Differentiable Hebbian Consolidation for Continual Lifelong Learning

by

Vithursan Thangarasa

A thesis presented to the University of Guelph

In partial fulfillment of requirements for the degree of Master of Applied Science in Engineering

Guelph, Ontario, Canada

© Vithursan Thangarasa, September, 2019
Catastrophic forgetting poses a grand challenge for continual learning systems. It prevents neural network models from protecting previously learned knowledge while learning new tasks in a sequential manner. As a result, neural network models that are deployed in the real world often struggle in scenarios where the data distribution is non-stationary (concept drift), imbalanced, or not always fully available, i.e., rare or novel edge cases. In this thesis, we propose a Differentiable Hebbian Consolidation model which replaces the traditional softmax layer with a Differentiable Hebbian Plasticity (DHP) Softmax that adds a fast learning plastic component to the fixed (slowly changing) parameters of the softmax output layer. Similar to the hippocampal system in Complementary Learning Systems (CLS) theory, the DHP Softmax behaves as a compressed episodic memory that reactivates existing long-term memory traces, while simultaneously creating new short-term memories. We demonstrate the flexibility of our approach by combining our model with existing well-known task-specific synaptic consolidation methods to penalize changes in the slow weights that are important for each target task. We evaluate our approach on the Permuted MNIST, Split MNIST and Vision Datasets Mixture benchmark problems, and introduce an imbalanced variant of Permuted MNIST — a dataset that combines the challenges of class imbalance and concept drift. Our proposed model requires no additional hyperparameters and outperforms comparable baselines by reducing forgetting.
Acknowledgements

This thesis not only represents the countless number of hours doing work in front of the computer screen but also, the many experiences I have encountered at University of Guelph's Machine Learning Research Group (MLRG) and industry internships at Tesla and Uber AI Labs. It would be very difficult to imagine completing this thesis without the help and support of many remarkable individuals and family members who I also wish to acknowledge.

First and foremost, thank you to my parents, Lavanya Thangarasa and Thangarasa Krishnapillai for their sacrifices so I can pursue graduate studies and always pushing me to become the best version of myself! Thank you to my younger sister Luxmi Thangarasa and brother Aharshan Thangarasa, for keeping me grounded through the tough times and looking up to me; reminding me of my responsibilities outside of school and work. Thank you to my family, for always being there and putting up with me for so long.

I am immensely grateful to my MASc supervisor, Graham Taylor, who did an amazing job of keeping me on the right track in my research journey; his experience, guidance, flexibility, patience, feedback, advice, and criticism were an invaluable source of inspiration for me. Thank you Graham for opening my eyes to new stages of opportunity; inspiring me to achieve at a level higher than I thought possible.

My two industry internships have been a formative experience of my MASc for which I am truly thankful for. Thank you to all of the engineers I worked with at Tesla on the Advanced Technology Group, namely Thomas (Yang) Kim, Harsha Malshe and Charly Mwangi, who showed me what can be achieved when a super talented group of individuals come together and work toward a common goal through hardwork and dedication. I sincerely thank all of the scientists I had the priveledge of working with at Uber AI Labs, namely Thomas Miconi, Aditya Rawal, Rui Wang, Joost Huizinga, Kenneth O'Stanley and Jeff Clune, and for welcoming me into your research groups. The experience of working with you has been invaluable and a lot of your work heavily influenced the work done in this thesis.

To my amazing friends from undegrad, Phil DeBoer and Graham Thoms, who pursued grad school with me, thank you for your friendship, the late-night fast food runs and for being there from day one. To
Terrence DeVries, Alaaeldin (Alaa) El-Nouby, Brendan Duke, Angus Galloway, Boris Knyazev, Eu Wern Teh, Michal Lisicki, Griffin Lacey, Dylan Drover, Colin Brennan, Adam Balint, Shamak Dutta and all former and current colleagues in MLRG, thank you for being my academic family. I would like to thank my thesis committee: Dr. Luiza Antonie, Dr. Hadis Karimipour and Dr. Shawki Areibi for the time and effort spent reading this thesis.

I would like to graciously acknowledge financial support from the Vector Institute for Artificial Intelligence, the Natural Sciences and Engineering Research Council of Canada (NSERC), the Canada Foundation for Innovation (CFI) and an Amazon Academic Research (ARA) Award.
# Table of Contents

Abstract ................................................................. ii
Acknowledgements ...................................................... iii
List of Tables ........................................................... vii
List of Figures ........................................................... viii
List of Abbreviations .................................................... xi

1 Prologue ...................................................................... 1
   1.1 Self-Paced Learning with Adaptive Deep Visual Embeddings ............... 1
      1.1.1 Article Details ......................................................... 1
      1.1.2 Context ................................................................. 2
      1.1.3 Contributions .......................................................... 2
      1.1.4 Recent Developments ............................................... 3
   1.2 Differentiable Hebbian Plasticity for Continual Learning ......................... 3
      1.2.1 Article Details .......................................................... 3
      1.2.2 Context ................................................................. 4
      1.2.3 Contributions .......................................................... 5

2 Introduction .................................................................. 6
   2.1 Motivations ............................................................. 8
   2.2 Objectives .............................................................. 9
   2.3 Contributions ........................................................ 10
   2.4 Relationship to Other Work .......................................... 11
   2.5 Organization .......................................................... 11

3 Background .................................................................. 13
   3.1 Continual Lifelong Learning ............................................ 14
      3.1.1 Problem Definition of Supervised Learning ......................... 15
      3.1.2 Problem Definition of Continual Learning ......................... 16
   3.2 Learning and Memory in Biological Neural Networks ......................... 17
      3.2.1 Hebbian Learning ..................................................... 18
List of Tables

6.1 The test accuracy on Permuted MNIST and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials. ........................................ 45
6.2 The test accuracy on the Imbalanced Permuted MNIST and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials. .......... 49
6.3 The test accuracy on SplitMNIST and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials. .............................. 52
6.4 Network architecture used for 5-Vision Mixture Datasets benchmark in 6.4. For convolutional layers, the output size denotes channel size of output. The negative threshold for all of the LeakyReLU nonlinearities were set to 0.2. .............................. 55
6.5 The test accuracy on 5-Vision Mixture and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials. ......................... 56
6.6 The per-task average test accuracy (% higher is better) after learning all tasks on the 5 vision datasets mixture benchmark. nMNIST refers to notMNIST, fMNIST refers to FashionMNIST and C10 refers to CIFAR-10. The results are averaged over 10 trials. ........................ 57
6.7 The average test accuracy (% higher is better) after learning all tasks on each continual learning benchmark, respectively. The results are averaged over 10 trials. ......................... 58
A.1 The average test accuracy (% higher is better) for different initial $\eta$ values after learning all tasks on the Permuted MNIST, Imbalanced Permuted MNIST and Split MNIST continual learning benchmarks, respectively. The results are averaged over 5 trials. .................. 71
List of Figures

3.1 The Complementary Learning Systems (CLS) theory framework copied from (Parisi et al., 2019). ................................................................. 20

4.1 $\Theta^*_1$ denotes the final optimal parameter values of the previously learned task $T_1$ (sky blue) and the training trajectories for three different gradient steps to learn task $T_2$. The training trajectory depicted for No Penalty approach indicates that the loss on task $T_2$ is minimized but information on the previously learned task $T_1$ is destroyed, thus taking a gradient direction in this step leads to catastrophic forgetting. If an $L2$ Penalty is imposed on each parameter, the parameters become constrained too severely and only task $T_1$ is learned, while task $T_2$ is not. However, methods such as EWC, SI and MAS show that the loss on task $T_2$ can be minimized without incurring a large loss on task $T_1$, by leveraging the parameter importance for Task $T_1$ and staying within a region of small error. ............... 27

4.2 A dual-weight summation model based on Hinton & Plaut (1987), where the input from one neuron to the other is an output of the activation produced by summing the slow and fast weights. ................................................................. 32

4.3 A dual-weight multiplicative model based on Gardner-Medwin (1989), where the input from one neuron to the other is an output of the activation produced by multiplying the slow and fast weights. ................................................................. 33

5.1 A multi-layered perceptron (MLP) network with two hidden layers and a final softmax output layer with dual-weight plastic connections (DHP Softmax) which outputs the softmax pre-activations (raw logits) before performing the softmax operation to obtain the predicted probabilities. .............................. 37
5.2 An example of a Hebbian update for the active class, $c = 6 \in y_{1:B}$. Here, we are given the hidden activations of the final hidden layer, $h_{1:B} \in H$. In this example, the multiple hidden activations corresponding to class $c = 6$ (represented by the pink boxes) are averaged into one vector denoted by $\tilde{h} \in \mathbb{R}^{1 \times m}$. This Hebbian update visualization reflects Lines 4-6 in Algorithm 1 and is repeated for each unique class in the target vector $y_{1:B}$.

6.1 In the Permuted MNIST benchmark, the pixels of each digit in the MNIST dataset is permuted with a fixed random permutation, where each task $t > 1$ has its own unique permutation. This generates a sequence of tasks where each dataset $\{D^{(t)}_{\text{train}}\}_{t=1}^{N_t}$ has its own data distribution and the same complexity.

6.2 The average test accuracy on a sequence of 10 Permuted MNIST datasets $T_{n=1:10}$. The addition of DHP in all cases either maintains or improves the model’s ability to reduce forgetting. The error bars correspond to the standard error across 10 trials.

6.3 (left) Hebbian learning rate and decay value $\eta$, (middle) Frobenius Norm of the Hebbian memory traces $\|\text{Hebb}\|_F$, (right) Frobenius Norm of the plasticity coefficients $\|\alpha\|_F$ while training each task $T_{1:10}$.

6.4 Visualization of plastic and fixed weights from Eq. 5.1.

6.5 Distribution of classes in each imbalanced training dataset $\{D^{(t)}_{\text{train}}\}_{t=1}^{10}$ for the respective tasks $\{T_t\}_{t=1}^{10}$ in the Imbalanced Permuted MNIST benchmark.

6.6 The average test accuracy on a sequence of 10 class-imbalanced Permuted MNIST datasets $T_{n=1:10}$. The addition of DHP improves the model’s ability to reduce forgetting. The error bars correspond to the standard error across 10 trials.

6.7 A sequence of $\{T_{t}\}_{t=1}^{5}$ tasks are generated by splitting the MNIST dataset into different binary classification problems (0/1, 2/3, 4/5, 6/7, 8/9).

6.8 The average test accuracy on a sequence of 5 binary classification tasks (0/1, 2/3, 4/5, 6/7, 8/9) from the original MNIST dataset $T_{t=1:5}$. The average test accuracy over all learned tasks is provided in the legend. The error bars correspond to the standard error across 10 trials.

6.9 The 5-Vision Datasets Mixture benchmark which consists of five different vision datasets: MNIST, notMNIST, FashionMNIST, SVHN, CIFAR-10 for tasks $\{T_t\}_{t=1}^{5}$. 

ix
6.10 The average test accuracy on a sequence of 5 different vision datasets $T_{n=1:5}$. The average test accuracy over all learned tasks is provided in the legend. The error bars correspond to the standard error across 10 trials.

A.1 A sensitivity analysis on the Hebb decay term $\eta$ in Eq. 5.2. We show the average test accuracy for different initial values of $\eta$ after learning all tasks on the (left) Permuted MNIST, (center) Imbalanced Permuted MNIST and (right) Split MNIST problems. The shaded regions correspond to the standard error of the mean across 5 trials.
## List of Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AI</td>
<td>Artificial Intelligence</td>
</tr>
<tr>
<td>CCE</td>
<td>Categorical Cross-Entropy</td>
</tr>
<tr>
<td>CL</td>
<td>Continual Learning</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>CLS</td>
<td>Complementary Learning Systems</td>
</tr>
<tr>
<td>DHP</td>
<td>Differentiable Hebbian Plasticity</td>
</tr>
<tr>
<td>DL</td>
<td>Deep Learning</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>ERM</td>
<td>Empirical Risk Minimization</td>
</tr>
<tr>
<td>MSE</td>
<td>Mean-Squared Error</td>
</tr>
<tr>
<td>GAN</td>
<td>Generative Adversarial Networks</td>
</tr>
<tr>
<td>EWC</td>
<td>Elastic Weight Consolidation</td>
</tr>
<tr>
<td>MAS</td>
<td>Memory Aware Synapses</td>
</tr>
<tr>
<td>ML</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>MLP</td>
<td>Multi-Layered Perceptron</td>
</tr>
<tr>
<td>RBM</td>
<td>Restricted Boltzmann Machines</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>SOM</td>
<td>Self-Organizing Maps</td>
</tr>
<tr>
<td>SGD</td>
<td>Stochastic Gradient Descent</td>
</tr>
<tr>
<td>SPL-ADVisE</td>
<td>Self-Paced Learning with Adaptive Deep Visual Embeddings</td>
</tr>
<tr>
<td>SI</td>
<td>Synaptic Intelligence</td>
</tr>
<tr>
<td>VAE</td>
<td>Variational Autoencoder</td>
</tr>
</tbody>
</table>
Chapter 1

Prologue

This prologue presents a brief background of the research that was completed during my MASc, and how the culmination of work lead to this thesis. I also provide additional context to understand the research problem and its significance, as well as the main contributions. Finally, I summarize recent developments that build upon this research.

1.1 Self-Paced Learning with Adaptive Deep Visual Embeddings

1.1.1 Article Details


Personal Contribution: I developed the Self-Paced Learning with Adaptive Deep Visual Embeddings
(SPL-ADVisE) model and Algorithm 2. Also, I conducted all of the experiments on various image classification datasets. I co-wrote the manuscript with Dr. Graham Taylor.

1.1.2 Context

A lot of the success in deep learning is attributed to a large amount of high quality labelled training data. The naive way of training deep neural networks is by sampling randomized mini-batches from a stationary dataset and training the model using stochastic gradient descent and backpropagation. However, this might not be an optimal training curriculum for deep networks. Previous work on curriculum learning (Bengio et al., 2009) and self-paced learning (Kumar et al., 2010) showed that training deep networks from easy to hard samples or gradually increasing the difficulty of tasks (training distributions) can help with convergence to the optimal solution. Self-paced learning allows a neural network to dynamically select the samples to learn from at different stages of training based on the measure of easiness given by the loss as a metric. The dependence on only a single sample importance prior can lead to overfitting on certain classes during sampling. This led to the question of whether the addition of another sample importance prior based on the sample’s diversity can be embedded to achieve an adaptive curriculum creation method that can progressively increase the diversity of the training examples.

1.1.3 Contributions

We introduced SPL-ADVisE which is a method for scheduling data examples in the minibatch setting. Moreover, it is a general sample selection framework that is independent of model architecture or objective, and learns when to introduce certain samples to the neural network during training. To our knowledge, we were the first to leverage metric learning to improve self-paced learning. We showed that biasing samples based on the easiness and true diverseness to select mini-batches shows improvement in convergence to achieve test performance comparable or better than the baselines on MNIST, FashionMNIST, SVHN, CIFAR-10 and CIFAR-100.
1.1.4 Recent Developments

SPL-ADVisE has successfully been extended or taken as inspiration in several recent works. Cheng et al. (2019) take inspiration from SPL-ADVisE and develop a Local to Global Learning (LGL) paradigm which trains the neural network model from fewer categories (local) to more categories (global) gradually within the complete training dataset. They also show that from an information-theoretic perspective the LGL paradigm can make the early stages of training more stable. They also demonstrate that the LGL paradigm outperforms competitive baselines on toy data, CIFAR-10, CIFAR-100 and ImageNet. Also, Joseph et al. (2019) propose a mini-batch selection strategy that is closely related to SPL-ADVisE except their method is based on submodular optimization which can select diverse and informative mini-batches for training deep networks. Chitta et al. (2018) also propose a method that is inspired by SPL-ADVisE, where Chitta et al. (2018) show that curriculum learning can be applied in the semi-supervised setting. Here, the unlabelled data is given the proxy labels provided by a partially trained network and this data is augmented to the training set. Then, by training with an appropriate curriculum (i.e., based on easiness and diversity) on the augmented training set, semantic segmentation networks can overcome domain shift without having to perform feature alignment from source and target domains.

1.2 Differentiable Hebbian Plasticity for Continual Learning

1.2.1 Article Details


Personal Contribution: I extended the framework for differentiable plasticity (Miconi, 2016; Miconi et al., 2018) to continual lifelong learning with feedforward neural networks and developed Algorithm 1 and
the updated quadratic loss function. Also, I conducted all of the experiments on various continual supervised learning benchmarks which contain changing data distributions, class imbalanced data distributions, increasing number of classes, or a combination of all three scenarios. I co-wrote the manuscript with Dr. Graham Taylor.

1.2.2 Context

As progress in deep learning continues to accelerate at an extremely fast pace, some of the recurring questions are:

1. how do we enable neural networks to quickly learn from a few number of examples?
2. how do we enable neural networks to solve a large number of tasks without forgetting?

As humans, we do not start learning from scratch (i.e., random initialization) everytime we need to learn to a new task. Humans are generally able to maintain good performance when learning a large number of tasks in succession (“continual”) over our lifespans. It is obvious that general intelligence is the result of an iterative process of:

• Forming and storing long- and short-term memories.
• Retrieving stored memories and combining them.
• Comparing the previously learned knowledge with the new inputs received.
• Using these recollected experiences in new environments and when learning new skills.

The ultimate challenge is to integrate this process in neural networks so that neural networks can also learn continually without forgetting. In this work, we focus on proposing a method to address question (2) from above. We take inspiration from neuroplasticity in biological brains for lifelong learning
capabilities, specifically to learn how to solve a large number of tasks in a continual manner without forgetting how to solve previously learned tasks.

1.2.3 Contributions

We developed the Differentiable Hebbian Plasticity (DHP) Softmax layer to improve a neural network’s ability to perform continual learning in dynamic environments while alleviating catastrophic forgetting. In addition, we demonstrated the simplicity and flexibility of the model by combining the proposed DHP Softmax Layer with existing task-specific synaptic consolidation methods. The model was demonstrated on two established continual supervised learning benchmarks, Permuted MNIST and SplitMNIST, and we introduced a class imbalanced variant of Permuted MNIST. DHP Softmax outperformed competitive state-of-the-art baselines and helped improved a neural network’s ability to reduce catastrophic forgetting when learning from different input data distributions and new classes in a continual fashion.
The field of artificial intelligence (AI) has witnessed a drastic increase in the possible real world applications due to the emergence of deep learning (DL), a subset of the more broader field of machine learning (ML). These applications include: autonomous self-driving cars, facial recognition systems, automatic machine translation, automatic speech recognition, medical imaging analysis and more. Deep neural networks (DNNs), in particular, have achieved state-of-the-art performance for solving many complex machine learning tasks from object recognition, semantic segmentation, pose estimation, activity recognition, question answering, natural language understanding, speech synthesis and more. This often involves compute intensive training of DNNs using the standard gradient descent and backpropagation (Rumelhart et al., 1986; 1995) on large stationary datasets with millions of training examples or more. In this training process, gradient descent adjusts the weights of the neural network according to the current training data or mini-batches of training samples drawn from the current stationary dataset. However, in real world environments, the distribution of the collected data is often nonstationary, as it changes over time (i.e. daily, weekly, monthly or yearly). As a result, DL models that are deployed for real world applications are subject to concept drift. Therefore, when a pretrained model is trained only on the newly acquired data, it experiences a degradation in performance with respect to the original
data as gradient descent adjusts the weights of the model based on the representations extracted from the new training data distribution. This phenomenon known as catastrophic forgetting or catastrophic interference (McCloskey & Cohen, 1989; French, 1999; Ratcliff, 1990) causes DNNs to “forget” previously acquired knowledge after being trained on new information. Catastrophic forgetting is linked to the “stability-plasticity dilemma”, a well-known constraint for both artificial and biological neural networks. Here, the learning in a parallel and distributed system (i.e. neural network) needs to be able to balance its ability to integrate new knowledge (plasticity) and prevent old knowledge from being forgotten (stability). This poses a key challenge for real world systems that are required to learn reliably from new data in a continuous manner without compromising the knowledge acquired from previous training experiences.

In recent years, there has been a renewed and growing interest in the study of continual learning (Ring, 1994), also called lifelong learning (Thrun & Mitchell, 1995; Thrun, 1998; Hong et al., 2018) to address the problem of catastrophic forgetting. The benefits of the continual learning paradigm are twofold. 1) It is viewed as another important step towards the grand challenge of creating artificial general intelligence (AGI), where agents can learn autonomously and acquire new complex skills and knowledge in a wide range of dynamic environments. 2) Continual learning enables two important properties: adaptability and scalability. Moreover, it allows DNN models to automatically integrate new knowledge from only the recent data (i.e. without retraining on the old and new data) each time a new task needs to be learned.

Humans and animals, however, appear to have this incredible ability to learn a large number of different tasks in a continual fashion without any of them negatively interfering with each other (Grossberg, 2013; Cichon & Gan, 2015). In biological neural networks, synaptic plasticity has been argued to play an important role in learning and memory (Howland & Wang, 2008; Takeuchi et al., 2013; Bailey et al., 2015) and two major theories have been proposed to explain a human's ability to perform continual learning. The first theory is inspired by synaptic consolidation in the mammalian neocortex (Clopath, 2011; Benna & Fusi, 2016) where a subset of synapses are rendered less plastic and therefore preserved for a longer timescale. The general idea for this approach is to consolidate and preserve synaptic parameters that are considered important for the previously learned tasks. This is normally achieved through task-specific updates of synaptic weights in a biological neural network. The second theory is the complementary
learning system (CLS) theory (McClelland et al., 1995; O'Reilly et al., 2014; Kumaran et al., 2016), which suggests that human beings extract high-level structural information and store it in a different brain area while retaining episodic memories.

Recent work on differentiable plasticity (Miconi, 2016; Miconi et al., 2018) has shown that plastic neural networks with “fast weights” that leverage Hebbian learning rules (Hebb, 1949) can be trained end-to-end through backpropagation and stochastic gradient descent (SGD) to optimize the standard “slow weights”, as well as also the amount of plasticity in each synaptic connection. These works use slow weights to refer to the weights normally used to train vanilla neural networks, which are updated slowly and are often associated with long-term memory. The fast weights represent the weights that are superimposed on the slow weights and change quickly from one time step to the next based on input representations. These fast weights behave as a form of short-term memory that enable “reactivation” of long-term memory traces in the slow weights.

In this thesis, we investigate a unifying framework of differentiable plasticity and synaptic memory consolidation in neural networks as a method for alleviating catastrophic forgetting in environments that require continual supervised learning. We believe that this is the first work that shows how gradient-optimized Hebbian plasticity and task-specific synaptic consolidation can be leveraged for forming new memories and retaining old ones to mitigate the forgetting problem in DNNs. Here, we focus our investigation on the effect of dynamically adjusting the strength of synaptic connections in a DNN to minimize interference when learning new tasks. We also study the effect of encoding episodic memory into a DNN to “reactivate” existing memory traces, while simultaneously creating new ones based on the Hebbian learning principle.

2.1 Motivations

Despite the tremendous success that DNNs have showed for solving various complex tasks very well, they are still affected by the catastrophic forgetting problem. While a number of methods and frame-
works have been recently proposed to mitigate its disruptive effects, catastrophic forgetting still poses a major obstacle toward effective continual learning implementations. Additionally, the problem becomes significantly more challenging when performing continual learning with fixed capacity DNN architectures, which is typically the case for DL models deployed in the real world. This thesis is motivated by the need for a method that allows fixed capacity DNN architectures to dynamically form memories of new experiences during training and then consolidate the knowledge for very long timescales when sequentially learning a large number of tasks.

2.2 Objectives

In this thesis, the primary research goal is:

*To study how catastrophic forgetting in neural networks can be mitigated through synaptic memory consolidation and episodic memory, and to implement a novel dual-weight memory model that can perform continual learning.*

This thesis aims to provide a simple and effective method to address the challenges of catastrophic forgetting through the development of a differentiable plasticity plus synaptic consolidation inspired framework. We demonstrate the proposed method on several continual supervised learning problems for object recognition and perform an empirical evaluation on a model’s ability to retain information after sequentially training on a varying number of tasks. Given that there are a large number of different DNN architectures in the DL literature, we focus our investigation toward feedforward neural networks also referred to as multilayer perceptrons (MLP) and convolutional neural networks (CNN), as most of the research on continual learning has focused on MLPs and CNNs (Goodfellow et al., 2014; Kemker et al., 2017).

In addition to these research objectives, this thesis presents several ideas for future work in this field leveraging our proposed framework for continual learning. Firstly, by providing the proposed method as
an open-source PyTorch (Paszke et al., 2017) implementation on GitHub, it allows future researchers to build on this line of research and also enables practitioners to apply it to real-world applications. Secondly, by taking inspiration from the computational neuroscience literature pertaining to learning and memory in biological neural networks, this thesis can open new investigations for their artificial counterpart.

2.3 Contributions

The contributions of this thesis are as follows:

- This thesis contributes to the development of a new softmax layer based on Differentiable Hebbian Plasticity (DHP) referred to as DHP Softmax. We demonstrate its simplicity and flexibility by combining it with recent state-of-the-art synaptic consolidation approaches such as: Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Online EWC (Schwarz et al., 2018), Synaptic Intelligence (SI) (Zenke et al., 2017) and Memory Aware Synapses (MAS) (Aljundi et al., 2018).

- This thesis contributes to the development of a new DNN model that unifies core concepts from Hebbian plasticity, synaptic memory consolidation and CLS theory to enable rapid adaptation to new unseen data, while consolidating synapses and leveraging compressed episodic memories in the softmax layer to remember previous knowledge and mitigate catastrophic forgetting.

- This thesis evaluates the proposed method on several established continual learning benchmark problems: Permuted MNIST (Goodfellow et al., 2013), SplitMNIST (Zenke et al., 2017; Shin et al., 2017; Nguyen et al., 2018) and the 5-Vision Datasets Mixture (Ritter et al., 2018; Zeno et al., 2018). It also introduces a class-imbalanced variant of Permuted MNIST.
2.4 Relationship to Other Work

We extend the work on differentiable plasticity (Miconi, 2016; Miconi et al., 2018) to a continual learning setting and develop a model that is capable of adapting quickly to changing environments as well as consolidating previous knowledge by selectively adjusting the plasticity of synapses. The work on differentiable plasticity was mainly demonstrated on recurrent neural networks (RNN) for solving simple tasks quickly. The proposed softmax layer in this thesis builds upon the Hebbian Softmax layer (Rae et al., 2018) which was mainly demonstrated in the meta-learning setup for simple few-shot image classification and medium-scale word-level language modelling problems. However, the work in (Rae et al., 2018) did not investigate the grand challenge of overcoming catastrophic forgetting in continual learning.

This work draws inspiration from the neuroscience literature where a process known as synaptic consolidation (Clopath, 2011; Benna & Fusi, 2016) enables synapses (i.e. weights) to retain their strengths for long timescales so that relevant memories can be consolidated within a single synapse and new memories cannot interfere with previously consolidated ones. The proposed method also takes inspiration from another biologically inspired strategy for representing memories to overcome the forgetting problem known as Complementary Learning Systems (CLS) theory (McClelland et al., 1995; O’Reilly et al., 2014; Kumarar et al., 2016). Here, learning and memory are modelled by a dual memory system where, structural knowledge is acquired through slow learning via the neocortex and rapid learning via the hippocampus.

2.5 Organization

The remainder of this thesis is organized as follows:

Chapter 3 provides background information on continual learning and its distinction between supervised learning, catastrophic forgetting in neural networks and ways in which biological neural networks perform learning and memory. We begin by providing background information on continual learning,
focused on the benefits of continual lifelong learning, the conventional supervised learning setup and the differences in the continual learning paradigm. We then discuss background on learning and memory in biological neural networks, including a brief overview of Hebbian learning, followed by a discussion of two neurophysiological methods based on task-specific synaptic memory consolidation and Complementary Learning Systems (CLS) theory.

Chapter 4 provides a review of contemporary literature, including prominent plastic neural networks, and concludes with a review of regularization strategies based on synaptic memory consolidation and dual-weight models.

Chapter 5 describes the methodology used in this work. It begins with a high-level discussion of existing methods that use dual-weight memory based on slow and fast weights for long term memory and quick learning. Then, we breakdown the proposed method into different components of our methodology and provide a discussion for each one including: the Differentiable Hebbian Plasticity (DHP) Softmax, Hebbian update rule and updated quadratic loss for Hebbian synaptic consolidation.

Chapter 6 provides results of an empirical evaluation of the proposed Differentiable Hebbian Consolidation model for continual learning on a sequence of supervised image classification tasks. Each continual learning benchmark has its own details regarding the experimental setup, datasets used and hyperparameter settings. We then present the results for each continual learning benchmark and evaluate against state-of-the-art baselines in synaptic memory consolidation.

Chapter 7 provides an overview of the investigations, draws conclusions on learning and memory in fixed capacity neural networks, and suggests directions to open new investigations for future work.
In traditional supervised learning, the most widely used learning rule to train shallow and deep neural networks is stochastic gradient descent (SGD) which uses the gradients computed by the backpropagation ([Rumelhart et al., 1986; 1995]) algorithm to update the existing parameters in the neural network. Learning and adaptation in biological neural networks rely on neural synaptic plasticity which is often described as “the ability of neurons to change in form and function in response to alterations in their environment” ([Kaas, 2001]). Common DNN architectures that have enjoyed recent successes in deep learning such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) are uniformly plastic. Therefore, in most supervised learning methods, these architectures require independent and identically distributed (iid) samples from a stationary training distribution. However, for ML systems in real-world applications that require continual learning, the iid assumption is easily violated when:

1. There is concept drift in the training data distribution.
2. There are imbalanced class distributions and concept drift occurring simultaneously.
3. Data representing all scenarios in which the learner is expected to perform are not initially avail-
In such situations, learning systems face the “stability-plasticity dilemma” which is a well-known problem for artificial and biological neural networks (Carpenter & Grossberg, 1987; Abraham & Robins, 2005). This presents a continual learning challenge for an ML system where the environment is non-stationary and the model needs to provide a balance between its plasticity (to integrate new knowledge) and stability (to preserve existing knowledge). Moreover, when the synaptic plasticity in neural network models is constantly active, it can result in catastrophic forgetting (Robins, 1995).

There have been a wide number of approaches proposed for implementing unsupervised learning in neural networks such as: self-organizing maps (SOM) (Kohonen, 1982), the Helmholtz Machine (Dayan et al., 1995), auto-encoders (Bourlard & Kamp, 1988), variational auto-encoders (VAE) (Kingma & Welling, 2014), Restricted Boltzmann Machines (RBM) (Hinton, 2006; Smolensky, 1986), Hebbian synaptic plasticity (Hebb, 1949; Gerstner & Kistler, 2002; Cooper, 2005), and General Adversarial Networks (GAN) (Goodfellow et al., 2014). The Differentiable Hebbian Consolidation method takes inspiration from Hebbian learning rules given its relationship to plasticity in biological neural networks for continual lifelong learning. Furthermore, there have been a number of approaches in deep learning inspired by memory consolidation in biological neural networks to help solve the “stability-plasticity dilemma”, where memories can be stored and retained based on plastic connections operating at different timescales. Another branch of work to solve the “stability-plasticity” problem is based on CLS theory, where memories are periodically replayed to the neural networks to reinforce existing connections to past memories, thus inducing consolidation of important weights for solving different tasks.

### 3.1 Continual Lifelong Learning

The fundamental characteristic of continual learning is based on the notion that learning is a lifelong experience, where the learner (i.e., a DNN) is continually acquiring new knowledge and information over time (Ring, 1994; Thrun, 1998; Hong et al., 2018). In this setup, the data representing all situations in
which the learner is expected to perform are not initially available, as it is difficult to come up with a single dataset that can cover all possible situations that a learner may encounter over its lifetime. Hence, as new instances are encountered, they are continuously collected in order for the learner to adapt accordingly. As a result, there is a strong need for DNN models that can learn reliably in a continual fashion without compromising on the knowledge acquired from training on previous data. The ultimate goal of continual learning is to enable a neural network model to learn from data as it is fetched in sequential chunks, often enumerated in the continual learning literature as tasks. Here, the primary focus is on developing learning algorithms or strategies that can handle an unlimited sequence of changing data while being bounded computationally, adhering to constraints on memory resources and maximizing a predefined metric for the learning performance over a model’s lifetime. These are the key distinguishing aspects of continual learning when compared to the traditional supervised training setup.

3.1.1 Problem Definition of Supervised Learning

In the supervised learning setup for the $t^{th}$ task $T_t$, the objective of a standard machine learning model (e.g., a DNN) is to learn an approximated mapping $f_t$ to the true underlying function $\tilde{f}_t$, parametrized by $\Theta_t$, from the input training data $\mathcal{X}(t)$ to the target ground truth data $\mathcal{Y}(t)$. Here, each task $T_t$ has its own associated training data $D_{\text{train}}^{(t)} = (\mathcal{X}(t), \mathcal{Y}(t))$, where $\{(x_i, y_i)\}_{i=1}^{n_t} \in \mathcal{X}(t), \mathcal{Y}(t)$ and $n_t$ is the total number of training samples. It is to be noted that the distribution of the training data $D_{\text{train}}^{(t)}$ and set of classes or categories $c \in \mathcal{Y}(t)$ contained in each task $T_t$ can be different from each other. Let us define $D_{\text{train}} = \bigcup_{t=1}^{N} D_{\text{train}}^{(t)}$ for the scenario where a model $f_t(\cdot; \Theta_t)$ is to learn $N$ number of datasets for all tasks $\{T_t\}_{t=1}^{N}$. Similarly, we denote the test dataset as $D_{\text{test}}^{(t)}$ for a task $T_t$, and the concatenated test datasets as $D_{\text{test}} = \bigcup_{t=1}^{N} D_{\text{test}}^{(t)}$.

If each task is treated independently of each other, $f_t$ learns to minimize the loss function:

$$\Theta_t^* = \arg\min_{\Theta_t} \mathcal{L}(\Theta_t; D_{\text{train}}^{(t)}) \quad (3.1)$$
Here, we use Empirical Risk Minimization (ERM) (Langford et al., 2011) to estimate the loss by minimizing Eq. 3.2, where the loss can be the categorical cross-entropy (CCE) loss for classification problems or mean-squared error (MSE) loss for regression problems, for instance.

\[
    L_t(\Theta_t; D_{\text{train}}^{(t)}) = \frac{1}{|D_{\text{train}}^{(t)}|} \sum_{(x_i, y_i) \in D_{\text{train}}^{(t)}} L(f_t(x_i; \Theta_t), y_i) \tag{3.2}
\]

ERM assumes that the samples from the entire training distribution in \(D_{\text{train}}^{(t)}\) are independently and identically distributed (iid). However, in the real-world, the distribution of the collected training data can vary over time. Therefore, employing this risk minimization principle breaks down when the iid assumptions no longer hold true, particularly when a model learns from data in a sequence of tasks with changing distributions. In fact, the naive application of ERM leads to catastrophic forgetting (McCloskey & Cohen, 1989; French, 1999; Ratcliff, 1990), where the learning of a new task impedes or interferes with the acquired knowledge from past tasks.

### 3.1.2 Problem Definition of Continual Learning

In the continual learning setup, the DNN model also learns a mapping \(f_t\), parametrized by \(\Theta_t\), from \(\mathcal{X}^{(t)}\) to \(\mathcal{Y}^{(t)}\). The model \(f_t(\cdot; \Theta_t)\) is trained sequentially on \(N\) tasks, where \(\Theta_t\) are the learned parameters collected up to the most recent task \(T_t\). However, in continual learning, after a model is trained on a task \(T_t\) using its respective training dataset \(D_{\text{train}}^{(t)}\), both \((D_{\text{train}}^{(t)}, D_{\text{test}}^{(t)})\) would not longer be accessible when training tasks from \(T_{t+1}\) to \(T_N\). Moreover, for learning in a continuous manner, we would ideally like \(\Theta_t^*\) (see Eq. 3.1), that is obtained after being trained with data from the \(t\)th task, to also retain the knowledge obtained from observing the training data from \(D_{\text{train}}^{(1)}\) to \(D_{\text{train}}^{(t-1)}\).

Therefore, the primary objective of continual learning is to maximize the test performance of the model \(f_t(\cdot; \Theta_t)\) after observing \(T_t\) while, minimizing the forgetting or interference for tasks from \(T_1\) to \(T_{t-1}\), all tasks being evaluated across the test datasets \(D_{\text{test}}^{(t)}\) where \(1 \leq t' \leq t\). If we assume that we have all of the
training data for tasks \( \{T_t\}_{t=1}^N \) initially available, the objective function that we aim to minimize can be more formally described as:

\[
\Theta_N^* = \arg\min_{\Theta_N} L(\Theta_N; D_{\text{train}}) \tag{3.3}
\]

\[
L(\Theta_N; D_{\text{train}}) = \sum_{t=1}^N L_t(\Theta_t; D_{\text{train}}^{(t)}) \tag{3.4}
\]

However, in the continual learning paradigm, we do not have access to all of the different datasets at the same time. Thus, we cannot directly compute and optimize the objective function shown in Eq. 3.3. The goal and ultimate challenge of continual learning is to remember the learned knowledge \( \sum_{t=1}^{t-1} L_t(\Theta_t; D_{\text{train}}^{(t)}) \), while being able to estimate \( \Theta_t \) by minimizing Eq. 3.2 independently. There have been a number of different approaches proposed in the continual learning literature for solving this major unsolved challenge in DL systems and is more comprehensively discussed in Chapter 4.

### 3.2 Learning and Memory in Biological Neural Networks

Interestingly, humans can learn to perform multiple tasks in succession and adapt to new environments over their lifespan (continual learning) while avoiding interference among them. In biological neural networks, synaptic plasticity has often been argued to play an important role in learning and memory (Howland & Wang, 2008; Takeuchi et al., 2013; Bailey et al., 2015). One of the major theories that have been proposed to explain a human’s ability to learn continually is Hebbian learning (Hebb, 1949; Gerstner & Kistler, 2002). In addition, there have been two major neurophysiological theories proposed in the neuroscience literature suggesting how biological neural networks overcome the catastrophic forgetting problem: 1) synaptic memory consolidation (Clopath, 2011; Benna & Fusi, 2016) and 2) Complementary Learning Systems (CLS) theory (McClelland et al., 1995; O’Reilly et al., 2014; Kumaran et al., 2016). This thesis draws inspiration from the fundamental principles in Hebbian learning, synaptic consolidation.
and CLS theory to develop a model that is capable of forming and retaining memories throughout its lifetime.

### 3.2.1 Hebbian Learning

The synaptic changes of neurons in a neural network are called synaptic plasticity and these changes are thought to be the basis of learning. The Hebbian learning rule (Hebb, 1949) suggests that learning and memory are attributed to weight plasticity, that is, the modification of the strength of existing synapses according to variants of Hebb’s rule (Paulsen & Sejnowski, 2000; Song et al., 2000; Oja, 2008). While often described in lay terms as “cells that fire together, wire together”, it is a form of activity-dependent synaptic plasticity where correlated activation of pre- and post-synaptic neurons leads to the strengthening of the connection between the two neurons. In Donald O. Hebb’s seminal work (Hebb, 1949), he wrote that:

“*When an axon of cell A is near enough to excite a cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A’s efficiency, as one of the cells firing B, is increased.*”

In short, if one neuron fires repeatedly and this neuron results in the firing of another, the strength of the first on the second neuron increases over time. As a result, the likelihood that the first neuron causes the second neuron to fire increases and forms associations between them. The simple form of this postulate can be more formally described as:

\[ \Delta w_{i,j} = \eta \cdot h_j \cdot h_i \]  

where \( w_{i,j} \) is the weight of the connection from \( j^{th} \) neuron to the \( i^{th} \) neuron and its change is proportional to the product between \( h_j, h_i \) which denote the activation levels of neurons \( i \) and \( j \). The Hebbian learning rate is denoted by \( \eta \) and is often a small scalar value. According to the Hebbian learning theory,
after learning, the related synaptic strength and connectivity are enhanced while the degree of plasticity decreases to protect the learned knowledge (Zenke et al., 2017). Hebbian plasticity based computational models can provide more understanding and ways in which neural networks can perform effective learning and memory to enable more complex machine intelligence.

3.2.2 Synaptic Memory Consolidation

Recent work in (Clopath, 2011; Benna & Fusi, 2016) suggest that biological neural networks perform synaptic memory consolidation in the mammalian neocortex where a subset of synapses are rendered less plastic and therefore preserved for a longer timescale. The general idea for this approach is to consolidate and preserve synaptic parameters that are considered important for the previously learned tasks. This is often done via task-specific updates of synaptic weights in a neural network. Previous work from Yang et al. (2009) showed that learning occurs through the formation and elimination of postsynaptic dendritic spines over time. The excitatory synapses are strengthened when a new task is acquired, which results in an increase in the number of dendritic spines of neurons. Thus, as new tasks are learned, the enlarged dendritic spines are able to retain the knowledge acquired from the previously learned tasks. However, a task can also be forgotten if these spines holding information on a particular learned task are “erased”. Essentially, this suggests that the performance on previously learned tasks can be retained by protecting these strengthened synapses, which is a process termed as task-specific synaptic consolidation. In task-specific synaptic consolidation, a proportion of less plastic synapses hold information on how to perform a previously learned task; this manifests to alleviating the catastrophic forgetting problem over time.

3.2.3 Complementary Learning Systems Theory

Another important study is the Complementary Learning Systems (CLS) theory (McClelland et al., 1995; O’Reilly et al., 2014; Kumaran et al., 2016), which suggests that human beings extract high-level structural information and store this information in a different brain area while retaining episodic memories.
This is a class of dual memory models where, the hippocampus and neocortex behave complementary to each other during learning and memory formation, suggesting that this is another mechanism for protecting consolidated knowledge in the mammalian brain. Short-term adaptation is exhibited by the hippocampus which allows for rapid learning of new experiences by moment-to-moment encoding of memories in an episodic spatiotemporal manner. This information is then transferred and combined with the long term memory in the neocortical system.

More specifically, the hippocampal system employs a higher learning rate for fast adaptation and encodes sparse representations of new experiences to minimize interference. The neocortex, however, utilizes a slow learning rate and progressively builds high-level overlapping representations. Therefore, the concurrent learning of the generalities in the environment and specifics through episodic memories is a result of the interactions between the neocortex and hippocampus.

3.3 Summary

In this chapter, we discussed background information on continual lifelong learning, supervised learning, and learning and memory in biological neural networks including Hebbian learning, synaptic mem-
ory consolidation and Complementary Learning Systems Theory. We focused on the problem definition for traditional supervised learning and continual learning. In our problem definition for the continual learning setup, we also discussed catastrophic forgetting in neural networks due to ERM and training with SGD and backpropagation. The background on continual lifelong learning focused on defining and showing how this learning paradigm can enable DNNs to learn continuously and adaptively at longer timescales. We also discuss some of the relevant works in the computational neuroscience literature which demonstrate how the mammalian brain performs learning and memory in a continual manner with minimal interference between new knowledge and previously learned information. Many of the techniques proposed in the continual learning literature take inspiration from such neurophysiological theories.
Chapter 4

Literature Review

This thesis connects i) plastic neural networks with a dual-weight architecture; and ii) continual learning strategies that involve task-specific synaptic consolidation and CLS theory for overcoming catastrophic forgetting. Here we review plastic neural networks for rapid learning and mitigating catastrophic forgetting by decreasing the plasticity of synapses related to previously acquired knowledge, along with compressed episodic memories.

4.1 Neural Networks with Non-Uniform Plasticity

Previous work has shown that small neural networks with non-uniformly plastic connections can be designed with evolutionary algorithms and have outperformed uniformly-plastic networks on various maze-learning tasks (Soltoggio et al., 2008; Risi & Stanley, 2012). Recent approaches in the meta-learning literature for few-shot classification and language modelling problems have shown that we can achieve strong results by binding labels to representations using a form of “memory mechanism” which involves
slow weights and fast weights implemented with Hebbian learning-based associative memory (Munkhdalai & Trischler, 2018; Rae et al., 2018).

Munkhdalai & Trischler (2018) proposed a model that augments fully-connected layers preceding the softmax with a matrix of fast weights to bind labels to representations, where the fast weight matrix behaves as an associative memory. Rae et al. (2018) proposed a Hebbian Softmax layer that can improve learning of rare classes by interpolating between the Hebbian learning update rule and stochastic gradient descent (SGD) updates on the output layer. This method combines the two rules to achieve better initial representations and also preserves these representations for longer time spans since the activations for one class are not competing for space with activations for frequently observed classes. However, this approach relies on an engineered scheduling scheme for annealing plastic and permanent weights to enable slow and fast weight updates and this scheme was designed for a meta-learning setup.

4.1.1 Differentiable Plasticity

Miconi et al. (2018) proposed differentiable plasticity, which uses stochastic gradient descent (SGD) to optimize the plasticity of each synaptic connection. Here, each synapse is composed of a slow weight and a plastic (fast) weight that automatically increases or decreases based on the activity over time, \( t \). Miconi et al. (2018) developed this framework (see Eq. 4.1 and Eq. 4.2) in the context of RNNs and the straightforward method to train standard RNNs is to sufficiently include previous experiences in their future responses within each episode/lifetime. Here, episodes are often defined as datasets consisting of a training and test set in the meta-learning literature (Ravi & Larochelle, 2017; Vinyals et al., 2016). RNNs have proven to be universal Turing machines, hence can effectively learn to solve any computable function of their inputs. Therefore, during an episode, an RNN can be trained to augment the inputs at time \( t \) with the output and error at \( t - 1 \), ultimately enabling the network to learn to self-integrate new information.
\[ h_j(t) = \sigma \left\{ \sum_{i \in \text{inputs to } j} \left( \Theta_{i,j} + \alpha_{i,j} \text{Hebb}_{i,j} \right) h_i(t-1) \right\} \] (4.1)

\[ \text{Hebb}_{i,j}(t+1) = \text{Clip}(\text{Hebb}_{i,j}(t) + \eta h_i(t-1) h_j(t)), \] (4.2)

where:

- \( h_i(t) \) is the output of the \( i^{th} \) neuron at time \( t \).
- \( \sigma \) is a non-linearity (e.g., tanh was used in their experiments).
- \( \Theta_{i,j} \) is the standard weights of the network (referred to as non-plastic) between the \( i^{th} \) and \( j^{th} \) neurons.
- \( \alpha_{i,j} \) is the plasticity coefficient that determines the maximum magnitude of the plastic component.
- \( \eta \) is a “learning rate” to determine how quickly to acquire new information.
- \( \text{Hebb}_{i,j} \) is an accumulation of the product of pre-synaptic activity at connection \( i \) and post-synaptic activity at connection \( j \), thus forms the Hebbian traces which represent the plastic component (see Eq. 4.2).

The \( \Theta_{i,j} \), \( \alpha_{i,j} \) and \( \eta \) in Eq. 4.1 and Eq. 4.2 all learned parameters of the networks that are tuned by a gradient descent optimizer between episodes during training, in effect minimizing the expected loss over an episode. The Clip function in Eq. 4.2 is any function to clamp all of the elements in the input between the range \([\text{min, max}]\), where Miconi et al. (2019) restrict \( \text{Hebb}_{i,j} \) to be in the range \([-1, 1]\):

\[ \text{Clip}(x) \left\{ \begin{array}{ll} -1 & \text{if } x < -1 \\ x & \text{if } -1 < x < 1 \\ 1 & \text{if } x > 1 \end{array} \right. \] (4.3)
This operation is performed in order to make the Hebbian learning-based updates to the plastic component more stable during training. In previous work, Miconi et al. (2018) showed that the Hebbian learning can be stabilized using a simple decay term or a multiplicative normalization rule using Oja’s method (Oja, 2008).

Although differentiable plasticity served to be a powerful new method for training neural networks, it was mainly demonstrated on recurrent neural networks (RNN) for solving pattern memorization tasks and maze exploration with reinforcement learning (RL). Also, these approaches were only demonstrated on meta-learning problems and not the continual learning challenge of overcoming catastrophic forgetting. Our work also augments the slow weights in the FC layer with a set of plastic (fast) weights, but implements these using DHP following the framework in (Miconi, 2016; Miconi et al., 2018). We only update the parameters of the softmax output layer in order to achieve fast learning and preserve knowledge over time.

4.2 Overcoming Catastrophic Forgetting

This work leverages two biologically inspired strategies to overcome the catastrophic forgetting problem: 1) Task-specific Synaptic Consolidation — Protecting previously learned knowledge by dynamically adjusting the synaptic strengths to consolidate and retain memories. 2) CLS Theory — A dual memory system where, the neocortex (neural network) gradually learns to extract structured representations from the data while, the hippocampus (augmented episodic memory) performs rapid learning and individuated storage to memorize new instances or experiences.

4.2.1 Regularization Strategies

There have been several notable works inspired by task-specific synaptic consolidation for overcoming catastrophic forgetting such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al., 2017), Synaptic
Intelligence (SI) (Zenke et al., 2017), and Memory Aware Synapses (MAS) (Aljundi et al., 2018). They are often categorized as regularization strategies in the continual learning literature (Parisi et al., 2019). All of these regularization approaches estimate the importance of each parameter or synapse, $\Omega_k$, where the least plastic synapses can retain memories for long timescales and more plastic synapses are considered less important. The parameter importance and network parameters $\Theta_k$ are updated in either an online manner or after learning task $T_t$. Therefore, when learning new task $T_{t+1}$, a regularizer or surrogate loss is added to the original loss function $L_t(\Theta_t)$, so that we dynamically adjust the plasticity w.r.t. $\Omega_k$ and prevent any changes to important parameters of previously learned tasks:

$$
\hat{L}_t(\Theta^{(t)}) = L_t(\Theta^{(t)}) + \lambda \sum_k \Omega_k(\Theta_k^{(t)} - \Theta_k^{(t-1)})^2
$$

(4.4)

where $\Theta_k^{(t-1)}$ are the learned network parameters after training on the previous $t-1$ tasks and $\lambda$ is a hyperparameter for the regularizer to control the amount of forgetting (old versus new memories).

The main difference in these regularization strategies is the method chosen to compute the importance of each parameter, $\Omega_k$. In Elastic Weight Consolidation (EWC), Kirkpatrick et al. (2017) used the values given by the diagonal of an approximated Fisher information matrix for $\Omega_k$, and this was computed offline after training on a task was completed. An online variant of EWC was proposed by Schwarz et al. (2018) to improve EWC’s scalability by ensuring the computational cost of the regularization term does not grow with the number of tasks. Zenke et al. (2017) proposed an online method called Synaptic Intelligence (SI) for computing the parameter importance, where $\Omega_k$ is the cumulative change in individual synapses over the entire training trajectory on a particular task. Memory Aware Synapses (MAS) is an online method that measures $\Omega_k$ by the sensitivity of the learned function to a perturbation in the parameters, instead of measuring the change in parameters to the loss as seen in SI and EWC. The authors use the cumulative change in individual synapses on the squared L2-norm of the penultimate layer hence decoupling the parameter importance from labels and allowing the parameter importance to be computed even in absence of labels for any given data point.
Figure 4.1: $\Theta_1^*$ denotes the final optimal parameter values of the previously learned task $T_1$ (sky blue) and the training trajectories for three different gradient steps to learn task $T_2$. The training trajectory depicted for No Penalty approach indicates that the loss on task $T_2$ is minimized but information on the previously learned task $T_1$ is destroyed, thus taking a gradient direction in this step leads to catastrophic forgetting. If an L2 Penalty is imposed on each parameter, the parameters become constrained too severely and only task $T_1$ is learned, while task $T_2$ is not. However, methods such as EWC, SI and MAS show that the loss on task $T_2$ can be minimized without incurring a large loss on task $T_1$, by leveraging the parameter importance for Task $T_1$ and staying within a region of small error.

**Elastic Weight Consolidation**

The elastic weight consolidation (EWC) method proposed by Kirkpatrick et al. (2017) takes inspiration from how neocortical circuits in the mammalian brain protect previously learned knowledge to help avoid the catastrophic forgetting problem in neural networks. The EWC algorithm is analogous to task-specific synaptic consolidation for neural networks, where Kirkpatrick et al. (2017) constrain or pull back certain parameters towards the old parameters (i.e., freezing learned parameters to a certain degree) based on the parameters’ importance given by the diagonal of the Fisher information metric, and the imposed regularization during training is a quadratic penalty.

Once a neural network is trained to convergence on a particular dataset or task, the sensitivity of the learned model w.r.t. each of its individual parameters $\Theta_k$ can be estimated by analyzing the curvature of the loss surface along the direction in which $\Theta_k$ changes. A higher curvature indicates that a small change in $\Theta_k$ can induce a sharp increase of the loss. Kirkpatrick et al. (2017) propose to estimate the importance
measure using the diagonal of the Fisher information matrix $F$ because it can be computed from first order derivatives and can be used to describe the local curvature of the loss surface (i.e., second-order derivatives or Hessian matrix) near a minimum. The diagonal values of $F$ are then used to apply higher parameter importance weights to $\Theta_k$ that are more important to the learned task to mitigate catastrophic forgetting. Therefore, each element in the diagonal of $F$ can be denoted by $k$, thus the $k^{th}$ element is referred to as $F_k$. The total loss $\tilde{L}(\Theta^{(t)})$ being minimized when learning task $T_2$ after learning task $T_1$ can be more formally described as:

\[
\tilde{L}_2(\Theta^{(2)}) = L_2(\Theta^{(2)}) + \lambda \sum_k F_k (\Theta_k^{(2)} - \Theta_k^{(1)})^2
\]

(4.5)

Here, the $F_k$ indicates if the parameter $\Theta_k^{(2)}$ is important to the previous task $T_1$. To compute $F_k$, the training data from $D_{\text{train}}^{(1)}$ is sampled once and the empirical Fisher information matrix is computed:

\[
F_k = \frac{1}{n_1} \sum_{i=1}^{n_1} \nabla_{\Theta} \log p(x_i^{(1)}|\Theta^{(1)}) \nabla_{\Theta} \log p(x_i^{(1)}|\Theta^{(1)})^T,
\]

(4.6)

where empirical samples are drawn from $p(x^{(1)}|\Theta^{(1)})$, the log-likelihood w.r.t. to the optimal learned parameters in $\Theta^{(1)}$. This is a very common formulation and can be used to optimize the likelihood of the data $X$ w.r.t. $\Theta$ for generative models or in supervised learning, we optimize $p(y = \hat{y}|X, \Theta)$ w.r.t. $\Theta$. In EWC, computing and storing the Fisher information matrix for each task is not memory efficient because this requires summing over all possible output labels. Consequently, this results in a complexity linear in the number of outputs which reduces its applicability to low-dimensional output spaces. Also, the diagonal of $F$ at the final parameter values has to be computed separately at the end of learning each task $\{T_t\}_{t=1}^N$. 

28
Synaptic Intelligence

Zenke et al. (2017) also take inspiration from biological neural networks, similar to Kirkpatrick et al. (2017), but here, Zenke et al. (2017) showed how the internal synaptic dynamics can allow neural networks to learn tasks sequentially without forgetting what was learned before. The authors propose a higher dimensional synaptic model termed as intelligent synapses or synaptic intelligence (SI), which keeps track of the past and current parameter values, and maintains an estimate of the each synapse’s importance towards solving previously learned tasks in an online manner. A synapse’s importance weight is computed efficiently and locally during training. This estimate of importance represents the effect of a synapse on the overall change in the global loss.

The contribution of each synapse or parameter $\Theta_k$ to the change in the loss is provided by a weight update during gradient descent training on a task:

$$\Delta L_k = \Delta \Theta_k \cdot g(\Theta_k).$$  \hspace{1cm} (4.7)

Eq. 4.7 involves multiplying the gradient of error with a change in the synaptic weight, where $\Delta \Theta_k = \Theta_k' - \Theta_k$ is the change in weight and $g(\Theta_k)$ denotes the gradient given by $\frac{\partial L}{\partial \Theta_k}$. The goal during training is to find learning trajectories where the final point stays close to a minimum of the loss function $L$ across all tasks.

The total loss then effectively becomes the running sum over the individual parameters over the learning trajectory which is the accumulation of gradient descent updates during training and can be represented as $\sum_k \Delta L_k$. Moreover, the importance measure for each parameter can be more formally computed as:

$$\Omega_k = \sum_k \frac{\Delta L_k}{M_k^2 + \xi}.$$  \hspace{1cm} (4.8)
where $M_k$ indicates the distance travelled by $\Theta_k$ in parameter space and $\xi$ is a small constant to prevent division by 0 during the computation. Therefore, the total loss $\tilde{L}(\Theta^{(t)})$ when learning task $T_t$ after learning $T_{t-1}$ tasks can be more formally described as:

$$
\tilde{L}_t(\Theta^{(t)}) = L_t(\Theta^{(t)}) + \lambda \sum_k \Omega_k (\Theta_k^{(t)} - \Theta_k^{(t-1)})^2,
$$

(4.9)

where $\lambda$ is a scalar hyperparameter to control the amount of forgetting. It is to be noted that $c$ is used instead of $\lambda$ in the original SI loss formulation (Zenke et al., 2017), but we use $\lambda$ to keep notation uniform with the previous section. The SI model computes the importance measure in an online manner, where the authors compute the change in loss due to a change in a particular parameter and build this information up over the entire learning trajectory. However, the SI approach can overestimate the importance measure because it relies on weight changes in mini-batch gradient descent optimization. Also, there is a possibility of underestimating the parameter importance when using pretrained CNN models for computer vision tasks because some of the weights may be used without much change.

### Memory Aware Synapses

Memory Aware Synapses (MAS) (Aljundi et al., 2018) is inspired by Hebbian learning, where synapses connecting neurons that fire synchronously are strengthened which results in high activations, whereas ones that fire asynchronously have low activations. MAS also computes the parameter importance in an online manner like SI, but MAS can determine the importance measure in an unsupervised way. This is done by estimating the parameter importance weights without depending on the loss like EWC and SI, and instead, by analyzing the sensitivity of the learned output function of the model.

Aljundi et al. (2018) help alleviate some of the drawbacks of the SI method and avoid complications with gradients being close to zero because they analyze the sensitivity of the learned function which is not being minimized, unlike the loss. After the model has learned an approximation $f$, in which the ap-
proximated mapping \( f \) is to be preserved while learning subsequent tasks. The sensitivity of the learned function output \( f \) to changes in network parameters is determined by applying a small perturbation \( \delta = \{\delta_{i,j}\} \) in the parameters \( \Theta = \{\Theta_{i,j}\} \), which can be approximated as,

\[
f(x_k; \Theta + \delta) - f(x_k; \Theta) \approx \sum_{i,j} g_{i,j}(x_k) \delta_{i,j},
\]

(4.10)

where \( g_{i,j}(x_k) \) is the gradient w.r.t. the parameter \( \Theta_{i,j} \) at data point \( x_k \) and the change in network parameter \( \Theta_{i,j} \) is given by \( \delta_{i,j} \). The goal is to preserve the learned function \( f \) of the neural network and refrain from making changes to parameters that are most important for these predictions. The authors obtain the importance measure by taking the magnitude of the gradient, which provides information on how the effect of a perturbation to a parameter affects the output of \( f \) for \( x_k \). Then, the running sum of the gradients over the given data points is taken as follows,

\[
\Omega_{i,j} = \frac{1}{n_t} \sum_{k=1}^{n_t} ||g_{i,j}(x_k)||.
\]

(4.11)

As each new data point is presented to the neural network, Eq. 4.11 can be updated in an online manner. The output is not affected much by the parameters with low importance, therefore these parameters can be changed to minimize the loss for future tasks \( \{T_t\}_{t=1}^{N} \), whereas parameters with high importance are left unchanged. Therefore, when learning a new task \( T_t \), the total loss then becomes:

\[
\tilde{L}_t(\Theta^{(t)}) = L_t(\Theta) + \lambda \sum_{i,j} \Omega_{i,j}(\Theta^{(t)}_{i,j} - \Theta^{(t-1)}_{i,j})^2,
\]

(4.12)

where similar to EWC and SI, we have a regularizer in addition to the task loss \( L_t(\Theta) \) and \( \lambda \) is a hyperparameter to controlling the amount of forgetting. MAS compares to an implicit memory for each parameter of the neural network, where the activations of the network are used to update when ap-
plied to new data points. In this method, a neural network model with limited capacity and continually changing tasks can learn what not to forget in an unsupervised manner.

4.2.2 Dual-Weight Models

Our work draws inspiration from CLS theory which is a powerful computational framework for representing memories with a dual memory system via the neocortex and hippocampus. There have been numerous approaches based on CLS principles involving pseudo-rehersal (Robins, 1995; Ans et al., 2004; Atkinson et al., 2018), episodic replay (Lopez-Paz & Ranzato, 2017; Li & Hoiem, 2018) and generative replay (Shin et al., 2017; Wu et al., 2018). However, in our work, we are primarily interested in neuroplasticity techniques inspired from CLS theory for alleviating catastrophic forgetting.

Hinton & Plaut (1987) showed how each synaptic connection can be composed of a fixed weight where slow learning stores long-term knowledge and a fast-changing weight for temporary associative memory (see Figure 4.2). The fast weights are quickly decaying and are used to learn new temporary associations without interfering with the slow weights. Consequently, old associations that were lost by interference by subsequent learning were “revived” by using the fast weights for relearning only from a subset of the old associations. This approach involving slow and fast weights is analogous to properties of CLS theory to overcome catastrophic forgetting during continual learning. The dual-weight system showed that older memories can be recovered without losing the newly learned knowledge.

![Figure 4.2: A dual-weight summation model based on Hinton & Plaut (1987), where the input from one neuron to the other is an output of the activation produced by summing the slow and fast weights.](image-url)
Gardner-Medwin (1989) also proposed a dual-weight model (shown in Figure 4.3) with the aim of forming short- and long-term memory, but proposed to use multiplicative weights based on the synaptic activity in biological neural networks, where synapses act on different timescales (i.e. slow and fast memory). Thus, the changes in synaptic activity combine to generate the resulting activation. Moreover, the model proposed by Gardner-Medwin (1989) reduced catastrophic forgetting and increased the network capacity by using Hebbian learning and an optimization rule that decreased overlap when training the long-term memory.

![Diagram of dual-weight multiplicative model](image)

Figure 4.3: A dual-weight multiplicative model based on Gardner-Medwin (1989), where the input from one neuron to the other is an output of the activation produced by multiplying the slow and fast weights.

Recent research in this vein has included replacing a soft attention mechanism with fast weights in RNNs (Ba et al., 2016), the Hebbian Softmax layer (Rae et al., 2018), augmenting slow weights in the FC layer with a fast weights matrix (Munkhdalai & Trischler, 2018), differentiable plasticity (Miconi, 2016; Miconi et al., 2018) and neuromodulated differentiable plasticity (Miconi et al., 2019). However, all of these methods were focused on rapid learning on simple tasks or meta-learning over a distribution of tasks or datasets. Furthermore, they did not examine learning a large number of new tasks while, alleviating catastrophic forgetting in continual learning.

### 4.3 Summary

In this chapter, we reviewed contemporary literature related to non-uniformly plastic neural networks, synaptic memory consolidation techniques such as EWC, SI and MAS, as well as CLS techniques based on dual-weight synaptic connections composed of slow and fast weights. We reviewed the literature
pertaining to recent work on Hebbian based plasticity in neural networks for rapid learning and auto-associative memory, particularly differentiable plasticity. We also reviewed strategies for overcoming catastrophic forgetting based on regularization and dual-weight models. We notice that, to the best of our knowledge, no other work has focused on unifying plastic neural networks with synaptic memory consolidation and episodic memories for continual lifelong learning.
Chapter 5

Methodology

The different task-specific synaptic consolidation approaches EWC, SI and MAS on their own have shown to mitigate the catastrophic forgetting problem. However, there has been little investigation on how episodic replay in combination with synaptic memory consolidation can be interlayed to mitigate the catastrophic forgetting problem. More specifically,

“how can fixed capacity neural networks adjust the degree of plasticity in their synaptic connections to learn and retain consolidated knowledge over long timescales?”

In this work, we primarily explore neural networks with dual-weight memory that interlay between each other during training to enable rapid learning of new experiences but also consolidating previous knowledge through short- and long-term plasticity. We also combine this with synaptic memory consolidation to enable a more powerful model for continual lifelong learning.

This chapter outlines Differentiable Hebbian (DHP) Softmax Layer which behaves as a hippocampal system that reactivates long-term memory traces from short-term memory. We also combine this with
synaptic consolidation approaches such as EWC, SI and MAS, and refer to our model as Differentiable Hebbian Consolidation. We discuss the different components of Differentiable Hebbian Consolidation including the DHP Softmax Layer, the Hebbian Update rule for updating the memory and the updated quadratic loss function.

### 5.1 Differentiable Hebbian Softmax (DHP Softmax)

In our model, each synaptic connection in the softmax layer has two weights:

1. The slow weights, $\Theta \in \mathbb{R}^{m \times d}$, where $m$ is the number of units in the final hidden layer, and $d$ is the number of outputs in the last layer.

2. A Hebbian plastic component of the same cardinality as the slow weights, composed of the plasticity coefficient, $\alpha$, and the Hebbian trace, $\text{Hebb}$.

Scaling parameter $\alpha$ adjusts the magnitude of the Hebb. Given the hidden activations of the final hidden layer, $H \in \mathbb{R}^{B \times m}$, the Hebbian traces accumulate the mean of these hidden activations for each unique target label in the mini-batch $\{y_{1:B}\} \in \mathcal{Y}$ of size $B$ which are denoted by $\tilde{h} \in \mathbb{R}^{1 \times m}$ (refer to Algorithm 1).

Given the activation of each neuron in $h_{1:B} \in H$ at the pre-synaptic connection $i$, the unnormalized log probabilities (softmax pre-activations) $z$ at the post-synaptic connection $j$ can be more formally computed using Eq. 5.1. The softmax function is then applied element-wise to $z$ to obtain the desired predicted probabilities $\hat{y}$ thus, $\hat{y} = \text{softmax}(z)$. The $\eta$ parameter in Eq. 5.2 is a scalar value that dynamically learns how quickly to acquire new experiences into the plastic component, and thus behaves as the “learning rate” for the plastic connections. The $\eta$ parameter also acts as a decay term for the Hebb to prevent instability caused by a positive feedback loop in the Hebbian traces.
\[ z_j = \sum_{i=1}^{m} (\Theta_{i,j} + \alpha_{i,j}\text{Hebb}_{i,j})h_i \]  

\[ \text{Hebb}_{i,j} \leftarrow (1 - \eta)\text{Hebb}_{i,j} + \eta\tilde{h}_{i,j} \]

The structural network parameters \( a_{i,j}, \eta \) and \( \Theta_{i,j} \) are optimized by gradient descent as the model is trained sequentially on different tasks in the continual learning setup. Here, the neural network uses backpropagation as meta-learning to tune the structural components in the plastic component (Eq. 5.1) and Hebbian update rule (Eq. 5.2). In the recent work from Miconi et al. (2019), a Clip function is used to clip the values in Hebb to a specified min and max range (i.e., -1 to 1). However, in practice, we found that using the \( \eta \) parameter as a decay term on the Hebb leads to better stability during continual learning, a method used by Miconi et al. (2018) for improving stability during training in the meta-learning setup.

In Figure 5.1 is a diagram of a shallow multi-layered perceptron network with DHP Softmax.
5.2 Hebbian Update Rule

The Hebbian traces are initialized to zero only at the start of learning the first task $T_1$ and during training, the Hebb is automatically updated in the forward pass using Algorithm 1. Specifically, the Hebbian update for a corresponding class $c$ in $y_{1:B}$ is computed on line 6. This Hebbian update $\frac{1}{s} \sum_{b=1}^{B} h[y_b = c]$ is analogous to another formulaic description of the Hebbian learning update rule $w_{i,j} = \frac{1}{N} \sum_{k=1}^{N} a_i^k a_j^k$ (Hebb, 1949), where $w_{i,j}$ is the change in weight at connection $i,j$ and $a_i^k, a_j^k$ denote the activation levels of neurons $i$ and $j$, respectively, for the $k^{th}$ input. Therefore, in our model, $w = \tilde{h}$ is the Hebbian weight update, $a_i = h$ is the hidden activations of the last hidden layer, $a_j = y$ is the corresponding target class in $y_{1:B}$ and $N = s$ is the number of inputs for the corresponding class in $y_{1:B}$ (see Algorithm 1). Across the model’s lifetime, we only update the Hebbian traces during training as it learns tasks in a continual manner. Therefore, during test time, we maintain and use the most recent Hebbian traces to make predictions. A visualization of the Hebbian update for the corresponding class $c = 6$ is shown in Figure 5.2.

---

**Algorithm 1** Batch update Hebbian traces.

1: **Input:** $h_{1:B} \in H$ (hidden activations of penultimate layer),
   $y_{1:B}$ (target labels),
   Hebb (Hebbian trace)

2: **Output:** $z_{1:B}$ (softmax pre-activations),
   Hebb (updated Hebbian traces)

3: for each target label $c \in \{ y_{1:B} \}$ do
4:   $s \leftarrow \sum_{b=1}^{B} [y_b = c]$ /*Count the total occurrences of class $c \in y.*/  
5:   if $s > 0$ then
6:     $\tilde{h} \leftarrow \frac{1}{s} \sum_{b=1}^{B} h[y_b = c]$ /* Perform the Hebbian update for the corresponding class $c.*/  
7:     Hebb_{c,c} \leftarrow (1 - \eta)\text{Hebb}_{c,c} + \eta \tilde{h}$
8:   end if
9: end for
10: $Z \leftarrow (\Theta + \alpha \text{Hebb})H$ /* Compute the softmax pre-activations for the mini-batch.*/
Figure 5.2: An example of a Hebbian update for the active class, \( c = 6 \in y_{1:B} \). Here, we are given the hidden activations of the final hidden layer, \( h_{1:B} \in H \). In this example, the multiple hidden activations corresponding to class \( c = 6 \) (represented by the pink boxes) are averaged into one vector denoted by \( \tilde{h} \in \mathbb{R}^{1 \times m} \). This Hebbian update visualization reflects Lines 4-6 in Algorithm 1 and is repeated for each unique class in the target vector \( y_{1:B} \).

The aim is to build a model that can enable high-quality short- and long-term memory and consolidate the short-term memory traces via the Hebb into long-term memory. The plastic component learns rapidly and performs sparse parameter updates to quickly store memory traces for each recent experience without interference from other similar recent experiences. Furthermore, the hidden activations corresponding to the same active class are accumulated into one vector \( \tilde{h} \), thus forming a compressed episodic memory in the Hebbian traces to reflect individual episodic traces (similar to the hippocampus in biological neural networks (Schapiro et al., 2017)). As a result, this method improves learning of rare classes and speeds up binding of class labels to deep representations of the data without introducing any additional hyperparameters besides \( \eta \). In our model, memory consolidation occurs by reducing the degree of plasticity in the short term Hebbian memory traces to prevent changes to the long-term weights during training. This is interesting because experimentally, it has been shown that decreasing rates of synaptic plasticity through specialized mechanisms in the mammillan brain, in fact protect knowledge about previously learned tasks from interference when learning new tasks (Cichon & Gan, 2015). An example of a PyTorch implementation of the DHP Softmax Layer can be found in Appendix B.
5.3 Updated Quadratic Loss

Following the existing work for overcoming catastrophic forgetting such as EWC (Kirkpatrick et al., 2017), Online EWC (Schwarz et al., 2018), SI (Zenke et al., 2017) and MAS (Aljundi et al., 2018), we regularize the loss $L_t(\Theta^{(t)}, \alpha, \eta)$ as in Eq. 4.4 and update the synaptic importance parameters of the network in an online manner. We rewrite Eq. 4.4 to obtain Eq. 5.3 and show that the network parameters $\Theta^{(t)}_{i,j}$ are the weights of the connections between pre- and post-synaptic activity, as seen in Eq. 5.1. The loss $L_t$ is now a function of $\alpha$ and $\eta$ given that they are trainable parameters of the network, in addition to the standard network parameters $\Theta^{(t)}$.

$$
\tilde{L}_t(\Theta^{(t)}, \alpha, \eta) = L_t(\Theta^{(t)}, \alpha, \eta) + \lambda \sum_{i,j} \Omega_{i,j}(\Theta^{(t)}_{i,j} - \Theta^{(t-1)}_{i,j})^2
$$

(5.3)

We adapt the existing task-specific consolidation approaches to our model and do not compute the synaptic importance parameters on the plastic component of the network, hence we only regularize the slow weights of the network. Furthermore, when training the first task $T_{t=1}$, the synaptic importance parameter, $\Omega_{i,j}$ in Eq. 5.3, was set to 0 for all of the task-specific consolidation methods that we tested on except for SI. This is because SI is the only method we evaluated that estimates $\Omega_{i,j}$ while training, whereas Online EWC and MAS compute $\Omega_{i,j}$ after learning a task. The plastic component of the softmax layer in our model can alleviate catastrophic forgetting of consolidated classes by allowing gradient descent to optimize how plastic the connections should be (i.e. less plastic to preserve old information or more plastic to quickly learn new information).

5.4 Summary

In this chapter, we introduced the Differentiable Hebbian Consolidation model for continual lifelong learning, which included a discussion on the DHP Softmax layer, Hebbian update algorithm for super-
vised classification and the new quadratic loss function for decreasing the learning on slow weights of the network. Throughout this chapter, we also provided intuitions on our design choices and refer back to theoretical studies and models in the computational neuroscience and continual learning literature. We also ensured to that we refer back to the challenges faced in the continual learning and existing approaches as we discussed our contributions to address them.
Chapter 6

Experiments

In our experiments, we compare our approach to vanilla neural networks with Online EWC, SI and MAS. Since our approach increases the capacity of the DNN due to the addition of plastic weights, we add an extra set of slow weights to the softmax output layer of the standard neural network to match the capacity. We do this to show that it is not the increased model capacity from the plastic weights that is helping mitigate the forgetting when performing sequential task learning, thus ensuring a fair evaluation. We tested our continual learning approach on the Permuted MNIST, Split MNIST and 5-Vision Datasets Mixture benchmarks. We also introduce the Imbalanced Permuted MNIST problem which combines the challenges of both class imbalance and concept drift, something that a real-world ML system would experience.

For all of the benchmarks, we evaluated the model based on the average classification accuracy on all previously learned tasks as a function of $n$, the number of tasks trained so far. To determine memory retention and flexibility of the model, we are particularly interested in the test performance on the first task and the most recent one. To establish a baseline for comparison of well-known task-specific consolidation methods, we trained neural networks with Online EWC, SI and MAS, respectively, on all tasks
in a sequential manner. In all of the benchmarks, the hyperparameters of the consolidation methods Online EWC, SI and MAS remain the same with and without DHP Softmax, and the plastic components were not regularized. Moreover, all of the ranges considered for performing grid search to find the best hyperparameter values were guided by experimental setups in the original work. All experiments were run on either a single Nvidia Titan V or a Nvidia RTX 2080 Ti.

6.1 Permuted MNIST

In this benchmark, we make use of the original MNIST dataset consisting of 60,000 training examples and 10,000 testing examples with images of size $28 \times 28$. The pixels of all class digits are permuted differently for each task with a fixed random permutation. Although the output domain is constant, the input distribution changes between tasks and is mostly independent of each other, thus, there exists a concept drift.

Figure 6.1: In the Permuted MNIST benchmark, the pixels of each digit in the MNIST dataset is permuted with a fixed random permutation, where each task $t > 1$ has its own unique permutation. This generates a sequence of tasks where each dataset $\{D^{(t)}_{\text{train}}\}_{t=1}^{N}$ has its own data distribution and the same complexity.

Experimental Setup

In the Permuted MNIST and Imbalanced Permuted MNIST benchmarks we use a multi-layered perceptron (MLP) network with two hidden layers consisting of 400 units each with ReLU nonlinearities, and a cross-entropy loss. The initial value of $\eta$ of the plastic component was set to be a small value of 0.001
and we want to emphasize that we spent little to no effort on tuning the initial value of this parameter (see Appendix A for a sensitivity analysis). We train the network on a sequence of tasks $T_{n=1:10}$ with mini-batches of size 64 and optimized using plain SGD with a learning rate of 0.01. We train for at least 10 epochs and perform early-stopping once the validation error does not improve for 5 epochs. If the validation error increases for more than 5 epochs, then we terminated the training on the task $T_n$, reset the network weights and Hebbian traces to the values that had the lowest test error, and proceeded to the next task.

For the Permuted MNIST experiments shown in Figure 6.2, the regularization hyperparameter $\lambda$ for each of the task-specific consolidation methods is set to $\lambda = 100$ for Online EWC (Schwarz et al., 2018), $\lambda = 0.1$ for SI (Zenke et al., 2017) and $\lambda = 0.1$ for MAS (Aljundi et al., 2018). We note that for the SI method, $\lambda$ refers to the parameter $\epsilon$ in the original work (Zenke et al., 2017) but we use $\lambda$ to keep the notation consistent across other task-specific consolidation methods. In SI, the damping parameter, $\xi$, was set to 0.1. To find the best hyperparameter combination for each of these synaptic consolidation methods, we performed a grid search using a task sequence determined by a single seed. For Online EWC, we tested values of $\lambda \in \{10, 20, 50, \ldots, 400\}$, SI — $\lambda \in \{0.01, 0.05, \ldots, 0.5, 1.0\}$ and MAS — $\lambda \in \{0.01, 0.5, \ldots, 1.5, 2.0\}$.

**Results**

We first compare the performance between our network with DHP Softmax and a fine-tuned vanilla MLP network we refer to as *Finetune* in Figure 6.2 and no task-specific consolidation methods involved. The network with DHP Softmax alone showed significant improvement in its ability to alleviate catastrophic forgetting across all tasks compared to the baseline network. Then we compared the performance with and without DHP Softmax using the same task-specific consolidation methods. Figure 6.2 shows the average test accuracy as new tasks are learned for the best hyperparameter combination for each task-specific consolidation method. We find our DHP Softmax with consolidation maintains a higher test accuracy throughout sequential training of tasks than without DHP Softmax for all variants. It is to be noted that the DHP Softmax + MAS method provides a $12.80 \pm 1.32\%$ improvement over the Finetune.
method (see Table 6.1).

Table 6.1: The test accuracy on Permuted MNIST and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Accuracy (%)</th>
<th>Improvement over Finetune (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>76.73 ± 0.67</td>
<td>(\uparrow 1.76 ± 0.96)</td>
</tr>
<tr>
<td>DHP Softmax</td>
<td>78.49 ± 0.29</td>
<td>(\uparrow 7.99 ± 1.60)</td>
</tr>
<tr>
<td>SI</td>
<td>84.72 ± 0.93</td>
<td>(\uparrow 8.47 ± 0.99)</td>
</tr>
<tr>
<td>DHP Softmax + SI</td>
<td>85.20 ± 0.32</td>
<td>(\uparrow 9.51 ± 1.37)</td>
</tr>
<tr>
<td>Online EWC</td>
<td>86.24 ± 0.70</td>
<td>(\uparrow 10.57 ± 1.17)</td>
</tr>
<tr>
<td>DHP Softmax + Online EWC</td>
<td>87.30 ± 0.50</td>
<td>(\uparrow 11.79 ± 1.17)</td>
</tr>
<tr>
<td>MAS</td>
<td>88.52 ± 0.50</td>
<td>(\uparrow 12.80 ± 1.32)</td>
</tr>
<tr>
<td>DHP Softmax + MAS</td>
<td>89.53 ± 0.65</td>
<td>(\uparrow 12.80 ± 1.32)</td>
</tr>
</tbody>
</table>

Figure 6.2: The average test accuracy on a sequence of 10 Permuted MNIST datasets \(T_{n=1:10}\). The addition of DHP in all cases either maintains or improves the model’s ability to reduce forgetting. The error bars correspond to the standard error across 10 trials.
6.1.1 Ablation Study

We further examine the structural parameters of the network and Hebb traces to provide further interpretability into the behaviour of our proposed model. The left plot in Figure 6.3 shows the behaviour of $\eta$ during training as 10 tasks in the Permuted MNIST benchmark are learned continually. Initially, in task $T_1$, the $\eta$ increases very quickly from 0.001 to 0.024 suggesting that the synaptic connections become more plastic to quickly acquire new information. Eventually, $\eta$ decays after the 3rd task to reduce the degree of plasticity to prevent interference between the learned representations. We also observe that within each task from $T_4$ to $T_{10}$, the $\eta$ initially increases then decays. The Frobenius Norm of the Hebb trace (middle plot in Figure 6.3) suggests that Hebb grows without runaway positive feedback everytime a new task is learned, maintaining a memory of which synapses contributed to recent activity. The Frobenius Norm of $\alpha$ (right plot in Figure 6.3) indicates that the plasticity coefficients grow within each task, indicating that the network is leveraging the structure in the plastic component. It is important to note that gradient descent and backpropagation are used as meta-learning to tune the structural parameters in the plastic component.

![Figure 6.3](image1)

**Figure 6.3:** (left) Hebbian learning rate and decay value $\eta$, (middle) Frobenius Norm of the Hebbian memory traces $\|\text{Hebb}\|_F$, (right) Frobenius Norm of the plasticity coefficients $\|\alpha\|_F$ while training each task $T_{1:10}$.

The left-matrix in Figure 6.4 visualizes $\alpha \in \mathbb{R}^{64 \times 10}$ which shows significant structure post-training on 5 tasks. Note that in our Permuted MNIST experiments, the softmax output layer is of the dimension $\mathbb{R}^{1000 \times 10}$, but the parameters depicted in Figure 6.4 are $\mathbb{R}^{64 \times 10}$ for illustration purposes. The Hebb matrix shows visually the hidden activations with high class-correlated outputs (high activations) which are more important for the given task than those with less class-correlated outputs (low activations). Finally,
the total effective weight ($\Theta + \alpha$Hebb) is shown in the right-most matrix.

Figure 6.4: Visualization of plastic and fixed weights from Eq. 5.1.

6.2 Imbalanced Permuted MNIST

We introduce the Imbalanced Permuted MNIST problem which is identical to the Permuted MNIST benchmark but, now each task is an imbalanced distribution where training samples in each class were artificially removed based on some random probability. This benchmark was motivated by the fact that class imbalance and concept drift can significantly hinder predictive performance, and the problem becomes particularly challenging when they occur simultaneously.

For each task in the Imbalanced Permuted MNIST problem, we artificially removed training samples from each class in the original MNIST dataset (LeCun et al., 2001) based on some random probability. For each class and each task, we draw a different removal probability from a standard uniform distribution $U(0, 1)$, and then remove each sample from that class with that probability. The distribution of classes in each dataset corresponding to tasks $T_{n=1:10}$ is given in Figure 6.5.
Figure 6.5: Distribution of classes in each imbalanced training dataset \( \{D^{(t)}_{\text{train}}\}_{t=1}^{10} \) for the respective tasks \( \{T_t\}_{t=1}^{10} \) in the Imbalanced Permuted MNIST benchmark.

**Experimental Setup**

For the Imbalanced Permuted MNIST experiments shown in Figure 6.6, the regularization hyperparameter \( \lambda \) for each of the task-specific consolidation methods is \( \lambda = 400 \) for Online EWC, \( \lambda = 1.0 \) for SI and \( \lambda = 0.1 \) for MAS. In SI, the damping parameter, \( \xi \), was set to 0.1. Similar to the Permuted MNIST benchmark, to find the best hyperparameter combination for each of these synaptic consolidation methods, we performed a grid search using a task sequence determined by a single seed. For Online EWC, we tested values of \( \lambda \in \{50, 100, \ldots, 1 \times 10^3\} \), SI — \( \lambda \in \{0.1, 0.5, \ldots, 2.5, 3.0\} \) and MAS — \( \lambda \in \{0.01, 0.05, \ldots, 1.5, 2.0\} \). Across all experiments, we maintained the same random probabilities determined by a single seed to artificially remove training samples from each class.
Results

Figure 6.6 shows the average test accuracy as new tasks are learned for the best hyperparameters of each method. We see that DHP Softmax achieves $80.85 \pm 0.65\%$ after learning 10 tasks with imbalanced class distributions in a sequential manner, thus providing significant improvement over the standard neural network baseline of $76.44 \pm 0.99\%$. The significance of the compressed episodic memory mechanism in the Hebbian traces is more apparent in this benchmark because the plastic component allows rare classes that are encountered infrequently to be remembered for a longer period of time. Similar to the Permuted MNIST benchmark, we also train neural networks with DHP Softmax plus EWC, SI and MAS, then compare the performance against their respective baselines. We find that DHP Softmax with MAS achieves $88.80 \pm 0.52\%$; outperforming all other methods and across all tasks. When we only use MAS without DHP Softmax, there is a noticeable drop in performance of $1.48 \pm 1.17\%$. Similar to the Permuted MNIST benchmark, the DHP Softmax + MAS method provides a $12.36 \pm 1.51\%$ improvement over the Finetune method (see Table 6.2).

Table 6.2: The test accuracy on the Imbalanced Permuted MNIST and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Accuracy (%)</th>
<th>Improvement over Finetune (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>76.44 ± 0.99</td>
<td>–</td>
</tr>
<tr>
<td>DHP Softmax</td>
<td>80.85 ± 0.65</td>
<td>(↑ 4.41 ± 1.64)</td>
</tr>
<tr>
<td>SI</td>
<td>85.92 ± 0.60</td>
<td>(↑ 9.48 ± 1.59)</td>
</tr>
<tr>
<td>DHP Softmax + SI</td>
<td>85.39 ± 0.43</td>
<td>(↑ 8.95 ± 1.42)</td>
</tr>
<tr>
<td>Online EWC</td>
<td>87.18 ± 0.36</td>
<td>(↑ 10.74 ± 1.35)</td>
</tr>
<tr>
<td>DHP Softmax + Online EWC</td>
<td>87.43 ± 0.34</td>
<td>(↑ 10.99 ± 1.33)</td>
</tr>
<tr>
<td>MAS</td>
<td>87.32 ± 0.29</td>
<td>(↑ 10.88 ± 1.28)</td>
</tr>
<tr>
<td>DHP Softmax + MAS</td>
<td>88.80 ± 0.52</td>
<td>(↑ 12.36 ± 1.51)</td>
</tr>
</tbody>
</table>
### Figure 6.6: The average test accuracy on a sequence of 10 class-imbalanced Permuted MNIST datasets $T_{n=1:10}$. The addition of DHP improves the model’s ability to reduce forgetting. The error bars correspond to the standard error across 10 trials.

### 6.3 SplitMNIST

We split the original MNIST dataset (LeCun et al., 2001) into a sequence of 5 binary classification tasks: $T_1 = \{0/1\}$, $T_2 = \{2/3\}$, $T_3 = \{4/5\}$, $T_4 = \{6/7\}$ and $T_5 = \{8/9\}$. The output spaces are disjoint between tasks, unlike the previous two benchmarks.

![SplitMNIST](image)
Experimental Setup

Similar to the network used by Zenke et al. (2017), we use an MLP network with two hidden layers of 256 ReLU nonlinearities each, and a cross-entropy loss. A multi-headed approach was used to avoid interference between digits at the softmax output layer due to changes in the label distribution. We compute the cross-entropy loss, $L$, at the softmax output layer for the digits present in the current task, $T$. We train the network on a sequence of $T_{n=1:5}$ tasks with mini-batches of size 64 and optimized using plain SGD with a fixed learning rate of 0.01 for 10 epochs.

The initial $\eta$ value was set to 0.001 as seen in previous benchmark experiments. We found that different values of $\eta$ yielded very similar final test performance after learning $T_5$ tasks. For the Split MNIST experiments shown in Figure 6.8, the regularization hyperparameter $\lambda$ for each of the task-specific consolidation methods is $\lambda = 400$ for Online EWC, $\lambda = 1.0$ for SI and $\lambda = 1.5$ for MAS. In SI, the damping parameter, $\xi$, was set to 0.001. To find the best hyperparameter combination for each of these synaptic consolidation methods, we performed a grid search using the 5 task binary classification sequence (0/1, 2/3, 4/5, 6/7, 8/9). For Online EWC, we tested values of $\lambda \in \{1, 25, 50, 100, \ldots, 1 \times 10^3, 2 \times 10^3\}$, SI — $\lambda \in \{0.1, 0.5, 1.0, \ldots, 5.0\}$ and MAS — $\lambda \in \{0.01, 0.05, 1.0, \ldots, 4.5, 5.0\}$.

Results

We observed that DHP Softmax achieves 99.23 ± 0.42% thus, provides a 7.80 ± 0.97% improvement on test performance compared to a finetuned MLP network (Figure 6.8 and Table 6.3). Also, combining DHP Softmax with task-specific consolidation consistently improves performance across all tasks, especially the most recent one, $T_5$. The DHP Softmax + SI method achieves the best test performance after learning task $T_5$ with 99.15 ± 0.09%, a 8.72 ± 0.64% improvement over Finetune.
Table 6.3: The test accuracy on SplitMNIST and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Accuracy (%)</th>
<th>Improvement over Finetune (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>90.43 ± 0.55</td>
<td>–</td>
</tr>
<tr>
<td>DHP Softmax</td>
<td>98.23 ± 0.42</td>
<td>(↑ 7.80 ± 0.97)</td>
</tr>
<tr>
<td>SI</td>
<td>97.77 ± 0.64</td>
<td>(↑ 7.34 ± 1.19)</td>
</tr>
<tr>
<td>DHP Softmax + SI</td>
<td>99.15 ± 0.09</td>
<td>(↑ 8.72 ± 0.64)</td>
</tr>
<tr>
<td>Online EWC</td>
<td>97.65 ± 0.44</td>
<td>(↑ 7.22 ± 0.99)</td>
</tr>
<tr>
<td>DHP Softmax + Online EWC</td>
<td>98.96 ± 0.20</td>
<td>(↑ 8.53 ± 0.75)</td>
</tr>
<tr>
<td>MAS</td>
<td>98.24 ± 0.13</td>
<td>(↑ 7.81 ± 0.68)</td>
</tr>
<tr>
<td>DHP Softmax + MAS</td>
<td>98.43 ± 0.33</td>
<td>(↑ 8.00 ± 0.88)</td>
</tr>
</tbody>
</table>

Figure 6.8: The average test accuracy on a sequence of 5 binary classification tasks (0/1, 2/3, 4/5, 6/7, 8/9) from the original MNIST dataset $T_{t=1:5}$. The average test accuracy over all learned tasks is provided in the legend. The error bars correspond to the standard error across 10 trials.
6.4 5-Vision Datasets Mixture

Following previous works (Ritter et al., 2018; Zeno et al., 2018), we perform continual learning on a sequence of 5 different image classification datasets: MNIST, notMNIST\(^1\), FashionMNIST (Xiao et al., 2017), SVHN (Netzer et al., 2011) and CIFAR-10 (Krizhevsky, 2009). Figure 6.9 shows a random batch of 36 training examples from each task in the 5-Vision Datasets Mixture benchmark. The details regarding each of the datasets in each task are presented below:

- **notMNIST**: The notMNIST dataset consists of font glyphs corresponding to letters ‘A’ to ‘J’. The original dataset has 500,000 and 19,000 grayscale images of size 28×28 for training and testing, respectively. However, similar to MNIST, we only use 60,000 images for training and 10,000 for testing.

- **FashionMNIST**: The FashionMNIST dataset consists of 10 categories of various articles of clothing, and there are 60,000 and 10,000 grayscale images sized 28×28 for training and testing, respectively.

- **SVHN**: The Street View House Numbers (SVHN) dataset consists of digits ‘0’ to ‘9’ from Google Street View images and there are 73,257 and 26,032 colour images of size 32×32 for training and testing, respectively.

- **CIFAR-10**: The CIFAR-10 dataset consists of 50,000 and 10,000 colour images of size 32×32 from 10 different categories for training and testing, respectively.

**Experimental Setup**

The MNIST, notMNIST and FashionMNIST datasets are zero-padded to be of size 32×32 and are replicated 3 times to create grayscale images with 3 channels, thus matching the resolution of the SVHN and CIFAR-10 datasets.

The 5-Vision Datasets Mixture benchmark which consists of five different vision datasets: MNIST, notMNIST, FashionMNIST, SVHN, CIFAR-10 for tasks $\{T_i\}_{i=1}^5$.

CIFAR-10 images. Here, we use a CNN architecture that is similar to the one used in (Ritter et al., 2018; Zeno et al., 2018), which consists of 2 convolutional layers with 20 and 50 channels respectively, and a kernel size of 5. Each convolution layer is followed by LeakyReLU nonlinearities (negative threshold of 0.3) and 2×2 max-pooling operations with stride 2. The two convolutional layers are followed by an FC layer of size 500 before the final softmax output layer (see Table 6.4 for details of the architecture). Similar to the Split MNIST benchmark, a multi-headed approach was used because the class definitions are different between datasets. The initial $\eta$ parameter value was set to 0.0001. We train the network with mini-batches of size 32 and using plain SGD with a fixed learning rate of 0.01 for 50 epochs per task.
Table 6.4: Network architecture used for 5-Vision Mixture Datasets benchmark in 6.4. For convolutional layers, the output size denotes channel size of output. The negative threshold for all of the LeakyReLU nonlinearities were set to 0.2.

<table>
<thead>
<tr>
<th></th>
<th>output size</th>
<th>kernel</th>
<th>padding</th>
<th>stride</th>
</tr>
</thead>
<tbody>
<tr>
<td>convolution</td>
<td>20</td>
<td>(5, 5)</td>
<td>(1,1)</td>
<td>(1,1)</td>
</tr>
<tr>
<td>LeakyReLU</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MaxPool</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>convolution</td>
<td>50</td>
<td>(5, 5)</td>
<td>(1, 1)</td>
<td>(1, 1)</td>
</tr>
<tr>
<td>LeakyReLU</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>MaxPool</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>fully-connected</td>
<td>500</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LeakyReLU</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>fully-connected</td>
<td>10</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

For the 5-Vision Datasets Mixture experiments shown in Figure 6.10 the regularization hyperparameter $\lambda$ for each of the task-specific consolidation methods is $\lambda = 100$ for Online EWC, $\lambda = 0.1$ for SI and $\lambda = 1.0$ for MAS. In SI, the damping parameter, $\xi$, was set to 0.1. To find the best hyperparameter combination for each of these synaptic consolidation methods, we performed a random search using the same task sequence ordering (MNIST, notMNIST, FashionMNIST, SVHN and CIFAR-10). For Online EWC, we tested values of $\lambda \in \{10, 50, 100, \ldots, 500\}$, SI — $\lambda \in \{0.01, 0.05, 0.1, \ldots, 1.0\}$ and MAS — $\lambda \in \{0.01, 0.05, 1.0, \ldots, 4.5, 5.0\}$.

Results

In the other benchmark problems, we use a single $\eta$ across all connections. In this benchmark, our model has a trainable $\eta$ value for each connection in the final output layer thus, $\eta \in \mathbb{R}^{m \times d}$ and we set the initial $\eta$ value to be 0.0001. We found that using separate $\eta$ parameters for each connection improved the stability of optimization and convergence to optimal test performance. This allows each plastic connection to modulate its own rate of plasticity when learning new experiences. We observed that using a single $\eta$ value across all connections lead to instability of optimization on the SVHN and CIFAR-10 tasks.
We found that DHP Softmax plus MAS provides a 2.14 ± 0.55% improvement over MAS on its own (see Table 6.5 and Figure 6.10). Also, SI with DHP Softmax outperforms other competitive methods with an average test performance of 81.75 ± 0.60% after learning all five tasks and an improvement of 21.73 ± 1.10% over the naive Finetune method.

Table 6.5: The test accuracy on 5-Vision Mixture and the improvement in test accuracy achieved over the naive Finetune method. Results are averaged over 10 trials.

<table>
<thead>
<tr>
<th>Method</th>
<th>Test Accuracy (%)</th>
<th>Improvement over Finetune (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>60.02 ± 0.50</td>
<td>−</td>
</tr>
<tr>
<td>DHP Softmax</td>
<td>62.94 ± 0.70</td>
<td>(↑ 2.92 ± 1.20)</td>
</tr>
<tr>
<td>SI</td>
<td>81.26 ± 0.42</td>
<td>(↑ 21.24 ± 0.92)</td>
</tr>
<tr>
<td>DHP Softmax + SI</td>
<td>81.75 ± 0.60</td>
<td>(↑ 21.73 ± 1.10)</td>
</tr>
<tr>
<td>Online EWC</td>
<td>78.61 ± 0.66</td>
<td>(↑ 18.59 ± 1.16)</td>
</tr>
<tr>
<td>DHP Softmax + Online EWC</td>
<td>79.10 ± 0.80</td>
<td>(↑ 19.08 ± 1.30)</td>
</tr>
<tr>
<td>MAS</td>
<td>78.51 ± 0.32</td>
<td>(↑ 18.49 ± 0.82)</td>
</tr>
<tr>
<td>DHP Softmax + MAS</td>
<td>80.66 ± 0.22</td>
<td>(↑ 20.64 ± 0.72)</td>
</tr>
</tbody>
</table>

Table 6.6 summarizes the average test performance for each task, $T_1$: MNIST, $T_2$: notMNIST, $T_3$: FashionMNIST, $T_4$: SVHN and $T_5$: CIFAR-10 after learning all tasks sequentially. In Table 6.7, we present a summary of the final average test performance after learning all tasks in the respective continual learning problems, as reflected on the plots in Figures 6.2, Figure 6.6, Figure 6.8 and Figure 6.10. Here, we summarize the average test performance across trials repeated over ten random seeds for each of the continual learning benchmarks considered.

6.5 Summary

In this chapter, we evaluated our proposed Differentiable Hebbian Consolidation model by demonstrating its ability to reduce the amount of catastrophic forgetting in several continual learning benchmarks.
Figure 6.10: The average test accuracy on a sequence of 5 different vision datasets $T_{n=1:5}$. The average test accuracy over all learned tasks is provided in the legend. The error bars correspond to the standard error across 10 trials.

Table 6.6: The per-task average test accuracy (%, higher is better) after learning all tasks on the 5 vision datasets mixture benchmark. nMNIST refers to notMNIST, fMNIST refers to FashionMNIST and C10 refers to CIFAR-10. The results are averaged over 10 trials.

<table>
<thead>
<tr>
<th>Method</th>
<th>MNIST</th>
<th>nMNIST</th>
<th>fMNIST</th>
<th>SVHN</th>
<th>C10</th>
</tr>
</thead>
<tbody>
<tr>
<td>SI</td>
<td>93.21 ± 0.40</td>
<td>93.53 ± 0.20</td>
<td>86.84 ± 0.70</td>
<td>81.20 ± 0.40</td>
<td>51.13 ± 0.60</td>
</tr>
<tr>
<td>DHP Softmax + SI</td>
<td>93.37 ± 0.40</td>
<td>89.23 ± 0.80</td>
<td>86.03 ± 0.60</td>
<td>80.84 ± 0.80</td>
<td>59.27 ± 0.40</td>
</tr>
<tr>
<td>Online EWC</td>
<td>83.16 ± 1.60</td>
<td>84.44 ± 0.70</td>
<td>85.89 ± 0.30</td>
<td>83.38 ± 0.50</td>
<td>55.72 ± 0.20</td>
</tr>
<tr>
<td>DHP Softmax +</td>
<td>86.93 ± 1.20</td>
<td>81.77 ± 1.30</td>
<td>83.72 ± 0.80</td>
<td>81.25 ± 0.60</td>
<td>61.82 ± 0.10</td>
</tr>
<tr>
<td>Online EWC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MAS</td>
<td>98.96 ± 0.01</td>
<td>92.29 ± 0.20</td>
<td>81.74 ± 1.00</td>
<td>73.00 ± 0.70</td>
<td>42.53 ± 0.60</td>
</tr>
<tr>
<td>DHP Softmax + MAS</td>
<td>99.04 ± 0.01</td>
<td>92.40 ± 0.10</td>
<td>84.27 ± 0.40</td>
<td>78.11 ± 0.20</td>
<td>49.47 ± 0.40</td>
</tr>
</tbody>
</table>

We discussed the test performance of our model after learning $N$ number of tasks sequentially and compared these evaluations against several competitive baselines in the literature (e.g., Online EWC, SI and MAS). We observed that our model, in terms of test performance, consistently improves a neural net-
Table 6.7: The average test accuracy (%) after learning all tasks on each continual learning benchmark, respectively. The results are averaged over 10 trials.

<table>
<thead>
<tr>
<th>Method</th>
<th>Permutated MNIST</th>
<th>Imbalanced Permutated MNIST</th>
<th>Split-MNIST</th>
<th>5-Vision</th>
</tr>
</thead>
<tbody>
<tr>
<td>Finetune</td>
<td>76.73 ± 0.67</td>
<td>76.44 ± 0.99</td>
<td>90.43 ± 0.55</td>
<td>60.02 ± 0.50</td>
</tr>
<tr>
<td>DHP Softmax</td>
<td>78.49 ± 0.29</td>
<td>80.85 ± 0.65</td>
<td>98.23 ± 0.42</td>
<td>62.94 ± 0.70</td>
</tr>
<tr>
<td>SI</td>
<td>84.72 ± 0.93</td>
<td>85.92 ± 0.60</td>
<td>97.77 ± 0.64</td>
<td>81.26 ± 0.42</td>
</tr>
<tr>
<td>DHP Softmax + SI</td>
<td>85.20 ± 0.32</td>
<td>85.39 ± 0.43</td>
<td><strong>99.15 ± 0.09</strong></td>
<td><strong>81.75 ± 0.60</strong></td>
</tr>
<tr>
<td>Online EWC</td>
<td>86.24 ± 0.70</td>
<td>87.18 ± 0.36</td>
<td>97.65 ± 0.44</td>
<td>78.61 ± 0.66</td>
</tr>
<tr>
<td>DHP Softmax + Online EWC</td>
<td>87.30 ± 0.50</td>
<td>87.43 ± 0.34</td>
<td>98.96 ± 0.20</td>
<td>79.10 ± 0.80</td>
</tr>
<tr>
<td>MAS</td>
<td>88.52 ± 0.50</td>
<td>87.32 ± 0.29</td>
<td>98.24 ± 0.13</td>
<td>78.51 ± 0.32</td>
</tr>
<tr>
<td>DHP Softmax + MAS</td>
<td><strong>89.53 ± 0.65</strong></td>
<td><strong>88.80 ± 0.52</strong></td>
<td>98.43 ± 0.33</td>
<td>80.66 ± 0.22</td>
</tr>
</tbody>
</table>

work’s ability to mitigate catastrophic forgetting while maintaining simplicity and flexibility with our approach. In addition, our model requires very minimal additional hyperparameter tuning effort on the learnable $\eta$ parameter, thus indicating its ease of implementation in practice.
Chapter 7

Conclusions

We have shown that the problem of catastrophic forgetting in continual learning environments can be alleviated by adding compressed episodic memory in the softmax layer through DHP and performing task-specific updates on synaptic parameters based on their individual importance for solving previously learned tasks. The compressed episodic memory allows new information to be learned in individual traces without overlapping representations, thus avoiding interference when added to the structured knowledge in the slow changing weights and allowing the model to generalize across experiences. The $\alpha$ parameter in the plastic component automatically learns to scale the magnitude of the plastic connections in the Hebbian traces, effectively choosing when to be less plastic (protect old knowledge) or more plastic (acquire new information quickly). The neural network with DHP Softmax showed noticeable improvement across all benchmarks when compared to a neural network with a traditional softmax layer that had an extra set of slow changing weights. The DHP Softmax does not introduce any additional hyperparameters since all of the structural parameters of the plastic part $\alpha$ and $\eta$ are learned, and setting the initial $\eta$ value required very little tuning effort.

We demonstrated the flexibility of our model where, in addition to DHP Softmax, we can regularize the
slow weights using EWC, SI or MAS to improve a model’s ability to alleviate catastrophic forgetting after sequentially learning a large number of tasks with limited model capacity. In the benchmarks where multiple readout heads were used for each task (e.g. Split MNIST and 5-Vision Datasets Mixture), DHP Softmax combined with SI outperforms other consolidation methods. The approach where we combine DHP Softmax and MAS consistently leads to overall superior results compared to other baseline methods on the Permuted MNIST and Imbalanced Permuted MNIST benchmarks. This is interesting because the local variant of MAS does compute the synaptic importance parameters of the slow weights $\Theta_{i,j}$ layer by layer based on Hebb’s rule, and therefore synaptic connections $i,j$ that are highly correlated would be considered more important for the given task than those connections that have less correlation. This gives a strong indication that Hebbian plasticity enables neural networks to learn continually and remember distant memories, thus reducing catastrophic forgetting when learning from sequential datasets in dynamic environments. Furthermore, continual synaptic plasticity can play a key role in learning from limited labelled data while being able to adapt and scale at long timescales. We hope that our work will open new investigations into gradient descent optimized Hebbian plasticity for learning and memory in DNNs to enable continual lifelong learning.

### 7.1 Future Work

There are a couple limitations with the Differentiable Hebbian Consolidation model which include: 1) implementation complexity in the multi-head setup and 2) the Hebbian update algorithm which adds an additional loop in the final output layer. In the multi-head setup, we require having to keep track of the Hebbian traces corresponding to the appropriate readout layer for each task. Moreover, the Hebbian update in the final output layer adds an additional complexity depending on the number of unique classes in the mini-batch of target labels. An interesting line of future work would be to investigate the dual-weight memory technique exploited in this work to deeper layers within the neural network. This would require an unsupervised version of the Hebbian update rule that does not require feedback directly from the class labels during training. As a result, this may lead to better consolidation without having to explicitly compute the importance parameters after each task is learned using synaptic memory consolidation
methods such as EWC, SI or MAS. In this case, a fixed capacity neural network can perform continual learning without depending on task descriptors to identify the task boundaries, thus learning in a more online manner. Also, in the online setup, it would also be interesting to study how this method can maximize forward transfer, where only a small number of training examples are required to learn new tasks efficiently in a continual fashion without forgetting.
References


64


66


Sensitivity Analysis

We provide a summary of the sensitivity analysis performed on the Hebb decay term $\eta$ and show its effect on the final average test performance after learning a sequence of tasks in the continual learning setup. The plots on the left and center in Figure A.1 show the effect of the initial $\eta$ value on the final test performance after learning tasks $T_{n=1:10}$ in a sequential manner for the Permuted MNIST and Imbalanced Permuted MNIST benchmarks, respectively. We swept through a range of values $\eta \in \{0.1, 0.01, 0.001, 0.0005, 0.0001\}$ and found that setting $\eta$ to low values led to the best performance in terms of being able to alleviate catastrophic forgetting. Similarly, we also performed a sensitivity analysis on the $\eta$ parameter for the Split MNIST problem (see the rightmost plot in Figure A.1). Table A.1 presents the average test accuracy across 5 trials for the MNIST-variant benchmarks, which corresponds to the sensitivity analysis plots in Figure A.1.
Figure A.1: A sensitivity analysis on the Hebb decay term $\eta$ in Eq. 5.2. We show the average test accuracy for different initial values of $\eta$ after learning all tasks on the (left) Permuted MNIST, (center) Imbalanced Permuted MNIST and (right) Split MNIST problems. The shaded regions correspond to the standard error of the mean across 5 trials.

Table A.1: The average test accuracy (%, higher is better) for different initial $\eta$ values after learning all tasks on the Permuted MNIST, Imbalanced Permuted MNIST and Split MNIST continual learning benchmarks, respectively. The results are averaged over 5 trials.

<table>
<thead>
<tr>
<th>Hebbian Plasticity Decay Term ($\eta$)</th>
<th>Permuted MNIST</th>
<th>Imbalanced Permuted MNIST</th>
<th>Split-MNIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\eta = 0.1$</td>
<td>77.43 ± 1.17</td>
<td>80.00 ± 1.20</td>
<td>98.43 ± 0.32</td>
</tr>
<tr>
<td>$\eta = 0.01$</td>
<td>78.60 ± 0.30</td>
<td>80.13 ± 0.37</td>
<td>98.47 ± 0.33</td>
</tr>
<tr>
<td>$\eta = 0.001$</td>
<td>78.49 ± 0.29</td>
<td>80.85 ± 0.65</td>
<td>98.32 ± 0.25</td>
</tr>
<tr>
<td>$\eta = 0.0001$</td>
<td>77.83 ± 0.22</td>
<td>80.47 ± 1.03</td>
<td>98.43 ± 0.32</td>
</tr>
<tr>
<td>$\eta = 0.0005$</td>
<td>78.05 ± 0.25</td>
<td>80.40 ± 1.39</td>
<td>98.37 ± 0.38</td>
</tr>
</tbody>
</table>
Appendix B

PyTorch Implementation of DHP Softmax

class DHP_Softmax_Layer(nn.Module):
    def __init__(self, in_features, out_features, eta_rate=0.001):
        super(DHP_Softmax_Layer, self).__init__()
        """Applies a linear transformation to the hidden activations of the last hidden layer with an additional plastic component implemented using Differentiable Hebbian Plasticity (DHP): math: z = (w + \alpha \cdot Hebb)h'.

Args:
    in_features: size of each input in last hidden layer.
    out_features: number of classes.
    eta_rate: initial learning rate value of plastic connections.

Returns:
    z: the softmax pre-activations (unnormalized log probabilities).
    hebb: the updated Hebbian traces for the next iteration.
""
    self.in_features = in_features
    self.out_features = out_features
    self.eta_rate = eta_rate
```python
# Initialize fixed (slow) weights with He initialization.
self.weight = Parameter(torch.Tensor(self.in_features,
        self.out_features))
init.kaiming_uniform_(self.weight, a=math.sqrt(5))

# Initialize alpha scaling coefficients for plastic connections.
self.alpha = Parameter((.01 * torch.rand(self.in_features,
        self.out_features)),
        requires_grad=True)

# Initialize the learning rate of plastic connections.
self.eta = Parameter((self.eta_rate * torch.ones(1)),
        requires_grad=True)

def forward(self, h, y, hebb):
    if self.training:
        for _, c in enumerate(torch.unique(y)):
            # Get indices of corresponding class, c, in y.
            y_c_idx = (y == c).nonzero()
            # Count total occurrences of corresponding class, c in y.
            s = torch.sum(y == c)

            if s > 0:
                # Perform Hebbian update (lines 6-7 in Algorithm 1)
                h_bar = torch.div(torch.sum(h[y_c_idx], 0),
                        s.item())
                hebb[: , c] = torch.add(torch.mul(torch.sub(1, self.eta),
                        hebb[: , c].clone()),
                        torch.mul(h_bar, self.eta))

            # Compute softmax pre-activations with plastic (fast) weights.
            z = torch.mm(h, self.weight + torch.mul(self.alpha, hebb))

    return z, hebb

def initial_zero_hebb(self):
    return Variable(torch.zeros(self.in_features, self.out_features)),
```
Listing B.1: PyTorch implementation of the DHP Softmax model which adds a compressed episodic memory to the final output layer of a neural network through plastic connections as described in Algorithm 1. We want to emphasize the simplicity of implementation using popular DL frameworks.