



Detecting Uncertain BNN Outputs on FPGA Using Monte Carlo Dropout Sampling

Tomoyuki Myojin^{1,2} , Shintaro Hashimoto¹, and Naoki Ishihama¹

¹ Japan Aerospace Exploration Agency, Tsukuba, Japan

{hashimoto.shintaro, ishihama.naoki}@jaxa.jp

² Hitachi, Ltd., Yokohama, Japan

tomoyuki.myojin.fs@hitachi.com

Abstract. Monte Carlo dropout sampling (MC Dropout), which approximates a Bayesian Neural Network, is useful for measuring the uncertainty in the output of a Deep Neural Network (DNN). However, because it takes a long time to sample DNN's output for calculating its distribution, it is difficult to apply it to edge computing where resources are limited. Thus, this research proposes a method of reducing a sampling time required for MC Dropout in edge computing by parallelizing the calculation circuit using FPGA. To apply MC dropout in an FPGA, this paper shows an efficient implementation by binarizing the neural network and simplifying dropout computation by pre-dropout and localizing parallel circuits. The proposed method was evaluated using the MNIST dataset and a dataset of satellite images of ships at sea captured. As a result, it was possible to reject approximately 60% of data which the model had not learned as “uncertain” on a classification identification problem of the image on an FPGA. Furthermore, for 20 units in parallel, the amount of increase in the circuit scale was only 2–3 times that of non-parallelized circuits. In terms of inference speed, parallelization of dropout circuits has achieved up to 3.62 times faster.

Keywords: Deep learning · FPGA · Binarized neural network · Uncertainty · Monte Carlo dropout sampling · SAR

1 Introduction



Image identification using deep learning has found wide use, especially in embedded edge computing. In embedded edge applications, power and computer resources are limited unlike computers with a powerful GPU, so a Deep Neural Network (DNN) requires power-saving features, real-time, and memory-saving. Despite these limitations, the DNNs requires reliability to provide high-quality systems and services in the edge computing application such as automobiles and satellites.

The reliability of the DNN can be further enhanced by determining the uncertainty in the output of the DNN using Monte Carlo Dropout Sampling (MC Dropout) [2, 3]. Whereas the original dropout technique is used to obtain regularization by intentionally dropping neurons of the neural network during learning, MC Dropout also applies dropout at the time of inference and samples the output value of the DNN. This approximates a Bayesian Neural Network, which can calculate the distribution of output values. By evaluating the variance calculated from the output distribution, the uncertainty in the inference by the trained model can be measured. There are two types of DNN uncertainties: the uncertainty of the data itself, such as noise, and the model uncertainty. The former is called aleatoric uncertainty and the latter is called epistemic uncertainty [5], which can be measured by MC Dropout.

An example of measuring epistemic uncertainty will be described using a handwritten numerical classification task. MNIST is a well-known dataset of hand-drawn numerical images and has a variety of the digits from 0 to 9. Among them, there are two accepted ways of writing “7”: one with a horizontal bar in the center and one without. In a particular linguistic sphere (e.g., in Japan), it is more common to write “7” without a horizontal bar. This paper calls the former 7_{bar} and the latter 7_{nobar} .

If 7_{bar} is inferred with the model $M_{7_{nobar}}$ trained on the dataset excluding 7_{bar} , the uncertainty in the prediction of 7_{bar} will be higher than that of 7_{nobar} (Table 1). Because 7_{bar} is unknown in the model $M_{7_{nobar}}$, the uncertainty of prediction (i.e., epistemic uncertainty) is high. Using this property makes possible to obtain a reliable inference by rejecting an inference result with a high uncertainty and classified as unreliable.

Table 1. Epistemic Uncertainties for the ($M_{7_{nobar}}$) model trained on dataset excluding 7_{bar} and for the ($M_{7_{bar}}$) model trained on dataset including 7_{bar} .

Input Image		
Variance for $M_{7_{nobar}}$	8.82×10^{-2}	9.14×10^{-10}
Variance for $M_{7_{bar}}$	4.66×10^{-13}	1.42×10^{-13}

MC Dropout has the advantage of being easy to implement because it is not necessary to change the network structure at the time of inference and learning. However, the disadvantage is that it requires multiple samplings during inference. According to Gal [3], it requires 20 samplings which means that the computation time is 20 times as long. In an environment with abundant computational resources or one that does not need to operate in real time, this is not a problem.

However, in edge computing for embedded devices, computational resources and power supply are limited and there are many situations that require

real-time computation. For example, in a task that identifies an object in a car, the time it takes to make inferences, or latency, is important. In another example, where power and computational resources are very limited, such as in outer space, it is necessary to have a power-saving and memory-saving DNN.

For this situation, inference using a DNN have been implemented in a Field Programmable Gate Array (FPGA) in recent years. In particular, a Binarized Neural Network (BNN), which represents the parameters of a Neural Network and/or the intermediate results of a calculation as a binary value instead of a floating point, has been realized in an FPGA [8,9].

This paper reports using MC Dropout in a BNN implemented on an FPGA, done in order to improve the reliability of inference results for edge computing for embedded devices. Parallelizing inferences on an FPGA ensures acceptable real-time performance. Experiments based on practical ship recognition using satellite images show that MC Dropout improves reliability in BNNs implemented on an FPGA.

The organization of this paper is shown below. Section 2 describes the method for measuring uncertainty using MC Dropout and rejecting unreliable judgments applied to image classification. Section 3 describes how to implement MC Dropout in an FPGA; three points are particularly important: (1) a method for constructing binarized neural network, (2) pre-dropout method that efficiently integrates dropout layer and activation layer, and (3) implementation of a circuit that parallelizes dropout. Section 4 evaluates the ability to reject uncertain results by MC Dropout implemented in FPGA and the overhead of inference time and circuit scale. In the evaluation, we use MNIST dataset and images of ships on the ocean captured by a satellite. Section 5 describes the evaluation results, and the Sect. 6 describes the conclusions of this paper.

2 Related Work

Gal [2,3] made a typical study of measuring uncertainty in DNNs using MC Dropout. Based on Gal’s approach, we have measured the uncertainty in a DNN that performs object detection based on YOLOv3 using MC Dropout [7]. By rejecting the results identified as uncertain, this paper proposes a method to improve precision. The method would do this without adversely affecting recall improving reliability of identifying lunar craters from satellite images. Miller [6] adopted MC Dropout Sampling for object detection based on the SSD, and Feng [1] adopted it for Faster R-CNN. These methods target an object detection and do not address a classification problem. In addition, these methods do not take into account their use in edge computing.

Xilinx proposed a highly efficient implementation of a BNN on an FPGA using the simplified Batch Normalization by the FINN algorithm [9]. The BNN was implemented by high-level synthesis in the C language, does not require the use of a hardware description language such as VHDL and Verilog HDL, and has quite a low learning and development cost. Takamaeda et al. have also developed their own compiler NNgen to synthesize a model-specific hardware accelerator

for deep neural [8]. NNgen generates a Verilog HDL source code from an input model definition written in Python. Hashimoto has developed an application that identifies the type and length of a ship from images captured by a satellite using a network based on the Xilinx FINN algorithm mentioned above [4]. This paper adopts the Xilinx FINN algorithm in consideration of its application to satellite images and its ease of implementation of parallelization.

3 Uncertainty of the DNN

3.1 Measuring Uncertainty with Monte Carlo Dropout

This section describes a method for measuring the uncertainty of inference results of a DNN. The DNN predicts the inference dataset in the inference phase using the optimal weights obtained in the learning phase from the learning dataset. Normally, the learning dataset and the inference dataset are independently and identically distributed, however, in some cases, the distribution of the inference dataset may differ from the learning dataset due to a lack in the learning dataset. If there is a shortage in the learning data set, it is not possible to obtain the optimum weights, resulting in bad inference results. In such a case, the model outputs an inference result with a high uncertainty for the inference dataset. Using the approximation of a deep Bayesian neural network using MC Dropout, it is possible to measure the uncertainty of the model for inference data. This uncertainty is called epistemic uncertainty.

Dropout is a regularization method to prevent over-fitting by probabilistically disabling neurons in the network and is usually used only during learning. By using dropout during inference, the network will generate a different result each time it is inferred. Inference results using MC Dropout are derived by the following.

$$\bar{y}(x) \approx \frac{1}{N_{mc}} \sum^{N_{mc}} y_{drop}(x) \quad (1)$$

where y_{drop} is output of the network that is dropped out, x is input to the network and N_{mc} is the number of times of sampling needed to get distribution. Because the neurons to drop out are randomly selected, $y_{drop}(x)$ will have different results for each sampling. Since the output of MC Dropout is a random variable, the expected value of the output can be obtained by calculating the average value. From another point of view, \bar{y} can also be regarded as the ensemble output of slightly different networks generated by dropout. The variance of inference results is represented by the following.

$$var(x) \approx \frac{1}{N_{mc}} \sum^{N_{mc}} (y_{drop}(x) - \bar{y}(x))^2 \quad (2)$$

The value of this variance represents the uncertainty. Kendall et al. reported that this epistemic uncertainty tends to increase if the data set is intentionally reduced [5]. Thus, inferring an unlearned dataset for the model increases uncertainty.

By using this property, when inferring unlearned or unknown data, the result is unreliable and can be rejected as if the model is indistinguishable. This can be expressed by the following equation.

$$y_{reliable}(x) = \begin{cases} \bar{y}(x) & (var_y(x) < thresholds) \\ unknown & (var_y(x) \geq thresholds) \end{cases} \quad (3)$$

If the uncertainty is less than the threshold, the inference result is adopted as is, and if it is above the threshold, the inference result is rejected as “unknown.” There are several ways to determine the threshold, e.g., based on the uncertainty at the time of inferring the learning data, or determined empirically.

3.2 Rejecting Uncertain Judgment in Classification

This section describes how to reject judgment of DNN based on uncertainty. In this section, we use the handwritten digit classification problem. MNIST, which is widely used as a dataset of handwritten digits, has almost 70,000 images of digits from 0 to 9. There are 7,293 numeral 7s; of these, 938 are with a horizontal bar (7_{bar}), 6,355 are without a horizontal bar (7_{nobar}).

Where only 7_{nobar} is used in a particular linguistic sphere, the model cannot correctly identify 7_{bar} , and the accuracy will be low. Moreover, the model may classify 7_{bar} as 7 with a high confidence. However, since this model does not learn that “ 7_{bar} is 7,” it is not desirable to classify 7_{bar} as 7.

In such a case, by determining whether the prediction result is reliable using uncertainty, it is possible to prevent 7_{bar} from being classified as 7. If the value of uncertainty is greater than or equal to a certain value, it provides the option of identifying “Unknown,” that is, no number. As a result, it is possible to prevent the wrong identification with a high confidence even though not learning, and it is a great help to improve the reliability of the identification.

Figure 1 shows the distribution of uncertainty when both the model $M_{7_{nobar}}$ learned in a dataset that does not contain 7_{bar} and the model $M_{7_{bar}}$ learned in a dataset that contains 7_{bar} identifies the 7_{bar} as 7. Despite the low uncertainty of $M_{7_{bar}}$, $M_{7_{nobar}}$ shows high uncertainty. By rejecting the identification result if the uncertainty is above a certain value, the accuracy is expected to be improved.

4 Uncertainty of BNN on FPGA

4.1 Binarized Neural Network

As described above, it is possible to improve the reliability of a DNN by using uncertainty. However, MC Dropout has the disadvantage that it takes a lot of computation time. MC Dropout requires at least 20 samplings of the identification result to obtain the variance of the identification result. This sampling is not a serious problem in a large-scale computer with abundant computer resources or an application that does not require real-time performance. Although, it can

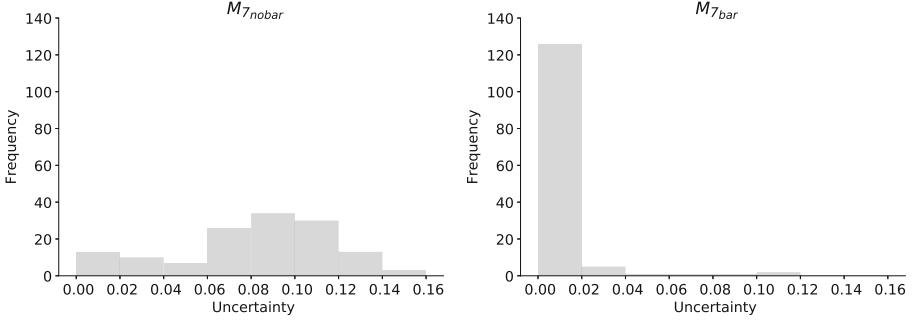


Fig. 1. Distribution of uncertainty when both the model M_{7nobar} learned in a dataset that does not contain 7_{bar} and the model M_{7bar} learned in a dataset that contains 7_{bar} identifies the 7_{bar} as ‘7’ ($n = 136$).

be a problem in situations where computer resources are limited or in real-time is required, such as automobiles.

Therefore, the research found a method to calculate uncertainty without compromising real-time performance. The method uses parallel computing using the characteristics of FPGA in edge computing with an FPGA to utilize MC Dropout. To realize MC Dropout on an FPGA, a network was designed to include dropout for the FPGA, improved the calculation by pre-dropout, and parallelized dropout.

First, the paper will show the network implemented in an FPGA. Since an FPGA does not have enough memory or computational resources compared with GPGPU, a binarized neural network (BNN) was used; it replaces inputs, outputs, and weights from floating-point values to binary values. While the implementation of BNN by various methods has been proposed, implementation by the FINN algorithm was used for the implementation and the computational cost. FINN is an implementation of the BNN proposed by Xilinx, which achieves advanced parallelization in BNN. The network that was implemented for this paper was based on FINN and is shown in Fig. 2.

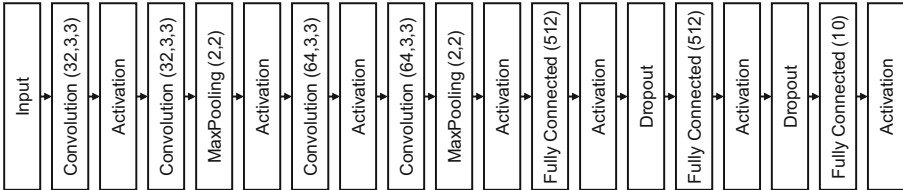


Fig. 2. Network architecture based on FINN algorithm

This network has several convolutional layers, max pooling layer, fully connected layers, and activation layers. The activation layer activate neuron upon

reaching a special threshold. This special threshold is a unique feature of FINN that reduces the amount of computation by simultaneously performing batch regularization and activation. If the output value of a neuron is above the threshold, the output of the neuron is activated.

4.2 Pre-dropout

This section describes how to implement dropout efficiently on an FPGA. When implementing dropout, there is usually a dropout layer behind a fully connected layer and an activation layer, which removes the neurons probabilistically (Fig. 3). However, if the neurons to be deleted are determined stochastically every time during inference, the computation time increases accordingly. Therefore, a pre-dropout method was developed that can perform dropout at the same time as activation by the threshold described above.

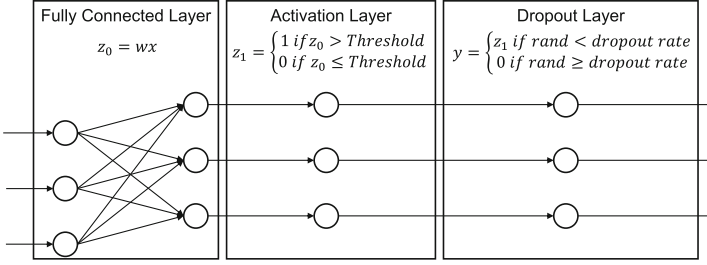
The pre-dropout method is described below (Fig. 4). First, set the part of the threshold that is chosen at random in advance to the maximum, and create a set of this thresholds corresponding to the number of samplings. Next, infer according to the threshold during the inference phase. If the threshold is the maximum value, neurons are always inactive, which allows for probabilistic neuron inactivation. Finally, by switching the set of thresholds for each sampling, it is possible to obtain an output distribution such as MC Dropout. Since it is necessary to prepare a set of thresholds only for the number of samplings, it will use more memory. However, since the memory capacity for thresholds is not large, there is no problem in practice.

4.3 Localization of Parallelization

The following network was created to efficiently parallelize MC Dropout with a DNN on an FPGA. The conceptual diagram of a parallelized circuit is shown in Fig. 5. Since dropout applies only to the fully connected layer following the convolution layer; the calculation result of the convolution layer upstream from it is the same for each sampling. Therefore, process is continued up to the convolution layer common to each sampling, locally parallelizing the processing behind the convolution layer. First, N_{mc} groups of fully connected layer circuits are prepared for a group of convolutional layer circuits. Then, after replicating the results of the convolution layer by the N_{mc} , input respectively in the fully connected layer circuit. Finally, all the output results from the fully connected layer circuit are received, obtained as an inference result of N_{mc} times. Using the average and variance of these results, we obtain inference results and uncertainty.

5 Evaluation

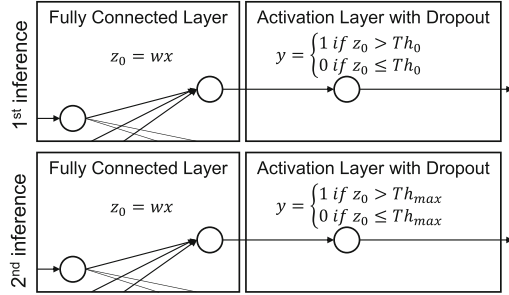
To confirm the effectiveness of the proposed method, evaluation was done using two types of dataset. The first is the MNIST dataset of handwritten number images, which is widely used as an evaluation dataset for DNNs that classify

**Fig. 3.** Dropout without pre-dropout method

1. Building a threshold table in advance to dropout

Th	Th^1	Th^2	...	$Th^{N_{mc}}$
Th_0	Th_0	Th_{max}		Th_0
Th_1	Th_{max}	Th_1		Th_1
Th_2	Th_2	Th_{max}		Th_2
Th_3	Th_{max}	Th_3		Th_{max}
Th_4	Th_4	Th_4		Th_{max}
	\vdots			
Th_k	Th_k	Th_{max}		Th_k

2. Use thresholds in the threshold table for each sampling

**Fig. 4.** Dropout with pre-dropout method

handwritten numbers. The second is the dataset of the ship captured images by the satellite, Advanced Land Observing Satellite 2 (ALOS-2), which was developed and launched by Japan Aerospace Exploration Agency. ALOS-2 has a L-band synthetic-aperture radar (SAR) sensor, which can capture images of ships on the ocean. Label data indicating the type of ship was created based on the information from the AIS (Automatic Identification System) using the latitude and longitude of the captured ship image. Figure 6 shows ship images.

To evaluate uncertainty for these two datasets, a part of data was chosen as “unknown” data to be detected. For the MNIST dataset, special labels were given to the 7 with a horizontal bar (7_{bar}), which is considered to be unknown data. The presence or absence of a horizontal bar was determined manually by eye. For the Ship SAR dataset, special labels were attached to ships longer than a certain length. Unknown data are removed from the learning dataset but included in the test dataset. This operation on the dataset can determine uncertainty about the dataset that is unknown to the model. The number of data and labels contained in the dataset are shown in Tables 2 and 3. An image in the MNIST dataset is 28×28 pixels, and an image in the Ship SAR dataset is 60×60 pixels.

Using these datasets, models are learned with the Theano Python library on GPU, which is NVIDIA TITAN V. After learning, the parameters of the weights and batch regularization are converted based on the algorithm of FINN.

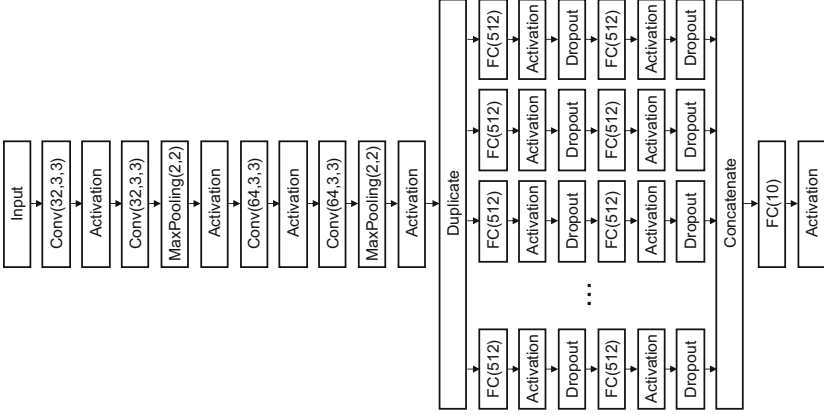


Fig. 5. Parallelization of dropout

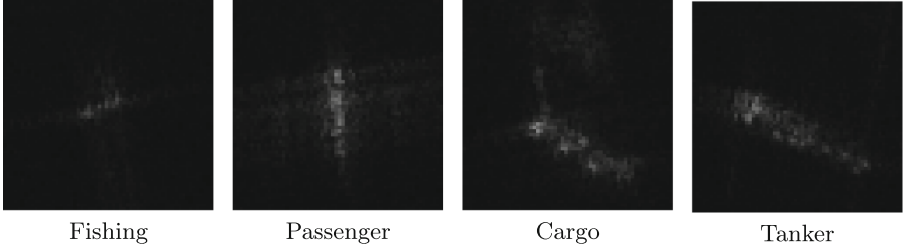


Fig. 6. Ship images captured by ALOS-2 with synthetic-aperture radar (SAR) sensor

The model used for the MNIST dataset is shown in Fig. 2 and the model used for the Ship SAR dataset is shown in Fig. 7. Since the input image size of Ship SAR dataset is larger than MNIST dataset, convolutional layers were added to the Ship SAR model. Since the image size of Ship SAR is larger than MNIST, a convolutional layer was added to the Ship SAR model. For the inference phase, Xilinx Zynq-7000 SoC ZC706 was used, which has 19.1 Mbit Block RAM, 218.6k Look-Up Tables and 437k Flip-Flops, whose size is assumed to be mounted on satellite.

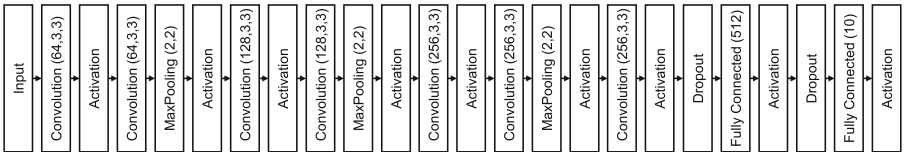


Fig. 7. Network architecture used for the Ship SAR dataset

Table 2. MNIST dataset

Label	Train	Test
0	5,923	980
1	6,742	1,135
2	5,958	1,032
3	6,131	1,010
4	5,842	982
5	5,421	892
6	5,918	958
7 (7_{nobar})	5,463	892
8	5,851	974
9	5,040	1,009
Unknown (7_{bar})	802	136

Table 3. Ship SAR dataset

Label	Train	Test
Fishing	187	34
Passenger	55	15
Cargo	1,170	178
Tanker	486	82
Unknown (Length > 265 m)	32	227

The viewpoints of evaluation are the rejection performance, circuit scale, and inference speed. The rejection performance is evaluated by two scores, Uncertainty Recall and Uncertainty Precision, especially defined for the experiment. Uncertainty Recall is the number that MC Dropout can reject as uncertain for a given input image that should be uncertain. Uncertainty Precision is a number that was unquestionably uncertain in the inference results determined to be uncertain by MC Dropout. The equations used to calculate these are shown in Eqs. 4 and 5.

$$Uncertainty\ Recall = \frac{|\{x \in U \mid var(x) \geq thresholds\}|}{|\{x \in U\}|} \quad (4)$$

$$Uncertainty\ Precision = \frac{|\{x \in U \mid var(x) \geq thresholds\}|}{|\{x \in X \mid var(x) \geq thresholds\}|} \quad (5)$$

where, all sets of input are X , the set of inputs that are uncertain is U , and the threshold to determine that it is uncertain is $thresholds$. The circuit scale measures the increase in circuit scale when compared without MC Dropout. Inference speed is measured time it takes to infer 20 samples compared to the case of not using MC Dropout.

6 Result

The evaluation results for the rejection performance are shown in Table 4. Threshold refers to the threshold values above which are considered to be uncertain. Each thresholds are selected so that Uncertainty Recall is maximized. In the MNIST dataset, Uncertainty Recall was 0.67 and Uncertainty Precision was 0.33 without binarization on a GPU. On the other hand, Uncertainty Recall was 0.66 and Uncertainty Precision was 0.50 with binarization on an FPGA. In the

Ship SAR dataset, Uncertainty Recall was 0.68 and Uncertainty Precision was 0.11 without binarization on a GPU. On the other hand, Uncertainty Recall was 0.63 and Uncertainty Precision was 0.15 with binarization on an FPGA. In both cases, the binary networks on the FPGA performed almost as well as the floating-point networks on the GPU. This result demonstrates that MC Dropout is useful for improving the reliability of edge computing using an FPGA because a BNN can obtain sufficient rejection performance.

Next, Table 5 shows the results of the overhead evaluation by MC Dropout. In the circuit of the MNIST model, utilization of block RAM, Flip-Flops, and Look-Up Table increased about two to three times for a parallelism of 20, and the Ship SAR model had almost the same results. From result, it is clear that the amount of increase in the circuit scale is sufficiently suppressed for the increase in the degree of parallelism by the local parallelization method. When the frequency of parallelism was 20, it took 9.341 ms to infer a image with the MNIST model whereas it takes 16.880 ms when the parallelism is 1, and it took 21.138 ms to infer an image with the Ship SAR model whereas it takes 76.540 ms when the parallelism is 1. Therefore, by parallelization, the MNIST model achieves about 1.81 times faster speeds, and the Ship SAR model achieves about 3.62 times faster speeds.

Table 4. Rejection performance

Dataset	Hardware	Threshold	Uncertainty recall	Uncertainty precision
MNIST	GPU (floating-point)	1.0×10^{-2}	0.67	0.33
MNIST	FPGA (binary)	8.0×10	0.66	0.50
Ship SAR	GPU (floating-point)	1.0×10^{-2}	0.68	0.11
Ship SAR	FPGA (binary)	7.1×10	0.63	0.15

Table 5. Utilization of circuit and inference time for sampling

Model	Frequency of parallel	Utilization of circuit			Inference time for 20 samples
		Block RAM	Flip-Flops	Look-Up Table	
MNIST	1	23%	9%	11%	16.880 ms
MNIST	20	72%	24%	28%	9.341 ms
Ship SAR	1	31%	11%	13%	76.540 ms
Ship SAR	20	93%	20%	23%	21.138 ms

7 Conclusion

This paper proposed a method to reject uncertain judgment results using MC dropout on an FPGA to improve the reliability of a DNN used in edge computing.

To implement MC dropout on an FPGA, it was necessary to binarize a neural network, to include a simplified dropout layer via the pre-dropout method, and to localize parallelized parts of a circuit. The evaluation showed that, using the proposed method with the MNIST and Ship SAR dataset, approximately 60% of unlearned data were rejected as “uncertain” on the FPGA. Furthermore, even if the parallelism had a frequency of 20, the circuit scale would only increase to about 2–3 times that of non-parallelized circuits. The inference speed of the parallelized circuits was achieved up to 3.62 times faster than of non-parallelized circuits. Therefore, the proposed method was able to improve the reliability of a DNN by detecting uncertain judgments implementing MC Dropout on an FPGA using edge computing.

References

1. Feng, D., Rosenbaum, L., Dietmayer, K.: Towards safe autonomous driving: capture uncertainty in the deep neural network for Lidar 3D vehicle detection. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (2018). <https://doi.org/10.1109/itsc.2018.8569814>
2. Gal, Y., Ghahramani, Z.: Bayesian convolutional neural networks with Bernoulli approximate variational inference. In: 4th International Conference on Learning Representations (ICLR) Workshop Track (2016)
3. Gal, Y., Ghahramani, Z.: Dropout as a Bayesian approximation: representing model uncertainty in deep learning. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML 2016, vol. 48, pp. 1050–1059. JMLR.org (2016)
4. Hashimoto, S., Sugimoto, Y., Hamamoto, K., Ishihama, N.: Ship classification from SAR images based on deep learning. In: Arai, K., Kapoor, S., Bhatia, R. (eds.) *Intell. Syst. Appl.*, vol. 868, pp. 18–34. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-01054-6_2
5. Kendall, A., Gal, Y.: What uncertainties do we need in Bayesian deep learning for computer vision? pp. 5574–5584. Curran Associates, Inc. (2017)
6. Miller, D., Dayoub, F., Milford, M., Sünderhauf, N.: Evaluating merging strategies for sampling-based uncertainty techniques in object detection. [arXiv:1809.06006](https://arxiv.org/abs/1809.06006) [cs.CV] (2018)
7. Myojin, T., Hashimoto, S., Mori, K., Sugawara, K., Ishihama, N.: Improving reliability of object detection for lunar craters using Monte Carlo dropout. In: Tetko, I.V., Kůrková, V., Karpov, P., Theis, F. (eds.) *ICANN 2019*. LNCS, vol. 11729, pp. 68–80. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30508-6_6
8. Takamaeda, S.: NNgen/nnngen (February 2020). <https://github.com/NNgen/nnngen>
9. Umuroglu, Y., et al.: FINN: a framework for fast, scalable binarized neural network inference. In: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, FPGA 2017, pp. 65–74. Association for Computing Machinery, New York, NY, USA (2017). <https://doi.org/10/ggmnh6>