Doctoral Thesis

Stable Learning Algorithm of Bayesian Deep Neural Networks towards Low Power Dedicated Processor

専用プロセッサによる低電力実行に向けたベイジアン深層ニューラルネットワークの安定した学習アルゴリズム

Keigo Nishida

Systems Science of Biological Dynamics
(Guest Prof. Makoto Taiji),
Graduate School of Frontier Biosciences, Osaka University

December 2022

Abstract

Deep learning has been studied by researchers interested in understanding the basic principles of information engineering, such as those pertaining to computer vision and language processing. In addition, deep learning has been investigated for application to fields such as automated driving, medical diagnostics, and drug discovery. Recent advances in deep learning have resulted in vast performance improvements; this is because continuous improvements in the computational power according to Moore's Law have made it possible to handle large-scale neural network models. However, existing deep learning models tend to lead to overconfident inferences, and this tendency becomes even stronger for inputs with distributions that differ from those of the training data, as is often the case in practical applications. Reliable inferences via deep learning are essential to precisely capture predictive uncertainty. In this regard, Bayesian neural networks (BNNs) are expected to capture predictive uncertainty to a high degree of accuracy because of their ability to model parameter uncertainty. However, in large-scale NNs such as those employed for deep learning, BNNs are problematic in that the stable convergence of parameters is difficult and the learning process of these NNs is computationally costly. Further development of Bayesian deep learning would therefore need to focus on improving the learning method and developing a dedicated processor with high power efficiency. Attempts to address these shortcomings are expected to improve Bayes by Backprop (BBB), an algorithm for BNNs that optimizes the variational parameter $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\rho}\}$. The advantage of BBB is its capability to deliver efficient inference with dedicated processors; however, because the variational parameters do not easily converge, the efficiency of learning with dedicated processors has not been sufficiently discussed in previous studies.

The objective of the research presented in this thesis was to develop algorithms for stable parameter convergence with BBB. The aim was to design these algorithms such that they are suitable for efficient learning with dedicated Bayesian deep learning processors. These algorithms and hardware are expected to enable reliable predictions of complex phenomena by BNNs. BBB operates by sampling neural network weights from variational parameters, which makes the loss function noisy. Although the parameters of noisy loss functions can be made to successfully converge using Adam as the optimizer, for BBB this optimizer prevents stable parameter convergence. To overcome this problem, I proposed using Adam with decoupled Bayes by Backprop (AdamB), which decouples the log-likelihood terms of the prior and posterior distributions in the BBB cost function from Adam. Using a covariate shift benchmark of the image classification tasks with a shifted distribution from the training data, I evaluated the accuracy and reliability of the models trained by AdamB. In addition, I discuss dedicated hardware architectures that enable AdamB to conduct training efficiently.

My study revealed that the difficulty of learning with BBB lies in the rapid and excessive update of parameter $\boldsymbol{\rho}$. This problem was fundamentally solved by AdamB using a Gaussian scale mixture prior (SM prior). In this research, I demonstrated that the rapid updates of parameter $\boldsymbol{\mu}$ can be decoupled from Adam, whereas the excessive increase can be suppressed by using an SM prior. Experiments also showed that parameter $\boldsymbol{\mu}$ takes a sparse distribution and, by using the SM prior, is strongly robust against noise type corruption. Currently, the robustness of AdamB has only been proven for covariate shift in image identification tasks, and its applications to other tasks would have to be assessed in the future. Additionally, by examining the basic architecture of Movitan, a system-on-a-chip (SOC) generator system capable of enabling AdamB to efficiently train neural networks, I provided guidelines for a dedicated processor capable of efficient and stable Bayesian deep learning. This study makes a fundamental contribution to the existing knowledge base by proposing a novel approach to solve the difficulties associated with handling BNNs because of their unstable convergence of parameters and increased computational cost.

Contents

Chapter 1

General Introduction

In this thesis, I first propose a stable learning method for Bayesian neural networks (BNNs) and present an in-depth analysis of the algorithm. Next, a basic study of the hardware architecture for low-power execution of the proposed learning method by a dedicated processor is presented. The outline of this thesis is as follows. Chapter 1 provides the necessary background information relating to this research. I explain the evolution of research on reliability in deep learning and the necessity for low-power execution with dedicated processors in the further development of deep learning. Potential applications of these studies to the life sciences are discussed. In chapter 2, I propose a stable learning method for BNNs and provide an in-depth analysis thereof. In chapter 3, I discuss the fundamentals of a dedicated processor that can run the proposed algorithm efficiently. In chapter 4, I present a general discussion based on these results and summarize the findings.

1.1 Background to this study

1.1.1 Reliability of deep learning

When humans make decisions, they consider information by taking a somewhat ambiguous viewpoint as to what the information is worth, rather than making a simple binary decision as to whether the information is useful or not useful [1]. For example, when we are informed by the weather forecast that rain is expected, many of us are unlikely to unconditionally take out an umbrella. Instead, people would be more inclined to decide whether to take along an umbrella based on the precipitation probability in the weather forecast. For example, if the forecast predicted a 30 percent chance of rain, people may leave their homes without an umbrella. On the other hand, if the forecasted probability was unreliable, they would be forced to take their umbrella with them. The reliability of this forecast probability is important in tools that support human decision-making, such as weather forecasts. In particular, it is highly important to reflect inaccurate forecasts in risky applications such as medical diagnostics [2]. Deep learning has achieved excellence in tasks based on available datasets, such as image recognition and natural language processing (NLP). On the other hand, deep learning was shown to be problematic in that it tends to return overconfident predictions[3] and this was confirmed in both computer vision [3] and NLP [4], as well as in drug discovery [5]. In other words, a situation has occurred in weather forecasting where the forecast always predicts rain with a 90% chance, but in reality it was only correct 3 out of 10 times or 30% of the time. Although it is possible to interpret the output of the last activation function (i.e., the softmax function [6]) of a deep neural network as a prediction probability (confidence), it does not necessarily reflect the actual percentage of correct responses. Predictive calibration, which reflects confidence as the actual percentage of correct responses, is a challenge in deep learning. This problem has been actively investigated for image recognition tasks, which has been one of the most remarkable achievements in deep learning. A deep learning model with good predictive calibration shows robustness to inputs that do not fall within the distribution of the training data. For example, for out of domain (OOD) inputs of which the correct label is not present in the test

data, the confidence has a small value and can be removed by thresholding[7]. The confidence is also adjusted for inputs that do not change the class to be classified, but which change the distribution of the training data, such as the addition of noise. Note that a model with good predictive calibration does not necessarily provide high accuracy. A model with good predictive calibration is able to effectively lower the output value of softmax for unconfident inputs. Predictive calibration, which reflects confidence as the actual percentage of correct responses, is a challenge in deep learning. Training models with superior predictive uncertainty have been designed following Bayesian approaches that consider neural networks probabilistically and incorporate uncertainty in the parameters[8]. Because estimation of the true posterior distribution with the use of BNNs is computationally expensive, the posterior distribution is approximated by the Monte Carlo dropout (MC-Dropout) technique [9] and variational inference methods [10][11]. Another commonly used approach is Deep Ensembles, which is non-Bayesian[7], and estimates the predictive uncertainty by using multiple models trained with different initial values. Deep ensembles may have properties closer to the true posterior distribution than variational inference methods, and they provide robust inference for changes in various training data distributions [12]. Because the individual models of deep ensembles resemble models based on ordinary deep learning, parameter convergence is stable. However, the approach is problematic in that the computational and memory costs increase with the number of ensembles at training time.

Alternatively, temperature scaling could be used to improve the prediction uncertainty by scaling the inputs to the softmax function without changing the parameters of the trained model. However, this is a post-hoc approach that uses a validation set to determine the scalar values. The disadvantage thereof is that predictive calibration is ineffective when the distribution shift from the training data is large[13]. Obtaining predictive uncertainty for distributions that differ from the training data may require some ingenuity in learning.

1.1.2 Dedicated hardware for deep neural networks

The exponential supply of computing resources in computer systems is a major factor in the remarkable progress of deep learning. However, the performance improvement comes with a limit. The performance of computers has continued to increase over the past half-century with the development of semiconductor miniaturization technology. The circuit resources available for processor design continue to increase in accordance with Moore's Law [14], according to which the number of transistors per chip continues to increase exponentially. Denard scaling, which contributed to improved power efficiency in semiconductor miniaturization, came to an end in 2005[15], and, even though the number of transistors available on a chip increased, power and thermal limitations made it difficult to increase the operating frequency. Since then, the tendency was to reduce the operating frequency and improve the computing performance based on parallelization with multiprocessor cores. However, on the basis of Amdahl's law [16], the multiprocessor core approach essentially cannot resolve execution time bottlenecks in those parts of the computation that cannot be parallelized. The computational power required for deep learning exceeds the rate of CPU performance improvement according to Moore's Law, and this is being covered by the acceleration of deep learning with dedicated hardware (see Figure 1.1).

Other approaches to improving the computational performance under power-constrained conditions include the development of specialized processors. In particular, approaches involving highly dedicated accelerators are used for graphics, compression, and machine learning. GPUs, accelerators dedicated to graphics processing, have contributed significantly to the evolution of deep learning. After AlexNet, invented in the early days of deep learning, achieved efficient learning by parallelization on GPUs[17], GPUs became the standard tool for deep learning. As deep learning models become larger, the computational resources must also become larger, and the cost of powering these resources is not negligible. This prompted Google to develop dedicated hardware for deep learning in the form of tensor processing units (TPUs), which accelerated the power performance by tens of times compared to the GPUs of the time[18]. Dedicated hardware for deep learning is optimized primarily

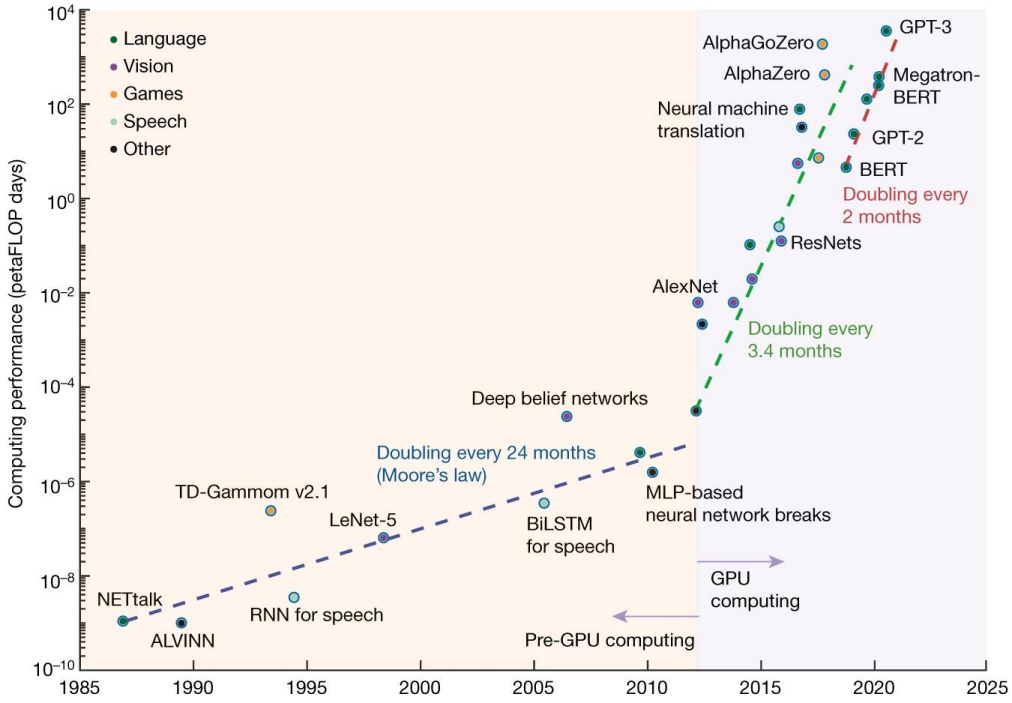

Figure 1.1: Increase in computational demand in neural networks over the past 40 years[19].

by modifying the configuration of the arithmetic unit and memory buffer capacity required for matrix multiplication. By reducing data and computational precision, deep learning inference and training can be dramatically accelerated, saving significant execution time and power consumption. Deep learning is still evolving today, with no guarantee that future deep learning designs would be able to run efficiently on conventional dedicated hardware. In fact, state-of-the-art deep learning models such as EfficientNet [20] and BERT [21] cannot map their workflows well to dedicated hardware [22]. This indicates that the hardware needs to keep pace with the evolution of deep learning and engineers need to continue to explore the best architecture. However, designing dedicated hardware in a rapidly evolving field is not easy. The search for architecture that enables a specific algorithm to become more efficient requires the integration of the hardware and software in the system and the verification to evaluate the viability of dedicated hardware. Furthermore, if efficient hardware architecture could be designed, the architect could proceed to the development of application specific integrated circuits (ASICs), where the physical design is conducted. For applications such as deep learning, where the

underlying algorithms are common but the workflow bottleneck changes depending on the network architecture, a consistent and efficient development flow from architecture exploration to workload evaluation is essential. In preparation for the future exploration of diversified dedicated hardware architectures, the development of a framework for generators capable of agile system-on-a-chip (SoC) development is actively underway[23]. Clear guidelines regarding hardware architecture that can efficiently handle predictive uncertainty in deep learning are non-existent [24], which is why it is important to follow the agile development approach.

1.1.3 Importance of reliability evaluation by deep learning in the life sciences

Medical diagnosis is one of the most important topics in the life sciences. The use of machine learning to diagnose infected patients is a highly interesting topic because it is expected to utilized efficiently and in a scalable manner. Coronavirus disease 2019 (COVID-19) [25] is an infectious disease that broke out in 2019 and was declared a pandemic in March 2020. COVID-19, which rapidly spread worldwide and caused spectacular social disruption, is still raging today. An effective test for COVID-19 infection, the reverse transcription polymerase chain reaction (RT-PCR), has become popular, although imaging is also used for diagnosis. However, although attempts have been made to use deep learning to diagnose COVID-19, the reported results are extremely optimistic and are not practical[26]. Deep learning models do not generalize well to differences in measurement devices and methods in different hospitals[27]. The COVID-19 protein has a variety of mutations[28], and the subtle changes in symptoms caused by this variety may make a diagnosis by deep learning more difficult. For these situations, deep learning models that represent high-quality predictive uncertainty are expected to be a tool to support diagnoses by medical experts[29].

1.2 Neural Networks

This research focuses on supervised multiclass classification using a training dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathrm{y_n})\}_{n=1}^{N}$ with N inputs $\mathbf{x}$ and target y. The probabilistic model of the neural network is denoted as $P(\mathcal{D}|\mathbf{w})$ with parameters or weights $\mathbf{w}$. The maximum a posteriori (MAP) estimation is a point estimate of the random variable $\mathbf{w}^{\mathrm{MAP}}$ that maximizes the posterior probability $P(\mathbf{w}|\mathcal{D})$. Random variable $\mathbf{w}^{\mathrm{MAP}}$ can be estimated by minimizing the cost function $L^{\mathrm{MAP}}(\mathcal{D}, \mathbf{w})$:

$$\begin{aligned} L^{\mathrm{MAP}}(\mathcal{D}, \mathbf{w}) &= -\log P(\mathcal{D}|\mathbf{w}) - \log P(\mathbf{w}) \\ &= L^{N}(\mathcal{D}, \mathbf{w}) - \log P(\mathbf{w}), \end{aligned} \tag{1.1}$$

where the negative log-likelihood (NLL) is defined as $L^{N}(\mathcal{D}, \mathbf{w}) := -\log P(\mathcal{D}|\mathbf{w})$. If $P(\mathbf{w})$ is a Gaussian distribution with zero mean and variance σ, then (1.1) is a form of L_2 regularization that penalizes the weights if they exhibit large values. The standard stochastic gradient descent (SGD) update of $\mathbf{w}$ at timestep t is as follows.

$$\begin{aligned} \mathbf{w}_t &= \mathbf{w}_{t-1} - \alpha \nabla_{\mathbf{w}} L^{\mathrm{MAP}}(\mathcal{D}, \mathbf{w}_{t-1}) \\ &= (1 - \lambda)\, \mathbf{w}_{t-1} - \alpha \mathbf{g}_t, \end{aligned} \tag{1.2}$$

where α is the learning rate, $\mathbf{g}_t$ is the $L^{N}(\mathcal{D}, \mathbf{w})$ gradient at timestep t, and λ is the regularization factor defined as $\lambda := \alpha\sigma^{-2}$. This factor functions as the weight decay [30] because the terms of the L_2 regularization are used directly to update the weights $\mathbf{w}$. However, L_2 regularization and weight decay are not necessarily equivalent. When using Adam [31], the sum of $\mathbf{g}_t$ and the L_2 regularization term was evaluated as the gradient estimate. Consequently, the L_2 regularization term is not directly reflected in the weight update, which is not the same as the weight decay.

1.2.1 Decoupled weight decay

To directly update the weights by the regularization term as a weight decay using Adam, the update rule for the weights $\mathbf{w}$ is expressed as

$$\begin{aligned}
\mathbf{m}_t &= \beta_1 \mathbf{m}_{t-1} + (1-\beta_1)\,\mathbf{g}_t \\
\mathbf{v}_t &= \beta_2 \mathbf{v}_{t-1} + (1-\beta_2)\,\mathbf{g}_t^2 \\
\hat{\mathbf{g}}_t &= \frac{\mathbf{m}_t / (1-\beta_1^t)}{\sqrt{\mathbf{v}_t / (1-\beta_2^t)}} \\
\mathbf{w}_t &= (1-\lambda)\,\mathbf{w}_{t-1} - \alpha \hat{\mathbf{g}}_t,
\end{aligned} \tag{1.3}$$

where β_1 and β_2 are the exponential decay rates for the momentum estimates $\mathbf{m}_t$ and $\mathbf{v}_t$, respectively, and $\hat{\mathbf{g}}_t$ is the gradient estimate by Adam. This method of decoupling $\mathbf{g}_t$ and the regularization term from the evaluation by Adam is referred to as Adam with decoupled weight decay (AdamW)[32]. AdamW is robust to the hyperparameter of the regularization factor λ and converges stably. This modification of the optimizer has found application in NLP, reinforcement learning, and image recognition[33].

1.2.2 Reliability Metrics

Normalization of the class prediction $\hat{y}$ by the inference model by 1 can be regarded as a probability distribution for the predicted label, and its maximum value can be evaluated as confidence $\hat{\mathbf{p}}$. On the condition that $\hat{\mathbf{p}}$ and the actual correct response rate are consistent, the probability calibration is determined. Because $\hat{\mathbf{p}}$ is a continuous random variable, it is impossible to evaluate this confidence level from a finite number of data samples. Therefore, the probability calibration is evaluated by assigning the model outputs to m equally spaced bins using a finite number of data samples. Let b_m be the set of samples in the mth bin. The errors between the average confidence value $conf\,(b_m)$ and the correct response rate $acc\,(b_m)$ within the

bin are determined, whereupon the weighted average of these errors is calculated as,

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|b_m|}{N_{ECE}} \left| acc\left(b_m\right) - conf\left(b_m\right) \right|. \tag{1.4}$$

The expected calibration error (ECE) is used to approximately evaluate the probability calibration [34]. N_{ECE} is the total number of data points.

1.2.3 Distribution shifts

Distribution shifts include dataset shifts that belong to one of the same classes as in the training dataset but the observed distribution has shifted (covariate shift). Domain shifts, which are completely different distributions with no corresponding class in the training dataset, could also occur. To evaluate the dataset shift, the relative robustness of each model can be evaluated by computing the change in ECE when noise of different degrees is added to the test data [13]. The proposed evaluation of the covariate shift entails the addition of a dataset with various types and levels of corruption to the image recognition dataset (Figure 1.2, Figure 1.3). In the case of domain shift, the correct answer label is not at the output of the inference model, but can be removed by setting a threshold value for the confidence [7].

1.2.4 Bayesian neural networks

Bayes by backprop (BBB)

Bayesian neural networks, which are robust to overfitting [36], find the posterior distribution $P\left(\mathbf{w}|\mathcal{D}\right)$ given data $\mathcal{D}$. Because obtaining the exact posterior distribution $P\left(\mathbf{w}|\mathcal{D}\right)$ is challenging, this study considered variational inference to obtain the distribution parameter $\boldsymbol{\theta}$ of the weights as the variational posterior distribution $q\left(\mathbf{w}|\boldsymbol{\theta}\right)$ [10][11][37]. In variational inference, the variational free energy $F\left(\mathcal{D}, \boldsymbol{\theta}\right)$

$$F\left(\mathcal{D}, \boldsymbol{\theta}\right) = \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\theta})}\left[L^N\left(\mathcal{D}, \mathbf{w}\right) \right] + \mathrm{KL}(q(\mathbf{w}|\boldsymbol{\theta})||P(\mathbf{w})) \tag{1.5}$$

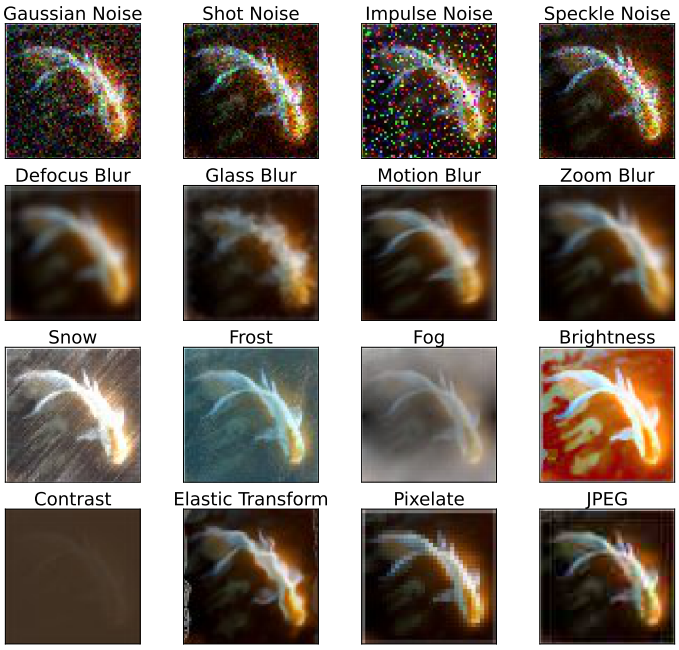

Figure 1.2: Effect of different types of image data corruption[35]

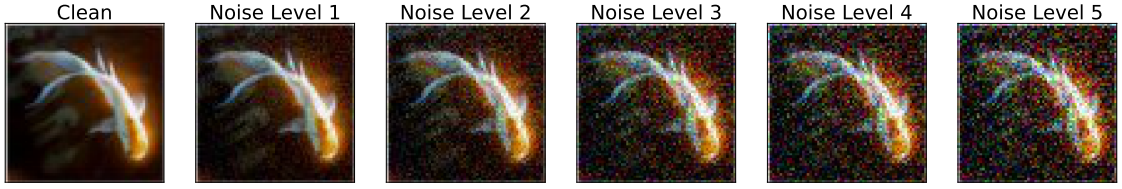

Figure 1.3: Effect of different levels of image data corruption [35]

is minimized, where KL is the Kullback-Leibler divergence. The first term of this cost function is interpreted as an expected NLL. Further, the second term is the complexity loss, which is interpreted as a minimum description length loss function[10] [11]. In BBB, KL is treated as an approximation by Monte Carlo sampling rather than a closed form. Therefore, with S as the number of samples obtained by Monte Carlo sampling, (1.5) for mini-batch training, where the training data $\mathcal{D}$ are divided into M subsets as $\mathcal{D}_1, \mathcal{D}_2, ..., \mathcal{D}_M$, becomes

$$F\left(\mathcal{D}_i, \boldsymbol{\theta}\right) = \frac{1}{S}\sum_{k=1}^{S} L^N\left(\mathcal{D}_i, \mathbf{w}^k\right) + \frac{1}{M}\left\{\log q\left(\mathbf{w}^k \middle| \boldsymbol{\theta}\right) - \log P\left(\mathbf{w}^k\right)\right\}. \tag{1.6}$$

When the posterior distribution is assumed to be normal, the weight $\mathbf{w}^k$ as the variational parameter $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\rho}\}$ is

$$\mathbf{w}^k = \boldsymbol{\mu} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}^k, \tag{1.7}$$

where

$$\boldsymbol{\epsilon}^k \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), \boldsymbol{\sigma} = \log(\mathbf{1} + \exp(\boldsymbol{\rho})),$$

where $\circ$ is pointwise multiplication. Now, the complex cost $L^C(\boldsymbol{\theta})$ at the BBB is defined as

$$L^C(\boldsymbol{\theta}) = \frac{1}{S} \sum_{k=1}^{S} \left\{ \log q\left(\mathbf{w}^k \middle| \boldsymbol{\theta}\right) - \log P\left(\mathbf{w}^k\right) \right\}. \tag{1.8}$$

If a sufficiently large number of datasets are used, the sample size $S = 1$ may cease to be a problem[11]. The log-likelihood of the posterior distribution is

$$\begin{aligned} \log q\left(\mathbf{w}^k \middle| \boldsymbol{\theta}\right) &= \sum_{j=1}^{J} \left\{ -\log \sqrt{2\pi}\sigma_j - \frac{\left(w_j^k - \mu_j\right)^2}{2\sigma_j^2} \right\} \\ &= \sum_{j=1}^{J} \left\{ -\log \sqrt{2\pi}\sigma_j - \frac{(\epsilon_j^k)^2}{2} \right\}, \end{aligned} \tag{1.9}$$

where J is the size of the weight parameter, $\sigma_j = \log(1 + \exp(\rho_j))$. If J is sufficiently large, the log-likelihood of the posterior distribution can be regarded as a function of σ_j. Therefore, the gradient for each of these variables is

$$\frac{\partial \log q\left(\mathbf{w}^k \middle| \boldsymbol{\theta}\right)}{\partial w_j} = 0, \tag{1.10}$$

$$\frac{\partial \log q\left(\mathbf{w}^k \middle| \boldsymbol{\theta}\right)}{\partial \mu_j} = 0, \tag{1.11}$$

$$\frac{\partial \log q\left(\mathbf{w}^k \middle| \boldsymbol{\theta}\right)}{\partial \sigma_j} = -\frac{1}{\sigma_j}. \tag{1.12}$$

Here, if $S = 1$, then the gradient of $F(\mathcal{D}_i, \boldsymbol{\theta}_{t-1})$ as $\mathbf{w} = \boldsymbol{\mu}_{t-1} + \log\left(\mathbf{1} + \exp\left(\boldsymbol{\rho}_{t-1}\right)\right) \circ \boldsymbol{\epsilon}$ is

$$\begin{aligned}
\boldsymbol{\Delta}_{\boldsymbol{\mu}} &= \nabla_{\mathbf{w}} F(\mathcal{D}_i, \boldsymbol{\theta}_{t-1}) \frac{\partial \mathbf{w}}{\partial \boldsymbol{\mu}} + \nabla_{\boldsymbol{\mu}} F(\mathcal{D}_i, \boldsymbol{\theta}_{t-1}) \\
&= \nabla_{\mathbf{w}} F(\mathcal{D}_i, \boldsymbol{\theta}_{t-1}) \\
&= \mathbf{g}_t - \frac{1}{M} \nabla_{\mathbf{w}} \log P(\mathbf{w}) \qquad (1.13) \\
\boldsymbol{\Delta}_{\boldsymbol{\rho}} &= \left\{ \nabla_{\mathbf{w}} F(\mathcal{D}_i, \boldsymbol{\theta}_{t-1}) \frac{\partial \mathbf{w}}{\partial \boldsymbol{\sigma}} \right. \\
&\qquad \left. + \nabla_{\boldsymbol{\sigma}} F(\mathcal{D}_i, \boldsymbol{\theta}_{t-1}) \right\} \frac{\partial \boldsymbol{\sigma}}{\partial \boldsymbol{\rho}} \\
&= \left\{ \boldsymbol{\Delta}_{\boldsymbol{\mu}} \boldsymbol{\epsilon} \right. \\
&\qquad \left. - \frac{1}{M} \frac{1}{\boldsymbol{\sigma}} \right\} \frac{1}{1 + \exp\left(-\boldsymbol{\rho}_{t-1}\right)}. \qquad (1.14)
\end{aligned}$$

and the variational parameters are updated as follows.

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} - \alpha \boldsymbol{\Delta}_{\boldsymbol{\mu}} \qquad (1.15)$$

$$\boldsymbol{\rho}_t = \boldsymbol{\rho}_{t-1} - \alpha \boldsymbol{\Delta}_{\boldsymbol{\rho}}, \qquad (1.16)$$

where α is the learning rate. The sampling from the variational posterior distribution $q(\mathbf{w}|\boldsymbol{\theta})$ can be used to evaluate the predictive distribution $P(\hat{\mathrm{y}}|\hat{\mathbf{x}})$ using an ensemble.

$$\begin{aligned}
P(\hat{\mathrm{y}}|\hat{\mathbf{x}}) &= \mathbb{E}_{q(\mathbf{w}|\boldsymbol{\theta})} \left[p(\hat{\mathrm{y}}|\hat{\mathbf{x}}, \mathbf{w}) \right] \\
&\fallingdotseq \frac{1}{S} \sum_{k=1}^{S} p\left(\hat{\mathrm{y}} \middle| \hat{\mathbf{x}}, \mathbf{w}^k\right), \qquad (1.17)
\end{aligned}$$

where $\hat{\mathbf{x}}$ is the new input and $\hat{\mathrm{y}}$ is the unknown label.

Low Precision BNNs

Monte Carlo-based BNN inference is computationally expensive because it requires Gaussian sampling in addition to general matrix-matrix multiplication (GEMM) in ordinary NNs. An approach to reduce the computational cost of GEMMs is to convert floating-point operations to integer operations of a few bits by quanti-

zation, as is usually done with NNs. Experiments confirmed that quantizing the weights of models trained with BBB to 8-bit integers does not degrade the probability calibration and accuracy [38]. Apart from the above-mentioned approaches, the computational cost of Gaussian sampling was also reduced by adapting the hardware architecture. In particular, a dedicated Bayesian NN accelerator for inference, which implements an integer GEMM accelerator and a dedicated circuit for low-precision Gaussian sampling, was developed to optimize the computational flow from sampling to GEMM at the hardware level. For example, Cai et al. proposed an FPGA-based BBB inference accelerator that samples the normal distribution from the binomial distribution based on the central limit theorem (CLT)[39]. Subsequently, Hirayama et al. proposed sampling of the normal distribution by using a lookup table (LUT) based on inverse transform sampling[40].

1.3 Agile system-on-a-chip development

Improving the performance of computer systems by developing dedicated hardware is essential. However, designing hardware from scratch for every application is impractical because of the implementation costs. Even if dedicated hardware was targeted at a specific application, prompt consideration of architectural changes is necessary when changes occur in the latest algorithms. Changes in the hardware design to accommodate the targeted algorithm are not simply a matter of changing a particular module and verifying it at the block level: the module must be integrated into the system as hardware and validated.

1.3.1 Chipyard

Chipyard is open source software that provides a unified framework and workflow for agile SoC development (Figure 1.4). Highly parameterized CPU cores, memory systems, peripherals, and dedicated circuits can be interconnected to generate a complete SoC. The generated SoC design can be verified using software simulation and FPGA prototyping. Once verification is complete, the design can be seamlessly shifted to the VLSI design flow, enabling rapid ASIC physical design. Therefore, by

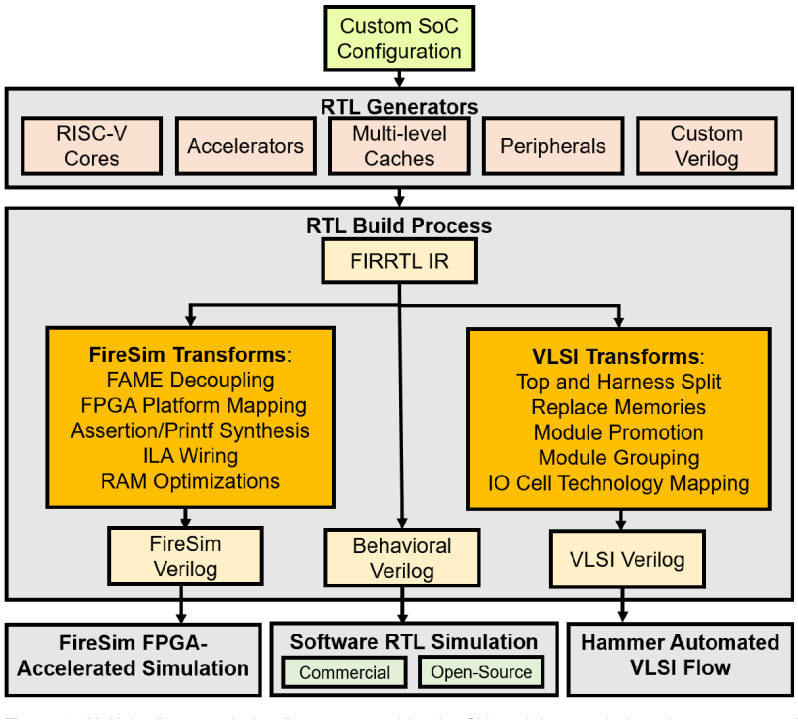

Figure 1.4: Chipyard frameworks[23]

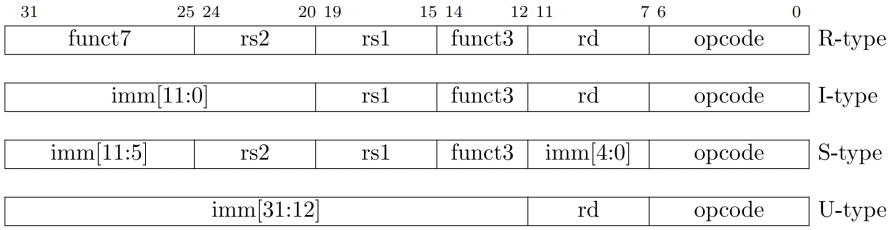

Figure 1.5: RISC-V base instruction formats[42]

designing dedicated hardware that can be integrated into the Chipyard framework, agile hardware design is possible even with a small number of developers. CPUs available in Chipyard can use RISC-V ISAs that support the rocket core processor [41]. RISC-V ISA is an instruction set architecture that defines the minimum functionality required for a processor and enables ISA extensions to be made as needed [42]. The base integer ISA is the foundation for RISC-V. Four basic instruction formats are defined: R-type, I-type, S-type, and U-type, as shown in Figure 1.5. Rocket core provides a rocket custom co-processor (RoCC) interface for connection to customized dedicated hardware. To access the RoCC from Rocket core, the user specifies one of the four custom opcodes supported by the R-type ISA.

1.3.2 Gemmini

Gemmini is a generative system based on the Systolic Array architecture integrated into the Chipyard framework. This system can be parameterized in terms of arithmetic precision of the arithmetic unit and the size and number of banks of scratchpads, allowing the design of efficient hardware architectures according to the architecture of deep learning. Gemmini issues execution commands from the RISC-V CPU through the RoCC interface. Software programming is accomplished via the C/C++ API, and the parts that depend on the Gemmini architecture utilize information in a dedicated C header that is generated simultaneously with the hardware.

1.4 The objective of this thesis

The goal of the research presented in this thesis is to develop algorithms that allow stable parameter convergence with BBB to enable efficient learning with dedicated Bayesian deep learning processors that make reliable predictions. In this regard, deep ensembles capture superior predictive uncertainty, but the training cost increases in proportion to the number of ensembles. This problem can be overcome by using BBB, which offers a viable solution, although it doubles the number of parameters. However, the generation of random numbers and the evaluation of probability distributions can be performed with dedicated hardware for low-power execution. This research therefore also involved a study of the fundamentals of dedicated hardware capable of supporting efficient BBB training.

Chapter 2

Decoupled Bayes by Backprop

The performance optimization of reliable Bayesian deep learning models requires the algorithm to be improved before a dedicated processor is developed. In this chapter, I propose Adam with decoupled Bayes by Backprop (AdamB), which serves to stabilize the training of Bayesian deep neural networks. The proposed algorithm is evaluated and discussed. Chapter 2.1 provides the background and objectives of the work presented in this chapter. Chapter 2.2 describes the ideas underlying AdamB. Chapter 2.3 presents my evaluation of AdamB on an image identification task and my assessment of its robustness to data corruption. In Chapter 2.4, I provide a detailed analysis of the covariate shifts of the models trained by AdamB. In Chapter 2.5, I evaluate the possibility of training AdamB with low-precision random numbers, which is necessary for the low-power execution of AdamB on a dedicated processor.

2.1 Stable Bayesian neural network training

2.1.1 Background

Overfitting is a major problem in machine learning. Because deep learning enables high predictive accuracy in image classification tasks, the research focus has shifted to improving the quality of the predictive uncertainty [43]. Weight decay can be implemented as the gradients of L_2 regularization with the assumption of a Gaussian distribution for prior maximum a posteriori (MAP) estimation. This is a traditional method for preventing overfitting[44] while improving the prediction uncertainty [3]. In another approach, high-quality prediction uncertainty can be represented by deep ensembles that train an ensemble of neural networks with different initial values[7]. Deep ensembles are non-Bayesian but may well approximate the ideal posterior distribution in Bayesian neural networks (BNNs)[12]. Although it is a simple approach that provides a good representation of predictive uncertainty, the ensemble increases the computational and memory costs during training. Dropout and mean-field variational inference (MFVI) approaches have been used to learn the posterior distribution of the BNNs. MC-dropout represents an ensemble of models obtained by sampling weights with dropout [9]. The implementation thereof is easy and the parameters converge stably; however, the quality of the predictive uncertainty is poorer than that with deep ensembles [7] [13]. Further, MFVI facilitates the marginalization of the weight parameters by assuming a computationally tractable posterior distribution[10] [11]. Unfortunately, the application of MFVI to large neural networks is limited and requires various techniques and tuning for parameter convergence, such as decorrelating gradients in mini-batches [45], the natural gradient method [46], and mapping of weights to uncertain activation [47][48]. In this respect, the predictive uncertainty of the empirical Bayes approach [49], which switches to variational inference after maximum likelihood estimation (MLE), is superior to that of deep ensembles [50]. This study showed that, although variational inference is not necessarily inferior to deep ensembles, it cannot overcome the challenge of parameter convergence in variational inference. The Adam optimizer employs a small step size for parameter updates by using automatic annealing when

the variance of the gradient estimate is large[31]. Thus, it is naturally expected to be a good optimizer for BBB; however, it is ineffective in practice. The radial BNN uses a light-tailed posterior distribution to suppress the increase in the variance of the gradient estimator and allow convergence with Adam [51]. Thus, it is necessary to determine whether Adam performs well in MFVI using the vanilla Gaussian posterior. A technique to improve parameter convergence is to use Adam with decoupled weight decay (AdamW), wherein the regularization term of the loss function is decoupled from the gradient estimate of Adam and applied directly to the weight update as a weight decay[32]. If the gradient of the regularization term, which tends to be significantly larger than the gradient of the neural network loss, is not decoupled, the gradient of the neural network loss does not hold properly within the momentum of Adam[33]. Moreover, an important insight gained from AdamW is that the performance is adequate even when Adam is used only for the gradients of the neural network loss. Bayes by Backprop (BBB) is an MFVI algorithm [37] that samples the weights $\mathbf{w}$ from the variational parameter $\boldsymbol{\theta} = \{\boldsymbol{\mu}, \boldsymbol{\rho}\}$ of the Gaussian posterior distribution using a reparameterization trick [52]. Parameter $\boldsymbol{\mu}$ is the mean of the Gaussian, and parameter $\boldsymbol{\rho}$ is transformed into a non-negative value by the softplus function and used as the standard deviation of the Gaussian. Consequently, the BBB cost function is formed by adding the log-likelihood of the variational posterior distribution to the cost function of the MAP estimation. However, even with such simple changes, training VGG [53] and ResNet [54] remains challenging and tends to fail[46][55]. BBB is preferred when using a Gaussian scale mixture (SM) prior [37], which is a mixture of two Gaussians with different variances. However, the significance of using the SM prior remains unclear. Moreover, although the use thereof has been reported to slightly improve the performance of large neural networks [56], other reports indicated that the effect is insignificant[51]. Thus, stabilizing the BBB can aid in evaluating the effectiveness of the SM prior.

2.1.2 Purpose of this study

This study led to the proposal of Adam with decoupled Bayes by Backprop (AdamB), which decouples the log-likelihood terms of the prior and posterior distributions in

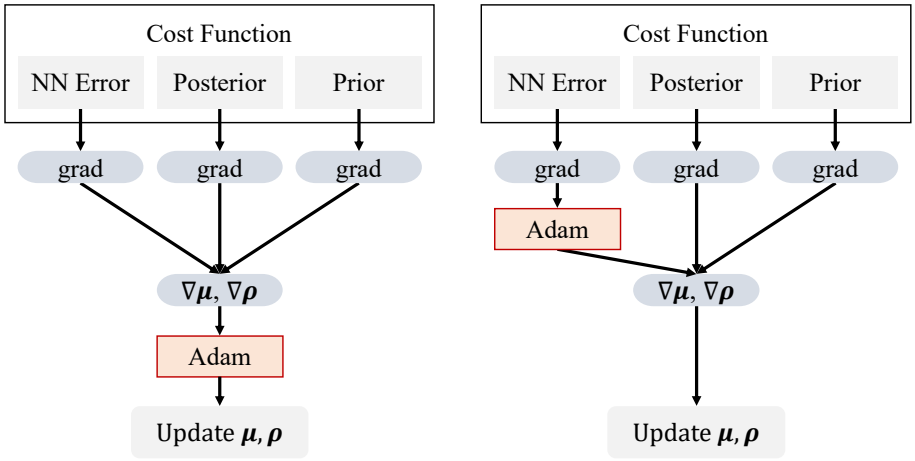

Figure 2.1: Updating the Bayes by Backprop (BBB) parameters using the Adam estimator. Adam with BBB (left) and Adam with decoupled BBB

Algorithm 1 Momentum SGD with Decoupled Bayes by Backprop

given $\alpha_\mu = 0.1, \alpha_\rho = 0.001, \beta_1 = 0.9, \lambda_d = 0.01$
initialize $\boldsymbol{\mu}, \boldsymbol{\rho}, \mathbf{m}_0, t, \eta_0$
repeat
  Randomly sample a data example $\mathcal{D}_i$
  $t \leftarrow t + 1$
  $\boldsymbol{\sigma} \leftarrow \log\left(\mathbf{1} + \exp\left(\boldsymbol{\rho}_{t-1}\right)\right)$
  $\mathbf{w} \leftarrow \boldsymbol{\mu}_{t-1} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{g}_t \leftarrow -\nabla_{\mathbf{w}} \log P\left(\mathcal{D}_i|\mathbf{w}\right)$ ▷ estimate NN gradient
  $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\, \mathbf{g}_t$
  $\boldsymbol{\Delta}_{\boldsymbol{\mu}} \leftarrow \mathbf{m}_t - \lambda_d \nabla_{\mathbf{w}} \log P_{\mathrm{SM}}(\mathbf{w})$
  $\boldsymbol{\Delta}_{\boldsymbol{\rho}} \leftarrow \left(\boldsymbol{\Delta}_{\boldsymbol{\mu}} \circ \boldsymbol{\epsilon} - \lambda_d / \boldsymbol{\sigma}\right) 1/\{1 + \exp\left(-\boldsymbol{\rho}_{t-1}\right)\}$,
  $\boldsymbol{\mu}_t \leftarrow \boldsymbol{\mu}_{t-1} - \eta_{t-1} \alpha_\mu \boldsymbol{\Delta}_{\boldsymbol{\mu}}$
  $\boldsymbol{\rho}_t \leftarrow \boldsymbol{\rho}_{t-1} - \alpha_{\boldsymbol{\rho}} \boldsymbol{\Delta}_{\boldsymbol{\rho}}$
  $\eta_t \leftarrow$ SchedulerUpdate(η_{t-1})
until $\boldsymbol{\mu}$ has converged

the BBB cost function from Adam (see Figure 2.1). The advantage of AdamB is that it facilitates stable parameter convergence in image classification tasks under training conditions that are nearly identical to MAP estimation. The update of parameter $\boldsymbol{\rho}$ is strongly influenced by the standard deviation of the prior distribution. However, the direct application of Adam to BBB causes this parameter to be excessively updated and this tendency was observed to relax by decoupling BBB. Furthermore, training AdamB with the SM prior improved the quality of predictive uncertainty in the covariate shift benchmark [13]. This approach outperformed deep ensembles for datasets wherein covariate shift was caused by noise.

Algorithm 2 Adam with Decoupled Bayes by Backprop (AdamB)

given $\alpha_\mu = 0.001, \alpha_\rho = 0.001, \beta_1 = 0.9, \beta_2 = 0.999, \lambda_d = 0.01$
initialize $\boldsymbol{\mu}, \boldsymbol{\rho}, \mathbf{m}_0, \mathbf{v}_0, t, \eta_0$
repeat
  Randomly sample a data example $\mathcal{D}_i$
  $t \leftarrow t + 1$
  $\boldsymbol{\sigma} \leftarrow \log\left(\mathbf{1} + \exp\left(\boldsymbol{\rho}_{t-1}\right)\right)$
  $\mathbf{w} \leftarrow \boldsymbol{\mu}_{t-1} + \boldsymbol{\sigma} \circ \boldsymbol{\epsilon}$, where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
  $\mathbf{g}_t \leftarrow -\nabla_{\mathbf{w}} \log P(\mathcal{D}_i|\mathbf{w})$ ▷ estimate NN gradient
  $\mathbf{m}_t \leftarrow \beta_1 \mathbf{m}_{t-1} + (1 - \beta_1)\, \mathbf{g}_t$
  $\mathbf{v}_t \leftarrow \beta_2 \mathbf{v}_{t-1} + (1 - \beta_2)\, \mathbf{g}_t \circ \mathbf{g}_t$
  $\hat{\mathbf{m}}_t \leftarrow \mathbf{m}_t / (1 - \beta_1^t)$
  $\hat{\mathbf{v}}_t \leftarrow \mathbf{v}_t / (1 - \beta_2^t)$
  $\hat{\mathbf{g}}_t \leftarrow \hat{\mathbf{m}}_t / (\sqrt{\hat{\mathbf{v}}_t} + eps)$
  $\boldsymbol{\Delta}_{\boldsymbol{\mu}} \leftarrow \hat{\mathbf{g}}_t - \lambda_d \nabla_{\mathbf{w}} \log P_{\mathrm{SM}}(\mathbf{w})$
  $\boldsymbol{\Delta}_{\boldsymbol{\rho}} \leftarrow (\boldsymbol{\Delta}_{\boldsymbol{\mu}} \circ \boldsymbol{\epsilon} - \lambda_d / \boldsymbol{\sigma})\, 1/\{1 + \exp\left(-\boldsymbol{\rho}_{t-1}\right)\}$,
  $\boldsymbol{\mu}_t \leftarrow \boldsymbol{\mu}_{t-1} - \eta_{t-1} \alpha_\mu \boldsymbol{\Delta}_{\boldsymbol{\mu}}$
  $\boldsymbol{\rho}_t \leftarrow \boldsymbol{\rho}_{t-1} - \alpha_{\boldsymbol{\rho}} \boldsymbol{\Delta}_{\boldsymbol{\rho}}$
  $\eta_t \leftarrow$ SchedulerUpdate(η_{t-1})
until $\boldsymbol{\mu}$ has converged

2.2 Adam with decoupled Bayes by backprop

2.2.1 Updates of the variational parameter

As is evident from the comparison of (1.13) and differentiated form of (1.1), there is no significant difference between the gradient of $\boldsymbol{\mu}$ in BBB and that of $\mathbf{w}$ in the MAP estimation. Because $\nabla_{\boldsymbol{\rho}} \boldsymbol{\sigma}$ is a sigmoid function, the gradient of parameter $\boldsymbol{\rho}$ and that of variable $\boldsymbol{\sigma}$ have the same sign. Therefore, by examining $\nabla_{\boldsymbol{\rho}} L^C(\boldsymbol{\theta})$, the sign of $\nabla_{\boldsymbol{\sigma}} L^C(\boldsymbol{\theta})$ can be determined.

$$\nabla_{\boldsymbol{\sigma}} L^C(\boldsymbol{\theta}) = \frac{1}{S} \sum_{k=1}^{S} \left\{ -\frac{1}{\boldsymbol{\sigma}} - \frac{\partial \log P\left(\mathbf{w}^k\right)}{\partial \mathbf{w}^k} \boldsymbol{\epsilon}^k \right\}. \tag{2.1}$$

Assuming that the prior distribution is Gaussian with mean zero variance σ_1 and the weights $\mathbf{w}$ are sufficiently sampled, (2.1) becomes

$$\begin{aligned}\frac{\partial L^C(\boldsymbol{\theta})}{\partial \sigma_j} &= \mathbb{E}_{\epsilon_j \sim \mathcal{N}(0,1)}\left[-\frac{1}{\sigma_j} - \frac{\partial \log P(\mathbf{w})}{\partial \mathrm{w_j}}\epsilon_j\right] \\ &= \mathbb{E}_{\epsilon_j \sim \mathcal{N}(0,1)}\left[-\frac{1}{\sigma_j} - \frac{-(\mu_j + \sigma_j \epsilon_j)}{\sigma_1^2}\epsilon_j\right] \\ &= \mathbb{E}_{\epsilon_j \sim \mathcal{N}(0,1)}\left[\frac{\sigma_j}{\sigma_1^2}\epsilon_j^2\right] - \frac{1}{\sigma_j} \\ &= \frac{\sigma_j}{\sigma_1^2} - \frac{1}{\sigma_j}. \end{aligned} \tag{2.2}$$

When $\frac{\partial L^C(\boldsymbol{\theta})}{\partial \sigma_j} = 0$ for certain j, parameter ρ_j is stable. This condition is satisfied by

$$\begin{aligned}\frac{\partial L^C(\boldsymbol{\theta})}{\partial \sigma_j} &= 0 \\ \sigma_j &= \sigma_1. \end{aligned} \tag{2.3}$$

Therefore, parameter ρ_j continues to change toward the hyperparameter value of the standard deviation σ_1 of the prior following the evaluation of the softplus function. This corresponds to the results shown in [11] when the weights $\mathbf{w}$ are marginalized. In practice, the weights are sampled only once per iteration; however, similar results can be obtained for sufficiently large iterations. Moreover, the problem is not too large to consider the changing trend of parameter ρ_j. This result indicates that the update of parameter ρ_j is significantly influenced by the design of the prior distribution. However, if the hyperparameters of the Gaussian prior distribution are not set explicitly, $\sigma_1 = 1$ is obtained regardless of the coefficients of $L^C(\boldsymbol{\theta})$.

2.2.2 Decoupled Bayes by backprop

The loss of BBB is very noisy; thus, it is desirable to use Adam, which is suitable for the stable convergence of noisy loss functions[31]. As observed, $L^C(\boldsymbol{\theta})$ causes the update of parameter ρ_j to be strongly dependent on the standard deviation of the prior distribution. For example, if the initial value of parameter ρ_j is small, it continues to increase until σ_j reaches σ_1. In this situation, the gradient estimate by

Adam causes parameter ρ_j to increase extremely rapidly. Consequently, an unnecessarily rapid increase in parameter $\boldsymbol{\rho}$ may prevent the convergence of parameter $\boldsymbol{\mu}$. Although the use of Adam is undesirable for parameter $\boldsymbol{\rho}$ because the sign of its gradient rarely changes until the parameter update stabilizes, it is necessary to update parameter $\boldsymbol{\mu}$ owing to the adoption of a noisy cost function. This motivated the proposal of AdamB in this study, inspired by AdamW [32]. Similar to AdamW, for the BBB cost function, Adam is evaluated only for the NLL gradient $\mathbf{g}_t$. Subsequently, using the gradient estimate $\hat{\mathbf{g}}_t$ evaluated by Adam, the gradients of the variational parameters are as follows:

$$\boldsymbol{\Delta}_{\boldsymbol{\mu}} = \hat{\mathbf{g}}_t - \lambda_d \nabla_{\mathbf{w}} \log P(\mathbf{w}) \tag{2.4}$$

$$\boldsymbol{\Delta}_{\boldsymbol{\rho}} = \left\{ \boldsymbol{\Delta}_{\boldsymbol{\mu}} \boldsymbol{\epsilon} - \lambda_d \frac{1}{\boldsymbol{\sigma}} \right\} \frac{1}{1 + \exp\left(-\boldsymbol{\rho}_{t-1}\right)}, \tag{2.5}$$

where λ_d denotes a decoupled regularization factor that is replaced by $\frac{1}{M}$ in (1.6). The update of the parameter $\boldsymbol{\mu}$ in BBB corresponds to the weight $\mathbf{w}$ in the MAP estimation. λ_d corresponds to λ in the MAP estimation; however, it is a hyperparameter that does not reflect the standard deviation of the prior distribution. As previously observed, the properties of the gradients of parameters $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ are different. Thus, the need to use the same learning rate is eliminated. The learning rates of $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ were set as α_μ and α_ρ. Therefore, the update of the variational parameters using AdamB becomes

$$\boldsymbol{\mu}_t = \boldsymbol{\mu}_{t-1} - \alpha_\mu \boldsymbol{\Delta}_{\boldsymbol{\mu}} \tag{2.6}$$

$$\boldsymbol{\rho}_t = \boldsymbol{\rho}_{t-1} - \alpha_\rho \boldsymbol{\Delta}_{\boldsymbol{\rho}}. \tag{2.7}$$

In this study, the initial values of the variational parameters were obtained from Xavier's initial values[57] as follows:

$$\mu_j^{\text{init}} \sim \mathcal{N}\left(0, \sqrt{2/\left(n_{\text{in}} + n_{\text{out}}\right)}\right) \tag{2.8}$$

$$\sigma\left(\rho_j^{\text{init}}\right) = \delta\sqrt{2/\left(n_{\text{in}} + n_{\text{out}}\right)}, \tag{2.9}$$

where δ is the initialization scale factor and n_{in} and n_{out} are the numbers of input and output neurons, respectively, at each layer in the neural network.

2.2.3 Gaussian scale mixture prior for stable optimization

AdamB stabilizes the convergence of the noisy cost function and avoids the rapid increase in parameter $\boldsymbol{\rho}$. However, it does not suppress the increase in the parameter $\boldsymbol{\rho}$ itself. As is evident from (2.3), for parameter $\boldsymbol{\rho}$ to decrease, the variance of the prior distribution must be set to a sufficiently small value. The simplest way to accomplish this is to set the variance of the prior distribution to be small. However, this approach results in a strong regularization of parameter $\boldsymbol{\mu}$, which may adversely affect convergence. This problem can be solved using the SM prior $P_{\text{SM}}(\mathbf{w})$:

$$P_{\text{SM}}(\mathbf{w}) = \prod_{j=1} \{a\mathcal{N}\left(\mathrm{w}_j \middle| 0, \sigma_1^2\right) + (1-a)\,\mathcal{N}\left(\mathrm{w}_j \middle| 0, \sigma_2^2\right)\},$$

where a is a constant, such that $0 < a < 1$. σ_1^2 and σ_2^2 are the variances of the Gaussians. Here, σ_1 was set as a long-tailed Gaussian and σ_2 as $\sigma_2 \ll 1$ such that the weights were densely concentrated around zero. This log-likelihood is

$$\begin{aligned}
\log P_{\text{SM}}(\mathbf{w}) &= \sum_{j=1}^{J} \log \left\{ a \frac{1}{\sqrt{2\pi}\sigma_1} \exp\left(-\frac{\mathrm{w}_j^2}{2\sigma_1^2}\right) \right. \\
&\quad \left. + (1-a)\,\frac{1}{\sqrt{2\pi}\sigma_2} \exp\left(-\frac{\mathrm{w}_j^2}{2\sigma_2^2}\right) \right\} \\
&= \sum_{j=1}^{J} \left[\log \frac{a}{\sqrt{2\pi}\sigma_1} - \frac{\mathrm{w}_j^2}{2\sigma_1^2} \right. \\
&\quad \left. + \log \left\{ 1 + \frac{1-a}{a} \frac{\sigma_1}{\sigma_2} \exp\left(-\frac{\mathrm{w}_j^2}{2\sigma'^2}\right) \right\} \right], \qquad (2.10)
\end{aligned}$$

where $\sigma'^{-2} = \sigma_2^{-2} - \sigma_1^{-2}$. The transformation of the last equation appears unnecessary; however, the derivatives must be evaluated correctly. The derivatives of the Gaussian mixture are moderately complex, resulting in implementation failure [10]. Further, modern deep learning frameworks[58] [59] tend to miss the backward numerical stability because they rely on automatic differentiation to evaluate

derivatives. The log-likelihood of SM diverges in the region where the tails of both Gaussians are zero. Thus, despite the addition of a small value for $P_{\mathrm{SM}}(\mathbf{w})$, a region where the gradient is zero will still exist, and consequently, the regularization of the parameters will fail.

2.3 Robustness of AdamB with SM prior

The performance of AdamB in combination with the SM prior was evaluated using the image recognition datasets CIFAR-10, CIFAR-100 [60], and TinyImageNet [61]. The neural network models were evaluated by using ResNet [54] and VGG [53]. In addition, covariate shift benchmarking was carried out using the corresponding corruption datasets [35]. All experiments were conducted by completing 210 epochs with a batch size of 256 and the learning rate scheduler was cosine annealed [32]. For all experiments using BBB, $\delta = 1.0$ and $\alpha_\rho = 0.001$. When Adam is used as the optimizer, $\alpha_\mu = 0.001$, $\lambda_d = 0.01$, otherwise (SGD) $\alpha_\mu = 0.1$, $\lambda_d = 5.0 \times 10^{-4}$. For MAP estimation, SGD has a momentum of 0.9 and $\lambda = 5.0 \times 10^{-4}$; AdamW had $\lambda = 1.0 \times 10^{-2}$. During training, data augmentation was applied to the input data. A horizontal flip was used in all experiments. For CIFAR, the image was padded with four pixels and then cropped to 32×32. Finally, for TinyImageNet, the image was padded with eight pixels and then cropped to 64×64. No data augmentation was used during testing.

Effects of Decoupling

First, the convergence of parameters was evaluated by decoupling the BBB loss; the momentum SGD and Adam by decoupled BBB are denoted as D-mBBB and AdamB, respectively. Further, to enable the optimizers to be compared, the vanilla BBB and non-decoupled momentum SGD (mBBB) and Adam (BBB-Adam) as well as vanilla optimization with momentum SGD and AdamW were employed. Subsequently, the NLL, accuracy, and log posterior distribution $\log q\left(\mathbf{w}|\boldsymbol{\theta}\right)$ were plotted for each epoch (Figure 2.2). In the evaluation of the accuracy and NLL with the test dataset, the weights $\mathbf{w}$ were not sampled; however, the parameter $\boldsymbol{\mu}$ was used in-

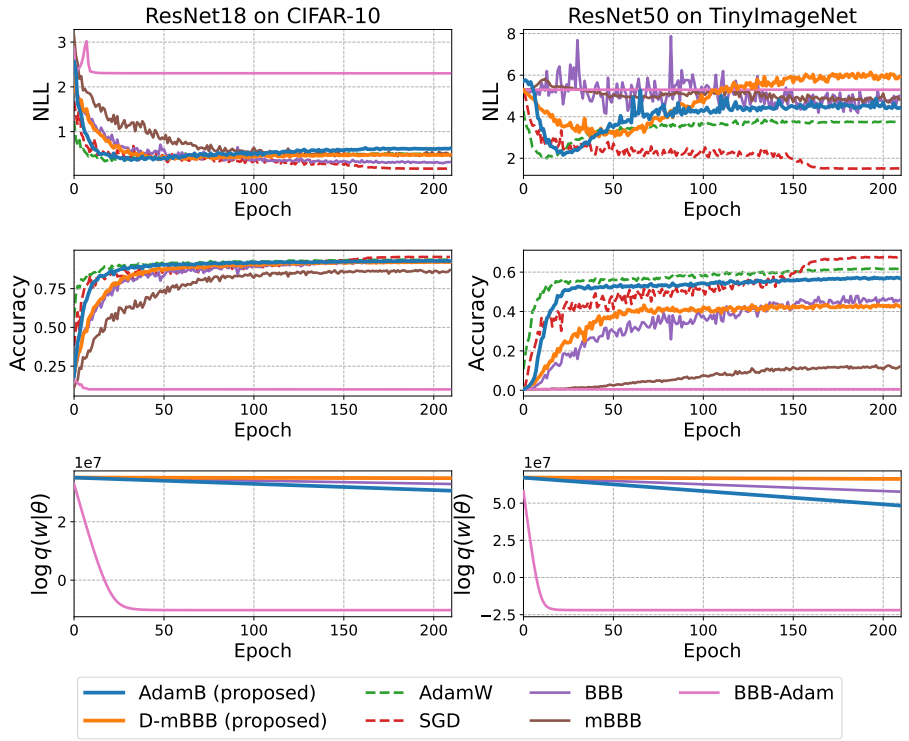

Figure 2.2: Learning curve of DNNs trained by BBB (solid curves) or MAP estimation (dashed curves). NLL curves (top), generalization curves (middle), and log-likelihood of the posterior distribution curves (bottom).

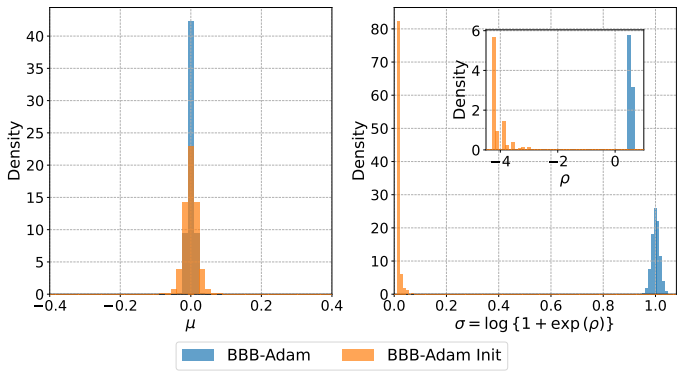

Figure 2.3: Transition of BBB parameter distribution when using Adam estimator. The parameter μ is concentrated near zero compared to the initial distribution (left). The parameter ρ concentrates on values for which σ is distributed near 1.0 (right).

stead. BBB can converge in ResNet and VGG on CIFAR-10, regardless of whether it is optimized by SGD. However, the training fails in ResNet and VGG because the learning rate is shared by variational parameters [46][55]. SGD tends to set a larger learning rate than Adam, and the update step size for the parameter $\boldsymbol{\rho}$ was probably excessive. In the experiments that were conducted, a distinction was

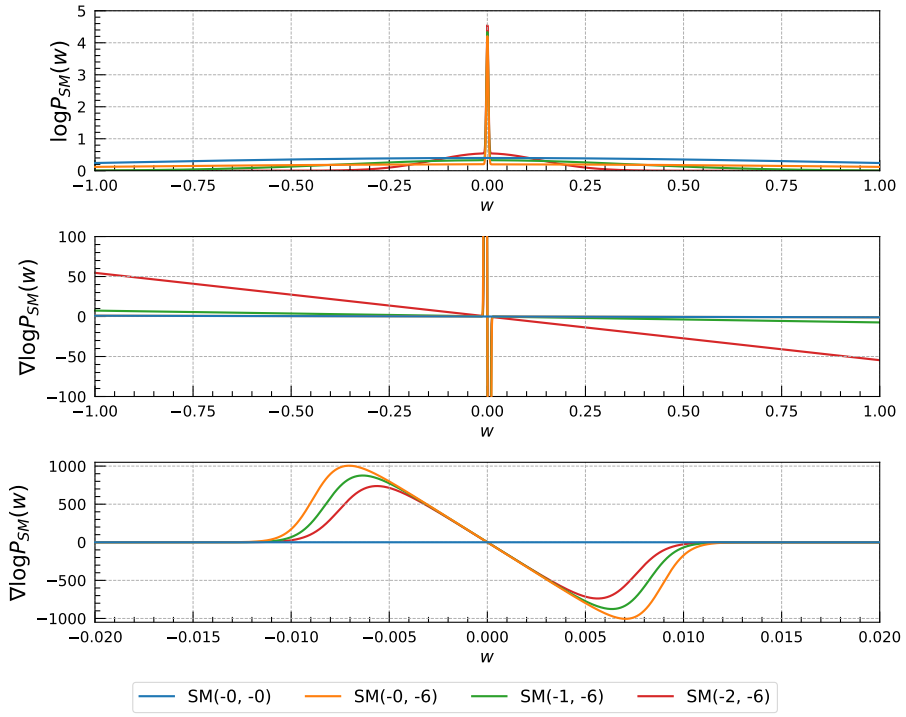

Figure 2.4: Log-likelihoods of the Gaussian scale mixture prior distributions (top) and their derivatives (middle, bottom). The derivatives extend over a wide range (middle) and are concentrated near zero (bottom). The hyperparameters of the Gaussian scale mixture are represented as $\mathrm{SM}(\log \sigma_1, \log \sigma_2)$. The other hyperparameter is $a = 0.5$.

made between α_μ and α_ρ and thus, successful convergence with SGD on BBB was realized. In TinyImageNet, which is difficult to use for learning, BBB with SGD did not notably increase the accuracy; however, AdamB achieved the same level of accuracy as MAP estimation. Evidently, BBB-Adam failed to learn. $\log q\left(\mathbf{w}|\boldsymbol{\theta}\right)$ in BBB-Adam decreased rapidly until it reached a specific value, implying a rapid increase in parameter $\boldsymbol{\rho}$. Because Adam tends to have a large step size when the sign of the gradient remains the same, this causes a rapid increase in $\boldsymbol{\rho}$. Figure 2.3 shows the distribution of the parameters for the initial values in ResNet18 and the values after training CIFAR-10 with BBB-Adam. Parameters $\boldsymbol{\mu}$ and $\boldsymbol{\rho}$ are evidently concentrated at zero and approximately 1.0, respectively, following evaluation using the softplus function. As Adam accelerated the update of parameter $\boldsymbol{\rho}$, it can be considered to have stabilized at the same value as in (2.3).

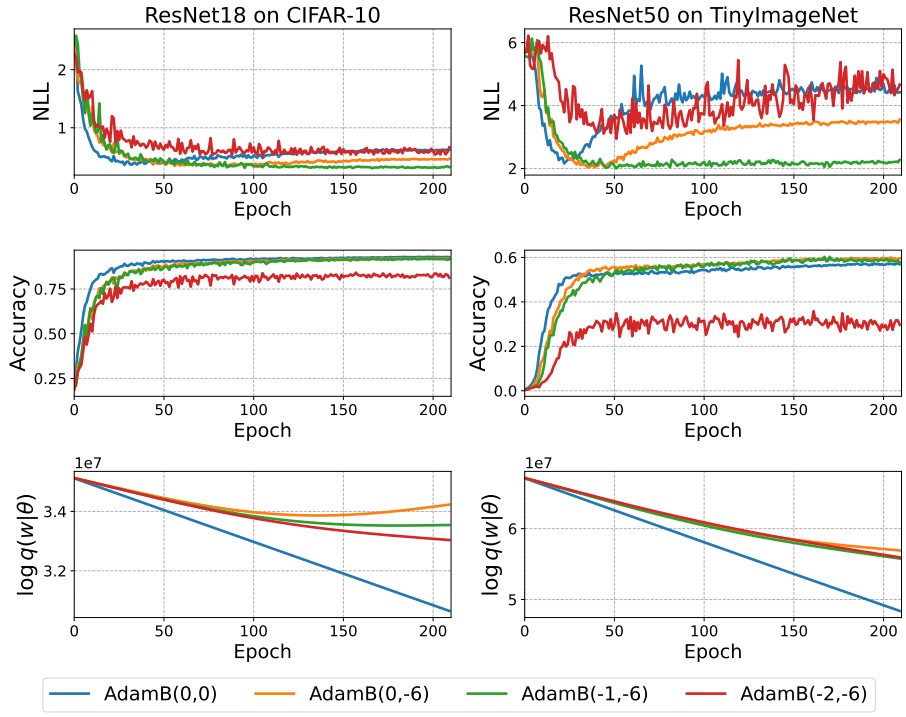

Figure 2.5: The learning curve of DNNs trained by BBB with different priors. NLL curves (top), generalization curves (middle), and log-likelihood of the posterior distribution curves (bottom).

Gaussian Scale Mixture Prior

As AdamB facilitates stable parameter convergence, a detailed evaluation of the effects of the SM prior is possible. In this experiment, the convergence of the variational parameters was examined using the SM prior. The hyperparameters of the SM prior were based on the values used in the BBB study [37]; that is, the hyperparameters are $a = 0.5$, $(\log \sigma_1, \log \sigma_2) \in \{(0,0), (0,-6), (-1,-6), (-2,-6)\}$. When $(\log \sigma_1, \log \sigma_2) = (0,0)$, the prior distribution is identical to a standard Gaussian distribution.

Figure 2.4 shows plots of the prior distribution and the gradient used in this experiment: σ_1 weakly regularizes over a wide range of widths, whereas σ_2 strongly regularizes at approximately zero. The experiment was repeated by changing only the prior to AdamB (Figure 2.5). AdamB in Figure 2.2 and AdamB (0,0) in Figure 2.5 were identical. The NLL significantly improved with AdamB (-1,-6) owing to the stronger regularization using the SM prior. Additionally, the SM prior exhibited a more moderate decrease or increase in $\log q(\mathbf{w}|\boldsymbol{\theta})$ than the Gaussian prior. This

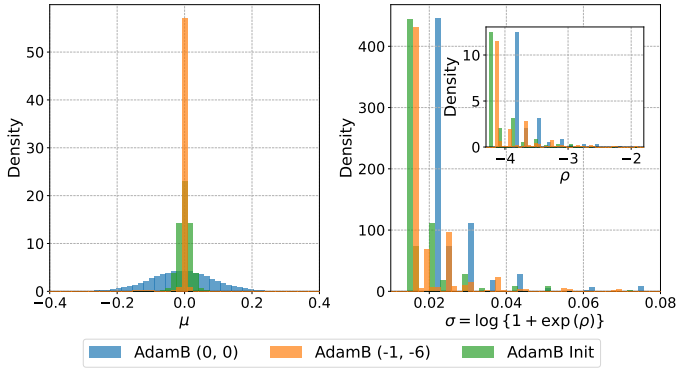

Figure 2.6: Transition of the BBB parameter distribution for different prior distributions when using AdamB. The parameter μ has a wider distribution when AdamB (0, 0) is used; however, the parameter values are concentrated near zero for AdamB (-1, -6) (left). The parameter ρ tends to increase consistently for AdamB (0, 0); however, it increases unevenly for AdamB (-1, -6). (right).

indicates that parameter $\boldsymbol{\rho}$ is controlled by the SM prior. Figure 2.6 shows the distribution of the initial parameters of ResNet18 and the parameters after training CIFAR-10 with AdamB (0,0) or AdamB (-1,-6). Despite the introduction of the SM prior to control the parameter $\boldsymbol{\rho}$, this made no clear difference and the change from the initial value was minimal. In contrast, a significant difference was observed in the parameter $\boldsymbol{\mu}$ in that its values were strongly concentrated near zero for AdamB (-1,-6); however, AdamB (0,0) had a wider distribution than the initial value. The AdamB (0,0) result is consistent with the fact that the weights optimized by AdamW are not concentrated at zero [62]. When a Gaussian distribution is used as a variational posterior distribution, the properties of the model may be considered by estimating the signal-to-noise ratio ($\mathrm{SNR} = 10\log_{10}|\mu_j|/\sigma_j$) from the variational parameters[11][37]. Figure 2.7 shows the SNR histograms and cumulative distribution functions (CDFs) for the convolutional (Conv) and fully connected (FC) layers from ResNet18 on CIFAR-10. The SM prior of AdamB (-1,-6) regularized parameter $\boldsymbol{\mu}$ such that it was concentrated near zero, thereby rendering the perturbation by parameter $\boldsymbol{\rho}$ as relatively strong. In addition, even with the SM prior, the FC layer for classification did not have an extremely small SNR compared to the Conv layer. Figure 2.8 shows that, percentage-wise, the small SNRs in the convolutional layer increase toward the output layer. This indicates that the perturbations become progressively stronger. Additionally, the SNRs smaller than zero continue to

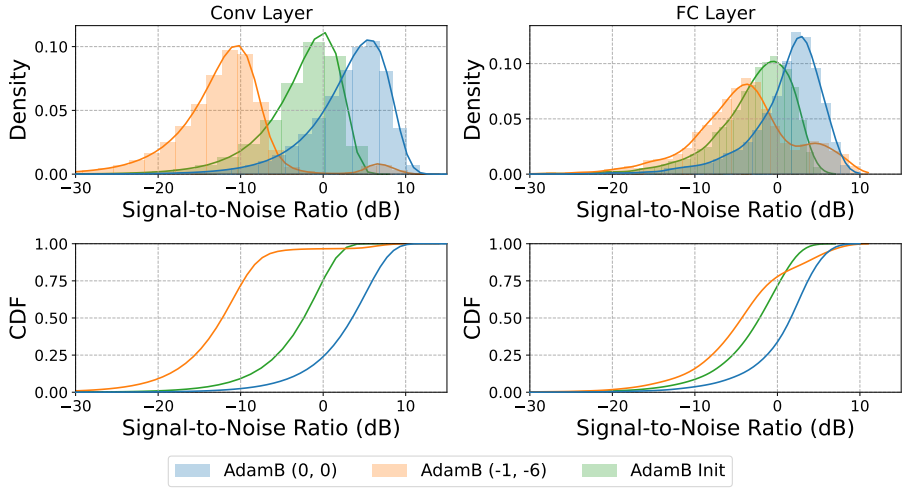

Figure 2.7: Comparison of SNR by varying the prior in AdamB. Histogram of SNR (top) and cumulative distribution function (CDF) of SNR (bottom).

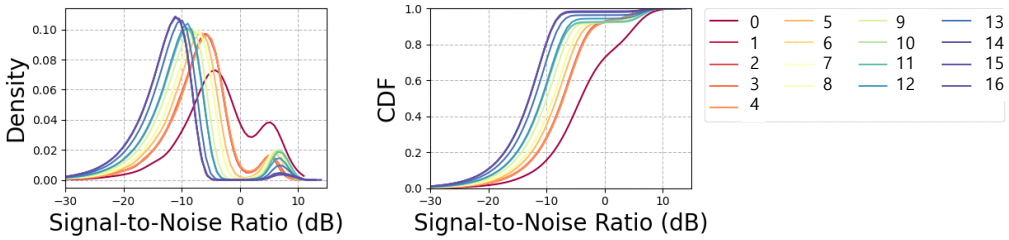

Figure 2.8: Distribution of SNRs in each convolutional layer of ResNet18 in the AdamB (-1,-6) prior (except for the 1×1 filter, which is a dimensionality adjustment layer). Histogram of SNR (left) and cumulative distribution function (CDF) of SNR (right). The graph numbers increase with the depth of the convolutional layer.

become increasingly smaller(Figure 2.9). In contrast, tshe SNRs greater than zero continue to gradually increase, suggesting that some parameter $\boldsymbol{\mu}$ is increasing. This increasing parameter $\boldsymbol{\mu}$ can be thought of as searching a parameter space that is robust to perturbations by other weights.

Table 2.1 lists the performance of the different datasets, neural network models, and optimization methods. The accuracy, area under the ROC curve (AUROC), and expected calibration error (ECE)[3], which quantifies the quality of the prediction uncertainty, were evaluated. The AUROC was evaluated using a multi-class one-vs.-one scheme[63]. When neural network models are used, not all outputs are useful for decision making and those deemed unreliable are removed by setting a threshold[7] [51]. Here, the confidence threshold τ was evaluated as $\tau = 0.6$. The sample rate is the percentage of outputs with a confidence threshold. In this study, evaluations

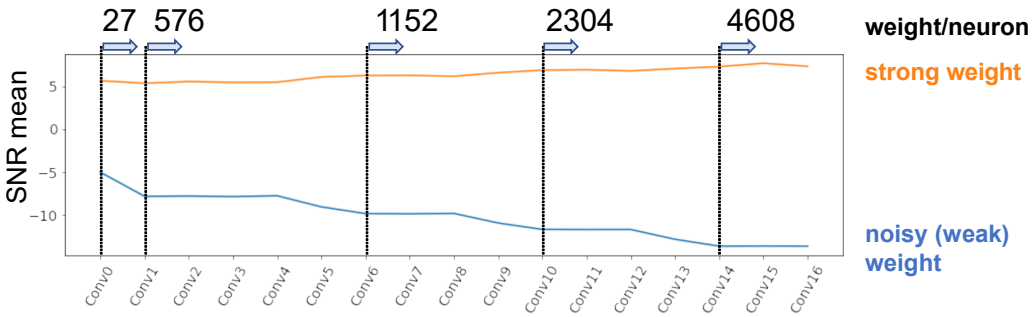

Figure 2.9: Changes in the respective mean values when fitting the SNR distribution in AdamB (-1,-6) with two Gaussian distributions. The numbers above the graph are the number of weights per neuron in each convolutional layer.

were conducted using both a single model and an ensemble. In the case of AdamB, 20 patterns of weights $\mathbf{w}$ were generated by MC sampling from the variational parameters and evaluated as an ensemble. For SGD, 20 models with different initial values were prepared and evaluated as deep ensembles. Further, for AdamB, when the ensemble was not evaluated, the parameter $\boldsymbol{\mu}$ was treated as the weight $\mathbf{w}$ and the performance was evaluated. Overall, the performance of AdamB was not as good as that of the SGD ensemble. This may be attributed to the fact that AdamB is designed to explicitly model the parameter uncertainty, which seems to make it difficult to improve the performance when evaluating the in-domain dataset. When training on TinyImageNet, AdamB (-1,-6) was more accurate using the threshold than the SGD ensemble; however, the number of valid samples was extremely small, and it is not necessarily considered a good result.

Table 2.1: Performance comparison on clean data

Dataset	Model	Optimizer	All Examples			Confidence threshold $\tau = 0.6$			
			Accuracy	AUROC	ECE	Accuracy	AUROC	ECE	Sample Rate(%)
CIFAR-10-C	VGG16	SGD	93.98 / **95.48**	0.996 / **0.998**	0.044 / **0.003**	94.69 / **97.12**	0.996 / **0.999**	0.043 / **0.003**	98.8 / 96.5
		AdamB(0,0)	91.31 / 91.14	0.994 / 0.995	0.073 / 0.023	91.68 / 94.01	0.994 / 0.996	0.073 / 0.021	99.1 / 94.2
		AdamB(-1,-6)	89.93 / 89.08	0.993 / 0.992	0.044 / 0.102	92.31 / 96.75	0.995 / 0.998	0.042 / 0.086	95.3 / 77.8
	Res-Net18	SGD	95.46 / **96.39**	0.998 / **0.999**	0.025 / **0.004**	96.23 / **97.57**	0.998 / **0.999**	0.024 / **0.003**	98.3 / 97.2
		AdamB(0,0)	92.77 / 92.80	0.996 / 0.996	0.061 / 0.031	93.19 / 94.24	0.996 / 0.997	0.060 / 0.031	99.2 / 96.8
		AdamB(-1,-6)	91.58 / 91.22	0.996 / 0.995	0.048 / 0.074	93.13 / 96.73	0.996 / 0.998	0.045 / 0.062	97.3 / 84.4
CIFAR-100-C	VGG16	SGD	74.42 / **78.77**	0.986/ **0.994**	0.130 / **0.023**	81.79 / 90.94	0.991/ **0.997**	0.130 / **0.019**	86.0 / 76.5
		AdamB(0,0)	66.47 / 66.49	0.980 / 0.984	0.257 / 0.115	70.04 / 79.99	0.983 / 0.991	0.256 / 0.105	92.6 / 73.2
		AdamB(-1,-6)	64.95 / 64.18	0.985 / 0.985	0.140 / 0.117	76.99 / **93.38**	0.991 / **0.997**	0.142 / 0.114	74.9 / 38.2
	Res-Net18	SGD	77.62 / **81.59**	0.990 / 0.995	0.053 / 0.066	89.83 / **94.25**	0.995 / **0.998**	0.039 / **0.026**	74.7 / 70.9
		AdamB(0,0)	68.79 / 68.66	0.985 / **0.996**	0.233 / 0.133	72.52 / 79.66	0.988 / 0.992	0.231 / 0.122	92.4 / 78.9
		AdamB(-1,-6)	70.06 / 68.91	0.989 / 0.988	0.172 / **0.050**	76.65 / 91.07	0.992 / 0.997	0.169 / 0.054	86.7 / 54.3
Tiny-ImageNet-C	Res-Net50	SGD	47.81 / **54.90**	0.955 / **0.977**	**0.072** / 0.094	79.42 / 89.67	0.977 / **0.992**	0.098 / **0.039**	39.6 / 34.4
		AdamB(0,0)	35.17 / 37.14	0.939 / 0.946	0.495 / 0.200	39.34 / 61.68	0.943 / 0.964	0.521 / 0.214	84.6 / 41.2
		AdamB(-1,-6)	38.98 / 36.35	0.950 / 0.952	0.308 / 0.091	52.43 / **92.07**	0.960 / 0.987	0.351 / 0.166	62.2 / 7.1

Each value represents a single/ ensemble model. In the case of multiple models, 20 models were used. For SGD, the values for multiple models are those evaluated by deep ensembles. For AdamB, the value for the single model was evaluated by the parameter μ, that is, without sampling. Multiple models were gathered into ensembles by sampling 20 times. [a]The TinyImageNet-C dataset does not correspond to the original TinyImageNet test or validation dataset. The values in this table were evaluated on the TinyImageNet-C dataset before corruptions were applied.

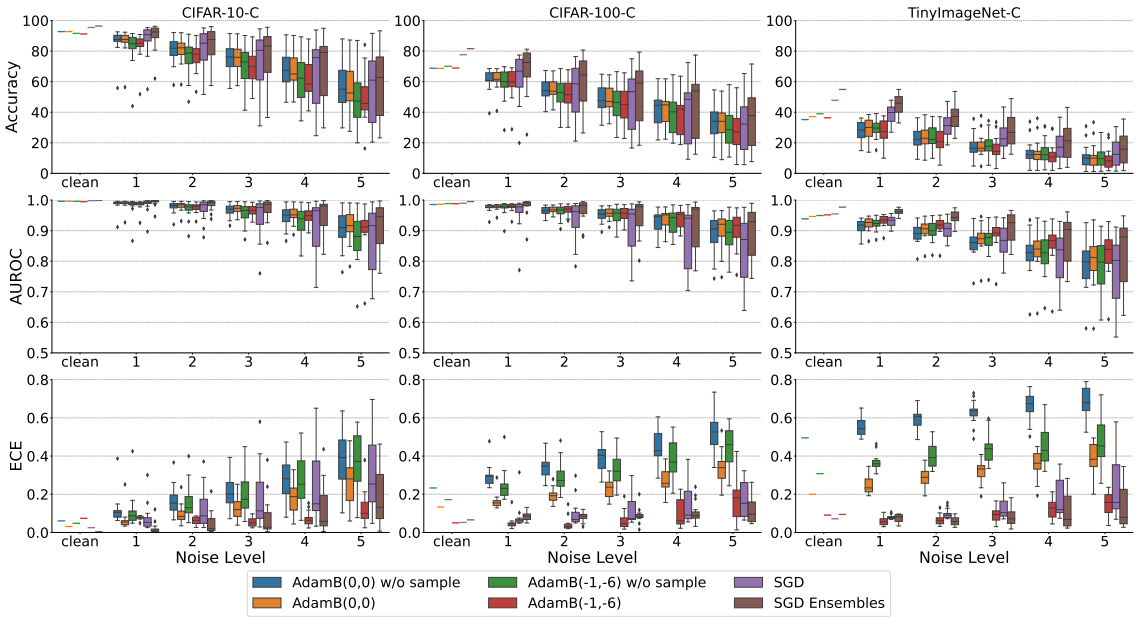

Figure 2.10: Predictive uncertainty evaluation under covariate shifts. Five levels of corruption intensity and 16 corruption types were used, the same as in [13]. CIFAR-10-C and CIFAR-100-C were evaluated with ResNet18, and TinyImageNet was evaluated with ResNet50. Accuracy, AUROC, and ECE were evaluated for each corruption level and are shown as box plots. Clean means that no covariate shift occurred in the dataset. Each box summarizes the results of 16 different corruption type evaluations.

Covariate Shift Benchmark

In real-world settings, the input data undergo covariate shifts from the training data. Therefore, the robustness of the model to various covariate shifts [13] must be evaluated. Covariate shift evaluation was conducted using CIFAR-10-C, CIFAR-100-C, and TinyImageNet-C as corruption datasets [35]. Figure 2.10 shows the results of the covariate shift benchmark under the same conditions as [13]. In this benchmark, a single model and an ensemble of 20 models were evaluated. Under conditions where the models are well trained, as in CIFAR-10, AdamB (-1,-6) expressed higher predictive uncertainty than the SGD ensemble. Overall, AdamB (-1,-6) was strongly robust to various covariate shifts, as shown by the small variations in the box according to the corruption type in AUROC and ECE. For easily trainable datasets, such as CIFAR-10, AdamB (-1,-6) expressed a higher predictive uncertainty compared to the SGD ensemble. However, for hard-to-learn datasets, such as TinyImageNet, the accuracy was clearly inferior. This can be improved by tuning the hyperparameters for AdamB; however, as the purpose of this experiment was

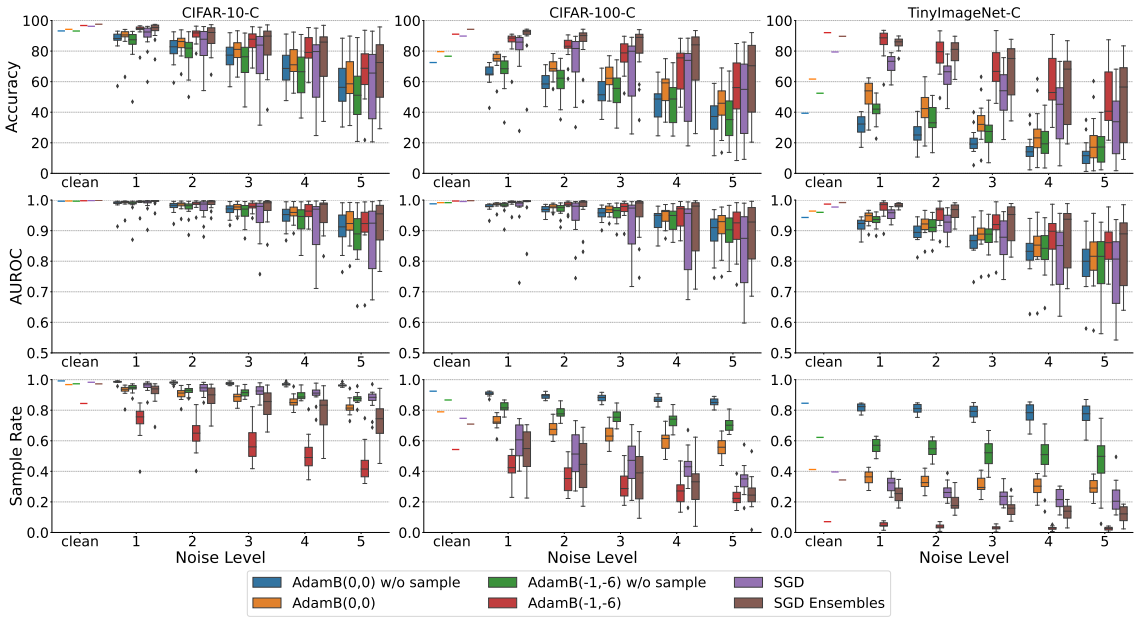

Figure 2.11: Predictive uncertainty evaluation under covariate shifts with confidence threshold $\tau = 0.6$. CIFAR-10-C and CIFAR-100-C were evaluated with ResNet18, and TinyImageNet was evaluated with ResNet50. In contrast to the evaluation in Figure 2.10, here the percentage of valid samples instead of the ECE was evaluated.

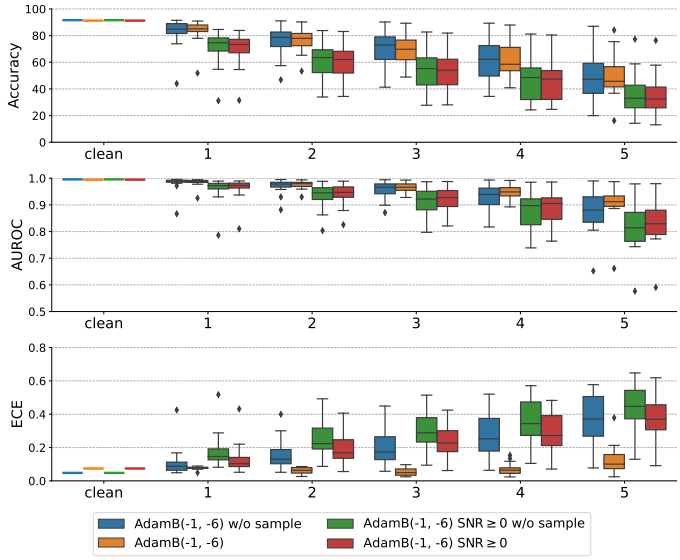

Figure 2.12: Comparison of covariate shift evaluation by pruning weights based on SNR of posterior distribution. Results are shown for ResNet18 on CIFAR-10 trained by AdamB (-1,-6). Covariate shift evaluation when weights are sampled only when the SNR of the variate parameter in the posterior distribution is greater than 0 dB. In other words, weights are pruned if the SNR is less than 0 dB.

to evaluate the performance of AdamB under normal MAP estimation training conditions, the experimental conditions were not changed. Covariate shift benchmarks

can also be evaluated by setting a confidence threshold value. Figure 2.11 shows the results of the evaluation with a confidence threshold $\tau = 0.6$. AdamB (-1,-6), whose good predictive uncertainty can be confirmed by the low ECE, achieved high accuracy with the threshold setting, whereas AUROC did not result in any noticeable change. In particular, in TinyImageNet, AdamB (-1,-6) dramatically improved the accuracy. However, the sample rate is so low that only a few inputs are accepted for decision-making. First, the redundancy of the variational parameters learned by using AdamB (-1,-6) was examined. The accuracy of BBB may be maintained even if the posterior distribution of small SNRs is pruned (replaced by zero weights)[37]. As shown in Figure 2.7, the SNR of the posterior distribution learned using AdamB (-1,-6) was mostly less than 0 dB. Figure 2.12 shows the results of the covariate shift benchmark when all posterior distributions are used and when only posterior distributions with SNR ≥ 0 are used (i.e., $w_j = 0$ when SNR < 0). Without the covariate shift (clean), the performance did not differ significantly. However, when covariate shifts occurred, the robustness degraded because of SNR-based weights pruning. This indicates that the seemingly unnecessary low SNR posterior distribution has an important function in ensuring robustness against covariate shifts.

Figure 2.13 shows the performance for each corruption type for corruption intensity of 4. As is evident, the models were generally less robust to covariate shifts in the noise type; however, the predictive uncertainty of AdamB (-1,-6) was superior to that of the other approaches. This confirms that the improvement in Accuracy (Figure 2.11) using AdamB (-1,-6) with confidence threshold $\tau = 0.6$ is mainly effective for the noise type (Figure 2.14). Further, the addition of Gaussian noise lowered the accuracy of all the models, and this was examined thoroughly. Table 2.2 lists the performance for Gaussian noise type corruptions under the same conditions as in Table 2.1. Overall, AdamB achieved more accurate confidence calibration than the MAP estimation approach. Thresholding greatly improved the accuracy but tended to reduce the number of valid samples. This result reflects the fact that the confidence for inputs that are recognized as difficult to classify is reduced to small values. Next, the number of valid test data samples and the change in accuracy and AUROC for confidence thresholds of 0, 0.3, 0.6, and 0.9 were examined for

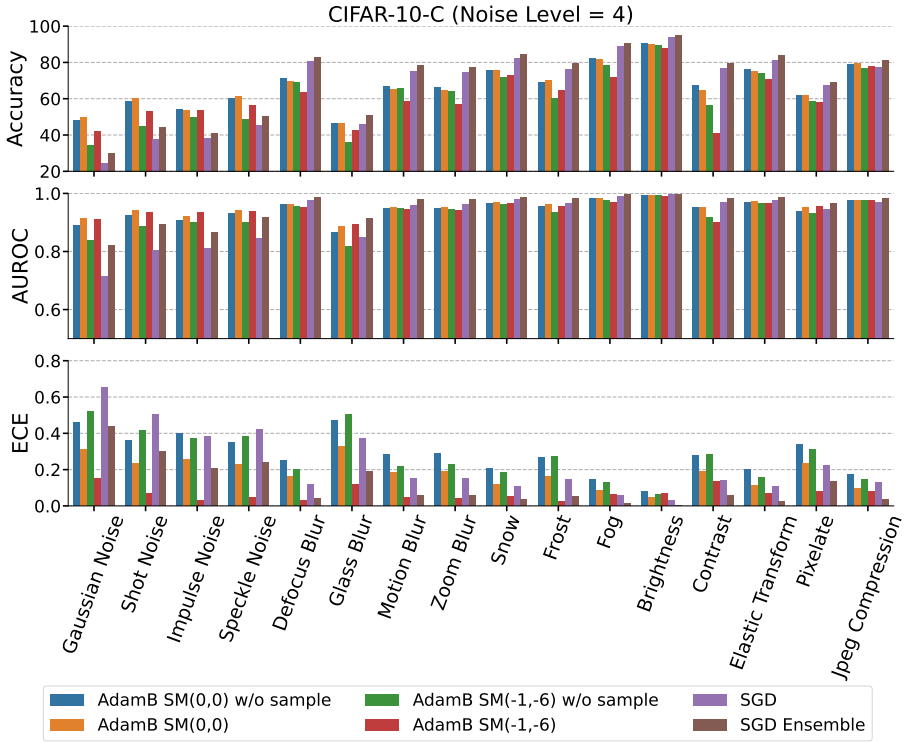

Figure 2.13: Evaluation of the covariate shift for each corruption type at corruption level 4. Accuracy, AUROC, and ECE were evaluated using models trained with ResNet18 on CIFAR-10.

data with no covariate shift (clean) and with Gaussian noise intensities of 2 and 4. Figure 2.15 shows the results of these evaluations and histograms of the information entropy. For clean data, SGD and the SGD ensemble delivered high performance; however, an increase in the intensity of the Gaussian noise resulted in a performance improvement owing to thresholding becoming poorer. In contrast, AdamB (-1,-6) maintained high performance with MC sampling ensembles and thresholding, even in the presence of high-intensity Gaussian noise. The ensemble results with MC-sampling of AdamB (-1,-6) showed a noticeable shift in information entropy to the right compared to the other methods. Thus, the predictive distribution widened with an increase in the covariate shift, indicating that the prediction uncertainty was captured.

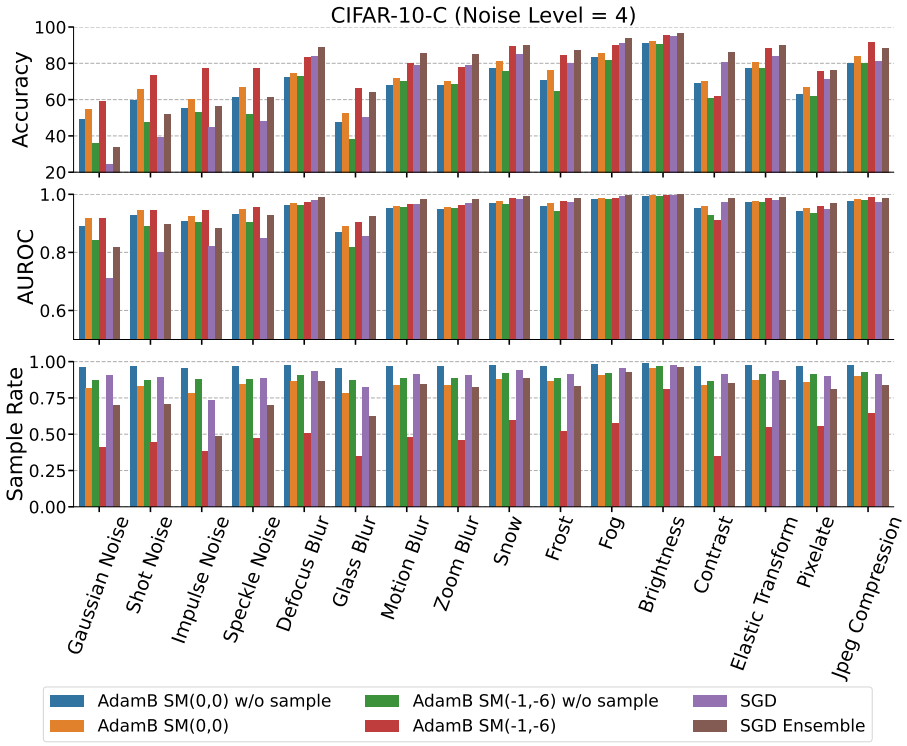

Figure 2.14: Evaluation of the covariate shift for each corruption type at corruption level 4 with confidence threshold $\tau = 0.6$. Accuracy, AUROC, and the sample rate were evaluated using models trained with ResNet18 on CIFAR-10.

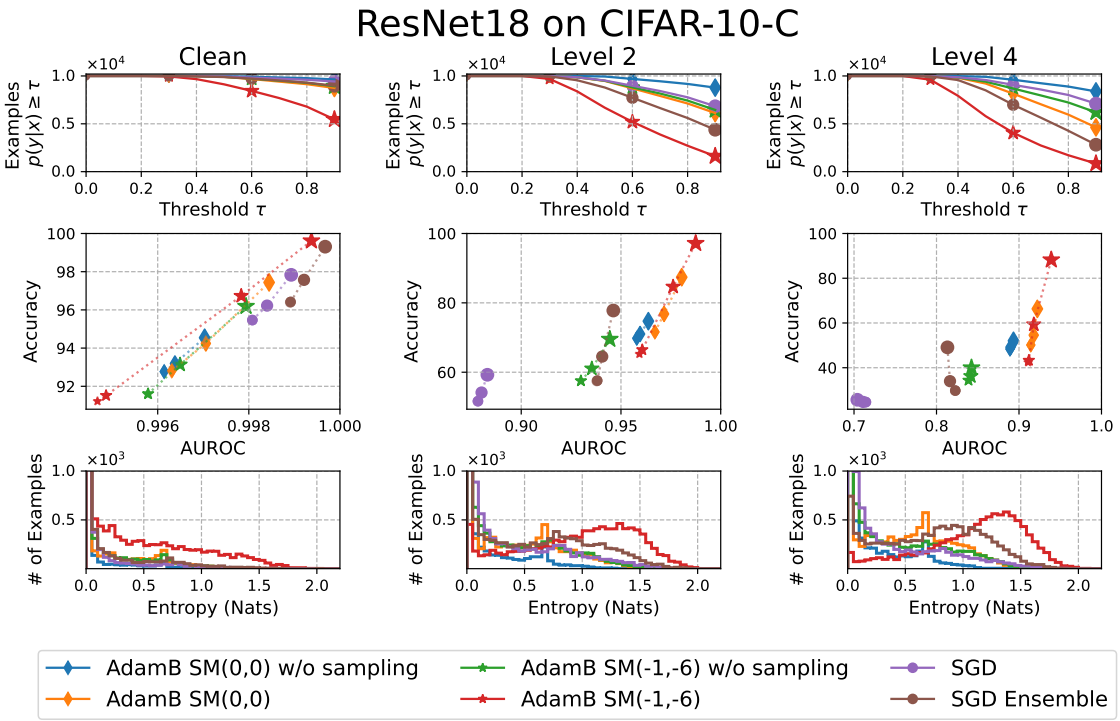

Figure 2.15: Estimation of detailed predictive uncertainty of ResNet18 on CIFAR-10 for clean data and Gaussian noise levels 2 and 4. The number of valid samples (top) and plots of Accuracy and AUROC (middle) when the predictive confidence threshold is set at 0.3, 0.6, and 0.9. The information entropy was evaluated over all samples (bottom).

Table 2.2: Performance comparison of corruption datasets with Gaussian noise level 4

Dataset	Model	Optimizer	All Examples			Confidence threshold $\tau = 0.6$			Sample Rate(%)
			Accuracy	AUROC	ECE	Accuracy	AUROC	ECE	
CIFAR-10-C	VGG16	SGD	32.03 / 30.28	0.740 / 0.825	0.584 / 0.444	33.14 / 32.79	0.740 / 0.815	0.611 / 0.521	91.5 / 72.2
		AdamB(0,0)	51.99 / **53.87**	0.881 / 0.913	0.425 / 0.249	53.12 / 60.80	0.883 / 0.918	0.432 / 0.271	95.9 / 76.7
		AdamB(-1,-6)	33.06 / 47.93	0.822 / **0.927**	0.511 / **0.070**	35.02 / **70.88**	0.821 / **0.934**	0.553 / **0.066**	85.0 / 34.5
	Res-Net18	SGD	24.66 / 29.76	0.714 / 0.823	0.650 / 0.436	24.73 / 34.00	0.711 / 0.816	0.691 / 0.499	90.7 / 69.9
		AdamB(0,0)	48.33 / **50.13**	0.889 / **0.914**	0.463 / 0.312	49.16 / 54.57	0.890 / **0.918**	0.472 / 0.338	96.0 / 81.5
		AdamB(-1,-6)	34.38 / 42.42	0.839 / 0.911	0.520 / **0.154**	36.21 / **59.33**	0.842 / **0.918**	0.558 / **0.187**	86.9 / 40.6
CIFAR-100-C	VGG16	SGD	16.02 / 19.20	0.743 / 0.828	0.562 / 0.326	19.23 / 31.35	0.739 / 0.807	0.683 / 0.472	67.0 / 34.0
		AdamB(0,0)	23.20 / **24.78**	0.850 / 0.882	0.603 / 0.336	26.09 / 39.21	0.854 / 0.891	0.647 / 0.418	83.1 / 43.3
		AdamB(-1,-6)	20.89 / 21.76	0.860 / **0.895**	0.414 / **0.113**	30.69 / **71.81**	0.867 / **0.917**	0.521 / **0.031**	51.6 / 7.8
	Res-Net18	SGD	13.33 / 14.68	0.708 / 0.769	0.322 / 0.239	20.81 / 26.00	0.683 / 0.709	0.613/ 0.505	31.9 / 19.3
		AdamB(0,0)	27.08 / **27.63**	0.871 / **0.890**	0.565 / 0.358	30.28 / 40.60	0.874 / **0.900**	0.605 / 0.425	83.2 / 51.3
		AdamB(-1,-6)	19.74 / 18.97	0.855 / 0.883	0.530 / **0.226**	24.40 / **43.28**	0.862 / 0.867	0.618 / **0.305**	67.7 / 16.8
Tiny-ImageNet-C	Res-Net50	SGD	8.04 / **8.57**	0.678 / 0.733	0.358 / 0.283	13.98 / 19.27	0.662 / 0.710	0.686 / 0.571	30.3 / 16.2
		AdamB(0,0)	8.36 / 8.38	0.784 / 0.800	0.763 / 0.450	9.12 / 14.63	0.787 / 0.806	0.819 / 0.630	85.0 / 34.5
		AdamB(-1,-6)	6.45 / 8.03	0.750 / **0.834**	0.669 / **0.171**	7.59 / **48.04**	0.755 / **0.838**	0.792 / **0.228**	68.5 / 2.0

Each value represents a single/ensemble model. In the case of multiple models, 20 models were used. For SGD, the values for multiple models are those evaluated by deep ensembles. For AdamB, the value for the single model was evaluated by the parameter μ, i.e., without sampling. Moreover, multiple models were gathered into ensembles by sampling 20 times.

Discussion

BBB assumes a single-mode posterior distribution in the parameter space. Therefore, this algorithm is less capable of capturing uncertainty than deep ensembles, which may provide a closer approximation to the true posterior distribution[12]. However, even with the assumption of a single mode in the posterior distribution, AdamB was more robust than deep ensembles for noise-type covariate shifts. With the SM prior now enabled by AdamB for large neural networks, perturbation of the weights by the extremely noisy posterior distribution appeared to be important for robustness in variational inference. BNNs, used in conjunction with the Hamiltonian Monte Carlo (HMC) method, do not assume a single mode in the posterior distribution, and are considered vulnerable to Gaussian noise corruption[64]. Although the data augmentation conditions are different from those in our experiment, the work provides valuable insight into the covariate shift of BNNs. To ignore neural network activations that are not needed for classification, the weight value for those activations should ideally be zero. BNNs tend to have non-zero weights because they model weights as a distribution, which induces false classification [64]. Even with AdamB, the variance of the prior distribution cannot be set to zero, so non-zero weights are sampled. However, AdamB (-1,-6) was confirmed to offer high robustness against Gaussian noise because parameter $\boldsymbol{\mu}$ was updated to values sufficiently large to ignore the noise in the posterior distribution. In this study, the hyperparameters were adjusted under high accuracy conditions in MAP estimation. Contrary to this, suitable hyperparameters in AdamB that can capture high quality predictive uncertainty still need to be found. Searching for these hyperparameters fell outside of the scope of this work because this study focused on a comparison of the properties of AdamB to those of MAP estimation under equal conditions. The predictive uncertainty of AdamB may be improved by combining variational inference and deep ensemble methods, as has been demonstrated with radial BNN [51]. In addition, AdamB has been evaluated only against the covariate shift benchmark on image classification; it has not been verified in other domains. Optimistically, AdamB should converge well for natural language processing and reinforcement learning tasks that have been verified to work well with AdamW[33].

2.4 Properties of AdamB in the classification layer

In the previous section, I considered whether stable convergence with AdamB is possible for standard convolutional neural networks (CNNs). The results suggest that a variational posterior distribution with high SNR has a significant impact on the classification of models trained with BBB. In this section, I review the properties of BNNs based on variational inference, focusing on a classification task with a simple fully connected neural network. In the case of BNNs that process weights as a distribution, the weights could respond to activations that do not contribute to the classification, resulting in incorrect identification. My investigation aimed to determine whether the same issue arises in models trained by AdamB.

PCA features with MNIST dataset

The experiments in this section are designed to evaluate the robustness of Bayesian neural networks to covariate shifts, as performed by Izmailov et al.[64]. Izmailov noted that images in the MNIST dataset [65] contain dead pixels around the objects in an image. These pixels rarely have values and do not contribute to classification. The covariate shift dataset utilizes vectors with Gaussian noise along the principal component (PC) for the MNIST dataset. They used the PCs with the 50 highest and 50 lowest variances (Top 50 and Last 50 PCs). Figure 2.16 shows examples of these components added to the MNIST test data in a scaled manner. In their experiment, MAP estimation does not differ significantly from BNNs in the case of the Top 50 PCs, but in the case of the Last 50 PCs, although MAP estimation does not degrade the accuracy, significant degradation occurs for BNNs. They confirmed that the weights of dead pixels that do not contribute to class classification are set to zero by regularization in the case of MAP estimation, whereas the posterior distribution converges to match the prior distribution in the case of BNNs. The Last 50 PC features are concentrated in the dead pixels, which they attribute to the non-zero values of the weights sampled by the BNNs, which have a negative impact on the inference results. My investigation aims to determine whether these tendencies are exhibited in the case of AdamB.

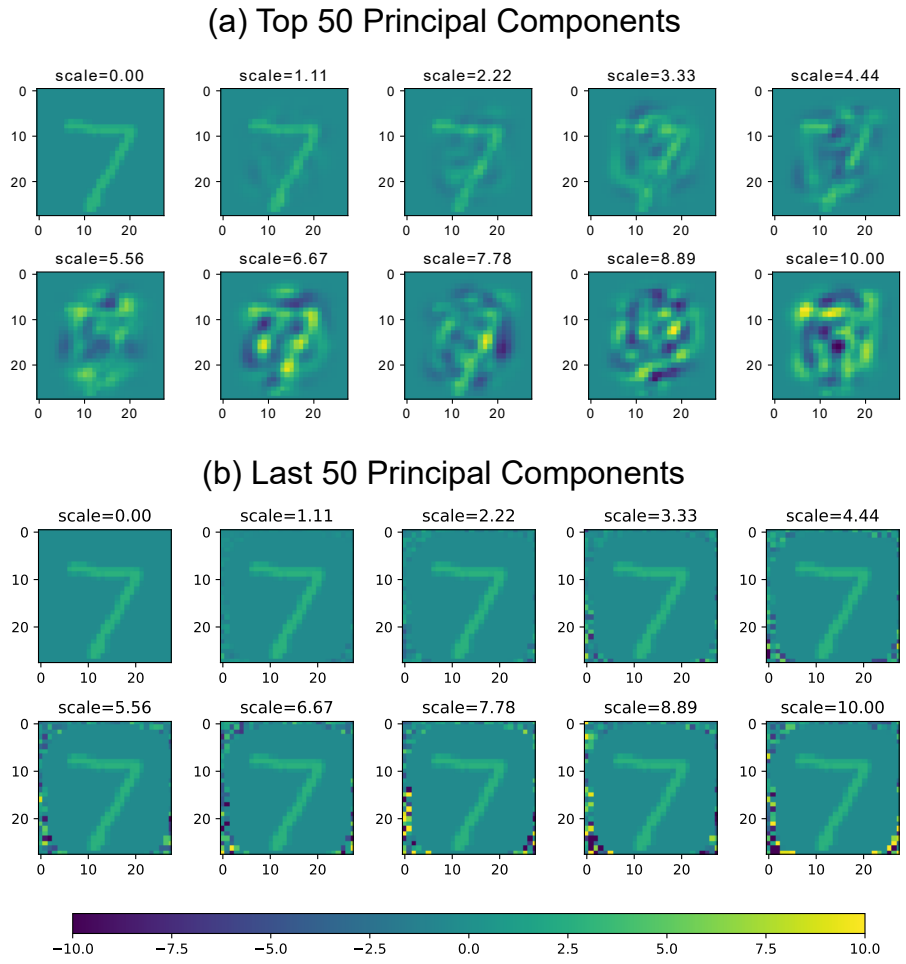

Figure 2.16: Data corruption created from eigenvectors of PCA taken from the MNIST training dataset

Sensitivity of variational parameters learned by AdamB to PCA features

This experiment uses a model of a fully connected neural network with two middle layers with 256 neurons. The activation function is a rectified linear unit (ReLU) [66]. The experimental conditions are the same as in the previous section, except that the number of epochs is set to 100 and no data augmentation is employed. Twenty models with different initial values were trained by SGD (i.e., MAP estimation) to form an ensemble during the inference evaluation. The dataset is MNIST, with 60,000 training values and 10,000 test values. Covariate shift evaluation using the Top 50 PCs and Last 50 PCs was conducted as reported by Izmailov et al. These features are scaled in 20 steps for the test dataset and added to evaluate the models

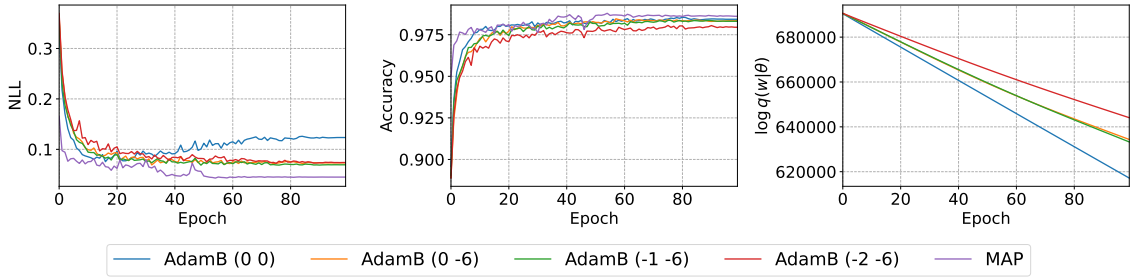

Figure 2.17: Learning curve of NNs trained by AdamB or MAP estimation. NLL curves (left), generalization curves (middle), and log-likelihood of the posterior distribution curves(right)

trained by AdamB and SGD.

Sensitivity of weights to PCA features

Figure 2.17 shows the metrics resulting from training. Similar to the trend described in the previous section, the Gaussian prior tends to result in overfitting as training progresses. The accuracy of AdamB (-2,-6) is also lower than that of the other models. The log-likelihood of the posterior distribution confirms that that of the SM prior model decreases less than that of the Gaussian prior. Figure visualizes the weights of the input layers trained by SGD. The ensemble represents the average of the weights projected onto the same neurons in the different models: the weights are distributed over the central part of the image where the MNIST data features are large, whereas the weights are almost zero for the peripheral regions of the image, regardless of the number of ensembles. On the other hand, the model trained by AdamB (0,0) with the usual Gaussian prior clearly has a value for the peripheral regions. This result shows the same trend as in the case of BNNs without the posterior distribution assumption: AdamB (-1,-6) also has weights for the regions surrounding the image, but they approach zero when the ensemble is used (Figure 2.19).

Figure 2.20 shows the distribution of SNR for the variational parameters; AdamB (-2,-6) tends to take small SNR values owing to the strong regularity resulting from the SM prior. On the other hand, AdamB (-1,-6) and AdamB (0,0) mostly have the same percentage of SNR as AdamB (-2,-6), but the variational parameters are

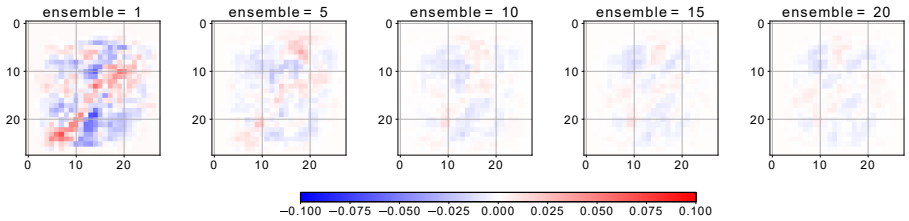

Figure 2.18: Weight heatmap in SGD ensembles

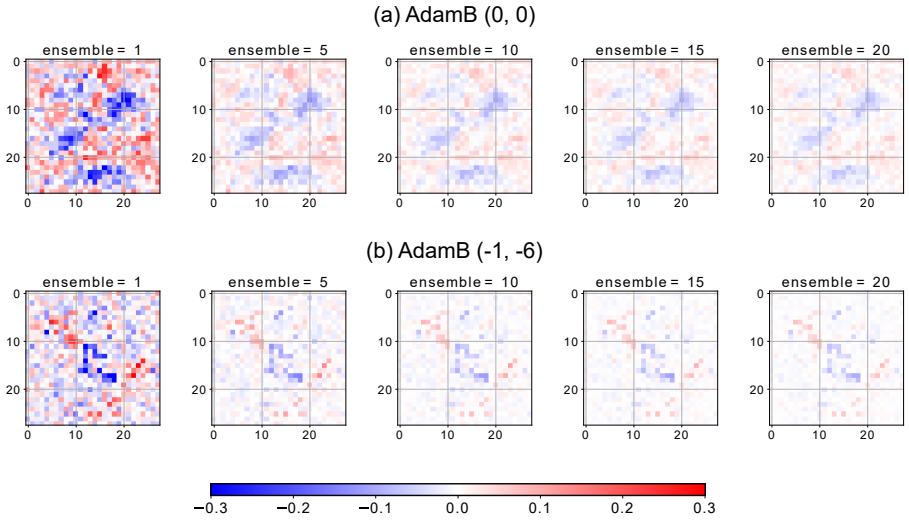

Figure 2.19: Weight heatmap trained by AdamB (0,0) (top), and AdamB (-1,-6) (bottom)

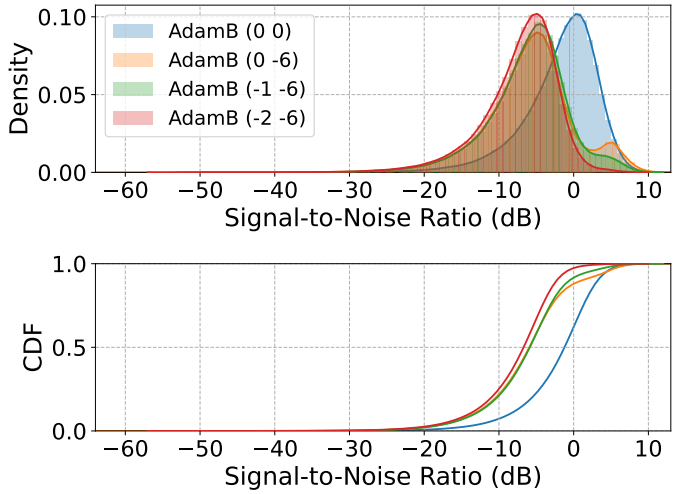

Figure 2.20: Comparison of SNR with different priors in AdamB. Histogram of SNR (top) and cumulative distribution function (CDF) of SNR (bottom).

observed to increase with SNR>0. Figure 2.21 shows the change in accuracy when mutations are added stepwise to the test dataset with the Top 50 PCs or Last 50

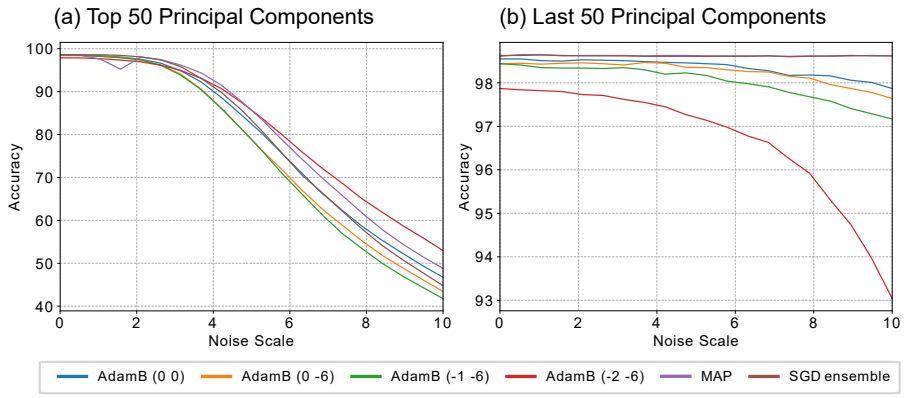

Figure 2.21: Change in accuracy by gradual addition of principal components: addition of Top 50 PCs (left), addition of Last 50 PCs (right).

PCs. For all conditions in the Top 50 PCs, the accuracy decreases as the mutation increases, but no significant difference is observed. On the other hand, when mutations with the Last 50 PC features are added stepwise, the accuracy of AdamB tends to be lower than that of MAP. This may be because AdamB has weight values for the image periphery, which affects the classification results. In particular, AdamB (-2,-6), which does not have a high SNR variational parameter, is clearly less accurate compared to other variations of AdamB. The sensitivities of the weights in the input layer of the model trained by AdamB were analyzed in greater detail. Figure 2.22(right) shows plots of the 2σ of the calculated variance σ of the value of the inner product of the weights and PCs, starting from the PC with the largest variance. In other words, the wider the width along the vertical axis, the greater the extent to which the activations of the neurons vary. As the PCs become smaller, the width becomes narrower (i.e., the variation decreases), which indicates that the weights of the PCs are approaching zero. Figure 2.22(left) plots the results in Figure 2.22(right) normalized with respect to the value calculated for the smallest PCA variance. The dashed line indicates the value of 1.0 on the vertical axis. The model trained by MAP estimation shows that the weights are more responsive to PCs with small variance. Conversely, the model trained by AdamB shows that the weights are less responsive to PCs with small variance. For AdamB (0,0), which was overfitted, and AdamB (-2,-6), which exhibited poor growth in Accuracy, the weights responded only as much as the smallest PC, with a PC of approximately

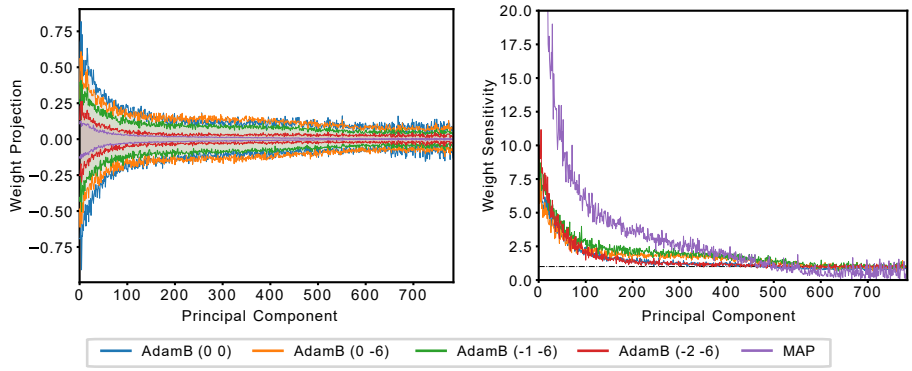

Figure 2.22: Projection of weights to PC. Variance of the activation values of neurons (left). Variance normalized by the smallest eigenvalue (right).

300. AdamB (0,-6) and AdamB (-1,-6), although weak, respond to PCs of as much as approximately 500, indicating that they are able to respond to more information.

Discussion

This series of experiments was designed to evaluate the behavior of the distribution of BNN parameters for values such as dead pixels that do not affect classification. MAP estimation was shown to successfully accommodate inputs that do not contribute to classification by assigning zero weights to them. AdamB processes the weight parameter as a distribution, and therefore has values by sampling, similar to a conventional BNN. The results presented in previous sections led to the observation that the design of an appropriate prior enables AdamB to control the SNR of variational parameters. I considered certain variational parameters to have larger SNRs, which makes it possible to ignore the effect of perturbation by noisy weights. In this experiment, small peaks were visible in the distribution with SNR¿0 for AdamB (0,-6) and AdamB (-1,-6). In fact, the evaluation of the Last 50 PCs revealed that AdamB (0,-6) and (-1,-6) were robust to perturbations because dead pixels were used as input, although not as robust as the MAP estimation. The fact that BNN has a value for weights that should be zero is considered a disadvantage, but the performance of AdamB is considered to be improved by its well-balanced SNR.

2.5 AdamB with low-precision Gaussian sampling

Training AdamB requires a large amount of random numbers to be generated. The main focus of improving the efficiency of random number generation with dedicated hardware is to reduce the precision required per variable. The performance changes that occur when the precision of the random numbers is reduced are considered in detail. Although efficient random number generation for BBB inference has received some attention[39], [40], the random numbers needed for training have not yet been investigated. This prompted an evaluation of the reliability and robustness against covariate shifts by gradually decreasing the precision of the uniform distribution used in generating the normal distribution using the Box-Muller transformation [67] from 32 bits. In this experiment, the parameters of the convolutional layer of VGG19, which is an overconfidence model, were fixed and the parameters of the FC layer were learned by AdamB to evaluate whether sufficient confidence calibration can be achieved. Evaluation of a dataset that makes it difficult to learn a confidence-calibrated model facilitates examination of the effect of random number precision.

Classification layer training on datasets that tend to be over-confident

The experiment made use of features obtained by using data from CIFAR-10, a benchmark dataset for object category recognition, as input into a CNN. The CNN uses the convolutional layer of VGG19 [53] without batch normalization [68], and the final CIFAR-10 feature dataset has 512 dimensions. Although VGG19 is a model trained on ImageNet [69], a benchmark dataset for object category recognition, the VGG19 convolutional layer for CIFAR-10 was not retrained for refinement purposes in this study. I used a training dataset of 50,000 CIFAR-10 features and a test dataset of 10,000 features. The shift in the dataset was evaluated by using the convolutional features of VGG19 on the CIFAR-10-C dataset with 15+4 different 5-level noises for the CIFAR-10 test dataset [35]. For OOD evaluation, I used the convolutional features of VGG19 on 10,000 test data of the SVHN dataset [70], which is a benchmark dataset for object category recognition.

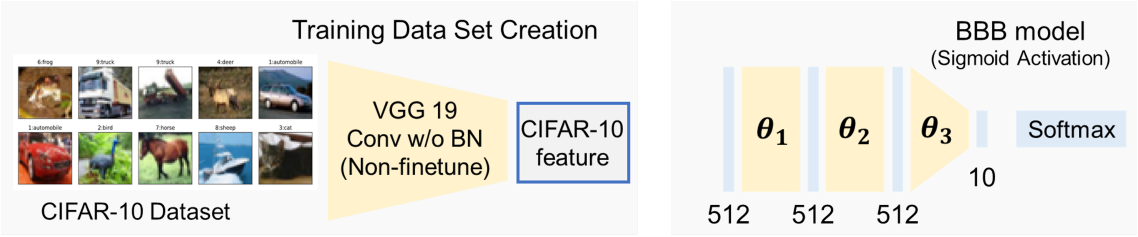

Figure 2.23: Experimental scenario

The training target by AdamB is a fully connected layer with two middle layers of 512 dimensions and an output layer of 10 dimensions, referring to the discriminative layer of VGG19, with activation by a sigmoid function. Figure 2.23 summarizes these experimental conditions. Apart from training AdamB, regular BBB and Adam were also trained, where AdamB is AdamB (0,-6). The values of the hyperparameters $\beta = \beta_1, \beta_2$ of the exponential moving averages of the gradient and the square of the gradient for Adam and AdamB were $\beta_1 = 0.9, \beta_2 = 0.999$. The batch size was set to 100, the number of epochs to 50, and the learning rates for Adam and BBB were 0.001 and 0.01, respectively. Additionally, the initial value of parameter $\delta = 0.5,\ 1.0,\ 2.0,\ 4.0$ was varied in the experiments. In the inference phase, the ensemble evaluation was performed by Monte Carlo sampling, and the average value of the confidence by multiple weight patterns was used to evaluate the accuracy and the ECE. The number of ensembles was set to 20. With reference to the parameters verified in AdamB, I again experimented with $32, 8, 4, 2, 1$ bits of precision per variable for uniform random numbers. To confirm in detail the impact of lowering the precision in random number generation, I performed a threshold evaluation for the data distribution shift and a covariate shift benchmark using the CIFAR-10-C feature dataset.

Dependency of the initial value of the variational parameter

The variation in the accuracy and NLL during training is plotted in Figure2.24. Only $\delta = 0.5$ is shown for BBB and Adam. Adam is overfitted until the accuracy increases sufficiently, whereas BBB is not overfitted, although it has not reached sufficient accuracy. In the case of AdamB, the final accuracy does not differ sig-

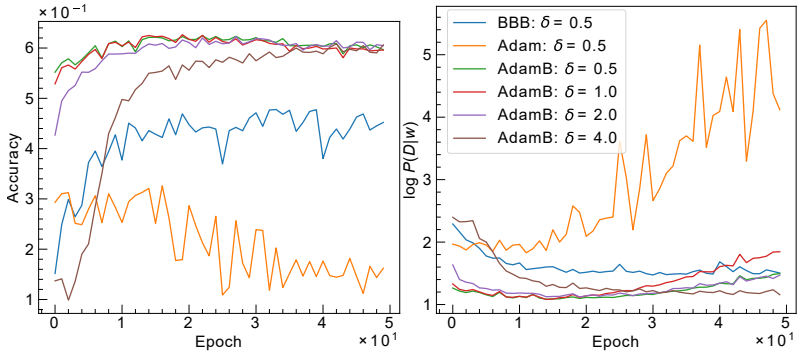

Figure 2.24: Learning curve of the VGG classification layer trained by AdamB with different priors or SGD. NLL curves (left), generalization curves (right).

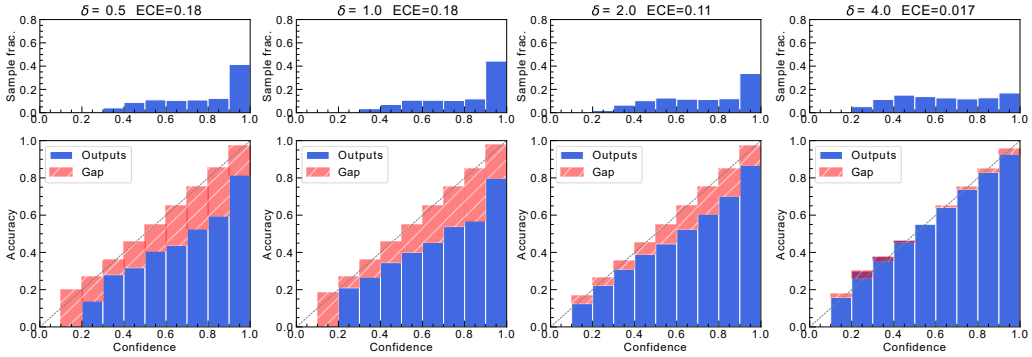

Figure 2.25: Initial value dependence of ECE

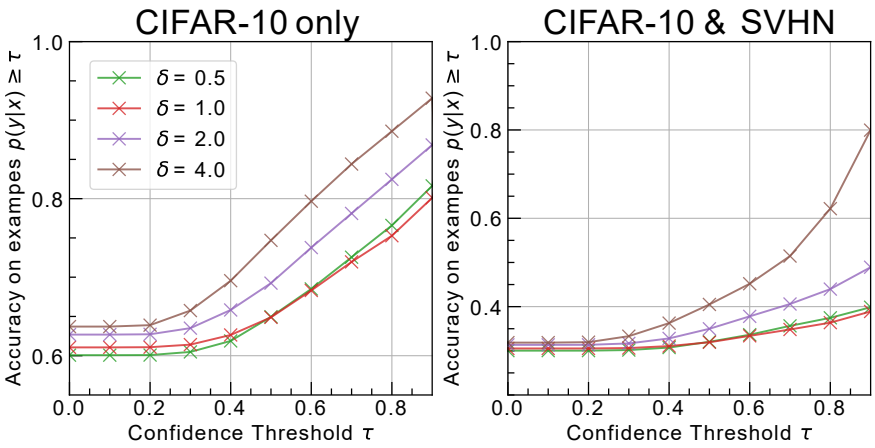

Figure 2.26: Changes in the accuracy using threshold settings with different initial values. In-domain dataset only (left). Combined OOD dataset (right).

nificantly regardless of the value of δ, except for $\delta = 4.0$, where the NLL increases and overfitting occurs. Figure 2.25 shows the confidence level of the accuracy for various values of the ECE with AdamB. The figure shows that the ECE improves as

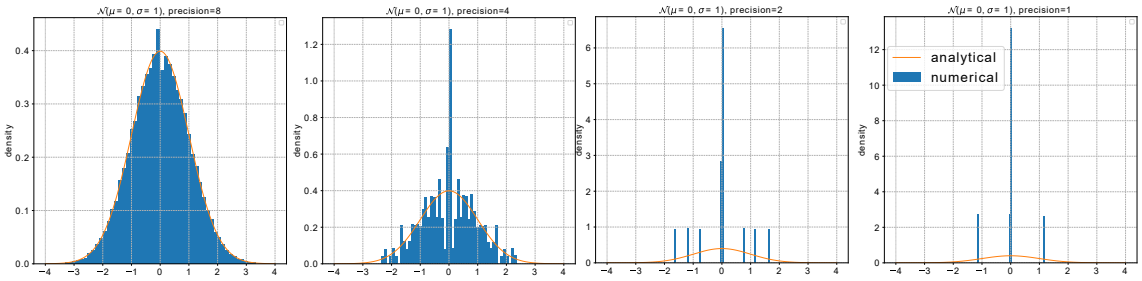

Figure 2.27: Low precision Gaussian samplings. From left to right: 8, 4, 2, and 1 bit per sampling.

δ increases. For small values of δ, the confidence level exceeds 0.9 in most cases, but the ECE is higher owing to overconfidence with low accuracy. On the other hand, when $\delta = 4.0$, the confidence and accuracy are approximately equal, indicating that the ECE has improved. In the classification layer, it is desirable to assign a larger initial value to the parameter ρ for features that tend to be overconfident. Figure 2.26 shows the change in the accuracy when the threshold is set using the variational parameters trained by AdamB. When evaluated with test data of CIFAR-10 features (Figure 2.26 left), the results confirmed that the increase in the value of δ results in an improved confidence calibration and thus a more favorable growth of the accuracy when the threshold value is set. The difference is particularly noticeable when the SVHN feature dataset without labels corresponding to the output layer is added to the inference dataset (Figure 2.26 right). The results confirmed that training with larger values of δ is robust against domain shift.

Classification layer training with low precision Gaussian sampling

Based on the previous experiment, we trained AdamB with $\delta = 4.0$, changing the precision of uniformly distributed random numbers to 32, 8, 4, 2, 1 bits. Figure 2.27 shows the results of sampling a normal distribution by Box-Muller transformation at different levels of random number precision for a uniform distribution. With 8 bits, sampling close to an ideal normal distribution is confirmed, but the use of 4 bits or fewer causes the sampling distribution to deviate from the normal distribution. The change in accuracy and NLL during training is shown in Figure2.28.

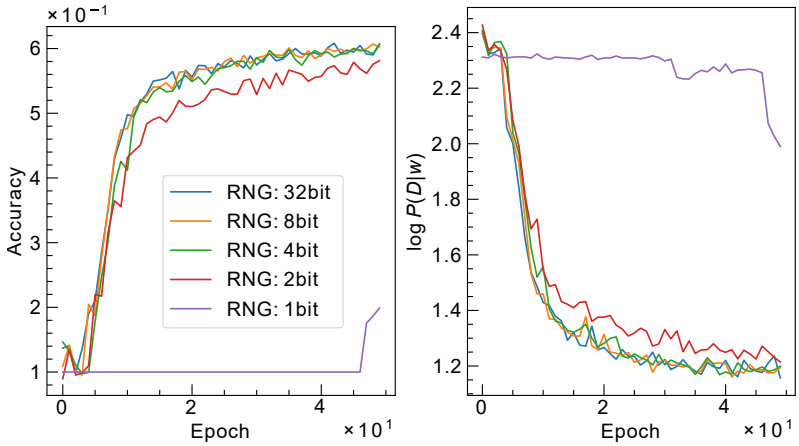

Figure 2.28: Learning curve of VGG classification layer trained by AdamB (0,-6) with different amounts of sampling precision. NLL curves (left), generalization curves (right).

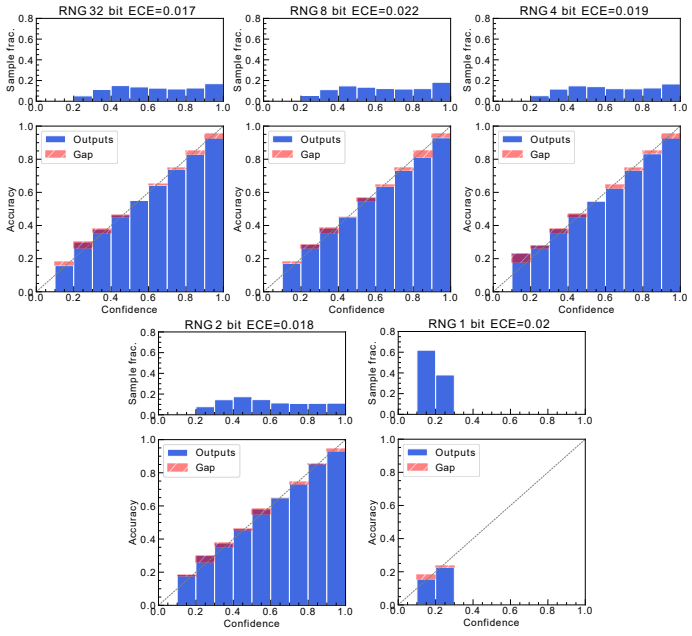

Figure 2.29: Sampling precision dependence of ECE

When the uniform distribution has a precision of 32, 8, or 4 bits, the accuracy and log-likelihood are not significantly affected, but a slight deterioration is confirmed for the 2-bit precision and a remarkable delay in convergence is confirmed at 1-bit. Under 2- and 1-bit conditions, the sampled distribution tends to have discrete values, which negatively affects the convergence of the parameters. On the other hand, for a precision of 4-bit, although the sampling distribution also deviates from the normal distribution, the convergence did not significantly differ from that under 32-

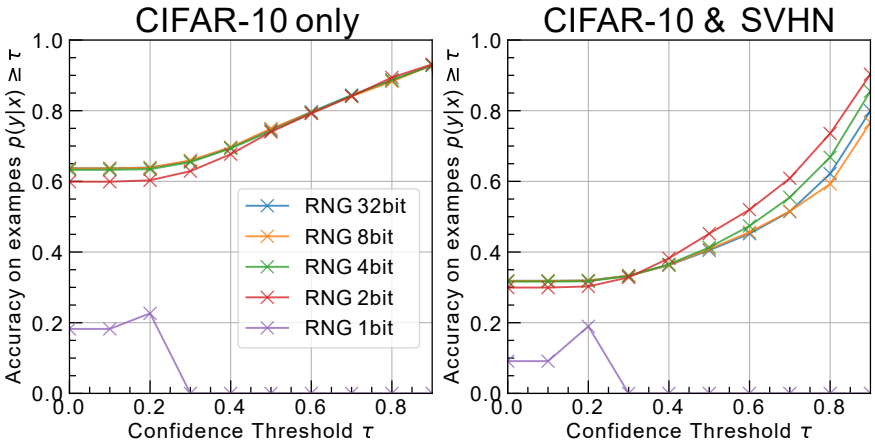

Figure 2.30: Accuracy changes with threshold settings with different sampling precisions. In-domain dataset only (left). Mixture of OOD dataset (right).

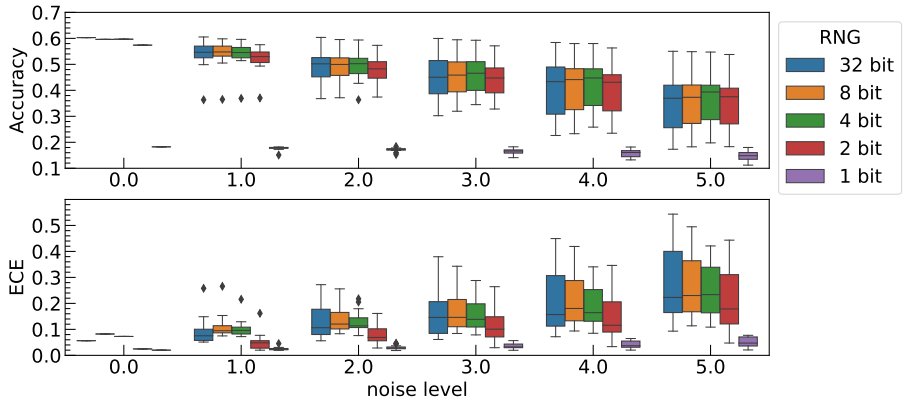

Figure 2.31: Predictive uncertainty evaluation under covariate shifts. Five levels of corruption intensity and 19 corruption types were used.

bit conditions. Figure 2.25 shows the value of ECE for different amounts of uniform precision. With the exception of 1-bit, the ECE values do not differ significantly, although for 2-bit precision, the percentage of high confidence levels is lower, indicating that this amount of precision is inferior to 4-bit precision or higher in terms of accuracy. Figure 2.30 shows the change in the accuracy when the threshold is set according to the precision of the random numbers. When evaluated with the test dataset of CIFAR-10 features (Figure2.30 left), no significant change is observed except for 1-bit precision. On the other hand, the results confirmed that the lower the precision (except for 1-bit) when SVHN features are added, the higher the accuracy that is maintained (Figure2.30 right). Figure 2.31 shows the results of the covariate shift benchmark in terms of the precision of the random number generation. The

smaller the precision of the random number generation, the smaller the ECE tends to be, but the accuracy also tends to be lower for 2- and 1-bit precision.

Discussion

The results confirmed that feature datasets extracted from the convolutional layer of overconfidence models show large differences in confidence calibration after training and that this depends on the initial value of δ of the variational parameter ρ. The accuracy of the test dataset does not depend on δ, but the accuracy differs markedly when the confidence levels are separated by bin. In addition, the ECE at high confidence levels tends to be smaller when δ is large. The histogram of confidence levels shows that the number of samples with high confidence decreases as δ increases. This can be interpreted to signify that the inference is well calibrated. A well-calibrated model shows a significant improvement in accuracy for thresholding and maintains the accuracy even when OOD inputs are added to the dataset. Effective learning with random numbers generated from a 32-bit uniform distribution neither gives rise to noticeable degradation in the OOD input or covariate shift benchmarks nor in the accuracy and ECE when the precision of the random numbers is changed to 4 bits or more. When thresholds are set for OOD inputs at 2-bit precision, the accuracy is maintained to a greater extent than at other precision levels. However, a certain degree of degradation in the accuracy and NLL is observed during learning, thus the use of random numbers with a precision of 4 bits or higher is preferable. Note that in this experiment, the evaluation of BBB-based algorithms using dedicated hardware should also take into account the change from single-precision floating-point format to data formats such as fixed point or bfloat.

Chapter 3

Movitan

3.1 Monte Carlo Variational Inference Training Accelerator for Neural Networks

The contribution of improvements in the exponential performance of computers has made it possible to increasingly scale neural networks for application to various domains for deep learning. Different types of dedicated hardware have been developed to perform deep learning more efficiently and the data flow of operations and memory capacity has been optimized according to the network model. Even though advances in deep learning continue unabated and new network architectures continue to be proposed, these architectures are not necessarily efficient with existing dedicated hardware. For applications that require risky decision making, such as self-driving and medical diagnostics, which are promising applications of deep learning, the inference results must be reliable. Despite Bayesian deep learning and deep ensemble methods being considered to be effective deep learning models for obtaining reliable inferences, evidence to this effect is not yet conclusive. Thus far, I have developed AdamB, which enables us to accommodate reliable deep learning models. Compared to ordinary MAP estimation, AdamB adds the log-likelihood of the variational posterior distribution to the cost function and generates weights using sampling with a normal distribution. This means that complex probability distribution calculations and sampling with a normal distribution of the order of the number of parameters of the NNs are necessary and this is computationally expensive. These challenges can be streamlined by using dedicated hardware to directly support an optimized data flow and lightweight random number generators. Dedicated hardware targeting Bayes by Backprop (BBB) inference, on which AdamB is based, focuses mainly on lowering the cost of the random number generator[39], [40]. In these studies, BBB training is assumed to be performed on other hardware and inference is performed by converting floating-point format to integer format. In contrast, my research targets BBB learning rather than inference only. Unlike inference, when learning by way of deep learning, it is difficult to simplify the workload of the neural network, and predicting the optimal hardware architecture in advance is challenging. Therefore, I decided to develop a system for generating dedicated

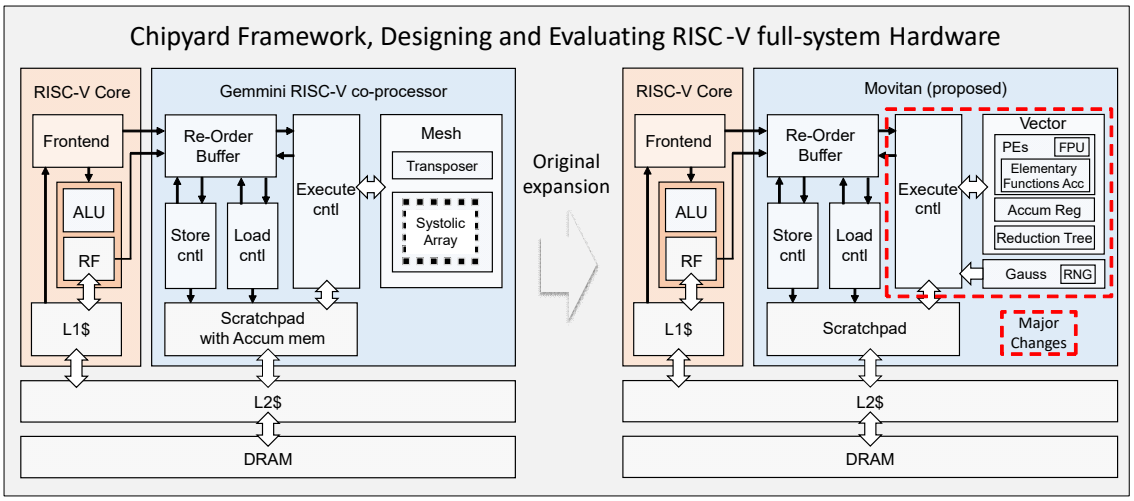

Figure 3.1: Overview of Movitan Extension

processors with dedicated circuits that enable efficient learning with AdamB, and build a system that can flexibly adapt to various deep learning architectures. Movitan (Monte Carlo Variational Inference Training Accelerator for Neural Networks) is a generating system based on Chipyard, an agile SoC development framework that enables efficient BBB training. The system generates instances of vector architectures that support Gaussian sampling flows and function approximation circuits optimized for BBBs tightly connected to RISC-V cores. Movitan is based on and extends Gemmini v0.2, a hardware generation system for deep learning supported by Chipyard3.1.

3.2 Hardware Architecture

Table 3.1 lists the main differences between Movitan and Gemmini. Gemmini v0.2 mainly targets inference, where matrix multiplication with fixed-length vectors is performed in one single step by systolic array hardware. Movitan flexibly simplifies the data flow of vector operations and random number generation, replacing them with simple vector type operators to facilitate the implementation of neural network training. In Gemmini, neural network activation relies on ReLU or ReLU6, whereas Movitan uses sigmoid and softplus functions as well as dedicated functions to efficiently calculate complex probability distributions such as a Gaussian scale

Table 3.1: Major differences between Gemmini v0.2 and Movitan

Module	function	Gemmini v0.2	Movitan
Execution Controller	PE architecture	Systolic array	Vector
	Tensorpose	supported	none
	Activations	ReLU, ReLU6	elementary function
	Accumulator place	near scratchpad	vector PEs
	scalar operation	none	supported
Re-Order Buffer	Data manager	fixed length	variable length

mixture. In addition, the Accumulator has a dedicated memory in Gemmini, but in Movitan, a register in the vector arithmetic unit plays this role. Scalar operations were not assumed in Gemmini v0.2, thus support for scalar operations were added to Movitan. The management of data dependencies in the reorder buffer has been changed to accommodate these changes.

Figure 3.6 shows the main block diagram for the operations in Movitan, which consists of an execution controller, a Gaussian sampling circuit, a vector unit, and a scratchpad. Movitan, similar to Gemmini, receives instructions from the CPU through the RoCC interface, and the execution controller mainly decodes the received instructions, makes data access requests to the scratchpad, and passes the data to the vector unit. Depending on the instruction, the input to the vector unit is switched by the arbiter to a value of zero or a sampled random number directly into the vector unit (Figure 3.3). The vector unit consists of a controller for broadcasting the input data, processing elements (PEs), and a reduction tree (Figure 3.4). The data input to the vector unit is passed to the internal PEs, which can broadcast any data element input just before it is passed to the PE, or switch the data elements to broadcast each cycle. The output from the PE can use the accumulator if necessary. The data computed by the vector unit is passed to the execution controller, which stores the data at the specified scratchpad address (Figure 3.5). The controller has been extended from the Gemmini v0.2 implementation to allow writing to each element of data when storing data in the scratchpad. Figure 3.6 shows the results of the predictive process design kit (PDK) for the academically available 7nm process, ASAP7[71], for Movitan and Rocket. The results are based on SoC floorplanning with Movitan and Rocket core at an operating frequency of

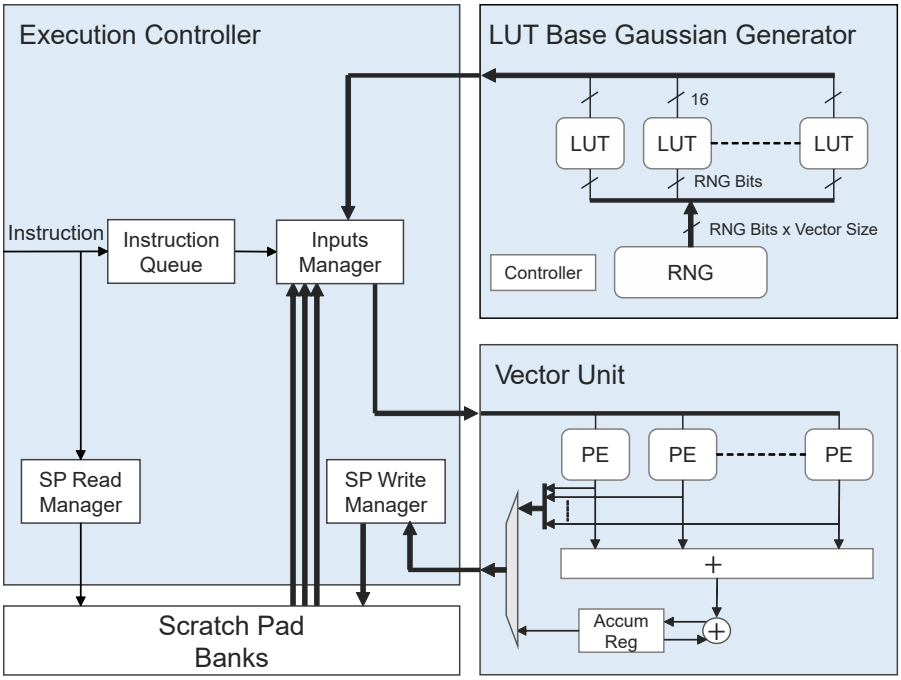

Figure 3.2: Movitan main block diagram

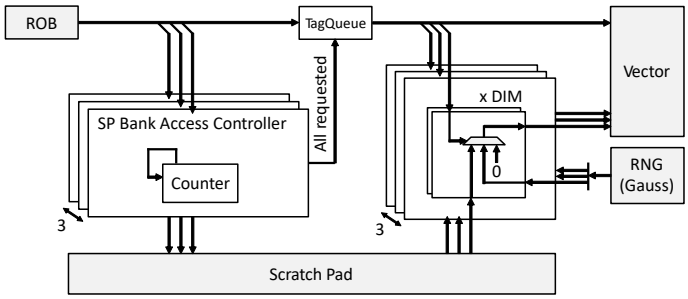

Figure 3.3: Execution controller to vector unit inputs flow

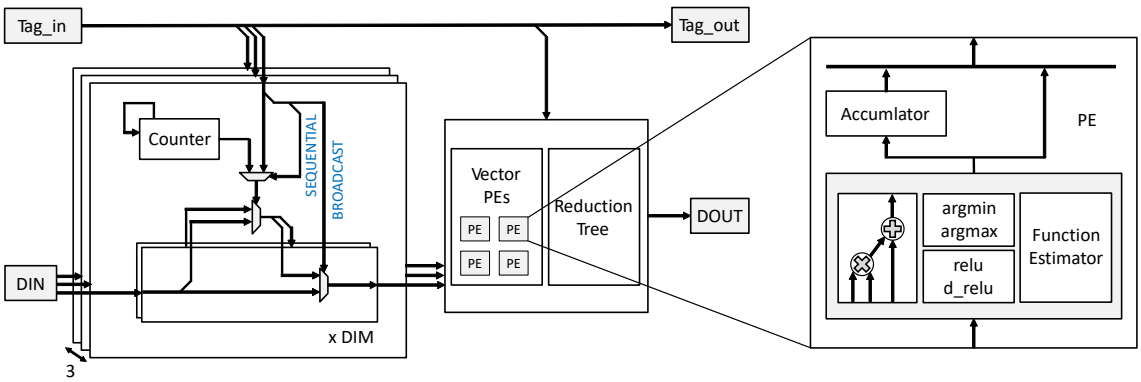

Figure 3.4: Vector unit block diagram

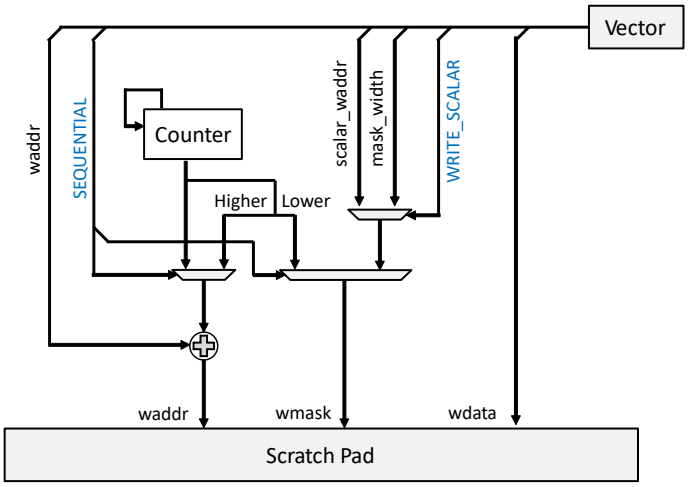

Figure 3.5: Vector unit outputs to the execution controller flow

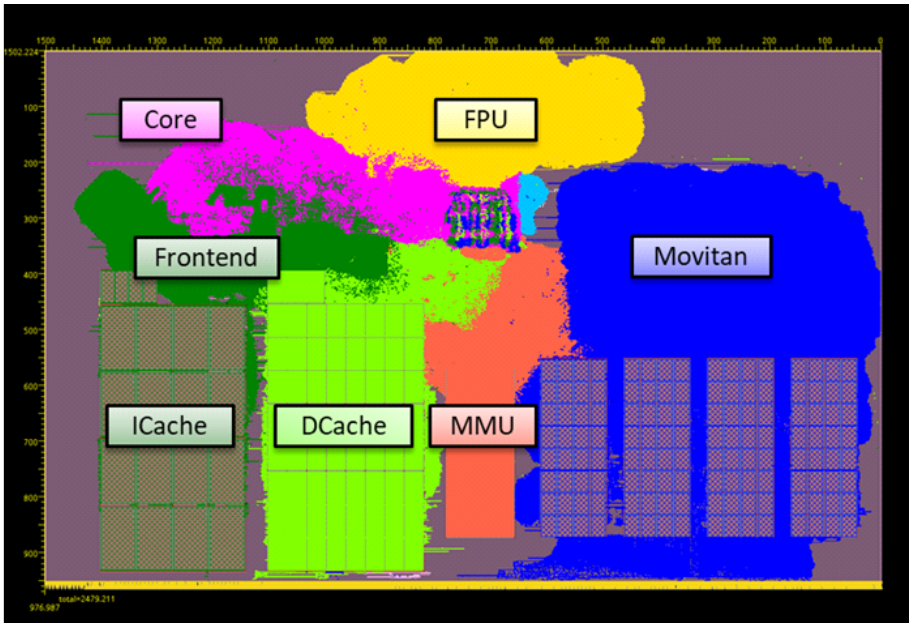

Figure 3.6: Movitan floorplan connected to Rocket core

500 MHz.

3.3 Software Architecture

The software API of Movitan is basically the same as that of Gemmini, and generates a C header file from the parameters necessary for processor generation. When software is developed to run Movitan, information about the hardware architecture can be obtained on the software side by linking to the generated C header. By

generating an SoC with the dedicated parameter listing 3.1 for configuring Movitan, a dedicated C header file is generated at the same time. The contents of the C header file are cut out of the generated parameters and the corresponding parts. Listing 3.3 is a sample for the inference of fully connected NNs by Movitan.

Listing 3.1: Movitan parameters

```
object MovitanConfigs {
    val defaultConfig = MovitanArrayConfig(
      vecLanes = 8,
      burst_size = 64,
      ld_queue_length = 8,
      st_queue_length = 2,
      ex_queue_length = 8,
      rob_entries = 16,
      sp_banks = 4,
      sp_capacity = CapacityInKilobytes(16), //CapacityInKilobytes(256),
      mem_pipeline = 2,
      dma_maxbytes = 64, // TODO get this from cacheblockbytes
      dma_buswidth = 128,//128*2, // TODO get this from SystemBusKey
      aligned_to = 1,
      inputType = rialBFloat(1,7,8),// Movitan Type SInt(8.W), Float(8,24) or Float(8,8)
      outputType = rialBFloat(1,7,8),//Float(8,8) ,
      pe_latency = 3,
   )
}
```

Listing 3.2: Movitan Macro

```
#define DIM 16
#define VECTOR_LANE 8
#define ADDR_LEN 32

#define BANK_NUM 4
#define SP_CAPACITY 256 * 1024 * 8
#define SP_WIDTH DIM * VECTOR_LANE
#define BANK_ROWS SP_CAPACITY/(BANK_NUM*SP_WIDTH)

#define MAX_BYTES 64
#define MAX_BLOCK_LEN (MAX_BYTES / (DIM * 1))

#define A_SP_ADDR_START 0
#define B_SP_ADDR_START 1 * BANK_ROWS
#define C_SP_ADDR_START 2 * BANK_ROWS
#define D_SP_ADDR_START 3 * BANK_ROWS
```

Listing 3.3: Movitan software (single batch)

```
void movitan_forward_mat_vec_mul(rocc_elem_t *InA, rocc_elem_t *InB,
                                 rocc_elem_t *Out, rocc_elem_t *InD,
                                 int col_num, int row_num) {

  const uint32_t vec_col_size_full    = col_num/DIM;
  const uint32_t vec_col_size_partial = col_num%DIM;
  const uint32_t vec_col_size = vec_col_size_full + (vec_col_size_partial != 0 ? 1 : 0);
  const uint32_t vec_row_size_full    = row_num/DIM;
  const uint32_t vec_row_size_partial = row_num%DIM;
  const uint32_t vec_row_size = vec_row_size_full + (vec_row_size_partial != 0 ? 1 : 0);
  const uint32_t sp_sift = row_num < DIM ? row_num : DIM ;

  // load vector A
  movitan_load_vector(InA, A_SP_ADDR_START, vec_col_size_full, vec_col_size_partial);
  // load vector D
  movitan_load_vector(InD, D_SP_ADDR_START, vec_row_size_full, vec_row_size_partial);
  // load matrix B
  movitan_load_matrix_blocked(InB, B_SP_ADDR_START, col_num,
                      vec_col_size_full, vec_col_size_partial, vec_col_size,
                      vec_row_size_full, vec_row_size_partial, vec_row_size);
  // product of matrix B and vector A
  movitan_vec_col_mat_col_mul(A_SP_ADDR_START, B_SP_ADDR_START, C_SP_ADDR_START,
                              vec_col_size_full, vec_col_size_partial, vec_col_size,
                              vec_row_size_full, vec_row_size_partial, vec_row_size);
  // add bias & activation
  movitan_fadd(C_SP_ADDR_START, D_SP_ADDR_START, C_SP_ADDR_START,vec_row_size);
  movitan_relu(C_SP_ADDR_START, D_SP_ADDR_START, vec_row_size);
  // store vector D
  movitan_store_vector(Out, D_SP_ADDR_START, vec_row_size_full, vec_row_size_partial);
  };
```

Chapter 4

Conclusion

4.1 Discussion

Algorithms with excessive computational cost should be avoided in deep learning, because the amount of computation continues to grow in scale[19]. In contrast, deep ensemble methods provide superior predictive uncertainty, but the computational and memory costs during training are proportional to the number of models. The variational inference approach with BBB requires only a doubling of the number of parameters for the weights, the sampling cost of which can be handled efficiently by dedicated hardware[39]. However, previous studies have shown that variational inference approaches tended to perform worse than ensemble approaches[13]. This motivated the proposal of AdamB as a stable learning method, which successfully outperformed deep ensembles for noise collapse in the input data. These results were only confirmed experimentally, and additional theoretical support would be required in the future. The mathematical difference between the Bayesian approach and an ensemble is discussed[72]. The Bayesian approach is interpreted as a "soft model selection'' approach that attempts to find an effective model that explains all of the given data satisfactorily. On the other hand, in the case of a combination of models, such as a deep ensemble, each individual model does not need to explain all of the data. Instead, the explanation of all of the data by multiple models collectively is sufficient. That is, the Bayesian approach is more difficult to optimize and performs less well than the combined model approach because this approach

attempts to explain everything well with a single model in a hypothesis. Although AdamB is a Bayesian approach, it uses ensembles by sampling from variational posterior distributions to assess the predictive uncertainty. Hence, the challenge lies in the interpretation of the multiple predictions by AdamB. In this regard, it would be helpful to investigate the reason for the superior generalization performance of deep ensembles [73]. A robustness study for adversarial perturbation[74] would be helpful to determine why AdamB enables robust inference of noise-type corruptions. Robustness to adversarial perturbation was confirmed by replacing the input layer of the convolutional neural network with a neurophysiological model of the actual visual cortex and adding stochasticity to the model[75]. This was interpreted as meaning that the noise inherent in the network formed manifolds for a single data sample, and the effects of adversarial perturbation were captured within these manifolds[76].

The connection between the variational parameters obtained by AdamB and neuroscience would be interesting to study. The synaptic strengths observed in the visual cortex of rats are known to obey a lognormal distribution. Leaky integrate-and-fire (LIF) simulations confirmed that the weights based on this distribution promote efficient neural firing[77]. This result confirms that it is not essential for the distribution of synapses to be lognormal, which is the neurophysiological distribution of synapses, but that it is important for neurons to have a large number of weak synapses and a small number of strong synapses[78]. With reference to AdamB (-1,-6), it is possible to interpret more than 90% of the SNRs of the variational parameters of the convolutional layer as noisy weights, i.e., weak weights (Figure 2.9). The fact that the remaining few large SNR variate parameters can be interpreted as strong weights, means that the SNRs of AdamB (-1,-6) correspond to the neurophysiological synapse distribution. However, it should be noted that the weak weights due to the variational parameters obtained by AdamB can easily change signs. The sign of the weights corresponds to excitatory and inhibitory synapses in neurophysiology, and they differ in the receptors that receive the chemical information. The scaffold protein droplet, which holds the receptors to the cell membrane, prevents the excitatory and inhibitory receptors from mixing at a single synapse[79]. Therefore, a situation in which the sign undergoes fluctuation at large

weights is unlikely to arise. On the other hand, excitatory and inhibitory switches could probably be modeled as a situation in which the receptors are not fixed, as in silent synapses[80]. This could be considered as an extension of the research on modeling neural networks assuming silent synapses[81]. The balance between excitability and inhibition is known to play an important role in neural information coding[82], and its relation to the behavior of noisy variational parameters during learning would also be interesting to examine.

The properties of noisy variational parameters may be applicable to neuromorphic devices. This approach attempts the low-power execution of neural networks by replacing synaptic weights with electrical resistance values[83]. Memristors[84], also referred to as synaptic devices, can change their resistance by changing the state of their crystal structure, but changing the resistance by switching causes the values to vary in a Gaussian distribution[85]. Initial research on the implementation of Bayesian neural networks in neuromorphic devices has already commenced[86], and optimization with AdamB may enable robust inference against variations in resistance change.

4.2 Conclusion

My study revealed that the difficulty of learning in BBB lies in the rapid and excessive update of parameter $\boldsymbol{\rho}$, which is fundamentally solved by AdamB using the SM prior. I demonstrated that the rapid updating of parameter $\boldsymbol{\rho}$ can be decoupled from Adam, whereas the excessive increase can be suppressed by using the SM prior. Experiments also showed that by using the SM Prior, the parameter $\boldsymbol{\mu}$ takes a sparse distribution and is strongly robust against noise type corruption. However, the evaluation of AdamB was restricted to evaluating the robustness against covariate shift in image identification tasks, and its application to other tasks would have to be assessed in future.

I also confirmed that the model trained by AdamB using the SM prior did not undergo any clear performance degradation even when the random number precision was reduced to 4 bits. In addition, I applied principal component analysis to the

MNIST dataset and found that AdamB responds well to eigenvector features with small eigenvalues if they have been properly trained. These results are expected to provide a validation metric for applications of dedicated processors generated by Movitan, a SoC generation system that enables efficient AdamB training. The development of end-to-end AdamB training flow by Movitan is a future task.

I believe that this series of studies will provide important insights into the efficient operation of reliable deep learning models on real-world settings. Overfitting of neural networks to training data is among the most significant problems in machine learning. Bayesian neural networks (BNNs) are known to be robust against overfitting owing to their ability to model parameter uncertainty. Bayes by Backprop (BBB), a simple variational inference approach that optimizes variational parameters by backpropagation, has been proposed to train BNNs. However, many studies have encountered challenges in terms of variational inference for large-scale models, such as deep learning. Thus, this study proposed Adam with Decoupled Bayes by Backprop (AdamB) to stabilize the training of BNN by applying the Adam estimator to evaluate the gradient of the neural network. The proposed approach stabilized the noisy gradient of the BBB and mitigated excessive changes in the parameters. In addition, AdamB combined with a Gaussian scale mixture as a prior distribution can suppress the intrinsic increase in variational parameters. The proposed AdamB exhibited superior stability compared to training using Adam with vanilla BBB. Further, the covariate shift benchmark using image classification tasks indicated that AdamB was more reliable than deep ensembles in the case of noise-type covariate shifts. The considerations for stable learning of BNNs by AdamB shown for image classification tasks are expected to offer important insights for application to other domains.

References

[1] L. Cosmides and J. Tooby, "Are humans good intuitive statisticians after all? rethinking some conclusions from the literature on judgment under uncertainty," *Cognition*, vol. 58, no. 1, pp. 1–73, Jan. 1996.

[2] X. Jiang, M. Osl, J. Kim, and L. Ohno-Machado, "Calibrating predictive model estimates to support personalized medicine," en, *J. Am. Med. Inform. Assoc.*, vol. 19, no. 2, pp. 263–274, Mar. 2012.

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

[4] L. Kong, H. Jiang, Y. Zhuang, J. Lyu, T. Zhao, and C. Zhang, "Calibrated language model fine-tuning for in- and out-of-distribution data," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Online: Association for Computational Linguistics, Nov. 2020, pp. 1326–1340.

[5] K. Han, B. Lakshminarayanan, and J. Liu, "Reliable graph neural networks for drug discovery under distributional shift," Nov. 2021. arXiv: `2111.12951 [cs.LG]`.

[6] J. Bridle, "Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters," in *Advances in Neural Information Processing Systems*, vol. 2, Morgan-Kaufmann, 1989.

[7] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[8] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" In *Advances in Neural Information Processing Systems*, vol. 30, Curran Associates, Inc., 2017.

[9] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 2016, pp. 1050–1059.

[10] G. E. Hinton and D. van Camp, "Keeping the neural networks simple by minimizing the description length of the weights," in *Proceedings of the sixth annual conference on Computational learning theory*, ser. COLT '93, Santa Cruz, California, USA: Association for Computing Machinery, Aug. 1993, pp. 5–13.

[11] A. Graves, "Practical variational inference for neural networks," in *Advances in Neural Information Processing Systems*, vol. 24, Curran Associates, Inc., 2011.

[12] P. Izmailov, S. Vikram, M. D. Hoffman, and A. G. G. Wilson, "What are bayesian neural network posteriors really like?" In *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 4629–4640.

[13] Y. Ovadia, E. Fertig, J. Ren, *et al.*, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems*, Red Hook, NY, USA: Curran Associates Inc., Dec. 2019, pp. 14 003–14 014.

[14] G. E. Moore, "Cramming more components onto integrated circuits," *Electronics*, vol. 38, no. 8, pp. 114–117, 1965.

[15] M. B. Taylor, "Is dark silicon useful? harnessing the four horsemen of the coming dark silicon apocalypse," in *DAC Design Automation Conference 2012*, Jun. 2012, pp. 1131–1136.

[16] D. P. Rodgers, "Improvements in multiprocessor system design," *SIGARCH Comput. Archit. News*, vol. 13, no. 3, pp. 225–231, Jun. 1985.

[17] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, Curran Associates, Inc., 2012.

[18] N. P. Jouppi, C. Young, N. Patil, *et al.*, "In-Datacenter performance analysis of a tensor processing unit," in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, ser. ISCA '17, Toronto, ON, Canada: Association for Computing Machinery, Jun. 2017, pp. 1–12.

[19] A. Mehonic and A. J. Kenyon, "Brain-inspired computing needs a master plan," en, *Nature*, vol. 604, no. 7905, pp. 255–260, Apr. 2022.

[20] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 97, PMLR, Jun. 2019, pp. 6105–6114.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186.

[22] D. Zhang, S. Huda, E. Songhori, *et al.*, "A full-stack search technique for domain optimized deep learning accelerators," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. New York, NY, USA: Association for Computing Machinery, 2022, pp. 27–42, ISBN: 9781450392051.

[23] A. Amid, D. Biancolin, A. Gonzalez, *et al.*, "Chipyard: Integrated design, simulation, and implementation framework for custom socs," *IEEE Micro*, vol. 40, pp. 10–21, 2020.

[24] Y. Ding, W. Jiang, Q. Lou, *et al.*, "Hardware design and the competency awareness of a neural network," en, *Nature Electronics*, vol. 3, no. 9, pp. 514–523, Sep. 2020.

[25] P. V'kovski, A. Kratzel, S. Steiner, H. Stalder, and V. Thiel, "Coronavirus biology and replication: Implications for SARS-CoV-2," en, *Nat. Rev. Microbiol.*, vol. 19, no. 3, pp. 155–170, Oct. 2020.

[26] M. Roberts, D. Driggs, M. Thorpe, *et al.*, "Common pitfalls and recommendations for using machine learning to detect and prognosticate for COVID-19 using chest radiographs and CT scans," en, *Nature Machine Intelligence*, vol. 3, no. 3, pp. 199–217, Mar. 2021.

[27] J. R. Zech, M. A. Badgeley, M. Liu, A. B. Costa, J. J. Titano, and E. K. Oermann, "Variable generalization performance of a deep learning model to detect pneumonia in chest radiographs: A cross-sectional study," en, *PLoS Med.*, vol. 15, no. 11, e1002683, Nov. 2018.

[28] W. T. Harvey, A. M. Carabelli, B. Jackson, *et al.*, "SARS-CoV-2 variants, spike mutations and immune escape," en, *Nat. Rev. Microbiol.*, vol. 19, no. 7, pp. 409–424, Jul. 2021.

[29] N. Band, T. G. Rudner, Q. Feng, *et al.*, "Benchmarking bayesian deep learning on diabetic retinopathy detection tasks," in *NeurIPS 2021 Workshop on Distribution Shifts: Connecting Methods and Applications*, 2021.

[30] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, J. Moody, S. Hanson, and R. P. Lippmann, Eds., vol. 4, Morgan-Kaufmann, 1991.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[32] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[33] J. Bjorck, K. Q. Weinberger, and C. P. Gomes, "Understanding decoupled and early weight decay," in *AAAI*, 2021.

[34] M. P. Naeini, G. F. Cooper, and M. Hauskrecht, "Obtaining well calibrated probabilities using bayesian binning," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15, Austin, Texas: AAAI Press, 2015, pp. 2901–2907, ISBN: 0262511290.

[35] D. Hendrycks and T. G. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[36] D. J. C. Mackay, "Bayesian methods for adaptive models," UMI Order No. GAX92-32200, Ph.D. dissertation, USA, 1992.

[37] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, 2015, pp. 1613–1622.

[38] M. Ferianc, P. Maji, M. Mattina, and M. Rodrigues, "On the effects of quantisation on model uncertainty in bayesian neural networks," in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, ser. Proceedings of Machine Learning Research, vol. 161, PMLR, 27–30 Jul 2021, pp. 929–938.

[39] R. Cai, A. Ren, N. Liu, *et al.*, "Vibnn: Hardware acceleration of bayesian neural networks," *SIGPLAN Not.*, vol. 53, no. 2, pp. 476–488, Mar. 2018, ISSN: 0362-1340.

[40] Y. Hirayama, T. Asai, M. Motomura, and S. Takamaeda, "A hardware-efficient weight sampling circuit for bayesian neural networks," en, *Int. J. High Perform. Comput. Networking*, vol. 10, no. 2, pp. 84–93, Jul. 2020.

[41] K. Asanović, R. Avizienis, J. Bachrach, *et al.*, "The rocket chip generator," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17, Apr. 2016. [Online]. Available: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-17.html`.

[42] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanović, "The risc-v instruction set manual, volume i: User-level isa, version 2.1," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-118, May 2016. [Online]. Available: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-118.html`.

[43] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., Dec. 2017, pp. 5580–5590.

[44] A. Krogh and J. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems*, vol. 4, Morgan-Kaufmann, 1991.

[45] Y. Wen, P. Vicol, J. Ba, D. Tran, and R. B. Grosse, "Flipout: Efficient pseudo-independent weight perturbations on mini-batches," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, OpenReview.net, 2018.

[46] K. Osawa, S. Swaroop, M. E. E. Khan, *et al.*, "Practical deep learning with bayesian principles," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[47] D. P. Kingma, T. Salimans, and M. Welling, "Variational dropout and the local reparameterization trick," in *Advances in Neural Information Processing Systems*, vol. 28, Curran Associates, Inc., 2015.

[48] A. Wu, S. Nowozin, E. Meeds, R. E. Turner, J. M. Hernández-Lobato, and A. L. Gaunt, "Deterministic variational inference for robust bayesian neural networks," in *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, OpenReview.net, 2019.

[49] R. Krishnan, M. Subedar, and O. Tickoo, "Specifying weight priors in bayesian deep neural networks with empirical bayes," en, *AAAI*, vol. 34, no. 04, pp. 4477–4484, Apr. 2020.

[50] R. Krishnan and O. Tickoo, "Improving model calibration with accuracy versus uncertainty optimization," in *Advances in Neural Information Processing Systems*, vol. 33, Curran Associates, Inc., 2020, pp. 18 237–18 248.

[51] S. Farquhar, M. A. Osborne, and Y. Gal, "Radial bayesian neural networks: Beyond discrete support in large-scale bayesian deep learning," in *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 108, PMLR, 26–28 Aug 2020, pp. 1352–1362.

[52] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

[53] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, 2015, pp. 730–734. DOI: 10.1109/ACPR.2015.7486599.

[54] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[55] G. Zhang, S. Sun, D. K. Duvenaud, and R. B. Grosse, "Noisy natural gradient as variational inference," in *ICML*, 2018.

[56] Z. Xiao, J. Shen, X. Zhen, L. Shao, and C. G. M. Snoek, "A bit more bayesian: Domain-invariant learning with uncertainty," in *ICML*, 2021.

[57] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[58] M. Abadi, P. Barham, J. Chen, *et al.*, "Tensorflow: A system for large-scale machine learning.," in *OSDI*, vol. 16, 2016, pp. 265–283.

[59] A. Paszke, S. Gross, F. Massa, *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, vol. 32, Curran Associates, Inc., 2019.

[60] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009.

[61] Y. Le and X. Yang, "Tiny imagenet visual recognition challenge," *cs231n.stanford.edu*, 2015.

[62] A. Yaguchi, T. Suzuki, W. Asano, S. Nitta, Y. Sakata, and A. Tanizawa, "Adam induces implicit weight sparsity in rectifier neural networks," *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 318–325, 2018.

[63] D. J. Hand and R. J. Till, "A simple generalisation of the area under the roc curve for multiple class classification problems," *Machine Learning*, vol. 45, pp. 171–186, 2004.

[64] P. Izmailov, P. Nicholson, S. Lotfi, and A. G. Wilson, "Dangers of bayesian model averaging under covariate shift," in *Advances in Neural Information Processing Systems*, vol. 34, Curran Associates, Inc., 2021, pp. 3309–3322.

[65] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[66] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ser. ICML'10, Haifa, Israel: Omnipress, Jun. 2010, pp. 807–814.

[67] G. E. Box, "A note on the generation of random normal deviates," *Ann. Math. Statist.*, vol. 29, pp. 610–611, 1958.

[68] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ser. ICML'15, Lille, France: JMLR.org, Jul. 2015, pp. 448–456.

[69] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848.

[70] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Ng, "Reading digits in natural images with unsupervised feature learning," 2011.

[71] L. T. Clark, V. Vashishtha, L. Shifren, *et al.*, "Asap7: A 7-nm finfet predictive process design kit," *Microelectronics Journal*, vol. 53, pp. 105–115, 2016, ISSN: 0026-2692.

[72] T. P. Minka, "Bayesian model averaging is not model combination," 2002.

[73] L. A. Ortega, R. Cabañas, and A. Masegosa, "Diversity and generalization in neural network ensembles," in *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, G. Camps-Valls, F. J. R. Ruiz, and I. Valera, Eds., ser. Proceedings of Machine Learning Research, vol. 151, PMLR, 2022, pp. 11 720–11 743.

[74] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2014.

[75] J. Dapello, T. Marques, M. Schrimpf, F. Geiger, D. Cox, and J. J. DiCarlo, "Simulating a primary visual cortex at the front of cnns improves robustness to image perturbations," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33, Curran Associates, Inc., 2020, pp. 13 073–13 087.

[76] J. Dapello, J. Feather, H. Le, *et al.*, "Neural population geometry reveals the role of stochasticity in robust perception," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 15 595–15 607.

[77] J.-N. Teramae, Y. Tsubo, and T. Fukai, "Optimal spike-based communication in excitable networks with strong-sparse and weak-dense links," *Sci. Rep.*, vol. 2, no. 1, pp. 1–6, Jul. 2012.

[78] H. Kada, J.-N. Teramae, and I. T. Tokuda, "Highly heterogeneous excitatory connections require less amount of noise to sustain firing activities in cortical networks," *Front. Comput. Neurosci.*, vol. 12, p. 104, Dec. 2018.

[79] X. Chen, X. Wu, H. Wu, and M. Zhang, "Phase separation at the synapse," *Nat. Neurosci.*, vol. 23, no. 3, pp. 301–310, Feb. 2020.

[80] G. A. Kerchner and R. A. Nicoll, "Silent synapses and the emergence of a postsynaptic mechanism for LTP," *Nat. Rev. Neurosci.*, vol. 9, no. 11, pp. 813–825, Oct. 2008.

[81] N. Brunel, V. Hakim, P. Isope, J.-P. Nadal, and B. Barbour, "Optimal information storage and the distribution of synaptic weights: Perceptron versus purkinje cell," *Neuron*, vol. 43, no. 5, pp. 745–757, Sep. 2004.

[82] S. Zhou and Y. Yu, "Synaptic E-I balance underlies efficient neural coding," *Front. Neurosci.*, vol. 12, p. 46, Feb. 2018.

[83] S. J. Kim, S. Kim, and H. W. Jang, "Competing memristors for brain-inspired computing," *iScience*, vol. 24, no. 1, p. 101 889, Jan. 2021.

[84] D. B. Strukov, G. S. Snider, D. R. Stewart, and R. S. Williams, "The missing memristor found," *Nature*, vol. 453, no. 7191, pp. 80–83, May 2008.

[85] D. Ielmini and G. Pedretti, "Device and circuit architectures for in ‐ memory computing," *Advanced Intelligent Systems*, vol. 2, no. 7, p. 2 000 040, Jul. 2020.

[86] A. Sebastian, R. Pendurthi, A. Kozhakhmetov, *et al.*, "Two-dimensional materials-based probabilistic synapses and reconfigurable neurons for measuring inference uncertainty using bayesian neural networks," *Nat. Commun.*, vol. 13, no. 1, pp. 1–10, Oct. 2022.

Acknowledgement

I would like to express my sincere gratitude to my supervisor, Guest Prof. Makoto Taiji for his continuous support throughout my Master's and Ph.D. studies.

I would also like to thank Drs. Yousuke Ohno, Gentaro Morimoto, Teruhisa Komatsu, Hao Zhang, Yohei Koyama, and Itta Ohmura of the RIKEN Center for Biosystems Dynamics Research (BDR) for their valuable advice on preparing this thesis.

My sincere appreciation to Prof. Masahiro Ueda, Prof. Shigeru Kitazawa, Prof. Shinji Nishimoto, and Guest Prof. Toshiyuki Kanoh for serving as members of my thesis committee.

I would like to thank Dr. Hideki Asoh, Dr. Tomoyuki Higuchi, Dr. Kenji Fukumizu, Dr. Ryosuke Kojima, Dr. Masahiro Suzuki, and Mr. Tsuyoshi Ishizone for scientific discussions and constructive criticism.

My research was supported by the Program for Leading Graduate Schools of the Ministry of Education, Culture, Sports, Science and Technology, Japan (K03), RIKEN Junior Research Associate Program, and Masason Foundation.

Achievements

PEER-REVIEWED JOURNAL ARTICLES

1. Keigo Nishida and Makoto Taiji, "AdamB: Decoupled Bayes by Backprop With Gaussian Scale Mixture Prior," in *IEEE Access*, vol. 10, pp. 92959-92970, 2022, doi: 10.1109/ACCESS.2022.3203484.

PEER-REVIEWED PROCEEDINGS

1. Gentaro Morimoto, Yohei M. Koyama, Hao Zhang, Teruhisa S. Komatsu, Yousuke Ohno, Keigo Nishida, Itta Ohmura, Hiroshi Koyama, Makoto Taiji, "Hardware acceleration of tensor-structured multilevel ewald summation method on MDGRAPE-4A, a special-purpose computer system for molecular dynamics simulations," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1-15, 2021

NON PEER-REVIEWED JOURNAL ARTICLES

1. Kazufumi Hosoda, Keigo Nishida, Shigeto Seno, Tomohiro Mashita, Hideki Kashioka, Izumi Ohzawa, "It's DONE: Direct ONE-shot learning with quantile weight imprinting," *arXiv*, Nov. 2, 2022. [Online]. Available: `https://arxiv.org/abs/2204.13361`

2. 西田圭吾, 泰地真弘人, "Bayes by Backprop 法における適応的最適化の提案と低精度サンプリングによる確率キャリブレーション性能評価," 信学技報, vol. 121, no. 155, PRMU2021-8, pp. 7-12, 2021.

OTHERS

1. Keigo Nishida, “Adopted Chipyard Framework for Bayesian Neural Networks Training Accelerator Development,” *RISC-V Day Vietnam 2020 Virtual Booths*, 2020.

2. 西田圭吾, “RISC-V ベースカスタム SoC 開発ツール Chipyard によるベイズニューラルネット向け学習アクセラレータ開発,” RISC-V Day Tokyo 2020 バーチャルブース, 2020.