G-2

A Study of Parallelism of Probabilistic Neural Network Using GPU
A Comparison of the Sequential Processing with CPU and Parallel Processing with GPU

*Shunpei Morita[1], Tetsuya Hoya[2]

Abstract: While probabilistic neural network (PNN), originally proposed for classification problems, has many advantages, such as inherently high parallelism, it has long been conceived that the reference mode is slower than other classification methods. In this study, we therefore focus on the parallelism and actually implement the PNN model in a parallel environment using a GPU. In the simulation study, we compared the parallelly implemented version of a PNN using GPU against that operated in ordinary serial mode of the CPU. We then observed that the processing time for the case of the GPU was improved over the ordinary CPU case for 6 out of 13 publicly available datasets.

I. INTRODUCTION

Probabilistic neural network (PNN) [1] is a layered neural network model. While PNN has many advantages such as fast learning, high parallelism, and high robustness against an adversarial attack [2], it has an inherent problem of slow processing in the reference mode. In this work, we focus upon the study of improving the processing speed in the reference mode by implementing a PNN in the parallel computing environment using a GPU.

II. THE PROBABILISTIC NEURAL NETWORK

PNN is a three layered neural network, with a single hidden layer consisting of radial basis functions (RBFs). Figure 1 illustrates the structure of PNN. The output value from each of the RBF layer units and that of the output layer units in a PNN are respectively defined as:

$$h_{c(j)} = \exp\left(-\frac{\left\|x - c_{C(j)}\right\|_2^2}{\sigma^2}\right) \quad (1)$$

$$o_c = \frac{1}{U_C}\sum_{j=1}^{U_C} h_{c(j)} \quad (2)$$

where $\|...\|_2$ denotes *L2*-norm, and $h_{C(j)}$, $c_{C(j)}$, o_C, and U_C are the output value of the j-th unit in the class C in the RBF layer, the attribute vector of the j-th unit in the class C, the output value of the output layer, and the number of training data in each class, respectively.

As in (2), the activation of each RBF is based upon the computation of the Euclidean distance (i.e. that given by the L2-norm) between the attribute and target vector, where each attribute vector corresponds to a single training data. The label of the output unit with the highest value yields the classification result of the target vector given.

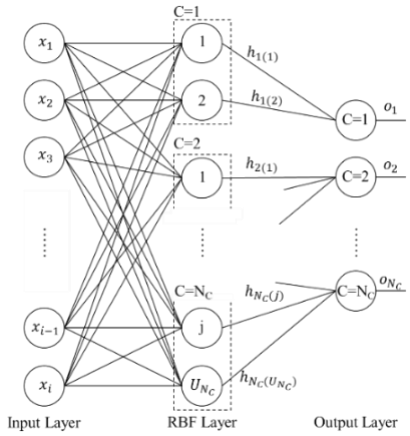

Figure 1. The structure of PNN

Table 1. A comparison of time-steps

Operation	Time-steps		#ID
	Serial	Parallel	
$s_{i\_C(j)} \leftarrow x_i - c_{i\_C(j)}$	L	1	1
$s_{i\_C(j)} \leftarrow s^2_{i\_C(j)}$	L	1	2
$h_{C(j)} \leftarrow \sum_{i=1}^{L} s_{i\_C(j)}$	$L \times Tr$	L	3
$h_{C(j)} \leftarrow h_{C(j)}/\sigma^2$	Tr	1	4
$h_{C(j)} \leftarrow exp(-h_{C(j)})$	Tr	1	5
$o_C \leftarrow \sum_{j=1}^{U_{N_C}} h_{C(j)}$	$U_1 + U_2 + \cdots\cdots + U_{N_C}$	$max(U_C)$	6
$o_C \leftarrow o_C/U_{N_C}$	N_C	1	7

III. PARALLEL IMPLEMENTATION OF THE PROBABILISTIC NEURAL NETWORK

It is known that the operations in the RBF layer and the output layer can be executed in parallel, since their operations do not depend on the outcomes from the other units in the same layer[3]. The comparison of time-step in serial and parallel

1: Department of Computer Science, CST, Nihon-U　2: Department of Computer Engineering, CST, Nihon-U

processing is shown in Table 1, where L is the number of attributes and Tr is the number of training data. In the table, since all the operations in the PNN can be executed in parallel, we consider that it is possible to increase the classification speed using a sufficient number of processing units i.e. those available within the GPU.

IV. SIMULATION STUDY

To compare the time-steps for classification in both the serial and parallel processing (except for ID 6 and 7) as summarized in Table 1, we conducted a series of the simulations with 13 publicly available datasets; 12 of 13 obtained from the UCI machine learning repository [4] and one remaining from the MNIST [5]. Then, each dataset is normalized within the range $[-1, 1]$. Tables 2 and 3 show the summary of the computing environment and the 13 datasets used for the simulation study, respectively.

Table 2. Computing environment for the simulation study

CPU	AMD Ryzen 7 3700X
GPU	NVIDIA GeForce RTX 3080
Programming Language	C++ (gcc 9.4.0, cuda 11.7)

Table 3. Summary of the 13 datasets

Dataset	#Features per pattern	#Classes	#Training	#Testing
abalone	7	3	2088	2089
balance-scale	4	3	428	187
cmc	9	3	1000	473
ionosphere	33	2	200	151
isolet	617	26	2252	1559
letter-recognition	16	26	16000	4000
MNIST	784	10	60000	10000
optdigits	64	10	3823	1797
pendigits	16	10	7494	3498
statlog	36	6	4435	2000
tic-tac-toe	9	2	663	295
wdbc	30	2	398	171
yeast	8	10	890	594

V. SIMULATION RESULTS

The simulation results are summarized shown in Table 4. In the table, each processing time shown is the averaged classification time per pattern actually taken during the testing for the case of using the CPU and GPU, respectively.

As in Table 4, the processing time was improved for the case of using the GPU where the 6 datasets of isolet, letter-recognition, MNIST, optdigits, pendigits, and statlog among the 13, comparing with the case of the CPU. For the rest, i.e. abalone, balance-scale, cmc, ionosphere, tic-tac-toe, wdbc, and yeast, the processing time was slower than the case of the serial, where these 7 datasets have fewer training data than the other 6 datasets. From these results, it is considered that the cause of the unexpected slower processing for the case of the GPU was the extra time required for the data transaction during performing the operations from ID 3 to 5 in Table 1.

Table 4. Summary of the simulation results

Dataset	Processing time per pattern [μs]		Accuracy [%]
	CPU avg.	GPU avg.	
abalone	20.721	96.040	58.07
balance-scale	3.316	96.294	67.91
cmc	13.243	96.476	46.30
ionosphere	8.172	99.027	89.40
isolet	1733.150	333.513	77.10
letter-recognition	523.827	124.043	96.20
MNIST	66065.800	3876.020	96.81
optdigits	270.519	153.794	98.16
pendigits	212.002	112.436	97.31
statlog	183.862	117.001	82.70
tic-tac-toe	8.312	93.197	75.59
wdbc	14.491	94.561	97.08
yeast	10.212	97.849	52.19

VI. CONCLUSION

In this study, we have compared the classification time of a PNN operated in both the serial and parallel environment. It has been empirically shown that the processing time in the parallel mode can be improved in the case of the datasets with a sufficient number of the training data. Future work includes the comparison of the processing time between the PNN and other classification methods operated in parallel situations.

VII. REFERENCES

[1] D. F. Specht: Probabilistic Neural Networks. Neural Networks 3. 109-118. 1990.

[2] I. J. Goodfellow, J. Shlens, and C. Szegedy: Explaining and Harnessing Adversarial Examples. International Conference on Learning Representations 2015. https://arxiv.org/abs/1412.6572. last accessed 2022/09.

[3] K. Takahashi, S. Morita, and T. Hoya: Analytical Comparison between the Pattern Classifiers Based upon A Multilayered Perceptron and Probabilistic Neural Network in Parallel Implementation. Artificial Neural Networks and Machine Learning – ICANN2022 3. 544-555. 2022.

[4] D. Dheeru, and G. Casey: UCI Machine Learning Repository. Irvine, CA. http://archive.ics.uci.edu/ml. last accessed 2022/09.

[5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner: The MNIST Database. http://yann.lecun.com/exdb/mnist/. last accessed 2022/09.