SEnsitivity Modulated Importance Networking and Rehearsal for Spike Domain Incremental Learning

Zaidao Mei
Syracuse University
Syracuse, United States
zmei05@syr.edu

Boyu Wang
Syracuse University
Syracuse, United States
bwang30@syr.edu

Daniel Rider
Syracuse University
Syracuse, United States
dprider@syr.edu

Nathan McDonald
AFRL/RITB
Syracuse, United States
nathan.mcdonald.5@us.af.mil

Qinru Qiu
Syracuse University
Syracuse, United States
qiqiu@syr.edu

Abstract

Incremental learning is a challenging task in the field of machine learning, and it is a key step towards autonomous learning and adaptation. With the increasing attention on neuromorphic computing, there is an urgent need to investigate incremental learning techniques that can work in this paradigm to maintain energy efficiency while benefiting from flexibility and adaptability. In this paper, we present SEMINAR (sensitivity modulated importance networking and rehearsal), an incremental learning algorithm designed specifically for EMSTDP (Error Modulated Synaptic-Timing-Dependent Plasticity), which performs supervised learning for multi-layer spiking neural networks (SNN) implemented on neuromorphic hardware, such as Loihi. SEMINAR uses critical synapse selection, differential learning rate and a replay buffer to enable the model to retain past knowledge while maintaining flexibility to learn new tasks. Our experimental results show that, when combined with the EMSTDP, SEMINAR outperforms different baseline incremental learning algorithms and gives more than 4% improvement on several widely used datasets such as Split-MNIST, Split-Fashion MNIST, Split-NMNIST and MSTAR.

Keywords: Spiking Neural Network, Neuromorphic computing, continual learning

1 Introduction

The proliferation of "big data" applications presents significant challenges in terms of speed and scalability for traditional computer systems. The increasing performance gap between CPUs and memory, commonly referred to as the "memory wall," greatly impedes the performance of traditional Von Neumann machines. As a result, neuromorphic computing systems have garnered considerable attention. These systems operate by emulating the charging and discharging processes of neurons and synapse potential in a biologically plausible computing paradigm. Electrical impulses or spikes facilitate inter-neuron communication. The unique encoding of information in the spike domain enables asynchronous event-driven computation and communication, potentially resulting in high energy efficiency. This new computing paradigm has been applied to various different machine learning applications from pattern classification to robotic controls [4]. Recent works [22, 23] have successfully realized supervised learning of multi-layer spiking neural networks (SNN) on Intel Loihi processor, a neuromorphic computing platform with bio-inspired neuron and synapse models. The proposed EMSTDP ((Error Modulated Synaptic-Timing-Dependent Plasticity) algorithm enables backpropagation to be performed on SNNs, where the same type of integrate-and-fire neurons are used to implement feed forward and feedback networks for data and error propagation.

Traditional supervised learning assumes that the training set provides an adequate representation of all potential application scenarios [26]. However, this paradigm may not be compatible with real-world applications. In real-world scenarios, there may be a need to continue augmenting a trained model to learn additional knowledge, such as new classes or tasks, as the model encounters novel input data distributions after its deployment. This life-long learning problem, which aims to bring machine learning more in-line with behaviors exhibited by the human brain, is known as incremental learning [14]. Like the brain, models must employ learning algorithms to incorporate new data into their existing knowledge. There is an urgent need to develop incremental learning for neuromorphic computing, which can provide flexibility and adaptability at a lower energy cost [25].

As discussed in paper [18], most recent works on continual learning focus on task incremental learning and class incremental learning. The difference between these two paradigms is that task incremental learning provides a learning id for each task, while class incremental does not. Specifically, in task incremental learning, the model learns the training sample in a tuple format $(\mathbf{x}, \mathbf{y}, \alpha)$, where α is the learning ID. And the models are normally equipped with multi-head[31] [11] classifier, where each head is trained to accommodate the designated task. During the testing phase,

the test samples are also provided with task ID ($x_{test}, y_{test}, \alpha$), and the model can then choose which classifier to use. In this setting, the model is more likely to differentiate the intra-task samples rather than the whole training categories. On the other hand, class incremental learning does not provide task IDs and the model classifies all the classes with a single output layer during the entire learning period. In this work, we focus on class incremental learning and test our framework under such paradigm.

A significant challenge of incremental learning is catastrophic forgetting, in which previously learned knowledge is lost while acquiring new information [8, 14, 15]. To address this issue, the deep learning community has proposed multiple techniques. One such approach involves controlling the forgetting through regularization [6, 11, 16, 30, 31]. Another common method uses a buffer to store examples for later replay [3, 5, 13, 19, 24]. The buffer that stores these examples of previously trained data is referred to as the *exemplar memory*. Additionally, dynamic architecture has been explored as another potential solution [1, 2, 7, 9, 21, 29].

While the effectiveness of incremental learning algorithms for traditional artificial neural networks (ANNs) has been demonstrated in previous studies, their applicability to neuromorphic computing hardware is limited by coarse-grained neuron activation and error gradients. These limitations hinder the performance of incremental learning technique when applied to SNNs. In this paper, we present SEMINAR (SEnsitivity Modulated Importance Networking and Rehearsal), an algorithm for class incremental learning on neuromorphic hardware that aims to mitigate catastrophic forgetting by selecting importance nodes via sensitivity analysis, exemplar balancing, and differential learning rate. SEMINAR is a hybrid of exemplar method and dynamic architecture and comprises three major components. First, SEMINAR employs a dynamic network structure during incremental learning by "freezing" some important synapses and releasing frozen synapses. This allows the algorithm to retain important knowledge previously learned and allocates new resources for potential new knowledge. Unlike existing dynamic architecture based techniques [9, 10], the architecture morphing in SEMINAR is specifically designed for SNN training and is more volatile. Secondly, SEMINAR interleaves the exemplar memory of previously learned classes with the current training stream to obtain stable learning. Thirdly, SEMINAR adopts a differential learning rate for new and old training data.

To further leverage the advantages of SNNs on neuromorphic hardware, we implemented SEMINAR on top of EMSTDP and evaluated the combined incremental learning framework using several standard datasets. Our experimental results demonstrate that SEMINAR outperforms the best comparison methods on all the datasets by more than 4%.

The rest of this paper is structured as the following. Section 2 reviews previous methods used to address catastrophic forgetting. Section 3 formally describes the class incremental learning problem and introduces the EMSTDP learning rule. The SEMINAR method is presented in Section 4. In Section 5, we provide experimental results and ablation studies to demonstrate the effectiveness of the SEMINAR method. Finally, we conclude with a summary of our work in Section 6.

2 Related Work

The primary obstacle in incremental learning is catastrophic forgetting, which occurs when a model experiences a significant decline in performance on previously learned tasks after learning new ones. In this section, we will provide a brief overview of incremental learning, which fall into three categories: regularization-based, memory-based and dynamic architecture-based methods.

2.1 Regularization-based Methods

Regularization based approaches add constraints to the model parameters or hyper-parameters during the training, aiming to consolidate the important weights for previously learned tasks. The constraint (i.e., regularization form), which minimizes changes to the consolidated weights, can be achieved in several ways. Elastic Weight Consolidation(EWC) [11] and Synaptic Intelligence (SI) [31] incorporate a penalty term into the loss function to constrain the changes made to the weights based on their importance to previously learned tasks. [11] measures the importance of the weights using the diagonal of the Fisher information matrix. And [31] estimates weights' importance with the help of three-dimensional synapses. Other than regularizing the loss function, [30] modifies the gradient during back-propagation to preserve the consolidated knowledge from previous tasks. [16] and [6] propose to regularize hyper-parameters such as learning rate and batch size to mitigate catastrophic forgetting.

2.2 Memory Based Methods

Memory based methods store data examples or other information from previous tasks in a memory buffer, then replay these memories along with the current task examples during training. These methods help the model retain knowledge learned from previous tasks while training on new tasks, effectively mitigating catastrophic forgetting. [19] proposes incremental Classifier and Representation Learning (iCARL), a classical method that separates representation learning and classification. For representation learning, iCARL preserves data from old tasks in a memory buffer and mixes them with current task examples during training. For classification, iCARL applies a nearest-mean-of-exemplars strategy to predict labels of test images. Another branch of works - [13], [5] and [24] - focus on optimizing the memory buffer to avoid overfitting. [13] proposes Gradient Episodic Memory(GEM) that develops an inequality that prevents the loss of previous

tasks from increasing. [5] trains a non-parametric classifier based on kernel ridge regression using data from the memory buffer. While [24] preserves the topology of the feature manifold of previous tasks in the memory buffer.

2.3 Architecture Based Methods

The methods above focus on solving the incremental learning tasks with a single model and a shared set of parameters. In contrast, architecture based methods adopt a dynamic architecture that allows for modifying neural networks and assigning different parameters for different learning tasks to prevent forgetting. The intuition behind this branch of works is that if we store the existing models and train a new model for the upcoming tasks, we are able to preserve the knowledge for previous tasks. Based on whether the model structure is fixed or not, architecture based methods can be roughly divided into two categories. [7] proposes PathNet, which freezes certain connections selected for previous tasks and assign other paths along the network for an upcoming new task. PathNet ensures that different subsets of the model parameters correspond to different tasks. [1] implements task-specific gating modules for each convolutional layer, selecting a limited set of kernels for each task. [21], [2] and [29] adopt dynamic model architectures where new modules are implanted so that the model can accommodate new tasks. In these works, distinct subnetworks are distributed to different tasks. And lateral connection enables the information exchange among these subnetworks.

3 Notations and Backgrounds

In this section, we will describe notations used throughout the paper and briefly explain the EMSTDP algorithm. This algorithm will serve as the foundation of the SEMINAR framework to update the synaptic weight of SNNs. The details of the proposed SEMINAR framework and its application in class incremental learning will be introduced in the following sections.

3.1 Problem Statement

In supervised learning, a model f parameterized by θ learns a task t from a corresponding training set with n_t samples. We denote the training set as $D_t = \{(x_i, y_i)_{i=1}^{n_t} | x_i \in X_t, y_i \in Y_t\}$, where n_t stands for the size of the training set, and X_t and Y_t stand for the data and label space of the training set. D_t follows a fixed distribution. In real-world settings, a single model may be required to learn multiple tasks with different distributions. After being exposed to N sets of training data, D_t, where $t \in [1, N]$ corresponding to N tasks with different distributions in sequential order; the goal is to have f_θ to remember all of them at the end. Without loss of generality, we assume that each task is a classification problem and we measure the performance using the average accuracy of all learned tasks as defined in Equation 1, where $a_{N,j}$ denotes the test accuracy of task j after learning task N.

$$A_N = \frac{1}{N} \sum_j a_{N,j} \tag{1}$$

3.2 SNN Model and EMSTDP Learning

We consider the incremental learning of multi-layer feedforward SNNs in this work. The neurons adopt the integrate and fire model, which is defined by the following equations:

$$u_i^l(t) = \begin{cases} 0, & \text{if } s_i^l(t-1) = 1 \\ \sum_j w_{ji} \cdot s_j^{l-1}(t) + u_i^l(t-1) + b_i^l & \text{otherwise} \end{cases} \tag{2}$$

$$s_i^l(t) = U(u_i^l(t) - V_{th}) = \begin{cases} 1, & \text{if } u_i^l(t) > V_{th} \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

Here, $u_i^l(t)$ represents the membrane potential of the ith neuron in layer l at time t, while $s_j^{l-1}(t)$ denotes the output of jth neuron in layer $l-1$ at time t, which also serves as the input of the neurons in the lth layer. The function $U()$ is the Heaviside activation function as defined in Equation 3. When the membrane potential $u_i^l(t)$ exceeds the threshold V_{th}, a spike is generated. The synaptic weight between the pre-synaptic neuron j in layer $l-1$ and the post-synaptic neuron i in layer l is denoted as w_{ji}, and b_i^l represents the bias. Once an output spike is generated in time $t-1$, the membrane potential $u_i^l(t)$ will be set to 0 in the next time step.

To facilitate the implementation on neuromorphic computing hardware, we adopt EMSTDP [23] as the learning rule of SEMINAR. An SNN with EMSTDP learning needs two networks: a feedforward network for inference and learning, and a feedback network for error (i.e., gradients) propagation. We use feedback alignment so that the two networks do not need to maintain the same set of weights A set of random weights, **B**, is used in the feedback network. The error propagated in the feedback network is calculated as follows: $e_i^l = \sum_j e_j^{l+1} B_{i,j} h_i'(u_i^l)$, 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. Since the Heaviside step activation is not differentiable, gradient surrogation must be adopted. Because the relationship between the number of output spikes and the neuron's accumulated sub-membrane potential can be approximated as a shifted ReLU function, EMSTDP replaces h_i' with the following equation:

$$h_i'(u_i^l) = \begin{cases} 1, & u_i^l(t) > V_{th} \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

EMSTDP operates in two phases: the free running phase and the error correction phase. In the free-running phase, the feedforward network predicts an output, h_j^l, based on the input received. In the error correction phase, errors propagated in the feedback network are fed into the feedforward network and "correcting" the prediction to a new value, $\hat{h}_j^l$.

The weight update of a synapse is calculated as follows: $\Delta w_j i = \eta * (\hat{h_j^l} - h_j^l) * h_i^{l-1}$, where η is the learning rate.

4 SEnsitivity Modulated Importance Networking and Rehearsal

4.1 Motivations

Taking inspiration from [9, 10], we have adopted a strategy of selecting and preserving important nodes in the hidden layers by freezing their synapses. This allows us to retain crucial knowledge of learned tasks. Specifically, we identify the neurons with the highest activation and designate them as important nodes if their collective activation accounts for a significant proportion of the total activation of all neurons in the same layer, based on a given threshold. We denote the set of important nodes as V_I. During training, the synapses between important nodes are frozen, while all other synapses remain plastic. The synapses between important nodes and all other neurons are referred to as important synapses, while the rest are referred to as plastic synapses.

Because the activation value of spiking neurons is bounded by the spiking rate, which ranges from 0 to 1, we have observed that neurons in SNN typically have a more balanced and symmetrical activation distribution than those in ANN. An example of the two distributions is shown in Figure 2. However, this creates a potential issue for SNN incremental learning. To maintain the same activation level, more important nodes must be selected for SNN than for ANN. As the number of tasks increases, new important nodes will be selected with their connections frozen. The size of important nodes and synapses will grow rapidly, and soon there will not be enough plastic synapses to learn new tasks. Another problem with frozen important nodes and synapses comes from potential knowledge transfer between tasks. After the training of a new task, some knowledge shared by both new and old tasks may be learned by plastic synapses, making the previously selected important nodes and synapses no longer critical. In this work, we propose a downgrade mechanism. After a new task is trained, the important neurons and synapses will be re-evaluated. Figure 1 illustrates the process of selecting and re-evaluating important nodes and synapses.

4.2 Overall Process

In this section, we introduce how SEMINAR handles incremental learning when a model f_θ is presented with a new task D_t with N_t training samples given an exemplar memory M_t. The SEMINAR uses a differential learning rate. The learning rates for the new task and the exemplars are denoted as η and $\eta_{\mathcal{M}}$ respectively, with $\eta_{\mathcal{M}} > \eta$. The differential learning rate is adopted to calibrate the gradient directions between new and previously learned tasks. Intuitively, the model f_θ will be exposed to more training data from the new task than from the exemplars. Therefore, the learning process of the new task needs to be guided with a lower learning rate. When training on the old exemplars, the model is expected to "drag" itself to the parameter subspace suitable for old tasks with fewer training data. Thus, a higher learning rate should be used.

Algorithm 1 Training one task with exemplar memory

Require: $\mathcal{M}_t$: exemplar memory, D_t:current task data $D_t = \{(x_i, y_i) \,|0 \le i \le N_t, 0 \le j \le N_t\}$, $(V_{I,t-1}, \theta_{I,t-1})$: set of important neurons and synapses after training task $t-1$.
Require: $\eta_{\mathcal{M}}$: learning rate for exemplar $\mathcal{M}_t$, η: learning rate for D_t , ζ: interleave hyperparameter.
$i \leftarrow 0$
while $i \neq N_t$ **do**
  if $i \mod \zeta == 0$ **then**
    $\mathcal{E} \leftarrow Balanced\_Sample(\mathcal{M}_t)$.
    for each $(x, y) \in \mathcal{E}$ **do**
      $\theta \leftarrow$ Masked_Training $(\theta$, x, y, $\theta_{I,t-t}, \eta_{\mathcal{M}})$.
    end for
  end if
$\theta \leftarrow$ Masked_Training $(\theta, x_i, y_i, \theta_{I,t-1}, \eta)$.
$i \leftarrow i + 1$
end while
$(V_{I,t}, \theta_{I,t}) \leftarrow$ Select_Important_Nodes(D_t).
$(V_{I,t}, \theta_{I,t}) \leftarrow$ Re-evaluate_Important_Nodes$(\mathcal{M}_t)$.
$Update\ \mathcal{M}_t$ and partial reset f_θ.

The overall flow of the training is given in algorithm 1. For every ζ training sample learned in current task t, we use function *Balanced_Sample* to select a subset ζ from exemplar memory M_t and apply EMSTDP to train f_θ on ζ with learning rate $\eta_{\mathcal{M}}$. How balanced sample is performed and how the learning rate is scheduled will be discussed later in section 4. For both ζ and D_t We update the weight parameters θ using the function $masked\_training()$ to prevent changing the frozen synapses in $\theta_{I,t-1}$. After that, we select the important nodes $V_{I,t}$ and regenerate the mask θ_{mask}. Finally, we update our exemplar memory M_t and reset the unimportant synapses whose weight is below the threshold.

4.3 Select Important Nodes

To select important nodes in every layer, we sort the activation values of validation samples as [10]. The validation samples are random samples drawn from the current training data. The total activation of a neuron i with N_s samples in a time window T is described in Equation 5, where s is the spiking function, $u_i^t(x)$ represents the membrane potential of a sample x of neuron i at timestamp t. Similarly, the total activation value A_l is defined in 6.

$$S(i) = \sum_{t \in T} \sum_{x \in N_s} s(u_i^t(x)) \tag{5}$$

$$A_l = \sum_{i \in l} S(i) \tag{6}$$

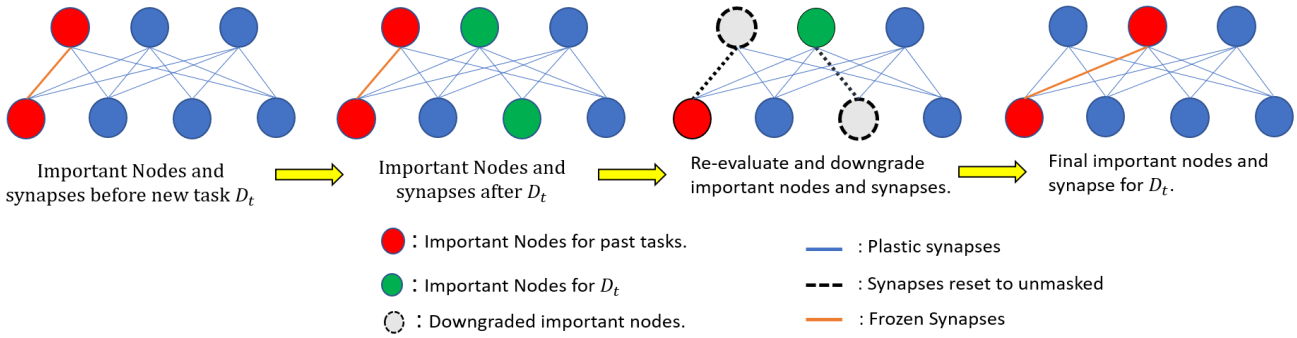

Figure 1. Selection and re-evaluation of important nodes and synapses

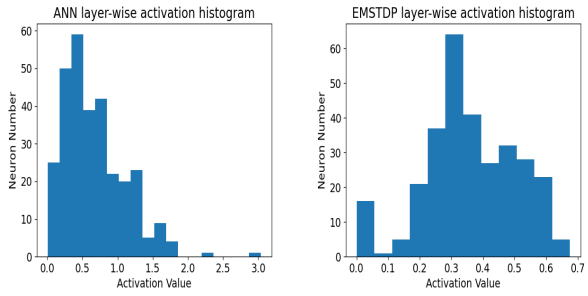

Figure 2. One Layer Activation Distribution of ANN and EMSTDP

In each layer, the important neurons are those neurons with the highest activities. Their activities can be used to represent the behavior of the entire layer. Therefore, we sort neurons based on the ascending order of their activation values and select the top k neurons such that $\sum_{i \in k} S(i) > \tau A_l$, where τ is a constant representing the portion of the important neurons' activation in the total activation. We denote those selected neurons as V_t. The set of overall important nodes $V_{I,t}$ for task 1 to t is the union of the V_t and $V_{I,t-1}$, $V_{I,t} = V_t \cup V_{I,t-1}$. The synapses between two important neurons are import synapses, whose weight will be crystallized, i.e., $\theta_{I,t} = \{\theta_{ij}\}, v_i, v_j \in V_{I,t}$.

4.4 Re-evaluate Important Nodes

The importance of neurons is not fixed. SEMINAR keeps the set of important nodes dynamic during each learning episode to make sure that there is always a sufficient number of plastic synapses as tasks increase. Following the concept of synaptic intelligence [31], we define the plasticity of neuron importance as the trajectory of weight changes by comparing the current model θ_t with a hypothetical model θ_h. The hypothetical model is trained with samples from exemplar memory $\mathcal{M}_t$ without freezing the important synapses. It represents a model that fine tuned for the past tasks. Given the current model f_{θ_t} and the hypothetical model f_{θ_t}, for each synapse θ_{ij}, the weight change $\Delta\theta_{ij}$ is calculated as the following, $\Delta\theta_{ij} = |\theta_{ij,t} - \theta_{ij,h}|$. For a neuron i in lth layer, the accumulated weight difference is calculated as Equation 7:

$$\Delta w_i = \sum_{j \in V^{l+1}} \Delta\theta_{ij} + \sum_{k \in V^{l-1}} \Delta\theta_{ki} \tag{7}$$

A significant accumulated weight difference indicates that adding task t affected its performance in previous tasks, and its (input or output) synapses need to be re-trained. Using Δw_i, the top ϵ percent of important neurons with the most drastically changed total weights will be downgraded to unimportant neurons. In our experiments, we gradually increase ϵ from 30% to 50% to guarantee the maximum percentage of important nodes is 25% for each hidden layer.

4.5 Masked Training

We apply EMSTDP to train the model f_θ on the given training data. It is believed that the important synapses in the set $\theta_{I,t-1}$ carry the knowledge that is critical for learned tasks (i.e., from task 1 to task $t-1$), we would like to keep them unmodified to avoid catastrophic forgetting. A binary mask is applied to the plasticity of the important synapses such that their value will not be updated by the EMSTDP algorithm.

4.6 Balanced Sample and Memory Interleave

The Balanced_Sample function ensures that the sampled exemplars $\mathcal{E}$ has a balanced distribution over all trained tasks, i.e., tasks 1 to $t-1$. Following an approach similar to that in [18], we sample an equal number of examples for each previous task from the exemplar memory.

The sampling (and training) of exemplar memory is interleaved with the training of the new task. Previous works that apply experience replay in incremental learning will attach memory exemplars to each batch. However, online learning on neuromorphic hardware does use batch training. In real-world continual learning scenarios, the data arrives one-by-one sequentially. Training all exemplars before the new task or training all of them after the new task is not a

good solution, as both of them will introduce inductive bias toward either the current or past tasks. Neither should we iteratively train the limited number of exemplars too many times, as it will lead to over fitting [28]. The interleaving of the training between the exemplars and the new tasks is introduced to address this challenge. Specifically, we sample N_m memory exemplars every time after training ζ samples of the new task, $\zeta = 2N_m$.

4.7 Partial Reset Weights

To increase the sparsity of the network, which helps continual learning [9]. We randomly prune the model by resetting any synapse with weights less than 0.01 to zero, if it is not an important synapse.

5 Experiments

Table 1. Comparison of Testing Accuracy

Dataset	S-MNIST	S-NMNIST	S-FMNIST	MSTAR
SEMNIAR	**0.861**	**0.825**	**0.735**	**0.747**
SI	0.197	0.195	0.182	0.325
EWC	0.196	0.196	0.185	0.321
ER	0.757	0.725	0.624	0.692
NISPA-R	0.757	0.725	0.624	0.692
AGS-CL	0.514	0.466	0.431	0.536
Oracle	0.971	0.954	0.879	0.795

Table 2. Memory Interleaving

Dataset	SEMINAR	MI	SE+ER	ER
Split-MNIST	**0.861**	0.791	0.795	0.755
Split-NMNIST	**0.825**	0.773	0.789	0.725
Split-Fashion MNIST	**0.735**	0.678	0.668	0.624
Mstar	**0.747**	0.710	0.721	0.692

In this section, we implement SEMINAR with other state-of-the-art continual learning methods. For a fair comparison, all of them use the EMSTDP learning rule to facilitate their implementation on neuromorphic hardware. We tested four vision datasets: split-MNIST [12], split-NMNIST [17], MSTAR [20], and split-Fashion MNIST [27]. The split-MNIST dataset divides the hand-written digits dataset MNIST into five tasks, each task classifies two classes of digits. The Fashion-MNIST is a more complex dataset that comprises 28 × 28 grayscale images of 70, 000 fashion products from 10 categories, we also split it into five classification tasks similar to split-MNIST. In addition, we conducted experiments on the neuromorphic dataset N-MNIST, which is produced by the DVS camera. We also split the N-MNIST dataset into five classification tasks, each with two digits. The last dataset is the MSTAR dataset, MSTAR dataset is a variant of synthetic aperture radar (SAR) imagery, we chose 6 classes of different tanks and split them into 3 tasks. For all datasets, we use a feedforward SNN with 2 hidden layers, each hidden layer has 300 neurons. We set $\eta_{\mathcal{M}_t}$ to 0.00025 and η to 0.00015, respectively.

5.1 Comparing with Baseline Algorithms

Five baseline incremental learning algorithms were implemented: SI [31], AGS-CL [10], EWC [11], ER [3], and NISPA-Replay [9]. SI and EWC are regularization-based methods as introduced in Section 2. ER is a memory-based method. It appends exemplars to every batch of training data and uses the reservoir sampling technique to construct episodic memory. NISPA is an architecture-based method, that also leverages the idea of important nodes and synapses and the important synapses are masked during the training. However, the important nodes and synapses in NISPA will never be downgraded. AGS-CL is similar to NISPA except that it allows the important synapses to be trained while using a regularization to limit the change of the important synapses to the minimum.

The comparison results of SEMINAR and baseline algorithms are shown in Table 1, term "Oracle" means no incremental learning. Figure 3 shows how the testing accuracy changes after training each task. We can see that SEMINAR outperforms the compared methods on all the datasets. Compared to the second-best algorithm, it improves the testing accuracy by 4.0% to 5.9%. Specifically, our method yields 86.1%, 82.5%, and 73.5% for the MNIST variant datasets, which is better than all other methods by more than 5% percent. For the MSTAR dataset, our method improves by 4.2% over the best baseline method.

5.2 Ablation Studies

We further conducted ablation studies to evaluate the effectiveness of unique techniques adopted by SEMINAR, and analyze our design choices.

5.2.1 Exemplar memory Size. To evaluate the performance of SEMINAR with varying exemplar memory budgets, we conducted experiments using different sizes of exemplar memory ranging from 50 to 150 samples per class. We compared our methods with two memory-based techniques: ER [3] and NISPA-Replay [9]. From Table 3. The results show that SEMINAR continuously performs the best under different memory sizes.

5.2.2 Memory Interleave. Table 2 showcases the effectiveness of our memory interleave technique. Two simplified versions of SEMINAR were created. In the first one, we strip out the *Masked_Training*$(\cdot)$ function from SEMINAR and leave only the experience replay with interleaved memory. The simplified version is called MI. In the second version, we remove the memory interleave, and append the entire

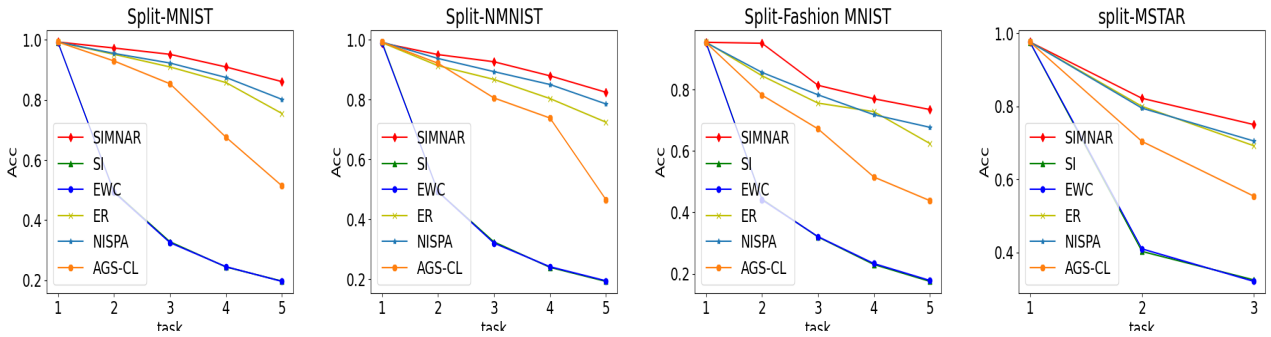

Figure 3. Average Test Accuracies after training each task.

Table 3. Different Exemplar Size

	50 Samples				100 Samples				150 Samples			
	MNSIT	N-MNIST	F-MNIST	MSTAR	MNSIT	N-MNIST	F-MNIST	MSTAR	MNSIT	N-MNIST	F-MNIST	MSTAR
Oracle	0.971	0.954	0.879	0.795	0.971	0.954	0.879	0.795	0.971	0.954	0.879	0.795
SEMINAR	0.705	0.657	0.603	0.554	0.794	0.743	0.703	0.651	0.861	0.825	0.735	0.747
ER-EMSTDP	0.615	0.632	0.507	0.543	0.733	0.694	0.627	0.618	0.757	0.725	0.623	0.692
NISPA-EMSTDP	0.676	0.639	0.583	0.571	0.758	0.724	0.631	0.648	0.802	0.786	0.678	0.705

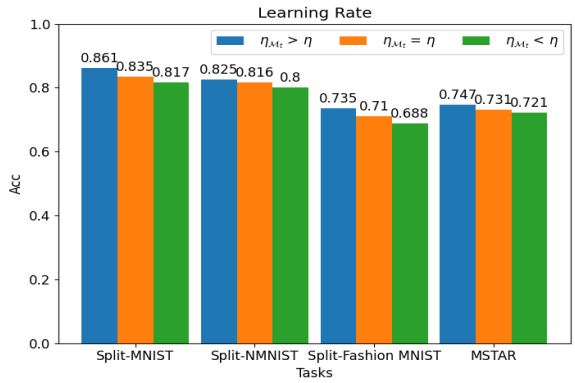

Figure 4. Impact of Differential Learning Rate

Portion	MNIST	N-MNIST	F-MNIST	MSTAR
5%	0.793	0.775	0.681	0.683
10%	0.801	0.792	0.715	0.697
25%	0.861	0.801	0.718	0.717
40%	0.811	0.773	0.653	0.673

Table 4. Portion Of Important Nodes

exemplar memory to every batch of ζ new training data. This version is called SE+ER because this is how ER [3] performs experience replay. The results show that SEMINAR outperforms the SE+ER by 2.6% to 6.6% for all four datasets. Even without important nodes and masked training, the MI outperforms ER by 1.8% to 5.4%. Note that the MI performs experience replay with fewer examples. For each batch of the ζ new data, only N_m exemplars are sampled from the $\mathcal{M}_t$ by MI, while ER attaches the entire $\mathcal{M}_t$ to the new data.

5.2.3 Portions of Important Neurons. Experiments were conducted with varying numbers of important nodes. Specifically, we vary the percentage of important nodes (i.e., τ) from 40% to 10%. The comparison results are given in Table 4. The results demonstrate that the size of the important neurons affects the performance of SEMINAR. With too few important neurons, the model has difficulty to remember old tasks. On the other hand, too many important neurons will prevent the model to learn new tasks. We found that the peak performance is achieved when 25% of neurons are identified as important.

5.2.4 Differential Learning Rate. Figure 4 compares different configurations of the relative learning rate for exemplars and new tasks. We can see that using a higher learning rate for exemplar memory ($\eta_m = 0.00015$) and a lower learning rate on the new task ($\eta = 0.0001$) consistently results in better performance than setting them equally or reversely.

6 Conclusion

This paper presents SEMINAR, an incremental learning algorithm that could work with the EMSTDP learning rule for neuromorphic implementation. Its performance is compared with existing incremental learning algorithms using regularization-based, memory-based, and dynamic architecture-based techniques. Our experimental results show that SEMINAR works the best in the spike domain learning scenario.

References

[1] Davide Abati, Jakub Tomczak, Tijmen Blankevoort, Simone Calderara, Rita Cucchiara, and Babak Ehteshami Bejnordi. 2020. Conditional Channel Gated Networks for Task-Aware Continual Learning. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2020), 3930–3939. https://doi.org/10.1109/CVPR42600.2020.00399 arXiv:2004.00070

[2] Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. 2017. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 3366–3375.

[3] Arslan Chaudhry, Marcus Rohrbach, Mohamed Elhoseiny, Thalaiyasingam Ajanthan, Puneet K Dokania, Philip HS Torr, and Marc'Aurelio Ranzato. 2019. On tiny episodic memories in continual learning. *arXiv preprint arXiv:1902.10486* (2019).

[4] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. 2021. Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook. *Proc. IEEE* 109, 5 (2021), 911–934. https://doi.org/10.1109/JPROC.2021.3067593

[5] Mohammad Mahdi Derakhshani, Xiantong Zhen, Ling Shao, and Cees Snoek. 2021. Kernel continual learning. In *International Conference on Machine Learning*. PMLR, 2621–2631.

[6] Sayna Ebrahimi, Mohamed Elhoseiny, Trevor Darrell, and Marcus Rohrbach. 2019. Uncertainty-guided continual learning with bayesian neural networks. *arXiv preprint arXiv:1906.02425* (2019).

[7] Chrisantha Fernando, Dylan Banarse, Charles Blundell, Yori Zwols, David Ha, Andrei A Rusu, Alexander Pritzel, and Daan Wierstra. 2017. Pathnet: Evolution channels gradient descent in super neural networks. *arXiv preprint arXiv:1701.08734* (2017).

[8] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. 2013. An Empirical Investigation of Catastrophic Forgetting in Gradient-Based Neural Networks. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings* (dec 2013). arXiv:1312.6211 https://arxiv.org/abs/1312.6211v3

[9] Mustafa Burak Gurbuz and Constantine Dovrolis. 2022. Nispa: Neuro-inspired stability-plasticity adaptation for continual learning in sparse networks. *arXiv preprint arXiv:2206.09117* (2022).

[10] Sangwon Jung, Hongjoon Ahn, Sungmin Cha, and Taesup Moon. 2020. Continual learning with node-importance based adaptive group sparse regularization. *Advances in Neural Information Processing Systems* 33 (2020), 3647–3658.

[11] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences* 114, 13 (2017), 3521–3526.

[12] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[13] David Lopez-Paz and Marc'Aurelio Ranzato. 2017. Gradient episodic memory for continual learning. *Advances in neural information processing systems* 30 (2017).

[14] Marc Masana, Xialei Liu, Bartłomiej Twardowski, Mikel Menta, Andrew D. Bagdanov, and Joost van de Weijer. 2023. Class-Incremental Learning: Survey and Performance Evaluation on Image Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45, 5 (2023), 5513–5533. https://doi.org/10.1109/TPAMI.2022.3213473

[15] Michael McCloskey and Neal J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychology of Learning and Motivation - Advances in Research and Theory* 24, C (jan 1989), 109–165. https://doi.org/10.1016/S0079-7421(08)60536-8

[16] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. 2020. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems* 33 (2020), 7308–7320.

[17] Garrick Orchard, Ajinkya Jayawant, Gregory K Cohen, and Nitish Thakor. 2015. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers in neuroscience* 9 (2015), 437.

[18] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. 2020. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*. Springer, 524–540.

[19] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H Lampert. 2017. icarl: Incremental classifier and representation learning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2001–2010.

[20] Timothy D Ross, Steven W Worrell, Vincent J Velten, John C Mossing, and Michael Lee Bryant. 1998. Standard SAR ATR evaluation experiments using the MSTAR public release data set. In *Algorithms for Synthetic Aperture Radar Imagery V*, Vol. 3370. SPIE, 566–573.

[21] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. 2016. Progressive neural networks. *arXiv preprint arXiv:1606.04671* (2016).

[22] Amar Shrestha, Haowen Fang, Daniel Patrick Rider, Zaidao Mei, and Qinru Qiu. 2021. In-hardware learning of multilayer spiking neural networks on a neuromorphic processor. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 367–372.

[23] Amar Shrestha, Haowen Fang, Qing Wu, and Qinru Qiu. 2019. Approximating back-propagation for a biologically plausible local learning rule in spiking neural networks. In *Proceedings of the International Conference on Neuromorphic Systems*. 1–8.

[24] Xiaoyu Tao, Xiaopeng Hong, Xinyuan Chang, Songlin Dong, Xing Wei, and Yihong Gong. 2020. Few-shot class-incremental learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12183–12192.

[25] Neil C. Thompson, Kristjan Greenewald, Keeheon Lee, and Gabriel F. Manso. 2021. Deep Learning's Diminishing Returns: The Cost of Improvement is Becoming Unsustainable. *IEEE Spectrum* 58, 10 (oct 2021), 50–55. https://doi.org/10.1109/MSPEC.2021.9563954

[26] Sebastian Thrun. 1995. Is Learning The n-th Thing Any Easier Than Learning The First?. In *Advances in Neural Information Processing Systems*, D. Touretzky, M.C. Mozer, and M. Hasselmo (Eds.), Vol. 8. MIT Press. https://proceedings.neurips.cc/paper_files/paper/1995/file/bdb106a0560c4e46ccc488ef010af787-Paper.pdf

[27] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[28] Xue Ying. 2019. An overview of overfitting and its solutions. In *Journal of physics: Conference series*, Vol. 1168. IOP Publishing, 022022.

[29] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. 2017. Lifelong learning with dynamically expandable networks. *arXiv preprint arXiv:1708.01547* (2017).

[30] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. 2019. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence* 1, 8 (2019), 364–372.

[31] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *International conference on machine learning*. PMLR, 3987–3995.