This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

Accelerating Monte Carlo Bayesian Prediction via Approximating Predictive Uncertainty over the Simplex

Yufei Cui, Wuguannan Yao, Qiao Li, Antoni B. Chan, Chun Jason Xue

***Abstract*—Estimating the predictive uncertainty of a Bayesian learning model is critical in various decision-making problems, e.g., reinforcement learning, detecting the adversarial attack, self-driving car. As the model posterior is almost always intractable, most efforts were made on finding an accurate approximation to the true posterior. Even though a decent estimation of the model posterior is obtained, another approximation is required to compute the predictive distribution over the desired output. A common accurate solution is to use Monte Carlo (MC) integration. However, it needs to maintain a large number of samples, evaluate the model repeatedly and average multiple model outputs. In many real-world cases, this is computationally prohibitive. In this work, assuming that the exact posterior or a decent approximation is obtained, we propose a generic framework to approximate the output probability distribution induced by the model posterior with a parameterized model and in an amortized fashion. The aim is to approximate the predictive uncertainty of a specific Bayesian model, meanwhile alleviating the heavy workload of MC integration at testing time. The proposed method is universally applicable to Bayesian classification models that allow for posterior sampling. Theoretically, we show that the idea of amortization incurs no additional costs on approximation performance. Empirical results validate the strong practical performance of our approach.**

I. Introduction

Bayesian inference is a principled method to estimate the uncertainty of probabilistic models. In most applications, especially in deep learning, the likelihood model and model prior are not conjugate hence marginalizing over model prior or posterior cannot be performed analytically, which hinders the practical applicability. For tractability, a simple point estimate such as maximum *a posteriori* (MAP) estimate could be used to approximate the full model posterior. The price paid is the loss of model uncertainty due to incomplete characterization of the model posterior. Approximate inference methods, such as Markov chain Monte Carlo and variational inference, enhance the approximate posterior by a better probability distribution while keeping inference tractability. However, even though a decent approximation of the posterior can be obtained, the computation of predictive distribution is usually intractable due to loss of conjugacy and is of high cost if tractable.

To introduce the problem, we consider a Bayesian classification model trained on dataset $\mathcal{D} = \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1}^N$, where $\mathbf{x}_n \in \mathcal{X}$ and $\mathbf{y}_n \in \mathcal{Y}$ are the nth input and output, respectively. Let the model posterior $p(\boldsymbol{\theta}|\mathcal{D})$ be approximated by Monte Carlo (MC) estimate $\frac{1}{S}\sum_{s=1}^S \delta(\boldsymbol{\theta} - \boldsymbol{\theta}_s)$, and the predictive distribution (a categorical distribution parameterized by the predicted *class probabilities*) is thus approximated by

$$p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{S}\sum_{s=1}^S p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}_s).$$

The predictive distribution can be accurately estimated as $S \to \infty$. However, to perform the computation, we need to maintain a large number of samples, repeatedly evaluate the model for S times and finally average the model outputs. This problem is impractical in many real-world cases. For example, an assisted-driving car system requires an accurate measure of uncertainty to avoid making mistakes with a high confidence [21, 32]. Due to the limited computational resources and storage in such systems, it is hard to maintain a large number of samples and perform S times evaluation of the Bayes model for the real-time image data.

In this work, aiming at boosting the prediction speed while maintaining a rich characterization of the prediction, we propose to approximate the *distribution of class probabilities* over the simplex induced by the model posterior $p(\boldsymbol{\theta}|\mathcal{D})$, in an amortized fashion. This naturally diverts the heavy-load MC integration process from the testing period to the approximation period. Different from the previous work in Bayesian knowledge distillation [3, 5], which only focuses on the output categorical distribution (a point on simplex), the induced distribution over the simplex provides: 1) rich knowledge including prediction confidence for identifying out-of-domain (OOD) data (see empirical examples in Fig. 3); 2) the possibility to use more expressive distributions as the approximate model.

We term the Bayes classifier as "Bayes teacher" and the approximate distribution as "student", due to the analogy with teacher-student learning. A Dirichlet distribution is used as the student due to its expressiveness, conjugacy to categorical distribution, and its efficient reparameterization for training. We propose to explicitly disentangle the parameters of the student into a prediction model (PM) and concentration model (CM), which capture class probability and sharpness of Dirichlet respectively. The CM output can directly be used as a measure for detecting OOD data. We term our method as One-Pass Uncertainty (OPU) as it simplifies real-world evaluation of Bayesian models by computing the predictive distribution with only one model evaluation. Note that, OPU allows choosing various types of student models (e.g., compressed neural network [23, 19, 13]) for further speedup on specific platforms with **no** extra design efforts.

As the amortized approximation of induced distributions

Yufei Cui, Qiao Li, Antoni B. Chan, and Chun Jason Xue are with the Department of Computer Science, City University of Hong Kong. Wuguannan Yao is with the Department of Mathematics, City University of Hong Kong. Corresponding author: Wuguannan Yao.
E-mails: yufeicui92@gmail.com, satie.yao@my.cityu.edu.hk, qiaoli045@gmail.com, abchan@cityu.edu.hk, jasonxue@cityu.edu.hk

is unexplored in the literature, we consider and compare several choices of probability distance measure: forward Kullback–Leibler (KL) divergence, earth mover's distance (EMD), and maximum mean discrepancy (MMD). We theoretically analyze the performance gap incurred by the amortized approximation and show that, under MMD, besides model loss due to the restriction of student distribution, the amortized approximation does not introduce additional loss.

Empirical evaluations show a significant speedup ($\sim 500\times$) of Bayes models. The results on Bayes NN show that OPU performs better in misclassification detection and OOD detection than state-of-the-art works in Bayesian knowledge distillation. It can also be observed that explicit disentangling of mean and concentration helps improve performance. The comparisons of different probability measures validate the theoretical analysis. We also conduct empirical evaluations and comparisons on Bayes logistic regression and Gaussian process, to show OPU is universally applicable to all Bayesian classification models.

The remainder of this paper is organized as follows. In Section II we review related work. ln Section III, we propose our one-pass uncertainty framework. In Sections IV and V, we present experiments applying OPU to traditional Bayesian models and Bayesian neural networks, respectively. Finally, Section VI concludes the paper.

II. Related Work

In this section, we review related works on Bayesian approximations using sampling and fast inference.

A. Approximation via sampling

In this section, we review approximate Bayesian inference using sampling and MC integration. As a simple and effective method for classification, logistic regression has been widely used. However, due to the loss of conjugacy, inference for the Bayesian version is intractable. Usually, a Laplace approximation or variational methods are used to infer posterior distribution [4]. Alternatively, Polson et al. [29] proposed a closed-form Gibbs sampler based on Polya-Gamma (PG) distributed augmenting variable. Finally, the Bayesian predictive distribution is not available in closed-form and usually, MC integration is employed for its approximation.

With the advance of deep neural networks in recent decades, researchers have applied sampling methods to provide tractable inference in Bayesian neural networks. Traditional Monte Carlo methods are batch algorithms, which cannot scale to large datasets. Welling et al. [35] proposed stochastic gradient Langevin dynamics (SGLD), which draws samples approximately from the posterior by adding noise to the stochastic gradient. As the step size decays, SGLD comes to dominate the SGD noise. Chandra et al. [7] improves the convergence and scalability of the MCMC method by introducing a parallel tempering MCMC sampling method. Gal et al. [10] show that a neural network with a dropout layer before a weight layer is mathematically equivalent to an approximation to deep Gaussian process (MCDP). Chandra et al. [6] develops an MCMC Bayesian learning framework to address the uncertainty quantification problem for multi-source data.

Gaussian process (GP) provides flexible modeling capabilities and strong interpretability in machine learning. To seek for a full Bayesian inference of GP, the MCMC method has been applied to GP [9]. It requires computation and inversion of the full covariance matrix at each iteration, which makes it not suitable for large scale problems. Hensman et al. [16] proposed a scalable MCMC method (SGPMC) based on variational inducing points. By providing sufficient inducing points to the model, SGPMC can well approximate full Bayesian inference over GP values and the covariance parameters.

In recent years, there are some works from the variational inference that also proposed to solve sampling for the exponential family, such as [20, 24, 26, 27]. These methods are particularly designed for variational inference, which is not applicable to our problem formulation, and thus are not directly comparable with our method.

B. Approximation for Fast Inference

In this section, related works are reviewed and compared with OPU in terms of methodology. The most related field is *knowledge distillation*, which aims to transfer the predictive knowledge from a larger model to a small one to make the evaluation faster with little or no degradation of the prediction performance. In CompactApprox [31], a parametric model composed of a small subset of "best samples" selected from the original MC samples is used to approximate the full predictive Categorical distribution. Extending the approximation to Bayes NN, BDK [3] proposes to use NN to approximate the predictive Categorical distribution of a Bayes NN trained by stochastic gradient Langevin dynamics (SGLD). Specifically, the teacher network generates samples via SGLD, and KL between the two distributions is minimized in an online fashion. However, the disadvantage is that data uncertainty, model uncertainty, and distributional uncertainty are all entangled in the class probabilities because categorical distributions are of limited expressiveness.

Different from these previous works that only approximate the class probabilities, our OPU approximates the *induced distribution* of class probabilities, which contains richer information including both class probabilities and prediction confidence (e.g., the three types of uncertainty observable via the samples in Fig. 3). We choose a Dirichlet distribution as the student model, and explicitly disentangle the mean and concentration to fully capture the three types of uncertainty. We also explore other probability distance measures (EMD and MMD), showing that KL yields degenerate prediction performance.

Using a Dirichlet to estimate uncertainty has also been explored by Deep Prior Network (DPN) [25], where a parameterized Dirichlet is used in a Bayesian model to characterize the "distributional uncertainty", i.e., to tell if the data is in the training domain or not. However, DPN adds a stochastic layer in the Bayes model, rather than approximating a well-trained Bayes teacher. Due to the intractable inference, DPN uses a MAP estimate of the model posterior which incurs a *loss of*

uncertainty. To compensate for the lost characterization of uncertainty, DPN uses a hand-crafted training goal that explicitly requires OOD examples (which are typically unavailable in real-world applications). In contrast to DPN, our OPU is able to: 1) extract predictive uncertainty from any Bayesian classification model according to the practical requirements; 2) choose various types for student model (e.g., quantized neural network) to enable fast prediction; 3) use only in-domain data in training to get a good uncertainty measure (see Sec. V). Note that **none** of these properties can be achieved by DPN.

In recent years, several works aim to evaluate predictive uncertainty both accurately and fast. Raghu et al. [30] proposes to directly estimate an uncertainty score function for medical opinions. Hendrycks et al. [15] improves the out-of-domain (OOD) detection performance by training on an auxiliary dataset of outliers. Liang et al. [22] proposes Out-of-Domain Image Detection (ODIN) to improve the OOD detection uncertainty by temperature scaling and input preprocessing. Hsu et al. [18] extends ODIN by decomposing the confidence into a dividend/divisor structure, where the "domain" of the data is implicitly learned. While these works improve the performance of uncertainty estimation without losing much inference speed, they either require training on auxiliary data of outliers, which is unrealistic in real applications or implicitly decompose the uncertainty, which is difficult to interpret. Furthermore, these works are only designed for deep neural networks. In contrast, our work requires no additional outlier data during training, and the out-of-domain uncertainty is decomposed in closed-form from the Dirichlet distribution in an intuitive way. Our proposed framework is widely applicable to classical Bayesian models like Bayesian logistic regression, Bayesian neural networks, and non-parametric models like Gaussian process. We compare with some of the works in Sec. V.

III. One-Pass Uncertainty Framework

In this section, we present our one-pass uncertainty (OPU) framework for a generic Bayesian parametric classifier. Our framework is illustrated in Fig. 1. We assume that we have a Bayesian classifier $\mathcal{T}_{\mathbf{x}} = \mathcal{T}(\mathbf{x}; \boldsymbol{\theta})$, parametrized by $\boldsymbol{\theta}$, with posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$. As the posterior is typically intractable, we also assume there is a set of posterior samples $\{\boldsymbol{\theta}_s\}_{s=1}^S$ that represents the model uncertainty. Each posterior sample produces a class probability vector $\boldsymbol{\pi}_s$, and the set $\{\boldsymbol{\pi}_s\}$ on the Simplex represents the prediction uncertainty for the input $\mathbf{x}$. Each particle requires evaluating a different network instance, and thus using a set of particles at inference time is slow. To improve efficiency, we learn an approximate distribution $q(\boldsymbol{\pi}|\mathbf{x})$ of the class probabilities on the Simplex, which is amortized over many inputs $\mathbf{x}$. We parametrize the approximation into a prediction and confidence term so that the uncertainty is easily obtainable. Our framework requires a single pass through the network to obtain the uncertainty measure, and thus we denote it as one-pass uncertainty (OPU).

A. Bayesian classification framework

We first introduce a generic Bayesian classification framework, e.g. as in Bayesian logistic regression (BLR) or Bayesian neural networks (NN). Let the classifier be specified with categorical likelihood and a parametric function,

$$p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta}) = \mathrm{Cat}(\mathbf{y}|\mathcal{T}(\mathbf{x}; \boldsymbol{\theta})) \tag{1}$$

where $\mathcal{T} : \mathcal{X} \times \Theta \to \mathcal{S}^{K-1}$ is a parametric function from input space to class-probability simplex, and K is the number of classes. For example, $\mathcal{T}$ could be a neural network with a softmax output layer. The Bayesian formulation assumes a prior distribution $p(\boldsymbol{\theta})$ over the parameter space $\boldsymbol{\theta}$.

Given a training dataset $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)_{i=1}^N\}$, the classifier is learned by calculating posterior distribution of the parameters,

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathbf{y}|\mathbf{x})}, \tag{2}$$

where the denominator is the marginal likelihood $p(\mathbf{y}|\mathbf{x}) = \int p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}$. At inference time, given a novel input point $\mathbf{x}^*$, the class prediction is obtained by marginalizing over the probable model parameters $\boldsymbol{\theta}$ according to the posterior,

$$p(\mathbf{y}|\mathbf{x}^*, \mathcal{D}) = \int_\Theta p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta} \tag{3}$$

$$= \int_\Theta \mathrm{Cat}(\mathbf{y}|\mathcal{T}(\mathbf{x}^*; \boldsymbol{\theta}))p(\boldsymbol{\theta}|\mathcal{D})\,\mathrm{d}\boldsymbol{\theta}. \tag{4}$$

In this paper, we assume the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is obtained and focus on the computation of the predictive distribution in Eq. 4. In what follows, let $p(\boldsymbol{\theta}|\mathcal{D})$ be approximate or exact (if available) model posterior, from which samples could be obtained.

B. Induced Distribution over Simplex

Consider the predictive distribution in Eq. 4 for a fixed $\mathbf{x}^*$. Since the data point $\mathbf{x}^*$ is fixed, we denote its transformation as $\mathcal{T}_{\mathbf{x}^*}(\boldsymbol{\theta}) = \mathcal{T}(\mathbf{x}^*; \boldsymbol{\theta})$, which is a function of $\boldsymbol{\theta}$. We next define the random variable $\boldsymbol{\pi} = \mathcal{T}_{\mathbf{x}^*}(\boldsymbol{\theta})$, which is the *distribution of class probabilities* given an input $\mathbf{x}^*$ and under the posterior $p(\boldsymbol{\theta}|\mathcal{D})$. We can then rewrite the predictive distribution as an integral over $\boldsymbol{\pi}$ rather than $\boldsymbol{\theta}$ (a change of variable),

$$p(\mathbf{y}|\mathbf{x}^*, \mathcal{D}) = \int_{\mathcal{S}^{K-1}} p(\mathbf{y}|\boldsymbol{\pi})p(\boldsymbol{\pi}|\mathbf{x}^*, \mathcal{D})\,\mathrm{d}\boldsymbol{\pi} \tag{5}$$

$$= \int_{\mathcal{S}^{K-1}} \mathrm{Cat}(\mathbf{y}|\boldsymbol{\pi})p(\boldsymbol{\pi}|\mathbf{x}^*, \mathcal{D})\,\mathrm{d}\boldsymbol{\pi}. \tag{6}$$

We denote the conditional distribution $p_{\mathbf{x}^*}(\boldsymbol{\pi}) = p(\boldsymbol{\pi}|\mathbf{x}^*, \mathcal{D})$ as an "induced distribution", since its distribution is induced by passing the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ through the transformation $\mathcal{T}_{\mathbf{x}^*}(\boldsymbol{\theta})$.[1] The induced distribution isolates the dependence between input and output (see Fig. 2 (b)), and simplifies the computation of the predictive distribution.

Our key insight is that $p_{\mathbf{x}}$ contains all information we need for prediction and uncertainty measurement. Hence $\boldsymbol{\pi}$ is sufficient in the sense that, given $\boldsymbol{\pi}$, $\mathbf{y}$ is independent of both $\boldsymbol{\theta}$ and $\mathbf{x}$. The isolation combines the complexities in both the likelihood and posterior into a single object $p_{\mathbf{x}}$ and keeps a simple dependence structure between $\mathbf{y}$ and $\boldsymbol{\pi}$. Also, the isolation renders the last probabilistic layer $\mathbf{y}|\boldsymbol{\pi}$

[1] In measure theory, this is a well-defined push-forward measure $\mathcal{T}_{\mathbf{x}}\#p(\boldsymbol{\theta}|\mathcal{D})$ over the simplex.

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

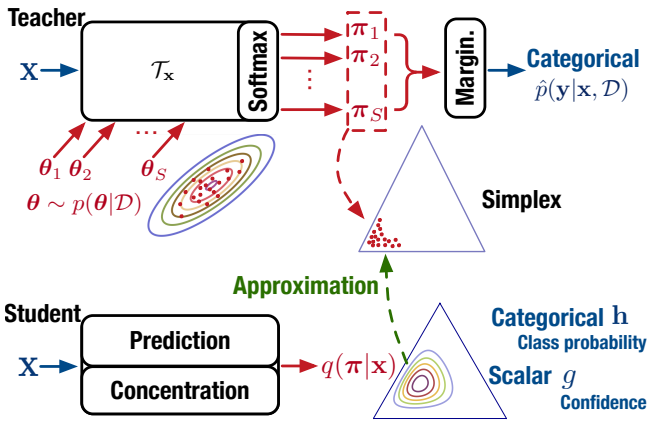


Fig. 1: An intuitive example of the proposed framework. Only one input $\mathbf{x}$ is consider in this example. "Margin." indicates marginalization over $\boldsymbol{\pi}$.

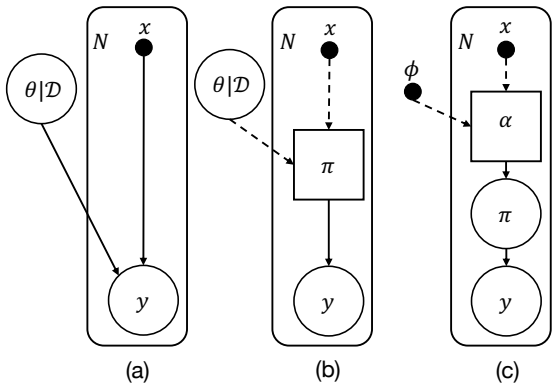


Fig. 2: Graphical representation of probabilistic structure of the Bayes teacher (left and middle) and the student (right). Dashed edges denote deterministic dependence, box nodes are deterministic and circle nodes are stochastic. The left and middle graphs correspond to Eq. 4 and Eq. 6, respectively.

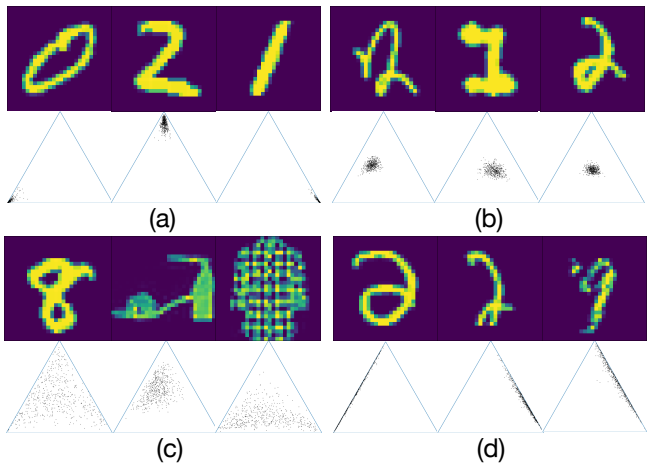


Fig. 3: An empirical example of MC estimate of the induced distribution over simplex. The classifier (MCDP) is trained on real-world digits images of $\{0, 1, 2\}$ (corresponding to left, right, top corners of simplex). The 1st and 3rd rows indicate the input while the 2nd and 4th rows indicate the MC estimate of $\boldsymbol{\pi}$ over a 2-simplex.

nuisance, which can thus be peeled off for prediction. To validate the idea, one can show that the predictive distribution is simply $p(\mathbf{y}|\mathbf{x}, \mathcal{D}) = \mathrm{Cat}(\mathbf{y}|\mathbb{E}_{p_\mathbf{x}}\boldsymbol{\pi})$. The difference between probabilistic structures of the original Bayesian model and the isolated version is shown in Fig. 2 (a) and (b).

The $\boldsymbol{\pi}$ represents the distribution of class probabilities, and thus it can be used to measure the uncertainty of the prediction from input $\mathbf{x}$. However, its density function $p_\mathbf{x}(\boldsymbol{\pi})$ is sometimes intractable to compute, especially for complicated $\mathcal{T}_\mathbf{x}$ (like NN), and thus we empirically observe its distribution based on a set of *particles* (samples) $\{\boldsymbol{\pi}_s\}_{s=1}^S$. The samples are obtained through the generative process: sampling the posterior, i.e. $\boldsymbol{\theta}_s \sim p(\boldsymbol{\theta}|\mathcal{D})$, and "push-forward" the samples by $\mathcal{T}_\mathbf{x}$, i.e. $\boldsymbol{\pi}_s = \mathcal{T}_\mathbf{x}(\boldsymbol{\theta}_s)$. Fig. 3 shows an example of the distribution of $\boldsymbol{\pi}$ samples for different types of input $\mathbf{x}$. In Fig. 3 (a), the inputs are similar to the training data, and thus the particles gather around a corner of the simplex, indicating a confident prediction. In Fig. 3 (b), the inputs are digit images but are relatively hard to predict. Therefore, the particles gather around the center, indicating the model is certain that the input is on the decision boundary. In Fig. 3 (c), the inputs are images outside the domain of training data, and the particles spread over the simplex, which means the model has a high uncertainty about input, indicating out-of-domain (OOD). In Fig. 3 (d), the particles spread along a line between two corners, indicating the model is confident that the result is not at the other corner.

Although the particles are effective at measuring uncertainty, their use for inference is time-consuming as each particle requires one evaluation of the model (e.g., $100\times$ slow-down for using 100 Monte-Carlo samples [21]). This motivates us to approximate $p_\mathbf{x}$ with a tractable conditional distribution $q(\boldsymbol{\pi}|\mathbf{x})$, thus requiring only a single pass to obtain the uncertainty.

C. Amortized Approximation

The view of $p_\mathbf{x}$ enables flexible choices of $q(\boldsymbol{\pi}|\mathbf{x})$, as any distribution defined on $\mathbb{R}^K$ can be transformed to $\mathcal{S}^{K-1}$ via logistic transformation [1]. However, modeling $q(\boldsymbol{\pi}|\mathbf{x})$ locally for every input is not practical, as the design efforts and the number of parameters grows linearly with the number of data points. Therefore, we propose to approximate $p_\mathbf{x}$ in two aspects: 1) use a single family of distribution q; 2) for generalizing to unseen examples, let the parameter of q, $\boldsymbol{\alpha}_\mathbf{x} = \boldsymbol{\alpha}(\mathbf{x}; \boldsymbol{\phi})$, be a function depending on $\mathbf{x}$ and parameterized by a set of global adaptive parameters $\boldsymbol{\phi}$, and thus $q_\mathbf{x} = q(\boldsymbol{\pi}|\mathbf{x}) = q(\boldsymbol{\pi}|\boldsymbol{\alpha}_\mathbf{x})$. The computational cost is amortized by casting the problem of learning a series of conditional distributions to a regression problem.

We term $p_\mathbf{x}$ as "teacher distribution" and $q_\mathbf{x}$ as "student". In our method, as seen in the graphical representation in Fig. 2, by proper approximation, the stochasticity, and knowledge in node $\boldsymbol{\theta}$ of the teacher is transferred into node $\boldsymbol{\pi}$ of the student model, such that the full predictive uncertainty is maintained. In terms of computation, the approximation requires sampling only in the training stage. While in the testing stage, to obtain the predictive distribution, only one evaluation is required. Thus, we denote the framework as the One-Pass Uncertainty (OPU) model.

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

The above benefits do not introduce any concession on generalizability. Since OPU is based on approximating the distribution of the output class probabilities which is common for all classifiers, the amortized approximation can be applied to any Bayesian classifier. Note that the approximation framework can be extended to a non-parametric model like the Gaussian process, where the computational cost of inference is high.

In this work, we choose the student model q to be a Dirichlet distribution, $q(\boldsymbol{\pi}|\boldsymbol{\alpha}_{\mathbf{x}}) = \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_{\mathbf{x}})$, where $\boldsymbol{\alpha}_{\mathbf{x}}$ is a function mapping input $\mathbf{x}$ to a Dirichlet parameter. The reasons for choosing Dirichlet is the tractability: the Dirichlet is the conjugate prior to the Categorical, and thus enables tractable integration of (6) given the parameters. To better disentangle the uncertainty measures, we use the design $\boldsymbol{\alpha}_{\mathbf{x}} = \mathbf{h}_{\mathbf{x}} \cdot \mathrm{e}^{g_{\mathbf{x}}}$, where $\mathbf{h}_{\mathbf{x}} = \mathbf{h}(\mathbf{x}; \boldsymbol{\phi}_1)$ and $g_{\mathbf{x}} = g(\mathbf{x}; \boldsymbol{\phi}_2)$ are two neural networks, and the vector output $\mathbf{h}$ sums to 1. Vector output $\mathbf{h}$ determines the mean of the Dirichlet (i.e., the predicted class probabilities), and g determines the concentration of the Dirichlet (i.e., the prediction confidence). To see this, the posterior of the class labels is the Dirichlet mean, $p(y_\ell = 1|\mathbf{x}, \boldsymbol{\phi}) = \int \mathrm{Cat}(y_\ell = 1|\boldsymbol{\pi})\mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha}_{\mathbf{x}})\,\mathrm{d}\boldsymbol{\pi} = \frac{\alpha_{\mathbf{x},\ell}}{\sum_c \alpha_{\mathbf{x},c}} = h_{\mathbf{x},\ell}$ where y_ℓ and $\alpha_{\mathbf{x},\ell}$ are the ℓ-th coordinate of $\mathbf{y}$ and $\boldsymbol{\alpha}_{\mathbf{x}}$ respectively. Therefore, we call $\mathbf{h}_{\mathbf{x}}$ as the "prediction model" (PM). Similarly, the precision parameter α_0 (determines sharpness) of the Dirichlet solely depends on $g_{\mathbf{x}}$, $\alpha_0 = \sum_c \alpha_{\mathbf{x},c} = \sum_c h_{\mathbf{x},c} e^{g_{\mathbf{x}}} = \mathrm{e}^{g_{\mathbf{x}}}$. Therefore, we term $g_{\mathbf{x}}$ as the "concentration model" (CM). Based on this property, whether the Dirichlet is flat or not, can be fully characterized by CM. It can be expected that when approximating particles in Fig. 3 (a) and (b), the output value of CM is high, as the samples are concentrated, which means high confidence. CM outputs a low value for particles in Fig. 3 (c), yielding a flat distribution and low confidence.

D. Learning

Our objective is to approximate the predictive distribution $p(\boldsymbol{\pi}|\mathbf{x}, \mathcal{D})$ (the teacher) using $q(\boldsymbol{\pi}|\mathbf{x})$ (the student). To measure the difference between the teacher and student, we use a probability distance (e.g., forward KL) $\rho(\cdot, \cdot)$, giving a point-wise loss of

$$\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = \rho(p_{\mathbf{x}}, q_{\mathbf{x}}). \tag{7}$$

where $\boldsymbol{\alpha}$ are the parameters of $q_{\mathbf{x}}$. To generalize to unseen examples, as in any empirical risk minimization, the point-wise losses are aggregated to get an overall goal for optimization, which is a similar treatment adopted in BDK [3] and others [36, 11]. The final objective is then to minimize the loss aggregated over the distribution of $\mathbf{x}$ (the training data), and thus

$$\min_{\boldsymbol{\alpha} \in \mathcal{F}} \mathbb{E}_{\mathbf{x} \sim p(\mathbf{x})} \mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}), \tag{8}$$

where $\mathcal{F}$ is some hypothesis space and $p(\mathbf{x})$ is some distribution over $\mathcal{X}$. In practice, we take $p(\mathbf{x}) = p_{\mathcal{D}'}(\mathbf{x})$ with $\mathcal{D}' = \{\mathbf{x}_m\}_{m=1}^M$ a dataset containing features only. As the amortized approximation of induced distributions in Bayesian classifiers is unexplored in the literature, we consider and compare several choices of ρ including KL divergence, earth mover's distance (EMD), and maximum mean discrepancy (MMD). The corresponding derivation of the training objectives and training algorithms are in the appendix.

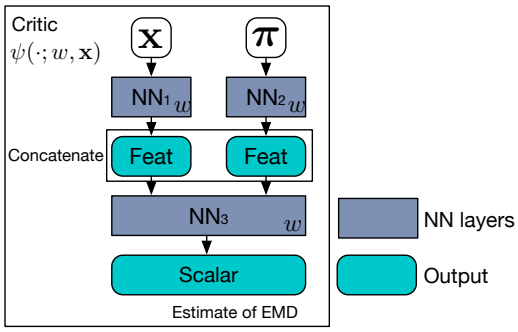


Fig. 4: A brief example of the critic in EMD.

1) Forward KL divergence: We first consider the forward KL divergence, $\mathbb{E}_{p_{\mathbf{x}}} \ln \frac{p_{\mathbf{x}}}{q_{\mathbf{x}}}$. Since the term $\mathbb{E}_{p_{\mathbf{x}}} \ln p_{\mathbf{x}}$ is irrelevant when optimizing w.r.t. $\boldsymbol{\alpha}$, it is equivalent to using cross-entropy as a local loss, i.e. $\mathcal{L}(\mathbf{x}, \boldsymbol{\alpha}) = -\mathbb{E}_{p_{\mathbf{x}}} \ln q_{\mathbf{x}}$. By plugging in a particle estimation $\hat{p}_{\mathbf{x}} = \frac{1}{S}\sum_{s=1}^S \delta(\boldsymbol{\pi} - \mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}_s))$, the training objective becomes $\min_{\boldsymbol{\alpha}} -\mathbb{E}_{p_{\mathcal{D}'}(\mathbf{x})} \frac{1}{S}\sum_s \ln q_{\mathbf{x}}(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}_s))$, which is equivalent to an "amortized" maximum likelihood estimation (MLE) problem with particles providing the estimation of sufficient statistics of $q_{\mathbf{x}}$. Due to the zero-avoiding nature, forward KL tends to over-estimate the support of $p_{\mathbf{x}}$. This leads to under-confidence approximation ("flat" approximate distribution) and hence might deteriorate the quality of uncertainty measurements. This is expected to be more serious when $p_{\mathbf{x}}$ is multi-modal.

Finally, we note that using reverse KL divergence $\mathbb{E}_{p_{\mathbf{x}}} \ln \frac{q_{\mathbf{x}}}{p_{\mathbf{x}}}$ as the probability distance ρ, as in the variational Bayes framework, cannot be used in our problem setting because we do not have an explicit form of the predictive log-density $\ln p_{\mathbf{x}}$. Furthermore, the "trick" of rewriting the reverse KL divergence to obtain a lower-bound to the marginal likelihood will not lead to a tractable solution because of the complex dependence between $\mathcal{D}$ and $\boldsymbol{\pi}$ through $\boldsymbol{\theta}$. In particular, computing the lower-bound would require the joint likelihood of the hidden and observation variables ($\boldsymbol{\pi}$ and $\mathcal{D}$ in our scenario), $p(\mathcal{D}, \boldsymbol{\pi}|\mathbf{x}) = \int p(\boldsymbol{\pi}|\mathbf{x}, \boldsymbol{\theta}) p(\mathcal{D}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) d\boldsymbol{\theta}$, which is still intractable. Thus reverse KL divergence, i.e. variational Bayes framework, is not applicable to our problem.

2) EMD: It is known that EMD provides much weaker topology than other probability distance measures [28]. In the application where data is supported on strictly lower-dimensional manifolds, EMD provides more stable gradient than KL divergence [2, 33]. An example of particles on a low-dimensional manifold is shown in Fig. 3 (d).

Specifically, the Kantorovich-Rubinstein (KR) dual representation of EMD [34] is defined as $W_1(p, q) = \sup_{\|\psi\|_{\mathrm{L}} \leq 1} \mathbb{E}_{p(z)}[\psi(z)] - \mathbb{E}_{p(z)}[\psi(z)]$. The function ψ, called a *critic function*, is an indication of how the two distributions p and q differ. Intuitively, in the context of adversarial training, the critic function is analogous to the discriminator that classifies fake and real data in a GAN. In a region where $p(z)$ has more mass comparing with $q(z)$, $\psi(z)$ tends to be larger. Optimizing $q(z)$ to better match $\psi(z)$ decreases EMD. In our problem, we replace $p(z)$ with $p(\boldsymbol{\pi}|\mathbf{x}, \mathcal{D})$ and $q(z)$ with $q(\boldsymbol{\pi}|\mathbf{x})$ then the EMD is transformed to $W_1(p_{\mathbf{x}}, q_{\mathbf{x}}) = \sup_{\|\psi\|_{\mathrm{L}} \leq 1} \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta}|\mathcal{D})}[\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}))] - \mathbb{E}_{\boldsymbol{\pi} \sim q_{\mathbf{x}}}[\psi(\boldsymbol{\pi})]$, where $\|\cdot\|_{\mathrm{L}}$

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

denotes the Lipschitz semi-norm and ψ is the *critic* (or the discriminator [2]). As the induced distribution is conditioned on $\mathbf{x}$, a local critic $\psi_{\mathbf{x}}$ should be defined for each $\mathbf{x}$.

Following [2], intractable supremum is solved by parameterizing $\psi_{\mathbf{x}} = \psi(\cdot; w_{\mathbf{x}})$. Optimizing each local implicit parameter $w_{\mathbf{x}}$ yields an approximate calculation of $W_1(p_{\mathbf{x}}, q_{\mathbf{x}})$. To avoid training $|\mathcal{D}'|$ local critics, we propose to let w be the global weight, and let ψ depends on $\mathbf{x}$, i.e., $\psi_{\mathbf{x}} = \psi(\boldsymbol{\pi}; w, \mathbf{x})$ (see Fig. 4). We assume the supremum is attained for some $w \in \mathcal{W}$ for a compact $\mathcal{W}$, then the representation of EMD becomes $\max_w \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}); w, \mathbf{x})] - \mathbb{E}_{q_{\mathbf{x}}}[\psi(\boldsymbol{\pi}; w, \mathbf{x})]$. The final aggregated training objective under EMD is,

$$\min_{\boldsymbol{\alpha}} \max_{w} \quad \mathbb{E}_{p_{\mathcal{D}'}(\mathbf{x})}[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}); w, \mathbf{x})] - \mathbb{E}_{q_{\mathbf{x}}}[\psi(\boldsymbol{\pi}; w, \mathbf{x})]] + \lambda \mathcal{R}(w), \tag{9}$$

where w is the introduced global parameter for ψ, $\mathcal{R}(w)$ is the imposed gradient penalty [12] over w to enforce the Lipschitz constraint. Solving the minimax problem requires the supremum to be attained for each $\mathbf{x}$ under the Lipschitz constraint. Specifically, in every optimization step, ψ is trained to generate a critic for each $\mathbf{x}$ that matches the exact EMD. Practically, this needs a high-capacity critic, and the required capacity increases with the number of classes K.

3) MMD: Let $\mathcal{H}_k$ be a reproducing kernel Hilbert space (RKHS) defined by a positive-definite kernel k, the MMD between $p_{\mathbf{x}}$ and $q_{\mathbf{x}}$ can be written as

$$\mathrm{MMD}_k(p, q) = \sup_{\psi \in \mathcal{H}_k, \|\psi\|_{\mathcal{H}} \leq 1} \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}))] - \mathbb{E}_{q_{\mathbf{x}}}[\psi(\boldsymbol{\pi})] \tag{10}$$

Compared with EMD, the advantage of MMD is that there is no need to train an NN as the critic that maximizes Eq. 10. With kernel trick, MMD can be readily estimated in closed-form with its empirical version under finite sample (Sec. B).

Compared with KL divergence, MMD is a valid statistical metric. Due to the symmetry property, the approximation is expected to be neither mean-seeking nor mode-seeking. Thus, MMD is not expected to have an under-confidence issue.

4) Reparameterization: Note that optimization under both EMD and MMD requires gradient of the expectation of critic via sampling from q, which contains parameters. To obtain an efficient gradient estimator and reduce variance, we reparameterize the Dirichlet by an equivalent product of K independent Gamma (ProdGamma) distributions. If $\tilde{\boldsymbol{\pi}} \sim \mathrm{ProdGamma}(\tilde{\boldsymbol{\pi}}|\boldsymbol{\alpha}) = \prod_{k=1}^{K} \mathrm{Gam}(\tilde{\pi}_k|\alpha_k)$, then $\boldsymbol{\pi} = (\sum_k \tilde{\pi}_k)^{-1} \tilde{\boldsymbol{\pi}} \sim \mathrm{Dir}(\boldsymbol{\pi}|\boldsymbol{\alpha})$. By Thm. 3 in [2], in each $\mathcal{L}(\boldsymbol{\phi}|\mathbf{x})$, the supremum is attained at $\psi^*_{\mathbf{x}} \in \mathrm{L}_1$ and the gradient is $\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}|\mathbf{x}) = -\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\boldsymbol{\pi}|\mathbf{x},\boldsymbol{\phi})}[\psi^*_{\mathbf{x}}(\boldsymbol{\pi})]$. Then as noted by [8], the gradient $\nabla_{\boldsymbol{\phi}} \mathcal{L}(\boldsymbol{\phi}|\mathbf{x})$ can be implicitly computed without knowing the inverse of standardization function (e.g., CDF). Specifically, by Eq. 5 in [8], $\nabla_{\boldsymbol{\phi}} \mathbb{E}_{q(\boldsymbol{\pi}|\mathbf{x};\boldsymbol{\phi})}[\psi^*_{\mathbf{x}}(\boldsymbol{\pi})] = \mathbb{E}_{\mathrm{PG}(\tilde{\boldsymbol{\pi}}|\mathbf{x};\boldsymbol{\phi})}[\nabla_{\tilde{\boldsymbol{\pi}}} \psi^*_{\mathbf{x}}(\boldsymbol{\pi}) \nabla_{\boldsymbol{\phi}} \tilde{\boldsymbol{\pi}}]$, where the first term $\nabla_{\tilde{\boldsymbol{\pi}}} \psi^*_{\mathbf{x}}(\boldsymbol{\pi})$ is computed via the chain rule and the second term $\nabla_{\boldsymbol{\phi}} \tilde{\boldsymbol{\pi}}$ is obtained by solving a local diagonal linear system.

E. Amortization Gap

To better understand the nature of the proposed approximation, we consider the "un-amortized" version of the approximation as an intermediate stage, which involves fitting separate approximations to each $p_{\mathbf{x}}$. To demonstrate the idea, we leverage MMD due to its nice property. For fixed $p_{\mathbf{x}}$, the optimal point-wise approximation within family $\mathcal{Q}$ is defined as

$$\bar{q}^*_{\mathbf{x}} = \operatorname*{argmin}_{q \in \mathcal{Q}} \mathrm{MMD}_k(q, p_{\mathbf{x}}). \tag{11}$$

Then we have the following lemma:

Lemma 1. *Let $\mathcal{P}(\mathcal{S}^{K-1})$ be the space of probability measures over the simplex, equipped with MMD metric defined by a universal kernel. If $\mathcal{T}$ satisfies Assumption 1, then the map $\mathbf{x} \mapsto p_{\mathbf{x}}$ is continuous. Further, if $\mathcal{Q} \subseteq \mathcal{P}(\mathcal{S}^{K-1})$ is a closed convex model space, the projection $\bar{q}^*_{\mathbf{x}}$ is unique and the map $\mathbf{x} \mapsto \bar{q}^*_{\mathbf{x}}$ is also continuous.*

Further, if we assume the model space is parameterized and identifiable in terms of MMD, i.e. $\mathcal{Q} = \{q(\boldsymbol{\pi}|\boldsymbol{\alpha}) : \boldsymbol{\alpha} \in \mathcal{A}\}$ and $\mathrm{MMD}_k(q(\boldsymbol{\pi}|\boldsymbol{\alpha}), q(\boldsymbol{\pi}|\boldsymbol{\alpha}')) = 0$ if and only if $\|\boldsymbol{\alpha} - \boldsymbol{\alpha}'\| = 0$, we may obtain continuity in parameter space. The continuity of optimal parameters implies there exists a continuous function $\boldsymbol{\alpha}^* : \mathbf{x} \in \mathcal{X} \mapsto \boldsymbol{\alpha}^*(\mathbf{x}) \in \mathcal{A}$, which serves as the essential target of our amortized goal.

To analyze how amortization affects the approximation, we define the local amortization gap as

$$\Delta(\mathbf{x}) = \mathrm{MMD}_k(q_{\mathbf{x}}, p_{\mathbf{x}}) - \mathrm{MMD}_k(\bar{q}^*_{\mathbf{x}}, p_{\mathbf{x}}). \tag{12}$$

Then it obviously holds that

$$0 \leq \Delta(\mathbf{x}) \leq \mathrm{MMD}_k(q_{\mathbf{x}}, \bar{q}^*_{\mathbf{x}}) \tag{13}$$

where the lower bound is because $\bar{q}^*_{\mathbf{x}}$ is the projection and the upper bound is due to triangle inequality. Then our goal, minimizing aggregated MMD loss $\mathbb{E}_{p(\mathbf{x})} \mathrm{MMD}_k(q_{\mathbf{x}}, p_{\mathbf{x}})$, is essentially equivalent to minimizing aggregated amortization gap $\mathbb{E}_{p(\mathbf{x})} \Delta(\mathbf{x})$, up to an irrelevant additive constant $\mathrm{MMD}_k(\bar{q}^*_{\mathbf{x}}, p_{\mathbf{x}})$, which does not contain student model parameters $\boldsymbol{\phi}$.

One can see that $\boldsymbol{\alpha}^*$ is the global minimizer of aggregated amortization gap and hence the global minimizer of the aggregated MMD loss. Intuitively, if $\mathcal{F}$ is of enough capacity to approximate $\boldsymbol{\alpha}^*$, then the infimum (over $\mathcal{F}$) of aggregated amortization gap could be close to 0, which means no additional cost is incurred by amortizing the approximation. In other words, model loss due to using restrictive $\mathcal{Q}$ and measured by $\mathrm{MMD}_k(\bar{q}^*_{\mathbf{x}}, p_{\mathbf{x}})$ will dominate. Further, if the global optimum is reached, OPU approximation must agree with the point-wise minimizer $\bar{q}^*_{\mathbf{x}}$, i.e. $\mathrm{MMD}_k(q_{\mathbf{x}}, \bar{q}^*_{\mathbf{x}}) = 0$, due to uniqueness of projection. Empirically, we observe the magnitude of the constant $\mathrm{MMD}_k(q_{\mathbf{x}}, \bar{q}^*_{\mathbf{x}}) = 0$ is acceptably small, as evinced by the averaged amortization error (AvgAmtErr) in Tab. VI in the experiments in Sec. VI.

IV. Experiments with traditional Bayesian models

In this section, we present experiments applying OPU to traditional Bayesian models, including Bayesian Logistic Regression (BLR) and Gaussian Process (GP) classification.

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

TABLE I: Uncertainty measure with traditional Bayesian models. $\mathcal{H}(\cdot)$ represents discrete entropy and $\mathcal{S}(\cdot)$ is the softmax function. $\boldsymbol{\pi}_s$, $\mathbf{h}_{\mathbf{x}}$ and $g_{\mathbf{x}}$ are defined in Sections III-B and III-C. The process of sampling $\boldsymbol{\mu}_s^*$ from GP is described in the Appendix C-D.

Model	Uncertainty measures	
PG CA-PG	Entropy (E) Max. Prob. (P)	$\mathcal{H}(\frac{1}{S}\sum_{s=1}^{S}\boldsymbol{\pi}_s)$ $\max_c(\frac{1}{S}\sum_{s=1}^{S}\boldsymbol{\pi}_s)$
SGPMC SVGP	Entropy (E) Max. Prob. (P)	$\mathcal{H}(\frac{1}{S}\sum_{s=1}^{S}\mathcal{S}(\boldsymbol{\mu}_s^*))$ $\max_c(\frac{1}{S}\sum_{s=1}^{S}\mathcal{S}(\boldsymbol{\mu}_s^*))$ $\boldsymbol{\mu}_s^* \sim P(\boldsymbol{\mu})$
OPU	Entropy (E) Max. Prob. (P) Concentration (C)	$\mathcal{H}(\mathbf{h}_{\mathbf{x}})$ $\max_c(\mathbf{h}_{\mathbf{x}})$ $e^{g_{\mathbf{x}}}$

A. Experimental Setup

1) Models and Tasks: We consider two models, BLR and GP classification. For each model, we choose a Bayesian method as the Bayes teachers and approximate them with our OPU. We also compare with state-of-the-art approximations that are proposed for the specific types of methods. For BLR, we use Polya Gamma (PG) [29] as the teacher, and CompactApprox [31] for comparisons. For GP, we use SGPMC [16] as the teacher, and SVGP [17] for comparison. For each type of model, in-domain misclassification (MisC) detection, out-of-domain (OOD) input detection, prediction performance, and prediction time are presented.

2) Data: For a fair comparison with other works, we let $\mathcal{D}' = \mathcal{D}$ when learning the approximation of OPU. The in-domain dataset is split into training data and testing data, i.e., $\mathcal{D}^{\text{in}} = \mathcal{D}' = \{\mathcal{D}^{\text{tr}}, \mathcal{D}^{\text{te}}\}$, which is used for training models, evaluating prediction and MisC detection. The OOD dataset $\mathcal{D}^{\text{ood}}$ and $\mathcal{D}^{\text{te}}$ are used for OOD detection. To reduce the time consumption in sampling from the Bayes teacher, we generate and store the set of MC samples beforehand and use the same set during approximation.

3) Uncertainty Measures and Evaluation Metrics: We show the measures of uncertainty for different models in Tab. I, which are Entropy (E), Max probability (P), and Concentration (C). For OOD detection, given two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$, if $q_{\mathbf{x}_1}$ has higher E (or lower P or lower C) than $q_{\mathbf{x}_2}$, then $q_{\mathbf{x}_1}$ is more likely to be an OOD data. The value of E, P, and C are computed for all testing data points (both in-domain data and out-of-domain data), which rank the data points based on their uncertainty values.

To assess the performance, we use accuracy, time, Area under the ROC (AUROC) and PR (AUPR), following the baseline in [14]. The ROC curve is plotted by setting a threshold on each uncertainty value and computing the True Positive Rate and False Positive Rate at each threshold. Similarly, the PR curve is plotted by computing the Precision and Recall. Then the area under two curves (AUROC and AUPR) can be obtained. For misclassification detection, a similar calculation process is applied. To save space, we only present the best performing uncertainty measure (E, P, or C) for each task and method.

Time is evaluated on the whole $\mathcal{D}^{\text{te}}$. We use the MXNet implementation of BDK and GPflow implementation of GP, and the remaining models are implemented in Pytorch. All experiments run on a desktop with an i7-8700 CPU and an RTX-2080 Ti GPU.

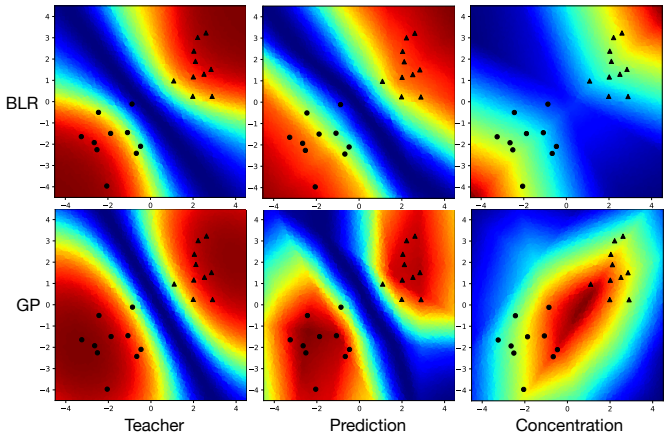


Fig. 5: Toy example of the uncertainty of (left) Bayes teacher; (middle) prediction model, i.e. class probabilities; (right) concentration model. Samples are generated by the BLR model with Polya Gamma, and Gaussian process classification. The uncertainty of the teacher and prediction model is measured by entropy, while that of the concentration model is given by its scalar output. Blue areas correspond to high uncertainty while red areas correspond to high confidence. (Better viewed in color.)

B. Toy Example

We first present a toy example to illustrate our OPU. In the toy example, we generate two clusters each with 10 data points using two Gaussian distributions centered at $(2.5, 2.5)$ and $(-2.5, -2.5)$ respectively. The covariance matrices are both identity matrices.

1) Bayesian logistic regression: For the first example, we use Bayesian logistic regression (BLR) with Polya Gamma as the Bayes teacher $p_{\mathbf{x}}$ and generate 100 samples for the parameters. The prediction model $h_{\mathbf{x}}$ and concentration model $g_{\mathbf{x}}$ in the student are both MLPs with a 2-10-2 architecture, where the activation function is ReLU. To train the student model, we use forward KL as $\rho(\cdot, \cdot)$ and train the student model with an SGD optimizer at a learning rate of 0.01 for 100 epochs. To visualize the results, we randomly select 1000 data points in a 2-D square, calculate the measures of uncertainty for each data point, and perform the interpolation. The uncertainty measures include: the entropy of mean of particle collected from the Bayes teacher, the entropy of output of prediction model $h_{\mathbf{x}}$, and output of concentration model $g_{\mathbf{x}}$.

The results are shown in Fig. 5 (top). The entropy map of the prediction model is similar to that of the teacher, with highly uncertain areas appearing around the decision boundary and the contours are almost parallel with the decision boundary. For the concentration model, the area between the two clusters are lighter showing lower uncertainty, while the CM is more uncertain on the Out-of-Domain (OOD) area. By comparison, BDK [3] and other knowledge distillation works [36, 11] only capture the uncertainty of the prediction model (middle figure), while they cannot tell whether a data point is out-of-domain or not.

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

TABLE II: Results on Bayesian logistic regression and Gaussian process classification models.

Bayes teacher	Data	Model	MisC detection AUROC	MisC detection AUPR	OOD detection AUROC	OOD detection AUPR	Acc. (%)	Time (sec.)
BLR	Pima	PG	**60.2**$\pm$0.7 (E)	24.2$\pm$1.5 (P)	87.0$\pm$3.0 (E)	76.7$\pm$0.8 (E)	**64.4**$\pm$0.5	1.010$\pm$0.001
		CA-PG	58.2$\pm$1.4 (E)	24.2$\pm$1.3 (E)	80.1$\pm$2.1 (E)	74.5$\pm$0.4 (E)	62.3$\pm$0.2	0.182$\pm$0.001
		OPU-PG	59.7$\pm$0.6 (P)	**25.6**$\pm$1.1 (E)	**100.0**$\pm$0.0 (C)	**100.0**$\pm$0.0 (C)	**64.4**$\pm$0.5	**0.011**$\pm$0.002
	Spam	PG	**83.9**$\pm$0.1 (P)	**24.7**$\pm$1.4 (E)	55.5$\pm$2.2 (E)	53.5$\pm$1.8 (E)	**92.4**$\pm$0.8	5.938$\pm$0.001
		CA-PG	64.1$\pm$1.2 (P)	23.9$\pm$0.9 (E)	71.5$\pm$2.4 (E)	67.5$\pm$3.1 (E)	85.4$\pm$1.7	0.362$\pm$0.001
		OPU-PG	**83.9**$\pm$0.1 (E)	23.8$\pm$0.8 (E)	**99.7**$\pm$0.0 (C)	99.3$\pm$0.0 (C)	**92.4**$\pm$0.9	**0.011**$\pm$0.002
GP	Pima	SGPMC	65.3$\pm$1.3 (E)	**46.4**$\pm$0.8 (E)	96.7$\pm$0.3 (E)	94.9$\pm$0.1 (E)	**79.3**$\pm$0.1	**0.003**$\pm$0.002
		SVGP	64.3$\pm$0.7 (E)	43.2$\pm$1.2 (E)	96.0$\pm$0.5 (E)	91.1$\pm$1.1 (E)	77.1$\pm$0.0	0.004$\pm$0.001
		OPU-SGPMC	**65.7**$\pm$0.2 (E)	44.6$\pm$1.6 (E)	**100.0**$\pm$0.0 (C)	**100.0**$\pm$0.0 (C)	79.2$\pm$0.1	0.010$\pm$0.002
	Spam	SGPMC	**86.7**$\pm$0.3 (E)	37.8$\pm$1.3 (E)	98.6$\pm$0.2 (E)	97.6$\pm$0.1 (E)	**92.4**$\pm$0.2	0.056$\pm$0.002
		SVGP	86.2$\pm$0.2 (E)	33.3$\pm$1.0 (E)	99.2$\pm$0.1 (E)	98.5$\pm$0.2 (E)	92.1$\pm$0.1	0.032$\pm$0.001
		OPU-SGPMC	86.5$\pm$0.2 (E)	**39.5**$\pm$0.8 (E)	**100.0**$\pm$0.0 (C)	**100.0**$\pm$0.0 (C)	92.0$\pm$0.1	**0.011**$\pm$0.002

2) Gaussian process classification: For the second example, we use a Gaussian process classification model SGPMC with RBF kernel as the Bayes teacher $p_{\mathbf{x}}$, following the same sampling schedule and experimental setup as BLR. The results are visualized in Fig. 5 (bottom). Similar to BLR, the entropy of particle mean and the entropy of the prediction output of GP only express the uncertainty of classification (left and middle). In contrast, the concentration model (right) fully captures the out-of-domain uncertainty, as the in-domain area is circled by the entropy contours. The area between the two clusters has high prediction uncertainty and high concentration (high confidence), showing that the model knows that this area is in-domain but does not know the class because it is on the decision boundary. This corresponds to the "known-unknown" case as mentioned in Fig. 3 (c).

C. BLR and GP on Real-World Datasets

We next present results on real-world datasets.

1) Datasets: This experiment uses Pima and Spambase datasets as $\mathcal{D}^{\text{in}}$. Pima is a medical dataset with 769 data points and 9 dimensions. Spambase is a text dataset with 4601 data points and 57 dimensions for identifying spam email. We generate the same number of data points from a zero-mean multivariate Gaussian distribution for $\mathcal{D}^{\text{ood}}$. For each dataset, 10% of data points are uniformly selected into the testing set $\mathcal{D}^{\text{te}}$. We normalize the data by features with L2 norm.

2) Models: For BLR, we test three models: Polya Gamma (PG), CompactApprox approximating PG (CA-PG), and OPU approximating PG (OPU-PG). We draw 500 posterior samples from PG (methods to draw samples are shown in Appendix C-A) and train OPU with the following hyperparameters: number of epochs 100, learning rate for student. CA-PG is trained by first drawing 5000 samples from PG then evaluating the model with 50 randomly selected samples from them (same setup as CompactApprox). The random selection is repeated for 10^5 times and we pick the best group of samples.

For GP, we use SGPMC [16] as the teacher, and SVGP [17] for comparison. There are 3 models tested: SGPMC, OPU approximating SGPMC (OPU-SGPMC) and SVGP. SGPMC and SVGP are trained with $\frac{1}{10}|\mathcal{D}^{\text{in}}|$ data points randomly selected from $\mathcal{D}^{in}$ as inducing points. Then 500 samples over functions of $\mathcal{D}^{\text{in}}$ are generated from SGPMC (methods to draw samples are in Appendix C-D).

3) Experiment Results: The results are shown in Tab. II. As the results for the three metrics are similar, we only show the results trained with forward KL divergence.

Looking at the BLR results, OPU-PG maintains a similar performance with the original PG on prediction accuracy and MisC detection. Meanwhile, OPU outperforms CA-PG on prediction accuracy, MisC, and OOD detection. For OPU, CM outperforms other uncertainty measures at OOD detection, which indicates it captures the distributional uncertainty well. OPU performs better than the PG Bayes teacher. The reason might be that a parametric model is learned to approximate the ensemble of discrete samples, which could produce a smoother output distribution (regularization), leading to better performance. OPU also achieves a $\sim$100-600x speedup from the original PG.

For the GP results, on MisC detection and prediction accuracy, OPU has a similar performance to SGPMC, which indicates the effectiveness of approximating prediction results with an ensemble of samples in the nonparametric family. With the same number of inducing points, SVGP performs slightly worse than SGPMC, because it incurs a two-fold approximation. Measuring uncertainty with CM in OPU outperforms other measures and models, which indicates the sharpness of the logistic-normal distribution can be captured via the CM in our designed Dirichlet.

SGPMC and SVGP are faster than OPU on Pima, but are slower than OPU on the larger Spambase. This is due to the static latency for setting up the GPU for OPU, which becomes the main time cost when the dataset is small (as in Pima). For the two GP methods, the computation time depends on the number of inducing points and the number of dimensions. Therefore, as the dataset becomes larger (Spambase), the computation time increases.

V. Experiments with Bayesian Neural Networks

In this section, we present experimental results using OPU on Bayesian neural networks.

A. Experimental Setup

1) Models and Tasks: We use MCDP [10] and SGLD [35] as the Bayes teacher. Given MCDP as the teacher models, MCDP-KL represents the performance of MCDP together with

the *locally* fitted $q^*_{\mathbf{x}}$ under forward KL. Similarly, we present the results of MCDP-EMD, MCDP-MMD, SGLD-KL, SGLD-EMD, SGLD-MMD.

We approximate the predictive distribution using OPU and compare with BDK [3] and DPN [25]. We denote these as OPU-MCDP (OPU approximating MCDP), OPU-SGLD, BDK-SGLD, BDK-Dir-SGLD, and DPN. For BDK-Dir-SGLD, we replace the Categorical distribution in BDK by a Dirichlet without disentangling the mean and concentration, then train it with the same MC ensemble as OPU-SGLD. This is to show the benefits of explicit disentanglement

The MisC and OOD detection, prediction performance, and prediction time are presented.

2) Datasets: For datasets, we use MNIST, balanced EMNIST datasets and Cifar10 as $\mathcal{D}^{\text{in}}$, and use Omniglot, SEMEION and LSUN dataset as $\mathcal{D}^{\text{ood}}$, as in [25]. MNIST is an image dataset of 28×28-dimensional handwritten digits from 0 to 9, which contains 60,000 training data points and 10,000 testing data points. Balanced EMNIST is an image dataset that contains 131,600 characters and 47 balanced classes. The CIFAR-10 dataset consists of 60000 32x32 color images in 10 classes, with 6000 images per class. Omniglot is an image dataset that contains 1,623 different handwritten characters from 50 different alphabets. SEMEION is an image dataset that contains 1,593 handwritten digits. LSUN contains around one million labeled images for each of 10 scene categories and 20 object categories.

TABLE III: Uncertainty measure with Bayesian neural networks. $\mathcal{H}(\cdot)$ represents discrete entropy, $\mathcal{DH}(\cdot)$ represents differential entropy and $\mathcal{S}(\cdot)$ is the softmax function.

Model	Uncertainty measures	
MCDP SGLD DPN	Entropy (E) Max. Prob. (P) Differential Entropy (D)	$\mathcal{H}(\frac{1}{S}\sum_{s=1}^{S}\boldsymbol{\pi}_s)$ $\max_c(\frac{1}{S}\sum_{s=1}^{S}\boldsymbol{\pi}_s)$ $\mathcal{DH}(q^*_{\mathbf{x}})$
BDK	Entropy (E) Max. Prob. (P)	$\mathcal{H}(\mathbf{y})$ $\max_c(\mathbf{y})$
OPU	Entropy (E) Max. Prob. (P) Concentration (C)	$\mathcal{H}(\mathbf{h}_{\mathbf{x}})$ $\max_c(\mathbf{h}_{\mathbf{x}})$ $e^{g_{\mathbf{x}}}$

3) Uncertainty Measures and Evaluation Metrics: As shown in Tab. III, the uncertainty measures are the discrete entropy (E) and maximum probability (P) for the teacher, and E, P, and concentration (C) for the student, similar to Section IV-A3. Furthermore, to show the amortization gap, we use the differential entropy (D) of a locally fitted Dirichlet $q^*_{\mathbf{x}}$, as presented in Sec. III-E, The D of $q^*_{\mathbf{x}}$ is expected to be the best uncertainty measure that OPU can approach theoretically (when the amortization loss is zero). For fairness, the same set of posterior particles is used for training D. The evaluation metrics used are the same as in Sec. IV-A3.

B. Results on MNIST

We next present the results on MNIST, which appear in Tables IV to IX.

1) NN implementation: The NN architecture used by these models is an MLP with size 784-400-400-10, ReLU activations, and softmax outputs, following Balan et al. [3]. For the concentration model, we use an MLP with size 784-400-400-1. MCDP is trained by SGD with hyper-parameters: dropout-rate of 0.5, learning rate 5×10^{-4}, mini-batch size of 256, number of iterations 10^3. For MMD, we use a summation of RBF kernel and polynomial kernel. OPU-MCDP is trained by Adam with hyper-parameters: number of iterations 100, learning rate for student 10^{-3}. The training of SGLD and BDK follows Balan et al. [3]. Then OPU-SGLD is trained with the same hyperparameters as OPU-MCDP. Results of DPN are from Malinin and Gales [25].

2) Computation time: The main results of OPU are shown in Tab. IV. For comparison, we also show the detailed results of using the original samples from the Bayes teacher MCDP/SGLD in Tab. V, whose values might be covered by differential entropy of locally fitted Dirichlet (D) in Tab. IV. OPU offers a $\sim$500x speedup compared to the original MCDP/SGLD, as OPU only evaluates the model twice (PM and CM in the student network) while MCDP/SGLD evaluates for S times. This confirms our idea of accelerating Bayesian prediction by *diverting the sampling process from the test period to the approximation period*. Note that the time cost of MCDP/SGLD increases with more posterior samples involved. BDK is slightly faster than OPU because it runs one network while OPU runs both PM and CM.

3) Comparisons with BDK: As seen in Tab. IV, in some tasks especially OOD detection, the concentration measure (CM) outperforms the BDK baseline. This shows the explicit disentanglement of mean and concentration helps "targeted" knowledge distillation, as shown in Sec. III-C. By comparing OPU-SGLD-KL and BDK (trained by forward KL), we observe that OPU-SGLD-KL is significantly better in OOD detection tasks and MisC detection measured by AUROC. BDK shows a slight advantage in the MisC detection task when measured by AUPR. The reason is knowledge distillation only happens between two categorical variables in BDK, which only captures prediction information. In contrast, OPU first extracts all information in a BNN with the induced distribution, then transfers the knowledge to a more expressive distribution with a guaranteed small loss (Sec. III-E). Adding a Dirichlet distribution to BDK (BDK-DIR-SGLD) helps to improve the performance of OOD detection. However, on SEMEION, which is expected to be harder as it is more similar to MNIST, there is a large performance difference from OPU. This further validates the necessity of explicit disentanglement of the mean and concentration.

4) Comparisons with DPN: OPU (without OOD data in training) has comparable performance to DPN (which uses a hand-crafted goal and OOD data in training). Another reason that DPN performs slightly better is that DPN uses VGG-6 (4 Convolutional layer and 1 FC layer), which is a much stronger model than the 2-layer MLP model that other models use.

5) Comparing KL, EMD, MMD: We next compare the probability distances ρ used to train the approximations (the amortized OPU and the locally-fit Dirichlet). As shown in Tab. IV, MCDP-MMD and OPU-MCDP-MMD have the best

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

TABLE IV: Results on MNIST for Bayesian Neural networks. The double quote means "same as above".

Model	MisC detection AUROC	MisC detection AUPR	OOD det. (Omniglot) AUROC	OOD det. (Omniglot) AUPR	OOD det. (SEMEION) AUROC	OOD det. (SEMEION) AUPR	Acc. (%)	Test time(s)
MCDP-KL	**97.3**$\pm$0.5 (E)	43.0$\pm$0.1 (E)	99.4$\pm$0.2 (D)	99.7$\pm$0.1 (D)	86.4$\pm$2.2 (E)	53.9$\pm$1.7 (P)	**97.9**$\pm$0.5	210.6$\pm$0.5
MCDP-EMD	**97.3**$\pm$0.5 (E)	43.0$\pm$0.1 (E)	99.6$\pm$0.1 (D)	99.9$\pm$0.0 (D)	86.4$\pm$2.2 (E)	53.9$\pm$1.7 (P)	**97.9**$\pm$0.5	"
MCDP-MMD	**97.3**$\pm$0.5 (E)	43.0$\pm$0.1 (E)	99.7$\pm$0.2 (D)	99.9$\pm$0.0 (D)	90.1$\pm$0.7 (D)	71.2$\pm$2.5 (D)	**97.9**$\pm$0.5	"
OPU-MCDP-KL	94.2$\pm$0.8 (E)	37.7$\pm$1.3 (E)	**100**$\pm$0.0 (C)	77.1$\pm$0.1 (C)	91.4$\pm$0.0 (C)	67.3$\pm$0.2 (C)	96.2$\pm$0.2	**0.441**$\pm$0.002
OPU-MCDP-EMD	95.3$\pm$0.3 (P)	**43.8**$\pm$0.9 (P)	**100**$\pm$0.0 (C)	**100**$\pm$0.0 (C)	93.3$\pm$0.2 (C)	82.5$\pm$0.5 (C)	96.1$\pm$0.2	"
OPU-MCDP-MMD	97.2$\pm$0.2 (P)	41.1$\pm$0.6 (P)	**100**$\pm$0.0 (C)	**100**$\pm$0.0 (C)	**99.8**$\pm$0.0 (C)	**98.6**$\pm$0.1 (C)	**97.9**$\pm$0.1	"
SGLD-KL	**97.9**$\pm$0.3 (P)	46.2$\pm$0.2 (E)	99.2$\pm$0.1 (E)	99.6$\pm$0.1 (E)	89.6$\pm$0.9 (E)	47.0$\pm$1.4 (E)	**98.4**$\pm$0.3	233.5$\pm$0.1
SGLD-EMD	**97.9**$\pm$0.3 (E)	46.2$\pm$0.2 (E)	99.4$\pm$0.1 (D)	99.7$\pm$0.0 (D)	89.9$\pm$0.2 (D)	47.1$\pm$0.7 (D)	**98.4**$\pm$0.3	"
SGLD-MMD	**97.9**$\pm$0.3 (E)	46.2$\pm$0.2 (E)	99.2$\pm$0.1 (E)	99.6$\pm$0.1 (E)	89.6$\pm$0.9 (E)	47.0$\pm$1.4 (E)	**98.4**$\pm$0.3	"
OPU-SGLD-KL	94.2$\pm$1.5 (E)	**46.7**$\pm$0.9 (E)	**100**$\pm$0.0 (C)	**100**$\pm$0.0 (C)	**99.5**$\pm$0.1 (C)	**98.4**$\pm$0.0 (C)	98.2$\pm$0.2	**0.443**$\pm$0.002
OPU-SGLD-EMD	93.7$\pm$1.7 (P)	44.4$\pm$0.6 (E)	**100**$\pm$0.0 (C)	**100**$\pm$0.0 (C)	98.9$\pm$0.3 (C)	96.2$\pm$0.3 (C)	98.0$\pm$0.1	"
OPU-SGLD-MMD	97.2$\pm$0.7 (P)	44.6$\pm$0.5 (E)	**100**$\pm$0.0 (C)	**100**$\pm$0.0 (C)	99.1$\pm$0.1 (C)	98.0$\pm$0.0 (C)	98.1$\pm$0.1	"
BDK-SGLD	85.9$\pm$1.3 (E)	**46.6**$\pm$2.7 (E)	46.1$\pm$0.8 (E)	41.7$\pm$2.3 (E)	35.3$\pm$3.1 (P)	46.5$\pm$1.4 (P)	92.1$\pm$0.5	**0.441**$\pm$0.002
BDK-MCDP	86.9$\pm$1.9 (E)	41.1$\pm$1.5 (E)	47.5$\pm$1.1 (P)	44.1$\pm$2.6 (P)	43.3$\pm$1.9 (P)	**47.2**$\pm$1.0 (P)	92.4$\pm$0.3	"
BDK-DIR-SGLD	**89.9**$\pm$0.7 (E)	40.0$\pm$1.1 (E)	**95.4**$\pm$0.6 (E)	**96.4**$\pm$0.3 (E)	**74.7**$\pm$0.9 (E)	38.3$\pm$1.2 (E)	**94.1**$\pm$0.1	"
DPN	99.0 (E)	43.6 (E)	100 (E)	100 (E)	99.7 (E)	98.6 (E)	99.4	/

TABLE V: Results of using the original samples from the Bayes teacher on MNIST.

Model	MisC detection AUROC	MisC detection AUPR	OOD (Omniglot) AUROC	OOD (Omniglot) AUPR	OOD (SEMEION) AUROC	OOD (SEMEION) AUPR	Acc. (%)
MCDP	97.3$\pm$0.5 (E)	43.0$\pm$0.1 (E)	**99.2**$\pm$0.3 (E)	98.8$\pm$0.2 (P)	86.4$\pm$2.2 (E)	**53.9**$\pm$1.7 (P)	97.9$\pm$0.5
SGLD	**97.9**$\pm$0.3 (E)	**46.2**$\pm$0.2 (E)	**99.2**$\pm$0.1 (E)	**99.6**$\pm$0.1 (E)	**89.6**$\pm$0.9 (E)	47.0$\pm$1.4 (E)	98.4$\pm$0.3

TABLE VI: Results on error/gap on EMNIST.

AvgApproxErr	0.0650
AvgModelErr	0.0601
AvgAmtGap	0.0049
AvgAmtErr	0.0053

TABLE VII: Results on EMNIST for Bayesian NN

Model	MisC detection AUROC	MisC detection AUPR	OOD (Omniglot) AUROC	OOD (Omniglot) AUPR	Acc. (%)
MCDP-KL	89.7$\pm$0.1 (P)	46.8$\pm$0.2 (P)	99.7$\pm$0.0 (E)	99.7$\pm$0.0 (E)	**88.8**$\pm$0.3
MCDP-MMD	89.7$\pm$0.1 (P)	46.8$\pm$0.2 (P)	99.9$\pm$0.0 (D)	99.9$\pm$0.0 (D)	**88.8**$\pm$0.3
OPU-MCDP-KL	84.9$\pm$0.5 (P)	40.7$\pm$0.6 (P)	96.2$\pm$0.4 (E) /67.5$\pm$0.2 (C)	96.5$\pm$0.3 (E) /63.7$\pm$0.1 (C)	87.9$\pm$0.1
OPU-MCDP-MMD	**89.8**$\pm$0.2 (P)	**49.6**$\pm$0.3 (P)	**100.0**$\pm$0.0 (C)	**100**$\pm$0.0 (C)	88.4$\pm$0.1

TABLE VIII: Results on MNIST for MCDP/SGLD (Bayes teacher) with half samples. The number in parentheses is the performance degradation compared to using all samples.

Model	MisC AUROC	Omn. AUROC	SEM. AUROC	Acc. (%)	Time (s)
MCDP	96.1$_{(-1.2)}$	98.5$_{(-0.9)}$	82.7$_{(-4.1)}$	96.9$_{(-1.0)}$	132.1
SGLD	97.1$_{(-0.9)}$	91.2$_{(-8.0)}$	82.5$_{(-6.8)}$	98.0$_{(-0.4)}$	141.9

TABLE IX: Results for Bayesian NN on Cifar10. ("M" denotes "MCDP". "O" denotes "OPU-MCDP". "MSP" denotes the Max.P results, "M-GAN" denotes the Max.P results with GAN-generated samples and "OE" is with outlier exposure from Hendrycks et al. [15]. "ODIN" is the results from Liang et al. [22].)

Model	MisC detection AUROC	MisC detection AUPR	OOD (LSUN) AUROC	OOD (LSUN) AUPR	Acc. (%)
M-KL	**92.2**$\pm$0.4(P)	**47.0**$\pm$0.2(P)	90.5$\pm$0.4(E)	88.7$\pm$0.2(E)	**92.4**
M-EMD	**92.2**$\pm$0.4(P)	**47.0**$\pm$0.2(P)	91.4$\pm$0.1(D)	89.1$\pm$0.1(D)	**92.4**
M-MMD	**92.2**$\pm$0.4(P)	**47.0**$\pm$0.2(P)	91.0$\pm$0.2(D)	89.3$\pm$0.1(D)	**92.4**
O-KL	87.2$\pm$0.6(P)	45.9$\pm$0.3(P)	86.1$\pm$0.7(E)	85.5$\pm$1.2(E)	89.9
O-EMD	91.8$\pm$0.3(E)	46.9$\pm$0.1(P)	**93.5**$\pm$0.1(C)	**92.0**$\pm$0.3(C)	91.8
O-MMD	91.3$\pm$0.2(E)	46.6$\pm$0.1(P)	92.9$\pm$0.2(C)	91.7$\pm$0.2(C)	91.8
MSP	/	/	88.1	/	/
M-GAN	/	/	89.6	/	/
OE	/	/	**97.8**	/	/
ODIN	/	/	91.3	/	/

performance, using differential entropy (D) and concentration measure (CM), respectively. This is because the samples of MCDP are relatively spread out over the simplex and might be multi-modal. The probability distance is fully captured by MMD under such case. EMD is expected to perform well as there are a lot of samples $\boldsymbol{\pi}_s$ residing on a low-dimensional manifold. However, the performance seems to be degenerated due to the limited capacity of the hyper-network and the difficulty to train the minimax problem. KL has the worst performance as expected because it is likely to be under confident with samples of MCDP. For SGLD, the performance using KL is the best except for AUROC of MisC detection. This is because the samples $\boldsymbol{\pi}_s$ of SGLD over the simplex are much denser and are typically unimodal.

6) Teacher with fewer samples: We also show the performance of the original MCDP/SGLD with fewer samples in Tab. VIII, to illustrate the necessity of a careful and accurate approximation of predictive distribution. Using fewer particles negatively affects the performance of the Bayesian classifier, especially the performance on OOD detection. In real-world cases like an automatic driving car where the robustness is critical, it is not worth to trade safety for speed. Therefore, OPU solves this issue by providing an accurate estimation of predictive uncertainty with a short evaluation time. Besides the speedup, OPU approximation provides a full distribution to characterize the predictive distribution, which is not available with particle approximations. This allows for better uncertainty measures such as differential entropy.

C. Results on EMNIST

The experiments are conducted on EMNIST whose number of classes is large, to study the performance on OPU fitted on a larger simplex and validate the theoretical analysis. We choose Omniglot as the OOD detection dataset as it also contains handwritten characters, which are harder than SEMEION for a model trained on EMNIST.

1) NN implementation: A CNN with a structure similar to LeNet is used for MCDP and OPU (20 and 50 output channels in two convolutional layers). The baseline approach achieves a classification accuracy of 88.8%. The samples of MCDP Bayes NN are expected to be even more dispersive and multi-modal than MCDP trained with MNIST. OPU trained with EMD failed to converge, possibly because the capacity of the hypernetwork was not enough. Therefore, we do not recommend to use EMD for the amortized approximation of predictive uncertainty, unless a more scalable estimator of EMD can be provided.

2) Results: As shown in Tab. VII, the performance gap between OPU-MCDP-KL and the baseline is larger because the multi-modality is severe – the entropy of sample mean is a better measure for OOD detection than the concentration. This further validates that OPU trained by KL suffers from an under-confidence issue. Specifically, KL forces OPU to cover the support of all samples, making the student distribution more dispersive. This inaccurate estimate of concentration affects the estimation of prediction results (mean) in turn, thus the accuracy is lower and MisC detection performance suffers. In contrast, MMD consistently provides an approximation that has similar performance with the teacher, which echoes the analysis in Sec. III-E.

3) Approximation Error: To validate the theoretical study, we use the four types of error/gap under MMD defined in Sec. III-E to show the effectiveness of amortized approximation and the suitability of using the Dirichlet family. Specifically, the following measures are used:

- *Averaged total approximation error* (AvgApproxErr): The averaged MMD between teacher's particles (for each x) and the predicted Dirichlet by OPU, i.e., $\frac{1}{N}\sum_{i=1}^{N}\mathrm{MMD}(q_{\mathbf{x}_i}, p_{\mathbf{x}_i})$.
- *Averaged total model error* (AvgModelErr): The averaged MMD between teacher's particles (for each x) and locally fitted Dirichlet, i.e., $\frac{1}{N}\sum_{i=1}^{N}\mathrm{MMD}(p_{\mathbf{x}_i}, \bar{q}^*_{\mathbf{x}_i})$.
- *Averaged local amortization gap* (AvgAmtGap): The difference between AvgApproxErr and AvgModelErr, i.e., $\Delta(\mathbf{x})$, defined in Eq. 12.
- *Averaged local amortization error* (AvgAmtErr): We define this type of error to be the averaged MMD between locally fitted Dirichlet (for each x) and the predicted Dirichlet by OPU, i.e., $\frac{1}{N}\sum_{i=1}^{N}\mathrm{MMD}(q_{\mathbf{x}_i}, \bar{q}^*_{\mathbf{x}_i})$. The relation between AvgAmtGap and AvgAmtErr is given by Eq.13.

We show the numerical results on EMNIST dataset in Tab. VI. It can be observed that the AvgAmtErr of 0.0053 is low and it bounds the AvgAmtGap, $\Delta(\mathbf{x}) = \text{AvgApproxErr} - \text{AvgModelErr} = 0.0049$, which is consistent with Eq. 13. The approximation error (AvgApproxErr) is mainly determined by the model error (AvgModelErr), which turns out to be acceptably small. This is consistent with the analysis and also shows the effectiveness and suitability of using the Dirichlet family.

D. Results on Cifar10

1) NN implementation: The NN model for teacher and student is standard VGG19 with the output dimension of the concentration model to be 1. The MCDP teacher model is trained with an SGD optimizer and a cyclical learning rate policy. The base learning rate is initialized to be 0.01 and linearly increased to 10x of the learning rate, then decreases back to the base learning rate. The base learning rate is then scaled by half. The number of MCDP particles is 700. The OPU students are trained with KL/EMD/MMD using the corresponding algorithm for 200 epochs.

2) Results: We show the experimental results of OPU approximating Bayesian NN on the Cifar10 dataset in Tab. IX. As the task is harder than MNIST, particles tend to be more spread over the corners, leading to more multi-modal predictive distributions. Under such case, it can be observed that the performance of EMD and MMD is much better than that of KL.

We compare with recent works on predictive uncertainty evaluation. OPU outperforms Max. P results trained with regular data (MSP) and with GAN-generated samples (M-GAN). The model with outlier exposure (OE) performs the best, but requires explicitly training on OOD data, which is unrealistic for real-world applications. OPU also performs ODIN, which is based on temperature scaling and input preprocessing, on out-of-domain detection.

VI. Conclusions

In this paper, we propose a generic framework that efficiently approximates the predictive distribution induced by a model posterior in an amortized fashion. The proposed framework is universally applicable to Bayesian methods that generate posterior samples, including both parametric and non-parametric models. Compared with traditional Monte Carlo Bayesian methods, our framework obtains the predictive distribution with a single pass of a neural network. By using a more expressive predictive distribution, our framework outperforms CompactApprox, BDK, and SVGP. We also explore different probability distances (i.e., forward KL, EMD, and MMD) for learning, and analyze the amortization gap theoretically and empirically. The experimental results show an acceptably low amortization gap.

The choice of student distribution for OPU is flexible. Although we verify that Dirichlet is suitable for the tested classification tasks, it is still possible that Dirichlet might incur large model errors. Note that, in our framework, using a Dirichlet for the student is a modeling choice, similar to assuming Gaussian posteriors for variational approximations. The OPU framework is general and any student distribution can be adopted, e.g., generalized Dirichlet, a mixture of Dirichlets. To do this requires: 1) a suitable parametrization of the model that can capture uncertainty; 2) deriving/computing the approximation loss (KL, MMD, EMD); 3) reparameterization

trick of expectations for efficient gradient estimation. The algorithms of KL, EMD, and MMD as well as the analysis still apply.

The idea of "transferring" the randomness from model posterior to a simple-structure distribution at the output can be generalized to other problems where a real-time evaluation of uncertainty is critical, e.g., object segmentation. This allows interesting designs of structured output distributions.

Regarding future work, there are two areas to explore. First, we may consider better metrics for student learning that are more robust, e.g., by designing a more accurate and efficient estimator for EMD or exploring different kernels for better estimation of MMD. Second, we will consider applying OPU to other computer vision tasks, such as image segmentation and object detection.

REFERENCES

[1] John Aitchison. The statistical analysis of compositional data. *Journal of the Royal Statistical Society: Series B (Methodological)*, 44(2):139–160, 1982.

[2] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein GAN. *arXiv preprint arXiv:1701.07875*, 2017.

[3] Anoop Korattikara Balan, Vivek Rathod, Kevin P Murphy, and Max Welling. Bayesian dark knowledge. In *Advances in Neural Information Processing Systems*, pages 3438–3446, 2015.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006. ISBN 0387310738.

[5] Samuel Rota Bulò, Lorenzo Porzi, and Peter Kontschieder. Dropout distillation. In *International Conference on Machine Learning*, pages 99–107, 2016.

[6] Rohitash Chandra and Arpit Kapoor. Bayesian neural multi-source transfer learning. *Neurocomputing*, 378:54–64, 2020.

[7] Rohitash Chandra, Konark Jain, Ratneel V Deo, and Sally Cripps. Langevin-gradient parallel tempering for bayesian neural learning. *Neurocomputing*, 359:315–326, 2019.

[8] Mikhail Figurnov, Shakir Mohamed, and Andriy Mnih. Implicit reparameterization gradients. In *Advances in Neural Information Processing Systems*, pages 441–452, 2018.

[9] Maurizio Filippone, Mingjun Zhong, and Mark Girolami. A comparative evaluation of stochastic-based inference methods for gaussian process models. *Machine Learning*, 93(1):93–114, 2013.

[10] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *International conference on machine learning*, pages 1050–1059, 2016.

[11] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018.

[12] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville. Improved training of Wasserstein GANs. In *Advances in Neural Information Processing Systems*, pages 5767–5777, 2017.

[13] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[14] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *Proceedings of International Conference on Learning Representations*, 2017.

[15] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyxCxhRcY7.

[16] James Hensman, Alexander G Matthews, Maurizio Filippone, and Zoubin Ghahramani. MCMC for variationally sparse Gaussian processes. In *Advances in Neural Information Processing Systems*, pages 1648–1656, 2015.

[17] James Hensman, Alexander G de G Matthews, and Zoubin Ghahramani. Scalable variational gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics*, 2015.

[18] Yen-Chang Hsu, Yilin Shen, Hongxia Jin, and Zsolt Kira. Generalized odin: Detecting out-of-distribution image without learning from out-of-distribution data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[19] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in neural information processing systems*, pages 4107–4115, 2016.

[20] Weonyoung Joo, Wonsung Lee, Sungrae Park, and Il-Chul Moon. Dirichlet variational autoencoder. *Pattern Recognition*, page 107514, 2020.

[21] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[22] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=H1VGkIxRZ.

[23] Xiaofan Lin, Cong Zhao, and Wei Pan. Towards accurate binary convolutional neural network. In *Advances in Neural Information Processing Systems*, pages 345–353, 2017.

[24] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[25] Andrey Malinin and Mark Gales. Predictive uncertainty estimation via prior networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 7047–7058. Curran Associates, Inc., 2018.

[26] Yishu Miao, Edward Grefenstette, and Phil Blunsom. Discovering discrete latent topics with neural variational

inference. *arXiv preprint arXiv:1706.00359*, 2017.

[27] Christian Naesseth, Francisco Ruiz, Scott Linderman, and David Blei. Reparameterization gradients through acceptance-rejection sampling algorithms. In *Artificial Intelligence and Statistics*, pages 489–498, 2017.

[28] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11(5-6):355–607, 2019.

[29] Nicholas G Polson, James G Scott, and Jesse Windle. Bayesian inference for logistic models using pólya–gamma latent variables. *Journal of the American statistical Association*, 108(504):1339–1349, 2013.

[30] Maithra Raghu, Katy Blumer, Rory Sayres, Ziad Obermeyer, Bobby Kleinberg, Sendhil Mullainathan, and Jon Kleinberg. Direct uncertainty prediction for medical second opinions. volume 97 of *Proceedings of Machine Learning Research*, pages 5281–5290, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/raghu19a.html.

[31] Edward Snelson and Zoubin Ghahramani. Compact approximations to Bayesian predictive distributions. In *Proceedings of the 22nd international conference on Machine learning*, pages 840–847. ACM, 2005.

[32] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018.

[33] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558*, 2017.

[34] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

[35] Max Welling and Yee Whye Teh. Bayesian learning via stochastic gradient Langevin dynamics. In *Proceedings of the 28nd international conference on Machine learning*, pages 681–688. ACM, 2011.

[36] Qizhe Xie, Minh-Thang Luong, Eduard Hovy, and Quoc V Le. Self-training with noisy student improves imagenet classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10687–10698, 2020.

APPENDIX A
PROOF

Assumption 1. *Let $\mathcal{T} : \mathcal{X} \times \Theta \to \mathcal{Y}$ be a map between finite dimensional vector spaces. We say $\mathcal{T}$ satisfies Assumption 1 for distribution p if $\mathcal{T}(\cdot; \boldsymbol{\theta})$ is Lipschitz and the Lipschitz constant $L_{\boldsymbol{\theta}}$ satisfies $\mathbb{E}_{\boldsymbol{\theta}\sim p} L_{\boldsymbol{\theta}} < +\infty$.*

Proof. We simply show both maps are Lipschitz continuous with MMD metric on $\mathcal{P}(\mathcal{S}^{K-1})$. Let $\mathbf{x}, \mathbf{x}' \in \mathcal{X}$ and $\psi \in \mathcal{H}_k$ such that $\|\psi\|_{\mathcal{H}_k} \leq 1$. For $\mathbf{x} \mapsto p_{\mathbf{x}}$, the "push-forward" definition of $p_{\mathbf{x}}$ leads to

$$\begin{aligned}
\mathrm{MMD}_k(p_{\mathbf{x}}, p_{\mathbf{x}'}) &= \sup_{\|\psi\|\leq 1} \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta})) - \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\psi(\mathcal{T}_{\mathbf{x}'}(\boldsymbol{\theta})) \\
&\leq \sup_{\|\psi\|\leq 1} \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}|\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta})) - \psi(\mathcal{T}_{\mathbf{x}'}(\boldsymbol{\theta}))| \\
&\leq \sup_{\|\psi\|\leq 1} \|\psi\|\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})} d_k(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}), \mathcal{T}_{\mathbf{x}'}(\boldsymbol{\theta})) \\
&\leq C\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\|\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}) - \mathcal{T}_{\mathbf{x}'}(\boldsymbol{\theta})\| \\
&\leq C\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})} L_{\boldsymbol{\theta}}\|\mathbf{x} - \mathbf{x}'\|,
\end{aligned}$$

where $d_k =$ is a kernel-based distance. The constant C emits when bounding d_k with Euclidean norm and the existence of such a constant is due to topology equivalence in finite-dimensional space. For $\mathbf{x} \to \bar{q}^*_{\mathbf{x}}$, we notice that $\bar{q}^*_{\mathbf{x}}$ is the projection of $p_{\mathbf{x}}$ onto $\mathcal{Q}$ (effectively the projection of kernel mean embeddings in $\mathcal{H}_k$). By projection theorem in Hilbert space, the projection map $p_{\mathbf{x}} \mapsto \bar{q}^*_{\mathbf{x}}$ is non-expansive, i.e.

$$\mathrm{MMD}_k(\bar{q}^*_{\mathbf{x}}, \bar{q}^*_{\mathbf{x}'}) \leq \mathrm{MMD}_k(p_{\mathbf{x}}, p_{\mathbf{x}'}),$$

which leads to Lipschitz property of $\mathbf{x} \mapsto \bar{q}^*_{\mathbf{x}}$. □

APPENDIX B
ALGORITHMS.

For presenting the algorithms, we slightly change the notation. We let $\boldsymbol{\alpha}(\mathbf{x}) = \boldsymbol{\alpha}(\mathbf{x}, \boldsymbol{\phi})$, $\mathbf{h}(\mathbf{x}) = \mathbf{h}(\mathbf{x}, \boldsymbol{\phi}_1)$, $g(\mathbf{x}) = g(\mathbf{x}, \boldsymbol{\phi}_2)$, where $\boldsymbol{\phi} = \{\boldsymbol{\phi}_1, \boldsymbol{\phi}_2\}$, $\boldsymbol{\phi}_1$ and $\boldsymbol{\phi}_2$ are the parameters of prediction model $\mathbf{h}$ and concentration model g, respectively.

KL. With the training objective

$$\min_{\boldsymbol{\phi}} \quad -\mathbb{E}_{p_{\mathcal{D}'}(\mathbf{x})} \frac{1}{S} \sum_s \ln q(\mathcal{T}(\mathbf{x}; \boldsymbol{\theta}_s)|\mathbf{x}, \boldsymbol{\alpha}), \tag{14}$$

The student model is updated by doing $\boldsymbol{\phi}_1^{t+1} := \boldsymbol{\phi}_1^t - \gamma^t(-\frac{1}{S}\sum_{\boldsymbol{\theta}_s} \nabla_{\boldsymbol{\phi}_1} \ln q(\mathcal{T}(\mathbf{x}^*; \boldsymbol{\theta}_s)|\alpha(\mathbf{x}^*; \boldsymbol{\phi}_1)))$ and $\boldsymbol{\phi}_2^{t+1} := \boldsymbol{\phi}_2^t - \gamma^t(-\frac{1}{S}\sum_{\boldsymbol{\theta}_s} \nabla_{\boldsymbol{\phi}_2} \ln q(\mathcal{T}(\mathbf{x}^*; \boldsymbol{\theta}_s)|\alpha(\mathbf{x}^*; \boldsymbol{\phi}_2)))$ alternately, where γ^t is the learning rate at iteration t and $\mathbf{x}^*$ is the input at this iteration.

EMD. With the following training objective,

$$\begin{aligned}
\min_{\boldsymbol{\phi}} \max_{w} \quad & \mathbb{E}_{p_{\mathcal{D}'}(\mathbf{x})}[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}[\psi(\mathcal{T}_{\mathbf{x}}(\boldsymbol{\theta}); w, \mathbf{x})] \\
& -\mathbb{E}_{q_{\mathbf{x}}}[\psi(\boldsymbol{\pi}; w, \mathbf{x})]] + \lambda\mathcal{R}(w)
\end{aligned} \tag{15}$$

MMD. The kernel mean embedding of $p_{\mathbf{x}}$ and $q_{\mathbf{x}}$ are given by $\mu_p = \mathbb{E}_{p_{\mathbf{x}}}[k(\boldsymbol{\pi}, \cdot)]$ and $\mu_q = \mathbb{E}_{q_{\mathbf{x}}}[k(\boldsymbol{\pi}, \cdot)]$. The training objective is then,

$$\min_{\boldsymbol{\phi}} \|\mu_p - \mu_q\|_{\mathcal{H}_k}, \tag{16}$$

APPENDIX C
SAMPLING

In this section, we illustrate the details for extracting samples from Bayesian logistic regression, Bayesian neural network, and Gaussian process. Under some contexts, we use $\mathbf{X}$ and $\mathbf{Y}$ to collectively denote the inputs and outputs respectively for previous $\mathcal{D}$.

Algorithm 1: OPU Training Algorithm with EMD

Input: Posterior samples: $\{\boldsymbol{\theta}_s\}_{s=1}^{S}$; OPU training data $\mathcal{D}'$; Gradient penalty coefficient λ; Number of training iterations: T_{stu} and T_{wit}.

while ϕ *not converge* **do**

  Sample $\mathbf{x}^{(i)} \sim p_{\mathcal{D}'}(\mathbf{x})$

  /* Update Approximation */

  for *iter in* $1 \dots T_{\text{stu}}$ **do**

    Sample $\{\boldsymbol{\pi}_{s'}\}_{s'=1}^{S'} \sim q(\boldsymbol{\pi}|\mathbf{x}^{(i)}, \boldsymbol{\phi})$

    $\mathcal{L}^{(i)}(\boldsymbol{\phi_1}) = -\sum_{s'=1}^{S'} \psi(\boldsymbol{\pi}_{s'}; w, \mathbf{x}^{(i)})$

    $\boldsymbol{\phi}_1 \leftarrow \text{Adam}(\nabla_{\boldsymbol{\phi}_1} \mathcal{L}^{(i)})$

    Sample $\{\boldsymbol{\pi}_{s'}\}_{s'=1}^{S'} \sim q(\boldsymbol{\pi}|\mathbf{x}^{(i)}, \boldsymbol{\phi})$

    $\mathcal{L}^{(i)}(\boldsymbol{\phi}_2) = -\sum_{s'=1}^{S'} \psi(\boldsymbol{\pi}_{s'}; w, \mathbf{x}^{(i)})$

    $\boldsymbol{\phi}_2 \leftarrow \text{Adam}(\nabla_{\boldsymbol{\phi}_2} \mathcal{L}^{(i)})$

  end

  /* Update Critic */

  for *iter in* $1 \dots T_{\text{wit}}$ **do**

    Sample $\{\boldsymbol{\pi}_{s'=1}^{S'}\} \sim q(\boldsymbol{\pi}|\mathbf{x}^{(i)}, \boldsymbol{\phi})$

    Compute $\mathcal{R}(v)$

    $\mathcal{L}^{(i)}(v) = \sum_{s=1}^{S} \psi(\mathcal{T}(\mathbf{x}; \boldsymbol{\theta}_s); w, \mathbf{x}^{(i)}) - \sum_{s'=1}^{S'} \psi(\boldsymbol{\pi}_{s'}; w, \mathbf{x}^{(i)}) + \lambda \mathcal{R}(w)$

    $v \leftarrow \text{Adam}(\nabla_v \mathcal{L}^{(i)})$

  end

end

Algorithm 2: OPU Training Algorithm with MMD

Input: Posterior samples: $\{\boldsymbol{\theta}_s\}_{s=1}^{S}$; OPU training data $\mathcal{D}'$; Gradient penalty coefficient λ.

while ϕ *not converge* **do**

  Sample $\mathbf{x}^{(i)} \sim p_{\mathcal{D}'}(\mathbf{x})$

  Sample $\{\boldsymbol{\pi}_{q,s'}\}_{s'=1}^{S'} \sim q(\boldsymbol{\pi}|\mathbf{x}^{(i)}, \boldsymbol{\phi})$, get $\{\boldsymbol{\pi}_{p,s'}\}_{s'=1}^{S'} = \{\mathcal{T}(\mathbf{x}|\boldsymbol{\theta}_{s'})\}_{s'=1}^{S'}$

  $\mathcal{L}^{(i)}(\boldsymbol{\phi_1}) = \frac{1}{S'(S'-1)} \sum_{m \neq n}^{S'} k(\boldsymbol{\pi}_{q,m}, \boldsymbol{\pi}_{q,m}) + \frac{1}{S'(S'-1)} \sum_{m \neq n}^{S'} k(\boldsymbol{\pi}_{p,m}, \boldsymbol{\pi}_{p,m}) - \frac{2}{S'(S'-1)} \sum_{m,n=1}^{S',S'} k(\boldsymbol{\pi}_{q,m}, \boldsymbol{\pi}_{p,n})$

  $\boldsymbol{\phi}_1 \leftarrow \text{Adam}(\nabla_{\boldsymbol{\phi}_1} \mathcal{L}^{(i)})$

  Sample $\{\boldsymbol{\pi}_{s'}\}_{s'=1}^{S'} \sim q(\boldsymbol{\pi}|\mathbf{x}^{(i)}, \boldsymbol{\phi})$

  $\mathcal{L}^{(i)}(\boldsymbol{\phi_1}) = \frac{1}{S'(S'-1)} \sum_{m \neq n}^{S'} k(\boldsymbol{\pi}_{q,m}, \boldsymbol{\pi}_{q,m}) + \frac{1}{S'(S'-1)} \sum_{m \neq n}^{S'} k(\boldsymbol{\pi}_{p,m}, \boldsymbol{\pi}_{p,m}) - \frac{2}{S'(S'-1)} \sum_{m,n=1}^{S',S'} k(\boldsymbol{\pi}_{q,m}, \boldsymbol{\pi}_{p,n})$

  $\boldsymbol{\phi} \leftarrow \text{Adam}(\nabla_{\boldsymbol{\phi}_2} \mathcal{L}^{(i)})$

end

A. Bayesian Logistic Regression

The Polya-Gamma (PG) scheme [29] is a data augmentation strategy that allows for a closed-form Gibbs sampler. In binary classification, i.e., $y_n \in \mathcal{Y} = \{1, 0\}$, let $\boldsymbol{\theta}$ be the regression coefficients with a Gaussian conditional conjugate prior $p(\boldsymbol{\theta}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1})$. The PG Gibbs sampler is composed of the following two conditionals,

$$\omega_n|\boldsymbol{\theta}, \mathbf{x}_n \sim \text{PG}(1, \mathbf{x}_n^{\mathsf{T}}\boldsymbol{\theta}) \quad (17)$$

$$\boldsymbol{\theta}|\boldsymbol{\omega}, \mathcal{D} \sim \mathcal{N}(\mathbf{m}_\omega, \mathbf{V}_\omega), \quad (18)$$

where ω_n is the augmenting data corresponding to the nth data point. The posterior conditional variance and mean are given by $\mathbf{V}_\omega = (\mathbf{X}^{\mathsf{T}}\Omega\mathbf{X} + \boldsymbol{\Lambda}^{-1})^{-1}$ and $\mathbf{m}_\omega = \mathbf{V}_\omega^{-1}(\mathbf{X}^{\mathsf{T}}(\mathbf{y} - \frac{1}{2}))$, respectively.

Given this formulation, we will be able to collect samples from the posterior after a burn-in period. The posterior samples $\{\boldsymbol{\theta}\}_{s=1}^{S}$, together with dataset $\mathcal{D}'$, are used to train our approximation with goal defined in Eq. 9.

As an MCMC method, the PG augmentation scheme offers accurate samples. Other alternative methods such as local variational approximation can be employed, which are much faster but sacrifice accuracy.

B. Monte Carlo Dropout

A neural network with dropout applied before every weight layer was shown to be an approximation to probabilistic deep Gaussian process (GP) [10]. Let $q(\boldsymbol{\theta})$ to be the approximate distribution to the GP posterior. Here, $\boldsymbol{\theta} = \{\boldsymbol{\Theta}_i\}_{i=1}^{L}$ and $\boldsymbol{\Theta}_i$ is a parameter matrix of dimensions $K_i \times K_{i=1}$ for NN layer i. In this approximation, $q(\boldsymbol{\theta})$ can be defined through direct modification:

$$\boldsymbol{\Theta}_i = \mathbf{M}_i \text{diag}([z_{i,j}]_{i=1}^{K_i}), \quad (19)$$

where $z_{i,j} \sim \text{Bern}(p_i)$ for $i \in [L]$ and $j \in [K_{i-1}]$, given some prior dropout probabilities p_i and matrices $\boldsymbol{\Theta}_i$ are treated as variational parameters. The binary variable $z_{i,j} = 0$ indicates that unit j in layer $i-1$ being dropped out as an input to layer i. The predictive mean of this approximation is given by $\mathbb{E}[\mathbf{y}|\mathbf{x}] \approx \frac{1}{S}\sum_{s=1}^{S} f(\mathbf{x}, \hat{z}_{1,s}, \dots, \hat{z}_{L,s})$, which hence referred to as MC dropout (MCDP).

We use OPU to approximate the uncertainty induced by $q(\boldsymbol{\theta})$. A sample in the MC ensemble is given by $\boldsymbol{\theta}_s = \{\boldsymbol{\Theta}_{i,s}\}_{i=1}^{L} = \mathbf{M}_i \text{diag}([z_{i,j}]_{i=1}^{K_i})$. We set $\boldsymbol{\phi}_1 = \{\boldsymbol{\Phi}_i\}_{i=1}^{L} Z$ to the mean of $\boldsymbol{\theta}_s$, i.e., $\boldsymbol{\Phi}_i = \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\Theta}_{i,s} = \mathbf{M}_i \frac{1}{S}\sum_{s=1}^{S} \text{diag}([z_{i,j,s}]_{i=1}^{K_i})$.

MCDP provides a simple way of approximating Bayesian inference through dropout sampling. However, it still introduces a variational approximation to the exact Bayesian posterior. Therefore, we further include a more accurate way to generate MC ensemble - stochastic gradient Langevin dynamics (SGLD).

C. Vanilla SGLD

SGLD enables mini-batch MC sampling from the posterior by adding a noise step to SGD [35]. We choose the "vanilla" version of SGLD in our approximation. Specifically, we start

This article has been accepted for publication in IEEE Transactions on Neural Networks and Learning Systems. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TNNLS.2020.3042525

training of $f_{\boldsymbol{\theta}}(\cdot)$ from $\boldsymbol{\theta}^{(0)}$. In each epoch with mini-batch size B,

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \epsilon_t \nabla \log p(\boldsymbol{\theta}^{(t)}|\mathcal{D}) + \eta_t \tag{20}$$

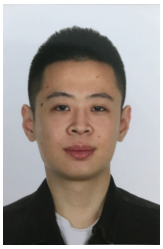
$$= \boldsymbol{\theta}^{(t)} + \epsilon_t \nabla (\log p(\boldsymbol{\theta}^{(t)}) + \sum_{b=1}^{B} \log p(\mathbf{y}_b|\mathbf{x}_b, \boldsymbol{\theta}^{(t)})) + \eta_t, \tag{21}$$

where B is the size of a mini-batch and $\eta_t \sim \mathcal{N}(0, 2\epsilon_t \mathbf{I})$. After SGLD converges at step T, the samples $\boldsymbol{\theta}_s = \boldsymbol{\theta}^{(T+s)}$ is collected by running the training process for another S iterations. We use averaged samples as parameter of h, i.e., $\phi_1 = \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{\theta}_s$ and train OPU by Eq. 9 with this ensemble.

D. Monte Carlo Gaussian Process

We apply the OPU framework to the GP framework, which demonstrates its use on a non-parametric classifier. Let data $\mathcal{D}$ be split into as input matrix $\mathbf{X}$ and output matrix $\mathbf{Y}$. We consider GP prior over the space of functions, i.e., $\boldsymbol{\mu} \sim \mathcal{GP}(0, \mathcal{K})$. where $\mathcal{K}$ is a positive definite kernel controlling the prior belief on smoothness. Existing techniques allow us to compute $q(\boldsymbol{\mu})$ which approximates $p(\boldsymbol{\mu}|\mathcal{D})$ and is comparable to previous $q(\boldsymbol{\theta})$ under a parametric model. In a classification task, the posterior $p(\boldsymbol{\mu}|\mathcal{D})$ can be sampled via MCMC [16] or approximated, e.g., via variational approximation [17]. Let $\boldsymbol{\mu}^*$ be a shorthand for $\boldsymbol{\mu}_{\mathbf{x}}$ and $\boldsymbol{\pi}$ is then defined as $\mathcal{S}(\boldsymbol{\mu}^*)$. If Gaussian variational approximation is used, the marginal posterior $p(\boldsymbol{\mu}^*|\mathbf{x}, \mathcal{D})$ at $\mathbf{x}$ induces a logistic-normal distribution for $p(\boldsymbol{\pi}|\mathbf{x}, \mathcal{D})$. In our approximation, we obtain samples $\{\boldsymbol{\mu}_s^*\}_{s=1}^S$ from $p(\boldsymbol{\mu}^*|\mathbf{x}, \mathcal{D})$. The optimization goal is the same as that in the NN case with a different target distribution defined as

$$\hat{p}(\boldsymbol{\pi}|\mathbf{x}, \mathcal{D}) = \frac{1}{S}\sum_{s=1}^{S} \delta(\boldsymbol{\pi} - \mathcal{S}(\boldsymbol{\mu}_s^*)). \tag{22}$$



Yufei Cui received the B.E. degree in Telecommunication from Shandong University, Shandong, China in 2015 and received the M.E. degree in Telecommunication from Hong Kong University of Science and Technology in 2016. He is currently working toward the PhD degree in the Department of Computer Science, City University of Hong Kong. His research interests include probabilistic models in machine learning and embedded system.

Wuguannan Yao received the B.Econ. degree from Hunan University, China, and M.Sc. degree from City University of Hong Kong, China, in 2015 and 2016, respectively. He is currently PhD student at Department of mathematics, City University of Hong Kong. His research interests include probabilistic models in machine learning and Bayesian inference.

Qiao Li received the B.S. and M.S. degrees in Computer Science and Technology from Chongqing University in China in 2014 and 2017 respectively. She is now a PhD candidate in Department of Computer Science, City University of Hong Kong. Her research interests include NAND flash memory, embedded systems and computer architecture.

Antoni B. Chan received the B.S. and M.Eng. degrees in electrical engineering from Cornell University, Ithaca, NY, in 2000 and 2001, and the Ph.D. degree in electrical and computer engineering from the University of California, San Diego (UCSD), San Diego, in 2008. He is currently an Associate Professor in the Department of Computer Science, City University of Hong Kong. His research interests include computer vision, machine learning, pattern recognition, and music analysis.

Chun Jason Xue received the B.S. degree in computer science and engineering from the University of Texas at Arlington, Arlington, TX, USA, in 1997, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2002 and 2007, respectively. He is currently an Associate Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include memory and parallelism optimization for embedded systems, software/hardware co-design, real-time systems, and computer security.