Knowledge Distillation at the Edge: Lightweight Deep Neural Networks Deployed on Neuromorphic Hardware

Rashedul Islam[1], Shahanur Alam[1], Chris Yakopcic[1], Nayim Rahman[1], Simon Khan[2], and Tarek M. Taha[1]
Dept. Of Electrical and Computer Engineering, *University of Dayton*, Dayton, OH, USA
Air Force Research Laboratory, Rome, NY, USA
{islamm24, alamm8, cyakopcic1, rahmanm12}@udayton.edu, simon.khan@us.af.mil, tarek.taha@udayton.edu

***Abstract*—Neuromorphic processors are highly suitable for edge intelligence due to their ultra-low power consumption and event-driven computation. However, deploying lightweight neural networks on such hardware often results in accuracy degradation caused by model compression, quantization, and ANN-to-SNN conversion. Improving accuracy while maintaining low energy and computational cost therefore remains a key challenge. In this paper, we investigate the impact of knowledge distillation for improving lightweight model performance on the Brainchip Akida. To the best of our knowledge, this is the first neuromorphic hardware-based study that systematically evaluates knowledge distillation. Two distillation strategies are explored: (1) knowledge distillation before quantization and (2) knowledge distillation applied after quantization. Experimental results show that knowledge distillation significantly mitigates quantization-induced accuracy loss, with post-quantization distillation consistently outperforming pre-quantization distillation. Even with extreme model compression (95% parameter reduction compared to the teacher models), the student networks achieved up to 11.45% accuracy improvement over the baseline, while preserving performance after ANN-to-SNN conversion. The results show that knowledge distillation is effective for both complex and lightweight models, enabling compact networks to achieve higher accuracy with minimal energy overhead. Thus, the proposed method is a key enabler for accurate and energy-efficient deployment of neuromorphic processing at the edge.**

Keywords*—*Lightweight, Neuromorphic, Knowledge Distillation, Quantization, Low power, SWaP, Akida.

I. Introduction

In recent years, deep learning has rapidly evolved and become an essential component of many real-world applications, including computer vision [1], healthcare [2], fraud detection [3], virtual assistants [4], natural language processing [5], and autonomous vehicles [6]. These applications continuously generate large volumes of data, and deep learning models rely heavily on such data to learn complex representations and achieve high accuracy. As model performance typically improves with increasing data size and network complexity, modern deep learning systems demand substantial computational resources for both training and deployment.

To meet these computational requirements, deep learning workloads are commonly accelerated (through GPUs and TPUs). However, these platforms are inherently power intensive, which limits their applicability in resource-constrained environments. For instance, an NVIDIA Tesla K80 GPU consumes an average of 60–70 W, while an Intel® Xeon® processor consumes approximately 22 W across different deep learning tasks [7]. Such power-hungry characteristics are unsuitable for battery-operated and edge devices, where strict constraints on energy consumption, memory capacity, and real-time latency must be satisfied. As data volume and model complexity continue to grow, reducing power consumption while maintaining high accuracy will become an even more critical research challenge. Alternatively, lightweight deep neural network models with fewer parameters have gained significant attention, as they are easier to deploy on edge and embedded devices. However, reducing model size typically leads to degraded performance, creating a trade-off between efficiency and accuracy.

Neuromorphic hardware offers a promising solution by drawing inspiration from the structure and functioning of the human brain. These systems employ biologically inspired spiking neural networks (SNNs) that operate in an event-driven and massively parallel manner. By integrating memory and computation and exploiting sparse spike-based activity, neuromorphic processors can achieve significantly improved energy efficiency compared to conventional artificial neural networks. However, neuromorphic platforms impose strict constraints on neuron counts, synaptic resources, and numerical precision, making the direct deployment of large, high-capacity models infeasible. As a result, compact and lightweight models are essential for efficient mapping onto neuromorphic hardware.

Knowledge distillation (KD) has emerged as an effective technique to mitigate this performance loss by transferring knowledge from a large, high-capacity teacher model to a compact student model. Through this process, the student model can achieve performance closer to that of the teacher while maintaining a significantly smaller parameter footprint. KD has been extensively studied for artificial neural networks and, more recently, for spiking neural networks in software-based environments. However, to the best of our knowledge no one has yet implemented KD on SNN (or neuromorphic) hardware.

In this paper, we investigate the impact of knowledge distillation on neuromorphic hardware. Specifically, we explore two KD strategies: conventional response-based knowledge distillation before quantization and knowledge distillation applied after quantization. The resulting lightweight models are deployed on a neuromorphic platform, BrainChip's Akida processor. Performance is evaluated under the different distillation configurations. Experimental results demonstrate that response-based knowledge distillation improves model accuracy when successfully deployed on neuromorphic hardware. Applying KD after quantization yields superior performance compared to pre-quantization distillation. To the best of our knowledge, this work represents the first comprehensive study of knowledge distillation deployed neuromorphic hardware.

The remainder of this paper is organized as follows. Section II reviews related work on knowledge distillation and Section III presents a description of the hardware and algorithm used in our work. Section IV describes the experimental setup, while Section V discusses the experimental results. Finally, Section VI concludes the paper.

II. Related Work

In recent years, deep learning models have demonstrated strong performance across a wide range of applications. However, their increasing depth and large number of parameters make them computationally expensive and often impractical for deployment on resource-constrained hardware platforms. Knowledge distillation (KD), particularly response-based distillation, has emerged as an effective solution to address this challenge by transferring knowledge from a large teacher model to a compact student model. Beyond model compression, KD can also improve the generalization performance of student networks, enabling efficient and accurate deployment on hardware-limited systems.

An et al. [8] proposed Dynamic Weighted Knowledge Distillation (DWKD) for 3D brain tumor segmentation. The method dynamically adjusts distillation weights based on the student model's prediction uncertainty rather than using static weights. By combining a dynamic weighted KL-divergence loss with a regularized cross entropy loss the approach achieves higher Dice scores on the BraTS 2019-2021MRI datasets while maintaining the lightweight student model. Ho et al. [9] explored the use of knowledge distillation for efficient classification of chest X-ray abnormalities. The authors evaluated multiple knowledge distillation-based strategies. In addition, they used self-training and further improved their performance by reducing computational complexities. Cho et al. [10] investigated how the performance of student models can be improved through KD. They showed that the success of KD strongly depends on factors such as teacher accuracy, capacity gap between teacher and student and dataset complexity. Through experiments on image classification benchmarks, the study demonstrated that distillation is most effective when the teacher provides complementary information beyond hard labels. Sun et al. [11] proposed a Z-score based logit standardization that normalizes logits to zero mean and unit variance, allowing the student to focus on relative logit relationships rather than magnitudes. The method improves performance on CIFAR-100 and ImageNet, and is compatible with existing logits-based KD approaches, especially when distilling from a large teacher model.

Xu et al. [12] proposed a knowledge distillation-based framework to construct deep spiking neural networks (SNNs) from pretrain artificial neural networks (ANNs). They used an ANN-SNN joint training strategy where a high-capacity ANN teacher transfers knowledge to a SNN student through both response-based and feature-based distillation. Experiments on MNIST and CIFAR-10 including noisy variants demonstrate improved accuracy with faster convergence. Kushawaha et al. [13] transfer knowledge by distilling temporal spike activation patterns rather than conventional softmax probabilities. They proposed novel spike-based distillation loss functions, including full and sliding window L1, L2 and KL divergence losses and further enhance their performance using a multistage distillation strategy.

KD has demonstrated impressive performance improvements for both artificial neural networks and spiking neural networks across a wide range of tasks. However, due to constraints of current neuromorphic hardware, not all training strategies developed in simulation environments can be directly implemented in hardware. In particular, approaches that rely on direct SNN training, such as joint ANN–SNN training methods, are not supported on platforms like the BrainChip Akida, where SNNs are obtained through ANN-to-SNN conversion. To the best of our knowledge, no prior work has investigated KD-based deployment on neuromorphic processors or SNN hardware. This work addresses this gap by presenting the first study of knowledge distillation on neuromorphic hardware using the Brainchip Akida.

III. Hardware and Algorithm Description

A. Spiking Neural Network (SNN)

Biological neurons communicate by transmitting electrical signals known as spikes or action potentials along. Inspired by this biological mechanism, Spiking Neural Networks (SNNs) have emerged as a neural computing paradigm that more closely models the temporal dynamics and event-driven behavior of the human brain compared to conventional artificial neural networks (ANNs). Unlike ANNs, which operate on continuous-valued activations, SNNs process information using discrete spikes over time, providing increased biological plausibility and potential benefits in terms of energy efficiency and real-time processing [14].

Supervised training of SNNs can be broadly categorized into two main approaches: (1) direct training and (2) indirect

training through ANN-to-SNN conversion. In direct training, input data are first encoded into spike trains using encoding schemes (such as rate or temporal coding). The network is then trained using Backpropagation Through Time (BPTT), where surrogate gradient methods are employed to address the non-differentiable nature of the spiking function. During training, the error is computed by comparing the network output such as spike counts, firing rates, or membrane potentials with the target labels. This error is subsequently backpropagated through time to update the synaptic weights. Although direct training enables precise temporal learning, it is computationally intensive and challenging due to the complex dynamics of spiking neurons.

In contrast, indirect training first trains a conventional ANN using standard gradient-based optimization techniques. After training, the ANN is converted into an SNN by replacing analog neurons, such as ReLU units, with spiking neuron models while appropriately mapping and scaling the learned weights and neuron thresholds. This approach benefits from the stable and efficient training of ANNs and enables easier deployment of SNNs, particularly on neuromorphic hardware platforms. Figure 1 illustrates the structural components and functional behavior of a spiking neuron [15].

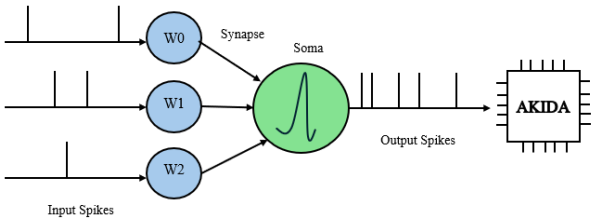

Fig. 1. Iillustration of spiking neural network inference as performed on the Akida SNN processor.

B. *Neuromorphic Processors*

Neuromorphic processors are specialized computing chips inspired by the structure and functioning of the human brain and are designed to execute spiking neural networks using hardware implementations of neurons and synapses. Unlike traditional von Neumann architecture, neuromorphic processors tightly integrate memory and computation within artificial synapses, thereby reducing data movement and significantly lowering power consumption compared to conventional CPUs and GPUs [16]. These processors operate in an event-driven manner and support massive parallelism, allowing multiple neural operations to be performed simultaneously. Due to their low power consumption and fast, real-time processing capabilities, neuromorphic processors have attracted considerable interest in applications such as pattern recognition, robotics, object detection, and sensory signal processing. In recent years, several companies and research laboratories have actively developed neuromorphic hardware; however, only a limited number of platforms have reached maturity. Notable examples include IBM's TrueNorth, Intel's Loihi, and BrainChip's Akida. We utilize the Akida in this work.

C. *Knowledge Distillation Procedure*

Although deep learning has achieved remarkable success across a wide range of applications, large-scale models are often impractical for deployment on edge devices due to hardware constraints, computational complexity, and energy consumption. Lightweight models often suffer from accuracy degradation. However, knowledge distillation (KD) addresses this challenge by enabling compact models to achieve competitive performance while significantly reducing model size and energy requirements.

Knowledge distillation is a machine learning technique in which knowledge is transferred from a large, high-capacity model, referred to as the *teacher*, to a smaller and lightweight model, known as the *student*. The teacher model is first trained to capture rich and informative representations from data. The soft output probabilities produced by the teacher contain additional information about class relationships beyond hard labels [17]. During student training, this knowledge is transferred by encouraging the student to mimic the teacher's softened output distribution, thereby reducing the performance gap caused by model compression. Figure 2 shows the illustration of knowledge distillation training.

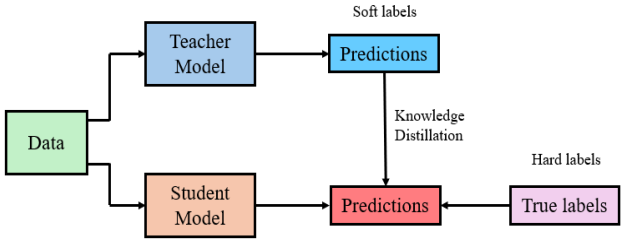

Fig. 2. The illustration of knowledge distillation training

During training, the total loss function is defined as a weighted combination of the student loss and distillation loss, given by

$$L_T = (1-\alpha)\, L_{Student} + \alpha\, L_{Distillation} \quad (1)$$

where L_T denotes the total loss, $L_{Student}$ is the standard supervised loss of student model, $L_{Distillation}$ represents the distillation loss, and α ($0 < \alpha < 1$) is hyperparameter that balances the contribution of the two loss terms.

The student loss is computed using the cross-entropy loss between the ground truth label vector y and the student predictions

$$L_{Student} = CE\,(y, \mathrm{softmax}(z_s)) \quad (2)$$

where CE denotes cross entropy and z_s denotes the student logits. The distillation loss is defined using the Kullback-Leibler (KL) divergence between the softened output distributions of the teacher and student

$$L_{Distillation} = T^2\, KL\,(P_t, P_s) \quad (3)$$

where, $T \in (0, +\infty)$ is a temperature parameter that controls the smoothness of the output distributions. The soft targets for the teacher and student are given by

$$P_t = \mathrm{softmax}\,(z_t/T),\ P_s = \mathrm{softmax}\,(z_s/T) \quad (4)$$

with, z_t denoting the teacher logits. By incorporating both hard labels and soft targets during training, KD enables the student

model to learn more informative decision boundaries. This property makes knowledge distillation particularly effective for training lightweight models intended for deployment on edge and neuromorphic hardware, where strict constraints on memory usage and energy consumption must be satisfied.

IV. Experimental Setup

A. Dataset

The experiments are conducted using the MNIST [18] and EMNIST [19] datasets. MNIST is a widely used handwritten digit classification dataset consisting of 10 classes (digits 0–9), with 60,000 samples for training and 10,000 samples for testing. For the proposed experiments, the MNIST images are encoded into spike trains using a rate-coding scheme. Extended MNIST (EMNIST) includes handwritten digits and letters and shares the same image resolution and preprocessing parameters as MNIST. In this work, the EMNIST Balanced subset containing 47 classes is used, with 90,239 samples for training and 22,560 samples for testing. Both datasets consist of grayscale images with a resolution of 28 × 28 pixels, resulting in 784 input features per sample.

B. Network Architecture

Two different neural network architectures are employed for the MNIST and EMNIST datasets. For the MNIST dataset, a multilayer perceptron (MLP) architecture is used, while a convolutional neural network (CNN) is used for the EMNIST dataset due to its higher complexity. In both cases, a teacher–student framework is employed, where the teacher model contains significantly more parameters than the corresponding student model. For the MNIST dataset, the teacher MLP consists of two hidden layers with 512 and 256 neurons, respectively. The student MLP contains a single hidden layer with 32 neurons.

For the EMNIST dataset, the teacher network is a convolutional neural network (CNN) architecture consisting of three convolutional layers with increasing filter sizes (32, 64, and 128 filters, respectively), all using 3×3 kernels. The first two convolutional layers are followed by ReLU activation and max-pooling operations, while the third convolutional layer is followed by ReLU activation only. Feature extraction is followed by a flattening operation and a fully connected classifier composed of two hidden dense layers and a final output layer producing class logits.

The student network is designed as a compact CNN tailored for neuromorphic deployment. It consists of two convolutional blocks with 16 and 32 filters, respectively, using 3×3 kernels, each followed by ReLU activation and max-pooling. A final 1×1 convolutional layer maps the extracted features directly to class scores, followed by a global average pooling layer instead of fully connected layers. This design significantly reduces the parameter count and memory footprint while maintaining accuracy.

Figure 3 illustrates the detailed CNN architectures of the teacher and student models for EMNIST dataset. A quantitative comparison of the number of parameters and corresponding memory requirements between the teacher and student networks is provided in Table I.

C. Training and Deployment

In this experiment, the MetaTF framework is used to deploy the trained models on the Akida neuromorphic processor. MetaTF is a Python-based framework built on top of TensorFlow [20] and supports the deployment of neural networks on BrainChip's Akida hardware through ANN–SNN conversion. After training the artificial neural network (ANN), model quantization is required prior to conversion into a spiking neural network (SNN).

To study the impact of knowledge distillation on neuromorphic deployment, two different training strategies are considered: KD is applied before quantization and after quantization. In the first approach, the teacher model is initially trained using the TensorFlow framework. The student model is trained under the guidance of the teacher using the total loss defined in equation (1), which combines the standard student loss and the distillation loss. After ANN training, the student model is quantized. This quantization process typically introduces a slight drop in accuracy; therefore, Quantization Aware Training (QAT) is performed before converting the model into an Akida-compatible format for hardware deployment. In the second approach, the student model is quantized, and KD is applied during the QAT process. In this case, the teacher's logits are used during the QAT to guide the student model and improve performance under quantization constraints.

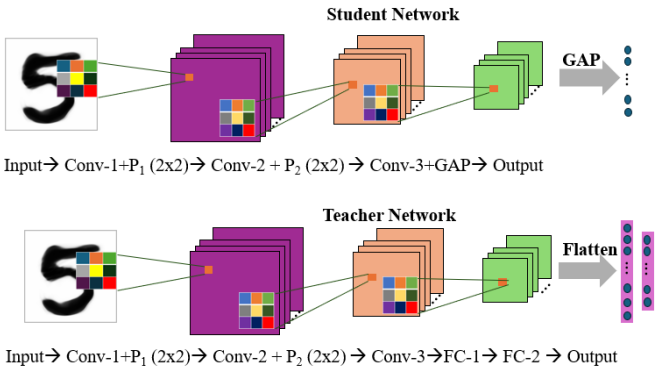

Fig. 3. CNN architecture of teacher and student models for EMNIST dataset. Conv denotes a convolution layer, P indicates max-pooling, FC represents a fully connected layer, and GAP denotes global average pooling.

TABLE I. Comparison of Model Complexity Between Teacher And Student

Dataset	Model	Parameters	Model Size (MB)	Parameter Reduction
MNIST	MLP teacher	535,818	2.040	-
MNIST	MLP student	25,450	0.099	95.25%
EMNIST	CNN teacher	906,927	3.460	-
EMNIST	CNN student	6,351	0.024	99.30%

In the second approach, the student model is first quantized, followed by the application of knowledge distillation using the same loss formulation given in (1). The quantized and distilled model is then converted into an Akida model and deployed on the neuromorphic hardware. The ANN-to-SNN conversion is performed using the convert function from the cnn2snn module,

which outputs a quantized Akida model suitable for execution on the Akida accelerator.

The Akida processor supports low-bit quantization, and in this work, the classifier student models are quantized from 32-bit floating-point precision to an 8/4/4 fixed point configuration [21]. All models are trained on a desktop PC equipped with an Intel Core™ i7-9700F processor (3 GHz) and 16 GB DDR4 RAM. During training, the Adam optimizer is used with a learning rate of 0.001 and sparse categorical cross-entropy loss for both teacher and student models across both architectures (MLP and CNN).

For the MNIST dataset, both teacher and student models are trained for 5 epochs, whereas for the EMNIST dataset, 10 training epochs are used. The same training procedure is applied consistently across both network architectures to ensure a fair comparison. Fig. 4 shows the training accuracy of the student model with KD before quantization for both datasets.

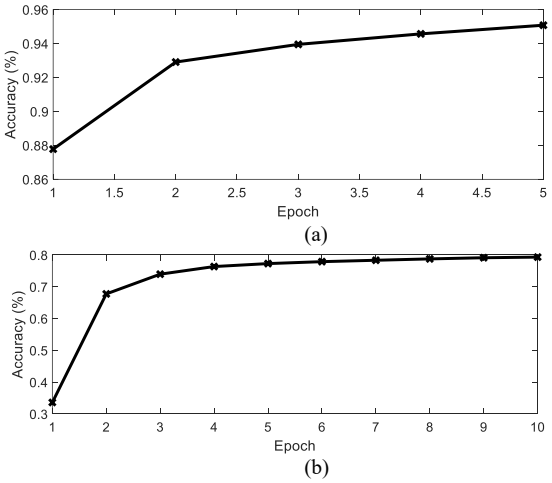

Fig. 4. Training accuracy of the student model trained using KD before quantization for (a) MNIST and (b) EMNIST datasets.

V. Results and Discussion

A. Performance Analysis

In this section, the impact of knowledge distillation on image classification performance when deployed on the Akida neuromorphic processor is analyzed. Tables II and III present the experimental results for the MNIST and EMNIST datasets, respectively. The BrainChip AKD1000 PCIe board was used for hardware deployment, where the trained and quantized models were converted to spiking neural networks and evaluated on the Akida accelerator. The results compare ANN and SNN accuracies for the teacher model, baseline student model, and student models trained using knowledge distillation before and after quantization. Additionally, the absolute improvement in SNN accuracy, measured in percentage points relative to the baseline student model is reported.

For both datasets, the teacher models achieve the highest accuracy in both ANN and SNN domains, serving as an upper performance bound. The baseline student models exhibit a noticeable accuracy degradation due to significant parameter reduction amounting to 95.25% for MNIST and 99.30% for EMNIST compared to the teacher model (Table I). However, applying knowledge distillation consistently improves the student model performance over the baseline in both ANN and SNN implementations.

For the MNIST dataset, knowledge distillation applied before quantization yields the highest improvement in SNN accuracy, achieving a gain of 1.69% over the baseline student model. Knowledge distillation after quantization also improves performance but with a smaller gain of 0.55%. This behavior can be attributed to both the relatively simple nature of the MNIST dataset and the use of a lightweight MLP architecture, which together result in minimal degradation during quantization and ANN-to-SNN conversion.

For the EMNIST dataset, although knowledge distillation improves ANN accuracy, a substantial accuracy drop is observed after quantization when distillation is applied before quantization. Knowledge distillation applied after quantization significantly mitigates this degradation, resulting in an 11.45% improvement in SNN accuracy over the baseline and outperforming pre-quantization distillation. This highlights the importance of quantization-aware knowledge distillation for more complex scenarios, where both the dataset complexity and the use of deeper CNN architectures introduce greater sensitivity to quantization and conversion effects.

TABLE II. Performance Comparison on MNIST Dataset with MLP Deployed on the Akida SNN Processor

Model	Training Method	ANN Accuracy (%)	SNN Accuracy (%)	Δ SNN Accuracy (%)
Teacher	NA	97.30	97.00	NA
Student	Baseline (No KD)	92.80	92.65	NA
Student	KD before Quantization	**95.32**	**94.34**	**1.69**
Student	KD after Quantization	93.35	93.20	0.55

TABLE III. Performance Comparison on EMNIST Dataset with CNN Deployed on the Akida SNN Processor

Model	Training Method	ANN Accuracy (%)	SNN Accuracy (%)	Δ SNN Accuracy (%)
Teacher	NA	88.31	82.10	NA
Student	Baseline (No KD)	78.90	65.10	NA
Student	KD before Quantization	**80.23**	69.47	4.37
Student	KD after Quantization	76.73	**76.55**	**11.45**

B. Cost Analysis

Table IV presents the cost analysis of teacher and student models for the MNIST and EMNIST datasets in terms of power consumption, energy per inference, static energy, and throughput when deployed on the Akida neuromorphic processor. Power measurements were obtained by mapping the trained models onto the Akida device and enabling on-chip power monitoring during inference. The Akida platform reports floor (static) power, average inference power, inference energy per frame, and throughput in frames per second.

TABLE IV. COST COMPARISION BETWEEN THE MODELS

Dataset	Model	Total Power (mW)	Total Energy (mJ)	Static Energy (mJ)	Throughput (FPS)
MNIST	Teacher	891.00	0.260	0.254	3457.81
	Student	879.82	0.255	0.253	3471.02
EMNIST	Teacher	1236.69	0.810	0.578	1524.22
	Student	929.28	0.260	0.257	3508.55

Static energy per inference is computed by dividing the measured floor power by the corresponding throughput. This metric represents the baseline energy consumption of the device that is independent of model activity, while the remaining portion of the total energy corresponds to the dynamic computation energy associated with neural activity.

For the MNIST dataset, both the teacher and student MLP models exhibit almost similar total power consumption, as the static power dominates the overall energy profile of the Akida device. Although the student model achieves a significant reduction in parameters, the total energy per inference is only slightly lower than that of the teacher model. This behavior is expected in neuromorphic hardware, where idle and background circuitry contribute a substantial portion of the power consumption when the deployed network is small. The student model does demonstrate lower energy per inference and slightly higher throughput, indicating improved efficiency at the inference level.

For the EMNIST dataset, a more pronounced difference is observed between the teacher and student models. The CNN teacher model exhibits substantially higher total energy per inference (about 3×) and static energy, accompanied by a significantly lower throughput (about 2×) when compared to the more efficient lightweight student model. This indicates increased computational load and sustained neural activity due to the higher architectural complexity of the teacher network. This demonstrates that model compression is particularly effective for complex datasets when deployed on neuromorphic hardware, significantly reducing energy per inference while maintaining high throughput. Overall, model compression determines the energy efficiency on neuromorphic hardware, while knowledge distillation improves the predictive performance of the compressed models without incurring additional power or energy overhead.

VI. CONCLUSION

This paper investigates the effectiveness of knowledge distillation for deploying lightweight neural networks on neuromorphic hardware. Results show that KD provides classification accuracy improvements by up to 11.45% over the baseline models, reducing accuracy degradation typically observed during ANN-to-SNN conversion. Thus, we achieve significant improvements in accuracy for parameter constrained edge deployable models. By systematically analyzing the interaction between distillation, quantization, and ANN-to-SNN conversion, this study will be a valuable asset to those aiming to deploy edge-oriented neuromorphic systems.

REFERENCES

[1] Zhang, Y., Song, C. and Zhang, D., 2020. Deep learning-based object detection improvement for tomato disease. *IEEE access*, *8*, pp.56607-56614.

[2] Murat, F., Yildirim, O., Talo, M., Baloglu, U.B., Demir, Y. and Acharya, U.R., 2020. Application of deep learning techniques for heartbeats detection using ECG signals-analysis and review. *Computers in biology and medicine*, *120*, p.103726.

[3] Alarfaj, F.K., Malik, I., Khan, H.U., Almusallam, N., Ramzan, M. and Ahmed, M., 2022. Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. *Ieee Access*, *10*, pp.39700-39715.

[4] Someshwar, D., Bhanushali, D., Chaudhari, V. and Nadkarni, S., 2020, July. Implementation of virtual assistant with sign language using deep learning and TensorFlow. In *2020 second international conference on inventive research in computing applications (ICIRCA)* (pp. 595-600). IEEE.

[5] Deng, L. and Liu, Y. eds., 2018. *Deep learning in natural language processing*. Springer.

[6] Huang, Y. and Chen, Y., 2020. Autonomous driving with deep learning: A survey of state-of-art technologies. *arXiv preprint arXiv:2006.06091*.

[7] Sun, Y., Ou, Z., Chen, J., Qi, X., Guo, Y., Cai, S. and Yan, X., 2021. Evaluating performance, power and energy of deep neural networks on CPUs and GPUs. In *Theoretical Computer Science: 39th National Conference of Theoretical Computer Science, NCTCS 2021, Yinchuan, China, July 23–25, 2021, Revised Selected Papers 39* (pp. 196-221). Springer Singapore

[8] An, D., Liu, P., Feng, Y., Ding, P., Zhou, W. and Yu, B., 2024. Dynamic weighted knowledge distillation for brain tumor segmentation. *Pattern Recognition*, *155*, p.110731.

[9] Ho, T.K.K. and Gwak, J., 2020. Utilizing knowledge distillation in deep learning for classification of chest X-ray abnormalities. *IEEE access*, *8*, pp.160749-160761.

[10] Cho, J.H. and Hariharan, B., 2019. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 4794-4802).

[11] Sun, S., Ren, W., Li, J., Wang, R. and Cao, X., 2024. Logit standardization in knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 15731-15740).

[12] Xu, Q., Li, Y., Shen, J., Liu, J.K., Tang, H. and Pan, G., 2023. Constructing deep spiking neural networks from artificial neural networks with knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7886-7895).

[13] Kushawaha, R.K., Kumar, S., Banerjee, B. and Velmurugan, R., 2021, January. Distilling spikes: Knowledge distillation in spiking neural networks. In *2020 25th International Conference on Pattern Recognition (ICPR)* (pp. 4536-4543). IEEE.

[14] Ponulak, F. and Kasinski, A., 2011. Introduction to spiking neural networks: Information processing, learning and applications. *Acta neurobiologiae experimentalis*, *71*(4), pp.409-433.

[15] Zhou, C., Zhang, H., Yu, L., Ye, Y., Zhou, Z., Huang, L., Ma, Z., Fan, X., Zhou, H. and Tian, Y., 2024. Direct training high-performance deep spiking neural networks: a review of theories and methods. *Frontiers in Neuroscience*, *18*, p.1383844.

[16] Parpart, G., Risbud, S., Kenyon, G. and Watkins, Y., 2023, August. Implementing and benchmarking the locally competitive algorithm on the loihi 2 neuromorphic processor. In *Proceedings of the 2023 International Conference on Neuromorphic Systems* (pp. 1-6).

[17] Hinton, G., Vinyals, O. and Dean, J., 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[18] MNIST dataset: https://www.tensorflow.org/datasets/catalog/mnist

[19] Cohen, G., Afshar, S., Tapson, J. and Van Schaik, A., 2017, May. EMNIST: Extending MNIST to handwritten letters. In *2017 international joint conference on neural networks (IJCNN)* (pp. 2921-2926). IEEE.

[20] Brainchip [online]: https://doc.brainchipinc.com/index.html.

[21] Brainchip [online]: https://brainchip.com/akida-foundations/