Unsupervised Adaptation of Spiking Networks in a Gradual Changing Environment

Zaidao Mei
Department of EECS
Syracuse University
Syracuse, New York, USA
zmei05@syr.edu

Mark Barnell
Air Force Research Laboratory
Information Directorate
Rome, New York, USA
mark.barnell.1@us.af.mil

Qinru Qiu
Department of EECS
Syracuse University
Syracuse, New York, USA
qiqiu@syr.edu

***Abstract*—Spiking neural networks(SNNs) have drawn broad research interests in recent years due to their high energy efficiency and biologically-plausibility. They have proven to be competitive in many machine learning tasks. Similar to all Artificial Neural Network(ANNs) machine learning models, the SNNs rely on the assumption that the training and testing data are drawn from the same distribution. As the environment changes gradually, the input distribution will shift over time, and the performance of SNNs turns out to be brittle. To this end, we propose a unified framework that can adapt non-stationary streaming data by exploiting unlabeled intermediate domain, and fits with the in-hardware SNN learning algorithm Error-modulated STDP. Specifically, we propose a unique self-training framework to generate pseudo labels to retrain the model for intermediate and target domains. In addition, we develop an online-normalization method with an auxiliary neuron to normalize the output of the hidden layers. By combining the normalization with self-training, our approach gains average classification improvements over 10% on MNIST, NMINST, and two other datasets.**

***Index Terms*—Spiking Neural Networks, in-hardware learning, domain adaptation**

I. Introduction

Spiking neural networks(SNNs) on neuromorphic hardware has become a promising machine learning technique as it mimics the behavior of the human's biological neural systems [1] in both spatial and temporal dynamics. Due to the sparse activities and event driven operation, SNNs is highly energy efficient when implemented in hardware. In recent studies, SNNs on neuromorphic hardware have made considerable progress in general machine learning tasks such as computer vision [2], object detection [3] and speech recognition [1].

In dynamically changing and non-stationary environments, distribution shift is an inevitable problem [4]. Such shift can be found in the operations of many autonomous edge devices that interact with physical world through sensors, where the input data distribution may alter slowly due to changes in the physical environment and sensor conditions, such as the sensor angle change or pixel damage due to aging. The shift of input distribution violates the condition of proper function of neural network models, which requires the testing and training data drawn from the same distribution, and will seriously degrade the model performance.

Some existing works mitigate the performance degradation using supervised training [5] [6] [7], by augmenting the training set with sufficient amount of data that cover all possible variations of testing domains. This usually associates with a high cost and unpredictable results as the actual testing environment is hard to anticipate, and annotated data in real-world settings is hard to obtain. Gradual domain adaptation, on the other hand, aims at unsupervised/semi-supervised model adaptation and self-calibration. It takes the advantage of the fact that the domain shift is a slow process. By leveraging unlabeled intermediate domains where subtle changes take place, it incrementally adapts the target model. In particular, given a model trained on limited data from the source domain D_0, and unlabeled data drawn from the intermediate domain's D_i, $(1 \leq i \leq N-1)$, where i indicates their sequential order of occurrence. Gradual domain adaptation aims to assist the generalization of the pre-trained model when tested on target domain D_N. Although the target domain is significantly different from the source domain, the shift takes place gradually, i.e., $\mathcal{D}(D_{i-1}, D_i) \leq \epsilon$, $1 \leq i \leq N-1$, $\mathcal{D}(\cdot)$, measures the distance between two distributions and ϵ is a very small number.

While Domain shift and adaptation is a concern for both ANN and SNN, almost all of the existing works in this area targets at ANN [6] [8] [9]. Adaptation of SNNs has its own challenges and considerations. This work discusses unsupervised gradual domain adaption of SNN models. To make sure that the proposed framework supports truly online in-situ adaptation, we build it on top of Error-Mouldated Spike Timing Dependent plasticity [10](EMSTDP). EMSTDP is an SNN learning rule that incorporates the idea of difference target propagation [11]. It updates the local weight with the difference between "target spikes" and the "propagated spikes". The simple weight update rule enables EMSTDP to be implemented on low power neuromorphic hardware such as Loihi [12].

We present a self-supervised framework. Our framework consists of a feature extractor that is trained on the source domain D_0, and two "swapping" classifiers with different architecture, one of them will serve as the "teacher" model to generate pseudo labels, and the other serves as the "student" model trained with unlabeled input data and their corresponding pseudo labels in every self-training episode. The

classifiers will exchange their roles once finishing a self-training episode. Although domain adaptation using teacher-student models has been studied for ANNs in [13], the spiking domain inference and online learning impose extra challenges. This paper discusses techniques that are necessary to mitigate those challenges, such as "swapping" teacher and student models, new label calibration algorithm and novel neuron circuits for online normalization, etc. Our contributions are summarized as follows:

- We present a self-training framework based on EMSTDP learning for Spiking Neural Networks to realize gradual domain generalization.
- We have demonstrated that techniques such as label calibration, online neuron normalization are necessary to regularize the firing activities of neurons for better adaptation.
- We test our approach on 4 different datasets. The results show that the gradual domain adaptation can enhance the robustness of the SNN model and improve the classification accuracy by over 10% on average.

II. Backgrounds

A. Spiking neurons and Error-modulated STDP learning

Different learning algorithms of SNNs have been explored. The majority of them are based on Backpropagation-Through-Time(BPTT) [14] [15], they adopt similar neural network topology as traditional artificial neural networks(ANNs), but use surrogate activation functions to replace the non-differentiable spike function. Another class of SNN learning algorithms are based on spike-timing-dependent plasticity (STDP) [16] [17], which is a local learning rule. According to the STDP rule, the change of the synaptic weight is determined by the timing of pre-synaptic and post-synaptic spikes, both of which can be obtained in a single neuron locally.

We use Error-modulated STDP (EMSTDP) [10] as our learning rule. The neurons adopt integrate and fire model specified by the following equations:

$$u_i^l(t) = \sum_j w_{ji} \cdot s_j^{l-1}(t) + u_i^l(t-1) + b_i^l \tag{1}$$

$$s_i^l(t) = \begin{cases} 1, & \text{if } u_i^l(t) > V_{th} \\ 0, & \text{otherwi} \end{cases} \tag{2}$$

where $u_i^l(t)$ denotes the membrane potential of the ith neuron in layer l at time t, $s_j^{l-1}(t)$ is input at time t, w_{ji} is the synaptic weight between the pre-synaptic neuron j in layer $l-1$, and the post-synaptic neuron i in layer l, b_i^l here denotes the bias. Equation 2 gives the Heaviside step activation functions. A spike will be generated once the membrane potential $u_i^l(t)$ exceeds the threshold V_{th}.

An SNN with EMSTDP learning consists of two networks, a feedforward network that performs the inference and learning, and a feedback network that propagate back errors (i.e., gradients). Feedback alignment is adopted, such that the two networks do not need to keep the same copy of weights. A set of random weight, **B**, is used in the feedback network. The error propagated in the feedback network is calculated as the following: $e_i^l = \sum_j e_j^{l+1} B_{i,j} h_i^{'}(u_i^l)$ where e_j^{l+1} is the error of neuron j in layer $l+1$ and $h_i^{'}(u_i^l)$ is the derivative of the output activation function of neuron i in layer l. The Heaviside step activation is not differentiable, gradient surrogation has to be adopted. Given that the relationship between the number of output spikes and neuron's accumulated sub-membrane potential can be approximated as a shifted ReLU function, $h(u_i) = max(\frac{u_i}{vth} - 1, 0)$, EMSTDP replaces $h_i^{'}$ using the following equation:

$$h_i^{'}(u_i^l) = \begin{cases} 1, & u_i^l(t) > V_{th} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

The EMSTDP operates in two phases, free running phase and error correction phase. In the free running phase, the feedforward network makes prediction,h_j^l,based on the received input. In the error correction phase, the prediction is compared with the target and the feedback network calculates errors for each layer. The error is then sent to the feedforward network to correct the prediction and generate the new prediction $\hat{h_i^l}$, where $\hat{h_i^l} - h_i^l \propto e_j^i$, $h_i = \lfloor \frac{u_i}{vth} \rfloor$, the weight update is calculated as $\Delta w_{ji} = \eta(\hat{h_i^l} - h_i^l) h_j^{l-1}$.

B. Gradual domain adaptation

The preliminary of gradual domain adaptation is that one has access to the source domain $D_0 := \{X_0, Y_0\}$ with input x_0 and its label y_0, where $\mathrm{x}_0 \in X_0, \mathrm{y}_0 \in Y_0$. The knowledge learned from the source domain must be applied to the target domain $\mathcal{D}_N$, which is notably different from the source domain, i.e., $\mathcal{D}(D_N, D_0) \gg \epsilon$, where ϵ is a very small number. Due to the discrepancy between source and target domain, applying the model trained on source domain to target domain can cause severe accuracy degradation. The transition from the source domain to the target domain is gradual. A sequence of intermediate domains, $D_i := \{X_i\}, 1 \leq i \leq N$, can be obtained from the slow transition process, such that $\mathcal{D}(D_{i-1}, D_i) < \epsilon, 1 \leq i < N$. Both the intermediate domains and the target domain are unlabeled. Our goal is to adapt the knowledge learned in the source domain and apply it to the target domain during testing time by utilizing the unlabeled intermediate domains from the slow transition process. To address above issues, several ANN works investigate gradual domain adaptation [8] [18] [9]. The most relevant existing work in spiking domain is [7], which deploys transfer learning to SNN. However, [7] requires labeled training data from the target domain, while we consider more realistic case where both target and intermediate domains are unlabeled.

III. Methods

We propose a self-training framework for gradual domain adaptation with a focus on its unique characteristics designed

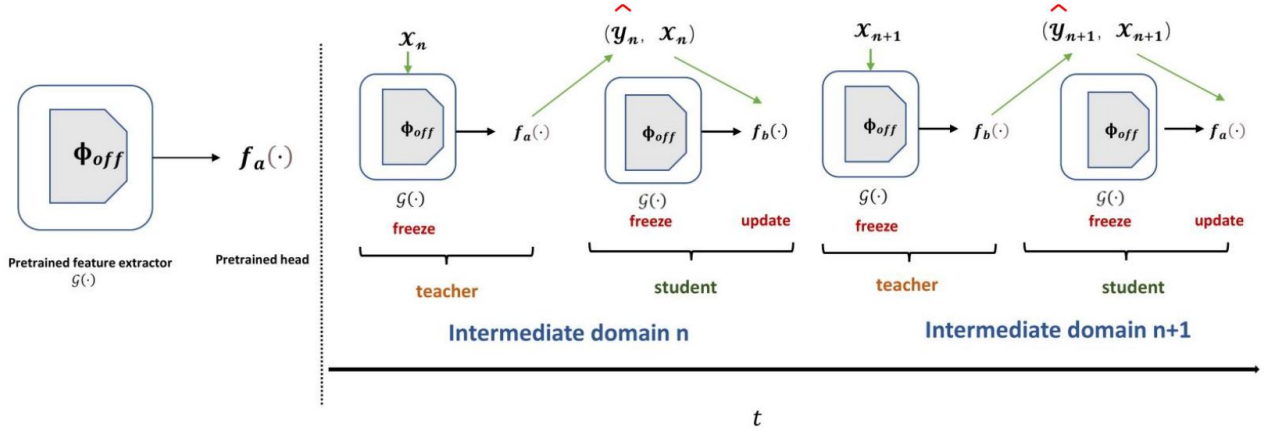

Fig. 1: Self-training framework for gradual domain adaptation

for SNN and EMSTDP learning.

A. Self-Training framework for EMSTDP learning

Self-training has been widely used for semi-supervised learning. It improves the performance of the model by retraining the model with self-generated pseudo labels derived from unlabeled data. In this work, we implement a teacher-student pipeline to produce the pseudo labels for the subsequent data. Specifically, given a teacher model $f(\cdot;\theta_t)$ trained with data from domains $D_0, D_1, ..., D_t$, for intermediate domain D_{t+1}, the pseudo label $\hat{y}_{t+1}$ is generated using the teacher model $\hat{y}_{t+1} = f(x_{t+1};\theta_t)$, $\hat{y}_{t+1}$ is further used to train the student model $f(;\theta_{t+1})$; by minimizing the cost function $\mathcal{L}$, such that $\theta_{t+1} = \arg\min_{\theta_{t+1}} \mathcal{L}(f(x_{t+1};\theta_{t+1}), \hat{y}_{t+1})$.

In most ANN self-training frameworks, the same network $f(x;\theta)$ is used as both teacher and student models in every self-training iteration, and the student model is adapted directly from the teacher model. However, the same strategy gives poorl results in SNN based implementation. Due to the similarity between the adjacent intermediate domains, each pair of teacher and student models are very similar. Which means the learning process operates on small error signals and low gradients all the time. With spike domain learning, the error signals e and the learning gradients must be represented using spiking activities, hence are highly quantized. When the values of error signal and gradients are too low, the rounding error will be more significant than the signal itself and cause the model to update inaccurately. In contrast to the conventional approach, we utilize networks with slightly different architectures as teacher and student models, and swap them for different intermediate domains, so that we can get sufficiently large error and gradient to keep the learning going on.

Specifically, our self-training framework consists of a convolutional spiking neural network $\mathcal{G}_(\cdot;\Phi)$, which serves as a feature extractor and two 3-layer EMSTDP classifiers $f_a(\cdot;\theta_i)$ and $f_b(\cdot;\omega_i)$, where i is the self-training episode index . The feature extractor $\mathcal{G}_(\cdot;\Phi)$ and the first classifier $f_a(\cdot;\theta_0)$ are pretrained on source domain. The feature extractor $\mathcal{G}(\cdot;\Phi)$ will remain fixed during self-training. As shown in figure 1, the two classifiers serve the role as teacher and student alternatively in different intermediate domains. For example, for intermediate domain n where $D_n = \{X_n\}$, the classifier $f_a(\cdot;\theta_{n-1})$ serves as the teacher model to generate pseudo label $\{\hat{Y_n}\}$. The student model $f_b(\cdot;\omega_n)$ is retrained with data and pseudo labels $\{X_n, \hat{Y_n}\}$. When we moved to domain $n+1$, to obtain a set of updated parameters θ_{n+1}, the updated classifier $f_b(\cdot;\omega_n)$ serves as the teacher model to generate pseudo labels $\hat{y_{n+1}}$, and $f_a(\cdot;\theta_{n-1})$ becomes the student model, and a new set of parameters θ_{n+1} is obtained.

By utilizing different classifiers during self-training, the testing accuracy gains significant improvements as shown in experimental results.

B. Label calibration and Sharpening

The motivation behind label filtering is to filter noisy labels based on the confidence score [9, 26]. The confidence score is defined as the entropy of the classification output and is specified by (5), where h_i denotes the spike count of the output neuron corresponding to output layer, and N is the size of output layer. A high enytropy means high uncertainty and low confidence. The label and its associated input will be discarded if the confidence score exceeds the threshold. We set the entropy threshold to 0.7 in this paper.

Instead of directly using the output of the teacher model as the target, we perform label sharpening to amplify error(gradient) for learning. Given the pseudo label $t = \arg\max_i h_i$, the target sequence of output neuron t is a spiking sequence with rate 1 and the target of all other neurons is expected to have no spikes. We further reduce the spiking threshold of error neurons in the feedback path to make them

Algorithm 1 Self-training framework

Require: Pre-trained classifier $f_a(\cdot, \theta_a)$, untrained classifier $f_b(\cdot, \omega)$, unlabeled intermediate domain data $\{D_1, ..., D_{N-1}\}$, pre-trained feature extractor $\mathcal{G}(\cdot, \Phi)$

$i \leftarrow 0$

while $i \neq N$ **do**

for $x_j \in D_i$ **do**

$\mathbf{z} \leftarrow \mathcal{G}(x_j; \Phi)$

if i is odd **then**

$\hat{y_j} \leftarrow f_b(\mathbf{z}; \omega)$

$\hat{y_j} \leftarrow$ **Entropy filtering**$(\hat{y_j})$

if $\{\hat{y}, x_j\}$ is not filtered **then**

Sharpen$(\hat{y_i})$

train $f_a(\cdot; \theta)$ with $\{\hat{y_j}^{'}, x_j\}$

end if

else if i is even **then**

$\hat{y_j} \leftarrow f_a(\mathbf{z}; \theta)$

Entropy filtering$(\hat{y_j})$

if $\hat{y_j}$, x_j is not filtered **then**

Sharpen$(\hat{y_j})$

train $f_b(\cdot; \omega)$ with $\{\hat{y_j}, x_j\}$

end if

end if

end for

$i \leftarrow i + 1$

end while=0

more sensitive to the error signals. The overall flow of the self-training is given in Algorithm 1.

$$p_i = \frac{softmax(h_i)}{\sum_{i=0}^{N} softmax(hi)} \tag{4}$$

$$S = -\sum p_i \log p_i \tag{5}$$

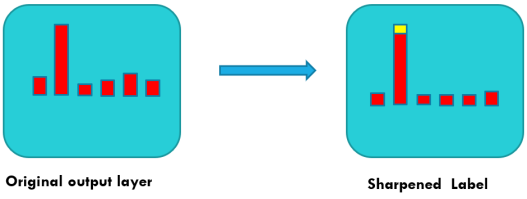

Fig. 2: label sharpening

C. Online Normalization for EMSTDP

Batch normalization is a go-to tool for training traditional ANNs and it mitigates covariance shifts of the inputs. As stated in [8] the batch normalization also improves the performance of gradual domain adaptation. Several spiking-based batch normalization methods have been proposed recently, for example, [19] applies temporal Batch Normalization Through Time (BNTT) to every convolutional layer along the time axis with a given batch. More specifically, BNTT directly normalizes $\sum_j w_{ij} s_j^t$ in every time step within a batch. However, batch normalization is hardware expensive as it requires extra memories to store neuron's activities for the entire batch. Furthermore, it is not compatible with online learning , where batch size is set to 1. To solve this problem, we propose an online normalization technique that whitens the firing rates of a given layer for EMTDP.

Dataset	Different Classifiers	Single Classifier
Rotated-MNIST	**90.7%**	79.1%
Cropped-MNIST	**92.8%**	89.5%
Portraits	**81.7%**	78.4%
Rotated-NMNIST	**84.0%**	70.8%
Rotated-Fashion MNIST	**49.7%**	33.3%

TABLE I: Classification accuracy of Different classifiers and Single classifier

The proposed online normalization has an auxiliary neuron-based normalization is inspired by NeuNorm network [20]. The dynamics of the auxiliary neuron is stated in Eq.6, where $x^{t+1,n}$ is the membrane potential of the auxiliary neuron x at time step $t+1$ for layer n, α is the hyper coefficient of time constant, H is the number of neurons in layer n and $s_j^{t+1,n}$ is the output spike at time $t+1$ in layer n. In short, the membrane potential $x^{t+1,n}$ is a leaky integration of all the spiking activities of layer n. Eq.7 describes how the auxiliary neuron affects the neurons in the subsequent layer, where $U_i^{n+1,t}$ is the membrane potential of the ith neuron in the $(n+1)$th layer, $g(\cdot)$ is the Heaviside step function, which transforms auxiliary neuron's membrane potential to discrete spikes. C_i is a trainable negative parameter. Intuitively, the auxiliary neuron can be regarded as a dynamic bias that regulates the spiking dynamics from the previous layer to the next layer. If the firing rate of the previous layer is too high, then the auxiliary neuron will assist to decrease the membrane potential and vice versa. Consequently, the auxiliary neuron normalizes the firing activity of the next layer by using the feature-wise statistics.

$$x^{t+1,n} = \alpha x^{t,n} + \frac{1-\alpha}{H} \sum_j s_j^{t+1,n} \tag{6}$$

$$U_i^{n+1,t+1} = U_i^{n+1,t} + C_i\, g(x^{t+1,n}) + \sum_j w_{i,j} s_j^{t+1,n} \tag{7}$$

IV. Experiment Settings

We tested the proposed framework on both neuromorphic datasets (neuromorphic MNIST) and static datasets (MNIST, Potraits, and Fashion MNIST). For each dataset, parametric non-linear transformation is applied to the data to emulate domain shifting. By controlling the parameters in transformation, we generate different intermediate and target domains.

A. MNIST dataset

MNIST [21] is a popular dataset for classification task, it consists of handwritten digits from zero to nine. Two

variations of MNIST dataset are generated to represent different domain shifts.

1) Rotated MNIST: Rotated MNIST consists of images that are manually rotated by up to 30 degrees. The first 10000 training samples are unrotated and used as the source domain. The next 50000 samples are used to generate data from intermediate domains. Those 50000 samples are rotated gradually from 0 to 30 degrees. All intermediate domain samples are unlabeled. The images are evenly distributed over all degrees of rotations to feature a steady change of environment. Those 5 intermediate domains are split into 5 intermediate domains; each domain has 10000 samples. The target domain has 10000 testing images that are rotated by 30 degrees.

2) Cropped MNIST: In cropped MNIST dataset, we randomly set the pixels of images to 0 to emulate inputs from an aging camera with dead pixels. The partition of the source, intermediate, and target domains is the same as the Rotated MNST. The intermediate domains contain images with the ratio of dead pixels increasing from 0% to 60%. The target domain consists of images that have 60% dead pixels.

B. Fashion MNIST

Fashion MNIST [22] is a more complex dataset comprising of 28×28 grayscale images of 70, 000 fashion products from 10 categories. We further generate rotated version where the setting is similar to Rotated MNIST, 20000 samples are chosen as the source domain, 50000 samples as the intermediate domains, which are also split into 5 runs. The target domain consists of 10000 30-degree rotated samples.

C. N-MNIST dataset

The Neuromorphic-MNIST(N-MNIST) [23] converts static MNIST images to 3 saccades by using the DVS camera. The first saccade starts from the first 105 ms (0–105 ms), the second saccade in the next 105 ms (105–210 ms), and the third saccade in the subsequent 105ms. To process this event-based data, we rotate the sampled frames up to 30 degrees. The domain settings of Rotated N-MNIST are the same as Rotated MNIST, and the event based data transformation is the same as Rotated MNIST.

D. Potraits

The portraits dataset [24] consists of photos of high school students with different haircuts and facial features across decades. The hair cut and facial features of those people varies from time. For this dataset, we adopt settings from [8], where they build a binary classifier to classify genders of given images across ages. The source and target domains consist of 2000 images while the intermediate domains have 14000 images that are split into 5 runs.

V. Experimental Result

A. Training Details

The network comprises of 3 spiking convolutional layers [25] for feature extraction and two fully connected layers for image classification. Specifically, for all datasets, the filter size is 32, kernel size is 5. and the stride is 2 the padding of these three convolutional layers are 2. The feature extractor is pre-trained on the source domain and is not adaptable during self-training. To differentiate student and teacher models, two 2-layer fully connected networks with different architectures are used for classification. One of them has 512 and 200 in the hidden layers, and the other has 512 and 100 neurons during self-training. The fully connected layers are adaptable during self-training.

Dataset	Gradual Adaptation	No Adaptation	Target Adaptation	Oracle
Rot MNIST	**90.7%**	70.7%	73.8%	93.5%
Crop MNIST	**92.8%**	87.1%	88.8%	94.9%
Portraits	**81.7%**	76.4%	76.2%	83.0%
Rot NMNIST	**84.0%**	69.9%	63.3%	92.6%
Rot F-MNIST	**49.7%**	31.3%	33.0%	49.3%

TABLE II: Classification Accuracy of Different Training Methods

Dataset	with Normalization	w/o Normalization
Rotated MNIST	**90.7**%	88.8%
Cropped MNIST	92.8%	**92.8**%
Portraits	**81.7**%	80.9%
Rotated NMNIST	**84.0**%	83.2%
Rotated Fashion MNIST	**49.7**%	45.3%

TABLE III: Ablation Study of Online Normalization

B. Results Analysis

we compare the classifiers with gradual domain adaptation to models without domain adaptation. We also compare gradual domain adaptation with target domain adaptation where model directly retrains on the target domain. Finally, we also list the accuracy of a model that is trained on the target domain with labels which is referred as $oracle$. Obviously, the oracle sets the upper bound of the accuracy of the domain adaptation. Table II lists the accuracy of the gradual domain adaptation and three baseline models for the 5 datasets. We can see that our self-trained gradual domain adaptation model outperforms both the baseline models on all datasets. For Rotated MNIST dataset, it boosts accuracy for over 10%. Compared to the model without adaptation, the self-training with gradual domain adaptation improves the accuracy by more than 5% on every dataset. The result also indicates that our self-training approach can achieve competitive accuracy as the oracle model, which is trained supervisly on the target domain. The results in Table II demonstrate that the model is able to adapt to the shifted data distribution in the target domain.

C. Ablation study

We further evaluate the impact of different enhancement techniques on the self-trained gradual domain adaptation.

Table I shows the impact of using different teacher and student models. As we can see, compared to the baseline

Dataset	with Filtering	w/o Filtering
Rotated MNIST	**90.7**%	80.5%
Cropped MNIST	**92.8**%	87.6%
Portraits	**81.7**%	79.5%
Rotated NMNIST	**84.0**%	78.5%
Rotated Fashion MNIST	**49.7**%	35.3%

TABLE IV: Ablation Study of Entropy Filtering

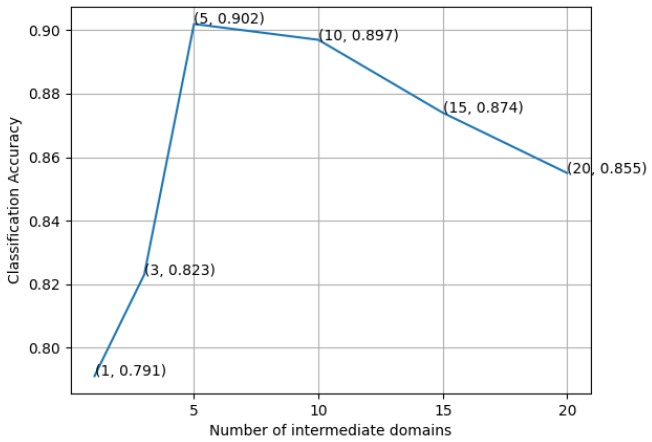

Fig. 3: Rotated MNIST results for different number of intermediate domains

where the same SNN architecture is used for both teacher and student, our approach improves accuracy by 3.3% to 16.3%

Table III compares the accuracy of self-training with and without online normalization. As we can see, using online normalization improves the accuracy by 1.47% on average over 5 datasets. Table IV compares the performance of self-training with and without entropy filtering. As we can see, entropy filtering plays a crucial role in self-training framework, as it directly determines the quality of pseudo labels. The average accuracy improvements on 5 datasets is 10.13%. We further investigate how the granularity of the intermediate domain affects the quality of domains adaptation. We vary the number of the intermediate domains on Rotated MNIST dataset and collect the classification accuracy of he adapted model on the target domain. As shown in Figure 3, the size of the intermediate domain plays an important role in determining the performance of self-adaptation. Too many intermediate domains or too few intermediate domains lead to poor results. On one hand, a very low number of intermediate domains means that its data distribution shifts more significantly from the domain where the teacher model is trained. Hence the teacher model is not able to generate correct pseudo labels for the self-training. On the other hand, a very high number of the intermediate domains means that each one of them contains small number of samples. Hence, the training of the student model is likely to have over-fitting. In real application, the indication that we have reach the boundary of the existing intermediate domain is when the confidence of the teacher model drops significantly while the performance of the student model starts to picking up. It is also when we should swap the role of student and teacher models.

VI. Conclusion

In this paper, we present a self-training framework to adapt SNN based classifiers to the gradually shifted input domain. The adaptation is built on top of online in-hardware learning algorithm, EMTSDP, so that it can be realized on edge devices. We show that, to support online in hardware adaptation, special considerations need to be placed on model architecture selection and regularization techniques. Experimental results show that the classifier with self-adaptation improves accuracy by 5%-20%.

VII. Acknowledgement

This work is received and approved for public release by the Air Force Research Laboratory (AFRL) on 16 Aug 2022, case number AFRL-2022-3919. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors, and do not necessarily reflect the views of AFRL or its contractors. The project was partially supported by NSF I/UCRC ASIC Center (CNS-1822165) and AFRL's NOCAL contract.

References

[1] J. P. Dominguez-Morales, Q. Liu, R. James, D. Gutierrez-Galan, A. Jimenez-Fernandez, S. Davidson, and S. Furber, "Deep spiking neural network model for time-variant signals classification: a real-time speech recognition approach," in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2018, pp. 1–8.

[2] L. Zhu, X. Wang, Y. Chang, J. Li, T. Huang, and Y. Tian, "Event-based video reconstruction via potential-assisted spiking neural network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 3594–3604.

[3] Y. Wang, Y. Xu, R. Yan, and H. Tang, "Deep spiking neural networks with binary weights for object recognition," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 13, no. 3, pp. 514–523, 2020.

[4] J. a. Gama, I. Žliobaitundefined, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, mar 2014. [Online]. Available: https://doi.org/10.1145/2523813

[5] E. Tzeng, J. Hoffman, T. Darrell, and K. Saenko, "Simultaneous deep transfer across domains and tasks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 4068–4076.

[6] P. Koniusz, Y. Tas, and F. Porikli, "Domain adaptation by mixture of alignments of second-or higher-order scatter tensors," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4478–4487.

[7] Q. Zhan, G. Liu, X. Xie, G. Sun, and H. Tang, "Effective transfer learning algorithm in spiking neural networks," *IEEE Transactions on Cybernetics*, pp. 1–13, 2021.

[8] A. Kumar, T. Ma, and P. Liang, "Understanding self-training for gradual domain adaptation," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5468–5479. [Online]. Available: https://proceedings.mlr.press/v119/kumar20c.html

[9] J. Hoffman, T. Darrell, and K. Saenko, "Continuous manifold based adaptation for evolving visual domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 867–874.

[10] A. Shrestha, H. Fang, Q. Wu, and Q. Qiu, "Approximating back-propagation for a biologically plausible local learning rule in spiking neural networks," in *Proceedings of the International Conference on Neuromorphic Systems*, 2019, pp. 1–8.

[11] D.-H. Lee, S. Zhang, A. Fischer, and Y. Bengio, "Difference target propagation," 2014. [Online]. Available: https://arxiv.org/abs/1412.7525

[12] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, Y. Liao, C.-K. Lin, A. Lines, R. Liu, D. Mathaikutty, S. McCoy, A. Paul, J. Tse, G. Venkataramanan, Y.-H. Weng, A. Wild, Y. Yang, and H. Wang, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[13] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning. 2006," *Cambridge, Massachusettes: The MIT Press View Article*, vol. 2, 2006.

[14] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/82f2b308c3b01637c607ce05f52a2fed-Paper.pdf

[15] W. Zhang and P. Li, "Temporal spike sequence learning via backpropagation for deep spiking neural networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 022–12 033, 2020.

[16] F. Ponulak and A. Kasiński, "Supervised learning in spiking neural networks with resume: sequence learning, classification, and spike shifting," *Neural computation*, vol. 22, no. 2, pp. 467–510, 2010.

[17] S. Song, K. D. Miller, and L. F. Abbott, "Competitive hebbian learning through spike-timing-dependent synaptic plasticity," *Nature neuroscience*, vol. 3, no. 9, pp. 919–926, 2000.

[18] H. Wang, H. He, and D. Katabi, "Continuously indexed domain adaptation," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 9898–9907. [Online]. Available: https://proceedings.mlr.press/v119/wang20h.html

[19] Y. Kim and P. Panda, "Revisiting batch normalization for training low-latency deep spiking neural networks from scratch," *Frontiers in Neuroscience*, vol. 15, 2021. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2021.773954

[20] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, ser. AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. [Online]. Available: https://doi.org/10.1609/aaai.v33i01.33011311

[21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[22] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[23] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor, "Converting static image datasets to spiking neuromorphic datasets using saccades," *Frontiers in Neuroscience*, vol. 9, 2015. [Online]. Available: https://www.frontiersin.org/article/10.3389/fnins.2015.00437

[24] S. Ginosar, K. Rakelly, S. Sachs, B. Yin, and A. A. Efros, "A century of portraits: A visual historical record of american high school yearbooks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 1–7.

[25] H. Fang, A. Shrestha, Z. Zhao, and Q. Qiu, "Exploiting neuron and synapse filter dynamics in spatial temporal learning of deep spiking neural network," in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, C. Bessiere, Ed. International Joint Conferences on Artificial Intelligence Organization, 7 2020, pp. 2799–2806, main track.