A Software-Hardware Co-exploration Framework for Optimizing Communication in Neuromorphic Processor

Shiying Wang, Lei Wang(✉), Ziyang Kang, Lianhua Qu, Shiming Li, and Jinshu Su

School of Computer, National University of Defense Technology, Changsha, China
{wangshiying18,leiwang,kangziyang14,qulianhua14,
lishiming15,sjs}@nudt.edu.cn

Abstract. Spiking neural networks (SNN) has been widely used to solve complex tasks such as pattern recognition, image classification and so on. The neuromorphic processors which use SNN to perform computation have been proved to be powerful and energy-efficient. These processors generally use Network-on-Chip (NoC) as the interconnect structure between neuromorphic cores. However, the connections between neurons in SNN are very dense. When a neuron fire, it will generate a large number of data packets. This will result in congestion and increase the packet transmission latency dramatically in NoC.

In this paper, we proposed a software-hardware co-exploration framework to alleviate this problem. This framework consists of three parts: software simulation, packet extraction&mapping, and hardware evaluation. At the software level, we can explore the impact of packet loss on the classification accuracy of different applications. At the hardware level, we can explore the impact of packet loss on transmission latency and power consumption in NoC. Experimental results show that when the neuromorphic processor runs MNIST handwritten digit recognition application, the communication delay can be reduced by 11%, the power consumption can be reduced by 5.3%, and the classification accuracy can reach 80.75% (2% higher than the original accuracy). When running FSDD speech recognition application, the communication delay can be reduced by 22%, the power consumption can be reduced by 2.2%, and the classification accuracy can reach 78.5% (1% higher than the original accuracy).

Keywords: Neuromorphic processor · Communication optimization · Network-on-Chip · Reservoir computing

1 Introduction

Neuromorphic computing is an important branch of artificial intelligence. The concept of Neuromorphic computing was proposed by Carver Mead [10], which refers to the use of large-scale integrated circuit systems to simulate the neurobiological structure existing in the nervous system to complete the calculation of large-scale

D. Dong et al. (Eds.): ACA 2020, CCIS 1256, pp. 87–100, 2020.
https://doi.org/10.1007/978-981-15-8135-9_7

neural networks. Recently, some excellent neuromorphic processors have emerged, such as SpiNNaker [6], BrianScales [18], TrueNorth [1], DYNAPs [16], Loihi [4], and Tianjic [17]. All of these neuromorphic processors contain thousands of neurons and synaptic connections. For better scalability and parallelism, they mostly use Network-on-Chip (NoC) to make connections between neurons.

In general, Spiking neural network (SNN) is used in the neuromorphic processor. SNN is the third generation artificial neural network (ANN) [15] which is originally inspired by brain science. It has been proved to be a powerful and energy-efficient computation model and has been applied to complex tasks such as pattern recognition [20], image classification [9], natural language processing [5], etc. Many SNN models can be implemented on the neuromorphic processor, such as multilayer perceptron (MLP), spiking convolutional neural network (SCNN) [8], reservoir computing (RC) [21], etc. Compared with other models, RC has more advantages in terms of application scope and implementation complexity.

Unlike other SNN models, neurons in the RC model can be connected not only to other neurons but also to themselves. This characteristic makes the connections denser among the neurons. Figure 1 shows the distribution of the number of neuron connections in an RC network. The horizontal axis represents the number of connections. The vertical axis refers to the number of neurons corresponding to a specific number of connections. We can find that all neurons are connected to at least one-third of the neurons in the RC.

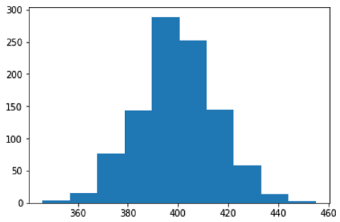

Fig. 1. Connections distribution in an RC network: 1000 neurons in total, 800 excitory neurons, 200 inhibit neurons. $P_{ee} = 0.40$, $P_{ei} = 0.40$, $P_{ie} = 0.50$.

In SNN, one pre-neuron cloud connects a large number of post-neurons. When the pre-neurons membrane voltage exceeds its threshold and fire, the same count of spike will be generated. Therefore, when the RC network is running in a neuromorphic processor, a large number of data packets will be generated at the same time. This will cause packet congestion in NoC and increase packet transmission latency. Figure 2 is a simple running process in a neuromorphic processor. Neuron A connects three neurons B, C, and D. They are mapped to different neuromorphic cores. At time-step t, the membrane voltage of A exceeds its threshold. Then, neuron A generates a spike, and send it to all the post-neurons connected to it. Because the router cannot transmit all packets immediately, the final packet has a greater transmission latency than other packets.

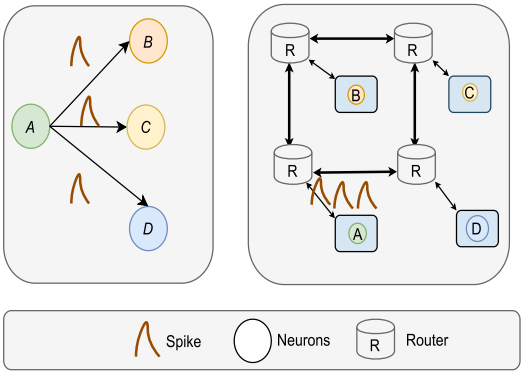

Fig. 2. The running process of SNN in neuromorphic processor. Neuron A connects three neurons B, C, and D. At time-step t, neuron A generates a spike.

When a large number of packets need to be transmitted at the same time, it will cause NoC congestion. At the same time, through experiments, we find that when a small number of data packets are discarded, it doesn't affect the classification accuracy of the software. Based on the above problems and phenomena, in this paper, we proposed a software-hardware co-exploration framework to optimize the communication in a neuromorphic processor. This framework consists of three parts: software simulation, packet extraction & mapping, and hardware evaluation. According to it, we can optimize the communication in the neuromorphic processor while ensuring the accuracy of software recognition. To this end, we make the following contributions:

1) We explored the impact of packet loss on the accuracy of SNN classification. It is found that a low spike loss rate doesn't affect the classification accuracy of SNN. In some cases, the classification accuracy is even higher than the original accuracy.
2) We propose a software-hardware co-exploration framework for optimizing communication in a neuromorphic processor. It can alleviate the congestion in NoC and reduce the packet transmission latency without reducing the accuracy of SNN classification.

After optimization, when the neuromorphic processor runs MNIST handwritten digit recognition application, the communication delay can be reduced by 11%, the power consumption can be reduced by 5.3%, and the classification accuracy can reach 80.75% (2% higher than the original accuracy). When running FSDD speech recognition application, the communication delay can be reduced by 22%, the power consumption can be reduced by 2.2%, and the classification accuracy can reach 78.5% (1% higher than the original accuracy).

2 Background and Related Work

2.1 RC Model

Reservoir calculation (RC) is a type of RNN model, which is mainly composed of an input layer, reservoir layer, and output layer, as shown in the Fig. 3. The reservoir layer consists of a certain number of excitatory spiking neurons and inhibitory spiking neurons. According to the complexity of the task, the number of two types of neurons and the connection probability between them are different. W_{In}, W_R, and W_{Out} represent the weights of the input layer, reservoir layer, and output layer, respectively. After the reservoir layer receives the input spike train and runs for a certain period, it will generate an RC state (usually the membrane voltage of the neurons in the RC). The output layer records the state of neurons in the reservoir layer.

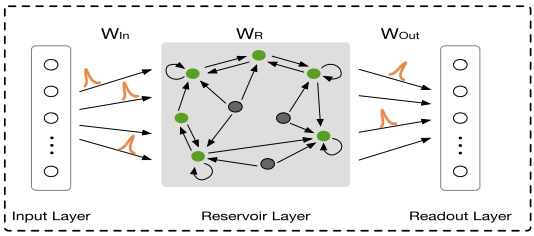

Fig. 3. The architecture of RC.

2.2 Network-on-Chip

With the advent of multi-core processors, the design interconnect of System on Chip (SoC) faces many challenges. SoCs typically use a Network on Chip (NoC) [2,12] solution, with various NoC topologies and router architectures, and provide low power and high quality of service (QoS) designs. NoC has many components, such as topology, routing algorithms, and router microstructure design.

TrueNorth [1] is a neuromorphic chip developed by IBM. In TrueNorth, its time-step is 1 ms. Operation in each time-step is divided into two phases: In the first phase, data packets will be routed through the Router. When the data packet reaches the corresponding core, it will change the membrane voltage of the corresponding neuron. In the second phase, all cores will receive a synchronizing signal with a period of 1 ms. Once the synchronizing signal is received, all neurons need to check whether their membrane voltage exceeds the threshold. If the threshold is exceeded, the neuron will send a data packet to the network.

TrueNorth uses a global synchronous clock to synchronize each time-step, so the size of the global clock must consider the worst case in the entire chip. However, not all packets have a large transmission latency. So its synchronization method will reduce the efficiency of the hardware.

3 The Software-Hardware Co-exploration Framework

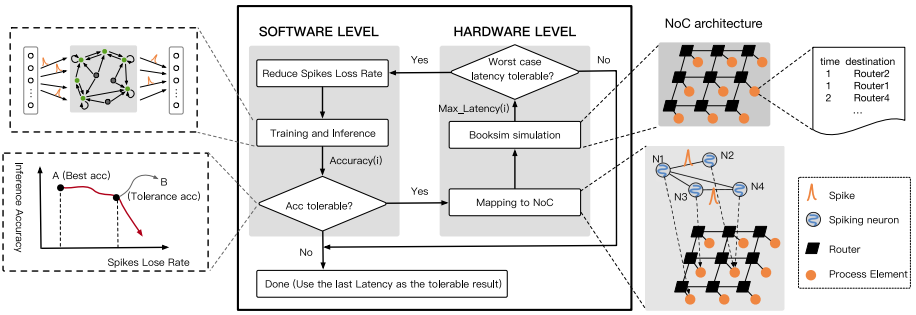

Fig. 4. Software hardware co-exploration framework.

3.1 Real-Time Definition

Define that a synchronization period in neuromorphic processor consumes L cycles. When an application is running in a neuromorphic processor, if the data packets generated by it meet the following conditions, we think that the application meets the real-time requirements of the neuromorphic processor:

$$Max(Latency_i) < L \tag{1}$$

Among them, i belongs to $[0, N-1]$, N is the total amount of data packets generated by the application in the inference process. In other words, the transmission latency of all packets is less than the length of the synchronization cycle.

3.2 Framework

Our proposed framework is shown in Fig. 4. It mainly composed three parts:

Software Level Simulation. We use the SNN simulator to run the RC network. When neurons communicate, we drop packets with different probabilities and explore the impact of different packet loss rates on software classification accuracy.

Many SNN simulators can be used, such as Brian2, CARLsim [3], Nest [7], etc. They can be used to simulate the behavior of neurons and simulate the operation of SNN. At the same time, log files can be extracted during the running process. During initialization, we can determine the connection probability and the weight of the connection in the network. After initialization, we extract the connection relationship.

We explore the impact of packet loss on the accuracy of SNN classification as follows: 1) Generate application input spike. 2) Train the readout layer, perform

classification, and finally test multiple test-sets to get the accuracy of the classification. 3) Change the different spike loss rates to get the relationship between classification accuracy and spike loss rate. By dropping part of the data packets, the communication delay in the neuromorphic processor can be reduced. Synchronization in the neuromorphic processor must satisfy the worst communication situation. Therefore, this can reduce the number of clock cycles required for global synchronization.

Trace Extraction and Mapping. The spike trace can be recorded when *Brian*2 is simulating. Each trace records spikes from one neuron to another neuron. The format of each trace in the trace file is:

$$[Source\ Neuron\ ID, Destination\ Neuron\ ID, time\text{-}step]$$

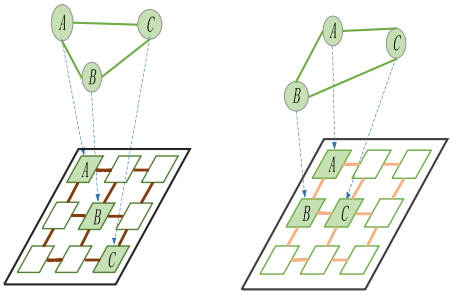

Fig. 5. Two different mapping methods.

Mapping is a process that map neurons in SNN to cores of neuromorphic processor. Figure 5 shows two different mapping methods [14].

We use NoC simulator to simulate the communication between the cores in the neuromorphic processor when the SNN network is running.

Hardware Level Evaluation. At the hardware level, we simulate communication between neuromorphic cores through a NoC simulator. The detailed hardware evaluation process is shown in Fig. 6. As mentioned before, the parameters of NoC include topology, routing algorithm, router micro-architecture, etc. In this work, we use a 2D-Mesh network structure and a dimension-order-first routing algorithm to route packets.

As shown in Fig. 6, NoC configuration file and trace files are inputs for hardware evaluation, latency and power consumption are outputs. Throughput refers to the number of data packets transmitted by the NoC within a specified time. In neuromorphic processors, each spike is a data packet. Throughput is generally defined by the following formula (2):

$$Throughput = \frac{(Total\ Spike\ packet\ finished) \times Packet_{Length}}{(Numbers\ of\ Router) \times Total_{time}} \tag{2}$$

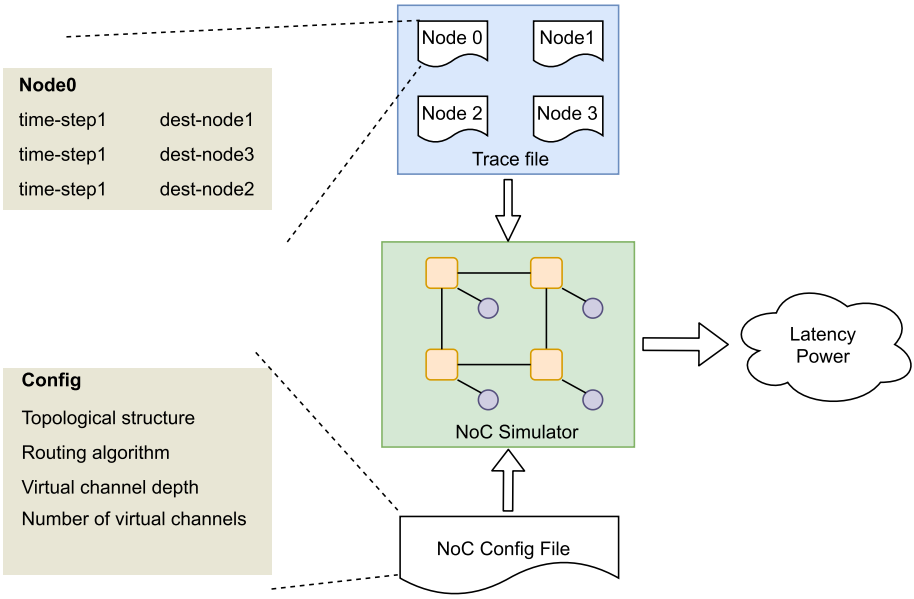

Fig. 6. Hardware level simulation.

The $Packet_{Length}$ is measured in flits. Each packet can be represented by a spike. So we defined the $Packet_{Length} = 1$.

Let P be the total number of messages reaching their destination and let L_i be the latency of each message i, where i ranges from 1 to F, as shown in formula (3):

$$L_i = receiving\ time\ of\ P_i - generating\ time\ of\ P_i \tag{3}$$

$Max\ transport\ latency$ is computed as shown in formula (4):

$$L_{Transport\ latency} = Max(L_i), 1 \leq i \leq F \tag{4}$$

3.3 Framework Workflow

The framework workflow is shown in Algorithm 1. We can tolerate a 1% reduction in classification accuracy at most. If the real-time requirements are not met, the packet loss rate will increase by 5% at a time. If the real-time requirement can be met, the packet loss rate and classification accuracy will be output. If the requirement is not met, the framework will explain the situation and exit.

Algorithm 1. Framework workflow

Input: *Real_time_Latency*
Output: *spike_loss_rate*, *Accu*

Real_time_Latency//Maximum packets transmission latency that meet real-time requirements;
$spike\_loss\_rate = 0$
$Accuracy(i)$//calculation classification accuracy when the spike loss rate is i;
$Tolerance\_acc = Accuracy(0) - 0.01$//The minimum value of classification accuracy we can tolerate;
$Max\_Latency(i)$//Maximun packets transmission latency when the packet discarding rate is i;
$Latency = Max\_Latency(0)$
while $Latency > Real\_time\_Latency$ **do**
  $spike\_loss\_rate+ = 0.05$
  $Accu = Accuracy(spike\_loss\_rate)$
  if $Accu > Tolerance\_acc$ **then**
    $Latency = Max\_Latency(spike\_loss\_rate)$
    if $Latency <= Real\_time\_Latency$ **then**
      return *spike_loss_rate*, *Accuracy*
    else
      continue
    end if
  else
    Classification accuracy reduced too much.
    Unable to meet the real-time requirement.
    break
  end if
end while

4 Experiment and Analysis

4.1 Experiment Setup

To verify the optimization effect of our framework, we set up the experiment as follows:

We use the *Brian*2 simulator [19] to simulate the running of the RC network. Software-level simulation includes two steps: (1) Generate spikes from the data set. The content of the original data set cannot be directly used as the input of the SNN. Therefore, the original data set needs to be extracted and transformed into a spike train that can be understood by the SNN; (2) RC simulation and classification. We input the spike obtained from (1) into the RC network. After a period of time, readout layer reads the state of the RC neurons to train and classify. Through multiple tests, a classification accuracy can be obtained.

We use the clock-accurate NoC simulator *Booksim*2 [11] to simulate the communication process of RC networks in NoC. We have modified Booksim so that it can read spike communication trace. The NoC topology in this work is

8×8. The routing algorithm is x-y routing. The NoC has 8 virtual channels, and the virtual channel depth is 1. Detailed configuration of NoC is shown in Table 1.

We tested two different applications, one is MNIST handwritten digit recognition and the other is FSDD speech recognition.

Table 1. NoC Config.

topology	mesh
routing_function	xy
vc_allocator	islip
arb_type	round_robin
priority	age
num_vcs	8
vc_buff_size	1

4.2 Result of MNIST Dataset

MNIST [13] is a frame-based dataset, which is used for handwritten digits recognition.

Optimization Effect of Framework. Figure 7 shows optimization results of MNIST handwritten digit recognition application. Our target maximum transmission latency is 400 cycles. After several rounds of exploration, when the packet loss rate is 10%, the maximum transmission latency is reduced from the original 436 cycles to 392 cycles. It meets the real-time requirements. After optimization, the power consumption is reduced by 5.3%. At the same time, the classification accuracy didn't decline but increased from 78.75% to 80.75%.

Impact of Packet Loss on Performance of RC Network. Figure 8 shows the relationship between the packet loss rate and software classification accuracy. When the packet loss rate is in the range of 0%–10%, the classification accuracy is almost unchanged. When the packet loss rate exceeds 20%, the classification accuracy decreases significantly.

Impact of Packet Loss on Data Transmission Latency in NoC. Figure 9 shows the relationship between the packet loss rate and the maximum transmission latency in NoC. We can find that as the packet loss rate increases, the maximum transmission latency decreases significantly. Reduced transmission latency can increase hardware efficiency.

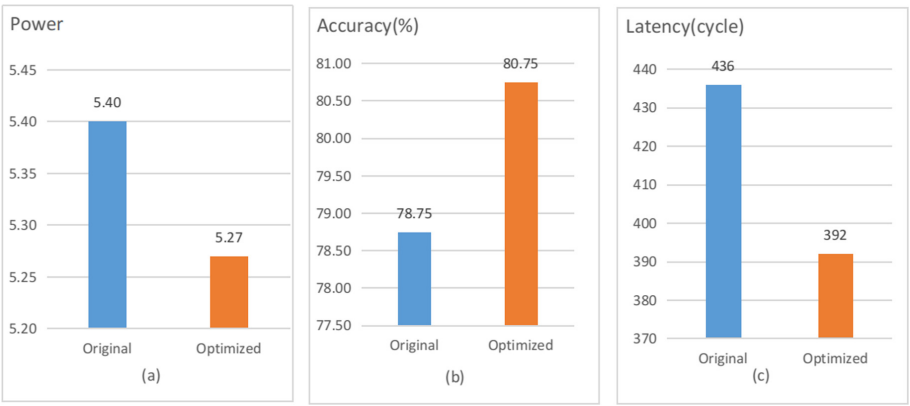

Fig. 7. Optimization results of MNIST handwritten digit recognition application. *Original* refers to the result without optimization. *Optimized* refers to the optimized result. (a) Power. (b) Accuracy. (c) Latency.

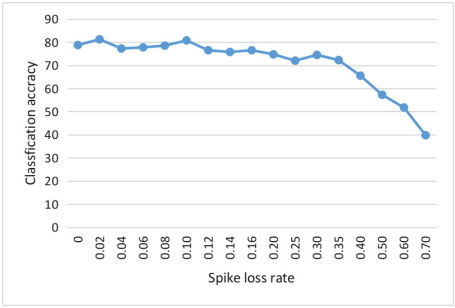

Fig. 8. Impact of packet loss on classification accuracy of MNIST handwritten digit recognition application.

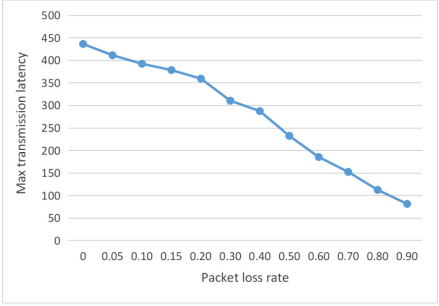

Fig. 9. Impact of packet loss on data transmission latency in MNIST handwritten digit recognition application.

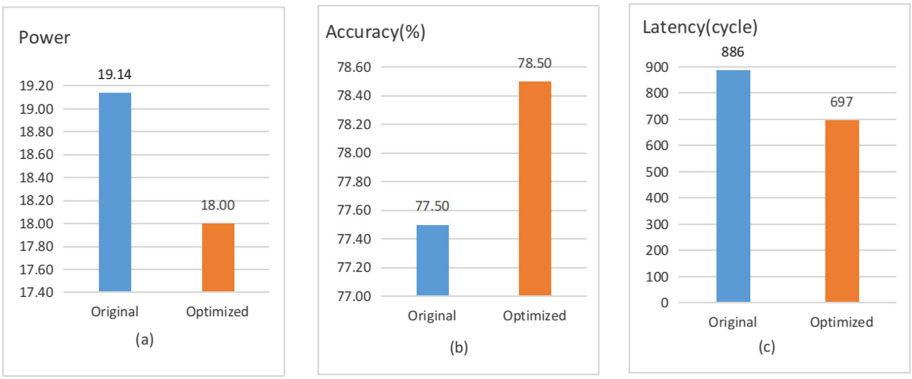

Fig. 10. Optimization results of FSDD speech recognition application. *Original* refers to the result without optimization. *Optimized* refers to the optimized result. (a) Power. (b) Accacury. (c) Latency.

4.3 Result of FSDD Dataset

FSDD is a simple audio and speech dataset consisting of recordings of spoken digits at 8 kHz.

Optimization Effect of Framework. Figure 10 shows optimization results of FSDD speech recognition applications. For FSDD speech recognition applications, our target maximum transmission latency is 700 cycles. After several rounds of exploration, when the packet loss rate is 19%, the maximum transmission latency is reduced from the original 886 cycles to 697 cycles. It meets the real-time requirements. After optimization, the power consumption is reduced by 2.2%. At the same time, the classification rate didn't decline but increased from 77.5% to 78.5%.

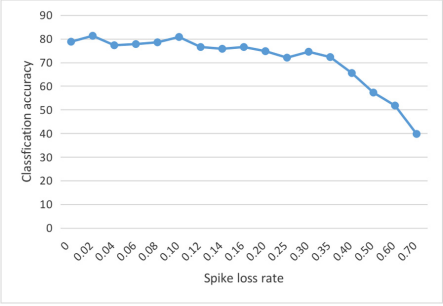

Fig. 11. Impact of packet loss on classification accuracy of FSDD speech recognition application.

Impact of Packet Loss on Performance of RC Network. As shown in Fig. 11, it shows the impact of packet loss on classification accuracy in FSDD speech recognition. When the packet loss rate is in the range of 0%–19%, the classification accuracy is almost unchanged. When the packet loss rate exceeds 19%, the classification accuracy decreases significantly. At the same time, with the packet loss rate is 1%, the classification accuracy is even better than the effect of no packet loss.

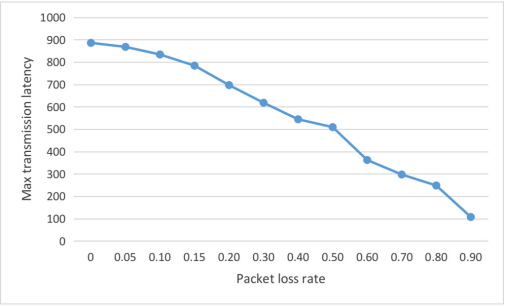

Fig. 12. Impact of packet loss on data transmission latency in FSDD speech recognition application.

Impact of Packet Loss on Packet Transmission Latency in NoC. Figure 12 shows the relationship between the packet loss rate and the maximum transmission latency in NoC for FSDD. Same as the result of MNIST handwriting recognition, we can find that as the packet loss rate increases, the maximum transmission latency decreases significantly.

4.4 Analysis

How Does Packet Loss Affect the Classification Accuracy of RC? As mentioned before, an RC network consists of three layers: the input layer, the reservoir layer, and the output layer. Once the reservoir layer is initialized, the connections between neurons and their corresponding weights no longer change. Training an RC network refers to training its readout layer. The input of the readout layer is the state of the neurons in the reservoir layer.

There are different ways to encode the state of a neuron, such as time encoding and frequency encoding. Frequency encoding means that the state of a neuron is represented according to the count of fire in the time window. Frequency encoding is used in this experiment. Packet loss adds "noise" to the reservoir layer. The inputs for training and testing are the states of the reservoir neurons that contain "noise". These "noises" are randomly generated with a specific probability.

When the packet loss rate is very small, the classifier trained by the dataset containing "noise" can classify normally. At the same time, because the packet loss behavior is random, the corresponding classification accuracy may fluctuate around the original classification accuracy.

When the packet loss rate is too high, the spiking frequency of neurons will decrease. If frequency coding is used, the state difference between neurons will become less obvious. At this time, it's hard for classifier to make right decision. As a result, the accuracy of classification will descend significantly.

5 Conclusion

In this paper, we proposed a software-hardware co-exploration framework for optimizing communication in neuromorphic processor. This framework consists of three parts: software simulation, packet extraction & mapping, and hardware evaluation. Without reducing the accuracy of SNN classification, we can alleviate the congestion in neuromorphic processor and reduce the packet transmission latency.

The experiment result shows that after optimization, when the neuromorphic processor runs MNIST handwritten digit recognition application, the communication delay can be reduced by 11%, the power consumption can be reduced by 5.3%, and the classification accuracy can reach 80.75% (2% higher than the original accuracy). When running FSDD speech recognition application, the communication delay can be reduced by 22%, the power consumption can be reduced by 2.2%, and the classification accuracy can reach 78.5% (1% higher than the original accuracy).

References

1. Akopyan, F., et al.: Truenorth: design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip. IEEE Trans. Comput. Aided Des. Integr. Circ. Syst. **34**(10), 1537–1557 (2015)
2. Beigné, E., Clermidy, F., Vivet, P., Clouard, A., Renaudin, M.: An asynchronous NOC architecture providing low latency service and its multi-level design framework. In: 11th IEEE International Symposium on Asynchronous Circuits and Systems, pp. 54–63. IEEE (2005)
3. Chou, T.S., et al.: CARLsim 4: an open source library for large scale, biologically detailed spiking neural network simulation using heterogeneous clusters. In: 2018 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2018)
4. Davies, M., et al.: Loihi: a neuromorphic manycore processor with on-chip learning. IEEE Micro **38**(1), 82–99 (2018)
5. Diehl, P.U., Zarrella, G., Cassidy, A., Pedroni, B.U., Neftci, E.: Conversion of artificial recurrent neural networks to spiking neural networks for low-power neuromorphic hardware. In: 2016 IEEE International Conference on Rebooting Computing (ICRC), pp. 1–8. IEEE (2016)
6. Furber, S.B., et al.: Overview of the spinnaker system architecture. IEEE Trans. Comput. **62**(12), 2454–2467 (2012)

7. Gewaltig, M.O., Diesmann, M.: Nest (neural simulation tool). Scholarpedia **2**(4), 1430 (2007)
8. Guo, S., et al.: A systolic SNN inference accelerator and its co-optimized software framework. In: Proceedings of the 2019 on Great Lakes Symposium on VLSI, pp. 63–68 (2019)
9. Iakymchuk, T., Rosado-Muñoz, A., Guerrero-Martínez, J.F., Bataller-Mompeán, M., Francés-Víllora, J.V.: Simplified spiking neural network architecture and STDP learning algorithm applied to image classification. EURASIP J. Image Video Process. **2015**(1), 4 (2015). https://doi.org/10.1186/s13640-015-0059-4
10. Izhikevich, E.M.: Simple model of spiking neurons. IEEE Trans. Neural Netw. **14**(6), 1569–1572 (2003). https://ieeexplore.ieee.org/document/1257420
11. Jiang, N., Michelogiannakis, G., Becker, D., Towles, B., Dally, W.J.: Booksim 2.0 user's guide. Standford University (2010)
12. Kumar, S., et al.: A network on chip architecture and design methodology. In: Proceedings IEEE Computer Society Annual Symposium on VLSI. New Paradigms for VLSI Systems Design, ISVLSI 2002, pp. 117–124. IEEE (2002)
13. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
14. Li, S., et al.: SNEAP: a fast and efficient toolchain for mapping large-scale spiking neural network onto NOC-based neuromorphic platform. In: Proceedings of the 2020 on Great Lakes Symposium on VLSI (2020)
15. Maass, W.: Networks of spiking neurons: the third generation of neural network models. Neural Netw. **10**(9), 1659–1671 (1997)
16. Moradi, S., Qiao, N., Stefanini, F., Indiveri, G.: A scalable multicore architecture with heterogeneous memory structures for dynamic neuromorphic asynchronous processors (dynaps). IEEE Trans. Biomed. Circ. Syst. **12**(1), 106–122 (2017)
17. Pei, J., et al.: Towards artificial general intelligence with hybrid Tianjic chip architecture. Nature **572**(7767), 106–111 (2019)
18. Schemmel, J., et al.: Live demonstration: a scaled-down version of the brainscales wafer-scale neuromorphic system. In: 2012 IEEE International Symposium on Circuits and Systems, pp. 702–702. IEEE (2012)
19. Stimberg, M., Brette, R., Goodman, D.F.: Brian 2, an intuitive and efficient neural simulator. Elife **8**, e47314 (2019)
20. Wysoski, S.G., Benuskova, L., Kasabov, N.: Fast and adaptive network of spiking neurons for multi-view visual pattern recognition. Neurocomputing **71**(13–15), 2563–2575 (2008)
21. Young, A.R., Dean, M.E., Plank, J.S., Rose, G.S.: A review of spiking neuromorphic hardware communication systems. IEEE Access **7**, 135606–135620 (2019)