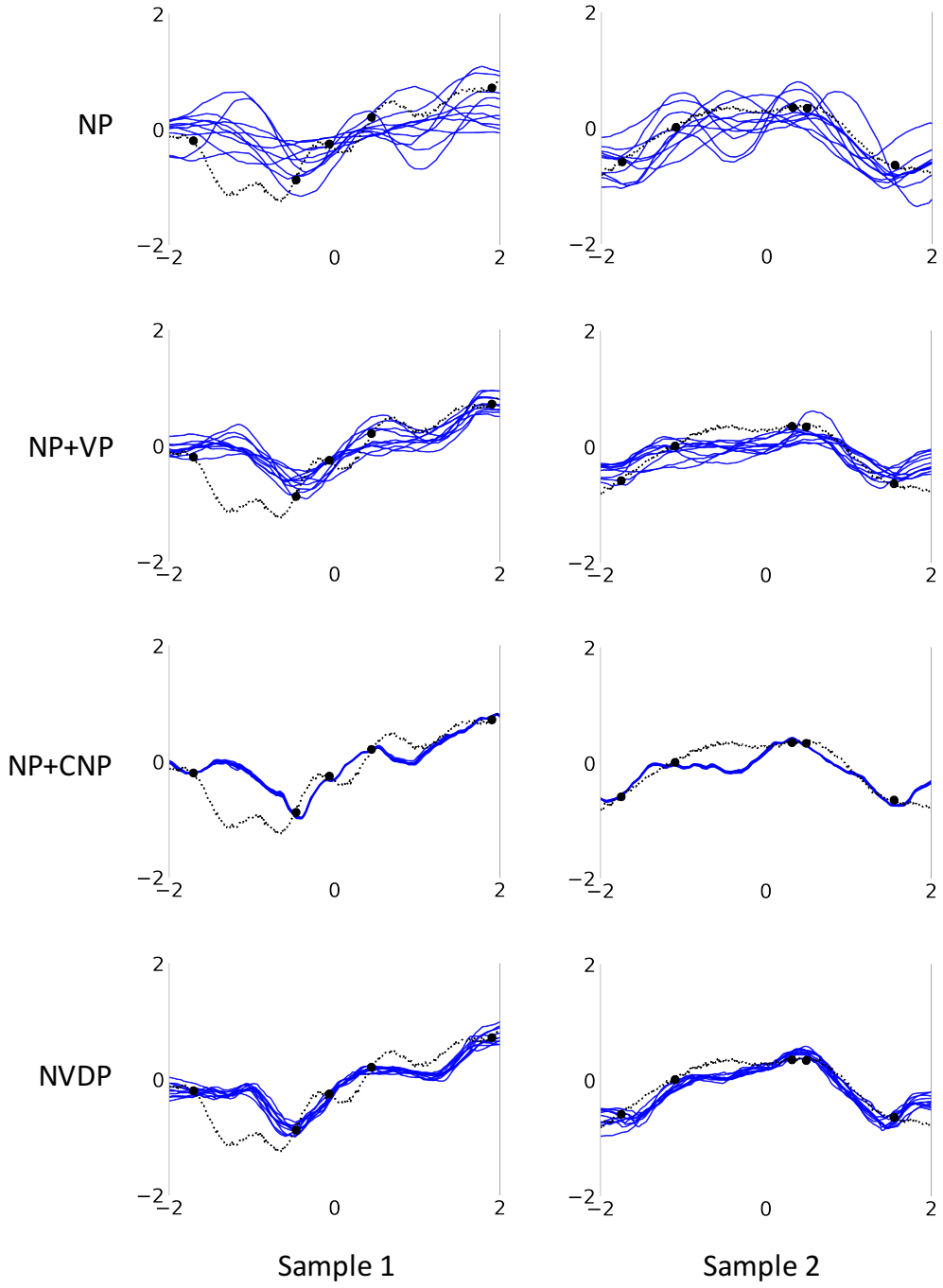


Figure A.1: The additional 1D few-shot regression results of the models on GP dataset in *fixed variance* settings. The black dotted lines represent the true unknown task functions. Black dots are a few context points ($S = 5$) given to the posteriors. The blue lines are mean values predicted from the sampled NNs.

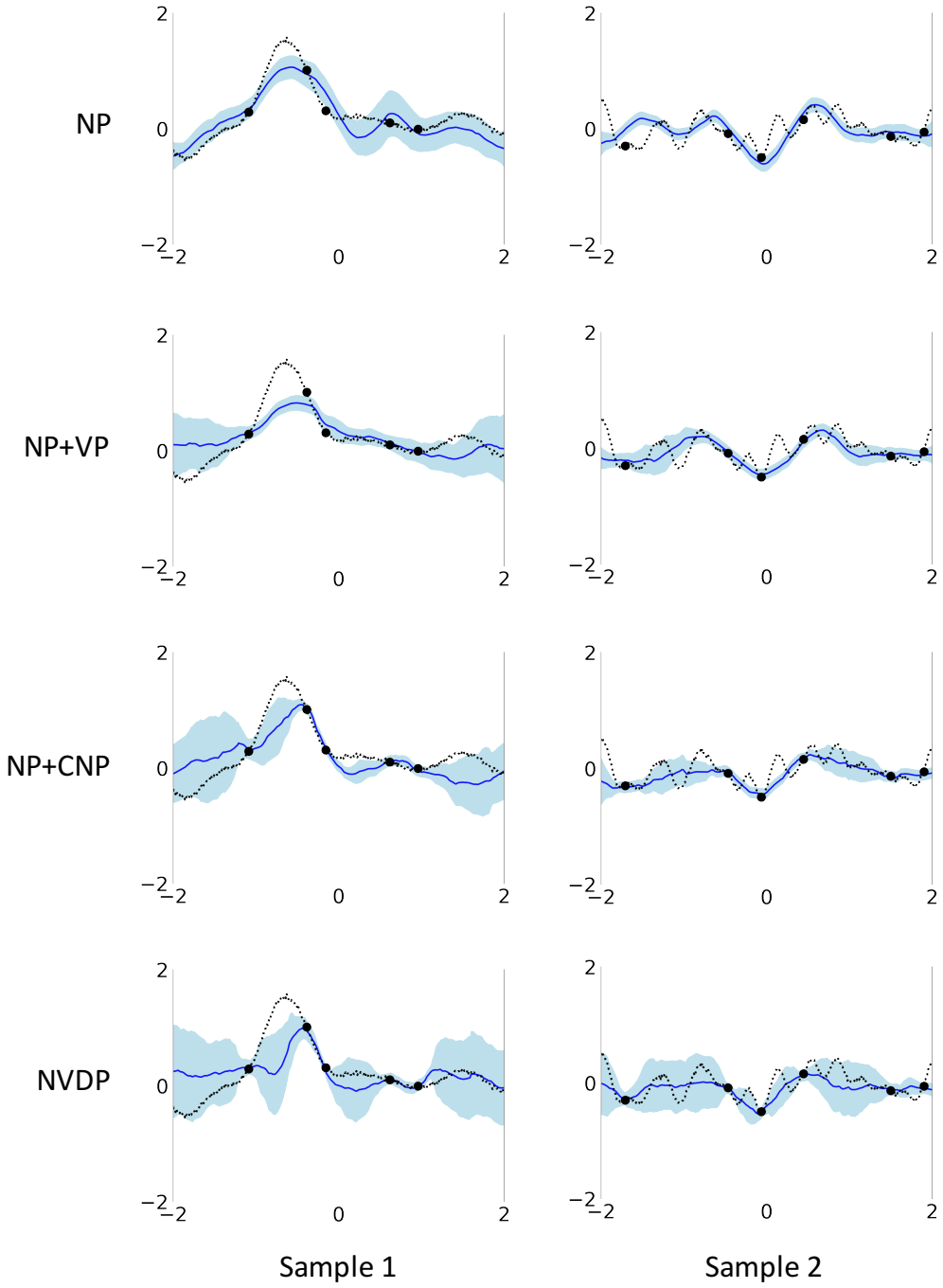


Figure A.2: The additional 1D few-shot regression results of the models on GP dataset in *learned variance* settings. The black (dash-line) represents the true unknown task function. Black dots are a few context points ($S = 5$) given to the posteriors. The blue lines and light blue area are mean values and variance predicted from the sampled NNs, respectively.

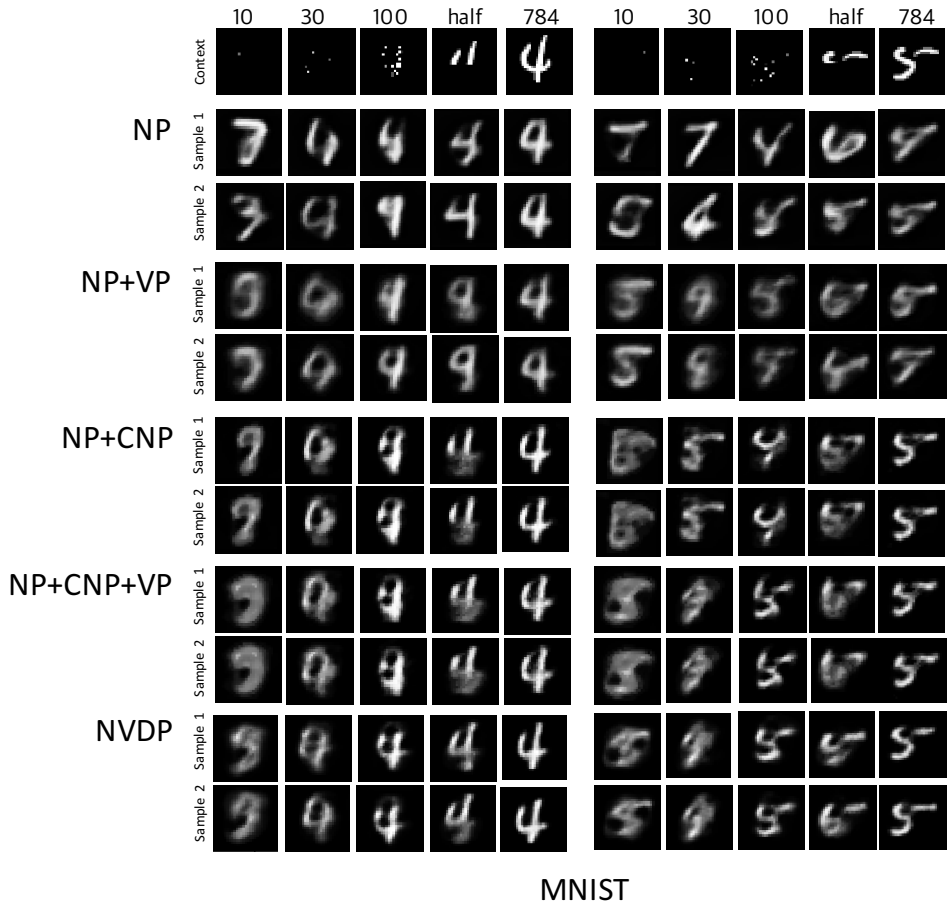


Figure A.3: The additional results from the 2D image completion tasks on the MNIST dataset. Given the observed context points (10, 30, 100, half, and full pixels), the mean values of two independently sampled functions from the models (i.e. NP, NP+CNP, and NVDP (ours)) are presented.

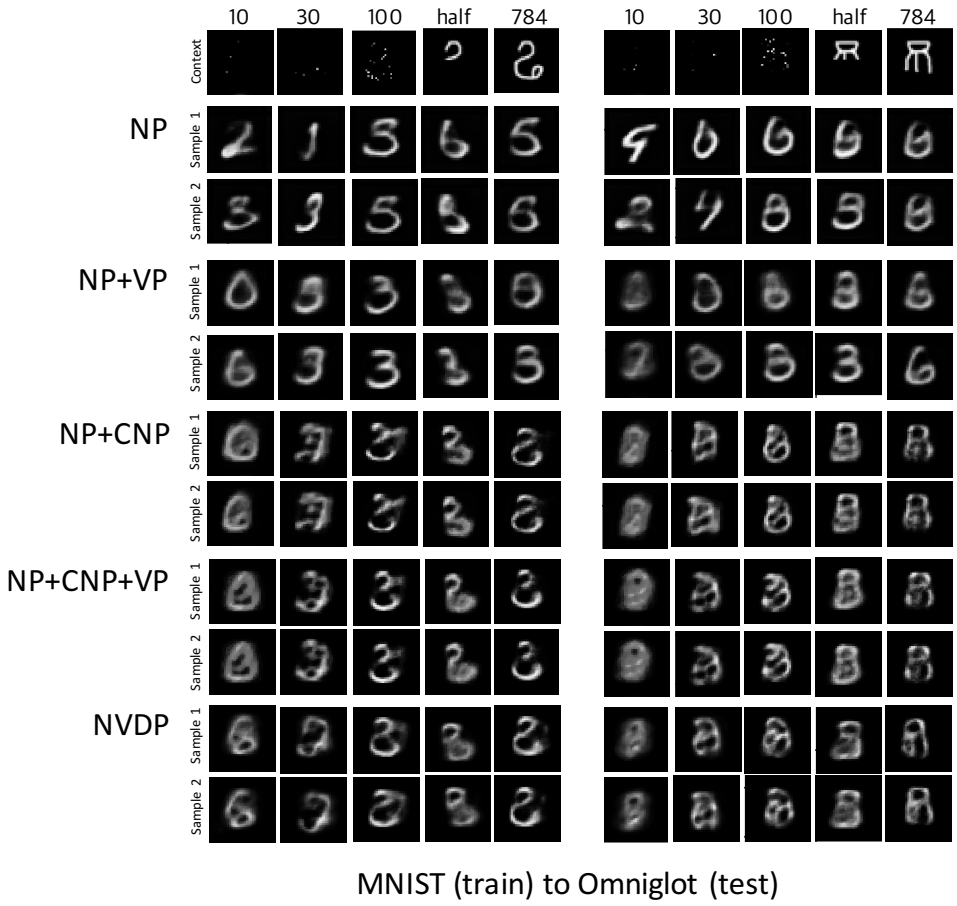


Figure A.4: The additional results from the 2D image completion tasks on the Omniglot dataset. Given the observed context points (10, 30, 100, half, and full pixels), the mean values of two independently sampled functions from the models (i.e. NP, NP+CNP, and NVDP (ours)) are presented.

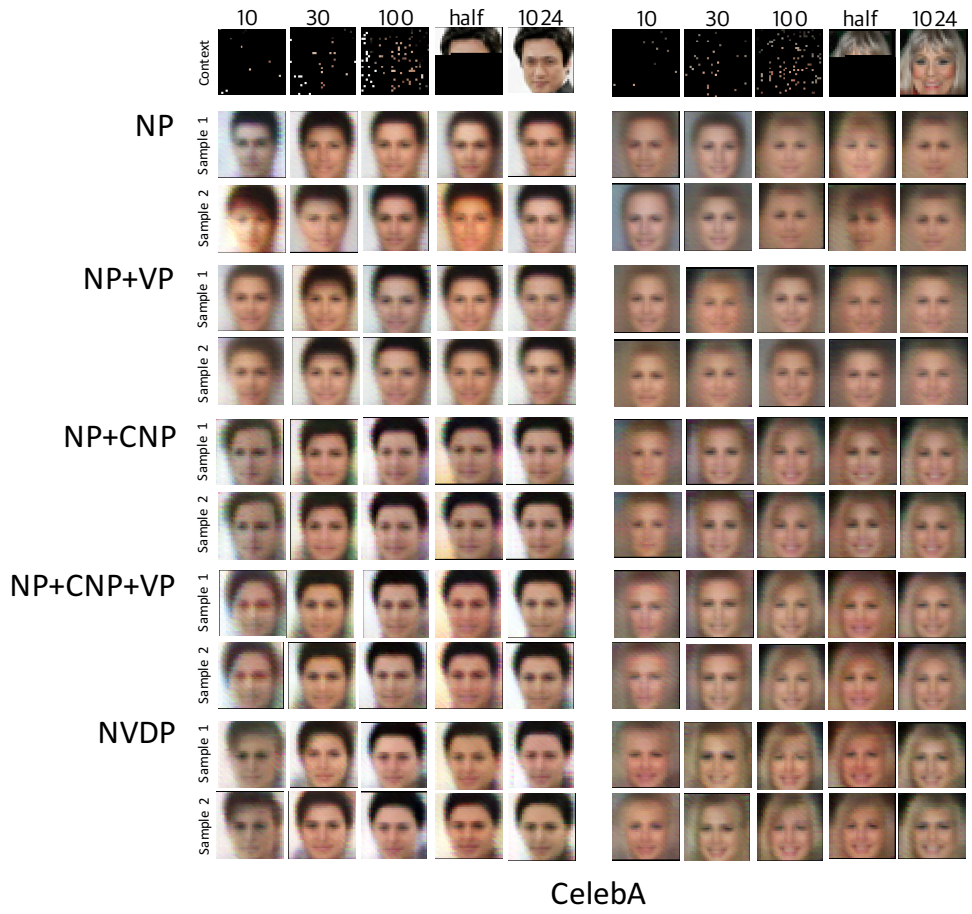


Figure A.5: The additional results from the 2D image completion tasks on the CelebA dataset. Given the observed context points (10, 30, 100, half, and full pixels), the mean values of two independently sampled functions from the models (i.e. NP, NP+CNP, and NVDP (ours)) are presented.

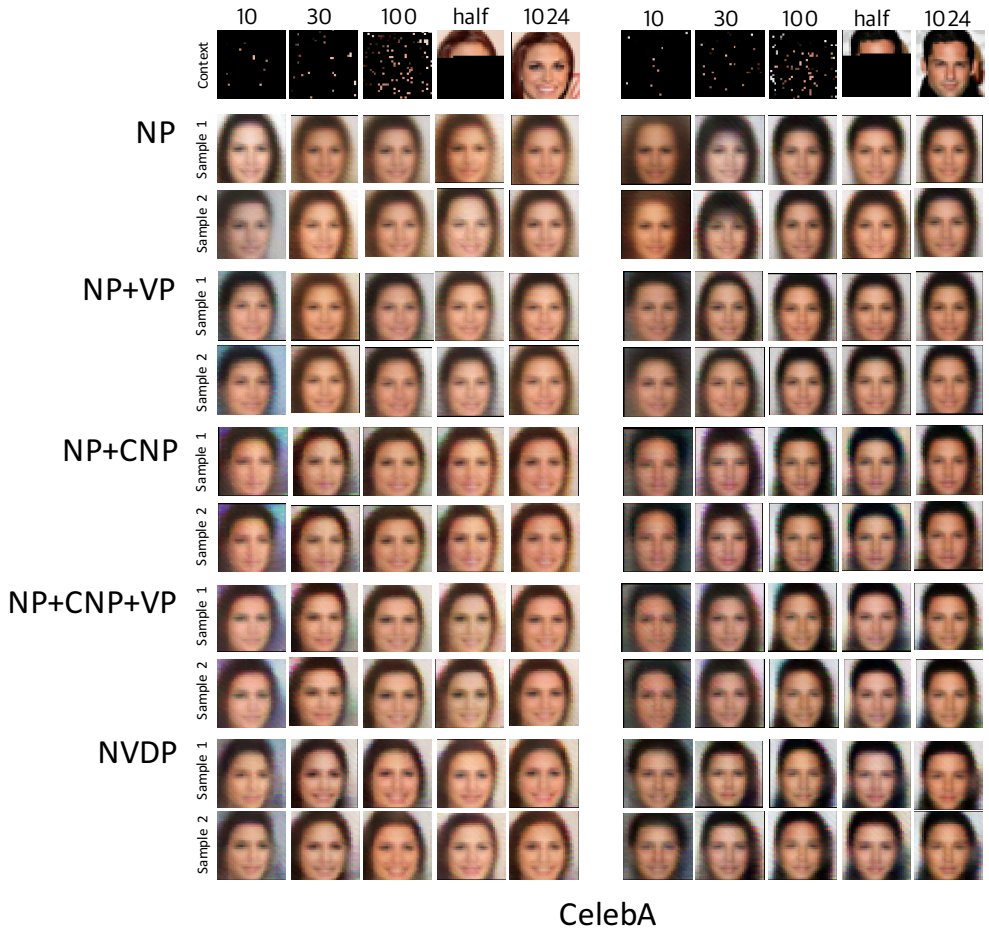


Figure A.6: The additional results from the 2D image completion tasks on the CelebA dataset. Given the observed context points (10, 30, 100, half, and full pixels), the mean values of two independently sampled functions from the models (i.e. NP, NP+CNP, and NVDP (ours)) are presented.

Appendix B

Application in Federated Learning

B.1 Additional Background

The Global Posterior Decomposition in Bayesian FL In Bayesian estimation, an alternative to Maximum Likelihood Estimation (MLE) is the inference or estimation of the posterior distribution of the parameters given all the data, denoted as $p(w|\mathcal{D} \equiv \mathcal{D}^1 \cup \cdots \cup \mathcal{D}^M)$. This posterior distribution is proportional to the product of the likelihoods and a prior, $p(w|\mathcal{D}) \propto p(w) \prod_{m=1}^{M} p(\mathcal{D}^m|w)$. In the case of a uniform prior, the modes of the global posterior coincide with the MLE solutions. This establishes an equivalence between the inference of the posterior mode and optimization. Under the uniform prior, any global posterior distribution that exists decomposes into a product of local posteriors:

$$P(\theta|\mathcal{D}) \propto \prod_{m=1}^{M} p(\theta|\mathcal{D}^m) \tag{B.1}$$

This proposition is well discussed in the recent Bayesian FL work [153]. In the FL context, the global model can be computed in the server by multiplicatively

aggregating the local models adapted to each client. However, in practice, aggregating the posterior with the overfitted local models into the global one is difficult due to the heterogeneity among clients' data and the permutation invariance property of the NN architecture [192]. Thus, frequent communications between the server and the client are still demanded in the FL. The challenge of inferencing the local and global model and improving communication efficiency remains an active research area for real federated applications.

Connection of Posterior Aggregation to FedAvg. In the Bayesian federated learning (BFL) objective, the multiplicative decomposition allows more accurate global posterior predictions by taking into account the uncertainties of local posteriors. Intuitively, local parameters with lower uncertainty will have smaller weights than those parameters with higher uncertainty in the global parameter aggregation of equation 5.5. A similar posterior decomposition has been successfully applied to scale Monte Carlo methods facilitating embarrassingly parallel posterior approximation for large datasets [165, 81], although those were not deep neural network-based approaches.

In fact, the posterior decomposition view also exhibits a direct connection to the FedAvg algorithm [125]. Assuming a uniform prior $p(w)$ across all clients, we can derive the MAP objective of Bayesian FL based on local log-likelihoods.

$$\arg\max_{w} \log p(w|\mathcal{D}) = \arg\max_{w} \log(\prod_{m \in S} p(w|D^m)) \tag{B.2}$$

$$= \arg\max_{w} \Big\{ \log p(w) + \sum_{m=1}^{M} \mathcal{L}^m(\mathcal{D}^m; w) \Big\} \tag{B.3}$$

$$\text{where} \;\; \mathcal{L}^m(\mathcal{D}^m; w) = \log p(\mathcal{D}^m|w) \tag{B.4}$$

where $\log p(\mathcal{D}^m|w)$ is log-likelihood on local data $\mathcal{D}^m$. This is due to the Bayes rule, which states that the posterior is proportional to the product of the like-

lihood and the prior. We can actually treat m't clients empirical risk $\mathcal{L}^m(w)$ in the equation 5.1 as the local log-likelihood in the equation B.2. Assuming Gaussian local distribution $p(\mathcal{D}^m|w) = \frac{1}{\sqrt{2\sigma^m}} \exp(-\frac{||w-w^m||^2}{2(\sigma^m)^2})$, the local log-likelihood equals to $\mathcal{L}_m(\mathcal{D}^m; w) = C_i - \frac{1}{\sqrt{2\sigma^m}}||w - w^m||^2$. Additionally, with the same covariance (*e.g.* ., $\sigma^m = \sigma$) for all clients, the optimization problem becomes

$$\arg\max_{\theta} \log(\prod_{m \in S} p(w|D^m)) = -\frac{1}{2\sigma^2} \sum_{m \in S} ||w - w^m||^2 + C. \tag{B.5}$$

This is equivalent to minimizing a sum of squares. This provides one justification of the arithmetic mean $\bar{w} = \sum_{m=1}^{M} w^m$ as a global approximation in FedAvg.

The Motivation of Hierarchical Prior in MetaVD. In 5.3.3, we adopt the hierarchical prior [163, 164, 193] proposed in [52] for several reasons. The first reason is simply that it is a well-posed Bayesian prior that can avoid a degenerate posterior problem of the conventional VD prior [44, 43, 51]. The second reason is that we want a sparse prior to reduce communication costs by compressing the model; the hierarchical prior has been proven effective for parameter pruning. Although we briefly mentioned the KL divergence term as $\mathrm{KL}(q(w^m; \phi)||p(w^m)) = \sum_{k=1}^{K} 0.5 \log(1 + (\alpha_k^m)^{-1})$ in the manuscript for readability, here we provide more detailed descriptions of applying the hierarchical prior.

Under the hierarchical prior assumption, we consider the joint prior and joint posterior distributions [1]. The joint prior, $p(w^m, \gamma^m) = p(w^m|\gamma^m)p(\gamma^m)$, is defined as a combination of a zero-mean Gaussian distribution, $p(w^m|\gamma^m) = \mathcal{N}(w^m|0, \gamma^m)$, and a uniform hyper-prior, $p(\gamma^m) = \mathcal{U}(\gamma^m|a, b)$, over the variance. Then, we define a (conditional) joint variational posterior, $q(w^m, \gamma^m|\phi) =$

$q(w^m|\phi)q(\gamma^m)$, comprising the (conditional) dropout posterior $q(w^m; \phi = (\theta, \phi, e^m))$ and an additional Dirac delta distribution, $q(\gamma^m)$, to approximate the true (joint) posterior $p(w^m, \gamma^m|\mathcal{D}^m)$ (given the client's dataset $\mathcal{D}^m$).

ELBO. With the hierarchical prior $p(w^m, \gamma^m) = p(w^m|\gamma^m)p(\gamma^m)$ and the (conditional) joint posterior $q(w^m, \gamma^m|\phi) = q(w^m|\phi)q(\gamma^m)$ of the MetaVD, we can derive the local objective for each client as follows:

$$\begin{aligned}
&\mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m|\mathcal{D}^m)) = \int q(w^m, \gamma^m|\phi) \log \frac{q(w^m, \gamma^m|\phi)}{p(w^m, \gamma^m|x^m, y^m)} \partial w^m \partial \gamma^m \\
&= \int q(w^m, \gamma^m|\phi) \log \frac{q(w^m, \gamma^m|\phi)p(y^m|x^m)}{p(y^m|x^m, w^m)p(w^m, \gamma^m)} \partial w^m \partial \gamma^m \qquad \text{(B.6)} \\
&= \int q(w^m, \gamma^m|\phi) \Big\{ \log \frac{q(w^m, \gamma^m|\phi)}{p(w^m, \gamma^m)} + \log p(y^m|x^m) - \log p(y^m|x^m, w^m) \Big\} \partial w^m \partial \gamma^m \\
&= \mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m)) + \log p(y^m|x^m) - \mathbb{E}_{q(w^m, \gamma^m|\phi)}[\log p(y^m|x^m, w^m)]. \qquad \text{(B.7)}
\end{aligned}$$

Eq. (2) is derived from Bayes' rule: $p(w^m, \gamma^m|\mathcal{D}^m) = \frac{p(y^m|x^m, w^m)p(w^m, \gamma^m)}{p(y^m|x^m)}$. By reordering Eq.(3), we get

$$\log p(y^m|x^m) \geq \mathbb{E}_{q(w^m, \gamma^m|\phi)}[\log p(y^m|x^m, w^m)] - \mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m)) \qquad \text{(B.8)}$$

$$\begin{aligned}
&= \mathbb{E}_{q(w^m|\phi)}\mathbb{E}_{q(\gamma^m)}[\log p(y^m|x^m, w^m)] - \mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m)) \\
&= \mathbb{E}_{q(w^m|\phi)}[\log p(y^m|x^m, w^m)] - \mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m)) \qquad \text{(B.9)}
\end{aligned}$$

The lower-bound in Eq.(4) is due to the positivity of the $\mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m|\mathcal{D}^m))$. Here, Eq. (5) corresponds to the ELBO objective of Eq. (3) in the manuscript.

KL term. If we further decompose the KL divergence term in Eq. (5),

$$\begin{aligned}
&\mathrm{KL}(q(w^m, \gamma^m|\phi)||p(w^m, \gamma^m)) = \mathrm{KL}(q(w^m|\phi)||p(w^m|\gamma^m)) + \mathrm{KL}(q(\gamma^m)||p(\gamma^m)) \\
&= \sum_{k=1}^{K} \{0.5 \log(1 + (\alpha_k^m)^{-1})\} + \sum_{k=1}^{K} \{\log(b - a)\} \qquad \text{(B.10)}
\end{aligned}$$

The $\log(b-a)$ is independent of the unknown variables α^m, θ, and γ^m. Thus, we do not need to specify the value of hyperparameters a and b and can neglect them in practice[1]. Eq. (6) provides the rationale behind the KL divergence term. In fact, the independence between the KL term and the parameter θ ensures compatibility with other optimization meta-learning algorithms. Also, the two-level structure in a hierarchical system can generate a much more complex distribution, expanding the potential solution spaces for selecting feasible prior. Thus, the hierarchical prior is a suitable prior for interpreting the variety of different clients' models in the FL environment. The same hierarchical prior is uniformly applied across all $1...M$ clients to ensure the global posterior decomposition assumption.

B.2 Dataset and Methods

We follow the datasets and the evaluation protocol of pFL-Bench [141] which is a recently proposed benchmark for federated learning.

Datasets. Here, we present descriptions of the dataset used in our experiment.

- The **CIFAR-10** and **CIFAR-100** datasets [194] are popular for 10-class and 100-class image classification respectively. Each dataset contains 50,000 training and 10,000 test images with a resolution of 32x32 pixels. Following the heterogeneous partition manners used in [141], we use Dirichlet allocation to split this dataset into 130 clients with different Dirichlet factors as $\dot{\alpha}$ = [5, 0.5, 0.1] (a smaller $\dot{\alpha}$ indicates a higher heterogeneous degree).

[1]For a more detailed proof of this, please see the appendix section of [52]

- The Federated Extended MNIST (**FEMNIST**) is a widely used FL dataset for 62-class handwritten character recognition [167]. The original FEMNIST dataset contains 3,550 clients and each client corresponds to a character writer from EMNIST [195]. Following [141], we adopt the sub-sampled version, which contains 400 clients and a total of 85350 training and 21536 test images with a resolution of 28x28 pixels.

- The **CelebA** is a FL dataset based on [196] for 2-class image classification; Smiling or Not. Following [141], we adopt the sub-sampled version, which contains 500 clients and a total of 8752 training and 2347 test images with a resolution of 84x84 pixels. Each client is assigned images of a single celebrity.

We randomly select 30 clients for CIFAR-10 and CIFAR-100 and 40 clients for FEMNIST as OOD clients who do not participate in the FL processes. For all of the datasets, we follow the same heterogeneous patterns exhibited in the pFL-Bench, which covers a wide range of scales, partition manners, and non-i.i.d degrees. This enables comprehensive comparisons and analysis among different methods in the non-i.i.d. data environment.

Baselines. We present an overview of the baseline models used in our experiment, covering various popular and state-of-the-art approaches across three categories: Non-PFL, meta-learning-based PFL, and Bayesian FL methods.

The following **Non-PFL methods** are considered in our experiments:

- FedAvg [125] is a standard FL algorithm that averages gradients weighted by the data size of clients in each FL round.

- FedProx [168] employs a proximal term to encourage updated local models for clients not to deviate too much from the global model.

The following (meta-learning-based) **PFL methods** are considered in our experiment:

- Reptile [145] inserts a meta-learning fine-tuning phase of [26, 160] after the federated averaging algorithm stage to provide a reliable personalized model.
- MAML [144] enhances federated learning by integrating a MAML-based meta-learner [160], dividing the dataset into support and query sets for more robust local training.
- PerFedAvg [146] method focuses on the convergence analysis of the HF-MAML algorithm [166] in the FL scenario, offering a provably convergent method based on MAML to tackle non-convex functions.

The following **Bayesian FL** algorithm are also compared in our experiment:

- FedBE [151] enhances robust aggregation by adopting a Bayesian inference approach, sampling high-quality global models, and combining them through Bayesian model ensemble using Gaussian or Dirichlet distributions fitted to local models.

The following **Pruning** algorithm is also compared in our compression experiment in the appendix:

- SNIP [197, 198] algorithm is a deep learning pruning technique that identifies and removes less important connections in neural networks before the training begins using a small subset of the dataset. It is compared with our algorithm in the model compression experiment in the FL environment.

Additionally, Fine-tuning (FT) in the baseline's name indicates fine-tuning the local models with a few steps before evaluation within the FL processes, which is similar to the adaptation step of the optimization-based meta-learning.

B.3 Implementation Details

Models. To maintain consistency with previous research, we employ the widely adopted CNN model for all algorithms and baselines [46, 169, 127]. Specifically, the global model comprises three convolutional layers with 64 filters and 3x3 kernels, followed by three fully-connected layers of 256, 128, and 64 hidden units.

The hypernetwork architecture of MetaVD consists of an embedding layer, followed by two consecutive blocks containing a linear layer and a LeakyReLU activation function, and one block containing a linear layer with exponential activation to output the dropout logit parameter α. The dimension of client embedding e^m is proportional to the number of clients M and is calculated as $(1 + M/4)$. The hidden units' size in the hypernetwork was set to 200. The predicted dropout logit parameter is then applied to each weight of the MetaVD layer within the global model. In our study, we selectively apply MetaVD to just one fully-connected layer right before the output layer of the global model. This simple adaptation of MetaVD only in one fully-connected layer yielded significant performance improvements across all experiments.

Hyperparameters. For all datasets, we set T to 1000 to ensure sufficient convergence following conventions [141]. The batch size was set to 64, and local steps was set to 5. Personalization was executed with a batch size of 64 and a 1-step update. In order to ensure a fair comparison between the algorithms, the results presented in all of our experiments are obtained with the

optimal hyperparameters for each model. To do so, we conducted an extensive parameter optimization using an optimization tool called Optuna[2] [199]. We employed both the Tree-structured Parzen Estimator algorithm and Random Sampler as hyperparameter samplers in Optuna.

For all methods, we investigated the server learning rate and local SGD learning rate within identical ranges. The server learning rate η was explored within the range of $[0.6, 0.7, 0.8, 0.9, 1.0]$. The local SGD learning rate was investigated within the range of $[0.005, 0.01, 0.015, 0.02, 0.025, 0.03]$. In MAML and PerFedAvg, an additional client learning rate γ is required, for which we searched within the range of $[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1]$. For MetaVD, an additional KL divergence parameter β is needed, and we sought its optimal value within the range of $[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]$. To ensure the reproducibility of the experiments, we will release all code, including baselines, on our GitHub repository.

Experiment Outline. To evaluate the effectiveness and robustness of the proposed PFL methods, we tested the algorithms under various FL scenarios, including:

- Degree of Data Heterogeneity: we assessed the performance of each algorithm in scenarios where data from different clients are heterogeneous, considering factors such as variations in data distributions, label imbalance, and situations while some clients have limited training data available.

- Ablation Study: we conducted an ablation study to verify the advantages of employing a hypernetwork in MetaVD compared to naive Variational Dropout (VD) or Ensemble VD approaches in Federated Learning.

[2]https://optuna.org/

- Client Participation: we evaluated the performance of each algorithm under different levels of client participation rates in each FL round.
- Uncertainty Calibration: we used Expected Calibration Error (ECE) measures to evaluate the accuracy of probabilistic predictions made by the predictive models. These measures help determine whether a model is overconfident or underconfident in its predictions and identify any biases in the model's predictions.
- Multi-domain Datasets: Multi-domain learning aims to leverage all available training data across different domains to enhance the performance of the model, but it typically results in a suboptimal global model. We tested our approach in FL with multi-domain learning.
- Model Compression: we compared MetaVD' s compression capabilities with the existing pruning algorithm and evaluated the performance of MetaVD with and without compression.

B.4 Additional Results with Non-i.i.d. Settings

To evaluate the generalization capabilities of the models under non-i.i.d. data conditions, we conducted tests on both CIFAR-10 and CIFAR-100 datasets with varying degrees of heterogeneous partitions. We randomly selected 30 out of 130 clients as non-participating out-of-distribution (OOD) clients who were not involved during the training phase. Table B.4 and Table B.5 display the weighted average accuracy while adjusting the non-i.i.d. degrees with different Dirichlet factors $\dot{\alpha}$. A smaller $\dot{\alpha}$ corresponds to a higher degree of heterogeneity. Test(%) denotes the weighted average accuracy of participating clients' test samples, OOD(%) signifies the weighted average accuracy of the

Hetrogenity	*CIFAR-100* dataset $\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	42.35	43.08	+0.73	41.92	41.96	+0.04	37.57	37.90	+0.33
FedAvg+FT [141]	41.49	42.45	+0.96	40.99	39.83	−1.16	36.79	37.00	+0.21
FedProx [168]	42.23	44.11	+1.88	42.03	40.51	−1.52	38.07	39.36	+1.29
FedBE [151]	45.17	45.43	+0.26	44.29	44.23	−0.06	40.89	41.56	+0.67
Reptile [145]	47.87	47.73	−0.14	46.13	45.94	−0.19	40.71	43.59	+2.88
MAML [144]	48.30	49.14	+0.84	46.33	46.65	**+0.32**	41.56	45.19	+3.63
PerFedAvg (HF-MAML) [146]	48.19	47.35	−0.84	46.22	46.36	+0.14	42.46	42.92	+0.46
FedAvg + MetaVD (ours)	47.82	50.26	**+2.44**	47.54	47.55	+0.01	42.34	42.65	+0.31
Reptile + MetaVD (ours)	**53.71**	**54.50**	+0.79	**52.06**	**51.50**	−0.56	**44.56**	44.62	+0.06
MAML + MetaVD (ours)	52.40	51.78	−0.62	50.21	49.75	−0.46	43.84	**48.17**	**+4.33**
PerFedAvg + MetaVD (ours)	51.67	51.70	+0.03	50.02	48.70	−1.32	43.31	46.19	+2.88

Table B.1: Results of classification accuracies with different (non-i.i.d.) heterogeneity strengths of $\dot{\alpha} = 5.0$ and $\dot{\alpha} = 0.5$ and $\dot{\alpha} = 0.1$ in the CIFAR-100 dataset. We report the results of participating clients during the training (Test) dataset and non-participating clients (OOD). The higher, the better.

Hetrogenity	*CIFAR-10* dataset $\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	80.73	79.98	−0.75	78.28	78.90	+0.62	71.65	71.57	−0.08
FedAvg+FT [141]	80.20	79.63	−0.57	76.96	75.83	−1.13	69.62	68.38	−1.24
FedProx [168]	80.79	80.30	−0.49	78.55	77.60	−0.95	72.27	73.75	+1.48
FedBE [151]	81.55	81.07	−0.48	79.44	79.46	+0.02	70.23	69.19	−1.04
Reptile [145]	83.48	82.88	−0.60	79.35	79.41	+0.06	73.93	76.36	+2.43
MAML [144]	82.61	81.69	−0.92	80.31	80.03	−0.28	76.06	74.89	−1.17
PerFedAvg (HF-MAML) [146]	82.57	81.99	−0.58	80.87	80.85	−0.02	75.42	79.56	+4.14
FedAvg + MetaVD (ours)	82.60	82.74	+0.14	79.08	80.33	**+1.25**	76.87	76.25	−0.62
Reptile + MetaVD (ours)	**84.70**	84.28	−0.42	**83.20**	**83.40**	+0.20	76.51	**82.07**	+5.56
MAML + MetaVD (ours)	83.93	**84.95**	**+1.02**	81.32	81.81	+0.49	**77.27**	79.05	+1.78
PerFedAvg + MetaVD (ours)	83.88	83.96	+0.08	81.06	81.47	+0.41	76.06	81.77	**+5.71**

Table B.2: Results of classification accuracies with different (non-i.i.d.) heterogeneity strengths of $\dot{\alpha} = 5.0$ and $\dot{\alpha} = 0.5$ and $\dot{\alpha} = 0.1$ in the CIFAR-10 dataset. We report the results of participating clients during the training (Test) dataset and non-participating clients (OOD). The higher the better.

non-participants, and Δ represents the participation generalization gap calculated as the difference between OOD accuracy and test accuracy.

In both datasets, PFL methods such as Reptile, MAML, and PerFedAvg generally outperform non-PFL methods like FedAvg and FedProx. The Bayesian ensemble approach on FedAvg (FedBE) offers a slight improvement in overall performance but still lags behind PFL methods in terms of OOD accuracy. Notably, when combined with MetaVD, all baselines experience significant performance enhancements, regardless of whether they employ FL or PFL approaches. Importantly, our method consistently improves OOD accuracy across all degrees of non-i.i.d. data, illustrating its versatility in effectively augmenting conventional FL algorithms without being limited by specific optimization techniques, addressing non-i.i.d. data and model overfitting issues.

B.5 Additional Ablation Results in FMNIST Dataset

	FEMNIST dataset		
Method	Test (%)	OOD (%)	Δ
Reptile	87.86	88.22	−0.36
Reptile+VD	87.93	85.88	**+2.05**
Reptile+EnsembleVD	87.99	87.97	+0.02
Reptile+MetaVD	**89.43**	**88.71**	+0.72

Table B.3: MetaVD ablation study results in the FEMNIST dataset.

In this experiment, we further performed the ablation study using the FEMNIST dataset to evaluate the benefits of MetaVD's hypernetwork in FL. We compared MetaVD to naive VD [44] and Ensemble VD [174] approaches. The naive (global) VD model keeps one global dropout parameter shared with all clients; the dropout parameter is treated as a global model parameter as in FedAvg. In EnsembleVD, we updated client-specific dropout rates in a manner

analogous to the local adaptation step of MetaVD but maintained independent variational dropout rates for different clients on the server. In contrast, MetaVD utilized a hypernetwork to learn the dropout rates. All models employed Bayesian posterior aggregation rules based on dropout rates to update the global model parameter.

The ablation study's results on the FEMNIST dataset are outlined in Table B.3 in the manuscript. It is evident from the table that MetaVD's conditional variational dropout-based hypernetwork surpasses all other baselines in classification accuracy. Here too, Reptile+MetaVD consistently outperforms other methods. We noticed that in baselines like VD or EnsembleVD, client-specific dropout rates were not fully optimized due to restricted client participation. Consequently, the initial dropout rates and KL divergence loss in VD and EnsembleVD remained fairly static. Conversely, in MetaVD, both dropout rates and KL divergence loss converged in all tests. This observation underscores that MetaVD's hypernetwork presents a more data-efficient approach to learning the client-specific model uncertainty compared to other baselines.

B.6 Additional Results on Uncertainty Calibration

Addressing uncertainty calibration issues is crucial for enhancing the accuracy and reliability of a model's predictions, particularly when the model is employed for making significant decisions. Uncertainty calibration metrics evaluate the accuracy of probabilistic predictions made by a predictive model. By identifying and addressing any calibration issues, it becomes possible to improve the model's prediction accuracy and reliability, which is of essential importance in decision-making processes. In a Federated Learning environment where clients have limited non-i.i.d. data, overfitting can easily occur, making the calibration of prediction uncertainty even more critical. We conducted experiments to

evaluate the effectiveness of our proposed algorithm in improving uncertainty calibration, demonstrating its potential to strengthen model performance in various real applications.

Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) are widely used metrics for measuring uncertainty calibration. ECE represents the average discrepancy between the model's confidence and its accuracy. To compute ECE, we group samples based on their confidence levels and calculate the average confidence and the percentage of correct samples for each group. ECE is then derived as the weighted average of the differences between the average confidence and the percentage of correct samples across all groups. ECE values range from 0 to 1, where 0 signifies perfect calibration and 1 indicates complete miscalibration. MCE, on the other hand, is akin to ECE but focuses on the largest gap for any group rather than the weighted average.

Table B.4 and Table B.5 summarize the ECE and MCE results for varying non-i.i.d. degrees in CIFAR-100 and CIFAR-10 datasets, with Dirichlet factors represented as $\dot{\alpha} = [5, 0.5, 0.1]$. Typically, ECE and MCE values tend to increase as clients possess more non-i.i.d. data, which is a result of overfitting each client's local data. In the CIFAR-10 dataset, all methods that incorporate MetaVD exhibit a decline in ECE and MCE values as $\dot{\alpha}$ decreases, in contrast to standard FL or PFL methods that show an increase in these values. MetaVD effectively mitigates overfitting to local data and enhances calibration by leveraging the heterogeneity present in the training data. Consequently, methods employing MetaVD consistently achieve the lowest ECE and MCE values in almost all scenarios.

Examining a reliability diagram can provide a visual comparison of uncertainty calibration results, as it illustrates the relationship between prediction probabilities and true labels. Ideally, if a model predicts a specific class with a

Hetrogenity	*CIFAR-100* dataset $\dot{\alpha} = 5.0$		$\dot{\alpha} = 0.5$		$\dot{\alpha} = 0.1$	
Method	ECE (%)	MCE (%)	ECE (%)	MCE (%)	ECE (%)	MCE (%)
FedAvg [125]	0.29	22.05	0.53	32.35	0.60	36.79
FedAvg+FT [141]	0.41	28.36	0.46	31.91	0.69	45.04
FedProx [168]	0.43	21.92	0.40	26.74	0.67	39.69
FedBE [151]	0.43	29.75	0.38	27.54	0.50	34.66
Reptile [145]	0.73	46.53	0.77	46.19	0.77	50.52
MAML [144]	0.40	27.74	0.70	48.16	0.75	46.47
PerFedAvg (HF-MAML) [146]	0.65	45.44	0.40	**23.74**	0.69	45.27
FedAvg+MetaVD (ours)	0.26	**19.99**	**0.32**	26.60	**0.39**	**25.27**
Reptile+MetaVD (ours)	0.37	25.24	0.32	23.86	0.57	42.40
MAML+MetaVD (ours)	0.31	23.18	0.38	32.56	0.52	37.26
PerFedAvg+MetaVD (ours)	**0.26**	20.46	0.39	27.59	0.43	30.20

Table B.4: Results of uncertainty calibration scores (ECE and MCE) in the CIFAR-100 dataset. The lower is the better.

Hetrogenity	*CIFAR-10* dataset $\dot{\alpha} = 5.0$		$\dot{\alpha} = 0.5$		$\dot{\alpha} = 0.1$	
Method	ECE (%)	MCE (%)	ECE (%)	MCE (%)	ECE (%)	MCE (%)
FedAvg [125]	1.93	28.87	2.39	26.45	3.16	27.37
FedAvg+FT [141]	2.02	20.80	2.71	27.86	3.20	33.22
FedProx [168]	2.01	25.96	2.52	25.24	3.06	28.62
FedBE [151]	1.78	21.33	2.18	21.89	3.35	29.72
Reptile [145]	2.40	30.53	3.06	38.89	2.75	27.76
MAML [144]	2.61	36.86	2.49	30.67	2.66	32.10
PerFedAvg (HF-MAML) [146]	2.19	26.78	2.39	27.13	2.58	24.47
FedAvg+MetaVD (ours)	1.52	18.02	1.76	21.17	1.40	13.92
Reptile+MetaVD (ours)	1.60	25.46	2.16	29.84	1.76	17.84
MAML+MetaVD (ours)	1.39	**16.46**	1.97	21.51	0.35	5.80
PerFedAvg+MetaVD (ours)	**1.14**	17.55	**1.51**	**20.38**	**0.29**	**3.63**

Table B.5: Results of uncertainty calibration scores (ECE and MCE) in the CIFAR-10 dataset. The lower is the better.

certain probability, the actual label should correspond to that confidence level, placing the point on the reliability diagram's diagonal line. However, if the prediction probability and the true label do not align, the point would be below or above the diagonal line. In this context, ECE represents the average distance of each point from the diagonal line, while MCE indicates the maximum distance of any point from the diagonal line, offering a comprehensive understanding of the model's calibration performance.

Figure B.1 and Figure B.2 present the reliability diagrams for CIFAR-100 and CIFAR-10 with $\dot{\alpha} = 0.5$ respectively. These figures allow us to compare the calibration performance of FedAvg, Reptile, MAML, and PerFedAvg, both with and without the integration of MetaVD. Remarkably, in most instances, employing MetaVD leads to improved calibration, drawing the reliability diagrams closer to the identity function. This supports our findings that MetaVD significantly improves the uncertainty calibration of the models as well.

B.7 Additional Results on Muti-domain Datasets

Unlike existing federated learning algorithms that usually assume a single-domain approach where only one dataset is used in the experiment. In this section, we further evaluate the performance of our method on real-world non i.i.d. FL experiments by introducing multi-domain datasets in which we assume each client can have data from different domains. Multi-domain learning [182, 183, 184] aims to leverage all available training data across different domains to enhance the performance of the model. However, directly utilizing data from different domains typically results in a suboptimal global model. In the context of federated learning, it becomes even more critical to apply multi-domain learning effectively. We use three different FL datasets to construct the multi-domain task distributions: CelebA, FEMNIST, and CIFAR-100. The

Hetrogenity	*CelebA + CIFAR-100* dataset								
	$\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	43.42	44.45	−1.02	43.65	43.45	**+0.20**	40.46	41.00	−0.54
Reptile [145]	50.40	49.07	+1.33	48.92	48.93	−0.01	43.41	42.16	**+1.24**
MAML [144]	49.97	48.40	**+1.58**	47.39	48.51	−1.11	44.80	44.79	+0.02
PerFedAvg (HF-MAML) [146]	50.61	49.61	+1.00	49.21	50.91	−1.71	44.20	44.13	+0.08
FedAvg+MetaVD (ours)	48.08	48.88	−0.79	48.23	48.62	−0.39	42.98	45.19	−2.21
Reptile+MetaVD (ours)	53.04	53.97	−0.92	**52.26**	**54.75**	−2.49	45.83	47.05	−1.22
MAML+MetaVD (ours)	52.82	53.47	−0.65	51.34	52.82	−1.48	**46.83**	**48.33**	−1.50
PerFedAvg+MetaVD (ours)	**53.14**	**54.26**	−1.12	51.18	53.06	−1.88	46.65	47.11	−0.46

Table B.6: Classification accuracies with different (non-i.i.d.) heterogeneity degrees of $\dot{\alpha} = [5.0, 0.5, 0.1]$ in multi-domain datasets (a); CelebA + CIFAR-100.

Hetrogenity	*CIFAR-100 + FEMNIST* dataset								
	$\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	63.90	53.57	+10.33	64.02	52.88	+11.13	62.13	49.65	**+12.48**
Reptile [145]	66.47	57.60	+8.86	66.13	56.22	+9.91	64.10	55.81	+8.30
MAML [144]	66.86	53.25	**+13.60**	66.56	55.72	+10.84	64.56	56.21	+8.35
PerFedAvg (HF-MAML) [146]	67.47	57.15	+10.32	66.57	55.24	**+11.34**	65.16	56.70	+8.46
FedAvg+MetaVD (ours)	65.91	57.59	+8.32	65.58	56.85	+8.73	63.26	52.78	+10.49
Reptile+MetaVD (ours)	**68.61**	**60.95**	+7.66	**68.35**	59.07	+9.28	65.29	58.11	+7.17
MAML+MetaVD (ours)	68.41	58.86	+9.55	68.24	61.21	+7.03	65.46	58.07	+7.39
PerFedAvg+MetaVD (ours)	68.27	58.72	+9.55	67.93	**61.32**	+6.61	**66.81**	**59.13**	+7.68

Table B.7: Classification accuracies with different (non-i.i.d.) heterogeneity degrees of $\dot{\alpha} = [5.0, 0.5, 0.1]$ in multi-domain datasets (b); CIFAR-100 + FEMNIST.

Hetrogenity	*CelebA + CIFAR-100 + FEMNIST* dataset								
	$\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ	Test (%)	OOD (%)	Δ
FedAvg [125]	63.57	53.10	**+10.48**	63.73	55.55	+8.18	61.86	52.98	+8.88
Reptile [145]	65.93	57.98	+7.95	66.98	57.12	+9.87	63.89	52.33	**+11.56**
MAML [144]	67.70	59.06	+8.64	67.08	58.15	+8.93	65.67	54.41	+11.26
PerFedAvg (HF-MAML) [146]	67.39	59.17	+8.22	67.20	57.03	**+10.17**	65.08	54.54	+10.54
FedAvg+MetaVD (ours)	63.62	55.87	+7.76	65.93	58.58	+7.35	64.59	55.61	+8.97
Reptile+MetaVD (ours)	67.63	60.95	+6.68	68.78	61.59	+7.19	65.18	56.46	+8.71
MAML+MetaVD (ours)	**69.42**	**61.52**	+7.90	**68.81**	**61.60**	+7.21	**66.56**	**56.54**	+10.03
PerFedAvg+MetaVD (ours)	67.83	60.74	+7.09	68.05	61.17	+6.87	65.11	55.64	+9.47

Table B.8: Classification accuracies with different (non-i.i.d.) heterogeneity degrees of $\dot{\alpha} = [5.0, 0.5, 0.1]$ in multi-domain datasets (d); CelebA + CIFAR-100 + FEMNIST.

non-i.i.d heterogeneous environment has also been assumed in the field of multi-domain learning. We utilize the Dirichlet sampling technique ($\dot{\alpha} = [5.0, 0.5, 0.1]$) to sample each client's local CIFAR-100 data.

Table B.6, B.7, and B.8 illustrate the classification accuracies for Test and OOD clients with varying degrees of heterogeneity. Employing MetaVD leads to significantly improved prediction accuracy, with larger improvements in OOD accuracy compared to the improvement in Test accuracy, across all multi-domain settings and degrees of heterogeneity. The results indicate that MetaVD can improve the versatility and performance of FL algorithms when applied to a broad range of multi-domain datasets.

B.8 Additional Results on Model Compression

Federated Learning optimization involves frequent communication of model parameters between devices and the central server, which can be slow and may raise privacy concerns. Therefore, it is crucial to minimize communication costs by reducing both the size of exchanged model parameters and the number of communication rounds. MetaVD also has the benefit of compressing the model parameters needed for each client device. To explore the compression capabilities of MetaVD, we conducted experiments on CIFAR-10 and CIFAR-100 datasets, comparing our method to the baseline Pruning algorithm SNIP [197]. For the implementation of the SNIP algorithm in the context of Federated Learning, we referred to the work of Jiang et al. (2022) [198]; we let SNIP prune the original model to the target sparsity right after the first round.

Table B.9 and Table B.10 demonstrate the results of Test(%), OOD(%), and Sparsity(%) percentage of the models with or without MetaVD+DP. Sparsity represents the ratio of zero-valued model parameters in the personalized layer. The DP refers to the process of dropping communication of weights in the FL

Hetrogenity	*CIFAR-10* dataset								
	$\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Sparsity (%)	Test (%)	OOD (%)	Sparsity (%)	Test (%)	OOD (%)	Sparsity (%)
FedAvg+SNIP	81.03	80.46	70.0	78.53	79.30	70.0	71.67	73.13	70.0
Reptile+SNIP	83.21	82.10	70.0	81.43	81.72	70.0	74.56	69.57	70.0
MAML+SNIP	83.35	81.59	70.0	81.47	81.84	70.0	72.51	69.27	70.0
PerFedAvg+SNIP	83.33	83.49	70.0	80.97	82.47	70.0	76.27	75.90	70.0
Reptile+MetaVD	**84.70**	84.28	0	**83.20**	83.40	0	76.51	82.07	0
MAML+MetaVD	83.93	84.95	0	81.32	81.81	0	77.27	79.05	0
PerFedAvg+MetaVD	83.88	83.96	0	81.06	81.47	0	76.06	81.77	0
Reptile+MetaVD+DP	84.19	84.46	77.12	82.67	82.79	79.25	75.85	80.52	76.18
MAML+MetaVD+DP	84.37	**84.95**	76.76	82.65	**83.64**	78.85	**77.77**	80.74	72.47
PerFedAvg+MetaVD+DP	84.04	83.69	78.55	82.64	83.52	79.32	77.10	**83.24**	72.87

Table B.9: Results of study on model compression with different (non-i.i.d.) heterogeneity strengths of $\dot{\alpha} = 5.0$, $\dot{\alpha} = 0.5$, and $\dot{\alpha} = 0.1$ in the CIFAR-10 dataset. MetaVD+DP does not communicate the model parameters when the dropout rate is larger than 0.9.

Hetrogenity	*CIFAR-100* dataset								
	$\dot{\alpha} = 5.0$			$\dot{\alpha} = 0.5$			$\dot{\alpha} = 0.1$		
Method	Test (%)	OOD (%)	Sparsity (%)	Test (%)	OOD (%)	Sparsity (%)	Test (%)	OOD (%)	Sparsity (%)
FedAvg+SNIP	43.76	44.11	70.0	42.53	41.54	70.0	37.80	40.34	70.0
Reptile+SNIP	49.54	49.61	70.0	48.38	48.62	70.0	42.03	43.68	70.0
MAML+SNIP	51.06	49.56	70.0	48.16	46.99	70.0	42.03	42.49	70.0
PerFedAvg+SNIP	50.66	48.85	70.0	48.71	48.46	70.0	43.23	43.09	70.0
Reptile+MetaVD	**53.71**	54.50	0	**52.06**	**51.50**	0	44.56	44.62	0
MAML+MetaVD	52.40	51.78	0	50.21	49.75	0	43.84	**48.17**	0
PerFedAvg+MetaVD	51.67	51.70	0	50.02	48.70	0	43.31	46.19	0
Reptile+MetaVD+DP	52.56	**55.03**	65.83	50.94	50.85	57.26	45.13	45.19	75.72
MAML+MetaVD+DP	52.31	52.29	74.61	50.30	50.88	77.88	**45.75**	45.78	72.26
PerFedAvg+MetaVD+DP	51.68	52.56	70.08	51.64	51.47	68.88	45.16	46.45	55.18

Table B.10: Results of study on model compression with different (non-i.i.d.) heterogeneity strengths of $\dot{\alpha} = 5.0$, $\dot{\alpha} = 0.5$, and $\dot{\alpha} = 0.1$ in the CIFAR-100 dataset. MetaVD+DP does not communicate the model parameters when the dropout rate is larger than 0.9.

algorithm, where weights with a dropout rate greater than 0.9 are dropped after 150 rounds. We have set these 150 rounds to allow MetaVD to adequately learn the client-specific dropout rates. The target sparsity of the SNIP algorithm was set to 0.7. Note that, in our MetaVD+DP setting, even though we dropped weights with a dropout rate greater than 0.9, the actual sparsity values are observed to be lower and tend to be around 70%, as indicated in the tables.

Overall, most models removed over 70% of their parameters in the personalized layer without losing much performance. In CIFAR-10, as heterogeneity increased, the performance gap between SNIP and MetaVD+DP tended to increase. For instance, in CIFAR-10 with $\dot{\alpha} = 5.0$, the OOD performance of SNIP model and MAML+MetaVD+DP model were 81.59 and 84.95, respectively, whereas in CIFAR-10 with $\dot{\alpha} = 0.1$, they were 69.27 and 80.74 respectively, demonstrating an increased difference in performance. This was similar in the case of Reptile+MetaVD+DP and PerFedAvg+MetaVD+DP. In CIFAR-100 dataset, the performance gap, according to the increase in heterogeneity, was not noticeable. Surprisingly, dropping model parameters could lead to performance improvements in some cases. For example, in CIFAR-100 with $\dot{\alpha} = 0.5$, the PerFedAvg+MetaVD+DP model's OOD performance increased from 48.70 to 51.47. This experiment demonstrates that the MetaVD method can decrease the communication cost in Federated Learning (FL) by compressing model parameters. This emphasizes the efficiency of MetaVD, this approach can be applied to numerous FL algorithms without significantly increasing communication costs. It also boosts the generalization performance for new, unseen clients. These aspects make our approach a significant addition to the field of Federated Learning

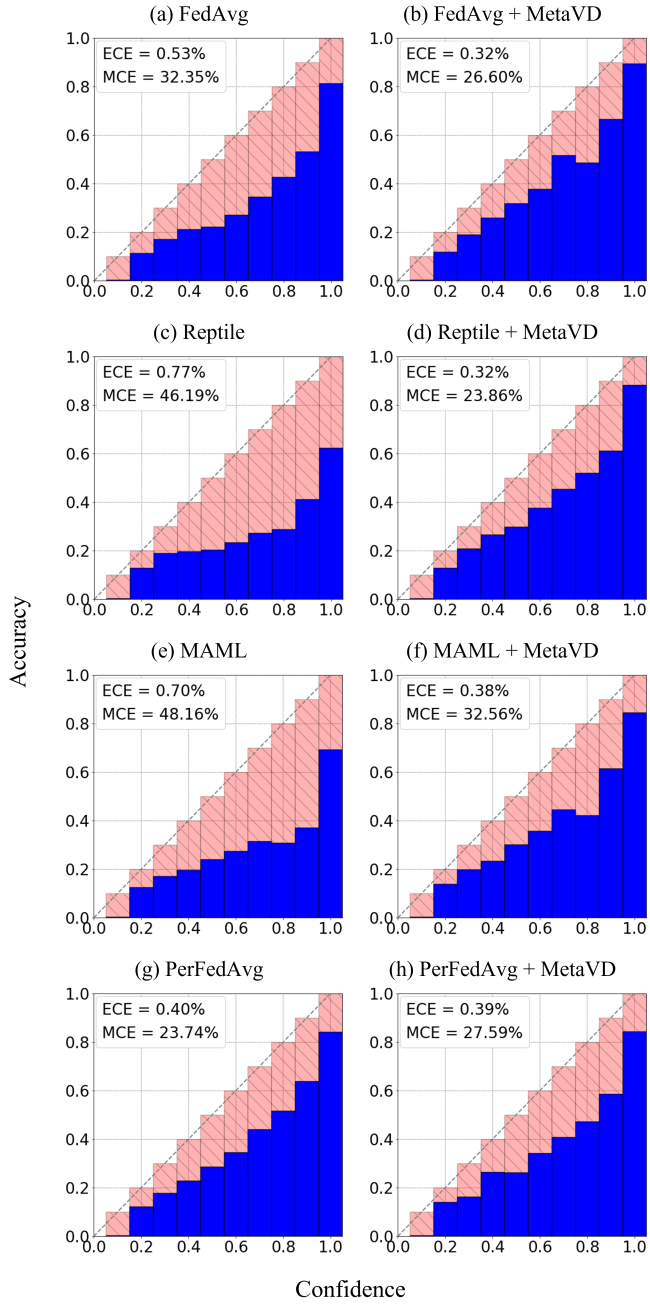


Figure B.1: Reliability diagrams for (a) FedAvg, (b) FedAvg + MetaVD, (c) Reptile, (d) Reptile + MetaVD, (e) MAML, (f) MAML + MetaVD, (g) PerFedAvg and (h) PerFedAvg + MetaVD in CIFAR-100 ($\dot{\alpha} = 0.5$).

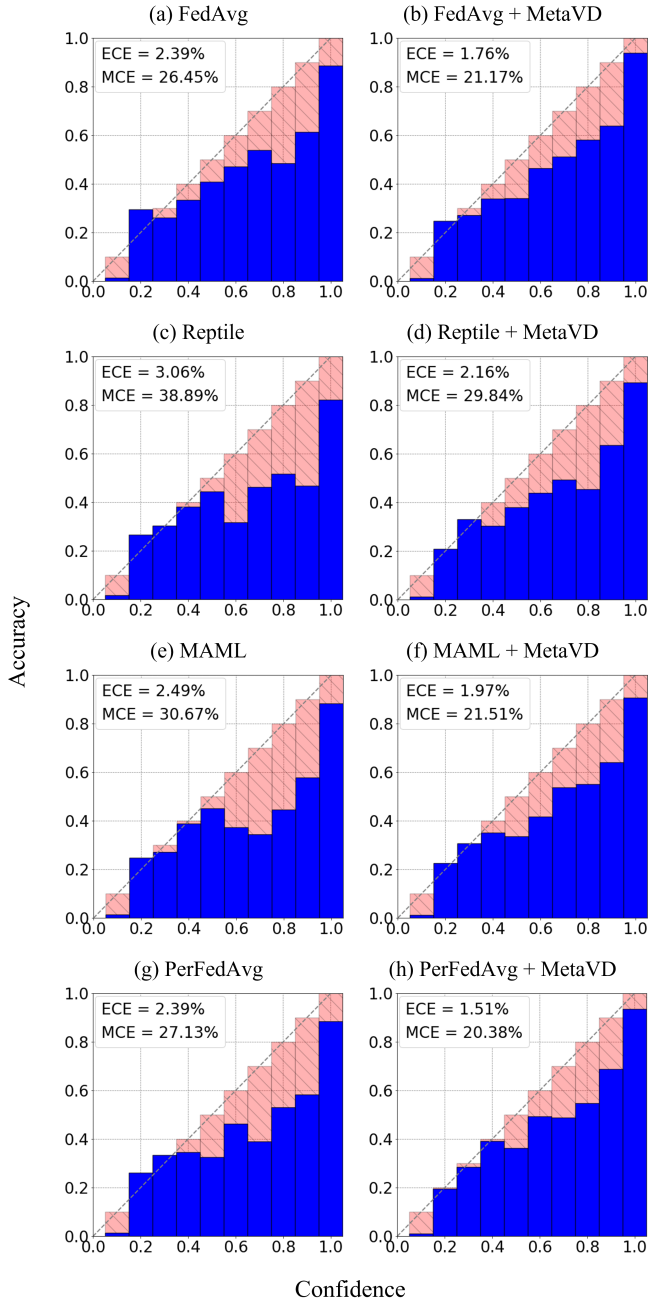


Figure B.2: Reliability diagrams for (a) FedAvg, (b) FedAvg + MetaVD, (c) Reptile, (d) Reptile + MetaVD, (e) MAML, (f) MAML + MetaVD, (g) PerFedAvg and (h) PerFedAvg + MetaVD in CIFAR-10 ($\dot{\alpha} = 0.5$).

요약

메타러닝은 기계학습의 하위 영역으로, 새로운 작업에 빠르게 적응하는 알고리즘 개발을 목표로 한다. 이러한 빠른 적응능력은 개별 작업 학습에 중점을 두는 대신, 학습 과정 자체를 학습하는 메타 학습자를 이용하여 구현할 수 있다. 이 메타 학습자는 기계가 새로운 작업에 효율적으로 적응하는 데 활용될 수 있다. 최근에는 메트릭 기반, 최적화 기반, 모델 기반 등 다양한 메타 학습 전략이 도입되어, 퓨샷 회귀, 퓨샷 분류, 주도적 학습, 강화학습 등 다양한 영역에 적용되고 있다. 그러나 현재의 메타 학습 방식, 특히 메타 학습자는 계산 요구량, 확장성, 모델 과적합 등의 한계가 여전히 존재한다.

본 연구에서는 Meta-Variaitonal Dropout (MetaVD)라는 새로운 베이지안 메타 학습 전략을 제안한다. MetaVD는 하이퍼네트워크를 이용하여 각 신경망 가중치에 대한 작업별 드롭아웃 비율을 추정한다. 이를 통해 다중 작업 환경에서의 데이터 효율적 학습과, 새로운 작업을 위한 전역적인 신경망의 빠른 재구성을 가능하게 한다. 이 프레임워크에서는 저차원 근사와 공유된 변분 사전 해석을 이용하여 드롭아웃 사후모델을 정규화하는 등의 새로운 기술들을 논의한다. MetaVD는 다양한 기존 딥 러닝 알고리즘에 적용 가능한 범용적인 접근법을 제공한다. 제안된 방법론은 1차원 회귀, 이미지 인페인팅, 분류를 포함한 다양한 퓨샷 학습 응용 사례에서 높은 적응 및 일반화 성능을 입증하였다.

연합 학습(FL)은 원격에서 분산된 로컬 클라이언트로부터 글로벌 추론 모델을 학습하는 것을 목표로 하는 기계 학습의 연구 분야이며, 데이터 개인정보 보호의 강화 덕분에 많은 주목을 받고 있다. 그러나 현재의 FL 접근법은 실제 시나리오에서의 모델 과적합과 제한된 비 독립 동일 분포 클라이언트 데이터 등의 문제를 가지고 있다. 이러한 문제를 해결하기 위해, MetaVD를 분산 학습 환경에 적용하기 위해 확장하였다. FL에서의 공유 하이퍼네트워크는 서버에 저장되며,

클라이언트별 드롭아웃 비율을 예측하는 방법을 학습한다. 이를 통해 제한된 비 i.i.d. 데이터 설정에서의 FL 알고리즘의 효과적인 모델 개인화를 가능하게 한다. 또한, 사후 집계를 위한 조건부 드롭아웃 사후 분포도 도입하였다. 본 연구에서는 희소하고 독립 동일 분포가 아닌 다양한 FL 데이터셋을 활용하여 광범위한 실험을 수행하였다. MetaVD는 분포 외 클라이언트에 대해 특히 뛰어난 분류 정확도와 불확실성 보정 성능을 보였다. MetaVD는 각 클라이언트에 필요한 로컬 모델 매개변수를 압축함으로써 모델 과적합을 완화하고 통신 비용을 줄인다. 또한 MetaVD는 다중 도메인 데이터셋을 포함한 FL에서 최첨단 성능을 발휘한다.

전반적으로 이 논문은 퓨샷 학습 및 연합 학습 영역에서의 문제를 해결하기 위한 베이지안 메타 학습 접근법을 위한 포괄적인 프레임워크를 다루었다. 조건부 드롭아웃 사후 모델링은 불확실성 추정 및 보정 외에도 효율적인 모델 적응 및 개인화를 가능하게 한다. 실험 결과는제안된 접근 방식의 뛰어난 성능을 보여주었으며, 이는 실제 시나리오에서의 메타 학습과 응용 분야의 발전에 기여한다.

주요어: 딥러닝, 메타러닝, 베이지안 신경망, 변형 드롭아웃, 멀티태스크 학습, 퓨샷 학습, 연합 학습

학번: 2012-23237

Acknowledgements

I would like to express my earnest gratitude to Prof. Gunhee Kim for his persistent encouragement and valuable advice over the past seven years. I also thank Prof. Suk I. Yoo for his generous guidance during my master's period. I also greatly appreciate my thesis committee –Prof. Sun Kim, Prof. Byoungtak Jang, Prof. Jinwook Seo, and Prof. Sungjun Choi – for their valuable guidance on writing a good academic thesis.

None of my work would have been impossible without my collaborators: Minui Hong, Wonkwang Lee, Junhyeog Yun, Youngjin Park, Myeongjang Pyeon, and Deokyoung Kang. I would also like to mention all the lab colleagues in the SNU Vision and Learning Laboratory: Junhyug Noh, Yunseok Jang, Juyong Kim, Yookoon Park, Byeongchang Kim, Youngjin Kim, Soochan Lee, Wonhee Lee, Jongseok Kim, Minjung Kim, Jaemin Cho, Amelie Schmidt-Colberg, Taeyoung Hahn, Chris Dongjoo Kim, Hyunwoo Kim, Jaekyeom Kim, Jaewoo Ahn, Junsoo Ha, Sungeun Kim, Jinseo Jeong, Heeseung Yun, Sangwoo Moon, Dongyeon Woo, Jihwan Moon, Seokhee Hong, Eunkyu Park, Junseo Koo, Keighley Overbay, Yeda Song, Dayoon Ko, Fatemeh Pesaran and Julian Paquerot. I will never forget the precious time we spent together.