# IMPROVING PERFORMANCE IN CONTINUAL LEARNING TASKS USING BIO-INSPIRED ARCHITECTURES

**Sandeep Madireddy**
Argonne National Laboratory
Lemont, IL, USA
smadireddy@anl.gov

**Angel Yanguas-Gil**
Argonne National Laboratory
Lemont, IL, USA
ayg@anl.gov

**Prasanna Balaprakash**
Oak Ridge National Laboratory
Oak Ridge, TN, USA
pbalapra@ornl.gov

## ABSTRACT

The ability to learn continuously from an incoming data stream without catastrophic forgetting is critical to designing intelligent systems. Many approaches to continual learning rely on stochastic gradient descent and its variants that employ global error updates, and hence need to adopt strategies such as memory buffers or replay to circumvent its stability, greed, and short-term memory limitations. To address this limitation, we have developed a biologically inspired lightweight neural network architecture that incorporates synaptic plasticity mechanisms and neuromodulation and hence learns through local error signals to enable online continual learning without stochastic gradient descent.

Our approach leads to superior online continual learning performance on Split-MNIST, Split-CIFAR-10, and Split-CIFAR-100 datasets compared to other memory-constrained learning approaches and matches that of the state-of-the-art memory-intensive replay-based approaches. We further demonstrate the effectiveness of our approach by integrating key design concepts into other backpropagation-based continual learning algorithms, significantly improving their accuracy. Our results provide compelling evidence for the importance of incorporating biological principles into machine learning models and offer insights into how we can leverage them to design more efficient and robust systems for online continual learning.

## 1 INTRODUCTION

Online continual learning addresses the scenario where a system has to learn and process data that are continuously streamed, often without restrictions in terms of the distribution of data within and across tasks and without clearly identified task boundaries Mai et al. (2021); Chen et al. (2020); Aljundi et al. (2019a). Online continual learning algorithms seek to mitigate catastrophic forgetting at both the data-instance and task level Chen et al. (2020). In some cases, however, such as on-chip learning at the edge, additional considerations such as resource limitations in the hardware, data privacy, or data security are also important for online continual learning.

A key challenge of online continual learning is that it runs counter to the optimal conditions required for optimization using stochastic gradient descent (SGD) Parisi et al. (2019), which struggles with non-stationary data streams Lindsey & Litwin-Kumar (2020). On the contrary, biological systems excel at online continual learning. Inspired by the structure and functionality of the mammal brain, several approaches have adopted replay strategies to counteract catastrophic forgetting during non-stationary tasks. However, online continual learning is present even in simpler species, such as invertebrates, lacking the basic structures required to implement any type of replay. This situation suggests that the use of local learning rules, combined with other structural features in the brain of invertebrates, should be enough to achieve online continual learning without needing to keep a memory buffer of past experiences.

Most continual learning approaches in the literature Hadsell et al. (2020); Delange et al. (2021) have been designed for offline continual learning scenarios where non-stationarity is assumed just between tasks while the data within tasks are assumed to be i.i.d and multiple passes are made through them. Some recent work Aljundi et al. (2019b;a); Chaudhry et al. (2018); Chen et al. (2020); Caccia et al. (2022) considered online continual learning; however, the majority of the works (see Chen et al. (2020)) adopt a task-incremental learning setting where task labels are explicitly provided at test time. Only a handful of works consider the class-incremental learning scenario Hsu et al. (2018) that does not have access to task labels, and is therefore much harder. To handle this scenario, most of the approaches (which employ SGD and global gradient updates) resort to expensive memory buffers Hsu et al. (2018); Buzzega et al. (2020); Caccia et al. (2022) and auxiliary data Zhang et al. (2020) to mitigate catastrophic forgetting in class-incremental learning.

In this work, we introduce the neuromodulated neural architecture (NNA), a biologically inspired architecture that exhibits excellent performance in various continual learning tasks. First, we mimic the heterogeneous plasticity of the insect brain by decoupling our networks in a feature extraction and a learning component. Second, we use local synaptic plasticity rules to carry out supervised learning on data streams, where labels are passed as modulatory signals akin to what happens in ternary synapses. Third, we introduce novel local learning rules that mimic the transition from short-term to long-term memory that takes place in individual synapses.

Using this architecture, we demonstrate accuracies (with Split-MNIST, Split-CIFAR-10, and Split-CIFAR-100 datasets) that outperform the memory-free approaches and match the approaches that employ memory buffers, with a shallow network and without memory replay, in both task-incremental and class-incremental settings Hsu et al. (2018). We also demonstrate that salient design principles from the NNA approach that are effective with mitigate catastrophic forgetting, i.e. the biologically inspired neural architecture and loss formulation, can also be integrated into existing continual learning algorithms and improve their online continual learning performance.

## 2 Related Work

**Continual learning:** Several continual learning approaches have been presented in the literature to circumvent catastrophic forgetting that can occur due to the non-stationarity in the data brought about by the task switching. These approaches can loosely be categorized as (1) memory buffer and replay based, which store samples from previously observed tasks and replay them while learning a new task; (2) regularization based, which impose constraints to boost knowledge retainment; (3) metalearning based, which use a series of tasks to learn a common parameter configuration that is easily adaptable for new tasks; and (4) parameter memory-based, which incorporate the task-specific memories directly onto in the network weights.

The *replay*-based approaches maintain an exemplar data buffer of samples from previous tasks. Trade-offs have to be made on the size of this buffer and the bias introduced by the use of data from just the most recent task. Notable approaches in this category include bioinspired dual-memory architecture Parisi et al. (2019) and deep generative replay Shin et al. (2017), which involves a cooperative dual model architecture framework inspired by the hippocampus, which retains past knowledge by the concurrent replay of generated pseudo data. Gradient episodic memory Lopez-Paz & Ranzato (2017), generative replay with feedback connections van de Ven & Tolias (2018), and gradient-based sampling Aljundi et al. (2019b) are other examples of replay-based approaches.

The *regularization*-based approach includes algorithms such as elastic weight consolidation (EWC) Kirkpatrick et al. (2017), which computes synaptic importance using a Fisher importance matrix-based regularization; synaptic intelligence Zenke et al. (2017), whose regularization penalty is similar to EWC but is computed online at per-synapse level; and leaefning without forgetting Li & Hoiem (2017), which applies a distillation loss on the attention-enabled deep networks seeking to minimize task overlap.

The *metalearning*-based continual learning consists of an inner loop that learns parameters in a prediction network and an outer loop that learns the parameters in a representation learning network. These loops are updated by using different strategies and data splits at meta-training and meta-testing. Recent approaches in this category include online metalearning (OML) Javed & White (2019), neuromodulated metalearning algorithm (ANML) Beaulieu et al. (2020), and incremental task-agnostic metalearning (iTAML) Rajasegaran et al. (2020). In terms of memory, a naive metalearning approach will maintain full task history to update the parameters for each traversal through the loop when a new task is observed. OML and ANML algorithms keep all the tasks in memory but randomly choose the task sequences for meta-training (not just from task history) and meta-testing updates. iTAML, on the other hand, uses data from the current task and a exemplar data (memory buffer) from the previously seen tasks to metalearn. The *parameter memory*-based approaches either have fixed architecture and dedicate different subsets of model parameters for different tasks with task-specific masks Serra et al. (2018) or dynamically grow the network, with a new branch for each task Aljundi et al. (2017).

**Online continual learning:** Most of these approaches however, have been designed for the traditional continual learning scenarios where non-stationarity is assumed between tasks while the data within tasks are assumed to be i.i.d and multiple passes are made through them. Recent works, such as Aljundi et al. (2019b;a); Chaudhry et al. (2018); Chen et al. (2020), started to look at the more challenging online continual learning scenario where the data is seen only once. In this scenario, however, the majority of the works Chen et al. (2020) have looked at the task-incremental learning setting where the task labels are explicitly provided at test time. Only a few works Aljundi et al. (2019b;a); Lee et al. (2020) have employed the class-incremental learning scenario, which does not have access to task labels and hence is much harder. The restriction of learning with only a single pass over the data makes this even harder. To

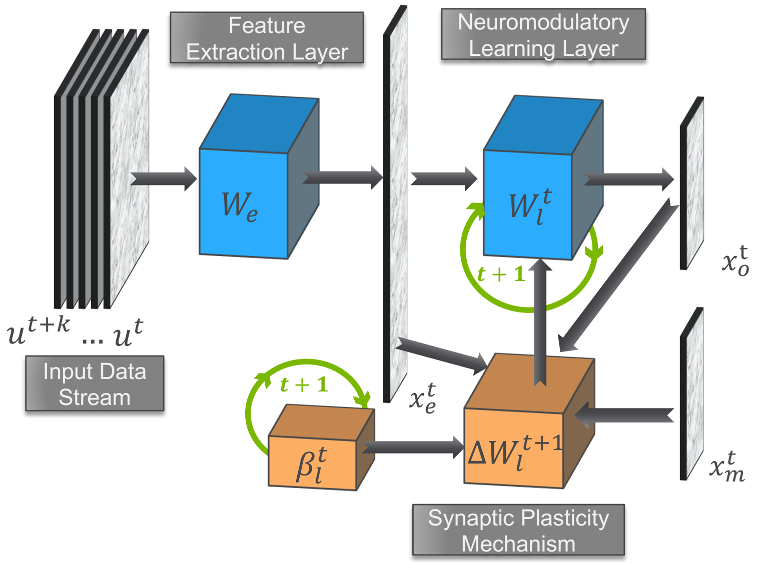

Figure 1: Multilayer neuromodulated architecture: (a) feature extraction, (b) neuromodulated learning layers; enabling synaptic plasticity mechanisms through local learning rules.

handle this scenario, most of the approaches resort to memory buffers Hsu et al. (2018); Buzzega et al. (2020); Caccia et al. (2022) and auxiliary data Zhang et al. (2020).

**Synaptic Plasticity and Neuromodulation:** Synaptic plasticity mechanisms and neuromodulation have been successfully adopted in supervised learning. For example, Miconi et al. (2018) adopts Hebbian plasticity for designing plastic neural networks capable of metalearning. Another work Miconi et al. (2020) extends the Hebbian plasticity to incorporate neuromodulation for similar tasks, Daram et al. (2020) adopts neuromodulated plasticity for one-shot image classification, and Vecoven et al. (2020) adopts it for reinforcement learning. Recently Thangarasa et al. (2020) adopts Hebbian synaptic consolidation combined with regularization-based approaches for task-incremental learning on simple MNIST-based benchmarks.

## 3 Model

### 3.1 Neuromodulated Neural Architecture

Our neuromodulated neural architecture (NNA) is elucidated in Fig. 1. Our architecture comprises two components, the feed-forward map $\mathcal{F}$ and the hidden states $S^t$, such that

$$x_o^t = \mathcal{F}\left(u^t, x_m^t; S^t\right), \tag{1}$$

where $u^t = \{x^t, y^t\}$, is the input, $x_m^t$ is the modulatory signal evaluated using the ground truth label $y^t$, and $x_o^t$ is label prediction. The evolution of the hidden state can be described as

$$S^{t+1} = \mathcal{G}\left(S^t, u^t, x_m^t, x_o^t\right), \tag{2}$$

where $\mathcal{G}$ is a function that models the evolution of the internal state. A key difference between our formulation of online learning and that of conventional stochastic gradient descent is that the evolution of $S^t$ is computed in real time based solely on the most recent inputs.

We propose a simple architecture that breaks down the input-output map $\mathcal{F}\left(u^t, x_m^t; S^t\right)$ into two components: a feature extraction layer $\mathcal{F}_e$ that transforms input into an internal representation $x_e^t = \mathcal{F}_e(u^t, W_e)$ and a neuromodulated learning layer $\mathcal{F}_l$ that transforms the internal representation into outputs, $x_o^t = \mathcal{F}_l(x_e^t, W_l^t)$. Here, $W_e$ is a projection matrix, while $W_l$ are plastic synaptic weights.

The internal state $S^t$ comprises the synaptic plasticity mechanism $\Delta\, W_l^t$ and a hyperparameter vector $\beta_l^t$ that feeds into the synaptic plasticity rule. Therefore, we can decompose $\mathcal{G}$ into two functions, $\mathcal{F}_w$ and $\mathcal{F}_\beta$, such that $\mathcal{G} = [\mathcal{F}_w, \mathcal{F}_\beta]$. The function $\mathcal{F}_w$ refers to the synaptic plasticity mechanism such that

$$\text{Synaptic Plasticity Mechanism:} \quad \Delta W_l^{t+1} = \mathcal{F}_w\left(x_e^t, x_o^t, x_m^t; W_l^t, \beta_l^t\right), \tag{3}$$

with the update for the weights $W_l^t$ in the neuromodulatory layer described as $W_l^{t+1} = W_l^t + \Delta W_l^{t+1}$. Furthermore, the the evolution of hyperparameters $\beta_l^t$ that constitute the synaptic plasticity mechanism is given by

$$\text{Hyperparameter Evolution:} \qquad \beta_l^{t+1} = \mathcal{F}_\beta \left( x_e^t, x_o^t, x_m^t; W_l^t, \beta_l^t \right). \tag{4}$$

Since training this system does not involve storage of batches of prior data or configurations, any memory of prior processes has to be built into the equations for $W_l^t$ and $\beta_l^t$. We note that, in contrast to conventional learning approaches, there is no separation between architecture and learning algorithm: the ability to learn is defined by the choice of $\mathcal{F}_w$, $\mathcal{F}_\beta$, and their hyperparameters. To this end, a sample from the hyperparameter configuration space comprising $\left\{ \mathcal{F}_e, \mathcal{F}_l, \mathcal{F}_w, \mathcal{F}_\beta, W_l^{t=1}, \beta_l^{t=1} \right\}$ uniquely defines an NNA. In the next sections, we describe the specific components chosen in this work and present the training mechanism. The NNA in Figure 1 shows one feature extraction and one neuromodulated learning layer.

### 3.1.1 Feature extraction layer

For single-channel datasets we have considered a sparse layer that follows the same design pattern found in the cerebellum and the insect's mushroom body (Litwin-Kumar et al., 2017; Yanguas-Gil, 2019). The $\alpha$-dimensional input ($u^t$) is projected into a much larger $\kappa$-dimensional space ($x_e^t$) through the sparse projection matrix $W_e$. We use dynamic thresholding in the sparse layer followed by a rectified linear unit to ensure that only the most salient features are included in the representation, which we refer to as *activity sparsity*. Putting these choices together defines our feature extraction layer $\mathcal{F}_e(\cdot)$ to be

$$\mathcal{F}_e : x_e^t = \text{ReLU} \left( \mathrm{W_e^t} u^t - \mu - k\sigma \right), \tag{5}$$

where $\mu$ and $\sigma$ are respectively the mean and standard deviation of the matrix multiplication product $\mathrm{W_e^t} u^t$, and $k$ is a constant controlling the cutoff. This approach highlights correlations between different input channels and represents the broadest possible prior for feature extraction, since it does not assume any spatial dependence or correlations between inputs. Here $\beta_e = \{\kappa, \mu, k, \sigma\}$ are all hyperparameters in the feature extraction layer that can be adjusted to the network.

For experiments involving 3-channel images, we transfer feature extractors from various pretrained models. Here, we have used the wide-ResNet50 (Zagoruyko & Komodakis, 2016) model pretrained on ImageNet data (Deng et al., 2009) after experimenting with a wide variety of pretrained model and data combinations. Hence, we compose these feature extractors $F_p^e(\cdot)$ with Eqn 5 to get the feature extraction layer

$$\mathcal{F}_e : x_e^t = \text{ReLU} \left( \mathrm{W_e^t} (F_p^e(u^t)) - \mu - k\sigma \right). \tag{6}$$

### 3.1.2 Neuromodulated learning layer

The neuromodulated learning layer comprises one or more all-to-all connected layers where online supervised learning occurs. $\mathcal{F}_l(\cdot)$ is chosen to be

$$\mathcal{F}_l : x_o^t = \text{Tanh} \left( \text{ReLU} \left( \mathrm{W_l^t} x_e^t \right) \right). \tag{7}$$

Synaptic weights ($W_l$) are updated in real time through a series of local learning rules ($\mathcal{F}_w(\cdot)$) that provide (local) gradient updates. In this work, we focus specifically on modulated synaptic plasticity rules for supervised learning that are biologically enabled by ternary synapses, where in addition to the presynaptic ($x_e^t$) and postsynaptic ($x_o^t$) inputs, learning is influenced by a local modulatory signal ($x_m^t$) controlling how learning takes place. In this context, the set of hyperparameters would consist of static parameters $\beta^f = \{\beta_1^f, \beta_2^f, \beta_3^f\}$ and dynamic, $\beta_l^t = \{l_r^t\}$ (e.g., learning rate). In particular, we consider the following four learning rules:

**Generalized Hebbian model** (GEN): a modulated version of the covariance rule commonly used in neuroscience, where the synaptic weight evolution is given by $\Delta W_l^{t+1} = l_r^t x_m^t (x_e^t - \beta_1^f)(x_o - \beta_2^f)$. This rule is well known to be unstable, and clamping or a regularization mechanism needs to be included in order to keep the synaptic weights bounded. The rule is at the core of some recent the neuromodulation-based approaches (Miconi et al., 2018).

**Oja's rule** (OJA): it is a heuristic modification of the basic Hebb's rule providing a normalization mechanisms through a first-order loss term (Oja, 1982) given by $\Delta W_l^{t+1} = l_r^t x_m^t (x_e^t x_o^t - \beta_1^f {x_o^t}^2 W_l^t)$.

**Mean square error rule** (MSE): a non-Hebbian rule based on synaptic plasticity mechanisms in the mushroom body, which is a key memory and learning center of the insect brain (Hige et al., 2015). The key assumptions of this rule are that learning takes place even in absence of postsynaptic activity and is modulated by the difference between the stimulus and the predicted response. We can model this effect using a mean square error (MSE) expression:

$\Delta W_l^{t+1} = l_r^t(x_m^t - x_o^t)x_e^t$. This rule therefore establishes a connection between a bioinspired process and a cost function used in stochastic gradient descent methods.

**Inelastic learning rule** (INEL): a metaplasticity rule that augments the MSE rule by introducing memory effects on the synaptic plasticity. In order to achieve long-term potentiation, some synapses in the mushroom body require the recurrence of a specific stimulus. One way of implementing this dependence with past inputs is by considering a synapse-specific window for plasticity based on how significant a weight $W_{ij}$ (which is the $W_l^t$ in equ. 4, where i, j are the indices corresponding to the dimension of $x_e^t$ and $x_o^t$) with respect to its default value. One simple way to codify this behavior is to use the mean synaptic weight of a layer as a reference value for the plasticity window. With this assumption, the learning rate at each synapse $K_{ij}$ can be expressed as $K_{ij}^t = H\left(1 - \beta_1^f |W_{ij}^t - \mu|\right)$, Where $H(\cdot)$ is the Heaviside function and $\mu$ is the average synaptic weight of each layer. The resulting $K_{ij}$ is then fed to an MSE rule. This rule introduces a simple mechanism for memory consolidation based on the inelasticity of the learning rate with respect to the value of the synaptic weight. Moreover, this consolidation is not irreversible: the evolution of the remaining weights can bring weights from frozen to active by reducing its significance.

For all the rules, we have added other conditions to provide symmetric or positive clamping of the synaptic weights in order to prevent instabilities in the algorithm. Finally, the evolution of the hyperparameters with time is given by $\mathcal{F}_\beta : \beta_l^{t+1} = (\beta_l^t + \beta_l^{t=1} * \beta_3^f)/(1 + \beta_3^f)$, The algorithm to train NNA in an online learning scenario is described in Algorithm 1.

**Algorithm 1:** Training NNA for online learning

```
$acc = 0$;
$N_{Iter} = N_{Train} \times N_{epochs}$;
while $t < N_{Iter}$ do
    $u, y_u$ = Data[$rand^t$];
    $x_m$ = zeros($\delta$);
    $x_m[y_u]$ = 1 ;
    /* Feature extraction */
    $x_e = F_e(u)$;
    /* Neuromodulated learning */
    $x_o = F_l(x_e, W_l^t, x_m)$;
    /* Synaptic plasticity mechanism */
    $W_l^{t+1} = W_l^t + \Delta W_l^{t+1}$ ;
    if argmax($x_o$ == lab) then
        $acc = acc + 1$;
    end
    $acc = acc/N_{Iter}$;
end
```

## 4 Methodology: Online Learning Scenarios and Architecture Configuration

### 4.1 Online Learning Scenarios

We have considered experiments in which the system is subject to a stream of data and labels, evolving its internal configuration during a predetermined number of epochs. This constitutes a single episode. At the end of the episode, the system is evaluated against the testing dataset to obtain its accuracy. By concatenating multiple episodes involving different tasks and datasets, we can create a curriculum to evaluate the system's ability to carry out online continual learning.

We consider three learning modalities. *(1) Single-Task Online Learning* – a single episode in the curriculum streams all the data for the system to perform a single task (e.g., multiclass classification). *(2) Task-Incremental Continual Learning* – the system is subjected to a sequence of episodes, each comprising a specific task (e.g. two-class classification). Each episode has a different output head (e.g., two-class classifier). Task labels are used both at train and test time. *(3) Class-Incremental Continual Learning* – episodes and curriculum are similar to task-incremental learning, except that the model learns with a single (shared) output head. The final accuracy measures the model's ability to predict on test data from all the classes, spanning across episodes. The learning algorithm utilizes the task labels only at train time. Thus, this is much harder scenario and closer to "task-free" continual learning.

| Bio-inspired design | Backprop-based algorithm |
|---|---|
| Sparse projection layer | Sparse projection with fixed weights |
| Bounded spike rate | ReLU + Tanh / sigmoid activation functions |
| Fixed activation threshold | Constant bias + hard threshold |
| Local learning rule | Modified MSE loss function<br>Weight clamping after optimization step |

Table 1: Strategy to transfer the key innovations from the bio-inspired model into a BackPropagation-based Algorithm

### 4.2 Transfer of bio-inspired design concepts to backPropagation-based algorithms

A key challenge of bio-inspired models is that they are hard to integrate with state-of-the-art approaches based on deep neural networks. To overcome this problem and integrate the core ideas of our model with other continual learning algorithms, we have transferred the key design concepts to a backprop setting. The key insight is that the MSE learning rule described above closely resembles the change in weights expected from applying an SGD optimizer with an MSE loss function. Consequently, we can recreate the main features of our model through a combination of optimizers, loss functions, custom layers, and weight clamping. Our approach is summarized in Table 1

From a network architecture perspective, the sparse layer is computed directly using Eq. 5, where the mean and standard deviation are calculated sample by sample (i.e., in contrast to batch normalization approaches). The output of the sparse layer is then passed to Eq. 7, which is implemented as a linear layer with constant bias, whose output is passed through a rectified linear unit and a hyperbolic tangent function. We can view the synaptic plasticity rule as a combination of loss function and optimizer. If we focus on the MSE rule, weight updates are given by:

$$\Delta W_l^{t+1} = l_r^t(x_m^t - x_o^t)x_e^t \tag{8}$$

Eq. 8 resembles the weight updates that would result from applying a mean square error loss function followed by an SGD optimizer without momentum. Therefore, we have adapted this combination in our backprop studies. This is in contrast to most continual learning algorithms, which tend to focus on a cross-entropy loss function. Furthermore, weight clamping is an essential feature of our bio-inspired model. Therefore, we have also introduced a clamping mechanism that can be invoked each time the optimizer is called. Finally, the use of constant bias in the bio-inspired model coupled with a hard threshold leads to a high probability of $x_e$ and $x_o$ being zero. This leads to modifications of the synaptic weights as long as the output neurons are below that threshold. We can generalize this idea by introducing modifications to the MSE rule that attempt to similarly minimize when learning takes place through the use of a filtered MSE that returns a zero loss function whenever the maximum output corresponds to the target label.

## 5 Results and Discussion

### 5.1 Single-Task Online Learning

To validate our architecture's ability to carry out online learning, we tested it against the MNIST, Fashion MNIST (Xiao et al., 2017b) (F-MNIST), CIFAR-10, and CIFAR-100 datasets. These are prevalent in the literature on continuous learning. In all cases, the networks were capable of achieving high accuracies in as few as 0.5 epochs. The comparison of the best accuracy obtained for each learning rule among the configurations evaluated in the search for each dataset is shown in Table 2a. In Table 2b we show the classification accuracies for each of the five data sets obtained using the optimal configurations but run during the 1 epoch. We also compare our approach with the kernelized information bottleneck (KIB) Pogodin & Latham (2020) approach, which is also an SGD-alternative local learning approach that is biologically plausible. Although we have a shallow network (approximately 100k learnable weights) and operate in online learning with 1 training epoch, we obtain a test accuracy of 96.81 on MNIST data, 85.22 on F-MNIST data, and 73.24 on CIFAR-10 data. These results are close to the accuracy of 98.1 obtained by KIB (trained with 3 layers, each with 1,024 neurons of a fully connected network for 100 epochs) on the MNIST data. Our approach significantly outperforms KIB on the CIFAR-10 data. Our accuracies are also on par with (or outperform) other SGD-based shallow network architectures (Yanguas-Gil et al., 2019; Xiao et al., 2017a; Cohen et al., 2017).

### 5.2 Continual Learning

We now discuss the results for the task-incremental and class-incremental continual learning scenarios with the Split-MNIST, Split-CIFAR-10, and Split-CIFAR-100 continual learning benchmarks that have been extensively adopted in the literature (Shin et al., 2017; Zenke et al., 2017; Nguyen et al., 2017). Split-MNIST data is prepared by splitting

| Rule/Data | MNIST | F-MNIST | CIFAR-10 |
|---|---|---|---|
| GEN | 31.20 | 32.38 | 32.39 |
| OJA | 77.63 | 63.18 | 52.87 |
| MSE | 96.16 | 85.13 | 77.37 |
| INEL | 96.40 | 85.09 | 77.96 |

| Algo/Data | MNIST | F-MNIST | CIFAR-10 |
|---|---|---|---|
| pHSIC (Shallow), 100 epoch | 98.1 | 88.8 | 46.4 |
| Ours, 1 epoch | 96.81 | 85.22 | 73.24 |

Table 2: (left) (a) Comparison of accuracy across datasets and learning rules; (right) (b) Classification accuracy in the single-task online learning scenario for MNIST, F-MNIST, and CIFAR-10 datasets using the optimal configurations learned through the optimization framework.

| | Method | Split-MNIST | Split-CIFAR-10 | Split-CIFAR-100 |
|---|---|---|---|---|
| Baseline | iid-offline | $99.46 \pm 0.08$ | $95.51 \pm 0.22$ | $80.89 \pm 0.75$ |
| | Fine-Tune | $97.29 \pm 0.96$ | $62.56 \pm 5.76$ | $47.54 \pm 1.61$ |
| Continual Learning Memory-free | Online EWC | $98.46 \pm 0.47$ | $59.98 \pm 2.27$ | $20.24 \pm 1.23$ |
| | SI | $97.95 \pm 0.70$ | $65.71 \pm 1.79$ | $33.00 \pm 2.59$ |
| | LwF | $99.26 \pm 0.09$ | $63.47 \pm 1.52$ | $19.45 \pm 1.19$ |
| Continual Learning Memory-based (Buffer=0.5k) | A-GEM | $99.16 \pm 0.08$ | $72.02 \pm 1.29$ | $38.39 \pm 1.98$ |
| | iCaRL | $98.40 \pm 0.09$ | $82.01 \pm 0.76$ | $50.56 \pm 0.23$ |
| | GSS | $97.80 \pm 0.80$ | $86.38 \pm 1.21$ | $56.86 \pm 1.68$ |
| | RPSNet | – | 67.0 | 40.1 |
| | DER++ | $99.29 \pm 0.02$ | $85.95 \pm 1.62$ | $58.28 \pm 1.50$ |
| | NNA-ST (INEL) | $65.13 \pm 0.66$ | $81.05 \pm 0.15$ | $23.04 \pm 0.26$ |
| | NNA-ST (MSE ) | $\mathbf{99.56} \pm 0.04$ | $\mathbf{94.43} \pm 0.15$ | $\mathbf{83.17} \pm 0.15$ |

Table 3: Classification accuracy for the task-incremental learning experiments on the Split-MNIST, Split CIFAR-10, and Split CIFAR-100 datasets. The accuracy metrics are reported as mean and standard deviation over 5 repetitions.

the original MNIST dataset (both training and testing splits) consisting of ten digits into five 2-class classification tasks. This defines a continual learning scenario in which the model sees these five tasks incrementally one after the other. The Split-CIFAR-10 is analogous to Split-MNIST, while for the split-CIFAR100 we follow Mai et al. (2021); Lee et al. (2020); Chen et al. (2020) and construct 10 tasks each with 10 classes. Additional experimental details are discussed in Appendix A.3, and hyperparameter optimization for NNA is discussed in Appendix A.4.

#### 5.2.1 Task-Incremental Learning

To test the architecture's performance for task-incremental learning, we transferred the optimal configurations for the single-task learning case (*NNA-ST*) to the Split-MNIST Farquhar & Gal (2018), Split-CIFAR-10, and Split-CIFAR-100 continual learning benchmarks Shin et al. (2017); Zenke et al. (2017); Nguyen et al. (2017). For the specific case of MNIST, we also included the case of INEL and MSE to evaluate the impact of the inelastic plasticity rule on the overall performance. We compared the accuracy of our approach with baseline, memory-free and memory buffer-based continual learning approaches, as well as with **Fine tuning**, where the model is continuously trained upon arrival of new tasks without any strategies to avoid catastrophic forgetting. As baselines, we consider **iid offline**, where the model is evaluated in a noncontinual learning scenario and trained with multiple passes through the data, which are sampled iid. Among the memory-free approaches, we choose **Online EWC** Schwarz et al. (2018), **SI** Zenke et al. (2017), and **LwF** Li & Hoiem (2017); and among the memory-based methods we compare with **A-GEM** Chaudhry et al. (2018), **iCaRL** Rebuffi et al. (2017), **GSS** Aljundi et al. (2019b), **RPSNet** Rajasegaran et al. (2019), , and **DER++** Buzzega et al. (2020) to cover the most recent works. In all these models, we ensured that the hyperparameters used are consistent and report results averaged across 5 runs. For RPSNet, we report the metrics from the paper Chen et al. (2020), since the experiments were run with the same hyperparameters as those used for other reported approaches in this work. The comparison is shown in Table 3, where we find that our approach obtains an accuracy of 99.56, 94.43, and 83.17 on MNIST data, Split-CIFAR-10, and Split-CIFAR-100, respectively. These experiments show that architectures based on local learning rules can outperform memory-free and memory-based approaches in the task incremental learning scenario, even though learning is confined to a single layer and without any explicit memory replay. Furthermore, we ran the Supermasks in Superposition model Wortsman et al. (2020) on Split-Cifar-100 (considered in their work) using hyperparameters from this work and found that its accuracy was 12.67, which is much lower than our approach. In addition, since the feature extraction layer of our NNA approach for the Split CIFAR-10 and Split CIFAR-100 experiments uses pretrained weights (see sec 3.1.1), we evaluated the effect of the pretrained weights on other algorithms in Appendix A.2 and found that they generally do not provide a significant advantage over working directly with the inputs in the online Task-Incremental learning scenario considered in

| | Method | Split-MNIST | Split-CIFAR-10 | Split-CIFAR-100 |
|---|---|---|---|---|
| Baseline | iid-offline | $95.82 \pm 0.33$ | $80.54 \pm 0.63$ | $48.092 \pm 0.90$ |
| | Fine-Tune | $19.68 \pm 0.02$ | $19.19 \pm 0.06$ | $8.32 \pm 0.23$ |
| Continual Learning Memory-free | Online EWC | $19.92 \pm 0.35$ | $16.18 \pm 1.37$ | $4.41 \pm 0.37$ |
| | SI | $19.76 \pm 0.01$ | $17.27 \pm 0.87$ | $5.87 \pm 0.21$ |
| | LwF | $20.54 \pm 0.64$ | $18.53 \pm 0.12$ | $6.93 \pm 0.32$ |
| Continual Learning Memory-based (Buffer=0.5k) | A-GEM | $48.57 \pm 5.26$ | $18.21 \pm 0.16$ | $6.18 \pm 0.20$ |
| | iCaRL | $72.55 \pm 0.45$ | $35.88 \pm 1.43$ | $15.76 \pm 0.15$ |
| | GSS | $54.14 \pm 4.68$ | $49.22 \pm 1.71$ | $11.33 \pm 0.40$ |
| | ER-MIR | $86.60 \pm 1.60$ | $37.80 \pm 1.80$ | $9.20 \pm 0.40$ |
| | CN-DPM | $\mathbf{93.81} \pm 0.07$ | $47.05 \pm 0.62$ | $16.13 \pm 0.14$ |
| | DER++ | $92.21 \pm 0.54$ | $52.01 \pm 3.06$ | $15.04 \pm 1.044$ |
| | ER-ACE | $82.98 \pm 1.79$ | $35.16 \pm 1.34$ | $8.92 \pm 0.25$ |
| | NNA-CIL (INEL+MNIST) | $77.25 \pm 1.02$ | $45.95 \pm 0.90$ | $\mathbf{25.56} \pm 0.69$ |
| | NNA-CIL (INEL+CIFAR10) | $60.82 \pm 2.00$ | $\mathbf{52.55} \pm 2.05$ | $18.87 \pm 0.33$ |

Table 4: Classification accuracy for the class-incremental learning experiments on the Split-MNIST, Split CIFAR-10, and Split CIFAR-100 datasets. The accuracy metrics are reported as mean and standard deviation over 5 repetitions

this work. However, a notable trend was that when adopting the sparse projection layer (with features derived from the pre-trained model) and a linear classification layer mirroring the methodology in our NNA approach, we observe an increase in the accuracy of memory-free approaches and A-GEM, but they still fall short of the best accuracy obtained.

#### 5.2.2 Class-Incremental Learning

We followed the same approach described above for the more challenging class-incremental learning scenario on the Split-MNIST, Split-CIFAR-10, and Split-CIFAR-100 datasets. We employ the configuration obtained by explicitly optimizing the learning rule configuration for class-incremental learning (*NNA-CIL*). In Table 4, we compare the accuracy of our approach in the context of class-incremental learning with baseline, memory-free, and memory buffer-based continual learning approaches. We use the same baselines and memory-free approaches as those adopted in task-incremental case; but for the memory buffer-based, we also compare against **ER-MIR** Aljundi et al. (2019a), **CN-DPM** Lee et al. (2020) and **ER-ACE** Caccia et al. (2022) , which are the most recently proposed approaches. With *NNA-CIL* we see a significant increase in performance (with both configurations obtained by optimizing on Split-MNIST and Split-CIFAR-10 datasets, respectively) on all three datasets. The accuracies obtained on all three datasets are significantly (in some cases, 3X) higher than the accuracies of the memory-free algorithms. In the case of split-CIFAR-10, the resulting accuracy of 52.55 with *NNA-CIL* is also on a par with 52.01 obtained with the memory-based DER++ algorithm, while for split-CIFAR-100, *NNA-CIL*'s accuracy of 25.56 significantly outperforms that of other approaches. In addition, we ran the Supermasks in Superposition model Wortsman et al. (2020) on Split-MNIST (considered in their work) using hyperparameters from this work and found that its accuracy was 40.87, which is much lower than that of our approach. Similarly to the task-incremental learning scenario, we evaluated the effect of the pretrained weights on other algorithms for the Split CIFAR-10 and Split CIFAR-100 experiments in the class-incremental learning scenario (see Appendix A.2). We found that even for this case, the pretrained weights generally decrease the accuracy of the considered continual learning algorithms over working directly with the inputs in the online Class-Incremental learning scenario. One notable exception is that of the iCaRL applied to Cifar-10 but with Resnet-18 model trained using features extracted from the pretrained model, where we observe an improvement in accuracy over training with original data but still falls short of the best accuracy.

The higher performance of the modulated inelastic rule can be attributed to the stabilization of the most significant weights for each class. This rule provides relative stabilization and is sensitive to changes in other synapses reaching the same neurons, where the sensitivity is modulated by hyperparameters. Consequently, since weight consolidation is not critical for the single-task learning, the optimal configurations do not fully exploit this feature, leading to threshold values that are not compatible with those of class-incremental experiments, which require stabilization over longer periods of time. The comparison of weight evolution of class-incremental learning on Split-MNIST data with *NNA-ST* and *NNA-CIL* (Figure 2) further highlights that the design principles necessary for single-task might differ from class-incremental learning. Further discussion can be found in the Appendix A.1.

### 5.3 Transferability Study

Identification of the best configuration for each group of classes (a dataset) is based on the assumption that all the data are available at the beginning of the training procedure. However, for many online learning scenarios, both continual and single-task learning, this configuration might not be known a priori. This raises the question of transfer

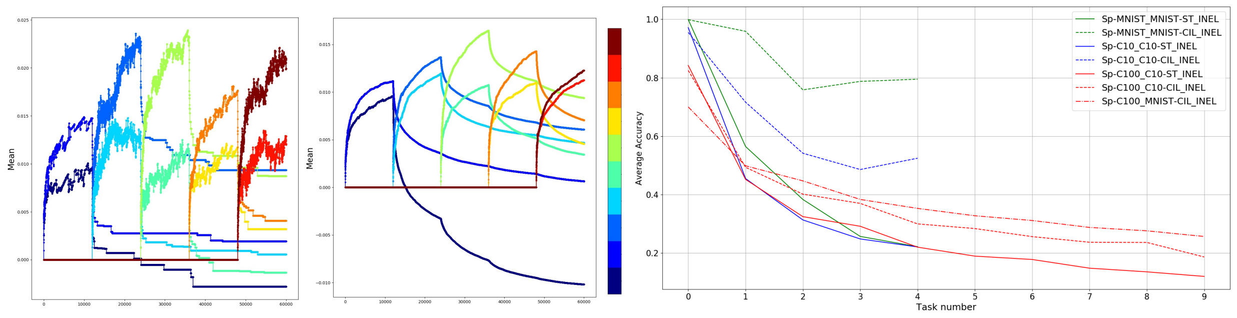

Figure 2: Comparison of the evolution of $W_l^t$ for split MNIST learned in the class-incremental learning scenario using (a) *NNA-ST*+INEL configuration shown on left-most plot and (b) that *NNA-CIL*+INEL shown in middle plot. The right-most plot shows the average accuracy as a function of task for each of the datasets (Split-MNIST, Split-CIFAR10) run with the hyperparameter configuration obtained from NNA-ST and NNA-CIL with INEL rule for their respective data. For Split-CIFAR100, we compare with INEL configurations from Split-CIFAR-10 and NNA-CIL+INEL from Split-MNIST. We see that NNA-CIL configurations consistently have higher average accuracy.

| | Single Task Learning (0.5 epoch) | | | | Task-Incremental Learning | | Class-Incremental Learning | |
|---|---|---|---|---|---|---|---|---|
| Config (↓), Dataset (→) | MNIST | F-MNIST | CIFAR10 | | SplitMNIST | SplitCIFAR-10 | SplitMNIST | SplitCIFAR-10 |
| MNIST (w/INEL) | 96.39 | 85.48 | 72.91 | | 65.25 | 77.63 | 21.56 | 21.19 |
| MNIST (w/MSE) | 96.16 | 80.75 | 69.87 | | 99.60 | 92.50 | 21.84 | 20.33 |
| F-MNIST | 94.52 | 85.13 | 71.29 | | 99.25 | 94.37 | 21.39 | 21.19 |
| CIFAR-10 (w/INEL) | 93.99 | 84.13 | 77.95 | | 64.99 | 81.20 | 20.86 | 22.33 |
| CIFAR-10 (w/MSE) | 94.45 | 84.21 | 77.37 | | 99.03 | 94.65 | 21.75 | 22.85 |

Table 5: Accuracy due to transfer metalearning from single task online learning to task-incremental and and class-incremental continual learning experiments.

metalearning: *how to effectively optimize a learning algorithm to learn unknown tasks and data*. To address this problem, we empirically studied the effect of transferring optimal configurations to the continual learning scenario, and across datasets for the single-task learning case.

In Table 5 we show the accuracy resulting from transferring optimal conditions across tasks and across datasets. From these results we can extract several conclusions. First, the transfer from a single task online to task-incremental continual learning is excellent for the MSE rule but not for the inelastic rule. We attribute this effect to the strong role that the synaptic weight distribution plays in the inelastic rule. Second, for the class-incremental case, the accuracies obtained when transferring hyperparameters are significantly lower than those achieved when conditions are specifically optimized for class-incremental learning. This result indicates that, at least in memory-constrained situations, learning rules must be highly optimized for class-incremental problems and that we must rely on surrogate data to carry out transfer metalearning.

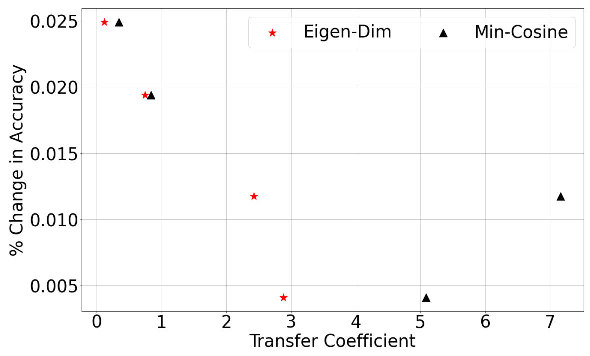

Figure 3: Drop in classification accuracy during hyperparameter configuration transfer as a function of a transfer coefficient obtained for eigen dimension and minimum cosine distance metrics.

**Distance metrics as a measure of transferability:** To rationalize the results shown in Table 5, we have explored the correlation between performance drop across datasets and their effective dimensionality and the distance between datasets. *(a) Eigen dimension*: We consider as a metric of dimensionality the sum of squares of the eigenvalues ($\lambda$) of the covariance matrix of the complete dataset, so that $d = (\sum_i \lambda_i)^2 / \sum_i \lambda_i^2$. This metric has been used in the past to characterize sparse representations Litwin-Kumar et al. (2017). *(b) Cosine distance*: We calculate the distance between two datasets using the minimum cosine distance between any two categories (see details in Appendix A.5).

Figure 3 shows the drop in classification accuracy during transfer metalearning as a function of a *transfer coefficient* obtained from two different metrics: the relative difference in the dimension of the eigenvalue and the minimum cosine distance between the two datasets. Both values are normalized as $D = |M_1 - M_2|/M_1$, where $M_1$ is the metric of the dataset selected to run the experiment and $M_2$ is the metric of the dataset whose optimal configuration is used. Note that this transfer coefficient is not symmetric because of the different normalization, as should be expected, since transfer metalearning is directional. The results clearly show that as the distance between the datasets increases, transfer learning the configurations across them leads to a decrease in accuracy.

## 6 Application to backpropagation-based approaches

In order to evaluate whether the main contributions of bio-inspired networks can be applied to existing backpropagation-based continual learning algorithms, we ported some of the key innovations into a backpropagation-based model described in Section 4.2 and implemented in Avalanche Lomonaco et al. (2021). This allowed us to integrate the networks, optimizers, and loss functions into other existing strategies. In particular, we chose to integrate our model with the Naive and ER-ACE algorithms, which are the worst and best performing approaches across the memory-free and memory-based approaches for the class-incremental learning setting. We found that the accuracy of online Naive (Fine-tune) improved from 19.68 to 41.10 and ER-ACE improved from 82.98 to 86.06 for the class-incremental SplitMNIST benchmark when we introduced the bio-inspired modifications summarized in Table 1. These preliminary results show that the biologically-inspired design patterns are not only effective at mitigating catastrophic forgetting through local learning rules, but also have the potential to improve the backpropagation-based continual learning approaches in the literature. This result is a step forward towards combining the strengths of biological mechanisms with modern deep learning to obtain more accurate and robust continual learning approaches.

## 7 Summary and Conclusions

In this work, we have applied bio-inspired design principles to the problem of memory-free online continual learning. Our architecture relies on local learning rules to carry out single task online and continual learning tasks. We demonstrate that our approach is able to obtain accuracies on par with other SGD-alternative approaches in an online learning scenario. In the task-incremental continual learning scenario, we obtained accuracies exceeding 99%, 94%, and 83% respectively for Split-MNIST, Split-CIFAR-10, and CIFAR-100, thus exceeding the state-of-the-art of memory-based and memory-free algorithms. In class-incremental learning, our architecture clearly outperforms memory-free algorithms in the Split-MNIST, Split-CIFAR-10, and Split-CIFAR-100 tests. It achieves an accuracy of 52.55 and 25.56 for Split-CIFAR-10 and Split-CIFAR-100, respectively, that also exceeds that of memory-based algorithms.

An important fraction of the highest-performing configurations identified in this work relied on a novel inelastic rule that implements a simple form of memory consolidation for synaptic weights that deviate from the pool of presynaptic weights of each neuron. This rule leads to the long-term stabilization of weights that are relevant for a specific class, mitigates catastrophic forgetting. This mechanism can be easily combined with any synaptic plasticity mechanism to stabilize networks learning from non-stationary data. Configurations using this rule achieved accuracies that are three times higher than the best results obtained by using memory-free algorithms in the class-incremental tasks considered in this work. This result highlights the potential of moving beyond conventional local learning rules to fully realize the possibilities afforded by recurrent neural networks to control learning and plasticity in feedforward neural networks.

A key characteristic of our model is that we provide depth to the learning layer through a sparse projection layer that highlights the most relevant feature combinations. This allows us to sidestep the problem of credit assignment when using local learning rules. On the other hand, this also provides a limitation of the proposed approach, since the plasticity for supervised learning is restricted to a single layer in our architecture. In conclusion, our NNA approach's salient design principles include the biologically inspired neural architecture and loss formulation, effectively mitigate catastrophic forgetting. We have demonstrated that these principles can be integrated into existing continual learning algorithms, leading to improved online continual learning. These findings have significant implications for developing intelligent systems that can learn continuously from incoming data streams without catastrophic forgetting.

### Acknowledgments

Primary development of this work was funded by the DARPA Lifelong Learning Machines (L2M) Program. This work has also been partially supported by the DOE Office of Science, Advanced Scientific Computing Research, and SciDAC programs under Contract No. DE-AC02-06CH11357. The computational resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under the DE-AC02-06CH11357 contract, and the Laboratory Computing Resource Center (LCRC) at the Argonne National Laboratory have been utilized for this work.

## REFERENCES

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

Rahaf Aljundi, Lucas Caccia, Eugene Belilovsky, Massimo Caccia, Min Lin, Laurent Charlin, and Tinne Tuytelaars. Online continual learning with maximally interfered retrieval. *arXiv preprint arXiv:1908.04742*, 2019a.

Rahaf Aljundi, Min Lin, Baptiste Goujaud, and Yoshua Bengio. Gradient based sample selection for online continual learning. *arXiv preprint arXiv:1903.08671*, 2019b.

David Alvarez-Melis and Nicolo Fusi. Geometric dataset distances via optimal transport. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21428–21439. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/f52a7b2610fb4d3f74b4106fb80b233d-Paper.pdf.

P. Balaprakash, M. Salim, T. Uram, V. Vishwanath, and S. Wild. DeepHyper: Asynchronous hyperparameter search for deep neural networks. In *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*, pp. 42–51, Dec 2018. doi: 10.1109/HiPC.2018.00014.

Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

Pietro Buzzega, Matteo Boschini, Angelo Porrello, Davide Abati, and Simone Calderara. Dark experience for general continual learning: a strong, simple baseline. *arXiv preprint arXiv:2004.07211*, 2020.

Lucas Caccia, Rahaf Aljundi, Nader Asadi, Tinne Tuytelaars, Joelle Pineau, and Eugene Belilovsky. New insights on reducing abrupt representation change in online continual learning. In *International Conference on Learning Representations*, 2022.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-gem. *arXiv preprint arXiv:1812.00420*, 2018.

Hung-Jen Chen, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Mitigating forgetting in online continual learning via instance-aware parameterization. *Advances in Neural Information Processing Systems*, 33, 2020.

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andre Van Schaik. EMNIST: Extending MNIST to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926. IEEE, 2017.

Anurag Daram, Angel Yanguas-Gil, and Dhireesha Kudithipudi. Exploring neuromodulation for dynamic learning. *Frontiers in Neuroscience*, 14, 2020.

Matthias Delange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Ales Leonardis, Greg Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. *arXiv preprint arXiv:1805.09733*, 2018.

Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in Cognitive Sciences*, 2020.

Toshihide Hige, Yoshinori Aso, Mehrab N. Modi, Gerald M. Rubin, and Glenn C. Turner. Heterosynaptic plasticity underlies aversive olfactory learning in ¡em¿Drosophila¡/em¿. *Neuron*, 88(5):985–998, 2020/07/05 2015. doi: 10.1016/j.neuron.2015.11.003. URL https://doi.org/10.1016/j.neuron.2015.11.003.

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

Khurram Javed and Martha White. Meta-learning representations for continual learning. In *Advances in Neural Information Processing Systems*, pp. 1818–1828, 2019.

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

Soochan Lee, Junsoo Ha, Dongsu Zhang, and Gunhee Kim. A neural Dirichlet process mixture model for task-free continual learning. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJxSOJStPr.

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

Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 21213–21223. Curran Associates, Inc., 2020.

Ashok Litwin-Kumar, Kameron Decker Harris, Richard Axel, Haim Sompolinsky, and L. F. Abbott. Optimal degrees of synaptic connectivity. *Neuron*, 93(5):1153–1164.e7, 2020/06/10 2017. doi: 10.1016/j.neuron.2017.01.030. URL https://doi.org/10.1016/j.neuron.2017.01.030.

Vincenzo Lomonaco, Lorenzo Pellegrini, Andrea Cossu, Antonio Carta, Gabriele Graffieti, Tyler L. Hayes, Matthias De Lange, Marc Masana, Jary Pomponi, Gido van de Ven, Martin Mundt, Qi She, Keiland Cooper, Jeremy Forest, Eden Belouadah, Simone Calderara, German I. Parisi, Fabio Cuzzolin, Andreas Tolias, Simone Scardapane, Luca Antiga, Subutai Amhad, Adrian Popescu, Christopher Kanan, Joost van de Weijer, Tinne Tuytelaars, Davide Bacciu, and Davide Maltoni. Avalanche: an end-to-end library for continual learning. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2nd Continual Learning in Computer Vision Workshop, 2021.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Advances in Neural Information Processing Systems*, pp. 6467–6476, 2017.

Zheda Mai, Ruiwen Li, Jihwan Jeong, David Quispe, Hyunwoo Kim, and Scott Sanner. Online continual learning in image classification: An empirical survey. *arXiv preprint arXiv:2101.10423*, 2021.

Thomas Miconi, Jeff Clune, and Kenneth O. Stanley. Differentiable plasticity: training plastic neural networks with backpropagation. *CoRR*, abs/1804.02464, 2018. URL http://arxiv.org/abs/1804.02464.

Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. *arXiv preprint arXiv:2002.10585*, 2020.

Cuong V Nguyen, Yingzhen Li, Thang D Bui, and Richard E Turner. Variational continual learning. *arXiv preprint arXiv:1710.10628*, 2017.

Erkki Oja. Simplified neuron model as a principal component analyzer. *Journal of Mathematical Biology*, 15(3): 267–273, 1982. doi: 10.1007/BF00275687. URL https://doi.org/10.1007/BF00275687.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 2019.

Roman Pogodin and Peter E. Latham. Kernelized information bottleneck leads to biologically plausible 3-factor hebbian learning in deep networks. *arXiv preprint arXiv:2006.07123, NeurIPS 2020*, 2020.

Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability, 2017.

Jathushan Rajasegaran, Munawar Hayat, Salman H Khan, Fahad Shahbaz Khan, and Ling Shao. Random path selection for continual learning. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32, pp. 12669–12679. Curran Associates, Inc., 2019.

Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. itaml: An incremental task-agnostic meta-learning approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13588–13597, 2020.

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

Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pp. 4528–4537, 2018.

Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pp. 4548–4557. PMLR, 2018.

Hanul Shin, Jung Kwon Lee, Jaehong Kim, and Jiwon Kim. Continual learning with deep generative replay. In *Advances in Neural Information Processing Systems*, pp. 2990–2999, 2017.

Vithursan Thangarasa, Thomas Miconi, and Graham W Taylor. Enabling continual learning with differentiable hebbian plasticity. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2020.

Gido M van de Ven and Andreas S Tolias. Generative replay with feedback connections as a general strategy for continual learning. *arXiv preprint arXiv:1809.10635*, 2018.

Nicolas Vecoven, Damien Ernst, Antoine Wehenkel, and Guillaume Drion. Introducing neuromodulation in deep neural networks to learn adaptive behaviours. *PloS one*, 15(1):e0227922, 2020.

Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33, 2020.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017a.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-Mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017b.

A. Yanguas-Gil, A. Mane, J. W. Elam, F. Wang, W. Severa, A. R. Daram, and D. Kudithipudi. The insect brain as a model system for low power electronics and edge processing applications. In *2019 IEEE Space Computing Conference (SCC)*, pp. 60–66, 2019.

Angel Yanguas-Gil. Memristor design rules for dynamic learning and edge processing applications. *APL Materials*, 7 (9):091102, 2019. doi: 10.1063/1.5109910. URL https://doi.org/10.1063/1.5109910.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 3987–3995. JMLR. org, 2017.

Junting Zhang, Jie Zhang, Shalini Ghosh, Dawei Li, Serafettin Tasci, Larry Heck, Heming Zhang, and C-C Jay Kuo. Class-incremental learning via deep model consolidation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1131–1140, 2020.

## A APPENDIX

### A.1 IMPACT OF CONFIGURATION ON CATASTROPHIC FORGETTING

To better understand why the configurations learned in the single-task scenario transferred well to the Task-Incremental scenario but not to the Class-Incremental case, we studied the evolution with time of the weight matrix ($W_l$) in the neuromodulated learning layer, which has dimensions of $W_l$ of $x_e \times x_o$. More specifically, we tracked the evolution of the mean weight presynaptic to each output neuron: by averaging $W_l$ across $x_e$, we produce a $x_o$ dimensional array for every data sample streamed to the model during learning. We use this to track learning for each class. The evolution of this mean weight is shown in Figure 2 (a) for the Split-MNIST data in the class-incremental learning scenario. The left plot corresponds to the INEL configuration transferred from the single task learning, while the bottom plot represents the evolution of the mean synaptic weight for the configuration learned by optimizing over the class-incremental learning itself. Each task is composed of 12k samples.

After a specific task is concluded, the mean synaptic weights of the classes learned during that task suffer a significant drop, which is consistent with the presence of catastrophic forgetting and lower accuracies. However, the magnitude of this drop is significantly more gradual in the configuration optimized for the continual learning case. This correlates with the good class incremental learning accuracy observed with this configuration. This more gradual decay is a consequence of the inelasticity of the learning rate, which remain zero for weights that differ significantly from the mean in the INEL rule. To evaluate the transferability of the configuration obtained from optimizing class-incremental learning across datasets, we applied this very same configuration, learned on Split-MNIST, to the Split-CIFAR-10 and Split-CIFAR-100 datasets. The resulting accuracies, $45.95$ for Split-CIFAR-10 and $25.56$ for Split-CIFAR-100, outperform most of the memory-free and memory-based methods we compared (Table 4).

Furthermore, we calculate the average forgetting metric (in addition to average accuracy shown in Figure 2) as a function of task id. Average Forgetting at task $\mathcal{T}_i$ is defined as the average loss in accuracy on a task $\mathcal{T}_j$ after training on a sequence of tasks $\mathcal{T}_1 ... \mathcal{T}_i$. Let the test accuracy on task $\mathcal{T}_j$ after learning with task $\mathcal{T}_i$ be $a_{i,j}$, then the average forgetting can be defined as Mai et al. (2021):

$$\text{Average Forgetting}(F_i) = \frac{1}{i-1}\sum_{j=1}^{i-1} f_{i,j}$$
$$\text{where } f_{k,j} = \max_{l\in\{1,\cdots,k-1\}}(a_{l,j}) - a_{k,j}, \forall j < k \quad (9)$$

The $F_i$ values for Split-MNIST, Split-CIFAR10, Split-CIFAR100 trained in the class-incremental learning scenario are shown in Figure 4. We find that the NNA-CIL configuration consistently gives the lowest average forgetting, which was also giving the highest average accuracy.

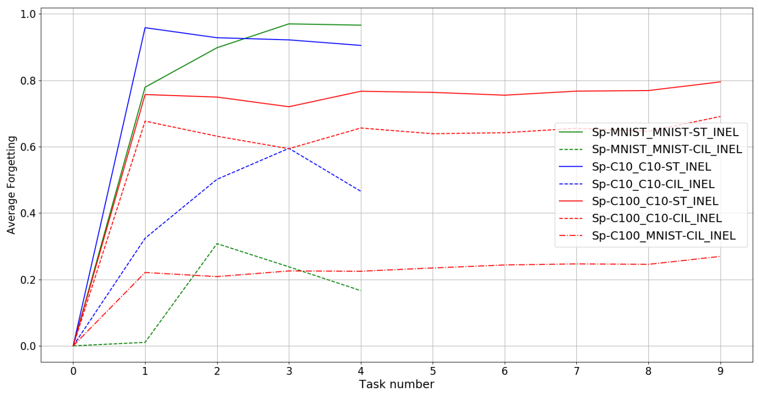

Figure 4: . Average forgetting as a function of task for each of the datasets (Split-MNIST, Split-CIFAR10) run with the hyperparameter configuration obtained from NNA-ST and NNA-CIL with INEL rule for their respective data. For Split-CIFAR100, we compare with INEL configurations from Split-CIFAR-10 and NNA-CIL+INEL from Split-MNIST. We see that NNA-CIL configurations consistently have lower average forgetting.

| | | Task-Incremental Learning | | | | Class-Incremental Learning | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Method | NoPretrain | Scenario 1 | Scenario 2 | Scenario 3 | NoPretrain | Scenario 1 | Scenario 2 | Scenario 3 |
| Continual | Online EWC | $59.98 \pm 2.27$ | $81.73 \pm 0.96$ | $57.36 \pm 3.29$ | $68.42 \pm 2.83$ | $16.18 \pm 1.37$ | $16.61 \pm 0.34$ | $15.68 \pm 1.65$ | $16.56 \pm 0.25$ |
| Learning | SI | $65.71 \pm 1.79$ | $81.92 \pm 0.43$ | $64.00 \pm 4.17$ | $67.81 \pm 1.43$ | $17.27 \pm 0.87$ | $16.88 \pm 0.37$ | $17.70 \pm 0.32$ | $16.82 \pm 0.13$ |
| Memory-free | LwF | $63.47 \pm 1.52$ | $76.82 \pm 0.37$ | $61.75 \pm 1.80$ | $68.10 \pm 1.19$ | $18.53 \pm 0.12$ | $16.67 \pm 0.54$ | $18.39 \pm 0.41$ | $16.51 \pm 0.42$ |
| Continual | A-GEM | $72.02 \pm 1.29$ | $81.49 \pm 0.59$ | $71.77 \pm 3.32$ | $73.72 \pm 1.03$ | $18.21 \pm 0.16$ | $17.16 \pm 0.29$ | $18.29 \pm 0.63$ | $16.88 \pm 0.07$ |
| Learning | iCaRL | $82.01 \pm 0.76$ | $69.64 \pm 0.12$ | $87.84 \pm 0.88$ | $79.16 \pm 0.53$ | $35.88 \pm 1.43$ | $28.62 \pm 0.12$ | $46.57 \pm 2.54$ | $36.63 \pm 0.28$ |
| Memory-based | GSS | $86.38 \pm 1.21$ | $77.38 \pm 0.56$ | $83.75 \pm 1.45$ | $76.95 \pm 0.68$ | $49.22 \pm 1.71$ | $29.35 \pm 1.20$ | $26.28 \pm 1.91$ | $26.46 \pm 0.64$ |
| (Buffer=0.5k) | DER++ | $85.95 \pm 1.62$ | $81.03 \pm 0.23$ | $86.66 \pm 0.85$ | $77.80 \pm 0.14$ | $52.01 \pm 3.06$ | $34.64 \pm 0.73$ | $41.36 \pm 3.22$ | $26.76 \pm 0.68$ |

Table 6: Classification accuracy for the task-incremental and class-incremental learning experiments on the Split CIFAR-10 dataset in three scenarios that characterize the impact of using pre-trained weights on continual learning with the baseline approaches. Accuracy metrics are reported as mean and standard deviation on 5 repetitions.

| | | Task-Incremental Learning | | | | Class-Incremental Learning | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Method | NoPretrain | Scenario 1 | Scenario 2 | Scenario 3 | NoPretrain | Scenario 1 | Scenario 2 | Scenario 3 |
| Continual | Online EWC | $20.24 \pm 1.23$ | $49.60 \pm 0.33$ | $29.54 \pm 3.97$ | $23.60 \pm 1.43$ | $4.41 \pm 0.37$ | $5.22 \pm 0.07$ | $6.86 \pm 0.30$ | $4.95 \pm 0.13$ |
| Learning | SI | $33.00 \pm 2.59$ | $50.04 \pm 0.26$ | $28.33 \pm 3.49$ | $22.28 \pm 0.59$ | $5.87 \pm 0.21$ | $5.25 \pm 0.11$ | $5.39 \pm 0.49$ | $4.50 \pm 0.19$ |
| Memory-free | LwF | $19.45 \pm 1.19$ | $47.01 \pm 0.34$ | $22.36 \pm 1.56$ | $27.80 \pm 0.59$ | $6.93 \pm 0.32$ | $5.43 \pm 0.17$ | $6.87 \pm 0.37$ | $5.01 \pm 0.31$ |
| Continual | A-GEM | $38.39 \pm 1.98$ | $48.18 \pm 0.27$ | $39.79 \pm 2.10$ | $23.93 \pm 1.42$ | $6.18 \pm 0.20$ | $5.1 \pm 0.15$ | $7.10 \pm 0.19$ | $4.93 \pm 0.16$ |
| Learning | iCaRL | $50.56 \pm 0.23$ | $34.57 \pm 0.16$ | $46.70 \pm 0.42$ | $38.56 \pm 0.52$ | $15.76 \pm 0.15$ | $10.04 \pm 0.22$ | $10.72 \pm 0.49$ | $9.78 \pm 0.24$ |
| Memory-based | GSS | $56.86 \pm 1.68$ | $38.17 \pm 0.56$ | $54.27 \pm 1.15$ | $36.22 \pm 1.00$ | $11.33 \pm 1.40$ | $6.88 \pm 0.36$ | $8.57 \pm 0.42$ | $6.91 \pm 0.19$ |
| (Buffer=0.5k) | DER++ | $58.28 \pm 1.50$ | $46.59 \pm 0.43$ | $56.71 \pm 1.24$ | $33.73 \pm 0.53$ | $15.04 \pm 1.04$ | $8.24 \pm 0.40$ | $10.01 \pm 0.79$ | $7.90 \pm 0.60$ |

Table 7: Classification accuracy for the task-incremental and class-incremental learning experiments on the Split CIFAR-100 dataset in three scenarios that characterize the impact of using pre-trained weights on continual learning with the baseline approaches. Accuracy metrics are reported as mean and standard deviation on 5 repetitions.

### A.2 Effect of pretrained weights on the continual learning performance

In this section, we evaluate the impact of using pretrained weights adopted in our Neuromodulated Neural Architecture (NNA) on the other continual learning algorithms evaluated for the CIFAR-10 and CIFAR-100 datasets. We considered the Online EWC, SI, LWF, which are memory-free approaches and the A-GEM, iCarl, GSS, and DER++ which are the memory-based approaches considered in the previous experiments and presented in the main text. We measure the performance of these continual learning algorithms by calculating the classification accuracy in the following scenarios: (1) Utilize the sparse projection layer and a linear classification layer to assess the classification accuracy based on features derived from the pre-trained model, mirroring the methodology in our NNA approach; (2) Bypass the use of original inputs and instead feed the Resnet-18 model with features extracted from the pre-trained model; (3) Replace the Resnet-18 model with a two-layer Multi-Layer Perceptron (MLP) for processing the pre-trained features.

We evaluate the continual learning performance for these three scenarios in both the task-incremental and the class-incremental learning scenarios and the results are shown in Table 6 for CIFAR-10 and Table 7 for CIFAR-100. We performed a grid search to find the hyperparameters that perform the best for each of the algorithms in the task-incremental and class-incremental learning settings. The hyperparameters considered in the grid search for these algorithms are the same as those presented in the work Buzzega et al. (2020), where the range of each parameter is discretized into four values and an exhaustive search is performed to find the best configuration. Finally, we report the accuracy on the test data averaged across five runs.

From these results in Table 6,7 we observe that the pre-trained weights in general do not appear to provide a significant advantage over working directly with the inputs in the online continual learning scenario considered in this work. However, we do see some trends, first one is that specifically in Scenario 1, the memory-free algorithms and A-GEM show increase in the accuracy when using the pre-trained weights for task-incremental learning setting, but they still fall short of the best accuracy obtained on both the CIFAR-10 and CIFAR-100 datasets. Another observed trend is that Scenario 2 provides the highest accuracy for the memory-based approaches in the task-incremental learning setting for both the datasets; however, they also do not surpass the accuracy obtained without pre-trained weights. Finally, we see that the class-incremental learning accuracy consistently decreases across all scenarios considered with pre-trained weights for both datasets except for iCaRL applied to Cifar-10 with scenario 2 where we see significant accuracy gain over No pretrain case but is still falls short of the best accuracy obtained across algorithms in the no pretrain case.

### A.3 Additional Experiment Details

The experiments for Split MNIST use a multilayered perceptron model with two layers and 400 nodes per layer, while the Split CIFAR-10 and Split CIFAR-100 experiments use a ResNet-18 model. The experiments for Split MNIST, Split CIFAR-10 and Split CIFAR-100 datasets using the models iid-offline, Fine-Tune, Online EWC, SI, LwF, A-GEM, iCaRL, GSS, DER++ has been run by adopting the code in `https://github.com/aimagelab/mammoth`. Experiments with ER-MIR have been run by adopting the code in `https://github.com/optimass/Maximally_Interfered_Retrieval`. The experiments with CN-DPM adopted the implementation in `https://github.com/soochan-lee/CN-DPM`. The experiments with OnlineER-ACE adopted the implementation in `https://github.com/ContinualAI/avalanche/`. In all the above models, we ensured that the hyperparameters used are consistent with the online continual learning scenario mentioned above.

## A.4 Architecture Configuration Optimization

The parameter space in the proposed multilayer neuromodulated architecture is composed of categorical variables (e.g., the selection of the local learning rule), integer parameters (e.g., the dimension of the hidden layer) and continuous parameters in each of the learning rules. We adopt Bayesian optimization Balaprakash et al. (2018) to find the high-performing parameter configuration in this mixed search space (categorical, integer, continuous). We used a holdout (validation) set of 10k samples split from the training data to perform the hyperparameter optimization.An example search trajectory showing the evolution of the final accuracy as the algorithm searches multiple configurations for MNIST is shown in 5 . The data, color coded according to the selected learning rule, shows how initially the algorithm searches across the different learning rules, until it quickly settles on the most promising rule. The comparison of the best accuracy obtained for each learning rule among the configurations evaluated in the search for each dataset shown in Table 2.

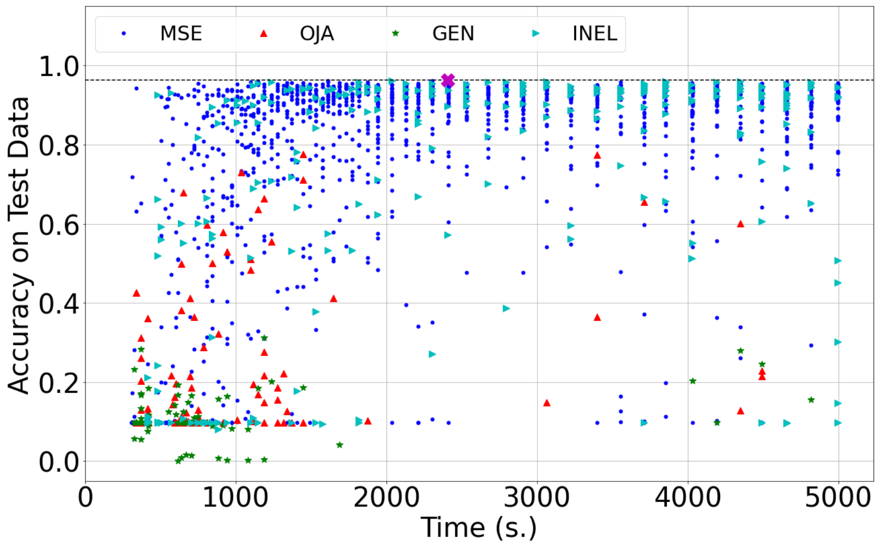

Figure 5: Search trajectory obtained for MNIST dataset from the mixed-integer black-box optimization. The configurations are colored by their evaluated learning rule, with blue, red, green and cyan corresponding to `MSE`, `OJA`, `GEN`, and `INEL` respectively. The accuracy on test data is plotted as a function of wallclock time, which highlights the diversity in the configurations evaluated as the search progressed

## A.5 Transferability Study

The Eigen dimension and cosine distance between the datasets used to generate Figure 3 are shown in Tables 8 and Table 9 respectively. Another recent work Alvarez-Melis & Fusi (2020) characterized the distances between datasets of

| Data | MNIST | F-MNIST | E-MNIST | CIFAR-10 |
|---|---|---|---|---|
| Eigen-Dim | 30.69 | 7.91 | 27.09 | 132.60 |

Table 8: Eigen dimension metric for each of the datasets.

same dimensionality using the geometric dataset distances obtained from optimal transport (OTDD). We observe that the minimum cosine distance presented in Table 9 follows the same trend as the pair-wise distances between MNIST,

| Data | MNIST | F-MNIST | E-MNIST |
|---|---|---|---|
| MNIST | $(0.073, 0.55)$ | $(0.17, 0.54)$ | $(0.044, 0.59)$ |
| F-MNIST | | $(0.012, 0.56)$ | $(0.13, 0.57)$ |
| E-MNIST | | | $(0.098, 0.52)$ |

Table 9: Minimum and maximum cosine distances between centroids of the categories of different datasets

F-MNIST, and E-MNIST obtained from OTDD. Since the cosine similarity considered in this work is restricted to the datasets with same sized inputs, we consider the Singular Vector Canonical Correlation Analysis (SVCCA) Raghu et al. (2017) metric to characterize the distance between inputs of different sizes. The SVCCA metric combines singular value decomposition and canonical correlation analysis, wherein first a singular value decomposition is applied to a dataset to obtain the directions to explain $99\%$ of variance and then a pairwise canonical correlation is calculated between the data that is intended to be compared. Specifically, we calculate this metric between each of the classes in the dataset pairs as done with the cosine distance. However, we use the sum of the off-diagonal elements in the distance matrix normalized with the number of elements for the same datasets and all the elements for distance matrix for dissimilar datasets. The values obtained are shown in Table 10, but we found that it might not be a reliable metric, as all the distance values are nearly the same, with variation only in the last few decimal places. We will explore other similarity metrics in future work.

| Data | MNIST | F-MNIST | E-MNIST | CIFAR-10 |
|---|---|---|---|---|
| MNIST | 0.0488 | 0.0491 | 0.0484 | 0.0531 |
| F-MNIST | | 0.0483 | 0.0486 | 0.0529 |
| E-MNIST | | | 0.0492 | 0.0533 |
| CIFAR-10 | | | | 0.0534 |

Table 10: The SVCCA metric between datasets

### A.6 Ablation Studies

We study the sensitivity of the results to the various components of the proposed Neuromodulated Neuromorphic Architecture through a systematic ablation study and illustrate the results using the MNIST dataset. We first consider the single-task learning scenario with the MNIST-ST (INEL) configuration that corresponds to the accuracy reported in Table 2a for these experiments. In the first experiment, the neuromodulated learning layer is replaced by a multilayer perceptron model with two layers, each with 400 nodes, while keeping the feature extraction layer intact. In this scenario, we found a drop in test accuracy to 37% (with 10 epochs) from the $96.84\%$ obtained with the proposed architecture. Next, we consider both feature extraction and neuromodulated learning layers, but turn off the sparsity imposed by the activity sparsity (Equ. 5). This lead to an accuracy drop to $23.24\%$, which signifies the importance of dynamic thresholding. Finally, we removed the feature extraction layer and directly fed the images to the neuromodulated learning layer. In this case, we obtain an accuracy of $90.72\%$, which underlines the importance of the feature extraction layer (sparse projection + activity sparsity in this work) in improving the accuracy.