Knowledge Distillation between DNN and SNN for Intelligent Sensing Systems on Loihi Chip

Shiya Liu[1] and Yang Yi[2]

[1]EMD Electronics, Tempe, AZ
[2]Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, VA
[1]shiya.liu@emdgroup.com, [2]yangyi8@vt.edu

2023 24th International Symposium on Quality Electronic Design (ISQED) | 979-8-3503-3475-3/23/$31.00 ©2023 IEEE | DOI: 10.1109/ISQED57927.2023.10129306

Abstract—Building accurate and efficient deep neural network (DNN) models for intelligent sensing systems to process data locally is essential. Spiking neural networks (SNNs) have gained significant popularity in recent years because they are more biological-plausible and energy-efficient than DNNs. However, SNNs usually have lower accuracy than DNNs. In this paper, we propose to use SNNs for image sensing applications. Moreover, we introduce the DNN-SNN knowledge distillation algorithm to reduce the accuracy gap between DNNs and SNNs. Our DNN-SNN knowledge distillation improves the accuracy of an SNN by transferring knowledge between a DNN and an SNN. To better transfer the knowledge, our algorithm creates two learning paths from a DNN to an SNN. One path is between the output layer and another path is between the intermediate layer. DNNs use real numbers to propagate information between neurons while SNNs use 1-bit spikes. To empower the communication between DNNs and SNNs, we utilize a decoder to decode spikes into real numbers. Also, our algorithm creates a learning path from an SNN to a DNN. This learning path better adapts the DNN to the SNN by allowing the DNN to learn the knowledge from the SNN. Our SNN models are deployed on Loihi, which is a specialized chip for SNN models. On the MNIST dataset, our SNN models trained by the DNN-SNN knowledge distillation achieve better accuracy than the SNN models on GPU trained by other training algorithms with much lower energy consumption per image.

Index Terms—spiking neural network, knowledge distillation, sensing system, deep neural network

I. Introduction

With the development of deep neural networks (DNNs), using DNNs on intelligent sensing systems for applications such as smart cities, smart factories, and autonomous vehicles has gained immense attention [1, 2, 3, 4, 5]. These applications heavily rely on cloud computing for data processing. However, cloud computing suffers from reliability, long latency, and weak privacy issues. On-device data processing is required for many time-critical applications. DNNs require large computation and storage resources [6, 7]. Therefore, it is inefficient to deploy a DNN model on resource-constrained devices. In recent years, spiking neural networks (SNNs) have demonstrated their feasibility in efficient hardware implementation [8, 9, 10]. Compared to DNNs, SNNs are more biologically plausible and exploit sparse and asynchronous discrete events for communication between neurons. Taking advantage of event-based computation, SNNs are capable to perform efficient inference on resource-constrained devices for intelligent sensing systems.

Loihi [8] is a neuromorphic computing chip for running SNNs. Loihi utilizes asynchronous SNNs to implement event-driven parallel computations for executing inference and learning with high efficiency. The innovative architecture of Loihi power future applications that need real-time processing and energy efficiency. In this paper, we focus on building intelligent sensing systems using SNNs on Loihi.

One of the challenges in SNNs is that it has lower accuracy compared to DNNs in image sensing applications [11]. One of the main reasons is that the mature training algorithms for DNNs such as gradient descent [12] cannot be adopted in SNNs because of the non-differentiable spike activities. STDP learning [13, 14] has been widely used for the training of SNNs because of its efficiency and simplicity. It updates the weight of one synaptic connection based on the relative timing of pre- and post-synaptic action potentials during a learning window. However, exploiting the STDP learning algorithm only is not sufficient to generate a high-performance SNN model. To address the training issue, several surrogate gradient descent learning algorithms have been introduced [15, 16]. In these research works, the authors first define an approximation function to approximate the spiking activities. Then, the back-propagation through time [17] algorithm is adopted to back-propagate gradients to both the spatial and temporal domain of an SNN model.

Even though a surrogate gradient descent training algorithm can improve the accuracy of an SNN, the accuracy gap still exist between an SNN and its DNN counterpart. To address this issue, we propose the DNN-SNN knowledge distillation algorithm to enhance the accuracy of an SNN. Conventional knowledge distillation algorithms are a model compression technique [18]. It lifts the accuracy of a smaller student model by learning the knowledge from a larger teacher model. There are three issues in conventional knowledge distillation algorithms. First of all, conventional knowledge distillation is designed for DNNs. How to effectively transfer knowledge from a DNN to an SNN remains an open question. Secondly, finding a suitable teacher model for a student model is challenging. Because of the mismatched capacity issue, a high-performance teacher model does not often yield a high-performance student model. Thirdly, a teacher model's parameters might be adaptable to the student model. Then, the knowledge in the teacher model would not be beneficial

for the student model.

To address the aforementioned issues of conventional knowledge distillation, we propose the DNN-SNN knowledge distillation algorithm to train a DNN and an SNN in parallel. An SNN is used as the student model and the DNN counterpart is utilized as the teacher model. This design diminishes the mismatched capacity issue between a DNN and an SNN. Our algorithm creates two knowledge transfer paths from a DNN to an SNN. One path is between the intermediate layer and another path is between the output layer. These two knowledge transfer paths improve the accuracy of the SNN model by allowing the SNN model to learn both output distribution and intermediate layer representation from the DNN model. DNNs use real numbers to convey information between neurons while SNNs use 1-bit spikes. To empower the communication between DNNs and SNNs, we utilize a decoder to decode spikes into real numbers. Also, we create a learning path from an SNN to a DNN. Using this path, the DNN can learn the knowledge from the SNN to better adapt itself to the SNN. Our contributions are summarized below.

- The DNN-SNN knowledge distillation is proposed. It enables knowledge distillation between DNNs and SNNs. Also, it diminishes the mismatched capacity issue between DNNs and SNNs and allows an SNN to learn both output distribution and intermediate representation from a DNN.
- The DNN-SNN knowledge distillation transfers knowledge from both the output and intermediate layer of a DNN to an SNN. A decoder is utilized to decode the spikes in an SNN into real numbers to empower the communication between a DNN and an SNN. Also, the knowledge from an SNN is transferred to a DNN to better adapts the DNN to the SNN.
- We have successfully deployed our SNN models on the Loihi. Compared to models on GPU trained by other surrogate gradient descent algorithms [19, 20, 21, 22, 23, 24], our SNN models on Loihi trained by the DNN-SNN knowledge distillation achieve better accuracy and energy efficiency for image classification tasks.

II. Background

A. Comparison of DNNs and SNNs

DNNs are a type of neural network inspired by the structure and function of the human brain. However, there are some fundamental differences between DNNs and the brain in terms of neural computations. DNNs use real numbers to propagate information between neurons while spike trains of action potentials are used to convey information between neurons in the brain [25]. Due to this essential difference, SNNs emerged. SNNs use 1-bit spikes to perform communications between neurons. Information is embedded in spike latency and rates. The differences between DNNs and SNNs are summarized in Fig. 1. DNNs transfer information using real numbers while SNNs utilize spikes. In SNNs, the spikes in a sequence are time-dependent. Therefore, SNNs can process temporal and spatial information at the same time.

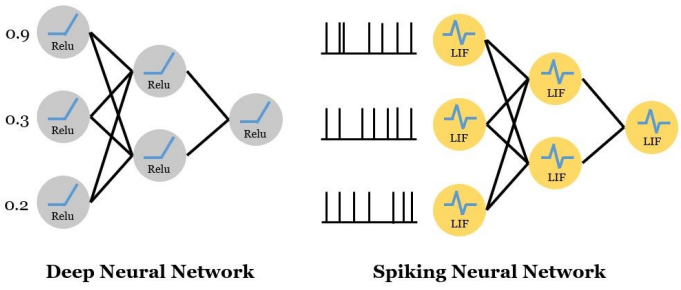

Fig. 1. Comparison of DNNs and SNNs

B. Knowledge Distillation

Knowledge distillation is introduced in [18]. It is a model compression technique by transferring knowledge from a larger teacher model to a smaller student model. In conventional knowledge distillation [18], a teacher model's knowledge is transferred to a student model through the soft target. The soft target is the prediction of each class yielded by the teacher model. In [18], the authors state that essential information is embedded in the ratios of very small probabilities between different classes. To strengthen the probability differences, a temperature parameter T is inserted into the softmax function. The modified softmax function is expressed as,

$$P(z_i; T) = \frac{\exp(z_i/T)}{\sum_{j=1}^{n} \exp(z_j/T)} \quad (1)$$

The student model is trained by two loss functions. The first loss function uses the ground truth labels from the dataset as the target. The second loss function utilizes the soft target from the teacher model as the target. The overall loss function for the student model is written as,

$$L(z_t, z_s, y) = (1 - \alpha)H(y, P(z_s; T = 1)) + \alpha H(P(z_t; T = k), P(z_s; T = k)), \quad (2)$$

where H represents the cross-entropy loss function. P is the modified softmax function with temperature parameter T. y is the ground truth target. α is the coefficient for the loss function. z_t and z_s are the logits of the teacher model and student model, respectively.

III. Intelligent Sensing Systems using Spiking Neural Networks on Loihi Chip

A. Applications of Intelligent Sensing Systems

Intelligent sensing systems is crucial for many applications such as smart factory, smart city, and autonomous systems [1, 2, 3, 4, 5]. These applications are summarized in Fig. 2. Incorporating advanced sensing systems into these applications can make the whole society more efficient and sustainable.

Many research works have applied SNNs in sensing systems for applications such as autonomous robots, keyword spotting, Visual-Tactile Sensing, and Hand-Gesture Recognition [26, 27, 28, 29]. These research works exploit SNNs to perform real-time and highly efficient data processing.

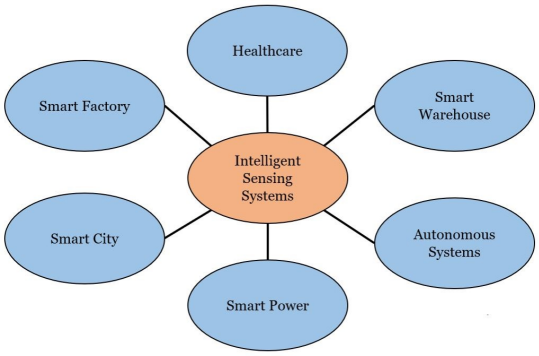

Fig. 2. Applications of intelligent sensing systems

B. Loihi Chip for Intelligent Sensing Systems

Loihi is a neuromorphic computing chip for SNNs from Intel Research Labs. Loihi exploits asynchronous SNNs to implement event-driven parallel computations to execute highly efficient inference. Loihi has several advantages over other neuromorphic computing chips such as Neurogrid [30] and BrainScaleS [31]. In the first place, using barrier synchronization, Loihi's cores in a mesh can execute independently and utilize barrier messages to perform global synchronization. Secondly, Loihi has up to 1 million neurons on each chip and the chip is fabricated with the Intel 4 process [32]. Thirdly, the behaviors of Loihi are more reliable and predictable than other neuromorphic computing chips such as Neurogrid [30] and BrainScaleS [31], which utilize mixed-signal or fully asynchronous digital circuit systems. Lastly, Loihi has better research community support and documentation. The software framework provided by Loihi is open, modular, and extensible.

Loihi chips have been used on many sensing systems [33, 34, 35]. In [33], the authors build a sensing system with Loihi to learn and recognize hazardous chemicals under significant noise conditions. The experimental results show that the algorithm achieves better accuracy than state-of-the-art algorithms such as DNNs. In [34], the authors create a high-performance gesture recognition system using Loihi and DVS camera, which is an event-based sensor. The system performs accurate gesture recognition with real-time processing. An event-based object tracking on the Loihi chip is introduced In [35]. Using an event-based DVS camera and Loihi chip, the system can execute accurate and fast object tracking when occlusions are present.

C. SNNs for Intelligent Sensing Systems on Loihi Chip

In this paper, we propose an energy-efficient SNN model for image sensing applications and deploy the SNN model on the Loihi chip. To reduce the accuracy gap between DNNs and SNNs, we introduce a knowledge distillation algorithm called DNN-SNN knowledge distillation. The algorithm improves the accuracy of an SNN by learning the knowledge from a DNN. Our experimental results show that the SNN models on Loihi trained by the DNN-SNN knowledge distillation achieve better results than other SNN training algorithms [15, 19, 21, 22, 23, 24] on image classification tasks with better energy efficiency.

IV. Spiking Neural Networks with DNN-SNN Knowledge Distillation

DNNs have gained significant attention nowadays due to their great success in many areas such as computer vision [36], nature language processing [37], and speech recognition [38]. With the development of DNNs, using DNNs in intelligent sensing systems for applications such as smart cities, smart factories, and autonomous vehicles is becoming more and more popular [39, 40]. Due to the large computation and storage resources required by DNNs, these emerging applications count on cloud computing for data processing. It is not practical for many applications because cloud computing has several issues such as long latency, reliability, and privacy. Many research works have been proposed to reduce the computation and storage resources of a DNN on resource-constrained devices [6, 7, 18, 41]. In recent years, SNNs have demonstrated their feasibility in efficient hardware implementation [8]. Compared to DNNs, SNNs are more biologically plausible and energy-efficient on neuromorphic computing chips such as Loihi. It is because SNNs exploit sparse and asynchronous discrete events for communication between neurons [25].

However, the accuracy of an SNN is usually worse than a DNN in image sensing applications because the training of SNNs is challenging. It is because neurons in an SNN use 1-bit spikes to communicate with other neurons. These spikes are sparse and non-differentiable. Therefore, The popular training algorithms such as gradient descent [12] cannot be used in SNNs. To address this issue, surrogate gradient descent algorithms have been proposed [15, 16]. In these works, the authors first define an approximation function to approximate the spiking activities. Next, the back-propagation through time [17] algorithm is exploited to propagate gradients to both the spatial and temporal domain of an SNN model.

Surrogate gradient descent algorithms can improve the accuracy of an SNN significantly. However, there is still a gap between the accuracy of a DNN and an SNN. To decrease the gap, a novel knowledge distillation algorithm called DNN-SNN knowledge distillation is proposed in this paper. Conventional knowledge distillation is a model compression technique by transferring knowledge from a larger teacher model to a smaller student model. By learning the knowledge from the teacher model, the student model's performance can be improved. Conventional knowledge distillation has three issues. Firstly, conventional knowledge distillation is introduced for DNNs and how to effectively use it between DNNs and SNNs remains an open question. Then, it is challenging to pick a teacher model for a student model because of the mismatched capacity between them. Many research works [42, 43] conclude that a high-performance teacher model often does not produce a good student model. Next, a teacher model's parameters might not be adaptable to a student model. The

knowledge from the teacher model would not be useful for the student model.

To address these issues of conventional knowledge distillation, an SNN is used as the student model and the DNN counterpart is exploited as the teacher model. This design reduces the mismatched capacity issue between a DNN and an SNN. The DNN and the SNN model are trained in parallel. During the training phase, there is two knowledge-transferring path from a DNN to an SNN. One path is from the intermediate layer and another path is from the output layer. Through these two learning paths, the SNN model can learn both output distribution and intermediate layer representation from the DNN model. DNNs use real numbers to propagate information between neurons while SNNs use 1-bit spikes. To enable the communication between DNNs and SNNs, we utilize a decoder to decode spikes into real numbers. Also, we create a learning path from an SNN to a DNN. Through this learning path, the DNN can learn the knowledge from the SNN to better adapt itself to the SNN.

A. DNN-SNN Knowledge Distillation

The training procedure of the DNN-SNN knowledge distillation is shown in Fig. 3. In DNN-SNN knowledge distillation,

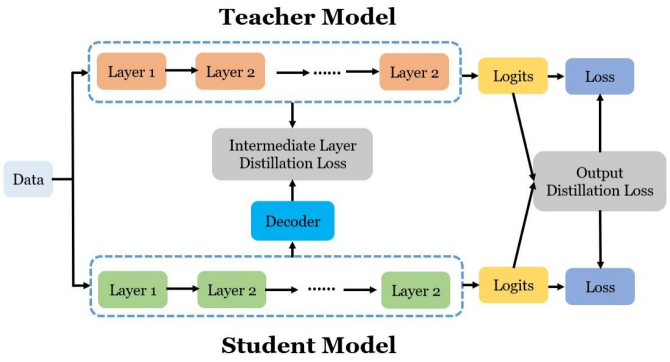

Fig. 3. The overall structure of DNN-SNN knowledge distillation

an SNN is used as the student model and the DNN counterpart is utilized as the teacher model. This setup helps to diminish the gap in model capacity between the student and teacher model. There are three loss functions used to train the student model during training. The first loss function is the output distillation loss function. It uses the predicted values from the teacher model as the target. The second loss function is the output loss function and it uses the ground truth labels as the target values. The last loss function is the intermediate layer distillation loss function, which utilizes the output of intermediate layers from the teacher model as the target values. Compared to conventional knowledge distillation, the teacher model's knowledge is transferred from both the output and intermediate layers to the student model. These two knowledge transfer paths improve the accuracy performance of the student model by allowing the student model to learn both output distribution and intermediate layer representation from the teacher model. We use a decoder to decode the spikes from the intermediate layers of the student model to real numbers to communicate with the teacher model. The details of the decoder will be discussed in later subsections.

The teacher model is trained by two loss functions during training. The first loss function is the output distillation loss function, which uses the predicted values of the student model as the target values. The second loss function is the output loss function and it utilizes the ground truth labels as the target values. Our algorithm transfers the knowledge from the student model to the teacher model because the knowledge helps the teacher model better fit the student model. Then, the teacher model can provide more useful information for the student model.

B. Decoder for DNN and SNN Communication

Our DNN-SNN knowledge distillation transfers the knowledge from both the intermediate layer and output layer of a DNN to an SNN. Due to the different representations of information, DNNs cannot communicate with SNNs. DNNs use real numbers to represent information while SNNs use a sequence of 1-bit spikes to convey information between neurons.

To enable communication, we create a decoder to decode the output of intermediate layers of an SNN to real numbers. Assume there are N_s neurons with T time steps in an intermediate layer of an SNN. An intermediate layer of a DNN has N_d neurons. The decoder is a 2D matrix with a shape of $(T \times N_s) \times N_d$. The values in the matrix are generated from a uniform distribution over the interval of -0.5 and 0.5. The decoder is not trained and fixed during training. During training, the output of the SNN's intermediate layer is flattened and multiplied by the decoder matrix. Then, the decoder's output is fed into the loss function. The loss function uses the output of the intermediate layer of a DNN as the target value.

C. DNN-SNN Knowledge Distillation Loss Function

As shown in Fig. 3, there are three loss functions to train the student model. The first loss function is the output loss function and it is an L1 loss function, which is written as,

$$L_{SO} = H(y, P_s), \tag{3}$$

where y is the ground truth label from the dataset and P_s is the output of the student model. H is a cross-entropy loss function.

The second loss function is the output distillation loss function, which is a Kullback-Leibler divergence loss function [44]. The loss function utilizes the output probability distribution of the teacher model as the target. The output distillation loss function is shown as,

$$L_{SKL}(P_t^\eta, P_s^\eta) = \sum_{c=1}^{M} P_t^\eta \log\left(\frac{P_t^\eta}{P_s^\eta}\right), \tag{4}$$

where P_s^η and P_t^η represent the predicted probability of each class after the modified softmax function defined in Eq. 1 for

the student and teacher model, respectively. M is the number of classes in the dataset.

The intermediate layer distillation loss function utilizes the output of the teacher model's intermediate layer as the target value. The loss function can be written as,

$$L_{SI}(I_t, I_s) = \|I_t - I_s\|_1, \tag{5}$$

where I_t represents the intermediate layer of the teacher model. I_s is the output of the intermediate layer of the student model decoded by the decoder.

The overall loss function for the student model is shown as,

$$L_s = L_{SO} + \alpha_s L_{S_{KL}} + \beta_s L_{SI} \tag{6}$$

where α_s and β_s are the coefficient for loss function $L_{S_{KL}}$ and L_{SI}. These two parameters are hyperparameters to control the strength of distillation loss functions.

There are two loss functions to train the teacher model, which are the output loss function and output distillation loss function, respectively. The output loss function for the teacher model is illustrated as,

$$L_{TO} = H(y, P_t), \tag{7}$$

where P_t is the final output of the teacher model.

The output distillation loss function for the teacher model utilizes the output probability distribution of the student model as the target. The output distillation loss function for the teacher model is a Kullback-Leibler divergence loss function and it is expressed as,

$$L_{T_{KL}}(P_s^\eta, P_t^\eta) = \sum_{c=1}^{M} P_s^\eta \log\left(\frac{P_s^\eta}{P_t^\eta}\right), \tag{8}$$

where P_s^η and P_t^η represent the predicted probability of each class after the modified softmax function defined in Eq. 1 for the student and teacher model, respectively. M is the number of classes in the dataset.

The overall loss function for the teacher model is,

$$L_t = L_{TO} + \alpha_t L_{T_{KL}} \tag{9}$$

where α_t is the coefficient for the output distillation loss function.

V. Experimental Results

A. Experimental Setup

In our experiments, the SNN model consists of two fully connected layers with 96 neurons in each layer. On MNIST classification task [45], the input and output sizes are 784 and 10, respectively. The input and output sizes are 3024 and 10 respectively on the SVHN classification task. The SNN model runs on a single Loihi chip. On Loihi, we use a Poisson encoder to encode input data. The voltage and current decay time constant of the LIF neuron are 10e-3 and 5e-3, respectively. all GPU experiments run on a single NVIDIA GeForce RTX 2080. In terms of the measurement of power consumption, the power consumption on GPU is measured using the NVIDIA GPU management and monitoring tool. The power on Loihi is measured using the internal tools provided by Loihi. In our experiments, we use the following abbreviations to represent different models and algorithms.

- "BaseSNN" is the fully-connected layers trained by the surrogate gradient-descent algorithm proposed in [15]. The model has two fully connected layers with 96 neurons in each layer. The model performs inference on the Loihi chip and is trained on GPU.
- "BaseSNN+DSKD" represents the fully-connected layers trained by the proposed DNN-SNN knowledge distillation training algorithm. The model has the same architecture as the BaseSNN model. The model performs inference on the Loihi chip and is trained on GPU.
- "eWB" represents the fully-connected layers trained by the training algorithm proposed in [19]. The model has the same architecture as the BaseSNN model. The model runs on GPU.
- "BindsNet" is the fully-connected layers trained by the training algorithm proposed in [21]. The model has the same architecture as the BaseSNN model. The model runs on GPU.
- "Q-SpiNN" represents the fully-connected layers trained by the training algorithm proposed in [23]. The model has the same architecture as the BaseSNN model. The model runs on GPU.
- "QSTDP" is the fully-connected layers trained by the training algorithm proposed in [24]. The model has the same architecture as the BaseSNN model. The model runs on GPU.

B. Training Setup

On both MNIST and SVHN datasets, we use a mini-batch size of 512 to train SNN models. The Adam learning algorithm is used and the learning rate is 0.001. We multiply the learning rate by 0.1 every 30 epochs. The training epoch is 100. Each spiking neuron has 100 time steps. The Hyperparameters such as α_s, β_s, and α_t in Eq. 6 and 9 are set to 0.10, 0.05, and 0.05.

C. DNN-SNN Knowledge Distillation Architecture

The architecture of our DNN-SNN knowledge distillation algorithm is shown in Fig. 3. The teacher model is a DNN model and the student model is an SNN model. Both the DNN and SNN models have two layers with the same number of neurons. This setup reduces the mismatched capacity issue between the DNN and the SNN. There is an intermediate layer transfer path from the DNN model to the SNN model. A decoder is used to decode the spikes in the intermediate layers of the SNN to real numbers. There is a bi-directional knowledge transfer path between the DNN and the SNN. The bi-directional path is used to transfer knowledge at the output layer between each other. Both the DNN and the SNN model have another output loss function, which uses the ground truth labels as the target values.

D. MNIST HandWritten Digits Classification

A commonly used dataset for evaluating SNN models is the MNIST dataset [45]. There are 60,000 training images and 10,000 testing images in the dataset. The image is in grayscale with a dimension of 28 × 28. Each image is a handwritten digit from 0 to 9. The accuracy and energy consumption per image comparison of different models are shown in Table I. We set the batch size on GPU to 1 because Loihi only supports a batch size of 1. As can be seen from Table I, our BaseSNN+DSKD model trained by the proposed DNN-SNN knowledge distillation algorithm achieves better accuracy than other models using other training algorithms [15, 19, 21, 22, 23, 24]. Also, our BaseSNN+DSKD model on Loihi has better energy efficiency than other models on GPU. The energy consumption per image of the BaseSNN+DSKD model on Loihi is 432X, 1193X, 147X, and 169X less than the eWB, BindsNet, Q-SpiNN, and QSTDP models on GPU with a batch size of 1.

TABLE I
ACCURACY AND ENERGY PER IMAGE COMPARISON ON MNIST DATASET BETWEEN DIFFERENT MODELS WITH BATCH SIZE = 1

Network	Energy/Image(J)	Accuracy(%)
eWB (GPU) [19]	5.612	95.35
BindsNet (GPU) [21]	15.506	95.00
Q-SpiNN (GPU) [23]	1.909	95.14
QSTDP (GPU) [24]	2.201	94.58
BaseSNN (Loihi)	0.013	95.06
BaseSNN+DSKD (Loihi)	0.013	96.87

Measuring GPU latency with a batch size of 1 is significantly slower than with a larger batch size. To better demonstrate the energy efficiency of the Loihi, we conduct another experiment and use a batch size of 128 for all models on GPU. Meanwhile, we keep the batch size of the model on Loihi at 1.

the energy consumption per image (J) in log scale with a batch size of 128 during the inference phase is shown in Fig. 4. As demonstrated in Fig. 4, the energy consumption per image of the BaseSNN+DSKD model on Loihi is 5.61X, 290.15X, 3.61X, and 4.69X less than the eWB, BindsNet, Q-SpiNN, and QSTDP models on GPU.

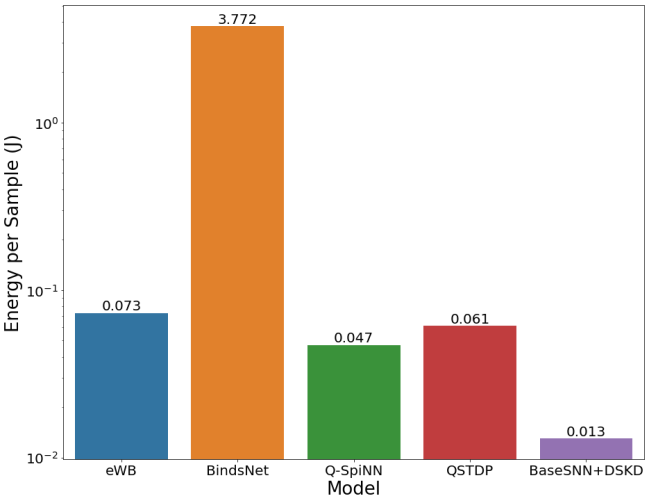

Fig. 4. Energy consumption per image (J) of different implementations in log scale with batch size 128

E. SVHN Dataset

SVHN is a real-world image dataset for digits classification. It has over 70000 digits for training and 20000 digits for testing. The image has a dimension of $32 \times 32 \times 3$. The target of the dataset is to classify digits from 0 to 9. The accuracy comparison of the BaseSNN and the BaseSNN+DSKD model is shown in Table II. As can be seen from the table, the BaseSNN+DSKD model improves the accuracy by approximately 2.6% compared to the BaseSNN model.

TABLE II
ACCURACY COMPARISON ON SVHN DATASET BETWEEN PTQ SNN AND QAT SNN MODEL

Network	Normalized Accuracy
BaseSNN	1.000X
BaseSNN+DSKD	1.026X

VI. CONCLUSION

In this paper, we propose to use SNNs for energy-efficient intelligent sensing systems on the Loihi chip. we introduce a knowledge distillation algorithm called DNN-SNN knowledge distillation to improve the accuracy of an SNN model by transferring knowledge between a DNN and an SNN. Our DNN-SNN knowledge distillation algorithm train a DNN and an SNN in parallel. During training, there are two knowledge transfer paths from the DNN to the SNN. One path is at the output layer and another path is between the intermediate layers. These two learning paths allow the SNN to learn both output distribution and intermediate layer representation from the DNN. To enable communication between the DNN and the SNN, we build a decoder to decode the spikes in the intermediate layers of the SNN to real numbers. Moreover, there is a knowledge transfer path at the output layer from the SNN to the DNN. This learning path helps the DNN to better adapt itself to the SNN. Then, more helpful guidance can be provided by the DNN. On the MNIST and SVHN datasets, our experimental results demonstrate that the DNN-SNN knowledge distillation achieves better accuracy than other training algorithms such as eWB, BindsNet, Q-SpiNN, and QSTDP. Meanwhile, our models on Loihi reduce energy consumption per image significantly compared to other models on GPUs. These results show that energy-efficient intelligent sensing systems can be achieved using SNNs on Loihi.

REFERENCES

[1] Anna Challoner and Gheorghe H Popescu. “Intelligent sensing technology, smart healthcare services, and internet of medical things-based diagnosis”. In: American Journal of Medical Research 6.1 (2019), pp. 13–18.

[2] Yaqiong Duan et al. “Smart city oriented Ecological Sensitivity Assessment and Service Value Computing based on Intelligent sensing data processing”. In: Computer Communications 160 (2020), pp. 263–273.
[3] Tahera Kalsoom et al. “Advances in sensor technologies in the era of smart factory and industry 4.0”. In: Sensors 20.23 (2020), p. 6783.
[4] Linga Reddy Cenkeramaddi et al. “A survey on sensors for autonomous systems”. In: 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA). IEEE. 2020, pp. 1182–1187.
[5] Qiang Fan et al. “Delay-aware resource allocation in fog-assisted IoT networks through reinforcement learning”. In: IEEE Internet of Things Journal 9.7 (2021), pp. 5189–5199.
[6] Benoit Jacob et al. “Quantization and training of neural networks for efficient integer-arithmetic-only inference”. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, pp. 2704–2713.
[7] Shiya Liu et al. “Accurate and efficient quantized reservoir computing system”. In: 2020 21st International Symposium on Quality Electronic Design (ISQED). IEEE. 2020, pp. 364–369.
[8] Mike Davies et al. “Loihi: A neuromorphic manycore processor with on-chip learning”. In: Ieee Micro 38.1 (2018), pp. 82–99.
[9] Kangjun Bai et al. “Enabling a new era of brain-inspired computing: energy-efficient spiking neural network with ring topology”. In: Proceedings of the 55th Annual Design Automation Conference. 2018, pp. 1–6.
[10] Chenyuan Zhao et al. “Energy efficient temporal spatial information processing circuits based on STDP and spike iteration”. In: IEEE Transactions on Circuits and Systems II: Express Briefs 67.10 (2019), pp. 1715–1719.
[11] Lei Deng et al. “Rethinking the performance comparison between SNNS and ANNS”. In: Neural networks 121 (2020), pp. 294–307.
[12] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: nature 323.6088 (1986), pp. 533–536.
[13] Natalia Caporale and Yang Dan. “Spike timing–dependent plasticity: a Hebbian learning rule”. In: Annu. Rev. Neurosci. 31 (2008), pp. 25–46.
[14] Chenyuan Zhao et al. “Spike-time-dependent encoding for neuromorphic processors”. In: ACM Journal on Emerging Technologies in Computing Systems (JETC) 12.3 (2015), pp. 1–21.
[15] Emre O Neftci, Hesham Mostafa, and Friedemann Zenke. “Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks”. In: IEEE Signal Processing Magazine 36.6 (2019), pp. 51–63.
[16] Yujie Wu et al. “Spatio-temporal backpropagation for training high-performance spiking neural networks”. In: Frontiers in neuroscience 12 (2018), p. 331.
[17] Paul J Werbos. “Backpropagation through time: what it does and how to do it”. In: Proceedings of the IEEE 78.10 (1990), pp. 1550–1560.
[18] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. “Distilling the knowledge in a neural network”. In: arXiv preprint arXiv:1503.02531 2.7 (2015).
[19] Dohun Kim et al. “eWB: Event-Based Weight Binarization Algorithm for Spiking Neural Networks”. In: IEEE Access 9 (2021), pp. 38097–38106.
[20] Peter U Diehl and Matthew Cook. “Unsupervised learning of digit recognition using spike-timing-dependent plasticity”. In: Frontiers in computational neuroscience 9 (2015), p. 99.
[21] Hananel Hazan et al. “Bindsnet: A machine learning-oriented spiking neural networks library in python”. In: Frontiers in neuroinformatics 12 (2018), p. 89.
[22] Christian Pehle and Jens Egholm Pedersen. “Norse-A deep learning library for spiking neural networks”. In: Version 0.0 6 (2021).
[23] Rachmad Vidya Wicaksana Putra and Muhammad Shafique. “Q-spinn: A framework for quantizing spiking neural networks”. In: 2021 International Joint Conference on Neural Networks (IJCNN). IEEE. 2021, pp. 1–8.
[24] SG Hu et al. “Quantized STDP-based online-learning spiking neural network”. In: Neural Computing and Applications (2021), pp. 1–16.
[25] Amirhossein Tavanaei et al. “Deep learning in spiking neural networks”. In: Neural networks 111 (2019), pp. 47–63.
[26] Guangzhi Tang, Arpit Shah, and Konstantinos P Michmizos. “Spiking neural network on neuromorphic hardware for energy-efficient unidimensional slam”. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE. 2019, pp. 4176–4181.
[27] Yexin Yan et al. “Comparing Loihi with a SpiNNaker 2 Prototype on Low-Latency Keyword Spotting and Adaptive Robotic Control”. In: Neuromorphic Computing and Engineering (2021).
[28] Tasbolat Taunyazov et al. “Event-driven visual-tactile sensing and learning for robots”. In: arXiv preprint arXiv:2009.07083 (2020).
[29] Enea Ceolini et al. “Hand-gesture recognition based on EMG and event-based camera sensor fusion: A benchmark in neuromorphic computing”. In: Frontiers in Neuroscience 14 (2020), p. 637.
[30] Dion Khodagholy et al. “NeuroGrid: recording action potentials from the surface of the brain”. In: Nature neuroscience 18.2 (2015), pp. 310–315.
[31] Chetan Singh Thakur et al. “Large-scale neuromorphic spiking array processors: A quest to mimic the brain”. In: Frontiers in neuroscience (2018), p. 891.

[32] B Sell et al. “Intel 4 CMOS Technology Featuring Advanced FinFET Transistors optimized for High Density and High-Performance Computing”. In: 2022 IEEE Symposium on VLSI Technology and Circuits (VLSI Technology and Circuits). IEEE. 2022, pp. 282–283.
[33] Nabil Imam and Thomas A Cleland. “Rapid online learning and robust recall in a neuromorphic olfactory circuit”. In: Nature Machine Intelligence 2.3 (2020), pp. 181–191.
[34] Riccardo Massa et al. “An efficient spiking neural network for recognizing gestures with a dvs camera on the loihi neuromorphic processor”. In: 2020 International Joint Conference on Neural Networks (IJCNN). IEEE. 2020, pp. 1–9.
[35] Matthew Evanusa, Yulia Sandamirskaya, et al. “Event-based attention and tracking on neuromorphic hardware”. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2019, pp. 0–0.
[36] Athanasios Voulodimos et al. “Deep learning for computer vision: A brief review”. In: Computational intelligence and neuroscience 2018 (2018).
[37] Tom Young et al. “Recent trends in deep learning based natural language processing”. In: ieee Computational intelligence magazine 13.3 (2018), pp. 55–75.
[38] Ali Bou Nassif et al. “Speech recognition using deep neural networks: A systematic review”. In: IEEE access 7 (2019), pp. 19143–19165.
[39] Elena Esposito et al. “Dynamic neural network architectures for on field stochastic calibration of indicative low cost air quality sensing systems”. In: Sensors and Actuators B: Chemical 231 (2016), pp. 701–713.
[40] Lucas Prado Osco et al. “A review on deep learning in UAV remote sensing”. In: International Journal of Applied Earth Observation and Geoinformation 102 (2021), p. 102456.
[41] Shiya Liu et al. “Efficient neural networks for edge devices”. In: Computers & Electrical Engineering 92 (2021), p. 107121.
[42] Jang Hyun Cho and Bharath Hariharan. “On the efficacy of knowledge distillation”. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, pp. 4794–4802.
[43] Seyed Iman Mirzadeh et al. “Improved knowledge distillation via teacher assistant”. In: Proceedings of the AAAI conference on artificial intelligence. Vol. 34. 04. 2020, pp. 5191–5198.
[44] Shinto Eguchi and John Copas. “Interpreting kullback–leibler divergence with the neyman–pearson lemma”. In: Journal of Multivariate Analysis 97.9 (2006), pp. 2034–2040.
[45] Yann LeCun. “The MNIST database of handwritten digits”. In: http://yann. lecun. com/exdb/mnist/ (1998).