CONTINUAL LEARNING WITH DEEP ARTIFICIAL NEURONS

Anonymous authors
Paper under double-blind review

ABSTRACT

Neurons in real brains are complex computational units, capable of input-specific damping, inter-trial memory, and context-dependent signal processing. Artificial neurons, on the other hand, are usually implemented as simple weighted sums. Here we explore if increasing the computational power of individual neurons can yield more powerful neural networks. Specifically, we introduce Deep Artificial Neurons (DANs)—small neural networks with shared, learnable parameters embedded within a larger network. DANs act as filters between nodes in the network; namely, they receive vectorized inputs from multiple neurons in the previous layer, condense these signals into a single output, then send this processed signal to the neurons in the subsequent layer. We demonstrate that it is possible to meta-learn shared parameters for the various DANS in the network in order to facilitate continual and transfer learning during deployment. Specifically, we present experimental results on (1) incremental non-linear regression tasks and (2) unsupervised class-incremental image reconstruction that show that DANs allow a single network to update its synapses (i.e., regular weights) over time with minimal forgetting. Notably, our approach uses standard backpropagation, does not require experience replay, and does need separate wake/sleep phases.

1 INTRODUCTION

Humans and other organisms are capable of acquiring new knowledge over time without completely forgetting the past, an ability commonly referred to as continual learning or learning without forgetting (Beaulieu et al., 2020; Kirkpatrick et al., 2016; Flennerhag et al., 2020; Javed & White, 2019; Li & Hoiem, 2016). Unfortunately, this ability has largely eluded artificial learning systems. Deep neural networks, in particular, are prone to catastrophic forgetting because backpropagation updates all the weights in the network during learning, which causes old features to be overwritten by new ones. Despite this drawback, backpropagation is a powerful technique because it directly correlates weight updates to their impact on the loss function. As noted by (Guerguiev et al., 2019), "any learning system that makes small changes to its parameters will only improve if the changes are correlated to the gradient of the loss function." This insight suggests that any architectural or algorithmic change to enable continual learning must (1) either be compatible with backpropagation or (2) offer an alternative way of assessing the impact of a given weight on performance.

In this paper, we propose Deep Artificial Neurons (DANs), a novel, backpropagation-compatible neural architecture that facilitates continual learning. As we detail below, networks with DAN units can learn over time with minimal forgetting using standard backpropagation and without the need for an experience replay buffer or sleep/wake cycles. Our DAN architecture is inspired by biological neurons, which, unlike their simple artificial counterparts, are extraordinarily complex (Izhikevich, 2007; Palavalli et al., 2020; Guergiuev et al., 2017; Jones & Kording, 2020; Beniaguev et al., 2020; Kandel et al., 2000; Hawkins & Blakeslee, 2004). In particular, the strength of connections between biological neurons is best modeled as a *vector-valued* function between the pre- and post-synaptic neurons. This vector represents the state of chemical concentrations, electrical potentials, proximity, and other temporal and spatial dynamics that affect how strongly a cell responds to its inputs. As we detail below, DANs perform a similar function in our networks; namely, they filter the output signals of neurons in one layer before passing it on to the subsequent layer. As our experiments show, this non-linear, vector-valued filtering allows a network with DAN units to learn over time, using only standard backpropagation, with minimal forgetting.

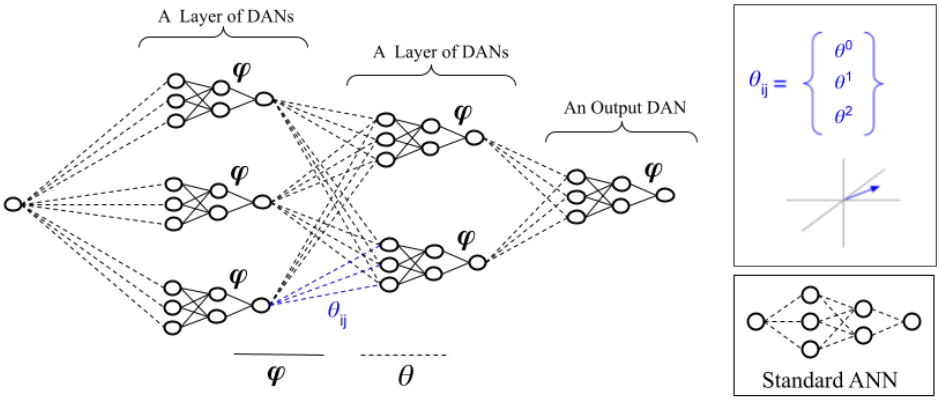

Figure 1: A Network of Deep Artificial Neurons (DANs). DANs are connected to one another by parameters θ, which can be regarded as Vectorized Synapses, or VECs. All DANs may share parameters φ, which we dub a neuronal phenotype. Synapses in a Network of DANs are vectors. The strength of the connection between 2 DANs is therefore an non-linear function of the magnitude and orientation of this synaptic vector.

2 Related Work: Neurons, Meta-Learning, Continual Learning

Many have begun to acknowledge that efficient learning in real brains results from innate priors which have arisen through evolution, and which are generally kept fixed during intra-life deployment (Zador, 2019). It is unsurprising, therefore, that there have been some very recent attempts to embed more powerful priors in artificial neural networks (Guergiuev et al., 2017; Hawkins & Ahmad, 2016; Trabelsi et al., 2017; Beniaguev et al., 2020; y Arcas, NeurIPS, 2019. URL https://slideslive.com/38922302.; Gregor, 2020; Jones & Kording, 2020). In a highly original work, (Mordvintsev et al., 2020) showed that it may be beneficial to think of individual cells, themselves, as networks with self-organizing properties. However, little work has been done to investigate whether this increase in computational power at the cellular level might improve performance specifically in continual learning settings.

Machine learning algorithms generally rely upon the *iid* assumption, which asserts that the training samples are independent of each other (i.e., no correlation between successive samples), and that the training and test distributions need to be (approximately) the same. For this reason, the most reliable technique to date for overcoming catastrophic forgetting (CF) given non-*iid* data is one which explicitly enforces this assumption, often referred to as experience replay (ER) (Mnih et al., 2013). A severe limitation of ER, however, is that it requires a system to retain all (or most) of the data that it encounters. This necessitates additional hardware, increases training time, and can even worsen learning efficiency. Therefore, three categories of alternative techniques have emerged to combat CF in settings where all data cannot be retained: (1) regularization-based approaches (Kirkpatrick et al., 2016; Farajtabar et al., 2019; Lesort et al., 2019), (2) knowledge-compression, or capacity expansion (Joseph & Balasubramanian, 2020; Mandivarapu et al., 2020; Lee et al., 2017; von Oswald et al., 2019), and (3) meta-learned representations and update rules (Beaulieu et al., 2020; Javed & White, 2019; Lindsey & Litwin-Kumar, 2020; Flennerhag et al., 2020; Gregor, 2020).

For this paper, we draw strongest inspiration from a promising approach known as Warped Gradient Descent (WGD) (Flennerhag et al., 2020). WGD mitigates catastrophic forgetting by meta-learning warp parameters ω, realized as warp-*layers*, which are interleaved between the standard layers of a neural network. These warp parameters are held *fixed* during deployment, allowing the rest of the network to learn, without forgetting, using standard backpropagation. As the name implies, parameters ω warp activations in the forward pass, and the gradients in the backwards pass. However, in contrast to WGD, which uses dense fully connected warp layers, we show that it is possible to learn small, common networks φ, constituting shared neuronal phenotypes (DANs) that can be distributed throughout larger networks of plastic Synapses, thereby influencing their learning trajectories.

Figure 2: Replay-free memory retention and transfer during deployment on MNIST (after meta-training on Omniglot). (Left) The model's initial attempt to reconstruct it's input (MNIST digits), after a single step of gradient descent on 0's. (Middle) The model has performed 60 steps of gradient descent on each digit 0-4, seen consecutively, 4's most recently. (RIGHT) Reconstructions of all previously seen digits (0-9) after training on all digits, 9's most recently.

3 MODELS

Deep Artificial Neurons are themselves realized as multi-layer neural networks. Conceptually, we can distribute a single DAN amongst all nodes of a traditional neural network, though less restrictive weight redundancy schemes can be adopted. Figure 1 offers an illustration of how to convert a standard ANN into a network of DANs with n_channels=3. Consider the topology of a standard, fully-connected, feed-forward neural network with l layers of nodes, and let n_l denote the number of nodes in layer l. Let l_0 be a special case, denoting the layer of input nodes, which are *not* DANs. We can convert this topology to a network of DANs in the following way. For each layer of nodes, we instantiate a layer of vectorized Synapses, or VECs, as a standard, fully-connected weight-matrix θ_l with dimensions $n_l \times (n_{l+1} \times n\_channels)$. Feed-forward propagation of a signal along these connections is therefore facilitated in the standard way, by computing the dot product of the activation vector σ_{l_out} from the previous layer and this layer of VECs θ_l. This yields a large input vector $\sigma_{(l+1)\_in}$, to be processed by the DANs in the next layer:

$$\sigma_{(l+1)\_in} = \sigma_{l\_out} \cdot \theta_l = \sum_i^{n_{l+1}} \theta_i^j \sigma_{l\_out}^j + b_i, \forall j \in n_l$$

where j denotes the index of nodes in layer l, and i is the index of nodes in layer $l+1$. When all DANs within a layer share parameters φ, the same DAN model processes each sub-vector, or slice, of $\sigma_{(l+1)\_in}$. In practice, this makes computation very efficient, since the entire inbound Synapse vector $\sigma_{(l+1)\_in}$ can be processed in a single shot by a single DAN. This is done by reshaping $\sigma_{(l+1)\_in}$ to the correct dimensions. In doing so, we abstract the notion of multiple instantiations of the same, common postsynaptic DAN, and improve efficiency. When weight sharing between DANs is not enforced, the input vector to the layer of DANs is sliced into n_{l+1} equally sized sub-vectors, and the output vector of a layer of DANs is obtained by passing each of these separate slices through a separate and unique DAN. This can significantly worsen computational efficiency, and should be avoided. As our experiments show, parameter sharing between DANs not only improves computational efficiency, but acts as implicit regularization, which can even improve performance in certain settings. Lastly, we also use skip connections, inspired by Deep Residual Networks (He et al., 2015) and (Flennerhag et al., 2020), in order to facilitate efficient learning. Skip connections are realized as additional layers of VECs $\theta_{skip(j,k)}$, with dimensions $n_j \times (n_k \times n\_channels)$, which bypass layers of DANs by providing a direct pathway from nodes in layer j to layer k, where $k = j + 2$. We describe the models for each our experiments (see Sec. 4) below.

3.1 INCREMENTAL REGRESSION MODEL

For our incremental regression experiments, we used a network topology of 1 input node, 2 hidden layers of 40 nodes each, and a single output node. Recall that, apart from the single node in the input

layer, each node represents a DAN, and the topology is therefore *converted* to a network of DANs. To this topology, we added 2 skip layers, as described in Section 3: from layer 0 to layer 2, and also from layer 1 to layer 3. The DAN itself is a three-layer neural network with $n\_channels$ input nodes, followed by a hidden layer with 15 nodes, another hidden layer with 8 nodes, and a single output node, parameterized by φ. We applied *tanh* activations to the hidden and output layers of the DAN. For all experiments except that depicted in Figure 4 (Bottom-Right), we set $n\_channels = 40$. For meta-training, we set the learning rate for VECs parameters $\theta = .001$, and the learning rate for DAN parameters $\varphi = .0001$.

3.2 Unsupervised Class-Incremental Learning(UCIL) Model

In order to process images in a manner consistent with Deep Artificial Neurons, we adopt a novel *interpretation* of traditional convolutional layers. The weight sharing scheme employed by convolutional neural networks has historically been criticized as biologically implausible (Pogodin et al., 2021). We argue, however, that this bio-implausibility vanishes when one interprets redundant convolutional weights not as Synapses, but rather as common parameters which govern the behavior of distinct *neurons* of the same *type*. Our assumption is that many distinct neurons, which share common behavioral tendencies, are each responsible for processing different parts of the input. Therefore, we can therefore realize DANs in the input layer of network as stacked convolutional layers. Like other DANs, these convolutional layers are meta-learned and held fixed during deployment, thereby influencing the updates to intermediate Synapses.

The model used to perform UCIL without replay (Figure 2) is comprised of a layer of input DANs, realized as 2 stacked convolutional layers with leaky ReLU activations on each, 3 hidden layers of DANs, with [120, 100, 120] DANs respectively, and a final output layer of 784 DANs, which reconstruct images of size (28x28). The DANs in the hidden and output layers are themselves small convolutional neural networks. These DANs receive $(\sqrt{n_{channels}})$x$(\sqrt{n_{channels}})$ inputs, and generate a single outbound activation signal, or rate. In our experiments, we set $n_{channels} = 100$. They have 2 stacked convolution layers [(1,8,2,1), (8,4,2,1)], followed by a max-pool layer, and 2 fully connected layers [(144, 30), (30,1)]. We again apply leaky ReLU activations to each hidden layer of each DAN. Between each layer of DANs, we instantiate fully connected layers of Synapses. We therefore have 2 fully connected layers for the Encoder [(784, 120*$\sqrt{n_{channels}}$), (120, 100*$\sqrt{n_{channels}}$)], and 2 fully connected layers for the Decoder [(100, 120*$\sqrt{n_{channels}}$), (120, 784*$\sqrt{n_{channels}}$)]. We again employ skip connections, one between the output generated by the first layer of DANs and the final layer of Encoder DANs (784, 100*$\sqrt{n_{channels}}$), and another between the output of the final layer of Encoder DANs and the the final layer of Decoder DANs (120, 784*$\sqrt{n_{channels}}$). For our Unsupervised Class-Incremental Learning experiments, we meta-learned a Single DAN type per layer, i.e. weight sharing was not enforced between DANs in different layers.

4 Problem Formulations

4.1 Incremental Regression

In our initial experiments, we considered the problem of incremental non-linear regression, wherein a model must try to fit to a complete function, when exposed to data from only part of that function in distinct time-intervals, similar to an experiment proposed in (Flennerhag et al., 2020; Finn et al., 2017). That is, the model must learn the complete function in a piece-wise, or *incremental* manner, since it cannot revisit data to which it was exposed during previous intervals. As in (Flennerhag et al., 2020), we split the input domain $[-5, 5] \subset \mathbb{R}$ into 5 consecutive sub-intervals, which correspond to 5 distinct tasks. Task_1 therefore corresponds to the sub-function falling within $[-5, -3)$; Task_2 corresponds to the sub-function within $[-3, -1)$, and so on. The model is exposed to Tasks 1 through 5 in sequential manner. During each sub-task, the network is exposed to 100 data points, drawn uniformly from the current task window. That is, during Task_1 the model performs 100 updates on data sampled from $[-5, -3)$. In our experiments, the model performs 1 update on every sample, equating to a batch size of 1. Sub-tasks are thus defined by their respective windows in the input domain. We slightly modify the target functions used in (Flennerhag et al., 2020). We define a task sequence by a target function that is a mixture of two sine functions with varying amplitudes, phases, and x-offsets. At the beginning of each meta-epoch, we randomly sample two amplitudes

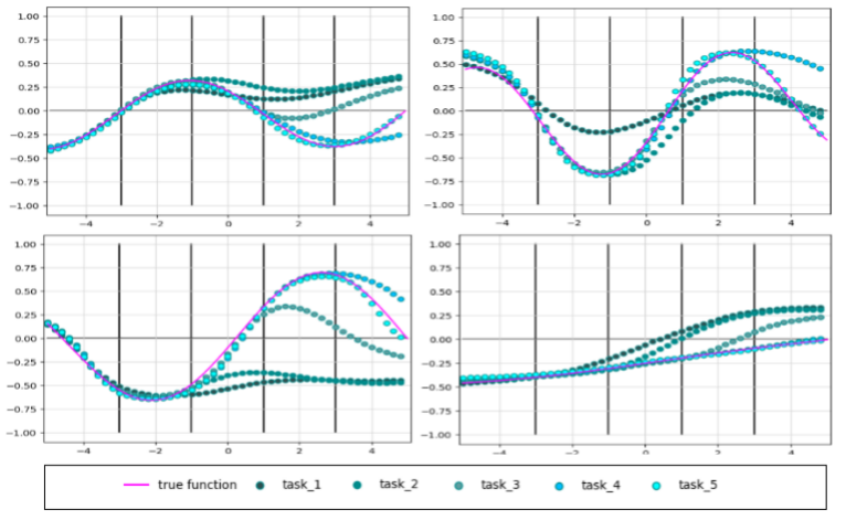

Figure 3: Continual Learning (Incremental Regression) during deployment of 4 non-linear functions, each divided into 5 sub-tasks. The model retains a good fit over the whole function even when it learns these sub-tasks in a sequential manner.

$\alpha_{(0,1)} \in$ (0,2), phases $\rho_{(0,1)} \in (0, \pi/3)$, and x-offsets $\phi_{(0,1)} \in [-5, 5]$. Summing two such sine functions yields a target function of the form: $y = \alpha_0 sin((\rho_0 x) + \phi_0) + \alpha_1 sin((\rho_1 x) + \phi_1)$.

4.2 Unsupervised Class-Incremental Learning

We also formulated an Unsupervised Class Incremental Learning experiment in order to assess the ability of our approach to scale to images. Under this paradigm, an autoencoder must learn to reconstruct an entire dataset, D_T, comprised of s samples from each of C classes. Crucially, the model is only allowed to perform k steps of gradient descent on each class, and the classes are encountered consecutively. In other words, at every time step the model must retain the ability to reconstruct all previously seen classes, as well as attempt to reconstruct instances from new classes.

5 Methodology

Our meta-learning algorithm relies heavily upon two key design choices: (1) we implement a meta-update after *each* and *every* inner-loop step, which we find to greatly improve meta-training efficiency; and (2) we train a *population* of models, which all share common DANs, but which all have unique *Synapses*. Since we meta-train on sequences of non-iid data, it is necessary to account for the fact that the Synapses themselves also evolve in a non-iid way. In other words, both the data and the Synapses influence what the DANs are exposed to (i.e. their training data). This is crucial to understand since DANs themselves must be trained under the *iid* assumption, even though we wish to optimize their performance on *non-iid* datasets. To enforce the *iid* assumption during meta-training, we compute and average the meta-gradients (for DANs) over this population of models, each trained on their own, unique *non-iid* dataset. Note that computing a single meta-update after an entire inner-loop has terminated, as is done in many competing meta-learning approaches, would also satisfy the *iid* assumption. However, we found it to be dramatically more efficient to adopt this population based approach, which *allows* meta-updates after each and every inner-loop step. The full meta-training procedure for our population based approach to continual learning without replay is outlined in Alg. 1. After meta-training is completed, the model is *deployed* on unseen tasks. DAN parameters φ are held ***fixed***, and the model is obligated to learn continually using standard backpropagation to update Synapses.

During meta-training for Incremental Regression, we randomly sample target functions of the form defined in Section 4, and use a population of M=1 models. The target functions are split into 5 sub-tasks, and sub-tasks are encountered sequentially. When we deploy the model, it is evalu-

ated on held-out functions of similar form, not seen during meta-training. For Unsupervised Class-Incremental Learning, we meta-train on sequences of 10 classes, randomly sampled from the Omniglot dataset (Lake et al., 2015), before eventually deploying on MNIST (Deng, 2012), and use a population of M=3 models.

Algorithm 1 Meta-Learning Neuronal Phenotypes for Unsupervised Class-Incremental Learning

Require: $p(\mathcal{T})$: Meta-Training task distribution
Require: M: Population of Models
Require: α, γ: learning rate hyper-parameters, for θ and φ respectively
Require: inner_steps: number of inner loop steps
for model m in M **do**
  $\theta \leftarrow \theta_0, \varphi \leftarrow \varphi_0$: randomly initialize all models to the same initial state
end for
while not done **do**
  Sample Task Sequences $\mathcal{T}_m \sim p(\mathcal{T})$ for all models $m \in M$
  for sub-task i in $\mathcal{T}_m$ **do**
    for step t in inner_steps **do**
      for model m in M **do**
        Compute $Task\text{-}Loss$ $\mathcal{L}_m^i$ w.r.t task $\mathcal{T}_i$
        Compute Synapse gradients $\nabla\theta_t^m$ via Backprop, w.r.t. $\mathcal{L}_m^i$
        Update Synapses: $\theta_{t+1}^m \leftarrow \theta_t^m - \alpha\nabla\theta_t^m$
        Sample Random Task Batch $\mathcal{T}_m^G$ from $\mathcal{T}_m$
        Given θ_{t+1}^m, Compute $Meta\text{-}Loss$ $\mathcal{L}_m^G$ w.r.t $\mathcal{T}_m^G$
        Compute DAN gradients $\nabla\varphi_t^m$ via Backprop, w.r.t. $\mathcal{L}_m^G$
      end for
      Average & Update DANs in all models: $\varphi_{t+1}^M \leftarrow \varphi_t^M - \gamma\nabla\varphi_t^M$
    end for
  end for
  $\theta \leftarrow \theta_0$: reset Synapses to initialization
end while

More formally, let the Historical Learning Trajectory $\mathcal{H}_t$ represent the dataset $[x_0, x_1, \ldots, x_t]$ comprised of all data encountered by the system, prior to and including timestep t; and let the Future Learning Trajectory $\mathcal{F}_t$ represent the dataset $[x_{t+1}, x_{t+2}, \ldots, x_{k*C}]$ comprised of all data *not* yet encountered by the system, where k is the number of steps per sub-task, and C is the number of sub-tasks. Let $\mathcal{T}_m$ therefore represent the full sequence of tasks for a given model m in a population of models M: $\mathcal{T}_m$ = $(\mathcal{H}_t \cup \mathcal{F}_t)$. We have VECs, parameterized by θ, and DANs, parameterized by φ, which together define the complete Model. Note that since DANs are distributed throughout the network, the gradients for VECs ∇_θ depend on parameters φ, and that the meta-gradients ∇_φ depend on parameters θ. This is true, since each set of parameters θ and φ are factors of both gradients. We can therefore define a Model State at timestep t as $(\theta_t\varphi_t)_m$. During meta-training, we iterate by alternating the updates to each set of parameters.

When exposed to a new sample at timestep $t+1$, a given model will result in a measurable Task-Loss with respect to the current data sampled from current sub-task i: $\mathcal{L}_{\mathcal{T}_i}^m$. For Incremental Regression, let the Task Loss $\mathcal{L}_\mathcal{T}$ be the mean-squared error between the model's predictions and the true target data: $\mathcal{L}_{\mathcal{T}_{t+1}}^m = (f_{\theta\varphi}(x_{t+1}) - y_{true})^2$. For Unsupervised Class-Incremental Learning, let $\mathcal{L}_{\mathcal{T}_{t+1}}^m = (f_{\theta\varphi}(x_{t+1}) - x_{true})^2$. The gradient w.r.t $\mathcal{L}_{\mathcal{T}_{t+1}}^m$, over the whole model, is therefore: $\nabla_{\theta_t\varphi_t}\mathcal{L}_{\mathcal{T}_{t+1}}^m$. To clarify notation, we can factor this gradient into its distinct components, but this should not be confused with a multiplication operation:

$$\nabla_{\theta_t\varphi_t}\mathcal{L}_{\mathcal{T}_{t+1}}^m = \nabla_{\theta_t}\nabla_{\varphi_t}\mathcal{L}_{\mathcal{T}_{t+1}}^m$$

This is desirable since we may want to assign separate learning rates to each set of parameters. For instance, let α denote the learning rate for parameters θ, and let γ denote the learning rate for parameters φ. Since we alternate updates to each set of parameters, we first perform an inner-loop step, and update Synapses at time-step $t+1$, like so:

$$\theta_{t+1}\varphi_t \leftarrow \theta_t\varphi_t - \alpha\nabla_{\theta_t}\mathcal{L}_{\mathcal{T}_{t+1}}^m$$

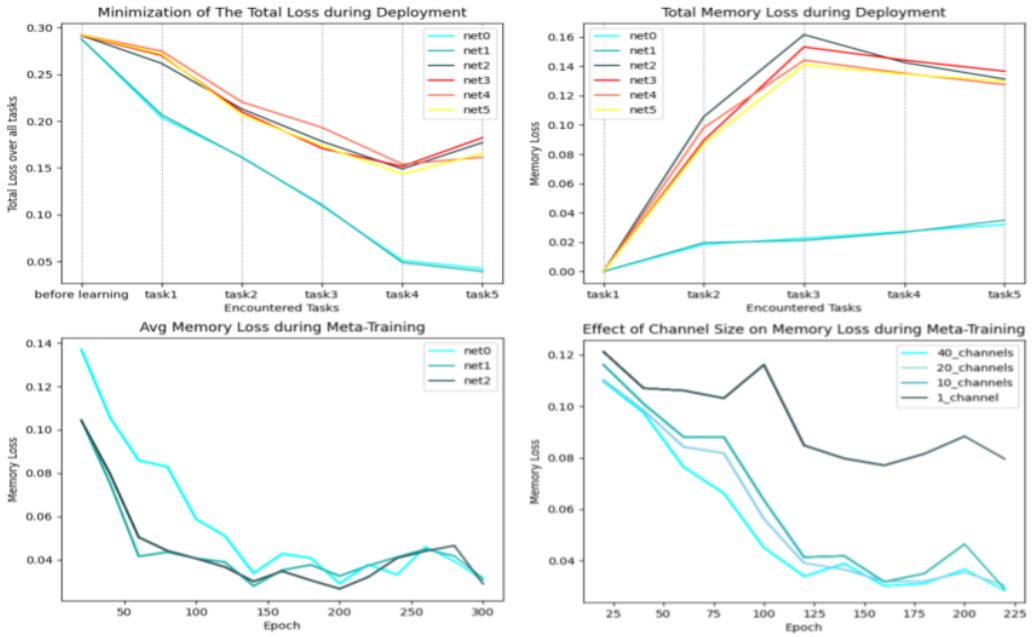

Figure 4: Incremental Regression Ablation Studies - See text for model definitions. (Top-Left) Meta-Trained DANs successfully minimize the Total Loss (averaged over all tasks) as Continual Learning progresses during deployment. (Top-Right) Meta-Trained DANs successfully minimize the Memory-Loss experienced during deployment. (Bottom-Left) A single DAN-type can successfully minimize the Memory-Loss during Meta-Training. (Bottom-Right) Vectorizing the connections between pairs of neurons has a clear impact on the efficiency with which the network learns to minimize the Memory Loss during Meta-Training.

This results in the new Model State $\theta_{t+1}\varphi_t$. Note that this Synaptic update, $\theta_{t+1}\varphi_t \leftarrow \theta_t\varphi_t$, may have caused forgetting over $\mathcal{H}_{t+1}$, which now includes the latest data sample x_{t+1}, and over-fitting to the current sub-task. We therefore wish to quantify and minimize a composite Meta-Loss composed of two terms: the Memory Loss $\mathcal{L}_M^{\mathcal{H}_{t+1}}$, and the Transfer Loss $\mathcal{L}_T^{\mathcal{F}_{t+1}}$: $\mathcal{L}_{\mathcal{M}} = (\mathcal{L}_T^{\mathcal{F}_{t+1}} + \mathcal{L}_M^{\mathcal{H}_{t+1}})$. In practice, this is easily done by sampling a random batch of tasks from the full task-sequence $\mathcal{T}_m^G$ and averaging the a Task-Loss over each: $\mathcal{L}_{\mathcal{M}}^m = \mathcal{L}^{\mathcal{T}_m^G}$. In order to ensure that DANs are in sync and shared by all models in the population at every time-step, we average the DAN gradients over the population, and update all DANs in a single population-wide update. (Note that in our Incremental Regression experiments, we optimized only with respect to the Memory Loss, which explains why transfer is not exhibited in Figure 3).

To clarify our intuition, we seek an optimal neuronal phenotype, defined by a single parameter vector φ^*, shared by all DANs, which would have resulted in the least amount of forgetting over $\mathcal{H}_{t+1}$, and the most improvement over $\mathcal{F}_{t+1}$. Said another way, had the original state of the model been $\theta_t\varphi_t^*$, instead of $\theta_t\varphi_t$, then the inner loop update would have been:

$$\theta_{t+1}^*\varphi_t^* \leftarrow \theta_t\varphi_t^* - \alpha\nabla_{\theta_t}\mathcal{L}_{\mathcal{T}_{t+1}}^m$$

By minimizing our composite meta-objective, we reduce the Meta-Loss experienced at $\theta_{t+1}\varphi_{t+1}^*$ and bias the model towards an optimal state from the outset. In other words, the prior over DAN parameters influences the learning trajectory of Synapses. By taking a step towards φ_{t+1}^*, we explicitly bias the model, via a prior, towards an approximation of the optimal configuration:

$$\varphi_{t+1} \leftarrow \varphi_t - \gamma(\tfrac{1}{m})\sum^M \nabla\varphi_t\mathcal{L}_m^G \;\forall\; m \in M \; s.t. \; \varphi_{t+1} \approx \varphi_t^*$$

At the end of each Outer-Loop epoch, the Synapses of each model are reset, and new task sequences are sampled for the new population before repeating the process described above.

6 EXPERIMENTS AND RESULTS

6.1 INCREMENTAL REGRESSION WITHOUT REPLAY ON NON-IID DATASETS

We performed experiments on tasks defined in Section 4. Figure 3 shows the ability of a meta-trained model to learn continually during ***deployment***; when it encounters tasks in a sequential manner, and is obligated to retain a good fit over previous sub-tasks, even though it is exposed to data from each task only once. In this plot, each uniquely colored scatter plot depicts the predictions of the model over the whole function after performing 100 updates on data from the current sub-task only. The darkest plot represents the model's predictions over the whole function after training only on task_1 $[-5, -3)$. Next, the model performs 100 updates on data from task_2 only $[-3, -1)$. The lightest plot (cyan) depicts the model's predictions after the last round of learning: 100 updates on data from task_5 $[3, 5]$.

Figure 4 depicts various ablation studies that were performed to assess the impact of key design choices employed during both meta-training and deployment. The Model definitions associated with Figure 4 (Top Row) are as follows: **net0** uses a single, meta-learned phenotype, shared by all DANs, fixed during deployment; **net1** uses the same meta-learned single phenotype as **net0**, but it is fully plastic during deployment (updates to the φ are allowed); **net2** uses a random, shared phenotype, fixed during deployment; **net3** uses a random, shared phenotype, fully plastic during deployment; **net4** uses random, but completely unique DANs (no parameter sharing), fixed during deployment; **net5** uses random, but unique DANs, fully plastic during deployment. In Figure 4 (Bottom-Left), we sought to isolate the effect of using a single set of parameters for DANs in the whole network. To do this, we compared 3 models: one which used a single parameter vector for all DANs (**net0**: a single phenotype throughout the network), another which used a separate parameter vector for each *layer* of DANs (**net1**: phenotypes unique to each layer), and a third which did not enforce any parameter sharing amongst DANs (**net2**; 81 unique DANs in the network). We also investigated the effect of the size of $n\_channels$ on the ability of the model to minimize Memory-Loss during meta-training. As seen in Figure 4 (Bottom-Right), as the number of connections between pairs of DANs grows, the speed with which the Memory-Loss is minimized is increased. In other words, vectorized connections accelerate optimization of our meta-objective.

6.2 UNSUPERVISED CLASS-INCREMENTAL LEARNING WITHOUT REPLAY ON NON-IID IMAGE DATASETS

Finally, Figure 2 demonstrates the ability of DANs to facilitate replay-free memory retention and strong ***transfer*** during the challenging setting of Unsupervised Class Incremental Learning. In this experiment, a population of 3 models was meta-trained on a distribution of 10-task sequences sampled from the Omniglot dataset, as described in Algorithm 1. After meta-training, the model was then deployed on MNIST, and was obligated to learn to reconstruct $s = 5$ instances of all 10 digits, seen consecutively. That is, during deployment, the model is allowed to learn 0's, before moving on to 1's, and then 2's, and so on. The model was allowed to perform 60 steps of gradient descent on the s samples from each class, where each class is seen only once, and the model is **not** permitted to revisit any old data. The middle image in Figure 2 shows that the model improves at reconstructing unseen digits as it learns to reconstruct each current digit, i.e. it exhibits forward transfer. Note that our algorithm explicitly encourages transfer by quantifying it during computation of the meta-loss. By measuring the model's performance over the full task sequence after each and every inner-loop step, the model is trained to learn information at every step that may improve its performance on future downstream tasks.

7 CONCLUSIONS AND FUTURE WORK

In this work, we offered a framework for thinking about artificial neurons as much more powerful functions which can be duplicated and distributed throughout larger networks of plastic Synapses. We have shown that the right meta-training algorithm can endow these Deep Artificial Neurons with an ability to facilitate multiple meta-objectives during deployment. In the process, we hope to inspire a deeper understanding about the responsibilities of neurons in both artificial neural networks, as well as real brains. In future work, we plan to investigate the potential of DANs in real-world vision and reinforcement-learning settings.

REFERENCES

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

David Beniaguev, Idan Segev, and Michael London. Single cortical neurons as deep artificial neural networks. *bioRxiv*, 2020. doi: 10.1101/613141. URL https://www.biorxiv.org/content/early/2020/03/19/613141.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning, 2019.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Sebastian Flennerhag, Andrei A. Rusu, Razvan Pascanu, H. Yin, and Raia Hadsell. Meta-learning with warped gradient descent. *ArXiv*, abs/1909.00025, 2020.

Karol Gregor. Finding online neural update rules by learning to remember, 2020.

Jordan Guergiuev, Timothy P. Lillicrap, and Blake A. Richards. Towards deep learning with segregated dendrites, 2017.

Jordan Guerguiev, Konrad P. Kording, and Blake A. Richards. Spike-based causal inference for weight alignment, 2019.

Jeff Hawkins and Subutai Ahmad. Why neurons have thousands of synapses, a theory of sequence memory in neocortex. *Frontiers in Neural Circuits*, 10, Mar 2016. ISSN 1662-5110. doi: 10.3389/fncir.2016.00023. URL http://dx.doi.org/10.3389/fncir.2016.00023.

Jeff Hawkins and Sandra Blakeslee. *On Intelligence*. Times Books, USA, 2004. ISBN 0805074562.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. URL http://arxiv.org/abs/1512.03385.

E Izhikevich. Dynamical systems in neuroscience. *MIT Press*, pp. 111, July 2007.

Khurram Javed and Martha White. Meta-learning representations for continual learning. *CoRR*, abs/1905.12588, 2019. URL http://arxiv.org/abs/1905.12588.

Ilenna Simone Jones and Konrad Paul Kording. Can single neurons solve mnist? the computational power of biological dendritic trees, 2020.

K J Joseph and Vineeth N Balasubramanian. Meta-consolidation for continual learning, 2020.

E. Kandel, E.R. Kandel, J. Schwartz, J.H. Jessell, T. Jessell, Professor of Biochemistry, and M.D. Molecular Biophysics Thomas M Jessell. *Principles of Neural Science, Fourth Edition*. McGraw-Hill Companies,Incorporated, 2000. ISBN 9780838577011. URL https://books.google.com/books?id=yzEFK7Xc87YC.

James Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *CoRR*, abs/1612.00796, 2016. URL http://arxiv.org/abs/1612.00796.

Brenden M. Lake, Ruslan Salakhutdinov, and Joshua B. Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. doi: 10.1126/science.aab3050. URL https://www.science.org/doi/abs/10.1126/science.aab3050.

Jeongtae Lee, Jaehong Yoon, Eunho Yang, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. *CoRR*, abs/1708.01547, 2017. URL http://arxiv.org/abs/1708.01547.

Timothée Lesort, Andrei Stoian, and David Filliat. Regularization shortcomings for continual learning, 2019.

Zhizhong Li and Derek Hoiem. Learning without forgetting. *CoRR*, abs/1606.09282, 2016. URL http://arxiv.org/abs/1606.09282.

Jack Lindsey and Ashok Litwin-Kumar. Learning to learn with feedback and local plasticity, 2020.

Jaya Krishna Mandivarapu, Blake Camp, and Rolando Estrada. Self-net: Lifelong learning via continual self-modeling. *Frontiers in Artificial Intelligence*, 3:19, 2020. ISSN 2624-8212. doi: 10.3389/frai.2020.00019. URL https://www.frontiersin.org/article/10.3389/frai.2020.00019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

Alexander Mordvintsev, Ettore Randazzo, Eyvind Niklasson, and Michael Levin. Growing neural cellular automata. *Distill*, 2020. doi: 10.23915/distill.00023. https://distill.pub/2020/growing-ca.

Amrutha Palavalli, Nicolás Tizón-Escamilla, Jean-François Rupprecht, and Thomas Lecuit. Deterministic and stochastic rules of branching govern dendritic morphogenesis of sensory neurons. *bioRxiv*, 2020. doi: 10.1101/2020.07.11.198309. URL https://www.biorxiv.org/content/early/2020/07/11/2020.07.11.198309.

Roman Pogodin, Yash Mehta, Timothy P. Lillicrap, and Peter E. Latham. Towards biologically plausible convolutional networks. *CoRR*, abs/2106.13031, 2021. URL https://arxiv.org/abs/2106.13031.

Chiheb Trabelsi, Olexa Bilaniuk, Dmitriy Serdyuk, Sandeep Subramanian, João Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J. Pal. Deep complex networks. *CoRR*, abs/1705.09792, 2017. URL http://arxiv.org/abs/1705.09792.

Johannes von Oswald, Christian Henning, João Sacramento, and Benjamin F. Grewe. Continual learning with hypernetworks. *CoRR*, abs/1906.00695, 2019. URL http://arxiv.org/abs/1906.00695.

Blaise Aguera y Arcas. Social intelligence, NeurIPS, 2019. URL https://slideslive.com/38922302. URL URLhttps://slideslive.com/38922302.

Anthony Zador. A critique of pure learning and what artificial neural networks can learn from animal brains. *Nature Communications*, 10, 12 2019. doi: 10.1038/s41467-019-11786-6.