SAMPLING-FREE LEARNING OF BAYESIAN QUANTIZED NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Bayesian learning of model parameters in neural networks is important in scenarios where estimates with well-calibrated uncertainty are important. In this paper, we propose Bayesian quantized networks (BQNs), quantized neural networks (QNNs) for which we learn a posterior distribution over their discrete parameters. We provide a set of efficient algorithms for learning and prediction in BQNs without the need to sample from their parameters or activations, which not only allows for differentiable learning in quantized models but also reduces the variance in gradients estimation. We evaluate BQNs on MNIST, Fashion-MNIST and KMNIST classification datasets compared against bootstrap ensemble of QNNs (E-QNN). We demonstrate BQNs achieve both lower predictive errors and better-calibrated uncertainties than E-QNN (with less than 20% of the negative log-likelihood).

1 INTRODUCTION

A Bayesian approach to deep learning considers the network's parameters to be random variables and seeks to infer their posterior distribution given the training data. Models trained this way, called *Bayesian neural networks* (BNNs) (Wang & Yeung, 2016), in principle have well-calibrated uncertainties when they make predictions, which is important in scenarios such as active learning and reinforcement learning (Gal, 2016). Furthermore, the posterior distribution over the model parameters provides valuable information for evaluation and compression of neural networks.

There are three main challenges in using BNNs: **(1) Intractable posterior:** Computing and storing the exact posterior distribution over the network weights is intractable due to the complexity and high-dimensionality of deep networks. **(2) Prediction:** Performing a forward pass (a.k.a. as *probabilistic propagation*) in a BNN to compute a prediction for an input cannot be performed exactly, since the distribution of hidden activations at each layer is intractable to compute. **(3) Learning:** The classic *evidence lower bound* (ELBO) learning objective for training BNNs is not amenable to backpropagation as the ELBO is not an explicit function of the output of probabilistic propagation.

These challenges are typically addressed either by making simplifying assumptions about the distributions of the parameters and activations, or by using sampling-based approaches, which are expensive and unreliable (likely to overestimate the uncertainties in predictions). Our goal is to propose a **sampling-free** method which uses probabilistic propagation to deterministically learn BNNs.

A seemingly unrelated area of deep learning research is that of *quantized neural networks* (QNNs), which offer advantages of computational efficiency and reduced model size compared to continuous-valued neural networks. QNNs, like BNNs, face challenges in training, though for different reasons: **(4.1)** The non-differentiable activation function is not amenable to backpropagation. **(4.2)** Training is not compatible to gradient descent, since their parameters in QNNs are coarsely quantized that gradient updates cease to be meaningful.

In this work, we combine the ideas of BNNs and QNNs in a novel way that addresses the aforementioned challenges **(1)(2)(3)(4)** in training both models. We propose *Bayesian quantized networks* (BQNs), models that (like QNNs) have quantized parameters and activations over which they learn (like BNNs) categorical posterior distributions. BQNs have several appealing properties:

- BQNs solve challenge **(1)** due to their use of categorical distributions for their model parameters.

- BQNs can be trained via sampling-free backpropagation and stochastic gradient ascent of a differentiable lower bound to ELBO, which addresses challenges **(2)**, **(3)** and **(4)** above.
- BQNs leverage efficient tensor operations for probabilistic propagation, further addressing challenge **(2)**. We show the equivalence between probabilistic propagation in BQNs and tensor contractions (Kolda & Bader, 2009), and introduce a rank-1 CP tensor decomposition (mean-field approximation) that speeds up the forward pass in BQNs.
- BQNs provide a tunable trade-off between computational resource and model complexity: using a refined quantization allows for more complex distribution at the cost of more computation.
- Sampling from a learned BQN provides an alternative way to obtain deterministic QNNs .

In our experiments, we evaluate the expressive power of BQNs and show that BQNs trained using our sampling-free method have much better-calibrated uncertainty compared with the state-of-the-art *Bootstrap ensemble of quantized neural networks* (E-QNN) trained by (Courbariaux et al., 2016). More impressively, our trained BQNs achieve comparable log-likelihood against Gaussian *Bayesian neural network* (BNN) trained with *stochastic gradient variational Bayes* (SGVB) augmented by *local re-parameterization trick* (Shridhar et al., 2019) (the performance of BNN is expected to be better than BQN since the BNN allows for continuous parameters). We further verify that BQNs can be easily used to compress (Bayesian) neural networks, and obtain deterministic QNNs. Finally, we evaluate the effect of mean-field approximation in BQN by comparing with its Monte Carlo realizations, where no mean-field approximation is used. We show that our sampling-free probabilistic propagation using mean-field approximation achieves similar accuracy and log-likelihood — justifying the use of mean-field approximation in BQNs.

Related Works. In Appendix A, we survey different approaches for training bf Bayesian neural networks including *sampling-free assumed density filtering* (Ghosh et al., 2016; Soudry et al., 2014; Hernández-Lobato & Adams, 2015; Minka, 2001), *sampling-based variational inference* (Hinton & Van Camp, 1993; Graves, 2011; Blundell et al., 2015; Bengio et al., 2013; Maddison et al., 2016; Jang et al., 2016; Wu et al., 2018; Balan et al., 2015) as well as *sampling-free variational inference* (Wu et al., 2018), **Quantized Neural Networks** Han et al. (2015); Zhu et al. (2016); Courbariaux et al. (2015); Kim & Smaragdis (2016); Zhou et al. (2016); Rastegari et al. (2016); Hubara et al. (2017); Esser et al. (2015); Peters & Welling (2018); Shayer et al. (2017) and **Tensor Networks and Tensorial Neural Networks** (Grasedyck et al., 2013; Orús, 2014; Cichocki et al., 2016; 2017; Su et al., 2018; Newman et al., 2018; Robeva & Seigal, 2017).

Contributions:

- We propose an alternative *evidence lower bound (ELBO)* for Bayesian neural networks such that optimization of the variational objective is compatible with the backpropagation algorithm.
- We introduce Bayesian quantized networks (BQNs), establish a duality between BQNs and hierarchical tensor networks, and show prediction a BQN is equivalent to a series of tensor contractions.
- We derive a sampling-free approach for both learning and inference in BQNs using probabilistic propagation (analytical inference), achieving better-calibrated uncertainty for the learned models.
- We develop a set of fast algorithms to enable efficient learning and prediction for BQNs.

2 Bayesian Neural Networks

Notation. We use bold letters such as $\boldsymbol{\theta}$ to denote random variables, and non-bold letters such as θ to denote their realizations. We abbreviate $\mathbf{Pr}[\boldsymbol{\theta} = \theta]$ as $\mathbf{Pr}[\theta]$ and use bold letters in an equation if the equality holds for *arbitrary* realizations. For example, $\mathbf{Pr}[\boldsymbol{x}, \boldsymbol{y}] = \mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}]\ \mathbf{Pr}[\boldsymbol{x}]$ means $\mathbf{Pr}[\boldsymbol{x} = x, \boldsymbol{y} = y] = \mathbf{Pr}[\boldsymbol{y} = y|\boldsymbol{x} = x]\ \mathbf{Pr}[\boldsymbol{x} = x], \forall x \in \mathcal{X}, y \in \mathcal{Y}$.

2.1 Problem Setting

Given a dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ of N data points, we aim to learn a neural network with model parameters $\boldsymbol{\theta}$ that predicted the output $y_n \in \mathcal{Y}$ based on the input $x_n \in \mathcal{X}$. (1) We first solve the **learning problem**: to find an *approximate posterior distribution* $Q(\boldsymbol{\theta}; \phi)$ over $\boldsymbol{\theta}$ with parameters ϕ such that $Q(\boldsymbol{\theta}; \phi) \approx \mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$. (2) We then solve the **prediction problem**: to compute the *predictive distribution* $\mathbf{Pr}[\boldsymbol{y}|x, \mathcal{D}]$ for arbitrary input $\boldsymbol{x} = x$ given $Q(\boldsymbol{\theta}; \phi)$. For notational simplicity, we will omit the conditioning on $\mathcal{D}$ and write $\mathbf{Pr}[\boldsymbol{y}|x, \mathcal{D}]$ as $\mathbf{Pr}[\boldsymbol{y}|x]$ in what follows.

In order to address the prediction and learning problems in BNNs, we analyze these models in their general form of *hierarchical Bayesian models* (shown in Figure 3b in Appendix B). Let $\boldsymbol{h}^{(l)}$, $\boldsymbol{\theta}^{(l)}$ and $\boldsymbol{h}^{(l+1)}$ denote the inputs, model parameters, and outputs of the l-th layer respectively. We assume that $\boldsymbol{\theta}^{(l)}$'s are *layer-wise independent*, i.e. $Q(\boldsymbol{\theta};\phi) = \prod_{l=0}^{L-1} Q(\boldsymbol{\theta}^{(l)};\phi^{(l)})$, and that the hidden variables $\boldsymbol{h}^{(l)}$ follow the *Markovian property*, i.e. $\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(:l)},\boldsymbol{\theta}^{(:l)}] = \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(l)},\boldsymbol{\theta}^{(l)}]$.

2.2 The Prediction Problem

Computing the predictive distribution $\mathbf{Pr}[\boldsymbol{y}|x,\mathcal{D}]$ with a BNN requires marginalizing over the random variable $\boldsymbol{\theta}$. The hierarchical structure of BNNs allows this marginalization to be performed in multiple steps sequentially. In Appendix B, we show that the predictive distribution of $\boldsymbol{h}^{(l+1)}$ given input x can be obtained from its preceding layer $\boldsymbol{h}^{(l)}$ by

$$\underbrace{\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|x]}_{P(\boldsymbol{h}^{(l+1)};\psi^{(l+1)})} = \int_{h^{(l)},\theta^{(l)}} \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|h^{(l)},\theta^{(l)}]\; Q(\theta^{(l)};\phi^{(l)})\; \underbrace{\mathbf{Pr}[h^{(l)}|x]}_{P(h^{(l)};\psi^{(l)})} dh^{(l)} d\theta^{(l)} \tag{1}$$

This iterative process to compute the predictive distributions sequentially, layer-by-layer is known as *probabilistic propagation* (Soudry et al., 2014; Hernández-Lobato & Adams, 2015; Ghosh et al., 2016). With this approach, we need to explicitly compute and store each intermediate result $\mathbf{Pr}[\boldsymbol{h}^{(l)}|x]$ in its parameterized form $P(\boldsymbol{h}^{(l)};\psi^{(l)})$ (the conditioning on x is hidden in $\psi^{(l)}$, i.e. $\psi^{(l)}$ is a function of x). Therefore, probabilistic propagation is a deterministic process that computes $\psi^{(l+1)}$ as a function of $\psi^{(l)}$ and $\phi^{(l)}$, which we denote as $\psi^{(l+1)} = g^{(l)}(\psi^{(l)},\phi^{(l)})$.

Challenge in Sampling-Free Probabilistic Propagation. If the hidden variables $\boldsymbol{h}^{(l)}$'s are continuous, Equation (1) generally can not be evaluated in closed form as it is difficult to find a family P of parameterized distributions for $\boldsymbol{h}^{(l)}$ such that $\boldsymbol{h}^{(l+1)}$ remains in P under the operations performed in a neural network layer. Therefore most existing methods consider approximations at each layer of probabilistic propagation. In Section 4, we will show that this issue can be (partly) addressed if we consider the $\boldsymbol{h}^{(l)}$'s to be discrete random variables, as in a BQN.

2.3 The Learning Problem

Objective Function. A standard approach to finding a good approximation $Q(\boldsymbol{\theta};\phi)$ is *variational inference*, which finds $\phi^\star$ such that the *KL-divergence* $\mathbf{KL}(Q(\boldsymbol{\theta};\phi)||\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}])$ from $Q(\boldsymbol{\theta};\phi)$ to $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$ is minimized. To minimize this KL-divergence, it is equivalent to maximize an objective function known as the *evidence lower bound (ELBO)*, denoted as $\mathcal{L}(\phi)$. In Appendix B, we show that the ELBO can be expressed as

$$\begin{gathered}\max_{\phi} \mathcal{L}(\phi) = -\mathbf{KL}(Q(\boldsymbol{\theta};\phi)||\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]) = \sum_{n=1}^{N} \mathcal{L}_n(\phi) + \mathcal{R}(\phi) \\ \text{where } \mathcal{L}_n(\phi) = \mathbb{E}_Q\left[\log \mathbf{Pr}[y_n|x_n,\boldsymbol{\theta}]\right] \text{ and } \mathcal{R}(\phi) = \mathbb{E}_Q\left[\log\left(\mathbf{Pr}[\boldsymbol{\theta}]\right)\right] + H(Q)\end{gathered} \tag{2}$$

Probabilistic Backpropagation. Optimization in neural networks heavily relies on the gradient-based methods, where the partial derivatives $\partial\mathcal{L}(\phi)/\partial\phi$ of the objective $\mathcal{L}(\phi)$ w.r.t. the parameters ϕ are obtained by *backpropagation*. Formally, if the output produced by a neural network is given by a (sub-)differentiable function $g(\phi)$, and the objective $\mathcal{L}(g(\phi))$ is an *explicit* function of $g(\phi)$ (and not just an explicit function of ϕ), then the partial derivatives can be computed by chain rule:

$$\frac{\partial\mathcal{L}(g(\phi))}{\partial\phi} = \frac{\partial\mathcal{L}(g(\phi))}{\partial g(\phi)} \cdot \frac{\partial g(\phi)}{\partial\phi}. \tag{3}$$

The learning problem can then be (approximately) solved by first-order methods, such as *stochastic gradient descent/ascent*. *Remarks:* (1) For classification, the function $g(\phi)$ returns the probabilities after the softmax function, not the categorical label; (2) An additional regularization term $\mathcal{R}(\phi)$ on the parameters will not cause difficulty in backpropagation, if $\partial\mathcal{R}(\phi)/\partial\phi$ is easily computed.

Challenge in Sampling-Free Probabilistic Backpropagation. Learning BNNs is not amenable to standard backpropagation because the ELBO objective function $\mathcal{L}(\phi)$ in (4b) is not an explicit (i.e.

implicit) function of the predictive distribution $g(\phi)$ in (4a):

$$g_n(\phi) = \mathbb{E}_Q\left[\mathbf{Pr}[y_n|x_n, \boldsymbol{\theta}]\right] = \int_\theta \mathbf{Pr}[y_n|x_n, \theta] Q(\theta; \phi) d\theta \tag{4a}$$

$$\mathcal{L}_n(\phi) = \mathbb{E}_Q\left[\log(\mathbf{Pr}[y_n|x_n, \boldsymbol{\theta}])\right] = \int_\theta \log\left(\mathbf{Pr}[y_n|x_n, \theta]\right) Q(\theta; \phi) d\theta \tag{4b}$$

Although $\mathcal{L}_n(\phi)$ is a function of ϕ, it is not an explicit function of $g_n(\phi)$. Consequently, the chain rule in Equation (3) on which backpropagation is based is not directly applicable.

3 Proposed Learning Method for Bayesian Neural Networks

3.1 Alternative Evidence Lower Bound

We make learning in BNNs amenable to backpropagation by developing a lower bound $\overline{\mathcal{L}}_n(\phi) \leq \mathcal{L}_n(\phi)$ such that partial derivatives $\partial\overline{\mathcal{L}}_n(\phi)/\partial\phi$ can be obtained by chain rule (i.e. $\overline{\mathcal{L}}_n(\phi)$ is an explicit function of output/intermediate results of the forward pass.) With $\overline{\mathcal{L}}_n(\phi)$ in hand, we can (approximately) find $\phi^\star$ by maximizing the alternative objective via gradient-based methods:

$$\phi^\star = \arg\max_\phi \overline{\mathcal{L}}(\phi) = \arg\max_\phi \left(\mathcal{R}(\phi) + \sum_{n=1}^N \overline{\mathcal{L}}_n(\phi)\right) \tag{5}$$

Theorem 3.1 (**Alternative Evidence Lower Bound**). *Define each term $\overline{\mathcal{L}}_n(\phi)$ in $\overline{\mathcal{L}}(\phi)$ as*

$$\overline{\mathcal{L}}_n(\phi) := \mathbb{E}_{\boldsymbol{h}^{(L-1)}\sim P;\, \boldsymbol{\theta}^{(L-1)}\sim Q}\left[\log\left(\mathbf{Pr}[y_n|\boldsymbol{h}^{(L-1)}, \boldsymbol{\theta}^{(L-1)}]\right)\right], \tag{6}$$

then $\overline{\mathcal{L}}_n(\phi)$ is a lower bound of $\mathcal{L}_n(\phi)$, i.e. $\overline{\mathcal{L}}_n(\phi) \leq \mathcal{L}_n(\phi)$, and therefore $\overline{\mathcal{L}}(\phi) \leq \mathcal{L}(\phi)$.

Remarks: The equality $\overline{\mathcal{L}}_n(\phi) = \mathcal{L}_n(\phi)$ holds if either of the following two conditions is met: (1) $\mathbf{Pr}[\boldsymbol{h}^{(L-1)}|x_n, \boldsymbol{\theta}^{(:L-2)}]$ follows a delta distribution, i.e. the hidden variable $\boldsymbol{h}^{(L-1)}$ is a deterministic given input x and all model parameters before the last layer $\boldsymbol{\theta}^{(:L-2)}$; (2) $\mathbf{Pr}[y_n|\boldsymbol{h}^{(L-1)}, \boldsymbol{\theta}^{(L-1)}]$ is constant over the support of $\mathbf{Pr}[\boldsymbol{h}^{(L-1)}|x_n, \boldsymbol{\theta}^{(:L-1)}]$.

We prove theorem 3.1 in Appendix C.1. Notice that if $\boldsymbol{\theta}^{(L-1)}$ is not random variable (typical for an output layer), the alternative objective $\overline{\mathcal{L}}_n(\phi)$ can be further simplified as:

$$\overline{\mathcal{L}}_n(\phi) = \int_{h^{(L-1)}} \log\left(\mathbf{Pr}[y_n|h^{(L-1)}; \phi^{(L-1)}]\right) P(h^{(L-1)}; \psi^{(L-1)}) dh^{(L-1)} \tag{7}$$

where we write $\mathbf{Pr}[\boldsymbol{h}^{(L-1)}|x]$ in its parameterized form $P(\boldsymbol{h}^{(L-1)}; \psi^{(L-1)})$. Now, the gradient $\partial\overline{\mathcal{L}}_n(\phi)/\partial\phi^{(L-1)}$ can be obtained by differentiating over Equation (7), while other gradients $\partial\overline{\mathcal{L}}_n(\phi)/\phi^{(:L-2)}$ further obtained by chain rule:

$$\frac{\partial\overline{\mathcal{L}}_n(\phi)}{\partial\phi^{(:L-2)}} = \frac{\partial\overline{\mathcal{L}}_n(\phi)}{\partial\psi^{(L-1)}} \cdot \frac{\partial\psi^{(L-1)}}{\partial\phi^{(:L-2)}} \tag{8}$$

which requires us to compute $\partial\overline{\mathcal{L}}_n(\phi)/\partial\psi^{(L-1)}$ and $\partial\psi^{(L-1)}/\partial\phi^{(:L-2)}$. While $\partial\overline{\mathcal{L}}_n(\phi)/\partial\psi^{(L-1)}$ can be derived from Equation (7), $\partial\psi^{(L-1)}/\partial\phi^{(:L-2)}$ can be obtained by backpropagating outputs of the $(L-2)^{\text{th}}$ layer obtained from probabilistic propagation in Equation (1). In other words: since $P(\boldsymbol{h}^{(L-1)}; \psi^{(L-1)})$ is an intermediate step of the forward pass, $\psi^{(L-1)}$ is a function of all parameters from previous layers $\phi^{(:L-2)}$, and if each step $\psi^{(l+1)} = g^{(l)}(\psi^{(l)}, \phi^{(l)})$ is differentiable w.r.t. $\psi^{(l)}$ and $\phi^{(l)}$, the partial derivatives $\partial\psi^{(L-1)}/\partial\phi^{(:L-2)}$ can be obtained by iterative chain rule.

3.2 Analytic Forms of $\overline{\mathcal{L}}_n(\phi)$ and $\partial\overline{\mathcal{L}}_n(\phi)/\partial\psi^{(L-1)}$

The analysis in Section 3.1 applies to BNNs with *arbitrary* distributions P on hidden variables $\boldsymbol{h}$, Q on model parameters $\boldsymbol{\theta}$, and *any* problem setting (e.g. classification or regression) depending on the form of $\mathbf{Pr}[y|\boldsymbol{h}^{(L-1)}, \boldsymbol{\theta}^{(L-1)}]$. In practice, sampling-free probabilistic backpropagation requires that the partial derivatives $\partial\overline{\mathcal{L}}_n(\phi)/\partial\psi^{(L-1)}$ and $\partial\overline{\mathcal{L}}_n(\phi)/\partial\phi^{(L-1)}$ can be analytically computed. This task is nontrivial and requires redesign of the output layer, i.e. the form of

$\mathbf{Pr}[y|\boldsymbol{h}^{(L-1)}, \boldsymbol{\theta}^{(L-1)}]$. In this part, we present two such designs, one for classification and another for regression. Since the gradients only involve one layer and one data point, we will omit the scripts of $\boldsymbol{h}^{(L-1)}$, $\psi^{(L-1)}$, $\phi^{(L-1)}$, x_n, y_n, and denote them as $\boldsymbol{h}$, ψ, ϕ, x, y in the following theorems.

Theorem 3.2 (**Analytic Form of $\overline{\mathcal{L}}_n(\phi)$ for Classification**). *Let $\boldsymbol{h} \in \mathbb{R}^K$ (with K the number of classes) be the pre-activations of a softmax layer (a.k.a. logits), and $\phi = s \in \mathbb{R}^+$ be a scaling factor that adjusts its scale such that $\mathbf{Pr}[\boldsymbol{y} = c|\boldsymbol{h}, s] = \exp(\boldsymbol{h}_c/s)/\sum_{k=1}^K \exp(\boldsymbol{h}_k/s)$. Suppose the logits $\{\boldsymbol{h}_k\}_{k=1}^K$ are pairwise independent (which holds under mean-field approximation) and $\boldsymbol{h}_k$ follows a Gaussian distribution $\boldsymbol{h}_k \sim \mathcal{N}(\mu_k, \nu_k)$ (therefore $\psi = \{\mu_k, \nu_k\}_{k=1}^K$) and s is a deterministic parameter. Then $\overline{\mathcal{L}}_n(\phi)$ takes an analytic form:*

$$\overline{\mathcal{L}}_n(\phi) = \frac{\mu_c}{s} - \log\left(\sum_{k=1}^K \exp\left(\frac{\mu_k}{s} + \frac{\nu_k}{2s^2}\right)\right) \tag{9}$$

whose derivatives are $\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial \mu_k} = -\frac{1}{s}\left(\frac{\exp\left(\mu_k/s+\nu_k/2s^2\right)}{\sum_{k=1}^K \exp(\mu_k/s+\nu_k/2s^2)} - \mathbf{1}[k=c]\right)$ where $\mathbf{1}[k=c]$ is an indicator function, and $\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial \nu_k} = -\frac{1}{2s^2}\left(\frac{\exp\left(\mu_k/s+\nu_k/2s^2\right)}{\sum_{k=1}^K \exp(\mu_k/s+\nu_k/2s^2)}\right)$.

Theorem 3.3 (**Analytic Form of $\overline{\mathcal{L}}_n(\phi)$ for Regression**). *Let $\boldsymbol{h} \in \mathbb{R}^I$ be the output of last hidden layer (with I the number of hidden units), and $\phi = (w, s) \in \mathbb{R}^I \times \mathbb{R}^+$ be the parameters that define the predictive distribution over output $\boldsymbol{y}$ as $\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{h}; w, s] = 1/\sqrt{2\pi s}\exp\left(-(\boldsymbol{y} - w^\top \boldsymbol{h})^2/2s\right)$. Suppose the hidden units $\{\boldsymbol{h}_k\}_{k=1}^K$ are pairwise independent (which holds under mean-field approximation), and each $\boldsymbol{h}_i$ has mean μ_i and variance ν_i, then $\overline{\mathcal{L}}_n(\phi)$ takes an analytic form:*

$$\overline{\mathcal{L}}_n(\phi) = -\frac{(y - w^\top \mu)^2 + (w^{\circ 2})^\top \nu}{2s} - \frac{\log(2\pi s)}{2} \tag{10}$$

where $(w^{\circ 2})_i = w_i^2$. The derivatives are $\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial \mu} = -\frac{(y - w^\top \mu)w}{s}$ and $\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial \nu} = -\frac{w^{\circ 2}}{2s}$.

The proofs for both theorems are deferred to Appendix C.2 and C.3.

4 Bayesian Quantized Networks (BQNs)

In Section 3 we provided a general solution to the problems of prediction and learning in BNNs, but this solution relies on the ability to perform probabilistic propagation efficiently. To address this, we introduce Bayesian quantized networks (BQNs) — Bayesian neural networks where both hidden units $\boldsymbol{h}^{(l)}$'s and model parameters $\boldsymbol{\theta}^{(l)}$'s take integer values — along with a set of novel algorithms for efficient sampling-free probabilistic propagation in BQNs.

4.1 Probabilistic Propagation as Tensor Contractions

For simplicity of exposition, we assume all activations and model parameters take values in the same discrete set $\mathbb{Q}$, and denote the *degree of quantization* as $D = |\mathbb{Q}|$, e.g. $\mathbb{Q} = \{-1, 1\}$, $D = 2$.

Lemma 4.1 (**Probabilistic Propagation in BQNs**). *After quantization, the iterative step of probabilistic propagation in Equation* (1) *is computed with a finite sum instead of an integral:*

$$P(\boldsymbol{h}^{(l+1)}; \psi^{(l+1)}) = \sum\nolimits_{h^{(l)}, \theta^{(l)}} \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|h^{(l)}, \theta^{(l)}]\, Q(\theta^{(l)}; \phi^{(l)})\, P(h^{(l)}; \psi^{(l)}) \tag{11}$$

and a categorically distributed $\boldsymbol{h}^{(l)}$ results in $\boldsymbol{h}^{(l+1)}$ being categorical as well. The equation holds without any assumption on the operation $\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(l)}, \theta^{(l)}]$ performed in the neural network.

All probability distributions in Equation (11) are high-order tensors. Suppose there are I input units, J output units, and K model parameters at the l-th layer, then $\boldsymbol{h}^{(l)} \in \mathbb{Q}^I$, $\boldsymbol{\theta}^{(l)} \in \mathbb{Q}^K$, and $\boldsymbol{h}^{(l+1)} \in \mathbb{Q}^J$, and their distributions are characterized by $P(\boldsymbol{h}^{(l)}; \psi^{(l)}) \in \mathbb{R}^{D^I}$, $Q(\boldsymbol{\theta}^{(l)}; \phi^{(l)}) \in \mathbb{R}^{D^K}$, $P(\boldsymbol{h}^{(l+1)}; \psi^{(l+1)}) \in \mathbb{R}^{D^J}$, and $\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(l)}, \boldsymbol{\theta}^{(l)}] \in \mathbb{R}^{D^J \times D^I \times D^K}$ respectively. Therefore, each step in probabilistic propagation is a *tensor contraction* of three tensors, which establishes the duality between BQNs and hierarchical tensor networks (Robeva & Seigal, 2017).

Since tensor contractions are differentiable w.r.t. all inputs, BQNs thus circumvent the difficulties involved in training QNNs (Courbariaux et al., 2015; Rastegari et al., 2016), whose outputs are not differentiable w.r.t. their parameters. This result is not surprising: if we consider learning in QNNs as an *integer programming (IP)* problem, solving its Bayesian counterpart is equivalent to the standard approach to relaxing an IP problem into a *continuous optimization* problem.

Complexity of Exact Propagation. The computational complexity to evaluate Equation (11) is exponential in the number of random variables $O(D^{IJK})$, which is intractable for quantized neural network of any reasonable size. We thus turn to approximations.

4.2 Approximate Propagation via Rank-1 Tensor CP Decomposition

A principled way to solve the "curse of dimensionality" in probabilistic propagation in BQNs is *tensor decomposition*, where an intractable high-order probability tensor is factored into tractable lower-order factors (Cichocki et al., 2016). In this paper, we consider the simplest *rank-1 CP decomposition* (Kolda & Bader, 2009) (equivalent to the *mean-field approximation* (Wainwright et al., 2008)), where the joint distributions of P and Q are fully factorized into products of their marginal distributions. With rank-1 CP decomposition on $P(h^{(l)}; \psi^{(l)}), \forall l \in [L]$, the tensor contraction in (11) reduces to a standard *Tucker contraction* (Kolda & Bader, 2009)

$$P(\boldsymbol{h}_j^{(l+1)}; \psi_j^{(l+1)}) \approx \sum\nolimits_{h^{(l)}, \theta^{(l)}} \mathbf{Pr}[\boldsymbol{h}_j^{(l+1)} | \theta^{(l)}, h^{(l)}] \prod\nolimits_k Q(\theta_k^{(l)}; \phi_k^{(l)}) \prod\nolimits_i P(h_i^{(l)}; \psi_i^{(l)}) \quad (12)$$

where each term of $\psi_i^{(l)}$, $\phi_k^{(l)}$ parameterizes a single categorical variable. In our implementation, we choose the *natural reduced form*, i.e. $Q(\theta_k^{(l)} = \mathbb{Q}(d)) = \exp(\psi_k^{(l)}(d)) / \sum_{q=1}^{D} \exp(\phi_k^{(l)}(q))$.

Complexity of Approximate Propagation and Fan-in Number E. In a practical model, each output unit $\boldsymbol{h}_j^{(l+1)}$ only depends on a subset of input units $\{\boldsymbol{h}_i^{(l)}\}$ and parameters $\{\boldsymbol{\theta}_k^{(h)}\}$. Denote the numbers of responsible input units and parameters as $\mathcal{I}(j)$ and $\mathcal{M}(j)$, and define the *fan-in number* E as $\max_j(\mathcal{I}(j) + \mathcal{M}(j))$. Then the computational complexity of approximate propagation is reduced from $O(D^{IJK})$ to $O(JD^E)$, which is linear in the number of output units J.

4.3 Fast Algorithms for Tensor Contractions

Small Fan-in: Direct Tensor Contraction. If E is small, tensor contraction in Equation (12) is immediately applicable. Representative cases of small E are *shortcut layer (a.k.a. skip-connection)* and *depth-wise layers*. (1) In a shortcut layer, $\boldsymbol{h}^{(l+1)}$ is an addition of two previous layers $\boldsymbol{h}^{(l)}$ and $\boldsymbol{h}^{(m)}$, and the distribution of $\boldsymbol{h}^{(l+1)}$ can be directly computed as $P(\boldsymbol{h}_i^{(l+1)}; \psi_i^{(l+1)}) = \sum_{h_i^{(l)}, h_i^{(m)}} \delta[\boldsymbol{h}_i^{(l+1)} = h_i^{(l)} + h_i^{(m)}] \; P(h_i^{(l)}; \psi_i^{(l)}) \; P(h_i^{(m)}; \psi_i^{(m)})$.

(2) In a depth-wise layer, each $\boldsymbol{h}_i^{(l+1)}$ is a transformation (parameterized by $\boldsymbol{\theta}_i^{(l)}$) of its corresponding $\boldsymbol{h}_i^{(l)}$: i.e. $P(\boldsymbol{h}_i^{(l+1)}; \psi_i^{(l+1)}) = \sum_{h_i^{(l)}, \theta_i^{(l)}} \mathbf{Pr}[\boldsymbol{h}_i^{(l+1)} | h_i^{(l)}, \theta_i^{(l)}] \; Q(\theta_i^{(l)}; \phi_i^{(m)}) \; P(h_i^{(l)}; \psi_i^{(l)})$. Depth-wise layers include *dropout layers* (where $\boldsymbol{\theta}^{(l)}$ are dropout rates), *nonlinear layers* (where $\boldsymbol{\theta}^{(l)}$ are threshold values) or *element-wise product layers* (where $\boldsymbol{\theta}^{(l)}$ are the weights). For both shortcut and depth-wise layers, the time complexity is $O(JD^2)$ since $E <= 2$.

Medium Fan-in: Discrete Fourier Transform. Summation of discrete random variables is equivalent to convolution of their probability mass function, and therefore can be efficiently evaluated via *fast Fourier transform*. Let $\boldsymbol{u}_i$ take values in $\{b_i, b_i + 1, \ldots, B_i\}$ between integers b_i and B_i. Then the summation $\boldsymbol{v} = \sum_{i=1}^{E} \boldsymbol{u}_i$ takes values between b and B, where $b = \sum_{i=1}^{E} b_i$ and $B = \sum_{i=1}^{E} B_i$.

Theorem 4.2 (Fast summation via discrete Fourier transform). *Let $C^{\boldsymbol{v}}$, $C^{\boldsymbol{u}_i}$ be the discrete Fourier transforms of $P^{\boldsymbol{v}}$, $P^{\boldsymbol{u}_i}$ respectively, i.e. $C^{\boldsymbol{v}}(f) = \sum_{v=b}^{B} P^{\boldsymbol{v}}(v) \exp(-\mathrm{j}2\pi(v-b)f/(B-b+1))$ and $C^{\boldsymbol{u}_i}(f) = \sum_{u_i=b_i}^{B_i} P^{\boldsymbol{u}_i}(u_i) \exp(-\mathrm{j}2\pi(u_i-b_i)f/(B_i-b_i+1))$. Then $C^{\boldsymbol{v}}(f)$ is the element-wise product of all Fourier transforms $C^{\boldsymbol{u}_i}(f)$, i.e. $C^{\boldsymbol{v}}(f) = \prod_{i=1}^{E} C^{\boldsymbol{u}_i}(f), \forall f$.*

We prove the Lemma 4.2 in Appendix D.2. By with the *fast Fourier transform*, the probability mass function of $\boldsymbol{v}$ can be computed as $P^{\boldsymbol{v}} = \mathrm{IFFT}(\prod_{i=1}^{E} \mathrm{FFT}(P^{\boldsymbol{u}_i}))$, which takes $O(E^2 D \log(ED))$ time with fan-in number E instead of $O(D^E)$ with direct tensor contraction.

Large Fan-in: Lyapunov Central Limit Theorem. In a typical linear layer, the fan-in E is large, and a super-quadratic algorithm using fast Fourier transform is still computational expensive. Therefore, we derive a faster algorithm in Appendix D based on the *Lyapunov central limit theorem*.

Theorem 4.3 (**Fast summation via Lyapunov Central Limit Theorem**). *Let $\boldsymbol{v} = \sigma(\tilde{\boldsymbol{v}}) = \sigma(\sum_{i\in\mathcal{I}} \boldsymbol{\theta}_i \boldsymbol{u}_i)$ is activations of a hidden layer computed via a nonlinearity σ. Suppose both inputs $\{\boldsymbol{\theta}_i\}_{i\in\mathcal{I}}$ and parameters $\{\boldsymbol{u}_i\}_{i\in\mathcal{I}}$ have bounded variance, then for sufficiently large $|\mathcal{I}|$ the distribution of $\tilde{\boldsymbol{v}}$ converges to a Gaussian distribution $\mathcal{N}(\tilde{\mu}, \tilde{\nu})$, with mean $\tilde{\mu} = \sum_{i\in\mathcal{I}} \mathbb{E}[\boldsymbol{\theta}_i]\mathbb{E}[\boldsymbol{u}_i]$ and variance $\tilde{\nu} = \sum_{i\in\mathcal{I}} (\mathbb{E}[\boldsymbol{\theta}_i]^2\mathbb{V}[\boldsymbol{u}_i] + \mathbb{E}[\boldsymbol{h}_i^{(l)}]^2\mathbb{V}[\boldsymbol{\theta}_i] + \mathbb{V}[\boldsymbol{u}_i]\mathbb{V}[\boldsymbol{\theta}_i])$. And if the nonlinear transform σ is a sign function, the activation $\boldsymbol{v}$ follows a Bernoulli distribution $P(\boldsymbol{v} = 1) = \Phi(\tilde{\mu}/\sqrt{\tilde{\nu}})$, where Φ is the culminative probability function of a standard Gaussian distribution $\mathcal{N}(0, 1)$.*

Remarks: With theorem 4.3, performing one addition operation with fan-in E takes only $O(ED)$. While CLT is always faster than the DFT of Theorem 4.2,the Gaussian approximation does not hold if E is not sufficiently large. Therefore, depending on computational resources, one can adopt CLT for linear layers with sufficiently large E, such as *fully connected layer* and *convolutional layers*, and DFT for those with smaller E, such as *average pooling layer*.

5 EXPERIMENTS

In this section, we demonstrate the effectiveness of BQNs on the MNIST, Fashion-MNIST and KMNIST image classification datasets, each of which contains $50,000$ training and $10,000$ test samples of 28×28 gray-scale images. In training, each image is augmented by a random shift within 2 pixels, and no augmentation is used in testing. In the experiments, we consider a class of quantized neural networks, with both *binary* weights and activations (i.e. $\mathbb{Q} = \{-1, 1\}$) with sign activations $\sigma(\cdot) = \text{sign}(\cdot)$. For BQNs, the distribution parameters ϕ were initialized by Xavier's uniform initializer, and all models are trained by ADAM optimizer (Kingma & Ba, 2014) for 100 epochs with batch size 100 and initial learning rate 10^{-2}, which decays by 0.98 per epoch.

Network Architectures and Training Objective of BQNs. We evaluate our BQNs with both *multi-layer perceptron* (MLP) and *convolutional neural network* (CNN) models. For MLP, we use a 3-layers network with 512 units in the first layer and 256 units in the second; and for CNN, we use a 4-layers network with two 5×5 convolutional layers with 64 channels followed by 2×2 average pooling, and two fully-connected layers with 1024 hidden units.

To allow for customized level of uncertainty in the learned Bayesian models, we introduce a regularization coefficient λ in the alternative ELBO proposed in Equation (5) (i.e. a lower bound of the likelihood), and train the BQNs by maximizing the following objective:

$$\overline{\mathcal{L}}(\phi) = \sum_{n=1}^{N} \overline{\mathcal{L}}_n(\phi) + \lambda \mathcal{R}(\phi) = \lambda \left(1/\lambda \sum_{n=1}^{N} \overline{\mathcal{L}}_n(\phi) + \mathcal{R}(\phi) \right) \tag{13}$$

where λ controls the uncertainty level, i.e. the importance weight of the prior over the training set.

Baseline. We compare our BQN against the baseline – *Bootstrap ensemble of quantized neural networks* (E-QNN). Each member in the ensemble is trained with in a non-Bayesian way (Courbariaux et al., 2016), and jointly make the prediction by averaging over the logits from all members.

To exhibit the effectiveness of our BQN, we compare against *continuous-valued Bayesian neural network* (abbreviated as BNN), with Gaussian distributed parameters. The model is with *stochastic gradient variational Bayes* (SGVB) augmented by *local re-parameterization trick* (Shridhar et al., 2019). Since the BNN allows for continuous parameters (different from the BQN which has quantized parameters), the predictive error is expected to be lower than BQN.

Evaluation of BQNs. While *0-1 test error* is a popular metric to measure the predictive performance, it is too coarse a metric to assess the uncertainty in decision making (for example it does not account for how badly the wrong predictions are). Therefore, we will mainly use the *negative log-likelihood* (NLL) to measure the predictive performance in the experiments.

Once a BQN is trained (i.e. an approximate posterior $Q(\boldsymbol{\theta})$ is learned), we consider three modes to evaluate the behavior of the model: **(1)** *analytic inference (AI)*, **(2)** *Monte Carlo (MC) sampling* and **(3)** *Maximum a Posterior (MAP) estimation*:

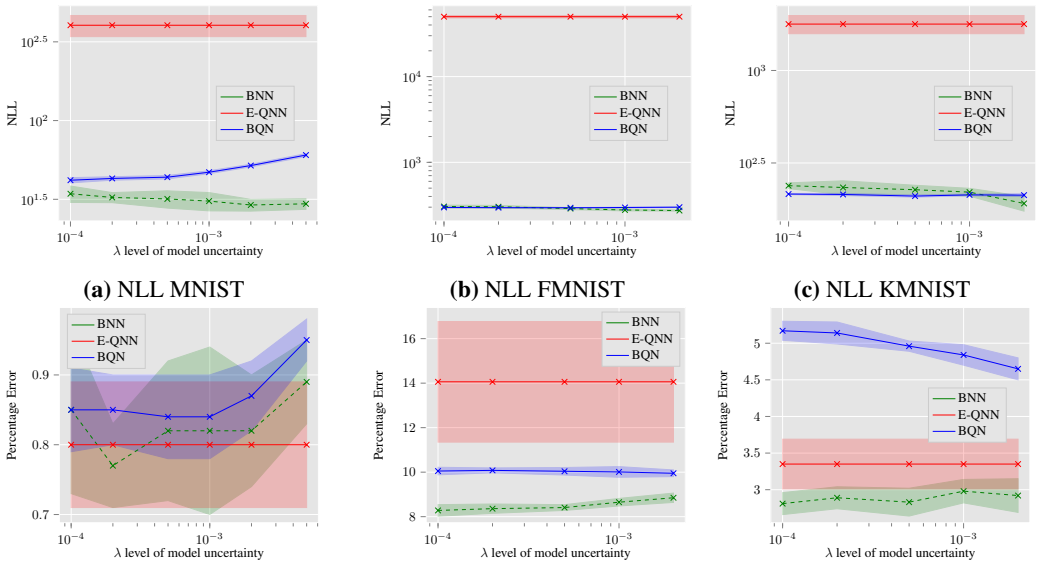
1. In analytic inference (AI, i.e. our proposed method), we analytically integrate over $Q(\boldsymbol{\theta})$ to obtain the predictive distribution as in the training phase. Notice that the exact NLL is not accessible with probabilistic propagation (which is why we propose an alternative ELBO in Equation (5)), we will report *an upper bound* of the NLL in this mode.
2. In MC sampling, S sets of model parameters are drawn independently from the posterior posterior $\theta_s \sim Q(\boldsymbol{\theta}), \forall s \in [S]$, and the forward propagation is performed as in (non-Bayesian) quantized neural network for each set θ_s, followed by an average over the model outputs. The difference between analytic inference and MC sampling will be used to evaluate **(a)** the effect of mean-field approximation and **(b)** the tightness of the our proposed alternative ELBO.
3. MAP estimation is similar to MC sampling, except that only one set of model parameters $\theta^\star$ is obtained $\theta^\star = \arg\max_\theta Q(\theta)$. We will exhibit our model's ability to compress a Bayesian neural network by comparing MAP estimation of our BQN with non-Bayesian QNN.

5.1 Analysis of Results

Methods	MNIST		KMNIST		Fashion-MNIST	
	NLL(10^{-3})	% Err.	NLL(10^{-3})	% Err.	NLL(10^{-3})	% Err.
Ensemble-QNN on MLP	546.60±157.90	3.30 ±0.65	2385.60±432.30	17.88±1.86	2529.40±276.70	13.02±0.81
Our BQN on MLP	**130.00±3.50**	**2.49±0.08**	**457.70±13.80**	**13.41±0.12**	**417.30±8.10**	**9.99±0.20**
Ensemble-QNN on CNN	425.3±61.80	0.85±0.13	3755.70±465.10	11.49±1.16	1610.70±158.40	**3.02±0.37**
Our BQN on CNN	**41.80±1.60**	**0.85±0.06**	**295.50±1.40**	**9.95±0.15**	**209.50±2.80**	4.65±0.15

Table 1: Comparison of performance of BQNs against the baseline E-QNN. Each E-QNN is an ensemble of 10 networks, which are trained individually and but make predictions jointly. We report both NLL (which accounts for prediction uncertainty) and 0-1 test error (which doesn't account for prediction uncertainty). All the numbers are averages over 10 runs with different seeds, the standard deviation are exhibited following the ± sign.

Expressive Power and Uncertainty Calibration in BQNs. We report the performance via all evaluations of our BQN models against the Ensemble-QNN in Table 1 and Figure 1. **(1)** Compared to E-QNNs, our BQNs have significantly lower NLL and smaller predictive error (except for Fashion-MNIST with architecture CNN). **(2)** As we can observe in Figure 1, BQNs impressively achieve comparable NLL to continuous-valued BNN, with slightly higher test error. As our model parameters only take values $\{-1, 1\}$, small degradation in predictive accuracy is expected.

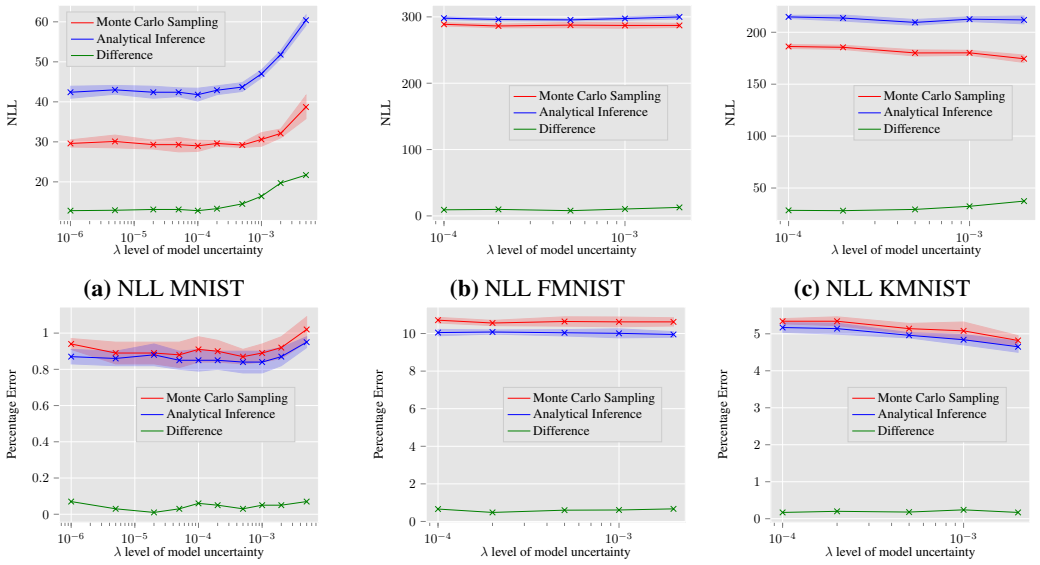


(a) NLL MNIST **(b)** NLL FMNIST **(c)** NLL KMNIST

(d) Error MNIST **(e)** Error FMNIST **(f)** Error KMNIST

Figure 1: Comparison of the predictive performance of our BQNs against the E-QNN as well as the non-quantized BNN trained by SGVB on a CNN. Negative log-likelihood (NLL) which accounts for uncertainty and 0-1 test error which doesn't account for uncertainty are displayed.

Evaluations of Mean-field Approximation and Tightness of the Alternative ELBO. If analytic inference (by probabilistic propagation) were computed exactly, the evaluation metrics would have been equal to the ones with MC sampling (with infinite samples). Therefore we can evaluate the approximations in probabilistic propagation, namely *mean-field approximation* in Equation (12) and *relaxation of the original ELBO* in Equation (5), by measuring the gap between analytic inference and MC sampling. As shown in Figure 2, such gaps are small for all scenarios, which justifies the approximations we use in BQNs.



(a) NLL MNIST **(b)** NLL FMNIST **(c)** NLL KMNIST

(d) Error MNIST **(e)** Error FMNIST **(f)** Error KMNIST

Figure 2: Illustration of mean-field approximation and tightness of alternative ELBO on a CNN. The performance gap between our analytical inference and the Monte Carlo Sampling is displayed.

To further decouple these two factors of *mean-field approximation* and *relaxation of the original ELBO*, we vary the regularization coefficient λ in the learning objective (13). **(1)** For $\lambda = 0$ (where the prior term is removed), the models are forced to become deterministic during training. Since the deterministic models do not have mean-field approximation in the forward pass, the gap between analytic inference and MC-sampling reflects the tightness of our alternative ELBO. **(2)** As λ increases, the gaps increases slightly as well, which shows that the mean-field approximation becomes slightly less accurate with higher learned uncertainty in the model.

Methods	MNIST		KMNIST		Fashion-MNIST	
	NLL(10^{-3})	% Err.	NLL(10^{-3})	% Err.	NLL(10^{-3})	% Err.
QNN on MLP	522.4±42.2	4.14±0.25	2019.1±281.2	19.56±1.97	2427.1±193.5	15.67±1.19
MAP of BQN on MLP	**137.60±4.40**	**3.69±0.09**	**464.60±12.80**	**14.79±0.21**	**461.30±13.40**	**12.89±0.17**
QNN on CNN	497.4±139.5	1.08±0.2	4734.5±1697.2	14.2±2.29	1878.3±223.8	**3.88±0.33**
MAP of BQN on CNN	**30.3±1.6**	**0.92±0.07**	**293.6±4.4**	**10.82±0.37**	**179.1±4.4**	5.00±0.11

Table 2: Deterministic model compression through direct training of QNN (Courbariaux et al., 2016) v.s. MAP estimation in our proposed BQN. All the numbers are averages over 10 runs with different seeds, the standard deviation are exhibited following the $\pm$ sign.

Compression of Neural Networks via BQNs. One advantage of BQNs over continuous-valued BNNs is that deterministic QNNs can be obtained for free, since a BQN can be interpreted as an ensemble of infinite QNNs (each of which is a realization of posterior distribution). **(1)** One simple approach is to set the model parameters to their *MAP estimates*, which compresses a given BQN to $1/64$ of its original size (and has the same number of bits as a single QNN). **(2)** *MC sampling* can be interpreted as another approach to compress a BQN, which reduces the original size to its $S/64$ (with the same number of bits as an ensemble of S QNNs). In Tables 2 and 3, we compare the models by both approaches to their counterparts (a single QNN for MAP, and E-QNN for MC sampling) trained from scratch as in Courbariaux et al. (2016). For both approaches, our compressed models outperform their counterparts (in NLL) . We attribute this to two factors: **(a)** QNNs are not trained

Methods	MNIST		KMNIST		Fashion-MNIST	
	NLL(10^{-3})	% Err.	NLL(10^{-3})	% Err.	NLL(10^{-3})	% Err.
E-QNN on MLP	546.60±157.90	3.30 ±0.65	2385.60±432.30	17.88±1.86	2529.40±276.70	13.02±0.81
MC of BQN on MLP	**108.9±2.6**	**2.73±0.09**	**429.50±11.60**	**13.83±0.12**	**385.30±5.10**	**10.81±0.44**
E-QNN on CNN	425.3±61.80	**0.85±0.13**	3755.70±465.10	11.49±1.16	1610.70±158.40	**3.02±0.37**
MC of BQN on CNN	**29.2±0.6**	0.87±0.04	**286.3±2.7**	**10.56±0.14**	**174.5±3.6**	4.82±0.13

Table 3: Bayesian Model compression through direct training of Ensemble-QNN vs a Monte-Carlo sampling on our proposed BQN. Each ensemble consists of 5 quantized neural networks, and for fair comparison we use 5 samples for Monte-Carlo evaluation. All the numbers are averages over 10 runs with different seeds, the standard deviation are exhibited following the ± sign.

in a Bayesian way, therefore the uncertainty is not well calibrated; and **(b)** Non-differentiable QNNs are unstable to train. Our compression approaches via BQNs simultaneously solve both problems.

6 Conclusion

We present a sampling-free, backpropagation-compatible, variational-inference-based approach for learning Bayesian quantized neural networks (BQNs). We develop a suite of algorithms for efficient inference in BQNs such that our approach scales to large problems. We evaluate our BQNs by Monte-Carlo sampling, which proves that our approach is able to learn a proper posterior distribution on QNNs. Furthermore, we show that our approach can also be used to learn (ensemble) QNNs by taking maximum a posterior (or sampling from) the posterior distribution.

References

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

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

Andrzej Cichocki, Namgil Lee, Ivan V Oseledets, Anh Huy Phan, Qibin Zhao, and D Mandic. Low-rank tensor networks for dimensionality reduction and large-scale optimization problems: Perspectives and challenges part 1. *arXiv preprint arXiv:1609.00893*, 2016.

Andrzej Cichocki, Anh-Huy Phan, Qibin Zhao, Namgil Lee, Ivan Oseledets, Masashi Sugiyama, Danilo P Mandic, et al. Tensor networks for dimensionality reduction and large-scale optimization: Part 2 applications and future perspectives. *Foundations and Trends® in Machine Learning*, 9(6):431–673, 2017.

Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. In *Advances in neural information processing systems*, pp. 3123–3131, 2015.

Matthieu Courbariaux, Itay Hubara, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1. *arXiv preprint arXiv:1602.02830*, 2016.

Steve K Esser, Rathinakumar Appuswamy, Paul Merolla, John V Arthur, and Dharmendra S Modha. Backpropagation for energy-efficient neuromorphic computing. In *Advances in Neural Information Processing Systems*, pp. 1117–1125, 2015.

Yarin Gal. *Uncertainty in Deep Learning*. PhD thesis, University of Cambridge, 2016.

Soumya Ghosh, Francesco Maria Delle Fave, and Jonathan S Yedidia. Assumed density filtering methods for learning bayesian neural networks. In *AAAI*, pp. 1589–1595, 2016.

Lars Grasedyck, Daniel Kressner, and Christine Tobler. A literature survey of low-rank tensor approximation techniques. *GAMM-Mitteilungen*, 36(1):53–78, 2013.

Alex Graves. Practical variational inference for neural networks. In *Advances in neural information processing systems*, pp. 2348–2356, 2011.

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.

José Miguel Hernández-Lobato and Ryan Adams. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, pp. 1861–1869, 2015.

Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pp. 5–13. ACM, 1993.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *Journal of Machine Learning Research*, 18:187–1, 2017.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Mohammad Khan. *Variational learning for latent Gaussian model of discrete data*. PhD thesis, University of British Columbia, 2012.

Minje Kim and Paris Smaragdis. Bitwise neural networks. *arXiv preprint arXiv:1601.06071*, 2016.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Tamara G Kolda and Brett W Bader. Tensor decompositions and applications. *SIAM review*, 51(3): 455–500, 2009.

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.

Thomas Peter Minka. *A family of algorithms for approximate Bayesian inference*. PhD thesis, Massachusetts Institute of Technology, 2001.

Elizabeth Newman, Lior Horesh, Haim Avron, and Misha Kilmer. Stable tensor neural networks for rapid deep learning, 2018.

Román Orús. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of Physics*, 349:117–158, 2014.

Jorn WT Peters and Max Welling. Probabilistic binary neural networks. *arXiv preprint arXiv:1809.03368*, 2018.

Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European Conference on Computer Vision*, pp. 525–542. Springer, 2016.

Elina Robeva and Anna Seigal. Duality of graphical models and tensor networks. *Information and Inference: A Journal of the IMA*, 2017.

Oran Shayer, Dan Levi, and Ethan Fetaya. Learning discrete weights using the local reparameterization trick. *arXiv preprint arXiv:1710.07739*, 2017.

Kumar Shridhar, Felix Laumann, and Marcus Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference. *arXiv preprint arXiv:1901.02731*, 2019.

Daniel Soudry, Itay Hubara, and Ron Meir. Expectation backpropagation: Parameter-free training of multilayer neural networks with continuous or discrete weights. In *Advances in Neural Information Processing Systems*, pp. 963–971, 2014.

Jiahao Su, Jingling Li, Bobby Bhattacharjee, and Furong Huang. Tensorized spectrum preserving compression for neural networks. *arXiv preprint arXiv:1805.10352*, 2018.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Hao Wang and Dit-Yan Yeung. Towards bayesian deep learning: A survey. *arXiv preprint arXiv:1604.01662*, 2016.

Anqi Wu, Sebastian Nowozin, Edward Meeds, Richard E Turner, José Miguel Hernández-Lobato, and Alexander L Gaunt. Fixing variational bayes: Deterministic variational inference for bayesian neural networks. *arXiv preprint arXiv:1810.03958*, 2018.

Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.

Chenzhuo Zhu, Song Han, Huizi Mao, and William J Dally. Trained ternary quantization. *arXiv preprint arXiv:1612.01064*, 2016.

Appendix: Sampling-Free Learning of Bayesian Quantized Neural Networks

A Related Work

Bayesian Neural Networks Various types of approaches have been proposed to learn the posterior distribution over model parameters in neural networks.

(1) Sampling-free Assumed Density Filtering (ADF), including EBP Soudry et al. (2014) and PBP (Hernández-Lobato & Adams, 2015), is an online algorithm which (approximately) updates the posterior distribution by Bayes' rule for each observation. If the model parameters $\boldsymbol{\theta}$ are Gaussian distributed, Minka (2001) shows that the Bayes' rule can be computed in analytic form based on $\partial \log(g_n(\phi))/\partial\phi$, and EBP Soudry et al. (2014) derives a similar rule for Bernoulli parameters in binary classification. Notice that ADF is compatible to backpropagation:

$$\frac{\partial \log(g_n(\phi))}{\partial \phi} = \frac{1}{g_n(\phi)} \cdot \frac{\partial g_n(\phi)}{\partial \phi} \tag{14}$$

assuming $g_n(\phi)$ can be (approximately) computed by probabilistic propagation as in Section 2. However, this approach has two major limitations: the Bayes' rule needed to be derived case by case, and analytic rule for most common cases are not known yet. Second, it is not compatible to modern optimization methods (such as SGD or ADAM) as the optimization is solved analytically for each data point, therefore difficult to cope with large-scale statistical models.

(2) Sampling-based Variational inference (SVI), formulates an optimization problem and solves it approximately via *stochastic gradient descent (SGD)* (Hinton & Van Camp, 1993; Graves, 2011; Blundell et al., 2015). The most popular method among all is, *Stochastic Gradient Variational Bayes (SGVB)*, which approximates $\mathcal{L}_n(\phi)$ by the average of multiple samples (Blundell et al., 2015). Before each step of learning or prediction, a number of independent samples of model parameters $\{\theta_s\}_{s=1}^S$ are drawn according to the current estimate of Q, i.e. $\theta_s \overset{i.i.d}{\sim} Q$. Then both the predictive function $g_n(\phi)$ and the loss $\mathcal{L}_n(\phi)$ can be approximated by

$$g_n(\phi) \approx \frac{1}{S}\sum_{s=1}^{S} \mathbf{Pr}[y_n|x_n,\theta_s] = \frac{1}{S}\sum_{s=1}^{S} f_n(\theta_s) \tag{15a}$$

$$\mathcal{L}_n(\phi) \approx \frac{1}{S}\sum_{s=1}^{S} \log\left(\mathbf{Pr}[y_n|x_n,\theta_s]\right) = \frac{1}{S}\sum_{s=1}^{S} \log\left(f_n(\theta_s)\right) \tag{15b}$$

where $f_n(\theta) = \mathbf{Pr}[y_n|x_n,\theta]$ denotes the predictive function given a specific realization θ of the model parameters. The gradients of $\mathcal{L}_n(\phi)$ can now be approximated as

$$\frac{\partial \mathcal{L}_n(\phi)}{\partial \phi} \approx \frac{1}{S}\sum_{s=1}^{S} \frac{\partial \mathcal{L}_n(\phi)}{\partial f_n(\theta_s)} \cdot \frac{\partial f_n(\theta_s)}{\partial \theta_s} \cdot \frac{\partial \theta_s}{\partial \phi} \tag{16}$$

This approach has multiple drawbacks: (1) Repeated sampling suffers from high variance, besides being computationally expensive in both learning and prediction phases; (2) While $g_n(\phi)$ is differentiable w.r.t. ϕ, $f_n(\theta)$ may not be differentiable w.r.t. θ. One such example is quantized neural networks, whose backpropagation is approximated by *straight through estimator* (Bengio et al., 2013). (3) The partial derivatives $\partial\theta_s/\partial\phi$ are difficult to compute with complicated *reparameterization tricks* (Maddison et al., 2016; Jang et al., 2016).

(3) Deterministic Variational inference (DVI) Wu et al. (2018) Our approach is most similar to Wu et al. (2018), which observes that if the underlying model is deterministic, i.e. $\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(l)},\boldsymbol{\theta}^{(l)}]$ is a Dirac function

$$\mathcal{L}_n(\phi) := \mathbb{E}_{\boldsymbol{h}^{(L-1)}\sim P;\, \boldsymbol{\theta}^{(L-1)}\sim Q}\left[\log\left(\mathbf{Pr}[y_n|\boldsymbol{h}^{(L-1)},\boldsymbol{\theta}^{(L-1)}]\right)\right] \tag{17}$$

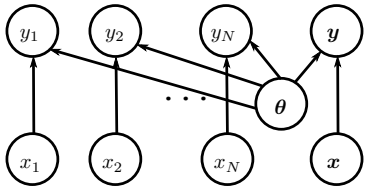
Our approach considers a wider scope of problem settings, where the model could be stochastic, i.e. $\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(l)},\boldsymbol{\theta}^{(l)}]$ is an arbitrary function. Furthermore, Wu et al. (2018) considers the case that the model parameters $\boldsymbol{\theta}$ are Gaussian distributed, whose probabilistic propagation is hard to approximate.

Quantized Neural Networks These models can be categorized into two classes: (1) *Partially quantized networks*, where only weights are discretized Han et al. (2015); Zhu et al. (2016); (2) *Fully quantized networks*, where both weights and hidden units are quantized (Courbariaux et al., 2015; Kim & Smaragdis, 2016; Zhou et al., 2016; Rastegari et al., 2016; Hubara et al., 2017). While both classes provide compact size, low-precision neural network models, fully quantized networks further enjoy fast computation provided by specialized bit-wise operations. In general, quantized neural networks are difficult to train due to their non-differentiability. Gradient descent by backpropagation is approximated by either *straight-through estimators* Bengio et al. (2013) or *probabilistic methods* Esser et al. (2015); Shayer et al. (2017); Peters & Welling (2018). Unlike these papers, we focus on Bayesian learning of fully quantized networks in this paper.

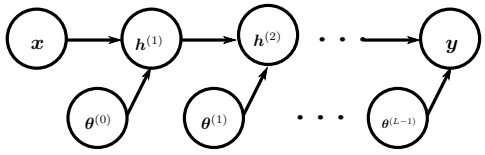
Tensor Networks and Tensorial Neural Networks *Tensor networks* (TNs) are widely used in numerical analysis Grasedyck et al. (2013), quantum physiscs Orús (2014), and recently machine learning Cichocki et al. (2016; 2017) to model interactions among multi-dimensional random objects. Various tensorial neural networks (TNNs) Su et al. (2018); Newman et al. (2018) have been proposed that reduce the size of neural networks by replacing the linear layers with TNs. Recently, Robeva & Seigal (2017) points out the duality between probabilistic graphical models (PGMs) and TNs. I.e. there exists a bijection between PGMs and TNs. Our paper advances this line of thinking by connecting hierarchical Bayesian models (e.g. Bayesian neural networks) and hierarchical TNs.

B SUPERVISED LEARNING WITH BAYESIAN NEURAL NETWORKS (BNNS)

The problem settings of general Bayesian model and Bayesian neural networks for supervised learning are illustrated in Figures 3a and 3b.



(a) Graphical model depiction of the problem setting in Bayesian neural networks.



(b) Graphical model depiction of a Bayesian neural network as a hierarchical model, where predicting $\boldsymbol{y}$ from $\boldsymbol{x}$ can be performed iteratively through the hidden variables $\boldsymbol{h}^{(l)}$'s.

Figure 3: Graphical models.

General Bayesian model Formally, the graphical model in Figure 3a implies the joint distribution of the model parameters $\boldsymbol{\theta}$, the observed dataset $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$ and any unseen data point $(\boldsymbol{x}, \boldsymbol{y})$ is factorized as follows:

$$\mathbf{Pr}[\boldsymbol{x}, \boldsymbol{y}, \mathcal{D}, \boldsymbol{\theta}] = (\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}]\mathbf{Pr}[\boldsymbol{x}]\,(\mathbf{Pr}[\mathcal{D}|\boldsymbol{\theta}]))\,\mathbf{Pr}[\boldsymbol{\theta}] \tag{18}$$

$$= \left(\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}]\mathbf{Pr}[\boldsymbol{x}]\left(\prod_{n=1}^{N}\mathbf{Pr}[y_n|x_n, \boldsymbol{\theta}]\mathbf{Pr}[x_n]\right)\right)\mathbf{Pr}[\boldsymbol{\theta}] \tag{19}$$

where $\mathbf{Pr}[x_i]$'s and $\mathbf{Pr}[\boldsymbol{x}]$ are identically distributed, and so are the conditional distributions $\mathbf{Pr}[y_i|x_i, \boldsymbol{\theta}]$'s and $\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}]$. In other words, we assume that (1) the samples (x_n, y_n)'s (and unseen data point $(\boldsymbol{x}, \boldsymbol{y})$) are are identical and independent distributed according to the same data distribution; and (2) x_n (or $\boldsymbol{x}$) and $\boldsymbol{\theta}$ together predict the output y_n (or $\boldsymbol{y}$) according to the same conditional distribution. Notice that the factorization above also implies the following equations:

$$\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}, \boldsymbol{\theta}] = \mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{\theta}] \tag{20a}$$

$$\mathbf{Pr}[\boldsymbol{\theta}|\boldsymbol{x}, \mathcal{D}] = \mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}] \tag{20b}$$

Without these implications, the posterior predictive distribution $\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}]$ can now expanded as:

$$\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{x}, \mathcal{D}] = \int_\theta \mathbf{Pr}[\boldsymbol{y}|x, \theta, \mathcal{D}]\mathbf{Pr}[\theta|x, \mathcal{D}]d\theta = \int_\theta \mathbf{Pr}[\boldsymbol{y}|x, \theta]\underbrace{\mathbf{Pr}[\theta|\mathcal{D}]}_{\approx Q(\theta;\,\phi)}d\theta \tag{21}$$

where we also assume the posterior distribution $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$ is (approximately) characterized by a parameterized distribution $Q(\boldsymbol{\theta};\phi)$.

Variational Learning The reason we are learning an approximate posterior Q and not the exact distribution $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$ is that for complex models the latter is intractable to compute. The exact posterior $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$ generally does not take the form of $Q(\boldsymbol{\theta};\phi)$ even if its prior $\mathbf{Pr}[\boldsymbol{\theta}]$ does.

A standard approach to finding a good approximation $Q(\boldsymbol{\theta};\phi)$ is *variational inference*, which finds $\phi^\star$ such that the *KL-divergence* $\mathbf{KL}(Q(\boldsymbol{\theta};\phi)||\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}])$ of $Q(\boldsymbol{\theta};\phi)$ from $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$ is minimized (or alternatively the negative KL-divergence is maximized.)

$$\phi^\star = \arg\max_\phi \left(-\mathbf{KL}(Q(\boldsymbol{\theta};\phi)||\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}])\right) \tag{22}$$

$$= \arg\max_\phi \left(-\int_\theta Q(\theta;\phi)\log\left(\frac{Q(\theta;\phi)}{\mathbf{Pr}[\theta|\mathcal{D}]}\right)d\theta\right) \tag{23}$$

where $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}]$ is obtained via standard *Bayes' rule*, i.e. $\mathbf{Pr}[\boldsymbol{\theta}|\mathcal{D}] = \mathbf{Pr}[\mathcal{D}|\boldsymbol{\theta}]\mathbf{Pr}[\boldsymbol{\theta}]/\mathbf{Pr}[\mathcal{D}]$. Now we are able to decompose the maximization objective into two terms by plugging the rule into (23):

$$\mathcal{L}(\phi) = -\int_\theta Q(\theta;\phi)\log\left(Q(\theta;\phi)\cdot\frac{\mathbf{Pr}[\mathcal{D}]}{\mathbf{Pr}[\theta]\mathbf{Pr}[\mathcal{D}|\theta]}\right)d\boldsymbol{\theta} \tag{24}$$

$$= \sum_{n=1}^{N}\int_\theta \log\left(\mathbf{Pr}[y_n|x_n,\theta]\right)Q(\theta;\phi)d\theta + \int_\theta Q(\theta;\phi)\log\left(\frac{Q(\theta;\phi)}{\mathbf{Pr}[\theta]}\right)d\theta + \text{const.} \tag{25}$$

$$= \sum_{n=1}^{N}\underbrace{\mathbb{E}_Q\left[\log\left(\mathbf{Pr}[y_n|x_n,\theta]\right)\right]}_{\mathcal{L}_n(\phi)} + \underbrace{\mathbf{KL}(Q(\boldsymbol{\theta};\phi)||\mathbf{Pr}[\boldsymbol{\theta}])}_{\mathcal{R}(\phi)} - \underbrace{\log\left(\mathbf{Pr}[\mathcal{D}]\right)}_{\text{const.}} \tag{26}$$

where (1) $\mathcal{L}_n(\phi)$ is the *expected log-likelihood (NLL)*, which reflects the predictive performance of the Bayesian model on the data point (x_n, y_n); and (2) $\mathcal{R}(\phi)$ is the KL-divergence between $Q(\boldsymbol{\theta};\phi)$ and its prior $\mathbf{Pr}[\boldsymbol{\theta}]$, which reduces to entropy $H(Q)$ if the prior of $\boldsymbol{\theta}$ follows a uniform distribution.

Hierarchical Bayesian Model A Bayesian neural network can be considered as a hierarchical Bayesian model depicted in Figure 3b, which satisfies the following two assumptions:

Assumption B.1 (Independence of Model Parameters $\boldsymbol{\theta}^{(l)}$). *The approximate posterior $Q(\boldsymbol{\theta};\phi)$ over the model parameters $\boldsymbol{\theta}$ are partitioned into L disjoint and statistical independent* layers $\{\boldsymbol{\theta}^{(l)}\}_{l=0}^{L-1}$ *(where each $\phi^{(l)}$ parameterizes $\boldsymbol{\theta}^{(l)}$ in the l-th layer) such that:*

$$Q(\boldsymbol{\theta};\phi) = \prod_{l=0}^{L-1} Q(\boldsymbol{\theta}^{(l)};\phi^{(l)}) \tag{27}$$

Assumption B.2 (Markovianity of Hidden Units $\boldsymbol{h}^{(l)}$). *The* hidden variables $\boldsymbol{h} = \{\boldsymbol{h}^{(l)}\}_{l=0}^{L}$ *satisfy the* Markov property *that $\boldsymbol{h}^{(l+1)}$ depends on the input $\boldsymbol{x}$ only through its previous layer $\boldsymbol{h}^{(l)}$:*

$$\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(:l)},\boldsymbol{\theta}^{(:l)}] = \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|\boldsymbol{h}^{(l)},\boldsymbol{\theta}^{(l)}] \tag{28}$$

where we use short-hand notations $\boldsymbol{h}^{(:l)}$ and $\boldsymbol{\theta}^{(:l)}$ to represent the sets of previous layers $\{\boldsymbol{h}^{(k)}\}_{k=0}^{l}$ and $\{\boldsymbol{\theta}^{(k)}\}_{k=0}^{l}$. For consistency, we denote $\boldsymbol{h}^{(0)} = \boldsymbol{x}$ and $\boldsymbol{h}^{(L)} = \boldsymbol{y}$.

Proof of probabilistic prorogation Based on the assumptions above, we now provide a proof for *probabilistic propagation* in Equation (1) as follows:

$$\overbrace{\mathbf{Pr}[\boldsymbol{h}^{(l+1)}|x]}^{P\left(\boldsymbol{h}^{(l+1)};\,\psi^{(l+1)}\right)} = \int_{\theta^{(:\,l)}} \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|x,\theta^{(:\,l)}]\; Q(\theta^{(:\,l)};\phi^{(:\,l)})\; d\theta^{(:\,l)} \tag{29}$$

$$= \int_{\theta^{(:\,l)}} \left(\int_{h^{(l)}} \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|h^{(l)},\theta^{(l)}] \mathbf{Pr}[h^{(l)}|x,\theta^{(:\,l-1)}] dh^{(l)} \right) Q(\theta^{(:\,l)};\phi^{(:\,l)})\; d\theta^{(:\,l)} \tag{30}$$

$$= \int_{h^{(l)},\theta^{(l)}} \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|h^{(l)},\theta^{(l)}] Q(\theta^{(l)};\phi^{(l)}) \left(\int_{\theta^{(:\,l-1)}} \mathbf{Pr}[h^{(l)}|x,\theta^{(:\,l-1)}] Q(\theta^{(:\,l-1)};\phi^{(:\,l-1)}) d\theta^{(:\,l-1)} \right) dh^{(l)} d\theta^{(l)} \tag{31}$$

$$= \int_{h^{(l)},\theta^{(l)}} \mathbf{Pr}[\boldsymbol{h}^{(l+1)}|h^{(l)},\theta^{(l)}] Q(\theta^{(l)};\phi^{(l)})\; \underbrace{\mathbf{Pr}[h^{(l)}|x]}_{P\left(h^{(l)};\,\psi^{(l)}\right)}\; dh^{(l)} d\theta^{(l)} \tag{32}$$

C Alternative Evidence Lower Bound and its Analytic Forms

C.1 Alternative Evidence Lower Bound (Proof for Theorem 3.1)

The steps to prove the inequality (6) almost follow the ones for probabilistic propagation above:

$$\mathcal{L}_n(\phi) = \mathbb{E}_Q\left[\log(\mathbf{Pr}[y_n|x_n,\boldsymbol{\theta}])\right] \tag{33}$$

$$= \int_\theta \log\left(\mathbf{Pr}[y_n|x_n,\theta]\right) Q(\theta;\phi) d\theta \tag{34}$$

$$= \int_\theta \log\left(\int_{h^{(L-1)}} \mathbf{Pr}[y_n, h^{(L-1)}|x_n,\theta] dh^{(L-1)} \right) Q(\theta;\phi) d\theta \tag{35}$$

$$= \int_\theta \log\left(\int_{h^{(L-1)}} \mathbf{Pr}[y_n|h^{(L-1)},\theta^{(L-1)}] \mathbf{Pr}[h^{(L-1)}|x_n,\theta^{(0:L-2)}] dh^{(L-1)} \right) Q(\theta;\phi) d\theta \tag{36}$$

$$\geq \int_\theta \left(\int_{h^{(L-1)}} \log\left(\mathbf{Pr}[y_n|h^{(L-1)},\theta^{(L-1)}] \right) \mathbf{Pr}[h^{(L-1)}|x_n,\theta^{(0:L-1)}] dh^{(L-1)} \right) Q(\theta;\phi) d\theta \tag{37}$$

$$= \int_{h^{(L-1)},\theta^{(L-1)}} \log\left(\mathbf{Pr}[y_n|h^{(L-1)},\theta^{(L-1)}] \right) Q(\theta^{(L-1)};\phi^{(L-1)}) \left(\int_{\theta^{(0:L-2)}} \mathbf{Pr}[h^{(L-1)}|x_n,\theta^{(0:L-2)}] Q(\theta^{(0:L-2)};\phi^{(0:L-2)}) d\theta^{(0:L-2)} \right) dh^{(L-1)} d\theta^{(L-1)} \tag{38}$$

$$= \int_{h^{(L-1)},\theta^{(L-1)}} \log\left(\mathbf{Pr}[y_n|h^{(L-1)},\theta^{(L-1)}] \right) Q(\theta^{(L-1)}) \mathbf{Pr}[h^{(L-1)}|x_n] dh^{(L-1)} d\theta^{(L-1)} \tag{39}$$

$$= \mathbb{E}_{\boldsymbol{h}^{(L-1)}\sim P;\,\boldsymbol{\theta}^{(L-1)}\sim Q}\left[\log\left(\mathbf{Pr}[y_n|h^{(L-1)},\theta^{(L-1)}] \right) \right] \tag{40}$$

where the key step is the Jensen's inequality of $\mathbb{E}_Q\left[\log(\cdot)\right] \geq \log\left(\mathbb{E}_Q\left[\cdot\right]\right)$ in (37).

C.2 Softmax Layer for Classification Problem (Proof for Theorem 3.2)

In this part, we first prove the alternative evidence lower bound (ELBO) for Bayesian neural networks with *softmax* function as their last layers. Subsequently, we derive the corresponding back-propagation rule for the softmax layer. Finally, we show a method based on *Taylor's expansion* to approximately evaluate a softmax layer without *Monte Carlo sampling*.

Proof. Recall that a softmax layer maps the pre-activations from the last hidden layer $\boldsymbol{h} = \boldsymbol{h}^{(L-1)} \in \mathbb{Q}^K$ to a categorical distribution $\triangle^{K-1}$ over possible classes $\boldsymbol{y} \in \{1,\cdots,K\}$ with a (scaled) softmax function.

$$\mathbf{Pr}[\boldsymbol{y} = c|\boldsymbol{h};s] = \frac{\exp(\boldsymbol{h}_c/s)}{\sum_{k=1}^K \exp(\boldsymbol{h}_k/s)} \tag{41}$$

As we show in Section 4, the distribution of $\boldsymbol{h}$ is assumed as a product of independent Gaussian distributions, i.e. $\boldsymbol{h}_k \sim \mathcal{N}(\mu_k, \nu_k)$'s are independent of each other, where μ_k and ν_k are mean and variance of $\boldsymbol{h}_k$. Then the lower bound follows by plugging the definitions of $\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{h}, s]$ and $\mathbf{Pr}[\boldsymbol{h}_k|x]$ into Equation (6).

$$\mathcal{L}_n(\phi) \geq \int_h \log\left(\mathbf{Pr}[y_n = c|h; s]\right) \mathbf{Pr}[h|x] dh \tag{42}$$

$$= \int_h \log\left(\mathbf{Pr}[y_n = c|h; s]\right) \left(\prod_{k=1}^{K} \mathbf{Pr}[h_k|x]\right) dh \tag{43}$$

$$= \int_h \left(\frac{h_c}{s} - \log\left(\sum_{k=1}^{K} \exp\left(\frac{h_k}{s}\right)\right)\right) \left(\prod_{k=1}^{K} \mathbf{Pr}[h_k|x]\right) dh \tag{44}$$

$$= \frac{1}{s}\int_{h_c} h_c \mathbf{Pr}[h_c|x_n] dh_c - \int_h \log\left(\sum_{k=1}^{K} \exp\left(\frac{h_k}{s}\right)\right) \left(\prod_{k=1}^{K} \mathbf{Pr}[h_k|x]\right) dh \tag{45}$$

$$= \frac{\mu_c}{s} - \int_h \log\left(\sum_{k=1}^{K} \exp\left(\frac{h_k}{s}\right)\right) \left(\prod_{k=1}^{K} \mathbf{Pr}[h_k|x]\right) dh \tag{46}$$

$$\leq \frac{\mu_c}{s} - \log\left(\int_h \sum_{k=1}^{K} \exp\left(\frac{h_k}{s}\right) \left(\prod_{k=1}^{K} \mathbf{Pr}[h_k|x]\right) dh\right) \tag{47}$$

$$= \frac{\mu_c}{s} - \log\left(\sum_{k=1}^{K} \int_{h_k} \exp\left(\frac{h_k}{s}\right) \mathbf{Pr}[h_k|x] dh_k\right) \tag{48}$$

$$= \frac{\mu_c}{s} - \log\left(\sum_{k=1}^{K} \int_{h_k} \exp\left(\frac{h_k}{s}\right) \cdot \frac{1}{\sqrt{2\pi\nu_k}} \exp\left(-\frac{(h_k - \mu_k)^2}{2\nu_k}\right) dh_k\right) \tag{49}$$

$$= \frac{\mu_c}{s} - \log\left(\sum_{k=1}^{K} \exp\left(\frac{\mu_k}{s} + \frac{\nu_k}{2s^2}\right)\right) = \overline{\mathcal{L}}_n(\phi) \tag{50}$$

where the last equation follows

$$\int_{h_k} \exp\left(\frac{h_k}{s}\right) \cdot \frac{1}{\sqrt{2\pi\nu_k}} \exp\left(-\frac{(h_k - \mu_k)^2}{2\nu_k}\right) dh_k \tag{51}$$

$$= \int_{h_k} \frac{1}{\sqrt{2\pi\nu_k}} \exp\left(-\frac{h_k^2 - 2(\mu_k + \nu_k/s)h_k + \mu_k^2}{2\nu_k}\right) dh_k \tag{52}$$

$$= \underbrace{\int_{h_k} \frac{1}{\sqrt{2\pi\nu_k}} \exp\left(-\frac{\left(h_k - (\mu_k + \nu_k)\right)^2}{2\nu_k}\right) dh_k}_{} \cdot \exp\left(\frac{\mu_k}{s} + \frac{\nu_k}{2s^2}\right) \tag{53}$$

where the under-braced term is unity since it is From Equation (46) to Equation (47), we use the Jensen's Inequality to achieve an upper bound for integral of log sum exponential. However, there are advanced techniques as in Khan (2012) for tighter bounds. □

Derivatives of $\overline{\mathcal{L}}_n(\phi)$ in (9) In order to use the backpropagation algorithm to obtain the gradients w.r.t. the parameters from previous layers as in Equation (8), we first need to obtain the derivatives w.r.t. $\psi^{(L-1)} = \{\mu_k, \nu_k\}_{k=1}^{K}$.

$$\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial \mu_k} = -\frac{1}{s}\left(\frac{\exp\left(\mu_k/s + \nu_k/2s^2\right)}{\sum_{k=1}^{K} \exp\left(\mu_k/s + \nu_k/2s^2\right)} - \mathbf{1}[k = c]\right) \tag{54a}$$

$$\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial \nu_k} = -\frac{1}{2s^2}\left(\frac{\exp\left(\mu_k/s + \nu_k/2s^2\right)}{\sum_{k=1}^{K} \exp\left(\mu_k/s + \nu_k/2s^2\right)}\right) \tag{54b}$$

Furthermore, the scale s can be (optionally) updated simultaneously with other parameters using the gradient of $\overline{\mathcal{L}}_n(\phi)$.

$$\frac{\partial \overline{\mathcal{L}}_n(\phi)}{\partial s} = -\frac{\mu_c}{s^2} + \frac{\sum_{k=1}^K \left(\mu_k/s^2 + \nu_k/s^3\right) \exp\left(\mu_k/s + \nu_k/2s^2\right)}{\sum_{k=1}^K \exp\left(\mu_k/s + \nu_k/2s^2\right)} \tag{55}$$

Prediction with Softmax Layer Once we learn the parameters for the Bayesian neural network, in principle we can predict the distribution of $\boldsymbol{y}$ by evaluating the following equation:

$$\mathbf{Pr}[\boldsymbol{y} = c|\boldsymbol{x}] = \int_h \mathbf{Pr}[\boldsymbol{y} = c|\boldsymbol{h}, s]\mathbf{Pr}[h|\boldsymbol{x}]dh = \int_h \ell_c(h)\mathbf{Pr}[h|\boldsymbol{x}]dh \tag{56}$$

$$\text{(Mean-field assumption)} \ = \int_{h_1} \cdots \int_{h_K} \ell_c(h) \left(\prod_{k=1}^K \mathbf{Pr}[h_k|x]\right) dh_1 \cdots dh_k \tag{57}$$

where we denote the softmax function as $\ell_c(h)$ for simplicity, i.e. $\ell_c(h) = \exp(h_c/s)/[\sum_k \exp(h_k/s)]$. Unfortunately, the equation above can not be computed in closed form. The most straight-forward work-around is to approximate the integral by Monte Carlo sampling: for each $\boldsymbol{h}_k$ we draw S samples $\{h_k^s\}_{s=1}^S$ independently and compute the prediction:

$$\mathbf{Pr}[\boldsymbol{y} = c|x] \approx \frac{1}{S}\sum_{s=1}^S \ell_c(h^s), \ \forall c \in [K] \tag{58}$$

Despite its conceptual simplicity, Monte Carlo method suffers from expensive computation and high sampling variance. In this paper, we propose an economical and accurate estimate based on Taylor's expansion. To begin with, we first expand the function $\ell_c(h)$ by Taylor's series at the point μ (up to the second order):

$$\ell_c(h) = \ell_c(\mu) + \left[\frac{\partial \ell_c}{\partial h}(\mu)\right]^\top (h-\mu) + \frac{1}{2}(h-\mu)^\top \left[\frac{\partial^2 \ell_c}{\partial h^2}(\mu)\right](h-\mu) + O\left(\|h-c\|^3\right) \tag{59}$$

$$= \ell_c(\mu) + \sum_{k=1}^K \left[\frac{\partial \ell_c}{\partial h_k}(\mu)\right](h_k - \mu_k) + \sum_{i=1}^K\sum_{j=1}^K \left[\frac{\partial^2 \ell_c}{\partial h_i h_j}(\mu)\right](h_i-\mu_i)(h_j-\mu_j) + O\left(\|h-\mu\|^3\right) \tag{60}$$

Before we derive the forms of these derivatives, we first show the terms of odd orders do not contribute to the expectation. For example, if $\ell_c(h)$ is approximated by its first two terms (i.e. a linear function), Equation (57) can be written as

$$\mathbf{Pr}[\boldsymbol{y} = c|x] \approx \int_{h_1} \cdots \int_{h_K} \left(\ell_c(\mu) + \sum_{k=1}^K \left[\frac{\partial \ell_c}{\partial h_k}(\mu)\right](h_k - \mu_k)\right)\left(\prod_{k=1}^K \mathbf{Pr}[h_k|x]\right) dh_1 \cdots dh_k \tag{61}$$

$$= \ell_c(\mu) + \sum_{k=1}^K \left[\frac{\partial \ell_c}{\partial h_k}(\mu)\right]\left(\int_{h_k} (h_k - \mu_k)\,\mathbf{Pr}[h_k|x]dh_k\right) = \ell_c(\mu) \tag{62}$$

where the second term is zero by the symmetry of $\mathbf{Pr}[\boldsymbol{h}_k|\boldsymbol{x}]$ around μ_k (or simply the definition of μ_k's). Therefore, the first-order approximation results exactly in a (deterministic) softmax function of the mean vector μ. In order to incorporate the variance into the approximation, we will need to derive the exact forms of the derivatives of $\ell_c(h)$. Specifically, the first-order derivatives are obtained from the definition of $\ell_c(h)$.

$$\frac{\partial \ell_c}{\partial h_c}(h) = \frac{1}{s}\cdot\frac{\exp(h_c/s) - \exp(2h_c/s)}{\left(\sum_{k=1}^K \exp(h_k/s)\right)^2} = \frac{1}{s}\left(\ell_c(h) - \ell_c^2(h)\right) \tag{63a}$$

$$\frac{\partial \ell_c}{\partial h_k}(h) = -\frac{1}{s}\cdot\frac{\exp(h_c/s)\cdot\exp(h_k/s)}{\left(\sum_{k=1}^K \exp(h_k/s)\right)^2} = -\frac{1}{s}\ell_c(h)\ell_k(h), \ \forall k \neq c \tag{63b}$$

and subsequently the second-order derivatives from the first ones:

$$\frac{\partial^2 \ell_c}{\partial h_c^2}(h) = \frac{1}{s}\left(\frac{\partial \ell_c}{\partial h_c}(h) - 2\ell_c(h)\frac{\partial \ell_c}{\partial h_c}(h)\right) = \frac{1}{s^2}\left(\ell_c(h) - 3\ell_c^2(h) + 2\ell_c^3(h)\right) \tag{64a}$$

$$\frac{\partial^2 \ell_c}{\partial h_k^2}(h) = -\frac{1}{s}\left(\frac{\partial \ell_c}{\partial h_c}(h)\ell_k(h) + \ell_c(h)\frac{\partial \ell_k}{\partial h_c}(h)\right) = \frac{1}{s^2}\left(-\ell_c(h)\ell_k(h) + 2\ell_c^2(h)\ell_k(h)\right),\ \forall k \neq c \tag{64b}$$

with these derivatives we can compute the second-order approximation as

$$\mathbf{Pr}[\boldsymbol{y} = c|x] \approx \int_{h_1,\cdot,h_K}\left(\ell_c(\mu) + \frac{1}{2}\sum_{i=1}^{K}\sum_{j=1}^{K}\frac{\partial^2 \ell_c}{\partial \mu_i \mu_j}(\mu)(h_i - \mu_i)(h_j - \mu_j)\right)\left(\prod_{k=1}^{K}\mathbf{Pr}[h_k|x]\right)dh_1 \cdots dh_K \tag{65}$$

$$= \ell_c(\mu) + \frac{1}{2}\frac{\partial^2 \ell_c}{\partial \mu_c^2}(\mu)\int_{h_c}(h_c - \mu_c)^2\mathbf{Pr}[h_c|x]dh_c + \frac{1}{2}\sum_{k\neq c}\frac{\partial^2 \ell_c}{\partial \mu_k^2}(\mu)\int_{h_k}(h_k - \mu_k)^2\mathbf{Pr}[h_k|x]dh_k \tag{66}$$

$$= \ell_c(\mu) + \frac{1}{2s^2}\left(\ell_c(\mu) - 3\ell_c^2(\mu) + 2\ell_c^3(\mu)\right)\nu_c + \frac{1}{2s^2}\sum_{k\neq c}\left(-\ell_c(\mu)\ell_k(\mu) + 2\ell_c^2(\mu)\ell_k(\mu)\right)\nu_k \tag{67}$$

$$= \ell_c(\mu) + \frac{1}{2s^2}\left(\ell_c(\mu) - 2\ell_c^2(\mu)\right)\left(\nu_c - \sum_{k=1}^{K}\ell_k(\mu)\nu_k\right) \tag{68}$$

$$\mathbf{Pr}[\boldsymbol{y}|x] \approx \ell(\mu) + \frac{1}{2s^2}\left(\ell(\mu) - \ell(\mu)^{\circ 2}\right) \circ \left(\nu - \ell(\mu)^\top \nu\right) \tag{69}$$

C.3 Gaussian Output Layer for Regression Problem (Proof for Theorem 3.3)

In this part, we develop an alternative evidence lower bound (ELBO) for Bayesian neural networks with Gaussian output layers, and derive the corresponding gradients for backpropagation. Despite the difficulty to obtain an analytical predictive distribution for the output, we show that its *central moments* can be easily computed given the learned parameters.

Regression: Gaussian Layer When QNNs are used in regression problems, the output layer needs to map the discretized hidden units $\boldsymbol{h} \in \mathbb{Q}^I$ back to a continuous prediction $\boldsymbol{y} \in \mathbb{R}$. One common choice is a linear function: $\boldsymbol{y} = w^\top \boldsymbol{h} + \boldsymbol{\epsilon}$, where $w \in \mathbb{R}^I$ is a continuous-valued vector and $\boldsymbol{\epsilon} \sim \mathcal{N}(0, s)$ represents the uncertainty of the prediction. Therefore, the *Gaussian layer* characterized by $\phi^{(L-1)} = \{w, s\}$ and computes a distribution of $\boldsymbol{y}$ as:

$$\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{h}; w, s] = \frac{1}{\sqrt{2\pi s}}\exp\left(-\frac{(\boldsymbol{y} - w^\top \boldsymbol{h})^2}{2s}\right) \tag{70}$$

Let $\mu_i = \mathbb{E}\left[\boldsymbol{h}_i|x\right]$ and $\nu_i = \mathbb{V}\left[\boldsymbol{h}_i|x\right]$ denote the mean and variance of $\boldsymbol{h}_i$,

and use $\mu = [\mu_1, \cdots, \mu_I]^\top \in \mathbb{R}^I$ and $\nu = [\nu_1, \cdots, \nu_I]^\top \in \mathbb{R}^I$ to represent the vectors of mean and variance. Now we are ready to prove Equation (10) by plugging the definition of $\mathbf{Pr}[\boldsymbol{y}|\boldsymbol{h}; w, s]$ into Equation (6).

$$\mathcal{L}_n(\phi) \geq \sum_{h_1}\cdots\sum_{h_I}\log\left(\mathbf{Pr}[y|h_1, \cdots, h_I; w, s]\right)\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x_n]\right) \tag{71}$$

$$= -\sum_{h_1}\cdots\sum_{h_I}\left(\frac{\left(y - \sum_{i=1}^{I} w_i h_i\right)^2}{2s} + \frac{\log(2\pi s)}{2}\right)\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x_n]\right) \tag{72}$$

$$= -\frac{1}{2s}\sum_{h_1}\cdots\sum_{h_I}\left(y - \sum_{i=1}^{I} w_i h_i\right)^2\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x_n]\right) - \frac{\log(2\pi s)}{2} \tag{73}$$

where the long summation in the first term can be further simplified with notations of μ and ν:

$$\sum_{h_1}\cdots\sum_{h_I}\left(y-\sum_{i=1}^{I}w_ih_i\right)^2\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x_n]\right) \tag{74}$$

$$=\sum_{h_1}\cdots\sum_{h_I}\left(y^2-2y\sum_{i=1}^{I}w_ih_i+\sum_{i=1}^{I}w_i^2h_i^2+\sum_{j=1}^{I}\sum_{k\neq j}w_jw_kh_jh_k\right)\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x_n]\right) \tag{75}$$

$$\begin{aligned}=&y^2-2y\sum_{i=1}^{I}w_i\left(\sum_{h_i}h_i\mathbf{Pr}[h_i|x]\right)+\sum_{i=1}^{I}w_i^2\left(\sum_{h_i}h_i^2\mathbf{Pr}[h_i|x_n]\right)\\&+\sum_{j=1}^{I}\sum_{k\neq j}w_jw_k\left(\sum_{h_j}h_j\mathbf{Pr}[h_j|x_n]\right)\left(\sum_{h_k}h_k\mathbf{Pr}[h_k|x_n]\right)\end{aligned} \tag{76}$$

$$=y^2-2y\sum_{i=1}^{I}w_i\mu_i+\sum_{i=1}^{I}w_i^2(\mu_i^2+\nu_i)+\sum_{j=1}^{I}\sum_{k\neq j}w_jw_k\mu_j\mu_k \tag{77}$$

$$=y^2-2y\sum_{i=1}^{I}w_i\mu_i+\sum_{i=1}^{I}w_i^2\nu_i+\left(\sum_{j=1}^{I}w_j\mu_j\right)\left(\sum_{k=1}^{I}w_k\mu_k\right) \tag{78}$$

$$=y^2-2y\,w^\top\mu+(w^{\circ 2})^\top\nu+\left(w^\top\mu\right)^2 \tag{79}$$

$$=(y-w^\top\mu)^2+(w^{\circ 2})^\top\nu \tag{80}$$

where $w^{\circ 2}$ denotes element-wise square, i.e. $w^{\circ 2}=[w_1^2,\cdots,w_I^2]^\top$. Thus, we show that $\mathcal{L}_n(\phi)$ is bounded by $\overline{\mathcal{L}}_n(\phi)$, completing the proof of theorem 3.3:

$$\mathcal{L}_n(\phi)\geq-\frac{(y-w^\top\mu)^2+(w^{\circ 2})^\top\nu}{2s}-\frac{\log(2\pi s)}{2}=\overline{\mathcal{L}}_n(\phi) \tag{81}$$

Derivatives of $\overline{\mathcal{L}}_n(\phi)$ in Equation (10) It is not difficult to show that the gradient of $\overline{\mathcal{L}}_n(\phi)$ can be easily backpropagated through the last layer. Actually, we only need to show the derivatives of $\overline{\mathcal{L}}_n(\phi)$ w.r.t. μ and ν:

$$\frac{\partial\overline{\mathcal{L}}_n(\phi)}{\partial\mu}=-\frac{(y-w^\top\mu)w}{s} \tag{82a}$$

$$\frac{\partial\overline{\mathcal{L}}_n(\phi)}{\partial\nu}=-\frac{w^{\circ 2}}{2s} \tag{82b}$$

with which the gradients of $\mathbf{Pr}[h_i|x_n]$'s are then obtained similar to Equation (113). Furthermore, the parameters of the Gaussian output layer can be updated along with all other parameters based on their gradients:

$$\frac{\partial\overline{\mathcal{L}}_n(\phi)}{\partial w}=-\frac{(y-w^\top\mu)\mu+(w\circ\nu)}{s} \tag{83a}$$

$$\frac{\partial\overline{\mathcal{L}}_n(\phi)}{\partial s}=-\frac{1}{2s}+\frac{(y-w^\top\mu)^2+(w^{\circ 2})^\top\nu}{2s^2} \tag{83b}$$

Prediction with Gaussian Layer Once we determine the parameters for the last layer, in principle we can compute the predictive distribution $\mathbf{Pr}[\boldsymbol{y}|x]$ for the output $\boldsymbol{y}$ given the input x according to

$$\begin{aligned}\mathbf{Pr}[\boldsymbol{y}|x]=\sum_{h}\mathbf{Pr}[\boldsymbol{y}|h;w,s]\mathbf{Pr}[h|\boldsymbol{x}]&=\sum_{h_1}\cdots\sum_{h_I}\mathbf{Pr}[\boldsymbol{y}|h;w,s]\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x]\right)\\&=\sum_{h_1}\cdots\sum_{h_I}\frac{1}{\sqrt{2\pi s}}\exp\left(-\frac{\left(\boldsymbol{y}-\sum_{i=1}^{I}w_ih_i\right)^2}{2s}\right)\left(\prod_{i=1}^{I}\mathbf{Pr}[h_i|x]\right)\end{aligned} \tag{84}$$

Unfortunately, exact computation of the equation above for arbitrary output value y is intractable in general. However, the central moments of the predictive distribution $\mathbf{Pr}[\boldsymbol{y}|x]$ are easily accessible: consider we interpret the prediction as $\boldsymbol{y} = w^\top \boldsymbol{h} + \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, s)$ its mean and variance can be easily computed as

$$\mathbb{E}\left[\boldsymbol{y}|x\right] = w^\top \mathbb{E}\left[\boldsymbol{h}\right] = w^\top \mu \tag{85a}$$

$$\mathbb{V}\left[\boldsymbol{y}|x\right] = (w^{\circ 2})^\top \mathbb{V}\left[\boldsymbol{h}\right] + \mathbb{V}\left[\boldsymbol{\epsilon}\right] = (w^{\circ 2})^\top \nu + s \tag{85b}$$

Furthermore, if we denote the (normalized) *skewness* and *kurtosis* of $\boldsymbol{h}_i$ as γ_i and κ_i:

$$\gamma_i = \mathbb{E}\left[(\boldsymbol{h}_i - \mu_i)^3|x\right] / \nu_i^{3/2} = \sum_{h_i} (h_i - \mu_i)^3 \mathbf{Pr}[h_i|x] / \nu_i^{3/2} \tag{86a}$$

$$\kappa_i = \mathbb{E}\left[(\boldsymbol{h}_i - \mu_i)^4|x\right] / \nu_i^2 \quad = \sum_{h_i} (h_i - \mu_i)^4 \mathbf{Pr}[h_i|x] / \nu_i^2 \tag{86b}$$

Then the (normalized) skewness and kurtosis of the prediction $\boldsymbol{y}$ are also easily computed with the vectors of $\gamma = [\gamma_1, \cdots, \gamma_I]^\top \in \mathbb{R}^I$ and $\kappa = [\kappa_1, \cdots, \kappa_I] \in \mathbb{R}^I$.

$$\gamma[\boldsymbol{y}|x] = \frac{\mathbb{E}\left[(\boldsymbol{y} - w^\top \mu)^3|x\right]}{\mathbb{V}\left[\boldsymbol{y}|x\right]^{3/2}} = \frac{(w^{\circ 3})^\top (\gamma \circ \nu^{\circ 3/2})}{\left[(w^{\circ 2})^\top \nu + s\right]^{3/2}} \tag{87a}$$

$$\kappa[\boldsymbol{y}|x] = \frac{\mathbb{E}\left[(\boldsymbol{y} - w^\top \mu)^4|x\right]}{\mathbb{V}\left[\boldsymbol{y}|x\right]^2} = \frac{(w^{\circ 4})^\top (\kappa \circ \nu^{\circ 2}) + s(w^{\circ 2})^\top \nu}{\left[(w^{\circ 2})^\top \nu + s\right]^2} \tag{87b}$$

D Probabilistic Propagation in Bayesian Quantized Networks (BQNs)

D.1 Layers in Neural Networks

Linear Layers Linear layers (followed by a nonlinear transformations $\sigma(\cdot)$) are the most important building blocks in neural networks, which include *fully-connected* and *(depth-wise) convolutional layers*. A linear layer is parameterized by a set of vectors $\boldsymbol{\theta}^{(l)} = \{\boldsymbol{\theta}_j^{(l)}\}_{j=1}^J$, and maps $\boldsymbol{h}^{(l)} \in \mathbb{R}^I$ to $\boldsymbol{h}^{(l+1)} \in \mathbb{R}^J$ with inner products:

$$\boldsymbol{h}_j^{(l+1)} = \sigma\left(\sum_{i \in \mathcal{I}(j)} \boldsymbol{\theta}_{ji}^{(l)} \cdot \boldsymbol{h}_i^{(l)}\right) = \sigma\left(\sum_{i \in \mathcal{I}(j)} \boldsymbol{u}_{ji}^{(l)}\right) = \sigma\left(\boldsymbol{v}_j^{(l+1)}\right) \tag{88}$$

where $\boldsymbol{u}_{ji}^{(l)} = \boldsymbol{\theta}_{ji}^{(l)} \cdot \boldsymbol{h}_i^{(l)}$ and $\boldsymbol{v}_j^{(l+1)} = \sum_{i \in \mathcal{I}(j)} \boldsymbol{u}_{ji}^{(l)}$. Notice (1) $\boldsymbol{u}_{ji}^{(l)}$ only depends on two variables $\boldsymbol{\theta}_{ji}^{(l)}$ and $\boldsymbol{h}_i^{(l)}$; and (2) $\boldsymbol{h}_j^{(l+1)}$ depends on $\boldsymbol{v}_j^{(l+1)}$ through an nonlinear function, both of which can be easily computed. The remaining problem is therefore how to compute $\boldsymbol{v}_j^{(l+1)}$ from $\{\boldsymbol{u}_{ji}^{(l)}\}_{i=1}^I$, or in general **summation of multiple discrete random variables**.

Pooling Layers We also derive fast algorithms for *max pooling layer* and *probabilistic pooling layer*. For each output unit, (1) a max pooling layer picks the maximum value from a number of input units, i.e. $\boldsymbol{h}_j^{(l+1)} = \max_{i \in \mathcal{I}(j)} \boldsymbol{h}_i^{(l)}, i \in \mathcal{I}(j)$; while (2) a probabilistic pooling layer selects the value randomly with a uniform distribution. For both cases, the predictive distribution of $\boldsymbol{h}_j^{(l+1)}$ can be computed $\forall, q \in \mathbb{Q}$ as:

$$\text{Max:} \quad P(\boldsymbol{h}_j^{(l+1)} \leq q) = \prod_{i \in \mathcal{I}(j)} P(\boldsymbol{h}_i^{(l)} \leq q) \tag{89}$$

$$\text{Prob:} \quad P(\boldsymbol{h}_j^{(l+1)} = q) = \sum_{i \in \mathcal{I}(j)} \frac{P(\boldsymbol{h}_i^{(l)} = q)}{|\mathcal{I}(j)|} \tag{90}$$

where $P(\boldsymbol{h}_i^{(l)} \leq q)$ is the *culminative mass function* of P. Complexities for both layers are $O(ID)$.

D.2 DISCRETE FOURIER TRANSFORM (PROOF FOR THEOREM 4.2)

In the part, we prove the *convolution theorem* for discrete random variables and show how *discrete Fourier transform (DFT)* (indeed *fast Fourier transform (FFT)*) greatly accelerates the computation of additions. In order to apply DFT neural networks, we also derive its backpropagation rule.

Proof of Convolution Theorem Consider two independent discrete random variables $\boldsymbol{u}_1 \in [b_1, B_1]$, $\boldsymbol{u}_2 \in [b_2, B_2]$ and their sum $\boldsymbol{v} = \boldsymbol{u}_1 + \boldsymbol{u}_2 \in [b, B]$, where $b = b_1 + b_2$ and $B = B_1 + B_2$. Denote the *probability vectors* (i.e. *probability mass functions*) of $\boldsymbol{u}_1$, $\boldsymbol{u}_2$ and $\boldsymbol{v}$ as $P_1 \in \triangle^{B_1 - b_1}$, $P_2 \in \triangle^{B_2 - b_2}$ and $P \in \triangle^{B-b}$ respectively, then the entries in P are computed with P_1 and P_2 by standard convolution as follows:

$$P(v) = \sum_{u_1=b_1}^{B_1} P_1(u_1) P_2(v - u_1) = \sum_{u_2=b_2}^{B_2} P_1(v - u_2) P_2(u_2),\ \forall v \in \{b, \cdots, B\} \tag{91}$$

The relation above is usually denoted as $P = P_1 * P_2$, where $*$ is the symbol for convolution. Now define the *characteristic functions* C, C_1, and C_2 as the *discrete Fourier transform (DFT)* of the probability vectors P, P_1 and P_2:

$$C(f) = \sum_{v=b}^{B} P(v) \exp\left(-\mathrm{j}\frac{2\pi}{R}(v-b)f\right),\ f \in [R] \tag{92a}$$

$$C_i(f) = \sum_{u_i=b_i}^{B_i} P_i(u_i) \exp\left(-\mathrm{j}\frac{2\pi}{R}(u_i - b_i)f\right), f \in [R] \tag{92b}$$

where R controls the *resolution* of the Fourier transform, typically chosen as $R = B - b + 1$ (i.e. the range of possible values). In this case, the characteristic functions are complex vectors of same length R, i.e. $C, C_1, C_2 \in \mathbb{C}^R$, and we denote the (functional) mappings as $C = \mathcal{F}(P)$ and $C_i = \mathcal{F}_i(P_i)$. Given a characteristic function, its original probability vector can be recovered by *inverse discrete Fourier transform (IDFT)*:

$$P(v) = \frac{1}{R}\sum_{f=0}^{R-1} C(f) \exp\left(\mathrm{j}\frac{2\pi}{R}(v-b)f\right),\ \forall v \in \{b, \cdots, B\} \tag{93a}$$

$$P_i(u_i) = \frac{1}{R}\sum_{f=0}^{R-1} C_i(f) \exp\left(\mathrm{j}\frac{2\pi}{R}(u_i - b_i)f\right),\ \forall u_i \in \{b_i, \cdots, B_i\} \tag{93b}$$

which we denote the inverse mapping as $P = \mathcal{F}^{-1}(C)$ and $P_i = \mathcal{F}_i^{-1}(C_i)$. Now we plug the convolution in Equation (91) into the characteristic function $C(f)$ in (92a) and rearrange accordingly:

$$C(f) = \sum_{v=b}^{B}\left(\sum_{u_1=b_1}^{B_1} P_1(u_1) P_2(v-u_1)\right)\exp\left(-\mathrm{j}\frac{2\pi}{R}(v-b)f\right) \tag{94}$$

$$(\text{Let } u_2 = v - u_1) \ = \sum_{u_1=b_1}^{B_1}\sum_{u_2=b_2}^{B_2} P_1(u_1) P_2(u_2) \exp\left(-\mathrm{j}\frac{2\pi}{R}(u_1 + u_2 - b)f\right) \tag{95}$$

$$(\text{Since } b = b_1 + b_2) \ = \begin{bmatrix}\displaystyle\sum_{u_1=b_1}^{B_1} P_1(u_1)\exp\left(-\mathrm{j}\frac{2\pi}{R}(u_1-b_1)f\right)\end{bmatrix} \begin{bmatrix}\displaystyle\sum_{u_2=b_2}^{B_2} P_2(u_2)\exp\left(-\mathrm{j}\frac{2\pi}{R}(u_2-b_2)f\right)\end{bmatrix} \tag{96}$$

$$= C_1(f) \cdot C_2(f) \tag{97}$$

The equation above can therefore be written as $C = C_1 \circ C_2$, where we use $\circ$ to denote element-wise product. Thus, we have shown summation of discrete random variables corresponds to element-wise product of their characteristic functions. This result can be easily extended to multiple variables case:

Let $\boldsymbol{v} \in [b, B]$ be the sum of I independent discrete random variables $\boldsymbol{u}_i \in [b_i, B_i]$ (therefore $b = \sum_{i=1}^{I} b_i$ and $B = \sum_{i=1}^{I} B_i$) and accordingly define their probability vectors P, P_i and characteristic functions C, C_i, then the probability mass function of $\boldsymbol{v}$ can be computed as

$$P = P_1 * P_2 * \cdots * P_I \tag{98}$$

$$= \mathcal{F}^{-1}\left(\mathcal{F}_i(P_1) \circ \mathcal{F}_2(P_2) \circ \cdots \circ \mathcal{F}_I(P_I)\right) \tag{99}$$

If the convolution is computed naively according to (98), the time complexity is $O(IR^2)$. However, if FFT is adopted per (99), the complexity will be reduced to $O(IR \log R)$.

Backpropagation When fast Fourier transform is used to accelerate additions in Bayesian quantized networks, we need to derive the corresponding backpropagation rule, i.e. equations that relate $\partial\mathcal{L}/\partial P$ to $\{\partial\mathcal{L}/\partial P_i\}_{i=1}^{I}$. For this purpose, we break the computation in Equation (99) into three steps, and compute the derivative for each of these steps.

$$C_i = \mathcal{F}_i(P_i) \Longrightarrow \frac{\partial \mathcal{L}}{\partial P_i} = R \cdot \mathcal{F}_i^{-1}\left(\frac{\partial \mathcal{L}}{\partial C_i}\right) \tag{100a}$$

$$C = C_1 \circ \cdots \circ C_I \Longrightarrow \frac{\partial \mathcal{L}}{\partial C_i} = \frac{C}{C_i} \circ \frac{\partial \mathcal{L}}{\partial C} \tag{100b}$$

$$P = \mathcal{F}^{-1}(C) \Longrightarrow \frac{\partial \mathcal{L}}{\partial C} = R^{-1} \cdot \mathcal{F}\left(\frac{\partial \mathcal{L}}{\partial P}\right) \tag{100c}$$

where in (100b) we use C/C_i to denote element-wise division. Since P_i lies into real domain, we need to project the gradients back to real number in (100c). Putting all steps together:

$$\frac{\partial \mathcal{L}}{\partial P_i} = \Re\left\{\mathcal{F}_i^{-1}\left(\frac{C}{C_i} \circ \mathcal{F}\left(\frac{\partial \mathcal{L}}{\partial P}\right)\right)\right\}, \forall i \in [I] \tag{101}$$

D.3 Lyapunov Central Limit Approximation (Proof of Theorem 4.3)

In this part, we show that Lyapunov central limit approximation accelerates probabilistic propagation (therefore also probabilistic backpropagation) in linear layers. For simplicity, we only consider *fully-connected layer* here, but the derived algorithm equally applies to other linear layers, including various types of *convolutional layer*. Again, to use the algorithm in neural networks, we will need to derive the corresponding backpropagation equation for each step in the forward pass.

Proof of Equations (102a) **and** (102b)

$$\tilde{\mu}_j^{(l+1)} = \sum_{i \in \mathcal{I}(j)} m_{ij}^{(l)} \mu_i^{(l)} \tag{102a}$$

$$\tilde{\nu}_j^{(l+1)} = \sum_{i \in \mathcal{I}(j)} \left({m_{ji}^{(l)}}^2 \nu_i^{(l)} + {\mu_i^{(l)}}^2 v_{ji} + \nu_i^{(l)} v_{ji} \right) \tag{102b}$$

Recall that we have defined the mean and variance of $\boldsymbol{h}_i^{(l)}$ and $\boldsymbol{\theta}_{ji}^{(l)}$ as:

$$\mathbb{E}\left[\boldsymbol{h}_i^{(l)} | x_n\right] = \mu_i^{(l)}, \quad \mathbb{E}\left[\boldsymbol{\theta}_{ji}^{(l)}\right] = m_{ji}^{(l)} \tag{103a}$$

$$\mathbb{V}\left[\boldsymbol{h}_i^{(l)} | x_n\right] = \nu_i^{(l)}, \quad \mathbb{V}\left[\boldsymbol{\theta}_{ji}^{(l)}\right] = v_{ji}^{(l)} \tag{103b}$$

With these notations, the Equations can be proved directly by definitions:

$$\tilde{\mu}_j^{(l+1)} = \mathbb{E}\left[\sum_{i=1}^{I} \boldsymbol{\theta}_{ji}^{(l)}\, \boldsymbol{h}_i^{(l)}\right] = \sum_{i=1}^{I} \mathbb{E}\left[\boldsymbol{\theta}_{ji}^{(l)}\, \boldsymbol{h}_i^{(l)}\right] \tag{104}$$

$$= \sum_{i=1}^{I} \mathbb{E}\left[\boldsymbol{\theta}_{ji}^{(l)}\right] \mathbb{E}\left[\boldsymbol{h}_i^{(l)}\right] = \sum_{i=1}^{I} m_{ji}^{(l)} \mu_i^{(l)} \tag{105}$$

$$\tilde{\nu}_j^{(l+1)} = \mathbb{V}\left[\sum_{i=1}^{I} \boldsymbol{\theta}_{ji}^{(l)}\, \boldsymbol{h}_i^{(l)}\right] = \sum_{i=1}^{I} \mathbb{V}\left[\boldsymbol{\theta}_{ji}^{(l)}\, \boldsymbol{h}_i^{(l)}\right] \tag{106}$$

$$= \sum_{i=1}^{I} \left(\mathbb{E}\left[{\boldsymbol{\theta}_{ji}^{(l)}}^2\right] \mathbb{E}\left[{\boldsymbol{h}_i^{(l)}}^2\right] - \mathbb{E}\left[\boldsymbol{\theta}_{ji}^{(l)}\right]^2 \mathbb{E}\left[\boldsymbol{h}_i^{(l)}\right]^2\right) \tag{107}$$

$$= \sum_{i=1}^{I} \left[\left({m_{ji}^{(l)}}^2 + v_i^{(l)}\right)\left({\mu_i^{(l)}}^2 + \nu_{ji}^{(l)}\right) - {m_{ji}^{(l)}}^2 {\mu_i^{(l)}}^2\right] \tag{108}$$

$$= \sum_{i=1}^{I} \left({m_{ji}^{(l)}}^2 \nu_i^{(l)} + v_{ji}^{(l)} {\mu_i^{(l)}}^2 + v_{ji}^{(l)} \nu_i^{(l)}\right) \tag{109}$$

For fully-connected layers, these two equations can be concisely written in matrix forms:

$$\tilde{\mu}^{(l+1)} = M^{(l)} \mu^{(l)} \tag{110a}$$

$$\tilde{\nu}^{(l+1)} = \left({M^{(l)}}^{\circ 2}\right) \nu^{(l)} + V^{(l)} \left({\mu^{(l)}}^{\circ 2} + \nu^{(l)}\right) \tag{110b}$$

Backpropagation With matrix forms, the backpropagation rule that relates $\partial\mathcal{L}/\partial\tilde{\psi}^{(l+1)} = \{\partial\mathcal{L}/\partial\tilde{\mu}^{(l+1)}, \partial\mathcal{L}/\partial\tilde{\nu}^{(l+1)}\}$ to $\partial\mathcal{L}/\partial\phi^{(l)} = \{\partial\mathcal{L}/\partial M^{(l)}, \partial\mathcal{L}/\partial V^{(l)}\}$ and $\partial\mathcal{L}/\partial\psi^{(l)} = \{\partial\mathcal{L}/\partial\mu^{(l)}, \partial\mathcal{L}/\partial\nu^{(l)}\}$ can be derived with matrix calculus.

$$\frac{\partial\mathcal{L}}{\partial M^{(l)}} = \left(\frac{\partial\mathcal{L}}{\partial\tilde{\mu}^{(l+1)}}\right) {\mu^{(l)}}^\top + 2M^{(l)} \circ \left[\left(\frac{\partial\mathcal{L}}{\partial\tilde{\nu}^{(l+1)}}\right) {\nu^{(l)}}^\top\right] \tag{111a}$$

$$\frac{\partial\mathcal{L}}{\partial V^{(l)}} = \left(\frac{\partial\mathcal{L}}{\partial\tilde{\nu}^{(l+1)}}\right) \left({\mu^{(l)}}^{\circ 2}\right)^\top \tag{111b}$$

$$\frac{\partial\mathcal{L}}{\partial\mu^{(l)}} = {M^{(l)}}^\top \left(\frac{\partial\mathcal{L}}{\partial\tilde{\mu}^{(l+1)}}\right) + 2\mu^{(l)} \circ \left[{V^{(l)}}^\top \left(\frac{\partial\mathcal{L}}{\partial\tilde{\nu}^{(l+1)}}\right)\right] \tag{111c}$$

$$\frac{\partial\mathcal{L}}{\partial\nu^{(l)}} = \left({M^{(l)}}^{\circ 2}\right)^\top \left(\frac{\partial\mathcal{L}}{\partial\tilde{\nu}^{(l+1)}}\right) \tag{111d}$$

Notice these equations do not take into account the fact that $V^{(l)}$ implicitly relates to $M^{(l)}$ (i.e. $v_{ji}^{(l)}$ is defined upon $m_{ji}^{(l)}$). Therefore, we need to adjust the backpropagation rule for the probabilities: denote $Q_{ji}^{(l)}(d) = Q(\boldsymbol{\theta}_{ji}^{(l)} = \mathbb{Q}(d); \phi_{ji}^{(l)})$.

$$\frac{\partial\mathcal{L}}{\partial Q^{(l)}(d)} = \left(\frac{\partial\mathcal{L}}{\partial M^{(l)}} + \frac{\partial\mathcal{L}}{\partial V^{(l)}} \cdot \frac{\partial V^{(l)}}{\partial M^{(l)}}\right) \frac{\partial M^{(l)}}{\partial Q^{(l)}(d)} + \frac{\partial\mathcal{L}}{\partial V^{(l)}} \cdot \frac{\partial\nu}{\partial Q^{(l)}(d)} \tag{112}$$

$$= \mathbb{Q}(d) \cdot \frac{\partial\mathcal{L}}{\partial M^{(l)}} + 2(\mathbb{Q}(d) - M^{(l)}) \circ \left(\frac{\partial\mathcal{L}}{\partial V^{(l)}}\right) \tag{113}$$

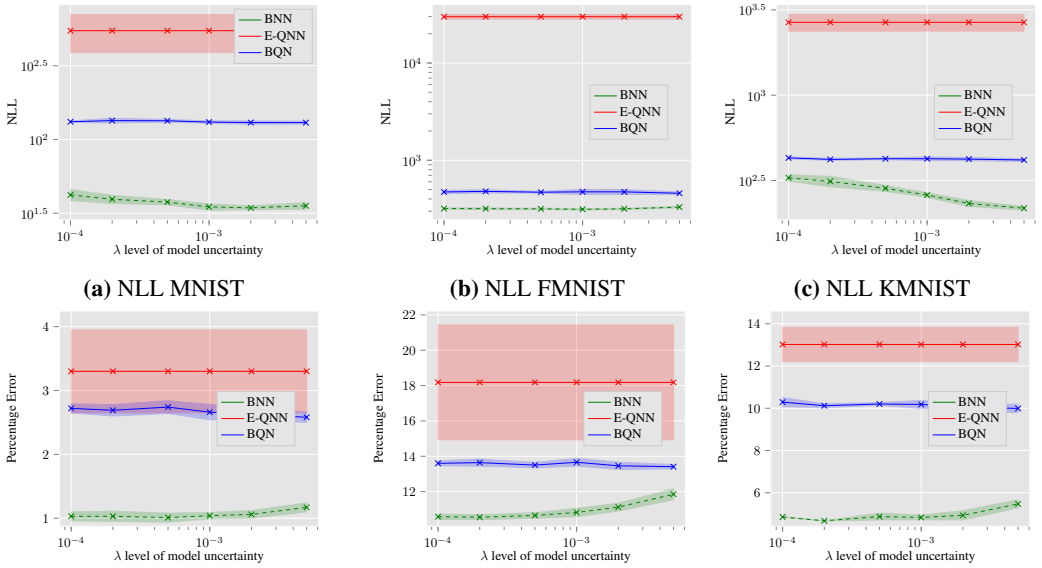
Lastly, we derive the backpropagation rules for binary neural networks with sign activations. Let $p_j^{(l+1)}$ denote the probability that the hidden unit $\boldsymbol{h}_j^{(l+1)}$ is "activated" $\mathbf{Pr}[\boldsymbol{h}_j^{(l+1)} = 1|x]$, $\partial\mathcal{L}/p_j^{(l+1)}$

relates to $\{\partial\mathcal{L}/\partial\tilde{\mu}_j^{(l+1)}, \partial\mathcal{L}/\partial\tilde{\nu}_j^{(l+1)}\}$ as:

$$\frac{\partial p_j^{(l+1)}}{\partial\tilde{\mu}_j^{(l+1)}} = \mathcal{N}\left(\frac{\tilde{\mu}_j^{(l+1)}}{\sqrt{\tilde{\nu}_j^{(l+1)}}}\right) \tag{114a}$$

$$\frac{\partial p_j^{(l+1)}}{\partial\tilde{\nu}_j^{(l+1)}} = -\frac{1}{2\left[\tilde{\nu}_j^{(l+1)}\right]^{3/2}} \cdot \mathcal{N}\left(\frac{\tilde{\mu}_j^{(l+1)}}{\sqrt{\tilde{\nu}_j^{(l+1)}}}\right) \tag{114b}$$

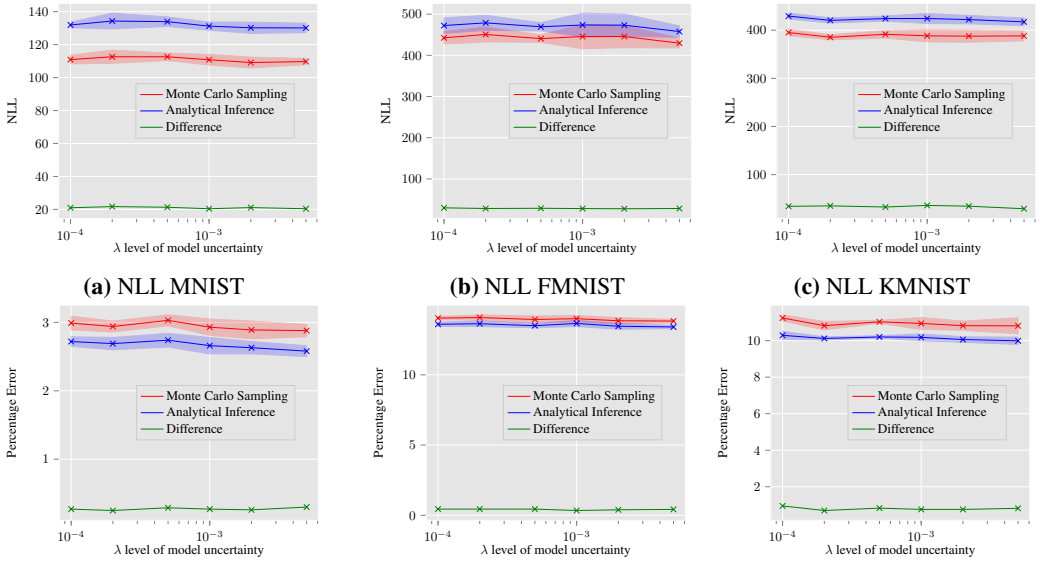
E EXPERIMENTAL RESULTS ON FULLY-CONNECTED NEURAL NETWORKS



(a) NLL MNIST **(b)** NLL FMNIST **(c)** NLL KMNIST

(d) Error MNIST **(e)** Error FMNIST **(f)** Error KMNIST

Figure 4: Comparison of the predictive performance of our BQNs against the E-QNN as well as the non-quantized BNN trained by SGVB on a MLP. Negative log-likelihood (NLL) which accounts for uncertainty and 0-1 test error which doesn't account for uncertainty are displayed.



(a) NLL MNIST **(b)** NLL FMNIST **(c)** NLL KMNIST

(d) Error MNIST **(e)** Error FMNIST **(f)** Error KMNIST

Figure 5: Illustration of mean-field approximation and tightness of alternative ELBO on a MLP. The performance gap between our analytical inference and the Monte Carlo Sampling is displayed.