On the Effects of Quantisation on Model Uncertainty in Bayesian Neural Networks

Martin Ferianc[1] **Partha Maji**[2] **Matthew Mattina**[3] **Miguel Rodrigues**[1]

[1]Electronic and Electrical Engineering Department, University College London, London, UK
[2]Arm ML Research Lab, Cambridge, UK
[3]Arm ML Research Lab, Boston, USA

Abstract

Bayesian neural networks (BNNs) are making significant progress in many research areas where decision-making needs to be accompanied by uncertainty estimation. Being able to quantify uncertainty while making decisions is essential for understanding when the model is over-/under-confident, and hence BNNs are attracting interest in safety-critical applications, such as autonomous driving, healthcare, and robotics. Nevertheless, BNNs have not been as widely used in industrial practice, mainly because of their increased memory and compute costs. In this work, we investigate quantisation of BNNs by compressing 32-bit floating-point weights and activations to their integer counterparts, that has already been successful in reducing the compute demand in standard pointwise neural networks. We study three types of quantised BNNs, we evaluate them under a wide range of different settings, and we empirically demonstrate that a uniform quantisation scheme applied to BNNs does not substantially decrease their quality of uncertainty estimation.

1 INTRODUCTION

Bayesian neural networks (BNNs) can describe complex stochastic patterns by treating their weights as learnable random variables that provide well-calibrated uncertainty estimates (Neal, 1993; Ghahramani, 2015; Blundell et al., 2015; Gal and Ghahramani, 2015; Chen et al., 2014). In addition to modelling uncertainty, by treating a neural network (NN) through Bayesian inference, it gains robustness to over-fitting thereby offering the means to leverage small data pools (Ghahramani, 2015).

BNNs have become relevant in practical applications where the quantification of uncertainty is essential such as in medicine (Liang et al., 2018), autonomous driving (McAllister et al., 2017) or risk assessment (MacKay, 1995). Nevertheless, Bayesian models come with a prohibitive computational cost during evaluation (Gal and Ghahramani, 2015; Blundell et al., 2015). While evaluating, it is analytically intractable to compute the posterior prediction. Hence, most methods approximate the posterior through Monte Carlo (MC) sampling (Gal and Ghahramani, 2015; Blundell et al., 2015; Chen et al., 2014), which depends on multiple feed-forward runs through the BNNs and optionally random number generation.

In contrast to pointwise NNs, that are increasingly used for applications on the edge, the computational cost associated with BNNs currently prevents their use on resource-constrained platforms. These platforms exhibit smaller memory and lower compute capabilities involving 8-bit integer arithmetic. Quantisation has been widely used in pointwise NNs (Jacob et al., 2018; Choukroun et al., 2019; Krishnamoorthi, 2018) to lower their compute demand and make them more compatible with edge devices. In quantisation, floating-point representation is reduced to an integer representation, which enables substantial resource savings in practical applications. By quantising weights and activations of pointwise NNs to 8-bit integers, it is possible to achieve up to $4\times$ improvements in latency with a quarter of the original memory footprint of the baseline 32-bit floating-point implementation (Guo et al., 2017b; Jacob et al., 2018). Nevertheless, there has not been a comprehensive study into whether BNNs could attain the same hardware benefits under quantisation and whether it impacts their predictive accuracy or uncertainty.

In this work, we study quantisation of BNNs based on three widely adapted Bayesian inference schemes: Monte Carlo Dropout (Gal and Ghahramani, 2015), Bayes-By-Backprop (Blundell et al., 2015) and Stochastic Gradient Langevin Dynamics with Hamiltonian Monte Carlo (Chen et al., 2014). Furthermore, we investigate the effect of quantisation of both weights and activations of BNNs using different integer representations through quantisation aware

Accepted for the 37th Conference on Uncertainty in Artificial Intelligence (UAI 2021).

training. Our main contributions are two-fold *1)* Methodology for uniform quantisation of three different types of Bayesian inference; *2)* An empirical demonstration that lowering arithmetic precision of weights and activations from 32-bit floating-point to $\leq$8-bit integers does not substantially detriment accuracy and uncertainty estimation quality of Bayesian neural networks across different datasets, network architectures and tasks. To the best of our knowledge, we are the first ones to attempt an empirical investigation in this direction with respect to widely compared and accessible benchmarks. The code is available at `https://git.io/JtSJG`.

2 PRELIMINARIES AND RELATED WORK

In this Section we review Bayesian learning, quantisation of neural networks and related work.

2.1 BAYESIAN NEURAL NETWORKS

The aim of Bayesian inference is to learn the distribution over the weights of the BNN $\boldsymbol{w}$ with respect to some training dataset of tuples $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)_{n=1}^N\}$, where $\boldsymbol{x}_n$ are the inputs and $\boldsymbol{y}_n$ are the associated targets. Given the belief about the noise in the data in the shape of the likelihood $p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})$ and the prior distribution over weights $p(\boldsymbol{w})$, they come together under the Bayes rule $p(\boldsymbol{w}|\boldsymbol{x}, \boldsymbol{y}) = \frac{p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|\boldsymbol{x})}$. Nevertheless, due to the high dimensionality of BNN it is intractable to compute the posterior $p(\boldsymbol{w}|\boldsymbol{x}, \boldsymbol{y})$ and it needs to be approximated with respect to $q(\boldsymbol{w}|\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$ and some learnable parameters $\boldsymbol{\theta}$. The resultant distribution $q(.)$ can then be used to make predictions for previously unseen data $\boldsymbol{x}^*, \boldsymbol{y}^*$ through an integral $p(\boldsymbol{y}^*|\boldsymbol{x}^*) = \int p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w})q(\boldsymbol{w}|\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})d\boldsymbol{w}$. This integral is again intractable due to the posterior and it needs to be approximated through through MC sampling with L samples as: $p(\boldsymbol{y}^*|\boldsymbol{x}^*) = \frac{1}{L}\sum_{l=1}^{L} p(\boldsymbol{y}^*|\boldsymbol{x}^*, \boldsymbol{w}_l); \boldsymbol{w}_l \sim q(\boldsymbol{w}|\boldsymbol{\theta}, \boldsymbol{x}, \boldsymbol{y})$. The sampling procedure requires efficient processing to reduce the compute cost of the forward pass through the BNN L times. In this work, we approach this challenge through investigating quantisation applied to BNNs' weights and activations in order to enable their efficient processing.

2.2 QUANTISATION

Reduction in bit-width precision (Jacob et al., 2018; Krishnamoorthi, 2018; Choukroun et al., 2019) has demonstrated significant benefits in lowering the resource consumption of pointwise NNs in hardware. In quantisation, 32-bit floating-point representations of weights, and optionally, activations are reduced to an integer, usually 8-bit representation, which enables substantial savings in memory and compute resources in real-world applications. This helps to reduce energy consumption and improve inference speed. If the quantisation happens after training, it is called post-training quantisation. If it happens with an additional training with fewer iterations and much smaller learning rate after the main portion of the inference, it is called quantisation aware training (QAT) (Jacob et al., 2018). By using QAT, practitioners have observed smaller accuracy drop in the quantised model (Jacob et al., 2018), compared to post-training quantisation. The support of only integer arithmetic in hardware has two main outcomes: (1) decrease in size of the required memory and the complexity of the hardware to perform the computation; (2) decrease in latency due to the simplicity of integer computation, in comparison to floating-point (Cai et al., 2018). These benefits present a strong case for investigating quantisation of BNNs.

2.3 RELATED WORK

Only recently, there appeared works outside of the realm of pointwise NNs that interconnected Bayesian thinking with quantisation (Su et al., 2019; Achterhold et al., 2018; Cai et al., 2018; van Baalen et al., 2020).

Achterhold et al. (2018) developed a sophisticated method for quantisation and pruning for pointwise NNs, albeit by using Bayesian inference. They initially train a BNN with improper priors, constructed to be quantisation and pruning-friendly, and after training, convert it to a quantised pointwise NN. Although, the pointwise NNs can achieve significant reduction in memory consumption, the resultant non-quantised BNNs are actually unable to estimate uncertainty, due to improper priors (Hron et al., 2017). Similarly, van Baalen et al. (2020) used Bayesian inference to obtain sparse quantised pointwise NNs. In VIBNN, Cai et al. (2018) developed an efficient hardware accelerator for feed-forward BNNs trained through Bayes-by-backprop (Blundell et al., 2015) algorithm. The authors demonstrated impressive compute resource savings, but they did not detail their quantisation scheme or its impact on the uncertainty estimation capabilities of the BNN. Su et al. (2019) proposed a method for learning quantised BNNs directly, where the range of the found activations and weights is limited to two integer values. They demonstrated that the uncertainty estimation can be preserved in the learned model. However, the work of Su et al. (2019) does not allow quantisation of modern networks involving batch normalisation and skip-connections (e.g. ResNet). Additionally, in their scheme binary weights (-1, 1) need to be stored as real-valued parameters. Furthermore, their scheme in practice would require development of a custom hardware accelerator. Custom hardware accelerators are rarely used in real-world settings. Most emerging NPUs are optimised for fixed 8-bit integer arithmetic only. For existing low-resource scenarios in embedded and IoT applications CPUs are optimised for 8-bit arithmetic.

In this paper we propose to learn quantised BNNs directly –

as in (Su et al., 2019). In contrast to their work, we consider a range of widely used Bayesian inference methods, without the need for changes in the method or architectures. In detail, we focus on uniform quantisation, that is commonly supported in hardware (NPU, TPU, GPU) (Krishnamoorthi, 2018).

3 METHODOLOGY

In this Section we describe quantised BNNs, by first discussing the theory behind quantisation followed by its applicability to the respective Bayesian inference methods.

3.1 UNIFORM AFFINE QUANTISATION

The most light-weight quantisation method is an uniform affine mapping of 32-bit floating-point values f to integers q (Jacob et al., 2018) as shown in (1):

$$f = S(q - Z) \tag{1}$$

where S and Z are the scale and the zero-point respectively, which are learnable parameters. The S remains in floating-point representation and it effectively represents a quantisation bin-width, whereas the Z is an integer of the same bit-width n as q and it represents the mapping of the value 0. The values of S and Z are affected by the target n, which restricts their range.

Assuming initially a standard pointwise linear layer with floating-point weights $\boldsymbol{f}_w \in \mathbb{R}^{M\times F}$, input $\boldsymbol{f}_i \in \mathbb{R}^{I\times M}$ and output $\boldsymbol{f}_o \in \mathbb{R}^{I\times F}$, where M and F correspond to the input and output feature size for a batch consisting of I samples, the computation for their quantised counterparts $\boldsymbol{q}_w, \boldsymbol{q}_i, \boldsymbol{q}_o$ is obtained with respect to (1) as follows: Linear output without quantisation is computed as $\boldsymbol{f}_o = \boldsymbol{f}_i\boldsymbol{f}_w$. Substituting each term with (1), we have $S_o(\boldsymbol{q}_o - Z_o) = S_w(\boldsymbol{q}_w - Z_w)S_i(\boldsymbol{q}_i - Z_i)$ which can be rewritten as in (2):

$$\boldsymbol{q}_o = Z_o + \frac{S_w S_i}{S_o}(MZ_wZ_i - Z_i\sum \boldsymbol{q}_w - Z_w\sum q_i + \boldsymbol{q}_w\boldsymbol{q}_i) \tag{2}$$

The respective sums are performed first for each column for $\boldsymbol{q}_w$ and each row $\boldsymbol{q}_i$ and broadcast to the resultant matrix dimension, similarly to scalars S and Z. Note that, the terms not involved with any $\boldsymbol{q}_i$ are independent of the input, which means they can be computed offline. Similarly, if the layer has a bias term, or it is followed by a batch normalisation (BN) (Ioffe and Szegedy, 2015), the BN affine parameters or bias can be fused into the weights after the individual S and Z have been inferred (Krishnamoorthi, 2018). The same pattern can then be used to compute the output of more complicated operations, such as convolutions (Jacob et al., 2018). Note that, the bit-width n does not need to be the same for weights and activations.

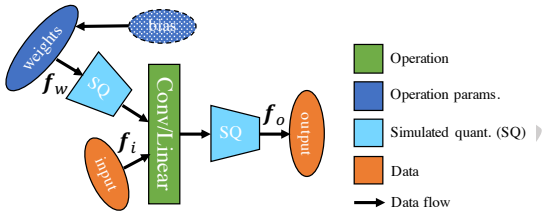


Figure 1: Fine-tuning of a standard pointwise Convolution/Linear layer with simulated quantisation (SQ). All computation is carried out using 32-bit floating-point arithmetic. SQ nodes are injected into the computation to simulate the effects of quantisation. After fine-tuning, the SQ modules are removed and the weights, with folded in bias, and computation are quantised. In a quantised regime, floating-point data $\boldsymbol{f}$ are replaced by $\boldsymbol{q}$.

The scale (S) and the zero-point (Z) parameters are learned by simulating quantisation through fine-tuning resulting in quantisation aware training (QAT). In this work we focus on QAT, which has been preferred to post-training quantisation since it is shown that it achieves higher accuracy, especially in smaller models (Jacob et al., 2018). In the next Section we introduce QAT-based methods applied to NNs with respect to Bayesian inference.

3.1.1 Quantisation Aware Training (QAT)

QAT is achieved by simulating quantisation effects in the forward pass of training, while backpropagation and all weights are represented in floating-point (Jacob et al., 2018). The simulation is achieved by implementing rounding behaviour, that can be hardware platform specific, while performing floating-point arithmetic and then using a straight through gradient estimator (Bengio et al., 2013) in the backward pass.

- Weights' quantisation is simulated prior to being combined with the input, to avoid dynamic quantisation during runtime.
- Activation or operation output quantisation is simulated at points where they would be during inference - after the activation function is applied or after addition or concatenation of outputs of several layers as in ResNets (Jacob et al., 2018; He et al., 2016).

Concretely in this work, we adopt the element-wise quantisation function as shown by Jacob et al. (2018) for all tensors individually, and we assume hardware fusion of the common ReLU activation, BN and bias into the operation as done in practice (Krishnamoorthi, 2018). The quantisation and its simulation are parametrised by n, which is user specified, and a clamping range consisting of a minimum $a; a = \min \boldsymbol{f}$ and a maximum $b; b = \max \boldsymbol{f}$ for the given tensor. Individual a and b are being observed on the train-

ing and validation datasets, for each activation output and weight. To observe the most efficient clamping range bounds a, b, it is necessary to record the minimum and maximum values of the respective tensors during training and then individually aggregate them via exponential moving average, because of perturbations in outputs and weights due to QAT fine-tuning. The a, b, n continually map to $S; S = \frac{b-a}{n-1}$ and $Z; Z = \text{round}(\min \frac{\boldsymbol{f}}{S})$ that are being used for the simulation and the end values are then used for the actual quantisation, following equation (1). The computational graph with respect to QAT is visually represented in Figure 1 and in pseudo code in Algorithm 1. In the next Section we describe how this scheme can be used to obtain quantised BNNs.

Algorithm 1 Quantisation Aware Training

1: Inference of a floating-point model until convergence.
2: Fusion of biases and batch normalisation statistics with weights
3: Insertion of simulated quantisation (SQ) modules after weights and operations' outputs.
4: Fine-tuning, simulating quantisation and recording individual a, b per tensor in the computational graph.
5: Computation of individual S and Z, quantisation of weights and computation of offline constants to prepare the model for integer arithmetic evaluation.

3.2 QUANTISED BAYESIAN NEURAL NETWORKS

In this work we develop schemes for performing quantisation aware training (QAT) for Bayesian inference methods for both their weights and activation outputs. Note that, we propose to use QAT exclusively after Bayesian inference, and with minimal fine-tuning, such that the parameters learned through the Bayesian inference are not compromised. We illustrate the quantisation process with the help of a linear layer and notation from Section 3.1. In general, it is necessary to only discuss the placement of SQ nodes with respect to the compute graphs and step 2. from Algorithm 1 for the respective Bayesian inference methods, following the rules introduced in the bullet-points in the previous Section. Other steps are exactly the same as for the pointwise counterpart.

3.2.1 Monte Carlo Dropout (MCD)

The quantisation of MCD in shown in Figure 2. We propose methodology for quantisation of the standard MCD implementation (Gal and Ghahramani, 2015), that corresponds to applying Bernoulli mask of zeros and ones $\boldsymbol{K} \in \mathbb{R}^{I \times M}$ with respect to an input with M features and I samples for each weight-bearing layer, except the input. Additionally the masked input is scaled by the proportion of zeros in the

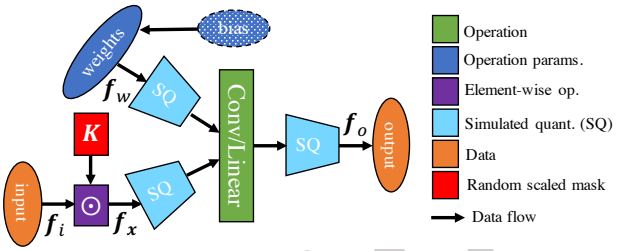


Figure 2: Quantisation for Monte Carlo Dropout.

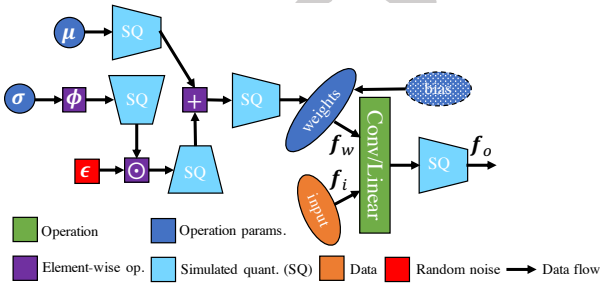
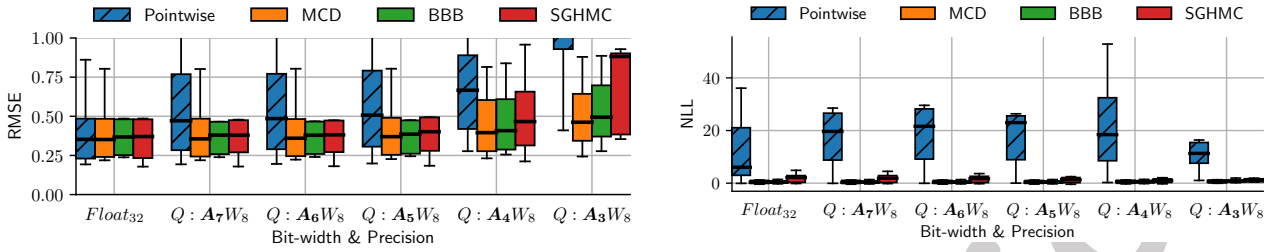


Figure 3: Quantisation for Bayes-by-Backprop.

mask, such that $\boldsymbol{f}_x = \boldsymbol{K} \odot \boldsymbol{f}_i \frac{1}{1-p}$ and $\boldsymbol{q}_x = \boldsymbol{K} \odot \boldsymbol{q}_i \frac{1}{1-p}$, where p is the probability of sampling zero and $\odot$ is an element-wise multiplication. Thus $\boldsymbol{f}_x, \boldsymbol{q}_x$ values replace $\boldsymbol{f}_i, \boldsymbol{q}_i$ values with respect to equation (2). We add separate SQ node to the multiplication of the $\boldsymbol{K}$ with the input, since due to the factor $\frac{1}{1-p}$ and zeroing-out some inputs, the respective S and Z will change. When generating the $\boldsymbol{K}$, we absorb the $\frac{1}{1-p}$ into the mask for efficient computation. Thus, $\boldsymbol{K}$ does not include optimisable parameters and SQ is not directly needed after $\boldsymbol{K}$. Note that during performing QAT it is necessary to generate the mask in floating-point, while in a quantised mode the $\boldsymbol{K}$ needs to take into account the Z of $\boldsymbol{q}_i$. Weights are simply quantised according to equation (1) and by adding an SQ node as discussed in the Section 3.1.1.

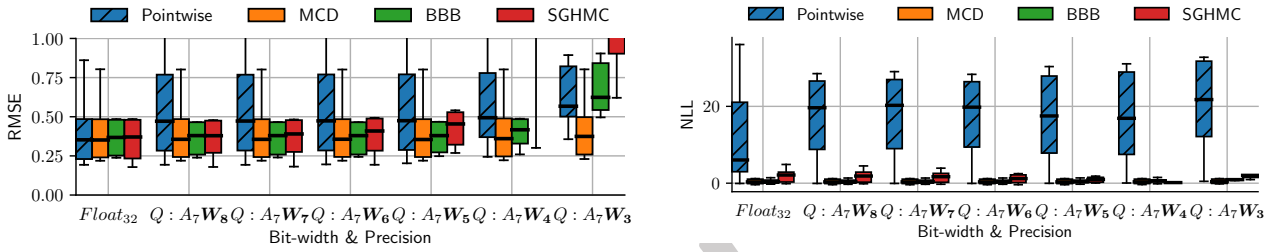
3.2.2 Bayes-By-Backprop (BBB)

We propose QAT methodology for BBB as shown in Figure 3. In BBB (Blundell et al., 2015), the distribution over the weights is modelled explicitly such that $\boldsymbol{f}_w, \boldsymbol{q}_w \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$, with mean $\boldsymbol{\mu} \in \mathbb{R}^{M \times F}$ and variance $\boldsymbol{\sigma}^2 \in \mathbb{R}^{M \times F}$ for each weight with respect to M input and F output features and $\mathcal{N}$ represents a Gaussian. Nevertheless, to enable backpropagation and efficient computation of the weights, the weights are sampled with respect to a Gaussian $\boldsymbol{\epsilon} \sim \mathcal{N}(0, 1)$, such that $\boldsymbol{f}_w = \boldsymbol{\mu} + \phi(\boldsymbol{\sigma}) \odot \boldsymbol{\epsilon}$ (Kingma and Welling, 2013). $\phi(.)$ constrains the output to be positive e.g.: softplus. It is necessary to add SQ nodes and observe the statistics after each operation: application of positive element-wise $\phi(.)$, addition and multiplication to obtain $\boldsymbol{f}_w$ and subsequently $\boldsymbol{q}_w$. We simulate quantisation to compute



(a) (b)

Figure 4: *Changing activation precision, fixing weight precision.* Regression results with respect to root-mean-squared error (RMSE) (a) and negative log-likelihood (NLL) (b) on UCI datasets. Q stands for quantised activations (A) and weights (W). Subscript denotes bit-width. Pointwise and SGHMC collapse when the bit-width ≤ 3 for A.



(a) (b)

Figure 5: *Fixing activation precision, changing weight precision.* Regression results with respect to root-mean-squared error (RMSE) (a) and negative log-likelihood (NLL) (b) on UCI datasets. Q stands for quantised activations (A) and weights (W). Subscript denotes bit-width. SGHMC collapses when the bit-width ≤ 4 for W.

the quantisation statistics for the means $\boldsymbol{\mu}$ as well as the positive standard deviation $\phi(\boldsymbol{\sigma})$. We do this to avoid dynamic quantisation during run-time. The quantisation for the standard deviation is performed after $\phi(.)$, which eventually bypasses the non-linearity, when quantised, and reduces the numerical errors induced by the reduced representation. Practically, this means that we do not have to perform $\phi(.)$ and there is no floating-point computation at runtime. Note that, depending on the regime, it is necessary generate $\boldsymbol{\epsilon}$ in floating-point or quantised. We found quantised $\boldsymbol{\epsilon}$ with respect to $S_\epsilon = 0.0236$ and $Z_\epsilon = 0$ to be performing well across different n and experiments. Due to the proposed scheme, there are no changes necessary to be made with respect to equation (2). We avoid the computation of the gradients with respect to the ELBO's regulariser (Blundell et al., 2015) during QAT. We found it practically difficult to perform the quantisation with respect to the non-linear computation of KL divergence (taking log, square, division).

3.2.3 Stochastic Gradient Langevin Dynamics with Hamiltonian Monte Carlo (SGHMC)

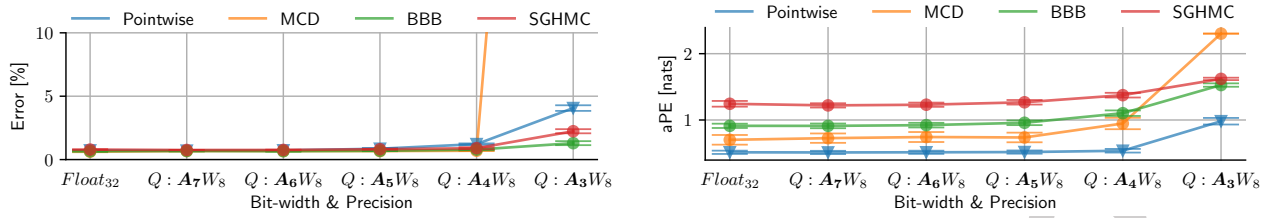
In comparison to the previous two methods, in SGHMC (Chen et al., 2014), there is no sampling of random variables during evaluation. Chen et al. (2014), following (Welling and Teh, 2011), demonstrated that by adding the right amount of Gaussian noise to a standard stochastic gradient optimisation algorithm, it is possible to collect weights $\boldsymbol{w}_l$ from $l = 1, \ldots, L$ several distinct optimisation steps, that can then be used to approximate the samples from the true posterior distribution over the BNN weights. Therefore, we propose to quantise each of the weight samples separately through QAT, similarly to a pointwise approach, as shown in Figure 1. The SQ nodes are applied to each set of weight samples l as well as the corresponding outputs. Thus, we propose to fine-tune each pre-trained network sample l separately.

In all instances, QAT was used with a very small learning rate for 10 epochs. However, it is crucial that Bayesian inference was adhered to in the main phase with a bigger learning rate and substantially more epochs.

4 EXPERIMENTS

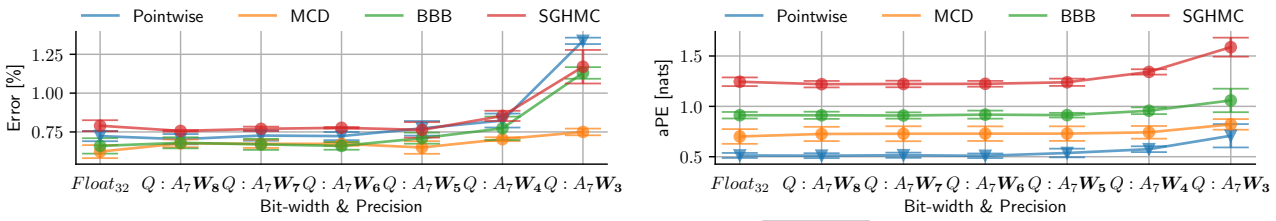
In this Section we present the tasks, datasets and their corresponding NN architectures, metrics and the implementation, followed by the observations.

We consider two classes of problems: *1)* regression and *2)* classification. We evaluate the networks on sample datasets of tuples $\mathcal{D}$. For regression the target y_n is assumed to be a



(a) (b)

Figure 6: *Changing activation precision, fixing weight precision.* MNIST results with respect to classification error on test data (a) and average predictive entropy (aPE) on FashionMNIST (b). Q stands for quantised activations (A) and weights (W). Subscript denotes bit-width. MCD collapses when the bit-width ≤ 3 for A.



(a) (b)

Figure 7: *Fixing activation precision, changing weight precision.* MNIST results with respect to classification error on test data (a) and average predictive entropy (aPE) on FashionMNIST (b). Q stands for quantised activations (A) and weights (W). Subscript denotes bit-width.

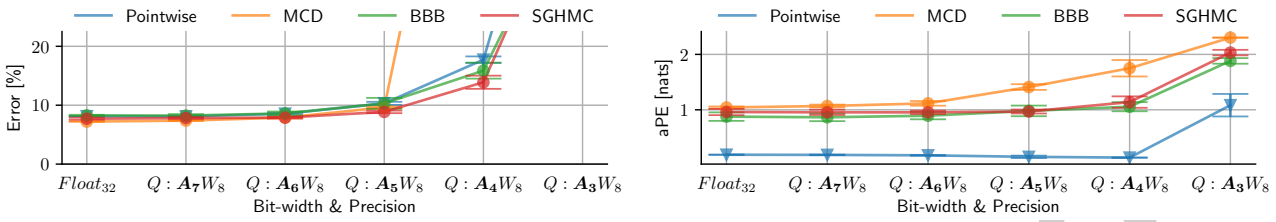
real-valued $y_n \in \mathbb{R}^1$, while for classification the target $\boldsymbol{y}_n$ is a one-hot encoding of $k = 1, \ldots, K$ classes such that $\boldsymbol{y}_n \in \mathbb{R}^K$. Given the input features $\boldsymbol{x}_n$, we use a BNN to model the probabilistic predictive distribution $p_w(y_n|\boldsymbol{x}_n)$ over the targets with respect to some model defined by weights $\boldsymbol{w}$, where the mean and the variance are approximated with respect to L samples as $\mu_w(\boldsymbol{x}_n) = \mathbb{E}[\frac{1}{L}\sum_{l=1}^{L} p_{w_l}(y_n|\boldsymbol{x}_n)]$ and $\sigma_w^2(\boldsymbol{x}_n) = \mathbb{E}[\frac{1}{L}\sum_{l=1}^{L}(p_{w_l}(y_n|\boldsymbol{x}_n) - \mu_w(\boldsymbol{x}_n))^2]$.

For the regression we consider UCI datasets (housing, concrete, energy, power, wine, yacht) whereas for classification we consider classifying MNIST digits and CIFAR-10 image datasets. We used a mixture of real data to control the complexity of the experiments and observe whether it affects the uncertainty estimation quality in a quantised regime. For the regression problem we consider an architecture with an input layer followed by 3 hidden layers with 100 nodes, each followed by a ReLU activation. For MNIST we implement the common LeNet-5 (LeCun et al., 1998), while for CIFAR-10 we implement ResNet-18 (He et al., 2016) with BN and skip-connections enabled. Similarly to the datasets, we chose NN architectures of increasing complexity to explore how the uncertainty estimation is impacted by trailing quantisation errors coming from a reduced precision and deeper architectures. We considered image augmentations: rotation, brightness and horizontal shift and confusion datasets: FashionMNIST for MNIST and SVHN for CIFAR-10 experiments to measure the level of uncertainty on distant or shifted datasets. The hyperparameters for all experiments were hand-tuned with reference to validation error.

From the quantisation point of view, we focus on quantisation of both weights and activations to improve on-device storage as well as computational efficiency. We considered $3 \leq n \leq 8$ for weights (W) and $3 \leq n \leq 7$ for activations (A) for all the proposed methods (MCD, BBB, SGHMC) and a standard pointwise implementation. We considered 1 bit lower precision for activations than for weights to avoid instruction overflow on our system. All experiments were repeated 3 times and we set $L = 20$ for all methods. The code is available at `https://git.io/JtSJG`. Additional experiments and observations are in the appendix.

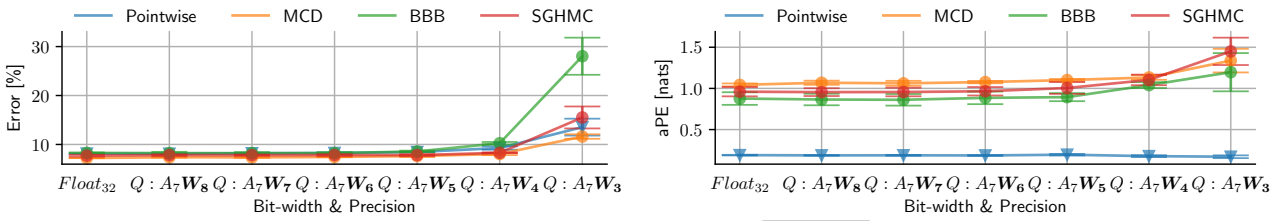
4.1 REGRESSION

The results for regression for the respective methods under quantisation are presented in Figures 4 (a,b) and 5 (a,b). We were measuring the root-mean-squared error (RMSE) and negative log-likelihood (NLL). Every box-plot is with respect to the UCI datasets and means of 10-fold cross validation that has been done with respect to independent models. First, examining the results for changing activation precision in Figure 4 (a), it can be seen from the RMSE that the Bayesian methods are more robust towards quantisation and they are able to maintain their accuracy, while a pointwise NN tends to lose its generalisability the quickest, even though it was initially marginally the most accurate.



(a) (b)

Figure 8: *Changing activation precision, fixing weight precision.* CIFAR-10 results with respect to classification error on test data (a) and average predictive entropy (aPE) on SVHN (b). Q stands for quantised activations (A) and weights (W). Subscript denotes bit-width. All methods collapse when the bit-width ≤ 4 for A.



(a) (b)

Figure 9: *Fixing activation precision, changing weight precision.* CIFAR-10 results with respect to classification error on test data (a) and average predictive entropy (aPE) on SVHN (b). Q stands for quantised activations (A) and weights (W). Subscript denotes bit-width. All methods collapse when the bit-width ≤ 3 for W.

At the same time, the Bayesian inference methods are able to maintain their uncertainty estimation capabilities which can be seen in the NLL plots in Figures 4, 5 (b). Second, results plotted in Figure 5 for changing weight precision and keeping the activation precision fixed, further solidify the previous observations. Nevertheless, the rate of change of the error with respect to quantisation of weights is slower in comparison to changing the activation precision. However, in both plots we notice that SGHMC is more affected by quantisation, especially weight quantisation. The weights' distributions for SGHMC for the different layers are more spread than the other 2 methods and uniform quantisation with respect to such a low precision for either weights or activations is unable to capture them.

4.2 CLASSIFICATION

In this Section we present the main results with respect to evaluation on MNIST and CIFAR-10 datasets. We focused on measuring classification error, expected calibration error (ECE) (Guo et al., 2017a) with respect to 10 bins and average predictive entropy (aPE). *Further results with respect to other metrics can be seen in the appendix.*

4.2.1 MNIST

The results for MNIST evaluation with respect to quantised BNNs are presented in Figures 6 (a,b) and Figures 7 (a,b). In general, the results follow the same trends as demonstrated in the regression results. Nevertheless, as seen in the classification error for changing activation precision in Figure 6 (a) the respective methods are more sensitive towards changing activation precision than weight precision in comparison to results in Figure 7 (a), in particular for MCD. The scaling factor that is applied during the MCD ($\frac{1}{1-p}$) distorts the activation distribution and results in a collapse of MCD if the bit-width for the activation is too small. However, the error of the BNNs increases marginally slower than for the pointwise NN. Nevertheless, as the error increases, the predictive entropy increases as well, which can be seen in both Figures 6 (b) and 7 (b) as a result their ECE also decreases. This means that quantisation has actually a regularising effect as with reduced precision for weights or activations the representation capabilities of NNs is limited and their confidence decreases. Interestingly for the collapsed MCD, this results in a complete and rightful uncertainty on the confusion dataset or the test set as seen in Figure 6 (b). These results translate also to measuring aPE and ECE on the test data, except for the pointwise control.

In Figures 10 (a,b) we detail results with respect to augmentations and 7-bit quantisation of the activations and 8-bit quantisation of the weights. It can be seen that the Bayesian inference methods remain to be robust towards domain shift even under quantisation and they record marginally smaller ECE and error than the pointwise control.

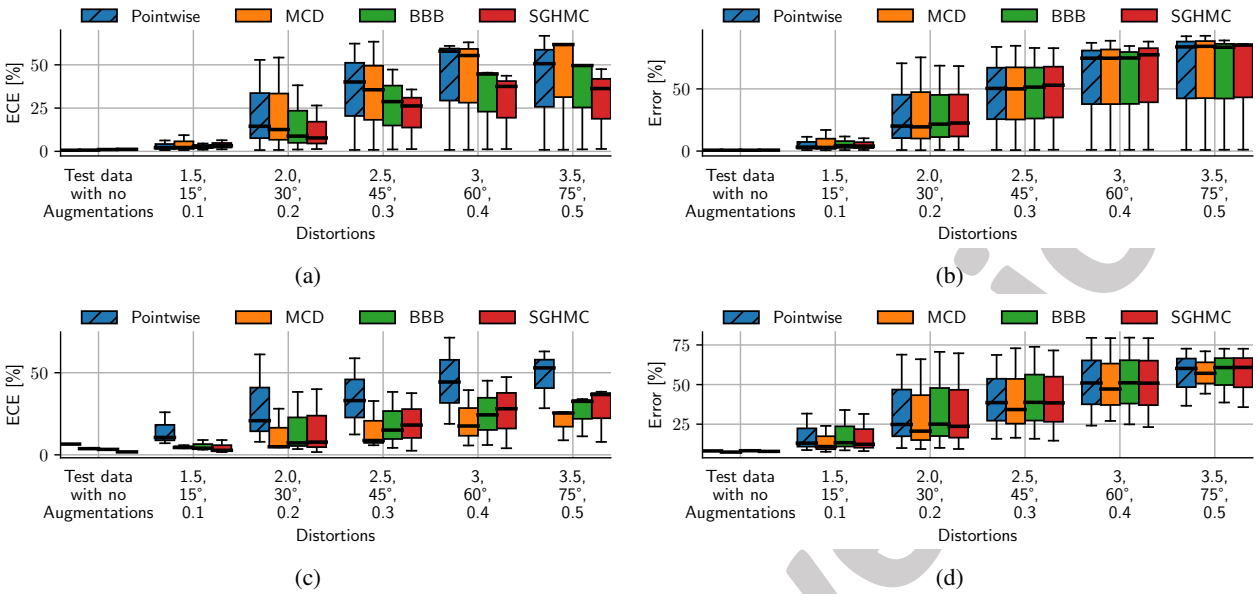


Figure 10: Expected calibration error (ECE) and classification error with respect to 7-bit activations and 8-bit weights and three augmentations applied to the LeNet-5 on MNIST test set (a) and (b) and ResNet-18 on CIFAR-10 test set (c) and (d). Augmentations were: Brightness [1.5-3.5], Rotation [15°-75°] and Horizontal shift [0.1-0.5 of image size].

4.2.2 CIFAR-10

The results for CIFAR-10 with respect to quantised BNNs are presented in Figures 8 (a,b) and Figures 9 (a,b). In this experiment the differences between the Bayesian methods and the pointwise control are widened. Similarly to the previous experiments, the quantised BNNs are more susceptible to activation quantisation in comparison to weight quantisation, while comparing the results in Figures 8 (a) and Figures 9 (a). Moreover, the qunatised nets collapse earlier, $n \leq 4$ for activations, given a more complex ResNet architecture. Nevertheless, as seen from Figures 8 (b) and Figures 9 (b) in no instance for any BNN method the uncertainty-related capability is damaged by quantisation, as the trends are clearly upwards in terms of the predictive entropy on the confusion dataset. However, as seen in Figure 9 (b) it is the pointwise model in particular which completely overfits the training dataset and quantisation has a negative effect on its predictive entropy. Similarly to previous experiments, as the error increases, the predictive entropy increases for BNNs which can be seen in both Figures 8 (b) and Figures 9 (b) as a result their ECE also decreases.

Next, if considering the domain shift as demonstrated in Figures 10 (c,d) it can be observed that while the error in Figure 10 (d) increases, the error of the Bayesian methods increases at the same rate as in a pointwise approach. However, when further examining Figure 10 (c), it can be seen that the ECE increases by far less in comparison to the pointwise approach, which makes BNNs, even under quantisation, to be more robust towards domain shift.

5 KEY TAKEAWAYS

In this work we proposed and evaluated a practical quantisation methodology for a variety of Bayesian inference methods applied to neural networks and in this Section we discuss the key takeaways of our empirical observations.

- An uniform quantisation scheme is viable for quantisation of Bayesian neural networks unless pushed to the extrema (≤ 4-bits for activations or weights). For the most commonly utilised 8-bit weights and activation quantisation scheme used in hardware, we did not observe any significant degradation in accuracy or quality of uncertainty estimation in Bayesian nets in comparison to their floating-point representation. Therefore from the hardware perspective, we expect the BNNs to follow trends of pointwise methods - latency potentially decreased by $2\times$ to $4\times$ depending on the underlying hardware platform and memory consumption decreased by $4\times$ if considering 8-bits. Quantisation below 8-bits would require a custom accelerator to see its benefits where the latency and memory consumption could be decreased (Guo et al., 2017b).
- The quality of predictive uncertainty of Bayesian networks stays unaffected or increases as a result of quantisation. The networks stay certain on the in-domain test data and become more uncertain on confusion or domain-shifted data.
- The prediction error increases at a slower rate in Bayesian neural networks, as their representation is reduced in the number of bits through quantisation, than

in pointwise networks unless considering extrema.

- Activation quantisation seemed to affect all the net types more than weight quantisation on the accuracy, predictive entropy or calibration. SGHMC was more sensitive to weight quantisation, MCD was the most sensitive to activation quantisation.
- In MCD random binary masks ($\boldsymbol{K}$) could be quantised to 1-bit whereas in BBB all parameters ($\boldsymbol{\mu}$, $\boldsymbol{\sigma}$ and $\boldsymbol{\epsilon}$) need to be quantised with same number of bits as in weights to maintain model accuracy.
- Experiments on different datasets and tasks suggest that Bayesian nets are relatively immune to quantisation. However, complex architectures (e.g. ResNet) seem to be more affected by quantisation than simpler architectures (LeNet-5) regarding their performance.

In the future work we are going to investigate more complex non-mean-field approximations for the respective Bayesian inference methods and more expressive quantisation schemes with respect to lower (≤ 4-bits) precision.

ACKNOWLEDGEMENTS

This work was partially completed while Martin Ferianc was an intern at Arm and completed through continued collaboration with Arm ML Research Lab. Martin Ferianc was also sponsored through a scholarship from the Institute of Communications and Connected Systems at UCL.

REFERENCES

Achterhold, J., Koehler, J. M., Schmeink, A., and Genewein, T. (2018). Variational network quantization. In *International Conference on Learning Representations*.

Bengio, Y., Léonard, N., and Courville, A. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*.

Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Cai, R., Ren, A., Liu, N., Ding, C., Wang, L., Qian, X., Pedram, M., and Wang, Y. (2018). Vibnn: Hardware acceleration of bayesian neural networks. *ACM SIGPLAN Notices*, 53(2):476–488.

Chen, T., Fox, E., and Guestrin, C. (2014). Stochastic gradient hamiltonian monte carlo. In *International conference on machine learning*, pages 1683–1691.

Choukroun, Y., Kravchik, E., Yang, F., and Kisilev, P. (2019). Low-bit quantization of neural networks for efficient inference. In *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, pages 3009–3018. IEEE.

Gal, Y. and Ghahramani, Z. (2015). Dropout as a bayesian approximation. *arXiv preprint arXiv:1506.02157*.

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553):452–459.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. (2017a). On calibration of modern neural networks. *arXiv preprint arXiv:1706.04599*.

Guo, K., Zeng, S., Yu, J., Wang, Y., and Yang, H. (2017b). A survey of fpga-based neural network accelerator. *arXiv preprint arXiv:1712.08934*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778. IEEE.

Hron, J., Matthews, A. G. d. G., and Ghahramani, Z. (2017). Variational gaussian dropout is not bayesian. *arXiv preprint arXiv:1711.02989*.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Krishnamoorthi, R. (2018). Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Liang, F., Li, Q., and Zhou, L. (2018). Bayesian neural networks for selection of drug sensitive genes. *Journal of the American Statistical Association*, 113(523):955–972.

MacKay, D. J. (1995). Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1):73–80.

McAllister, R., Gal, Y., Kendall, A., Van Der Wilk, M., Shah, A., Cipolla, R., and Weller, A. (2017). Concrete problems for autonomous vehicle safety: Advantages of

bayesian deep learning. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 4745–4753. AAAI Press.

Neal, R. M. (1993). Bayesian learning via stochastic dynamics. In *Advances in neural information processing systems*, pages 475–482.

Su, J., Cvitkovic, M., and Huang, F. (2019). Sampling-free learning of bayesian quantized neural networks. *arXiv preprint arXiv:1912.02992*.

van Baalen, M., Louizos, C., Nagel, M., Amjad, R. A., Wang, Y., Blankevoort, T., and Welling, M. (2020). Bayesian bits: Unifying quantization and pruning. *arXiv preprint arXiv:2005.07093*.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688.