NEO: Neuron State Dependent Mechanisms for Efficient Continual Learning

Anurag Daramt
anurag.daram@utsa.edu
Neuromorphic AI Laboratory
University of Texas at San Antonio
Texas, USA

Dhireesha Kudithipudi
dk@utsa.edu
Neuromorphic AI Laboratory
University of Texas at San Antonio
Texas, USA

ABSTRACT

Continual learning (sequential learning of tasks) is challenging for deep neural networks, mainly because of *catastrophic forgetting*, the tendency for accuracy on previously trained tasks to drop when new tasks are learned. Although several biologically-inspired techniques have been proposed for mitigating catastrophic forgetting, they typically require additional memory and/or computational overhead. Here, we propose a novel regularization approach that combines neuronal activation-based importance measurement with neuron state-dependent learning mechanisms to alleviate catastrophic forgetting in both task-aware and task-agnostic scenarios. We introduce a neuronal state-dependent mechanism driven by neuronal activity traces and selective learning rules, with storage requirements for regularization parameters that grow slower with network size - compared to schemes that calculate weight importance, whose storage grows quadratically. The proposed model, NEO, is able to achieve performance comparable to other state-of-the-art regularization based approaches to catastrophic forgetting, while operating with a reduced memory overhead.

KEYWORDS

catastrophic forgetting, task agnostic, Task Incremental learning, Domain Incremental learning, Neuron Importance

ACM Reference Format:
Anurag Daramt and Dhireesha Kudithipudi. 2023. NEO: Neuron State Dependent Mechanisms for Efficient Continual Learning. In *Neuro-Inspired Computational Elements Conference (NICE 2023), April 11–14, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3584954.3584960

1 INTRODUCTION

Continual learning refers to an area of study that focuses on how artificial neural networks (ANNs) can learn tasks sequentially from a continual stream of data. Continual learning algorithms aim to address a fundamental trade-off: the *stability-plasticity dilemma* whereby a model that emphasizes stability tends to suffer from poor forward transfer and adaptation to new tasks, whereas one that is too plastic is unable to retain previously learned information, a phenomenon commonly known as catastrophic forgetting or interference [24]. Biological brains seem to have solved this dilemma, being able to learn continuously throughout their lifetime. Taking inspiration from neuroscience, researchers have proposed several types of mechanisms to address catastrophic forgetting [6, 16]. These can broadly be classified into (i) activity-dependent parametric adjustments or regularization [12, 28, 31, 37], (ii) dynamic architectures, including neurogenesis. [7, 20, 25–27], and (iii) rehearsal or replay of previous experiences [4, 23, 30, 32]. In this paper, we focus on regularization-based approaches (also known as metaplasticity-based [1]), which accommodate new learning without expanding the network. Regularization methods work by identifying network parameters that are important for previously learned tasks and restricting modification of those parameters when learning new tasks.

Typically, regularization-based methods implement a per-synapse parameter that reflects each synapse's importance for each task. Examples include the diagonal Fisher information matrix used by EWC [13], path integrals of gradient vectors in SI [37], Bernoulli transmission probabilites and magnitudes in Stochastic Synapses [28], a variance term for each synapse in VCL [38], and hidden metaplastic weights in Binary Metaplastic BNN [18]. While these methods have proved effective in addressing catastrophic forgetting, the need to store regularization parameters can increase a model's memory requirements by a factor of two or more, which can be prohibitive for large models.

To avoid this problem, we here introduce a novel metaplasticity approach (NEO) that is driven solely by neuron activations, without requiring additional storage of per-synapse regularization parameters for every synapse. Utilizing neuron-importance-based regularization reduces the space complexity from $O(n^2)$ to $O(n)$, where n represents the number of neurons per layer in the network. In NEO, each neuron's importance for previously learned material is gauged based on its activity, and used to regularize synapses and update the learning mechanism. This method does not require additional storage of per-synapse regularization parameters for every synapse, thereby reducing the space complexity from direct quadratic growth. There have been other attempts to implement regularization based on neuron importance [10, 11]; however these were able to operate only in scenarios where task-identifying information was provided to the network.

The proposed approach, NEO introduces a neuronal state variable that depends on each neuron's activity history and is used to selectively regularize, rescale, or prune synapses to alleviate forgetting in both task-aware and task-agnostic scenarios.

NICE 2023, April 11–14, 2023, San Antonio, TX, USA

ACM ISBN 978-1-4503-9947-0/23/04.
https://doi.org/10.1145/3584954.3584960

For evaluation of NEO, we use both the task-incremental learning (Task-IL) and the domain-incremental learning (Domain-IL) scenarios for sequential learning. In both scenarios, the probability distribution of inputs differs between tasks, but the distribution of target labels does not change. They differ in that Domain-IL requires that no task-identity or task-boundary information be provided during training or inference [9, 33, 34].

1.1 Related Work

Regularization approaches. Techniques like EWC [12] and SI [37] select important parameters by using a Fisher Information matrix or tracking synapses' credit in improvement on a task, respectively. Other models [5, 14, 21] select parameters that preserve the distribution of latent representations for each task. Laborieux et al.[18] applied a version of metaplasticity to a binary neural network model, making synapses with weights of greater magnitude less plastic, thus protecting them from modification by subsequent training. Steger et al. [28] used stochastic weights with Bernoulli transmission probabilities and altered the learning rate of important weights to preserve information.

Importance measurement. Importance-based regularization methods may be classified according to which entities they attribute importance to (neurons or weights) and how they measure importance. Kim and Lee [11] identify three categories: 1) Weight Importance measurement using weight statistics, 2) Neuronal importance measurement using weight statistics and 3) Neuronal importance measurement using neuronal behavior/statistics. A feature that is common to the previously mentioned regularization approaches is the calculation of weight importance metrics using weight statistics. In the second of these regularization approaches, neuronal importance is measured based on presynaptic weight statistics. Rather than maintaining an importance parameter for each weight, a single importance value is calculated for each neuron and used to regulate the learning of all of its incoming weights.

In *Uncertainty-regularized Continual Learning* [2], neuronal importance is measured by means of an 'uncertainty' factor that is computed from the variability during training of each neuron's incoming weights. The idea is that weights that are important for a task tend to vary less during training, thus a neuron's importance for a task can be measured based on the stability of its incoming weights during training of that task.

An alternative approach to identifying important neurons involves using only neuronal activation levels and firing dynamics. The work presented in [10] utilizes average activation level to determine the importance of neurons and further exploits proximal gradient descent to perform regularized updates. After calculating the neurons' importance values, incoming weights to important neurons are frozen and outgoing weights from unimportant neurons are pruned.

Neuronal-importance-based regularization approaches have been shown to achieve near-state-of-the-art performance on image classification tasks, but only in scenarios where task identity was known. *By contrast, weight-importance-based approaches have demonstrated superior performance even in task-agnostic settings, specifically in the domain-incremental learning (Domain-IL) scenario.*

1.2 Main contribution

We introduce a novel neuronal-state-based regularization approach that eliminates the need for storing per-synapse regularization parameters. The state of each neuron is calculated based on its cumulative average activation level, and each synaptic weight is updated using one of a set of update rules, selected based on the states of the synapse's presynaptic and postsynaptic neurons.

Our model achieves performance on par with state-of-the-art weight-importance based regularization mechanisms. Unlike previous neuronal-importance-based regularization approaches, our model is able to learn in an online task-agnostic manner while demonstrating performance comparable to other task-agnostic regularization based mechanisms in the domain-incremental learning scenario. NEO is able to show these capabilities while operating within ~ 1.002-1.5x memory overhead.

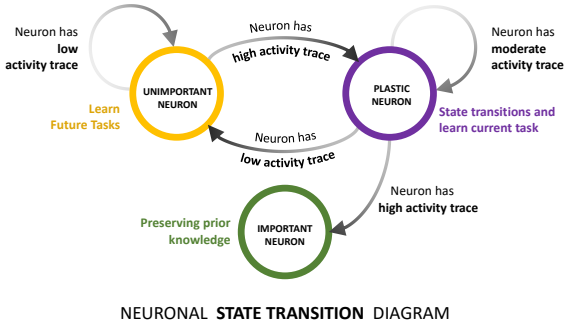


Figure 1: Neuron states and transitions in the proposed model. A neuron can be in any of three states, Unimportant, Plastic or Important. At every state transition interval, depending on trace activity level, it may transition from one state to the other.

2 METHODOLOGY

2.1 Problem Formulation and Notation

We formulate the continual learning problem $\mathcal{F}$ as the ability to learn tasks in a sequential manner, without suffering severe performance loss on previously learned tasks when new ones are learned. Formally, we consider a distribution of tasks $\mathcal{D} = \{T^1, T^2 ... T^N\}$ for $N \in \mathbb{Z}^+$, wherein each task T^k is a set $(\mathcal{X}_k, \mathcal{Y}_k)$ of ordered pairs of input data points and their corresponding class labels. Performance on the problem $\mathcal{F}$ is evaluated by first training the network on the tasks $\{T^i\}$ sequentially, then measuring the mean performance $\Psi(\mathcal{D}) \equiv \mu(\{\psi(T^1), \psi(T^2), ..., \psi(T^k)\})$, where $\psi(T^k)$ represents the network's test performance on task T^k. The goal of the network is to maximize $\Psi(\mathcal{D})$ after learning task T^k.

In this manuscript, we denote a^n to represent the activation and θ_t^n to represent the activity trace of neuron n at sample t represented by (x_t, y_t). The weights are represented using w_{ij}, wherein i represents the pre synaptic neuron and j represents the post synaptic neuron. The net activity of a layer is represented by A^l, where l

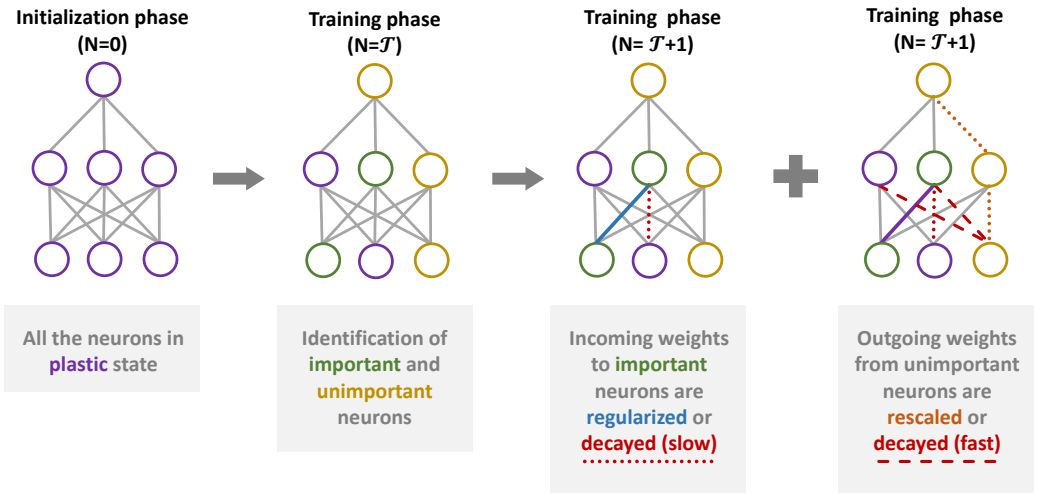


Figure 2: The network training phase with the neuronal state update and the policies applied after the states have been identified. After every state update interval $\mathcal{T}$, the state of the neurons are identified. Then based on the state, the connections are either regularized, decayed, rescaled or remain unchanged.

represents the layers in the network. τ is used to denote the activation thresholds in the network and the state update interval is represented by $\mathcal{T}$.

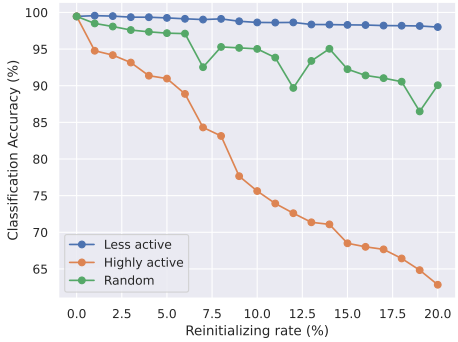


Figure 3: The correlation between the classification performance and the active neurons in an MLP trained on MNIST. When the connections to the highly active neurons are reinitialized, a significant drop in performance is observed, whereas reinitializing the connections from least active neurons barely affects the performance.

2.2 Neuronal States and Importance

In the proposed model, neurons have a state variable that may take any of the three values, *Unimportant*, *Plastic* or *Important*, see figure 1. All neurons are initialized with the state set to *Plastic*, in which new information can be learned by updating adjacent synapses as needed. The state of each neuron is computed by keeping track of its activation value during training. The activation trace (θ^n) is computed as the rolling average of the neuron's activation value (a^n) using Welford's online algorithm [35].

$$\theta_t^n = \theta_{t-1}^n + \frac{(a^n - \theta_{t-1}^n)}{t} \tag{1}$$

where t represents the sample number being presented to the network. At every state update interval, state transitions can take place. The trace value is compared to two thresholds (τ^{up} and τ^{low}) to determine whether a neuron can transition to the *Unimportant*, *Plastic*, or *Important* state:

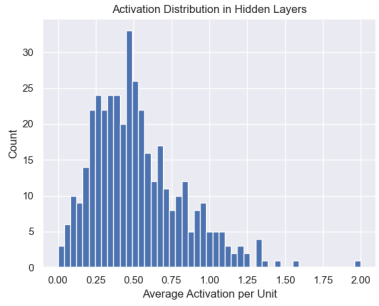
$$Neuron\_state = \begin{cases} \textbf{IMP}, & if\, \theta_t^n > \tau^{up} \\ \textbf{PLASTIC}, & if\, \tau^{low} \leq \theta_t^n \leq \tau^{up} \\ \textbf{UN-IMP}, & if\, \theta_t^n < \tau^{low} \end{cases} \tag{2}$$

The parameter θ^n represents the average activity of the neuron in response to the observed samples. The conditions presented in Equation 2, show that the highly active neurons are considered to be important with respect to the task being learned. Moreover, it is important to note that with reference to figure 1, once a neuron enters an *Important* state, the neuron does not shift states. Equation 2 applies to the plastic and unimportant neurons which can switch and change states.

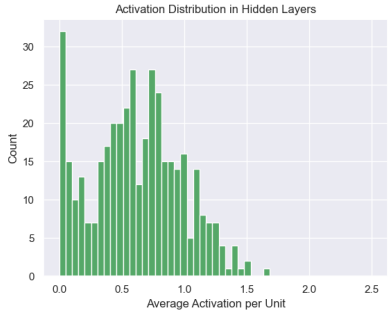
There are two ways of representing neuronal importance, namely importance based on the weight measurement and importance based on neuron activity measurement.

2.3 Why activation-based importance

An activation can be considered as the basic unit for representing the learned information from a task. The importance of the neurons can be determined by measuring the values of activation of the neuron in response to the inputs being sent in. Catastrophic forgetting occurs when the information received by the neurons changes while learning new tasks. To understand how activations can determine the importance of a task, we performed an analysis



(a) Distribution of per-unit average activations in layer 1



(b) Distribution of per-unit average activations in layer 2

Figure 4: The distribution of activations (ReLU) in a MLP with 2 hidden layers of size 400 each, trained on the MNIST dataset using backpropagation. The distribution is computed by taking an average of the activations over 1000 samples post training. It can be observed that the distributions are highly skewed.

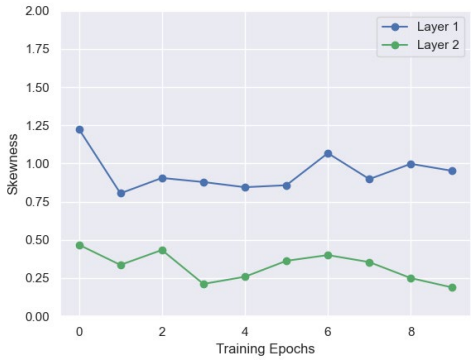


Figure 5: The skewness of the per-layer activation distribution in the hidden layers of an MLP trained on MNIST for 10 epochs. It can be seen the skewness of the activations does not change with respect to the training time.

to measure how important the activity of neurons correlates to the performance. Figure 3 shows that removal and reinitialization of neurons which were highly active (with high θ^n values) hurts the performance of the model in comparison to reinitialization of neurons with lower activity. Additionally, to determine how to select the important neurons from the rest of the activations, the skewness and the distribution of average activation response is measured. Figure 4 shows the distribution of per-unit average activation for 1000 samples of MNIST dataset. As can be observed, the distribution is heavily skewed in both the hidden layers, which demonstrates that there are few set of important neurons that could be used to represent the important features selective to a particular task. Here, the average activation is measured as a function of the state update interval and the activity over time. Every average activation value shown in figure 4 is computed as

$$\mathbf{d^l} = (\frac{1}{\mathcal{T}}) * \sum_{t=1}^{\mathcal{T}} \mathbf{a_t^l} \tag{3}$$

, where $\mathbf{a_t^l}$ represents the vector of activation values of a layer at sample t and $\mathbf{d^l}$ represents a vector of average activation distribution over the interval $\mathcal{T}$ which is set to 1000 in this case. Since figure 4 shows the distribution post training, the skewness of activations was tested while training, as shown in figure 5. The high skewness among the activations remains high throughout the training process. This can be attributed to the use of ReLU activation function which maps all negative pre-activations to 0. Moreover, prior studies have observed that using ReLU leads to highly skewed per layer activation distribution for different architectures [8, 17]. Therefore the skewness can be used to determine the τ^{up} and τ^{low} thresholds presented in Equation 2.

2.4 Neuronal state-based learning mechanisms

The works presented in [10, 11] either regularize or prune and reinitialize the entire set of outgoing or incoming weights connected to the neuron in reference. This does not encourage greater information transfer between tasks and shuts off regions of the network entirely while learning later tasks. Moreover, reinitalizing the connections entirely can affect the neuron activation response of the previously important and unimportant neurons therein leading to a loss in performance. To avoid the unstable behavior and sections of network being pruned off for the entire lifetime, a set of learning rules are derived based on the state of the neuron.

Based on the states of the neurons, there are 9 different types of possible connections between neurons across the layers. For some of the connections, additional mechanisms are required along with SGD weight updates to address catastrophic forgetting, and rest of the weights undergo standard SGD updates. Table 1 shows the mapping of the connections to the different learning mechanisms.

- **Important-> Important neuron** - The connections between important neurons are regularized and any change in the weight connecting those neurons is penalized. This enables greater retention of information for the newer tasks. The weight w_{ij}^l connecting i_{th} neuron in layer l to the j_{th} neuron in layer $l+1$ is regularized according to

$$\Delta w_{ij}^l * = \alpha * (\left|1 - \left(\frac{\theta_t^i}{\theta_{t-1}^i}\right)^2\right|) \tag{4}$$

,where θ_t^i represents the activation trace for the neuron in layer l (pre-synaptic neuron) and α is a learning rate parameter in the $\mathbb{R} \in [0.001, 0.1]$. Equation 4 represents that the neurons selected as important are penalized based on how high the activation trace is. A higher activity trace leads to the connection being regularized heavily (similar to freezing the connection). This rule enables the previous knowledge to be preserved among the important neurons.

- **Plastic -> Important neuron** - The connections from plastic to important neurons go through a slow decay process. While a neuron is plastic, either it can become active while learning earlier tasks or switch to unimportant state due to low activation response over time. However, these changes in states can in turn lower the activation response for the important neurons in the next layer. Therefore a slow decay process is applied to the connections wherein the outgoing connection from the plastic neuron despite of the sudden changes in later tasks attempts to keep the weight closer to the initial value for consolidation of previous knowledge. The weight from a plastic neuron i in layer l to an important neuron j in layer $l+1$ is updated according to

$$w_{ij} = w_{ij} + \underbrace{\beta * (w_{ij}^{ref} - w_{ij})}_{\text{Decay term}} - \underbrace{(1 - \frac{w_{ij}^{ref}}{w_{ij}}) * \beta^{'}}_{\text{Magnitude penalty}} \tag{5}$$

wherein the w_{ij}^{ref} represents the reference weight that was noted when neuron j in layer $l+1$ was identified as important. The decay term penalizes based on how much the weight has moved from the reference value post update. The magnitude penalty adds further decay to the weights with higher magnitude and active plastic neuron. β is the scaling parameter in $\mathbb{R} \in [0.001, 0.1]$ based on the type of connection. $\beta^{'}$ is a magnitude and activity rate parameter wherein $\beta^{'} = \theta_t^i * \beta$. It is important to note that the weight decay mechanism is applied post weight update using gradients.

- **Unimportant -> Important neuron** - These connections also undergo a similar decay mechanism wherein the dependency of the unimportant neuron in the activation behavior of the important neuron is reduced. This leads to an almost stable connection which is unaffected by the change in the state of the unimportant neuron while learning new tasks.

Connection Type	Learning Mechanism
IMP->IMP	Regularize (Eq 4)
Plastic->IMP	Slow decay (Eq 5)
UN-IMP->IMP	Fast decay (Eq 5,6)
UN-IMP->Plastic	Rescale/Re-init/Unchange (Eq 7)
UN-IMP->UN-IMP	Rescale/Re-init/Unchange (Eq 7)
Others	No additional rule

Table 1: Mapping of the different connection types and the learning mechanisms based on the neuronal states.

To enable this, the decay rate β specified in Equation 5 is increased to a higher rate. In this case w_{ij}^{ref} can be set to a fixed value in $\mathbb{R} \in [w_{min}, 0.1]$, therein avoiding the need to keep track of a reference weight. This ensures that the weight is decayed to the reference value despite of the changes in the activation trace of the unimportant neuron for future tasks. Therefore, for these neurons, $\beta^{'}$ is set to 0, completely removing the magnitude penalty term and incorporating higher ranges for β. One way to ensure a dynamic representation of β is by using the following:

$$\beta_n = \frac{1}{\mathcal{T}} * |A^l - \theta_t^n|^2 * \beta_{base} \tag{6}$$

where, A^l represents the average activation of the layer, $\mathcal{T}$ represents the state update interval, and θ_t^n represents the activity trace of the neuron of interest (pre-synaptic neurons i.e. unimportant neuron). β_{base} represents the base decay parameter in $\mathbb{R} \in [0.01, 0.1]$.

- **Unimportant->Plastic/Unimportant neuron** - The outgoing weights from the unimportant neurons can undergo either of the four different options. One option is to entirely prune the connections off [10]. This option can be realized by setting w^{ref} in Equation 5 to 0 and make the weight eventually decay to a value close to 0. However, this leads to neurons being pruned off and having an inefficient use of the capacity of the network. To avoid this, the other option was to prune and then reinitialize the weights of the unimportant neurons. Re-initialization allows the unimportant neurons to get the opportunity to become plastic/important for future tasks. Enhancing plasticity using reinitialization can lead to interference with the previously learned knowledge by either having shadow activations (duplicate neurons firing for the same input) or affecting the response of other important neurons. To address this, a third option is introduced which involves connection strength based rescaling between the unimportant neurons (upscaling the low values weights). Rescaling enhances network plasticity without interfering with the previous activity response of the network.

$$w_{ij} = w_{ij} * (1 + \eta * (1 - exp\Big(\frac{1}{w_{ij}}\Big))) \tag{7}$$

Equation 7 represents the scaling formula for the weights connecting unimportant neurons, where w_{ij} represents the outgoing weight from an unimportant neuron i in layer l to another unimportant/plastic neuron in layer $l+1$. The final option involves keeping the outgoing weights unchanged for unimportant neurons. An ablation study is performed and included in Section 3.

3 RESULTS AND ANALYSIS

3.1 Experimental Setup

We evaluate NEO on split image classification benchmarks using 5-Split MNIST [19], Fashion-MNIST [36] and CIFAR-10 [15]

datasets under both the task incremental and task-agnostic domain-incremental continual learning scenario [9, 33]. In the case of task-incremental learning (Task-IL), the model is aware of the task identity during training and inference whereas in domain-incremental learning (Domain-IL) scenario, tasks share the same output layer while the model is unaware of task identity during both training and inference. There were two sets of network configurations based on the datasets used. For the Split MNIST and Fashion-MNIST benchmark, the configuration was fixed to 400 neurons per hidden layer (2 hidden layers), and either a two-neuron output layer (Domain-IL) or two-neurons each per task (Task-IL). The network is trained using backpropagation with ReLU activation function for the neurons. For the baseline, we train using SGD a learning rate of $1e^{-3}$ and the network is trained for 5 epochs each on every task. For the Split CIFAR-10 benchmark, we use a pre-trained CNN front-end of ResNet-18 architecture with the latent space as input to the configuration of 2 hidden layers with 400 neurons at the output.

3.2 Continual learning performance in Domain-IL and Task-IL scenarios

We evaluated and compared the model's performance in both task-incremental and domain-incremental learning scenarios. For the Task-IL scenario, we evaluated NEO on Split-MNIST, FMNIST and CIFAR-10 benchmarks. For the Domain-IL scenario, we evaluated the models on Split MINST and FMNIST benchmarks respectively. Based on the results shown in Tables 3 and 2, NEO is able to achieve comparable performance as other weight and neuron measurement based regularization approaches.

Model	FMNIST (MA%)	MNIST (MA%)	CIFAR-10
Baseline	86.13 ± 2.64	84.32 ± 0.99	76.84 ± 1.05
EWC– [12]	97.53 ± 0.15	98.64 ± 0.22	85.78 ± 1.2
SI – [37]	97.00 ± 0.25	99.09± 0.15	85.61 ± 1.51
MAS– [3]	97.43 ± 0.14	99.71 ± 0.02	82.13 ± 1.8
BGD– [39]	98.32 ± 0.12	99.47 ± 0.18	85.57 ± 3.89
UCL– [10]	98.10 ± 0.07	99.32 ± 0.05	86.72 ± 1.65
NAI– [11]	97.84 ± 0.06	99.6 ± 0.12	87.14 ± 0.68
NEO	97.22 ± 0.09	99.04± 0.14	86.4 ± 1.74

Table 2: Comparison of NEO's mean accuracy (MA) with other regularization-based approaches on the Split MNIST, Split Fashion-MNIST(FMNIST) and Split-CIFAR10 benchmarks in a Task-IL scenario. *Each result was averaged across 5 different initializations.

As shown in Table 3, NEO is able to achieve mean accuracy almost comparable to other weight measurement based regularization approaches on both the Split-MNIST and Split-FMNIST benchmarks while having minimal memory overhead. The metric mean accuracy (MA) is measured, across the entire set of tasks $T^1 - T^N$, after training on the final task T^N, represented by the equation

$$\text{MA} = \sum_{t=1}^{N} \frac{\psi^{t,N}}{N} \tag{8}$$

Model	FMNIST (MA%)	MNIST (MA%)	Memory Overhead (MO)
Baseline	65.52 ± 1.31	60.13 ± 1.66	1
LwF– [22]	71.02 ± 0.46	71.5 ± 1.63	2x
MAS– [3]	68.57 ± 6.85	66.42 ± 2.47	3x
Online-EWC–[29]	65.55 ± 3.30	64.32 ± 2.48	3x
BGD– [39]	89.73 ± 0.88	80.44 ± 0.45	3.44x
SS– [28]	91.98 ± 0.12	82.9 ± 0.01	2x
Metaplastic BNN– [18]	82.44 ± 1.34	73.23 ± 2.31	1.03x
UCL– [10]	69.72 ± 2.53	66.40 ± 1.74	1.5x
NAI– [11]	68.82 ± 1.15	68.35 ± 1.34	1.002x
NEO	86.82±0.6	78.14± 2.23	1.002 - 1.5x

Table 3: Comparison of NEO's mean accuracy (MA) and memory overhead (MO) with other regularization-based approaches on the Split MNIST and Split Fashion-MNIST(FMNIST) benchmarks in the Domain-IL scenario. *Each result was averaged across 5 different initializations.

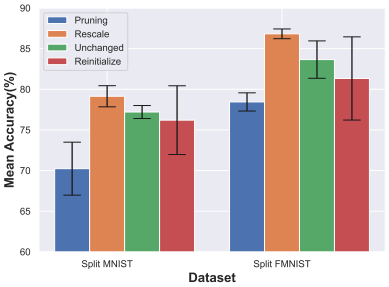
, where $\psi^{t,N}$ is the accuracy on task T^t after training task T^N. In addition to this, to measure the model's cost we use the memory overhead metric, MO, calculated as the average amount of memory that a model requires per task, $Mem(T^t)$, in units of the baseline model's memory size Mem_b,

$$\text{MO} = min\Big(1, \frac{1}{N}\sum_{t=1}^{N}\frac{Mem(T^t)}{Mem_b}\Big) \tag{9}$$

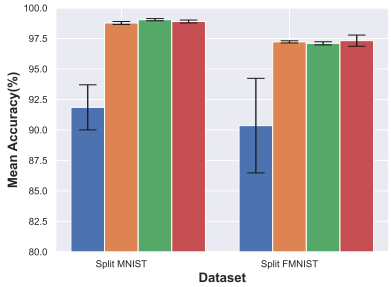
It can be observed that incorporating neuron-based importance significantly reduces the memory overhead and this overhead does not grow quadratically with the increasing number of tasks, thereby making the mechanism amenable to be combined with other learning methodologies. Additionally, since the storage requirements increase as $O(\text{n}+n^i * n^p)$, wherein n^i and n^p represent the important and plastic neurons, respectively. We observe that the total number of important neurons are generally less than half of the total neuronal count (refer to Figure 7b) therein reducing the storage requirements significantly. Therefore, NEO ideally operates within a worst case overhead of 1.5x. The results in Tables 2 and 3 show that NEO is not able to perform better than UCL [10] and NAI [11] on Task-IL scenarios. This can be attributed to the way we model the state update interval and regularization policies. UCL and NAI both utilize weight reinitialization policies which are able to help with plasticity when task boundaries are known. However, when we attempt to reinitialize all the weights in task-agnostic scenarios or when the state update interval cannot define the shift in context, it affects the network activity thus leading to a significant drop in performance. As the proposed model does not incorporate these weight reinitialization schemes, we notice a slight degradation in performance in Task-IL scenarios. On the other hand, NEO outperforms UCL and NAI in task-agnostic scenarios.

3.3 Impact of reinitializing connections

To evaluate the effect of different connection topologies for the outgoing connections from unimportant neurons, we performed an ablation analysis for the model by selecting different schemes. Figure 6 shows the difference in performance for different connection choices selected for the outgoing weights from the unimportant

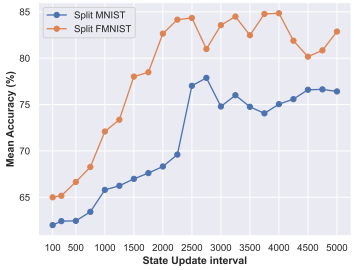


(a) Performance with respect to different learning mechanisms in Domain-IL scenarios.

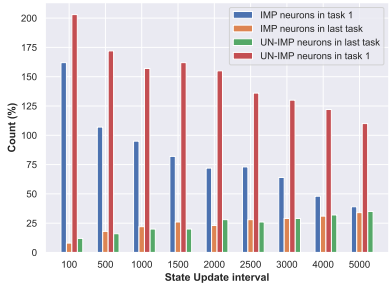


(b) Performance with respect to different learning mechanisms in Task-IL scenarios.

Figure 6: The performance of NEO on Split-MNIST and Split-FMNIST in a task-aware and task-agnostic scenarios, categorized by different learning mechanisms for the outgoing weight of unimportant neurons.



(a) Mean accuracy of model with different state update intervals on Split MNIST and FMNIST benchmarks.



(b) Average per-layer count of neurons that transitioned to Important (IMP) and Unimportant (UN-IMP) state in the first and last tasks.

Figure 7: Analysis of the model's performance and state distribution with respect to the state update interval. Having a higher state update interval generally leads to meaningful state transitions and better performance.

neurons. The works in UCL and NAI proposed different pruning and reinitialization strategies for unimportant neurons. Therefore, we evaluated those mechanisms to see how plasticity is affected by incorporating them in our framework. The different strategies we selected are namely, i) Pruning - the outgoing connections from unimportant to plastic/unimportant neurons are set to 0, ii) Weight rescaling - The weight rescaling principle specified in Section 2.4, iii) Unchanged - The weights from the unimportant neurons are not changed using any other mechanism and are updated normally using SGD, and iv) Reinitialization - The outgoing weights are reinitialized at every state update interval. As it can be noticed in figure 6b, reinitialization boosts the performance in Task-IL scenarios as the weights are being reinitialized at the task boundary. This boosts the model plasticity for learning new tasks, however when we reinitialize in Domain-IL scenarios as shown in figure 6a, adding sudden plasticity interferes with learning and there is a high variance in performance depending on the type of initialization. Overall, weight rescaling works better in task-agnostic scenarios, wherein the weights are scaled based on their strengths and therein regulating plasticity without having the issue of shadow activations. One surprising observation was that, not attempting to add any other mechanism on the weights also performed comparably to when the weights were being fine-tuned using additional mechanisms. The reason behind could pertain to the fact that the error gradients to those weights were low enough to drive any significant change, thereby not requiring further rescaling or reinitialization.

3.4 Modeling the state update interval

The state update interval ($\mathcal{T}$) can play a critical role in determining the neuronal states within a task. For task-aware scenarios, the state update interval does not significantly contribute, since the best time to update the states corresponds to the task boundary. However, when learning in an online manner in task-agnostic scenarios, the rate at which states are updated becomes necessary to avoid too much stability or plasticity. Figure 7 shows the impact of different state update interval values on the performance of the model and how the number of neuronal states vary across the layers based on the update interval. Figure 7a shows the impact of interval on the performance. Overall, the performance improves with higher state update interval. This can be attributed to the fact that the network gets more time to adjust to the changes in the input distribution and consolidates those changes periodically. To observe how these changes occur, we can look into figure 7b, which

shows the distribution of the neuronal states at the end of first and final tasks. With lower state update intervals, the network starts to assign important neurons quickly and apply the consolidation and plasticity mechanisms quickly thereby leading to many neurons transitioning to the either important or unimportant states. This behavior impacts the performance for learning later tasks which is clearly visible by looking at the number of neurons transitioning in later tasks. However, with increased interval count, the model uniformly distributes the state transitions across different tasks. We currently set $\mathcal{T}$ to 2500 for the current benchmarks. However, a dynamic way of selecting state update interval can also be derived using the activity status in the network in conjunction with other attention mechanisms which would be explored in future work.

CONCLUSION

We introduce a novel neuron importance based regularization approach coupled with selective state-dependent learning mechanisms that is able to address catastrophic forgetting in both task-aware and agnostic scenarios. NEO has minimal memory overhead, making it a viable solution for larger models and for models solving complex tasks. We also present how different learning mechanisms such as weight rescaling, pruning and reinitialization affect the performance of the model in different learning scenarios. In future work, we aim to combine NEO with other activity dependent mechanisms, such as neuromodulation, to enable few shot continual learning capabilities. Another extension is to incorporate dynamic context detection module to select the state update interval based on context.

REFERENCES

[1] Wickliffe C. Abraham. 2008. Metaplasticity: Tuning Synapses and Networks for Plasticity. *Nature Reviews Neuroscience* 9, 5 (May 2008), 387–399. https://doi.org/10.1038/nrn2356

[2] Hongjoon Ahn, Sungmin Cha, Donggyu Lee, and Taesup Moon. 2019. Uncertainty-based continual learning with adaptive regularization. *Advances in neural information processing systems* 32 (2019).

[3] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. 2018. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 139–154.

[4] Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. 2019. Gradient based sample selection for online continual learning. In *Advances in Neural Information Processing Systems*. 11816–11825.

[5] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. 2018. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*. 532–547.

[6] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. 2021. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence* 44, 7 (2021), 3366–3385.

[7] Sayna Ebrahimi, Franziska Meier, Roberto Calandra, Trevor Darrell, and Marcus Rohrbach. 2020. Adversarial Continual Learning. *arXiv preprint arXiv:2003.09553* (2020).

[8] Georgios Georgiadis. 2019. Accelerating convolutional neural networks via activation map compression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7085–7095.

[9] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. 2018. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488* (2018).

[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] Sohee Kim and Seungkyu Lee. 2022. Continual Learning with Neuron Activation Importance. In *International Conference on Image Analysis and Processing*. Springer, 310–321.

[12] 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.

[13] 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.

[14] Soheil Kolouri, Nicholas A Ketz, Praveen K Pilly, and Andrea Soltoggio. 2020. Sliced Cramer synaptic consolidation for preserving deeply learned representations. In *International Conference on Learning Representations*.

[15] Alex Krizhevsky and Geoffrey Hinton. 2009. *Learning multiple layers of features from tiny images*. Technical Report. Citeseer.

[16] Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew P Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, et al. 2022. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence* 4, 3 (2022), 196–210.

[17] Mark Kurtz, Justin Kopinsky, Rati Gelashvili, Alexander Matveev, John Carr, Michael Goin, William Leiserson, Sage Moore, Nir Shavit, and Dan Alistarh. 2020. Inducing and exploiting activation sparsity for fast inference on deep neural networks. In *International Conference on Machine Learning*. PMLR, 5533–5543.

[18] Axel Laborieux, Maxence Ernoult, Tifenn Hirtzlin, and Damien Querlioz. 2021. Synaptic metaplasticity in binarized neural networks. *Nature communications* 12, 1 (2021), 1–12.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (Nov. 1998), 2278–2324. https://doi.org/10.1109/5.726791

[20] Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. 2020. A neural dirichlet process mixture model for task-free continual learning. *arXiv preprint arXiv:2001.00689* (2020).

[21] Sang-Woo Lee, Jin-Hwa Kim, Jaehyun Jun, Jung-Woo Ha, and Byoung-Tak Zhang. 2017. Overcoming catastrophic forgetting by incremental moment matching. In *Advances in Neural Information Processing Systems*. 4652–4662.

[22] Zhizhong Li and Derek Hoiem. 2017. Learning without forgetting. *IEEE transactions on pattern analysis and machine intelligence* 40, 12 (2017), 2935–2947.

[23] David Lopez-Paz et al. 2017. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*. 6467–6476.

[24] Michael McCloskey and Neal J. Cohen. 1989. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. In *Psychology of Learning and Motivation*. Vol. 24. Academic Press, 109–165. http://www.sciencedirect.com/science/article/pii/S0079742108605368

[25] Tej Pandit and Dhireesha Kudithipudi. 2020. Relational neurogenesis for lifelong learning agents. In *Proceedings of the Neuro-inspired Computational Elements Workshop*. 1–9.

[26] German I Parisi, Xu Ji, and Stefan Wermter. 2018. On the role of neurogenesis in overcoming catastrophic forgetting. *arXiv preprint arXiv:1811.02113* (2018).

[27] 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).

[28] Steger. Angelika Schug. Simon, Benzing. Frederik. 2020. Task-Agnostic Continual Learning via Stochastic Synapses. https://sites.google.com/view/cl-icml/accepted-papers?authuser=0. (Accessed on 09/21/2020).

[29] Jonathan Schwarz, Jelena Luketina, Wojciech M Czarnecki, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. 2018. Progress & compress: A scalable framework for continual learning. *arXiv preprint arXiv:1805.06370* (2018).

[30] James Smith, Yen-Chang Hsu, Jonathan Balloch, Yilin Shen, Hongxia Jin, and Zsolt Kira. 2021. Always Be Dreaming: A New Approach for Data-Free Class-Incremental Learning. *International Conference on Computer Vision (ICCV)* (2021).

[31] Nicholas Soures, Peter Helfer, Anurag Daram, Tej Pandit, and Dhireesha Kudithipudi. July 2021. TACOS: Task Agnostic Continual Learning in Spiking Neural Networks. In *Theory and Foundation of Continual Learning Workshop at ICML'2021*.

[32] Gido M van de Ven, Hava T Siegelmann, and Andreas S Tolias. 2020. Brain-inspired replay for continual learning with artificial neural networks. *Nature communications* 11, 1 (2020), 1–14.

[33] Gido M. van de Ven and Andreas S. Tolias. 2019. Three Scenarios for Continual Learning. *arXiv:1904.07734 [cs, stat]* (April 2019). arXiv:1904.07734 [cs, stat]

[34] Gido M van de Ven, Tinne Tuytelaars, and Andreas S Tolias. 2022. Three types of incremental learning. *Nature Machine Intelligence* (2022), 1–13.

[35] BP Welford. 1962. Note on a method for calculating corrected sums of squares and products. *Technometrics* 4, 3 (1962), 419–420.

[36] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *CoRR* abs/1708.07747 (2017), 6 pages. arXiv:1708.07747 http://arxiv.org/abs/1708.07747

[37] Friedemann Zenke, Ben Poole, and Surya Ganguli. 2017. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning*, Vol. 70. JMLR. org, 3987–3995.
[38] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. 2018. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123* (2018).
[39] Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. 2018. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123* (2018).