University of St Andrews | FOUNDED 1413

Towards New Generation Deep Learning Neural Networks

Submitted August 2020, in partial fulfillment of
the conditions for the award of the degree **MSc Artificial Intelligence.**

Anirudh Apparaju

190013978

Supervised by Ognjen Arandelovic

School of Computer Science

University of St. Andrews

Signature: A. Anirudh

Date: 14 / 08 / 2020

Declaration

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated. The main text of this project report is 10,807 words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Abstract

Artificial neural networks (ANN) are simplified models of biological neural networks that can be trained to perform specific tasks. End-to-end backpropagation [1] is a supervised learning technique to efficiently train multi-layer ANN's. However, while ANN's were inspired by biological neural networks, the end-to-end backpropagation algorithm to train them is biologically implausible. This implausability arises because of the following important characteristics of this training algorithm that do not occur in biological systems: non-local learning rules; symmetry of forward and backward weights; and the requirement of a large labelled dataset.

To overcome this limitation, Krotov and Hopfield developed a semi-supervised learning algorithm [2] motivated by Hebb's idea [3] that changes in synapse strength are defined by local learning rules involving just the pre- and post-synaptic neurons. Krotov and Hopfield implemented this learning algorithm to train a neural network (referred to as a "KH model") that resulted in having comparable classification performance to neural networks trained with end-to-end backpropagation (referred to as "classic NN") [2; 4].

In addition to the issue of biological implausibility, there exist inherent issues from a machine learning standpoint regarding classic NN models [5]. One such limitation is that the backpropagation learning algorithm is very data demanding and often requires millions of labelled training examples to function effectively.

The work described in this dissertation investigates the ability of the KH model to train effectively under conditions of scarce labelled training data. Moreover, random noise is injected into the input to test the robustness of the KH model [6]. To this end, I implemented a KH model based on the paper by Krotov and Hopfield [2] that learns, in an unsupervised manner, good feature detectors in the lower layers of the neural network that are useful for tasks to be performed effectively by higher layers of the network [2; 7].

I ran a series of data scarcity experiments whose results indicate that the KH model performs better than the classic NN model under circumstances of limited labelled training data. This performance provides promise for using such architectures in situations where labelled data is scarce. However, experiments that add noise to data indicate that the KH model is less robust to random noise than a neural network trained with end-to-end backpropagation.

Acknowledgements

This dissertation would not have been possible without the support of many people in the unprecendented times of the Covid-19 pandemic.

First and foremost, I would like to thank my supervisor Oggie Arandelovic. Not only did you introduce me to the multidisciplinary fields underlying Artificial Intelligence (Neural Computation, Representation Learning, Biological Learning and many others), but also provided me with the freedom to explore and pursue research in the direction I wanted to.

With all of my dissertation work happening through a global pandemic and times of physical distancing, I am truly grateful to the support and camaraderie from all my friends here at St. Andrews (in alphabetical order): AJ, Andreea, Azamat, Balint the Boi, Claire, Farouq, Golf, Gunay, Guylain, Hassan, Jess, Katerina, Kitkat, Marika, Neha, Ritomo, Sajjad, Sakina, Saurav, Sergio, Shams, Shubham, ShuenJen, Steinar and Vasu.

A special shout out goes to my dearest friends Dilawar, James and Mary Jane: the three musketeers without whom my time at St. Andrews would have been a truly impoverished experience.

My biggest appreciation goes out to my dearest mother, father and brother. My family has been a bedrock of support and their power of unconditional love, regardless of circumstance, has enabled to to rise to new heights academically, professionally and most importantly, personally. Without needing to say any words, they have shown me the true meaning of the inspirational phrase, "You'll Never Walk Alone".

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

The main aim of this dissertation project is to explore and investigate the means by which the mechanisms behind biologically plausible learning algorithms could help to overcome limitations of modern artificial neural networks (ANN) trained with the classic end-to-end backpropagation algorithm (BP). Throughout this dissertation report, the terms "biologically plausible learning algorithm" and its variations are used rather metaphorically to mean an algorithm which follows the most important constraints of biological neural networks. Specifically, the work presented in this report strives to evaluate the effectiveness of the Hebbian synaptic plasticity-inspired learning algorithm proposed by Krotov and Hopfield [2] to mitigate the ANN training requirement of a very large labelled dataset, and to evaluate their model's robustness.

The following are the main tasks of this project:

i. Implement the semi-supervised model, including an unsupervised "biological" learning algorithm, proposed by Krotov and Hopfield [2]. Their model presented in their paper [2] is referred to throughout this work as the "KH model";

ii. Evaluate possibility of obtaining similar or better generalization results (with the model proposed by Krotov and Hopfield [2] as compared to a network trained with classic end-to-end backpropagation) using fewer labelled examples of MNIST and CIFAR-10;

iii. Evaluate robustness of the KH model [2] by introducing random noise into the data.

1.2 Achievements

The work presented in this dissertation achieves the following goals:

i. Successfully implemented a PyTorch implementation of KH model [2]. Trained and applied this network to classification tasks for MNIST and CIFAR-10, and achieved generalization accuracy (on a held-out test set) of 97.39% and 48.62% respectively. Moreover, this PyTorch implementation includes the unsupervised "biological" learning algorithm proposed by Krotov and Hopfield [2] that generates feature detectors for MNIST and CIFAR-10 similar to those in the original paper [2];

ii. Under conditions of scarce labelled training data ($\leq 1\%$ of total labelled training data for MNIST and $\leq 4\%$ for CIFAR-10), my implementation of the KH model achieves better generalization accuracy than a fully-connected network trained with classic end-to-end backpropagation;

iii. Determined that the KH model is less robust to noisy input data compared to a fully-connected network trained with classic end-to-end backpropagation.

1.3 Thesis Structure

The remainder of the present thesis is organized as follows. Chapter 2 provides a background of ideas relevant to our work, and describes related work in this space. Chapter 3 outlines the design of the networks and algorithms developed based on Krotov and Hopfield's paper [2]. Chapter 4 contains details about various experiments run on these models, including a discussion of the results from these experiments. Chapter 5 concludes the report by considering the importance of the findings, and provides recommendations on future work that builds on the work presented in this report.

Chapter 2

Context Survey

2.1 Background

Techniques incorporating the concepts of Deep Learning [9] were developed to attempt to solve a diverse range of complicated tasks, including the performance of complex cognitive tasks. These programs are based on ANN's whose origins lie in the perceptron: the simplest abstraction of a single-layer neural network proposed by Frank Rosenblatt in 1957-1960 [10]. However, the inability of a Rosenblatt perceptron to recognize many classes of patterns [11], especially the simple logical operation XOR that outputs 'true' only when its 2 inputs differ, and the inability of multilayer neural networks to be trained using Rosenblatt's algorithms resulted in sub-optimal use of these architectures for many decades.

2.1.1 End-to-end Backpropagation

End-to-end backpropagation is a technique to efficiently train multi-layer neural networks. The technique was developed by researchers in the 60's but the first to propose using it on neural networks was PhD student Paul Werbos in his 1974 thesis [1; 12]. Despite solving the question of how multilayer neural networks could be trained, and testing the process while on his PhD thesis, Werbos did not publish until 1982 due to the chilling effects of the AI Winter at the time. He even proposed the paper to Marvin Minsky while visiting MIT to no response [12].

It was a decade later that this method was popularized by Rumelhart et al. [7]. In spite of a solution being available years ago, it was this paper in 1986 that showed how multilayer neural nets could be trained, and resolved issues pertaining to limitations of the perceptron that were raised decades earlier by Minsky and Papert [11]. The potential power of ANNs was only properly utilised after this popularisation of end-to-end backpropagation [7],

and it has been very effective in enabling ANN's to be applied to solve a diverse range of complicated tasks including processing natural language and images [9], playing strategy and arcade games [13; 14; 15], and more recently, autonomous navigation.

After each forward pass through the ANN, backpropagation uses the chain rule to perform a backward pass to propagate errors to train the network by adjusting the model's synaptic weights. Figure 2.1 succinctly shows this propagation.

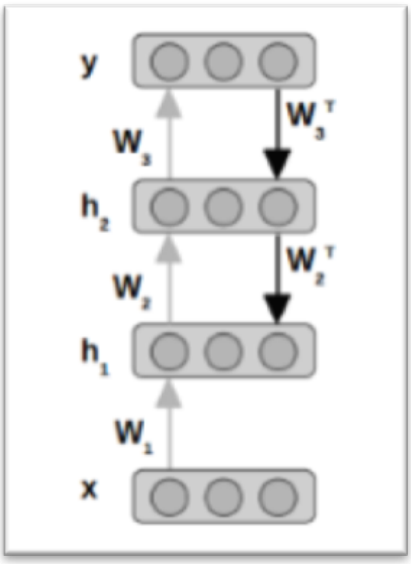


Figure 2.1: Backpropagation of errors in a network uses the same weights (transposed) as the forward pass. Diagram adopted from Nokland's paper on the topic [8].

2.1.2 Biological Implausibility of End-to-end Backpropagation

While ANN's were developed based on simplified models of biological neural networks, the end-to-end backpropagation algorithm used to train them is biologically implausible. This implausibility challenges the functional correctness of biological brain models that utilise end-to-end back-propagation in its learning processes.

The following are main reasons for questioning the biological realism of the end-to-end backpropagation algorithm:

1. Lack of local error representation/non-local learning rules: The classic end-to-end backpropagation algorithm uses a non-local rule for updating the weights between nodes, while Hebb motivates that change in synaptic strength should depend only on the neurons local to that synapse [3].

2. Symmetry of forward and backward weights: In ANN's the weights during forward (information) and backward (error) propagation are identical for the same synapse.

The symmetry in such ANN models suggests that identically weighted forward and backward connections should exist between biological neurons. However, this is not the case as best stated by Whittington and Bogacz, "Although bidirectional connections are significantly more common in cortical networks than expected by chance, they are not always present" [16].

3. The end-to-end backpropagation algorithm requires a very large labelled dataset, while biological neural systems use unsupervised learning with observations from their extensive sensory experience to train feature detectors [2].

4. Unrealistic models of neurons: ANNs use neurons that fire a continuous output while real neurons output discrete spikes [17].

Hence, researchers have strived to develop new models of ANN's using biologically plausible learning algorithms to try to provide more effective insights into processes in the brain [16]. There is a diverse range of such algorithms which are further discussed in Section 2.2. Studies in this domain have helped researchers, especially neuroscientists, to build better neurocomputational models that help to better understand and investigate learning processes in the brain [2; 8; 16; 18; 19; 20; 21; 22].

Herein, the interest is in the impact of these more biologically plausible algorithms on the field of Machine Learning rather than on Neuroscience. As a result, spiking neurons are not considered and discussion is limited to exploring algorithms that attend to the first three issues above. As best expressed by Bartunov et al., "it should nevertheless be possible to gain algorithmic insight to the brain without tackling all of the elements of biological complexity simultaneously" [19]. The authors were referring to insight into workings of the brain in that context, but similar logic is applied in this work to set aside spiking neurons so that the focus can be entirely applied to evaluating the learning algorithm (and the effect on synapse strengths) rather than evaluating the effect of different types of neurons.

2.1.3 Limitations of End-to-end Backpropagation

From a purely machine learning standpoint, despite the great success of deep learning models trained using end-to-end backpropagation (BP), the architecture of these developed models contains several inherent issues [5]. These limitations arise because ANN's trained with end-to-end BP are:

1. Very data demanding and often require millions of labelled training examples to function effectively.

2. Easily fooled by adversarial examples.

3. Computationally intensive to train and deploy (tractably requires GPU's).

4. Can be subject to algorithmic bias.

5. Poor at representing and conveying uncertainty (how can one know what the model knows?)

6. Difficult to encode structure and prior knowledge during learning.

7. Uninterpretable black boxes (which are difficult to establish trust in).

8. Often require expert knowledge to design, fine-tune architectures.

To this effect, researchers have evaluated to what extent the biological plausible learning algorithms can help to overcome some of these limitations [23]. The work presented in this dissertation report builds on this prior work, which is summarised in the next section, particularly in addressing the first limitation outlined above.

2.2 Related Work

Aided by the accessibility of image datasets of varying levels of complexity (MNIST, CIFAR-10, ImageNet), there is an increasing body of work dedicated to developing algorithms that perform similar training as end-to-end backpropagation but without breaking any "rules" of neurobiology. These proposed "bio-plausible" algorithms adopt a wide range of algorithmic ideas including supervised learning algorithms like feedback alignment [8; 20] and target propagation [21], and various unsupervised learning techniques [18; 19; 22; 23]. While many of these algorithms have been developed with the intention of understanding more about learning processes in the brain they have also been evaluated for practical use by analysing generalization performance on classification tasks on the MNIST, CIFAR-10 and ImageNet datasets [18; 19; 22].

Each of the aforementioned learning algorithms previously mentioned has its own unique network with its own distinct characteristics. This section briefly describes and evaluates each of these proposed biologically plausible learning algorithms and corresponding ANN architectures through the lens of various themes of interest.

2.2.1 Supervised Learning Approaches

Target Propagation

The main idea of target propagation (TP) is to associate with each feedforward unit's activation value a target value rather than a loss gradient. These computed targets, like gradients, are propagated backwards. In a way that is related but different from previously proposed proxies for back-propagation, which rely on a backwards network with symmetric weights, target propagation relies on auto-encoders at each layer.

Lee et al. describe how this general idea of target propagation by using auto-encoders to assign targets to each layer can be employed for supervised training of deep neural networks [21]. Their experiments show that target propagation performs comparable to back-propagation on ordinary deep networks and de-noising auto-encoders. Moreover, target propagation can be directly used on networks with discretized transmission between units and reaches state of the art performance for stochastic neural networks on MNIST.

Feedback Alignment

The feedback alignment (FA) algorithm was proposed by Lillicrap et al. [16; 20] as an effective and simple solution to the weight symmetry problem presented in the previous section of this chapter.

The main idea underlying feedback alignment is that when propagating errors backwards, use a random matrix rather than the transpose of the synaptic weight matrix used on the forward pass. Said another way, this algorithm assigns random feedback weights to synapses. Feedback alignment address the limitation of weight symmetry present in classic backpropagation. This notion is supported by studies [19] which have shown that good generalisation performance on classification tasks can be achieved by randomly back-propagating errors. Moreover, this mechanism transmits such error signals across multiple layers of neurons and is shown to perform training as effectively as the classic end-to-end backpropagation on several tasks [20].

Moreover, the feedback alignment algorithm has a couple sister algorithms called "Direct Feedback Alignment" (DFA) and "Indirect Feedback Alignment" (IFA) [8]. While the original FA algorithm showed that the weights used for propagating the error backward need not be symmetric with the weights used for forward propagation of the neuron activation, the DFA and IFA algorithms can run on networks that disconnect the feedforward path from the feedback path so that each layer is not reciprocally connected to the layer above (and below). These architectures are displayed in Figure 2.2 for easy reference.

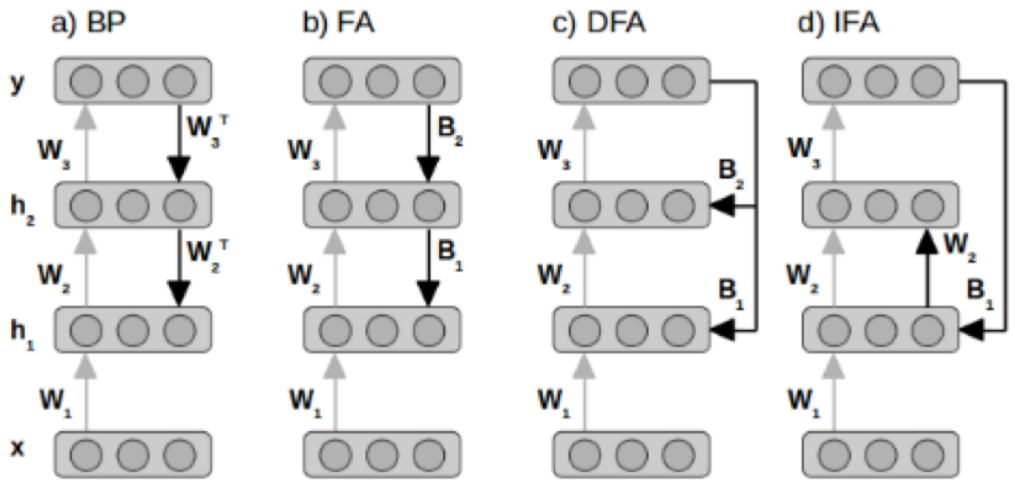


Figure 2.2: Figure reproduced from Nokland's paper [8]. This figure shows different network configurations for transferring activation information and error. "Weights that are adapted during learning are denoted as W_i, and weights that are fixed and random are denoted as B_i. a) Back-propagation. b) Feedback alignment. c) Direct feedback alignment. d) Indirect feedback alignment" [8]. Networks c) and d) show that learning is possible in feedback loops where the forward and feedback paths are disconnected.

What sets feedback alignment apart from target propagation is that the former relies on implicit dynamics inherent in the algorithm to evolve the weight matrices, while the latter relies on autoencoders, a tangible construct of the network. This implies more simplicity in network architectures that incorporate FA: a theme that is a major factor of interest to the work presented in this report.

2.2.2 Semi-supervised Learning Approaches

In prior works, unsupervised and semi-supervised learning algorithms were evaluated according to the representations of the data they were able to learn (feature detectors) in a biologically-plausible manner [18]. Below are brief descriptions of a few influential algorithms.

Autoencoder (AE) and Restricted Boltzmann Machines (RBM)

A popular unsupervised learning approach is to train a hidden layer to reproduce the input data as in AE's and RBM's [24]. The AE and RBM networks trained with a single hidden layer are quite relevant because adjusting weights of the input-to-hidden-layer connections relies on local gradients, and the representations can be stacked on top of each other to extract hierarchical features.

Unsupervised Learning with Hidden Competing Units (KH Model)

The recently proposed "KH model" [2] addresses the problem of learning with local gradients by learning hidden representations solely using an unsupervised method. In the network the input-to-hidden connections are trained and additional (non-plastic) lateral inhibition provides competition within the hidden layer. For evaluating the representation, the weights are kept fixed, and a linear classifier trained with labels is used for the final classification. Moreover, this unsupervised algorithm was also used to establish weights in a locally-connected convolutional neural network (CNN) [4].

Bayesian Confidence Propagation Neural Network (BCPNN)

The feedforward BCPNN model is a probabilistic graphical model with a single hidden layer. It frames the update and learning steps of the neural network as probabilistic computations, the mechanics of which are structurally outlined by Ravichandran et al. [18].

The receptive fields generated by all these algorithms support the claim that "local bottom-up unsupervised training is capable of learning useful and task-independent representations of images in networks" [4]. These feature descriptors are then used as a base of a model that is supplied labelled data as part of the supervised part of training. Moreover, these useful features are necessary to achieve a good generalization performance in line with networks trained with end-to-end BP. Furthermore, the results from experiments conducted on these networks suggest that localised receptive fields enable better generalization than networks with full connectivity [4; 18] to the extent that networks with localised receptive fields can reach classic backpropagation performance on MNIST.

That being said, there is a balance to strike between generalization performance and network efficiency and simplicity. Whittington and Bogacz suggest that "there is a clear evolutionary benefit to propagating information via fewer synapses, as it would result in faster responses and a smaller number of noise sources" [16]. I aim to follow this neuroscientific theme of favouring efficient and simple architectures to use for performing experiments and evaluations.

Furthermore, Grinberg et al. [4] best summarize an important finding that describes an important difference between the supervised and unsupervised approaches outlined above. They conclude that supervision is not crucial for learning useful early layer representations from the data: "This result is at odds with the common belief that the first layer feature detectors should be crafted specifically to solve the narrow task specified by the top layer

classifier" [4]. Hence, this unsupervised approach not only has biological plausibility (where it overcomes issue number 3 mentioned in Section 2.1.2), but also has strong generalisation performance that make these approaches strongly preferable for further study. In particular, the KH model is of notable significance to us because it incorporates the concept of Hebbian synaptic plasticity [3], resulting in synapse strength updates that are locally determined.

2.2.3 Scalability

Scalabiity is an important concept to consider when evaluating the performance of a learning algorithms. There are several factors that contribute to the 'scalability performance' of a model (CPU cycles, memory usage, generalization performance amongst others) but what I are interested in my work is the generalization performance of the learning algorithm to more advanced and complex tasks.

The importance of scalability of generalization performance of learning algorithms is best outlined by Bartunov et al.: "the brain possesses a powerful, general-purpose learning algorithm for shaping behavior. As such, researchers can, and should, seek learning algorithms that are both more plausible physiologically, and scale up to the sorts of complex tasks that humans are capable of learning. Augmenting a model with adaptive capabilities is unlikely to unveil any truths about the brain if the model's performance is crippled by an insufficiently powerful learning algorithm. On the other hand, demonstrating good performance with even a vanilla artificial neural network provides evidence that, at the very least, the learning algorithm is not limiting" [19].

From this statement, one can infer that the key to exploring and improving the impact of learning algorithms seems not to be in actually in augmenting a model with additional capabilities but rather by focusing on the learning algorithm itself. While the former optimisations would work in actually improving the performance of a model, they would not necessarily provide deep insight into the learning algorithm itself. Hence, following the type of methodology and analysis performed by Bartunov et al. [19] and Illing et al. [22], the impact of a bio-plausible learning algorithm in this dissertation is measured by observing the generalization performance on different datasets.

2.2.4 Competitive Learning

Competitive learning is an unsupervised learning paradigm which "provides a way to discover the salient, general features which can be used to classify a set of patterns" [25]. The mechanics of this learning paradigm involves a competitive mechanism that is capable

of discovering a set of feature detectors that capture important aspects of input stimulus patterns.

Rumelhart and Zipser analysed these learning processes to "further our understanding of how simple adaptive networks can discover features important in the description of the stimulus environment in which the system finds itself in" [25]. This learning paradigm is appealing because the salient features learned are useful for classification tasks.

The basic components of such a competitive learning scheme are:

1. Start with a set of units with a randomly distributed parameter which enables each unit to respond slightly differently to inputs.
2. Limit unit "strength" [25].
3. Units compete in a way for the "winner-takes-all" right to respond to input.

Applying these components to a learning paradigm enables units of the model to learn to specialize on groups of similar patterns (thus becoming feature detectors) [25]. These feature detectors can then be used as starting weights in a semi-supervised model which is supplied labelled data in its supervised component (Section 2.2.2). In this regard, competitive learning can be considered a form of representation learning where "a good representation is one that makes a subsequent learning task easier... Specifically, we can learn good representations for the unlabeled data, and then use these representations to solve the supervised learning task" [24].

The KH model's [2] unsupervised component is based on this competitive learning paradigm and incorporates the basic functional components described above. Moreover, the architecture and mechanics of the KH model's unsupervised learning component strongly mirror that of the competitive learning system described by Rumelhart and Zipser in that it "consists of a set of hierarchically layered units in which each layer connects, via excitatory connections, with the layer immediately above it... each unit of a layer receives an input from each unit of the layer immediately below and projects to each unit in the layer immediately above. Moreover, within a layer, the units are broken into a set of inhibitory clusters in which all elements within a cluster inhibit all other elements in the cluster. Thus, the elements within a cluster at one level compete with one another to respond to the (input) pattern" [25].

2.2.5 Model Robustness

The importance of model robustness is best surmised by Hendrycks et al. [6]: "The human vision system is robust in ways that existing computer vision systems are not [26; 27].

Unlike current deep learning classifiers [28; 29; 30], the human vision system is not fooled by small changes in query images. Humans are also not confused by many forms of corruption such as snow, blur, pixelation, and novel combinations of these. Humans can even deal with abstract changes in structure and style. Achieving these kinds of robustness is an important goal for computer vision and machine learning. It is also essential for creating deep learning systems that can be deployed in safety-critical applications."

Most of the work on evaluating robustness of deep learning models for vision has targeted the following challenges [6]: robustness to adversarial examples [31; 32; 33]; unknown unknowns [34; 35; 36]; and data/model poisoning [37; 38]. In addition, Hendrycks et al. [6] conduct a comprehensive series of experiments on the ImageNet dataset to establish rigorous standards on image classifier robustness. Specifically, unlike other recent robustness research, their work "evaluates performance on common corruptions and perturbations not worst-case adversarial perturbations" [6]. This comprehensive analysis establishes a precedent and guide as to how to properly explore and evaluate a model's robustness.

2.3 Conclusion

The primary aim of my work is not to design a biologically plausible deep learning framework to better understand learning processes in the brain. Rather, the stress here is on the field of machine learning and characteristics of deep neural networks rather than neuroscience. Specifically, as stated previously, the main goal of this work is to investigate a biologically plausible learning algorithm and its usefulness in addressing current limitations with ANN's. I specifically use the KH model [2] introduced in this chapter because the simplicity of the network architecture provides an ideal setting to evaluate the effectiveness of the learning algorithm itself.

Chapter 3

Design

Of all the biologically-plausible models introduced in the previous chapter, the Krotov-Hopfield (KH) model [2] is of particular interest because of the innate simplicity of its network architecture. This chapter provides a brief overview of both the fully connected feedforward network and the KH model, and describes the details of the biologically plausible learning algorithm used by the latter. These details include both a description of the mathematical framework underpinning the KH model originally detailed by Krotov and Hopfield [2], and a description of the infrastructure of each model.

3.1 Problem Statement

This section begins by reiterating the discussion in the introductory chapter to repeatedly stress on the motivation behind this work. My goal is to explore the potential of a biologically plausible algorithm in terms of determining whether this algorithm might be valuable from the artificial intelligence (AI) and machine learning perspective, as opposed to the biological motivation. This idea ties closely to the intentions and evaluations performed by Krotov and Hopfield [2]. Of particular interest is the potential ability of the KH model to address some of the shortcomings of ANN's trained with classic end-to-end backpropagation as discussed in Section 2.1.3.

3.2 Krotov-Hopfield Model

The Krotov-Hopfield (KH) model is a semi-supervised, biologically-plausible model that consists of two separately trained components. The first component is an unsupervised layer trained with an algorithm based of an implementation of Hebbian synaptic plasticity [2; 3]. The second component is a fully connected perceptron trained with conventional

stochastic gradient descent. Further sections will dive into the details of each of these components. It is important to note that this KH model was developed with the goal of engineering a learning algorithm and training model that leads to a good generalization performance given the unsupervised learning aspect incorporating locality of synaptic plasticity rules [2; 3].

As briefly mentioned previously, it is worth noting that the "biologically-plausible" aspect of this model refers to the behaviour of the learning algorithm that obeys the most important constraints of biological neural networks as outlined in section 2.1.2. Moreover, Krotov and Hopfield recognize that the "proposed algorithm omits many known aspects of neurobiology" [2], and so the terms "biological algorithm", "biologically plausible", "biological model" etc. are more figurative than literal titles.

3.2.1 Model Design

I use the same network architecture as in Krotov and Hopfield's paper [2] which is shown in Figure (3.1). This architecture consists of one fully connected network with one hidden layer: there is a layer of visible neurons v_i, a layer of hidden neurons h_μ, and a layer of output neurons c_α. This network is trained using a semi-supervised method based off the concepts of Hebbian Synaptic Plasticity [2; 3] and conventional stochastic gradient descent techniques.

Moreover, this model requires no backpropagation of signals past the final layer, which utilises stochastic gradient descent (SGD), and which is the only place where any form of gradient descent is applied. Furthermore, all of the information in the network is transmitted only from forward-propagating signals. This behaviour gives credence to the "biologically plausible" description assigned to this model in that all the synapse updates happen locally, with no information propagated from any subsequent nodes in the network.

The supervised and unsupervised components are discussed in further detail below.

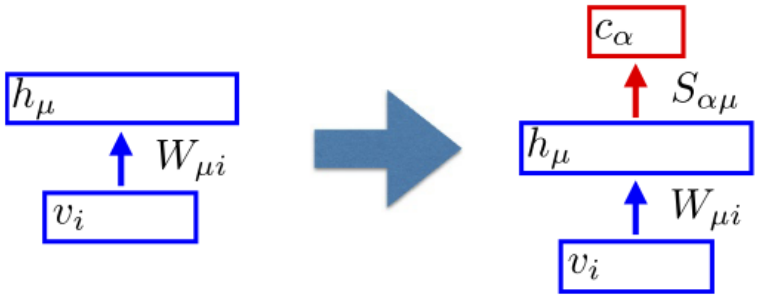


Figure 3.1: "Lower layers of the neural network (weights $W_{\mu i}$) are trained using the proposed biological learning algorithm. Once this phase is complete, the weights are plugged into a fully connected perceptron. The weights of the top layer $S_{\alpha\mu}$ are then learned using SGD in a supervised way." Figure and quote referenced from Krotov and Hopfield's paper on the subject [2].

Forward Pass

Based on this the network shape, the forward pass on this network is specified by the following equations from Krotov and Hopfield's paper [2] (we can sum over repeated indices of i and μ):

$$\begin{cases} h_\mu = f(W_{\mu i} v_i) \\ c_\alpha = tanh(\beta S_{\alpha\mu} h_\mu) \end{cases} \quad where \quad f(x) = \begin{cases} x^n, & x \geq 0 \\ 0, & x < 0 \end{cases} \tag{3.1}$$

where

- μ represents an external index that corresponds to the index of the hidden unit that we are trying to update.
- $W_{\mu i}$ represents the synapse weights that correspond to the index of the hidden unit that we are trying to update.
- v_i represents the layer of input neurons.
- h_μ represents the layer of hidden neurons.
- c_α represents the layer of output neurons.
- β is a numerical constant.
- $S_{\alpha\mu}$ represents the weights of the top layer.

- n represents a constant ≥ 1.
- f(x) corresponds to rectified linear unit (ReLU) raised to the power of n.
- tanh represents the hyperbolic tangent function.

3.2.2 Unsupervised Component

Mathematical Framework

As previously mentioned, Krotov and Hopfield [2] base the mathematical underpinnings of the unsupervised learning algorithm on concepts of Hebbian learning and Hebbian synaptic plasticity [3]. The key characteristic of standard Hebbian learning is that in order to update the synapse between two cells, we consider just the activities of the pre-synaptic and post-synaptic cells [3; 39]. To this end, Krotov and Hopfield [2] present a comprehensive description and explanation of computations which involve dynamical processes that support this tenet of standard Hebbian learning. A summary of these mechanisms is available in Figure (3.2).

In addition, Krotov presents a subtly different version of this computational algorithm in a guest lecture at MIT on this very topic [39]. This presented version of the unsupervised learning algorithm computations is discussed subsequently.

Activations of Hidden Units

The theoretical workings of the KH model depend on dynamical processes that can be represented with differential equations. One of these differential equations defines the steady-state activations of the hidden units. These equations and processes describe a form of Hebbian learning with competition between hidden units [2]. To reiterate, this stage is unsupervised and so requires no labels for the data presented.

After presenting data to the KH network, the activities of the hidden neurons are calculated according to the following differential equation (3.2). This is presented as equation [3] in Krotov and Hopfield's paper [2]. The following equation defines the dynamics that lead to the steady state activations of the hidden units [2].

$$\tau \frac{\partial h_\mu}{\partial t} = I_\mu - \omega_{inh} \sum_{v \neq u} r(h_v) - h_\mu \tag{3.2}$$

where

- τ represents a positive constant that defines the overall timescale of these dynamical processes.

- μ represents an external index that corresponds to the index of the hidden unit that we are trying to update.
- h_μ represents the activity of the hidden layer.
- I_μ represents the input current. This is calculated by taking the dot product of the weights and incoming data. It is calculated as $I_\mu = \langle \mathbf{W}, \mathbf{v} \rangle = \sum_{i=1}^{N} W_{\mu i} v_i$ where N is the number of input nodes of the network.
- $\omega_{inh} \sum_{v \neq \mu} r(h_v)$ represents a term that introduces competition between hidden units. When we present all the data to the network, all the hidden units want to be activated. However, if some hidden units become more strongly activate than others, those more strongly activated hidden units will suppress the activations of the others. Hence, subtracting this term introduces some inhibitory connections between hidden nodes. By introducing these inhibitory connections, the network stops being strictly feed-forward as lateral connections are introduced in the hidden layer.
- ω_{inh} is the parameter that defines the strength of the global inhibition.
- h_v represents the activity of the hidden layer nodes v that are different from nodes μ.
- r represents the ReLU activation function.

Temporal Competition

Rather than just applying the postulates of classic Hebbian learning, a slight twist is applied where instead of directly taking the activity of the post-synaptic cell, we use the results of a function applied to the activity of the post-synaptic cell. This function is non-linear and is represented in Equations [9, 10] in Krotov and Hopfield's paper [2] where they best describe the mechanics of the function. Below is a brief summary of the mechanics they described.

g(h) is a function that receives as input activations defined by the dynamics in (3.2). This function implements temporal competition between the steady-state activation patterns passed in. It has the following properties:

- A value h* corresponds to the point where the output of g(h) changes from a negative value to a positive value (as can be seen in Figure (3.2)).
- The region above h* corresponds to Hebbian learning.
- The region below h* corresponds to anti-Hebbian learning.

- Activities that are below zero do not contribute to training and are ignored.

The basic intuition behind using such a non-linear function is that "the synapses of hidden units that are strongly driven are pushed toward the patterns that drive them, while the synapses of those hidden units that are driven slightly less are pushed away from these patterns. Given a random temporal sequence of the input stimuli, this creates a dynamic competition between the hidden units and results in the synaptic weights that are different for each hidden unit and specific to features of the data" [2].

Synaptic Plasticity Rule

This plasticity rule performs the weight updates for synapses and is an extension of the famous Oja rule [40]. It is another dynamical process represented as a differential equation that accepts as arguments: the steady state activations from (3.2) passed through the temporal competition function g(h) described in the previous section; the input data v_i; and input current I_μ. This rule is defined by (3.3) which is written below as was presented in Krotov's MIT Lecture [39], and is slightly different from Eq. [3] in Krotov and Hopfield's paper [2], but performs the same function.

$$\tau_L \frac{\partial W_{\mu i}}{\partial t} = g(h_\mu)[v_i - (I_\mu W_{\mu i})] \tag{3.3}$$

where

- τ_L represents a positive constant that defines the overall timescale of these learning dynamics. $\tau_L >>> \tau$ from Eq. (3.2) because this learning process happens on a much larger timescale.
- μ represents an external index that corresponds to the index of the hidden unit that we are trying to update.
- $W_{\mu i}$ represents the synapse weights that correspond to the index of the hidden unit that we are trying to update.
- h_μ represents the activity of the hidden layer.
- $g(h_\mu)$ represents the Hebbian learning term as described in the previous "Temporal Competition" section.
- I_μ represents the input current. This is calculated by taking the dot product of the weights and incoming data. It is calculated as $I_\mu = \langle \mathbf{W}, \mathbf{v} \rangle = \sum_{k=1}^{N} W_{\mu k} v_k$ where N is the number of input nodes of the network.

- v_i represents the input data.
- $[v_i - (I_\mu W_{\mu i})]$ represents a specially engineered term which ensures that the synaptic weights $W_{\mu i}$ do not grow to be too large. This constraint is inspired from neuroscientific models of biological learning where there are all kinds of homeostatic constraints that ensure that the biological neural synapses do not become too strong [39]. Hence, this term ensures that the synapses are somewhat limited in size.

  Moreover, the way this equation is engineered ensures that the fixed point of the dynamics in the long term (as time tends to infinity) satisfies the following constraint: $\sum_{i=1}^{N} W_{\mu k}^2 = 1$. This constraint ensures that the weights connecting one hidden unit with all divisible units eventually converge to the surface of a unit sphere. This organisation happens dynamically and is "a nice feature to have" [2; 39].

Summary

In summary, what is happening in the unsupervised learning component is that we present the network with data (in the form of mini-batches or online) and wait for the system to equilibrate according to (3.2). After a steady state in hidden neurons is achieved, the synaptic plasticity learning rule defined by (3.3) is applied to adjust the weights. This process is repeated for as many cycles as specified.

The weight updates dependencies can be succinctly summarised in the following equation.

$$\Delta W_{\mu i} \sim g(h_\mu) v_i \tag{3.4}$$

The intuition of using lateral inhibition in conjunction with Hebbian synaptic plasticity draws inspiration from a series of papers where it is also known as competitive or winner-takes-all learning [2; 7; 25; 41; 42; 43; 44]. Moreover, the concept of global inhibition has also been used in several unsupervised learning algorithms [45; 46; 47; 48; 49].

This unsupervised algorithm in the KH model accepts raw image data as input and tries to find 'usable' representations of this data. Moreover, this unsupervised phase is given no explicit task regarding what to do with the data (classification, regression etc.) [2]. These learned representations are similar to the high level features learned in the feature learning stage of convolutional neural networks (CNN's) [5; 39], and so this unsupervised algorithm performs a form of representation learning [50]. Figures (4.1) and (4.2) show the feature detectors learned by the unsupervised algorithm for MNIST and CIFAR-10 classification, which are similar to those learned by the early convolutional layers of CNN's applied to the same dataset [51; 52]. As identified by Krotov and Hopfield (amongst several other researchers), these feature detectors resemble the simple observed selectivities of the response of neurons in early visual processing areas in the brains of higher animals [2].

Furthermore, this unsupervised process, as suggested by Krotov in his MIT lecture [39], maps the incoming data into its latent space representation which contains all the important information needed to represent the original data points [5].

Fast Implementation

However, as theoretically effective as this algorithm may be, Krotov and Hopfield [2] identify and point out a few time-consuming properties of this algorithm which would make it intractable to apply in practice. One of these drawbacks has to do with the time-complexity of waiting for the hidden units to reach a steady state in the presented algorithm. As a solution, their paper [2] presents a "Fast Implementation" version which is an approximation of this algorithm. This "Fast Implementation" version is what is implemented for running experiments discussed in the next Chapter, and the complete description is available from the "A Fast Implementation" section of the original paper [2].

To summarize the algorithmic difference from the original implementation, Krotov and Hopfield's words are most effective: "First, instead of solving dynamical equations we use the currents as a proxy for ranking of the final activities" [2] and (3.5) represents this change. Given that ranking, units are pushed towards or away from the training examples; hence, this provides a "heuristic (that) significantly speeds up the algorithm" [2].

What exactly this means from a mathematical perspective is that for this "Fast Implementation", one just assumes the steady state for (3.2) which looks like below. This equation is referenced from Krotov's lecture at MIT [39].

$$\frac{\partial h_\mu}{\partial t} = 0 \implies h_\mu = I_\mu \tag{3.5}$$

This equation does not have the second term in (3.2) that introduces inhibitory connections between hidden neurons scaled by a constant that represents global inhibition. In spite of this sacrifice, the "Fast Implementation" algorithm proves to be an effective one as indicated by Krotov and Hopfield's experiments [2; 4] and by our own experiments which are presented in Chapter 4.

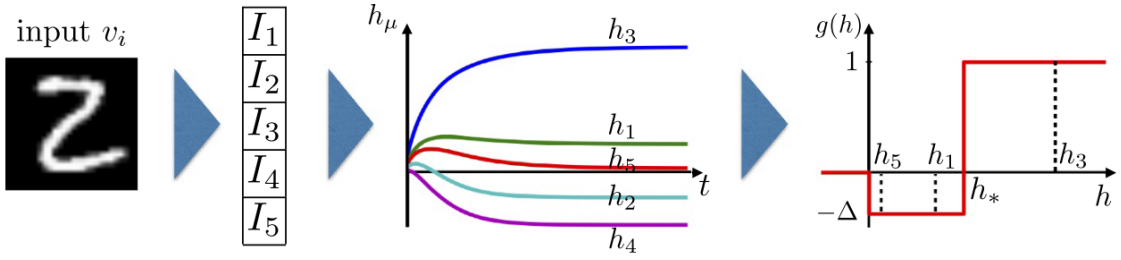


Figure 3.2: The pipeline of the training algorithm. "Inputs v_i are converted to a set of input currents I_μ. These currents define the dynamics (3.2) that lead to the steady-state activations of the hidden units. These activations are used to update the synapses using the learning rule (3.3). The learning activation function changes the sign at h*, which separates the Hebbian and anti-Hebbian learning regimes. The second term in the plasticity rule (3.3), which is the product of the input current I_μ and the weight $W_{\mu i}$." Figure and quote referenced from Fig. [2] in Krotov and Hopfield's paper on the subject [2].

3.2.3 Supervised Component

The supervised component applies to only the very last synapse and corresponding nodes of the network, which altogether can be treated as a fully connected perceptron. Using Figure (3.1) as a reference, the supervised aspect applies to the hidden layer h_μ, the final layer of weights $S_{\alpha\mu}$, and the layer of output neurons c_α. This section of the model is trained with classic stochastic gradient descent (SGD) techniques with details available in the original paper [2]. The specific configurations used in our experiments are described in Chapter 4. This final layer that applies SGD is the only part of the whole KH model where labelled data is used.

3.3 Classic Fully-Connected Network

We follow the established example of Krotov and Hopfield [2] in using a fully connected ANN as a control and baseline to evaluate generalization performance against. As mentioned previously, we are focused on using as simple a model as possible so as to focus our attention on the learning algorithm itself, and not on any specific attributes or characteristics of the network architecture. Grinberg et al. [4] build on the work presented in their 2019 paper [2] to evaluate generalization performance (amongst other things) on convolutional neural networks (CNN's). The results obtained and observations made are very interesting; however, to reiterate, we make the explicit choice of keeping the network as simple as possible in order to focus our analysis on the learning algorithm itself.

Hence, we use only a fully connected network with a single hidden layer which looks exactly like the network on the right of Figure (3.1) but with all weights trained using

conventional end-to-end backpropagation techniques. Moreover, this network uses the activation function specified in Section 3.2.1.

3.4 Intuition

The differences in training algorithms suggests some intuitive ideas that are explored and experimented with in the following chapter. Firstly, the KH model requires a labeled dataset for training only the final set of weights, while the fully-connected network requires labels to train all sets of weights across all layers with classic backpropagation. This difference suggests that it should be possible to train the KH model with fewer labeled examples. Taking this into account, the main question explored in the next chapter is, "At what threshold of labelled data scarcity does the KH model consistently outperform the classic NN based on generalization performance?" A second question we explore is, "How robust and stable is the KH model against noisy input?" The experiments and results are detailed in the following chapter.

Chapter 4

Implementation and Evaluation

This chapter describes the implementation details (Section 4.1) which includes a brief overview of the datasets and their train-validation and train-test splits strategies (Section 4.1.1); the architecture configuration (Section 4.1.2) based off the design outlined in Chapter 3; and the choice of hyperparameters (Section 4.1.3). Subsequent sections discuss the process and metrics used in evaluating the models (Section 4.2); establish the baseline results (Section 4.2.2) that will be used to compare against when evaluating the experimental results; and describe the feature detectors (Section 4.2.3) learned by the models. The rest of the chapter describes the experiments and the corresponding results (Section 4.3).

4.1 Implementation

I used the PyTorch framework to implement and run experiments on a KH model (described in Chapter 3) and a classic neural network trained with end-to-end backpropagation (referred to as 'classic NN'). Krotov and Hopfield's paper was the source of inspiration [2] for architecture details, and I adopted their public GitHub repository [53] which implements the unsupervised learning algorithm. Moreover, I consulted GitHub repositories that provided introductory references for using the PyTorch framework appropriately [54], GitHub repositories that provided a basic PyTorch implementation of the KH Model [55], and other sources (GitHub and otherwise) for references on recommended development practices in PyTorch. These references were used to build a PyTorch implementation of both the KH and classic NN models [56] while corroborating with the original authors when possible and required. Moreover, for techniques with no publicly available code at the time of this report, I adhered to following the theoretical guidelines presented in the original paper [2].

Moreover, all the experiments and training were performed on a host provided by The

University of St. Andrews with a 6 GB NVIDIA GeForceGTX 1060 graphics card using CUDA version 10.2, and with the Linux distro: CentOS Linux release 7.8.2003.

4.1.1 Datasets

Like in the paper of inspiration [2], two publicly available datasets were used: MNIST and CIFAR-10. MNIST consists of 60,000 images as training data and 10,000 images as test data [57]. When performing validation tests for MNIST, I use a random 80-20 train-validation split of the training data resulting in 48,000 training images and 12,000 validation images. Moreover, each MNIST image is a small square 28x28 pixel grayscale image of a handwritten single digit between 0 and 9.

The CIFAR-10 dataset consists of 50,000 images as training data and 10,000 images as test data [58]. Like with the MNIST dataset, I use a random 80-20 train-validation split of the training data when performing validation tests. This split results in 40,000 training images and 10,000 validation images. All CIFAR-10 images are 32x32 colour images (3 channels of RGB) split evenly into 10 categories [58].

4.1.2 Architecture Configuration

I followed the architectural configuration described in Krotov and Hopfield's paper [2] and whose design is documented in Chapter 3. A brief description of this architecture is presented in this section.

This configuration involves using two fully connected neural networks with one hidden layer of 2000 hidden units. To be specific, the network shape for MNIST classification is $(784 \rightarrow 2000 \rightarrow 10)$, while the shape for CIFAR-10 classification is $(3072 \rightarrow 2000 \rightarrow 10)$. For each dataset, one network is trained with the aforementioned semi-supervised learning algorithm (making the network a KH model) while the other network is trained with classic end-to-end backpropagation.

The full configuration settings of the networks are available from the "Testing the Model" and "Appendix B" sections of Krotov and Hopfield's paper [2]. As mentioned previously, I implemented their "Fast Implementation" version for the unsupervised component of the KH model. While the reader is left to be informed of all the details of the training architecture from that paper [2], the cost function used in the supervised part (for both the KH model and classic NN model) is pasted in (4.1) for easy reference.

Cost Function

"The supervised part of the training was done using the loss function (labels t_α are one-hot–encoded vectors of Nc = 10 units of ±1)" [2].

$$C = \sum_{examples} \sum_{\alpha=1}^{N_c} | \, c_\alpha - t_\alpha \, |^m \tag{4.1}$$

where

- α represents an index that corresponds to the index of the unit in the final output layer.
- N_c represents the number of output neurons.
- c_α represents the prediction made by the network in the form of a vector of N_c units.
- t_α represents the actual labels in the form of one-hot-encoded vectors of N_c units.
- m represents a constant and serves as a hyperparameter (Section 4.1.3).

4.1.3 Choice of Hyperparameters

There are several hyperparameters to be specified for both the KH model and classic NN model. I use values that worked best in my implementation, and which are slightly different to those used by Krotov and Hopfield [2]. The ideal set of hyperparameters was identified after running several validation tests with the randomly constructed 80-20 train-validation split sets (Section 4.1.1). This section outlines the hyperparameters used for each model and dataset.

MNIST

KH Model Unsupervised Component:

- Lebesgue norm = p ∈ {2, 3, 4, 5, 6} with best results obtained with p=2.
- Ranking = k ∈ {2, 3, 4, 5, 6, 7, 8} with best results obtained with k=2.
- Anti-hebbian learning strength = Δ ∈ {0, 0.1, 0.2, 0.3, 0.4} with best results obtained with Δ=0.4.
- Batch size = 100.

- Number of epochs = 1000.
- Learning rate = 0.02 at the first epoch and linearly decreases to 0 at the last epoch.

KH Model Supervised Component:

- Optimizer = Adam.
- Batch size = 100.
- Number of epochs = 600.
- Learning rate = 0.0001.
- m $\in$ {4, 6, 8, 10, 12} and use m = 6. m is used in the cost function (4.1).
- n = 4.5. n is used in the forward pass (3.2.1).
- $\beta \in \{0.01, 0.1, 1\}$ and use $\beta = 0.01$. β is used in the forward pass (3.2.1).

Classic NN Model:

- Optimizer = Adam.
- Batch size = 100.
- Number of epochs = 600.
- Learning rate = 0.001 for the first 100 epochs and "after that the learning rate decreases every 50 epochs as 0.0005, 0.0001, 0.00005, 0.00001" [2] and stays at 0.00001 thereafter.
- m $\in$ {4, 6, 8, 10, 12} and use m = 6. m is used in the cost function (4.1).
- n = 1. n is used in the forward pass (3.2.1).
- $\beta \in \{0.01, 0.1, 1\}$ and use $\beta = 0.01$. β is used in the forward pass (3.2.1).

CIFAR-10

KH Model Unsupervised Component:

- Lebesgue norm = p $\in \{2, 3, 4, 5\}$ with best results obtained with p=2.
- Ranking = k $\in \{2, 3\}$ with best results obtained with k=2.
- Anti-hebbian learning strength = $\Delta \in \{0, 0.1, 0.2, 0.3\}$ with best results obtained with Δ=0.3.
- Batch size = 1000.
- Number of epochs = 1000.
- Learning rate = 0.02 at the first epoch and linearly decreases to 0 at the last epoch.

KH Model Supervised Component:

- Optimizer = Adam.
- Batch size = 10.
- Number of epochs = 500.
- Learning rate = 0.004 for the first 100 epochs and "0.004 for 100 epochs and then each 50 epochs the learning rate changed as 0.002, 0.001, 0.0005, 0.0002, 0.0001, 0.00005, 0.00002, 0.00001" [2].
- m $\in \{4, 6, 8, 10, 12\}$ and use m = 6. m is used in the cost function (4.1).
- n = 10. n is used in the forward pass (3.2.1).
- $\beta \in \{0.01, 0.1, 1\}$ and use $\beta = 0.1$. β is used in the forward pass (3.2.1).

Classic NN Model:

- Optimizer = Adam.
- Batch size = 100.
- Number of epochs = 100.
- Learning rate = 0.004 for the first 50 epochs and then 0.001 for the second 50 epochs.

- m $\in \{4, 6, 8, 10, 12\}$ and use m = 4. m is used in the cost function (4.1).
- n = 1. n is used in the forward pass (3.2.1).
- $\beta \in \{0.01, 0.1, 1\}$ and use $\beta = 0.01$. β is used in the forward pass (3.2.1).

4.2 Evaluation

4.2.1 Evaluation Process

The evaluation process of the KH and classic NN models occurs in three different stages. Firstly, I attempted to replicate the results obtained by Krotov and Hopfield [2] on MNIST and CIFAR-10 classification tasks using the KH model. The aim is to have the KH model perform as close as possible to, or even better than, how the classic NN performs on these classification tasks. This performance is measured by generalization performance on the held-out test set.

The second stage evaluation involved running scarcity and robustness tests on both types of models only after the KH model implemented provided 'decent' performance. The word 'decent' here signifies that the KH models developed (for MNIST and CIFAR-10 classification) have obtained a baseline generalization performance (Section 4.2.2) that is sufficiently reasonable but not near state of the art as indicated in Krotov and Hopfield's original paper [2]. While there is a scope for improvement, the KH model developed was good enough to run data scarcity and simple robustness experiments on to evaluate if it offers any advantages over the classic NN model.

4.2.2 Baseline Performance Results

I measure generalization performance by measuring the classification accuracy scores of the trained KH and classic NN models on the MNIST and CIFAR-10 hold-out test sets. Moreover, for each experiment, confusion matrices on this hold-out test set were generated for additional analysis. The appropriate hyperparameter values were selected after running validation tests to determine their best values and combinations. Moreover, for each set of hyperparameters, each model was trained and tested thrice with the accuracy scores averaged across the three runs.

Following this procedure, I established the baseline generalization accuracy scores of the classic NN and KH model on MNIST and CIFAR-10 datasets. For MNIST, the classic NN model gave a test-set accuracy of 98.52% while the KH model returns a test-set accuracy of 97.39%. For CIFAR-10, the classic NN model gave a test-set accuracy of 55.29% while the KH model returns a test-set accuracy of 48.63%.

I use these scores as a baseline and observe changes in this metric when running experiments.

4.2.3 Feature Detectors

As shown by Krotov and Hopfield [2], the early-layer weights learned by the different networks form the feature detectors for the networks. These feature detectors are shown for both the classic NN and KH models in Figures (4.1) and (4.2). These feature detectors represent what the models view as the underlying structure of the image data [2]. Moreover, to reiterate for emphasis, the KH model learns these feature detectors in a purely unsupervised manner as opposed to the completely supervised processes in the classic NN.

The unsupervised process of the KH model generates weights that appear to be much more 'natural' in appearance when compared to the speckled nature of weights from the classic NN model (Figures 4.1 and 4.2). Moreover, the unsupervised phase of the KH model training generates feature detectors that have areas that are both positively and negatively correlated. These negative elements suggest that these feature detectors are not just copies of the training images, which would have only positive elements. These learned weights encapsulate a latent representation of the image data [2] learned by the KH model in a purely unsupervised manner.

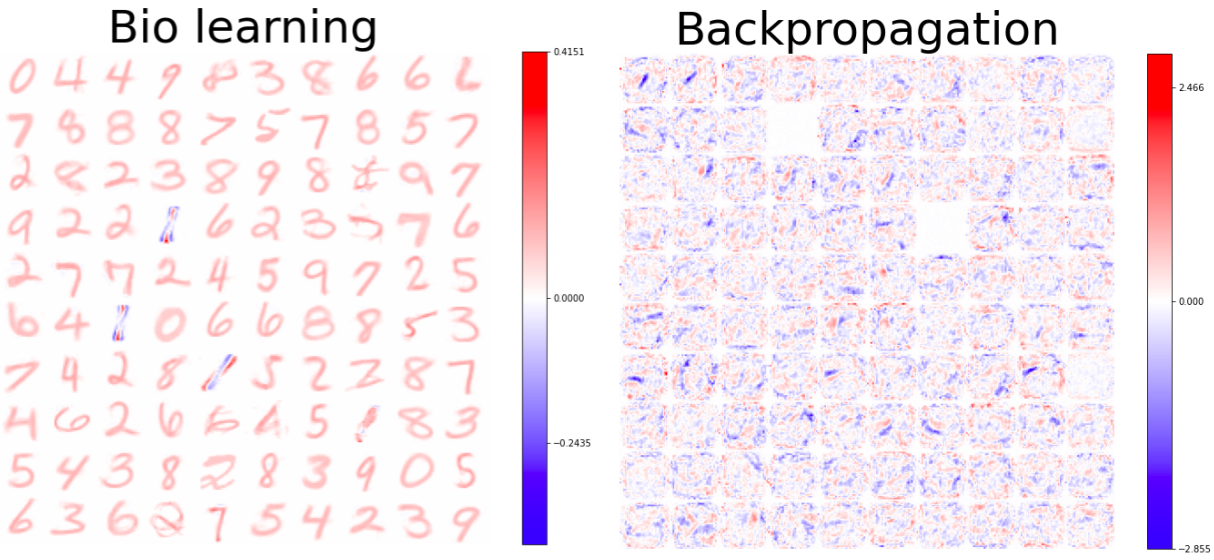


Figure 4.1: A hundred randomly selected feature detectors for MNIST classification out of 2000 are shown. (Left) Weights learned by the KH model using the unsupervised learning component of the training algorithm. (Right) Weights learned by the classic NN model using end-to-end backpropagation.

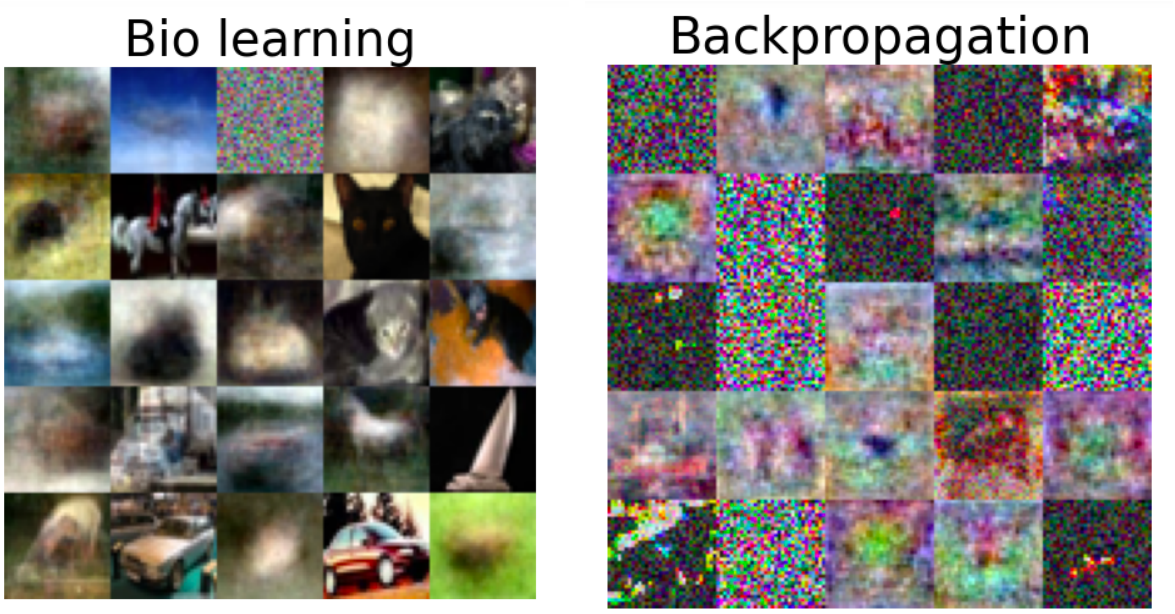


Figure 4.2: 25 randomly selected feature detectors for CIFAR-10 classification out of 2000 are shown. (Left) Weights learned by the KH model using the unsupervised learning component of the training algorithm. (Right) Weights learned by the classic NN model using end-to-end backpropagation.

4.3 Experiments

Two different sets of experiments were run on the KH and classic NN models. The first set consists of data scarcity experiments intended to evaluate how the KH model performs compared to the classic NN model when very little labelled training data is available. With this goal in mind, I ran a series of experiments described in Section 4.3.1.

The second set of experiments evaluate the robustness and stability of the KH model compared to the classic NN model. These properties are evaluated by introducing random noise into the input data and comparing how well each network performs. This set of experiments is described in Section 4.3.2.

4.3.1 Data Scarcity Experiments

These experiments are intended to evaluate how well, if at all, the 'biological' KH model can overcome the 'data hungry' limitation (Section 2.1.3) of ANN's trained with end-to-end backpropagation. Generalization performance on the hold-out test set is measured on both models after the amount of labeled training data was constricted.

Data Filtering Procedure

Labelled data is constricted according to the following procedures. There are three versions of MNIST and CIFAR-10 datasets: (i) with all the data labels present and intact; (ii) with one class of labelled images completely missing; (iii) with half of the classes of labelled images completely missing.

Moreover, for each of these three datasets, the total amount of labelled data is limited by reducing the size of the labelled training dataset available for the models. Specifically, I downsample each version of the labelled training dataset specified in the previous paragraph according to the following specifications: (i) 100% - 10% of the training data in decremental steps of 10%. (ii) 10% - 1% of the training data in decremental steps of 1%. (iii) 1% - 0.2% of the training data in decremental steps of 0.1%.

I ensured that for each dataset, downsampled or not, there is a balanced number of each class present in the dataset. Moreover, I experimented with removing entire labels from the training dataset to evaluate how well the models can classify all images when presented with many that have not been seen at all in the labelled training stage.

It is important to note again that I am testing generalization performance when labelled data is limited. As a result, the unsupervised phase of the KH model is given the whole complete dataset as input, with no missing labels or downsampled data. This input to the unsupervised component of the KH model is valid because it does not require any labelled information about the data (Section 3.2.2). This process is similar to how animals tune their visual system as succinctly phrased by Krotov and Hopfield, "higher animals require extensive sensory experience to tune the early (in the processing hierarchy) visual system into an adult system. This experience is believed to be predominantly observational, with few or no labels, so that there is no explicit task"[2].

Results

This section displays the results of running the experiments according to the procedure previously described in Section (4.3.1). The generalization performance results (based on test set accuracy) show that when a sufficiently large amount of labelled training data is available, the classic NN performs better than my implementation of the KH model.

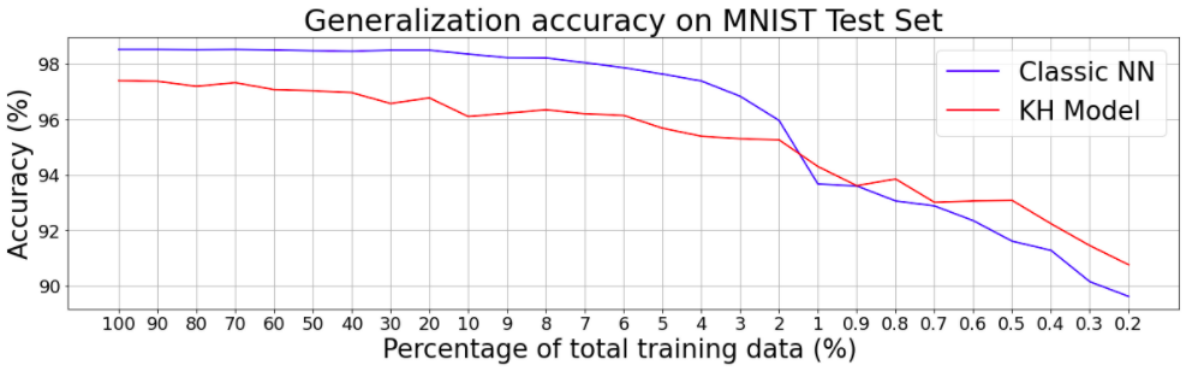
However, The KH model achieves better generalization performance than the classic NN when labeled data is very limited ($\leq 1\%$ of total available training data for MNIST and $\leq 4\%$ for CIFAR-10). The generalization performances for both models when MNIST labelled data is scarce are displayed in Figures (4.3) and (4.4).

Similar behaviour is observed when running the same experiments on the CIFAR-10 dataset. Unlike for MNIST, the CIFAR-10 baseline accuracies (Section 4.2.2) are con-

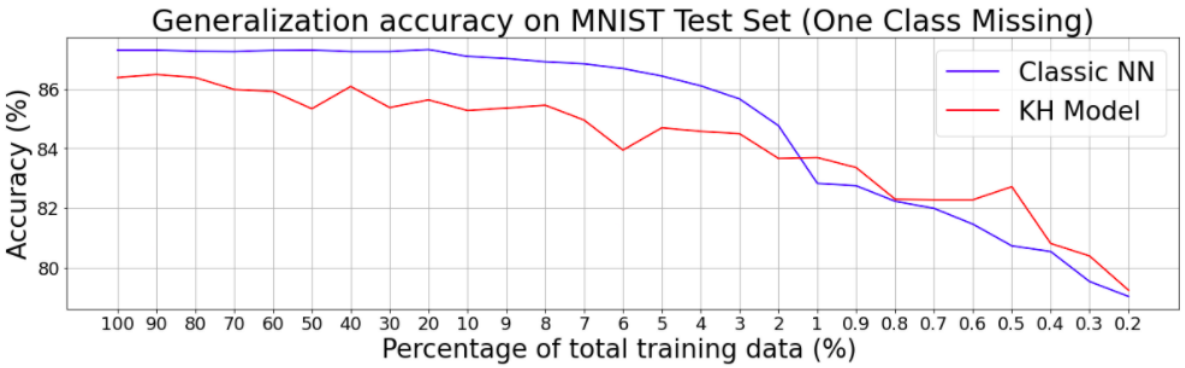
siderably lower than the state-of-the-art accuracies [2; 19; 22]. However, the scores I obtained provide a baseline metric that can be used to evaluate generalization performance degradation (measured by test accuracy degradation) when constricting labelled data in the supervised training components for both the KH and classic NN models. The generalization performances for both models when CIFAR-10 labelled data is scarce are displayed in Figures (4.5) and (4.6).

The experiments indicate that the KH model performs better than the classic NN model when labelled data is scarce. This performance suggests that the KH model was able to perform an effective form of representation learning [50] via its unsupervised component. Representation learning [50] incorporates a set of techniques that allow a system to discover the representations needed for feature detection or classification from raw data. Specifically, the experimental results suggest that the KH model was able to generate an effective latent space representation of the input data which contains important information needed to represent the original data points. This latent space representation is a key difference between the KH and classic NN models (Section 4.2.3), and hence plays an important role in the difference in generalization performances observed under conditions of labelled data scarcity. Krotov alludes to this difference in his MIT lecture [39], "Intuition is that it should be possible to train the (KH) classifier with fewer labelled examples because there are already a nice set of weights". The experimental results provide evidence to this intuition.

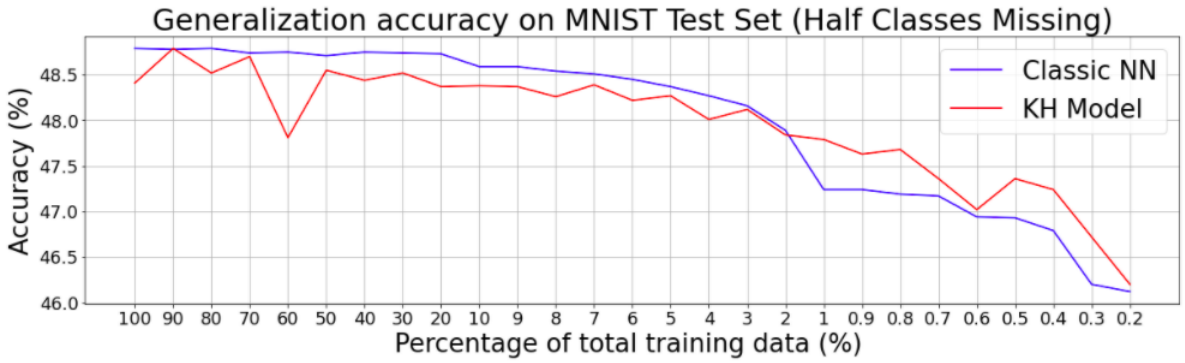
Moreover, this difference in generalization performance under conditions of labelled data scarcity is greater for the CIFAR-10 dataset than for MNIST. This behaviour raises the question of whether there a connection between the complexity of a dataset and this discrepancy in generalization performance between the 2 models when labels are scarce. Do more complicated datasets result in the KH model being even more effective at classification when labelled data is limited? These are questions to explore in further investigation (Section 5.2).



(a) All classes are in the MNIST training set.

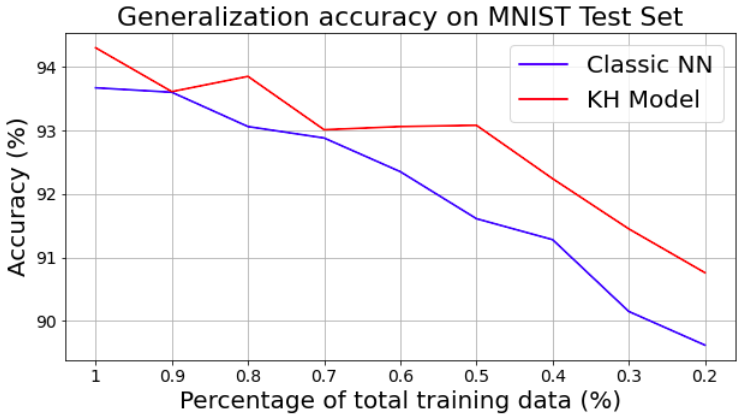


(b) The class "1" is missing from the MNIST training set.

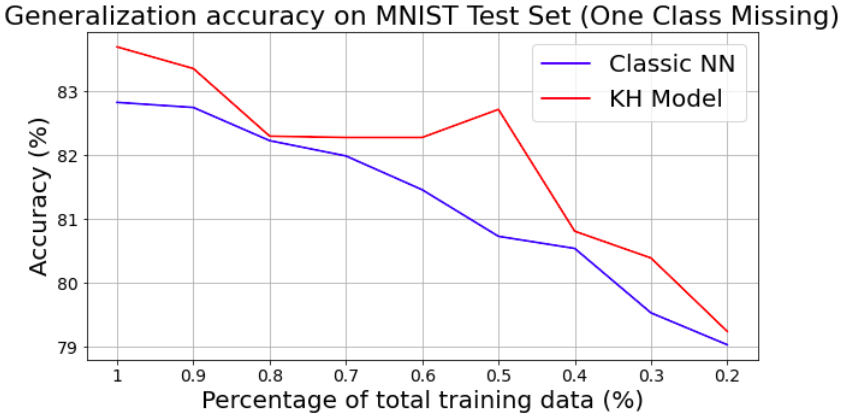


(c) Half of the classes are missing from the MNIST training set.

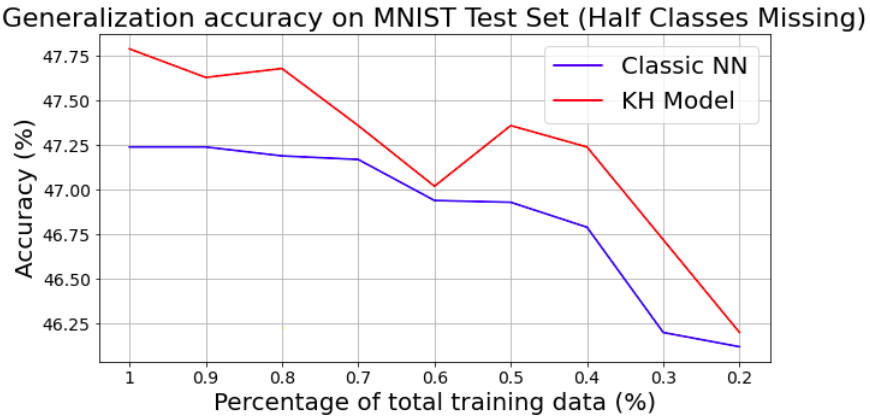
Figure 4.3: Test set accuracy for MNIST when constricting labelled training data.



(a) All classes are in the MNIST training set.

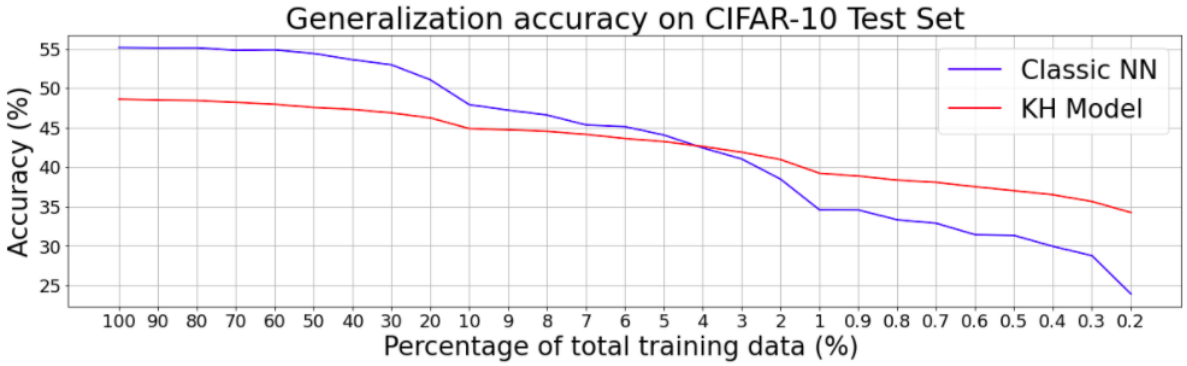


(b) The class "1" is missing from the MNIST training set.

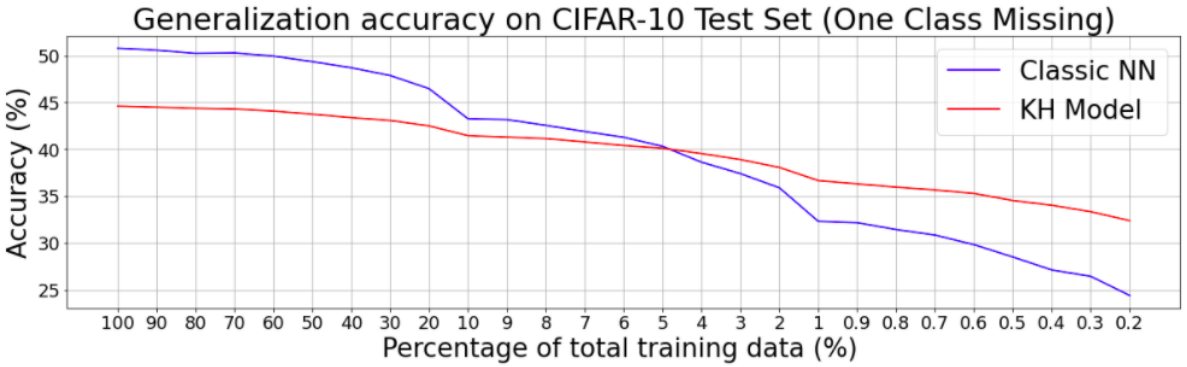


(c) Half of the classes are missing from the MNIST training set.

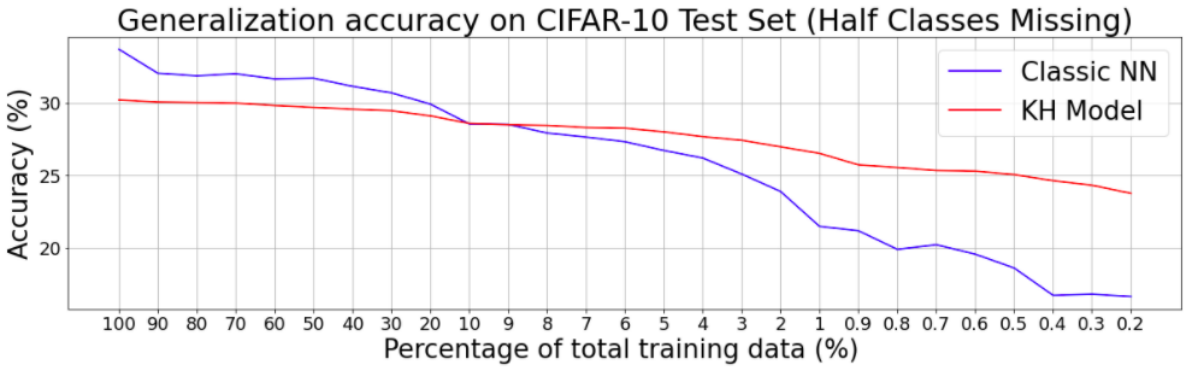
Figure 4.4: Test set accuracy for MNIST when training data is limited to less than 1% of the original training set. These plots are zoomed-in versions of Figure (4.3) to clearly show the KH model's better generalization performance with limited labelled training data.



(a) All classes are in the CIFAR-10 training set.

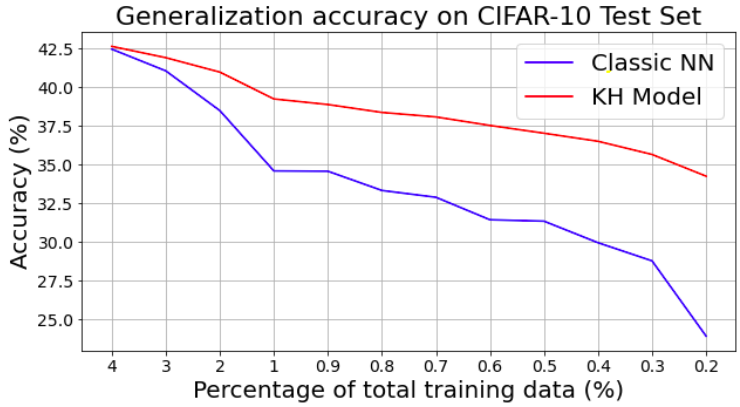


(b) The class “car” is missing from the CIFAR-10 training set.

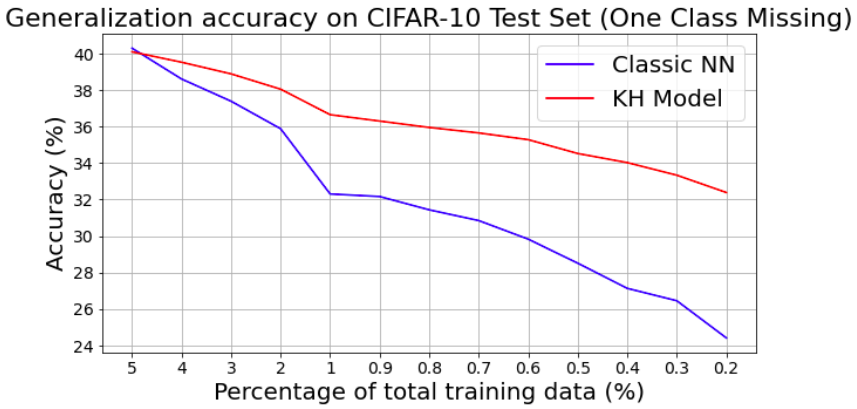


(c) Half of the classes are missing from the CIFAR-10 training set.

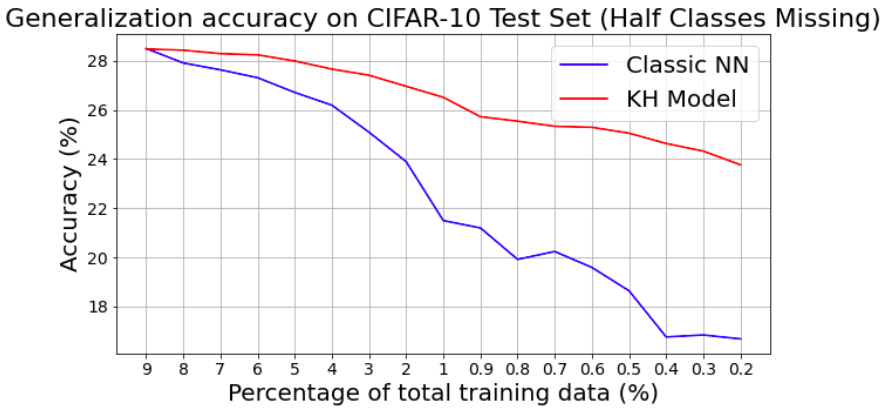
Figure 4.5: Test set accuracy for CIFAR-10 when constricting labelled training data.



(a) All classes are in the CIFAR-10 training set.



(b) The class “car” is missing from the CIFAR-10 training set.



(c) Half of the classes are missing from the CIFAR-10 training set.

Figure 4.6: Test set accuracy for CIFAR-10 when training data is limited to a small percentage of the original training set. These plots are zoomed-in versions of Figure (4.5) to clearly show where and how the KH model’s generalization performance diverges with that of the classic NN’s when both are provided input with the same limited labelled training data.

4.3.2 Network Robustness Experiments

These experiments evaluate the robustness of the KH model by injecting noise from random gaussian distributions into the MNIST and CIFAR-10 training and test images. These training images then pass through both the unsupervised and supervised phases of the KH model as well as through the supervised training phase of the classic NN model. Generalization performance of the KH model on the hold-out modified test set is measured and compared with that of the classic NN, which serves as a baseline to compare and contrast against. All training images are used and none are filtered out or constricted.

Gaussian Noise

MNIST and CIFAR-10 images are modified by adding noise from random gaussian distributions. Specifically, three different distributions are used: (i) mean=0 and standard deviation = 1; (ii) mean = 0 and standard deviation = 0.5; (iii) mean = 0 and standard deviation = 0.1. MNIST and CIFAR-10 images subjected to such noise are displayed in Figures 4.7 and 4.8.

Feature Detectors

The feature detectors learned by the unsupervised component of the KH model reflect the noise in the input data. The learned feature detectors for this experiment are increasingly speckled and uninterpretable (to the human eye) as the noise in the data increases. The feature detectors learned for both MNIST and CIFAR-10 classification are displayed in Figures 4.9 and 4.10.

Robustness to Random Noise

The generalization performance results (based on test set accuracy) indicate that the KH model is not as robust to random noise as compared to the classic NN model. The accuracy for the KH model degrades faster than that of the classic NN model for when any level of noise is introduced. The underlying cause for this may have to do with the increasingly hazy and speckled feature detectors learned by the KH model under increasing amounts of noise as seen in Figures 4.9 and 4.10. These learned feature detectors may prove to be ineffective when used by the higher layers of the network. However, this hypothesis needs additional research and analysis (Section 5.2).

These results suggest that the KH model is more susceptible to random permutations and corruption of incoming signals than the classic NN model, and that the KH model

may be fooled by small changes or perturbations in the input images. These results are summarized in Tables 4.1 and 4.2.

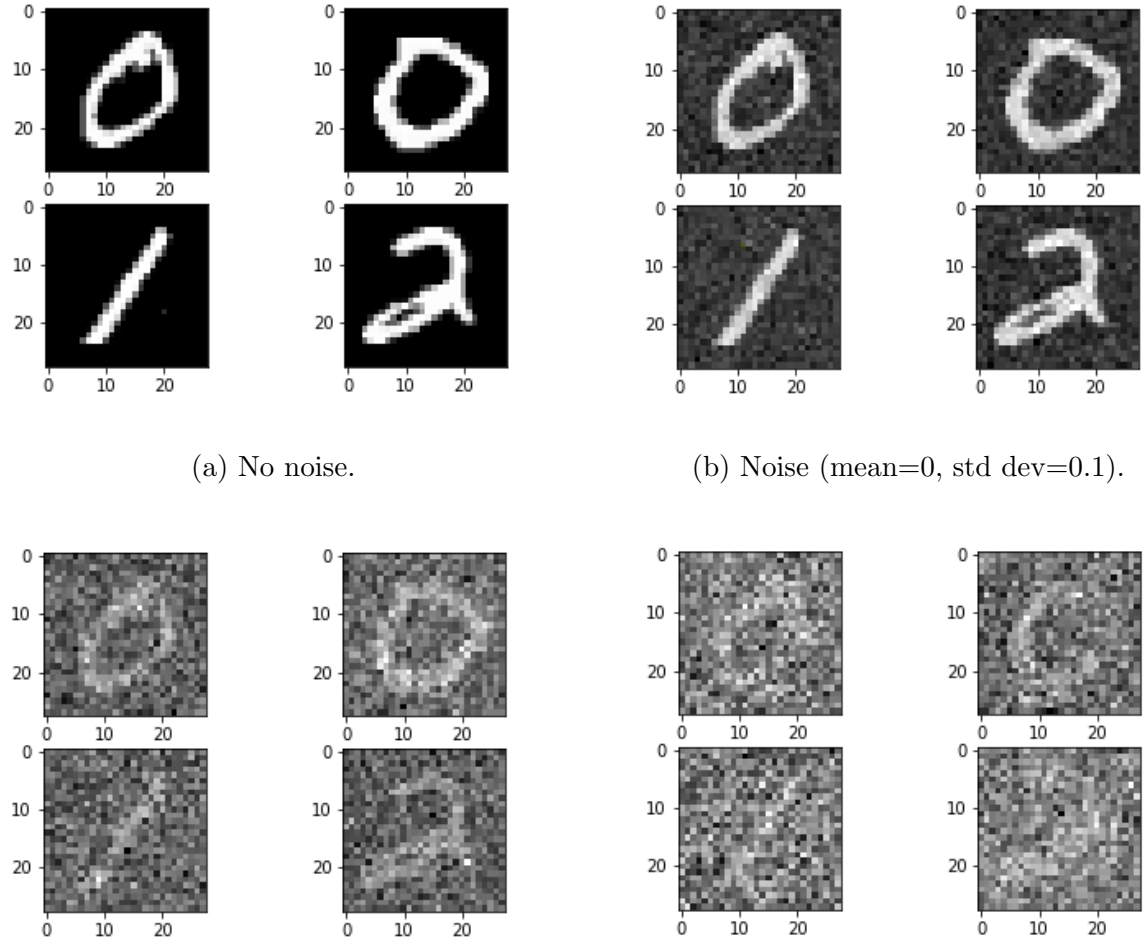
In his MIT lecture [39], Krotov hypthesizes that because the origin of adversarial perturbation lies in the speckled-ness of decision boundaries, and since the unsupervised "biological" learning algorithm results in less speckled feature detectors (Section 4.2.3), intuition suggests that the KH model may be more robust against both noisy input and adversarial attacks than the classic NN model [39]. These experimental results for handling noisy input indicate otherwise. At the same time, robustness against adversarial perturbations is recommended to be further investigated (Section 5.2).

	Classic NN	KH Model
No Noise	98.52	97.39
Noise (mean=0, std dev=0.1)	98.52	96.87
Noise (mean=0, std dev=0.5)	96.60	92.86
Noise (mean=0, std dev=1)	87.74	73.13

Table 4.1: Generalization accuracy on hold-out test sets for MNIST.

	Classic NN	KH Model
No Noise	55.15	48.62
Noise (mean=0, std dev=0.1)	54.84	48.18
Noise (mean=0, std dev=0.5)	51.18	35.70
Noise (mean=0, std dev=1)	43.50	21.07

Table 4.2: Generalization accuracy on hold-out test sets for CIFAR-10.



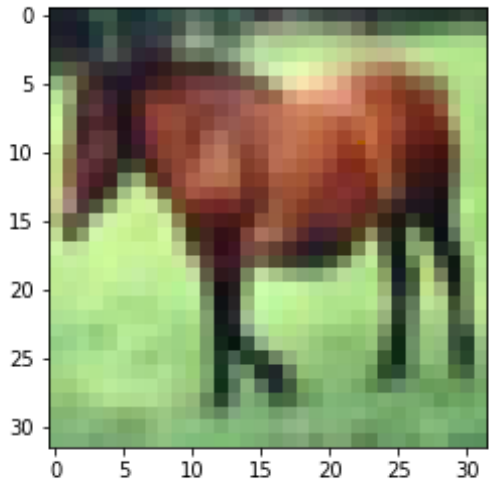
(a) No noise.

(b) Noise (mean=0, std dev=0.1).

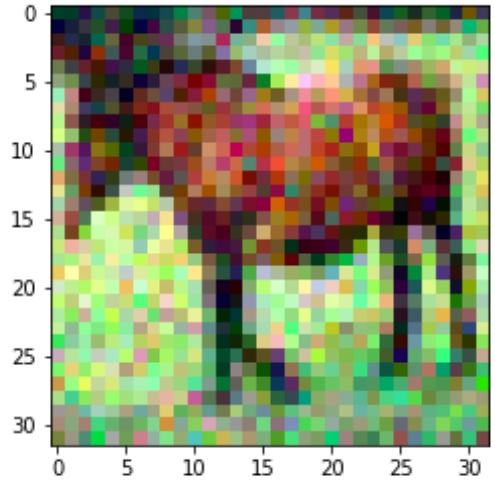
(c) Noise (mean=0, std dev=0.5).

(d) Noise (mean=0, std dev=1).

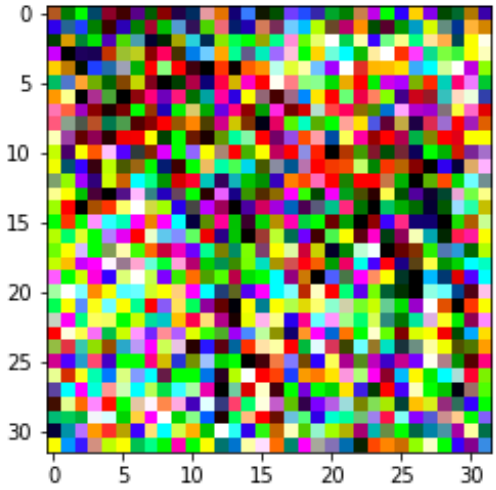
Figure 4.7: MNIST training data images (28x28 pixels) injected with noise from gaussian distributions.



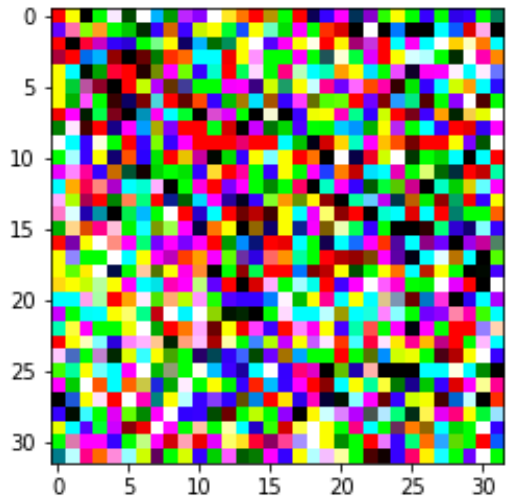
(a) No noise.



(b) Noise (mean=0, std dev=0.1).

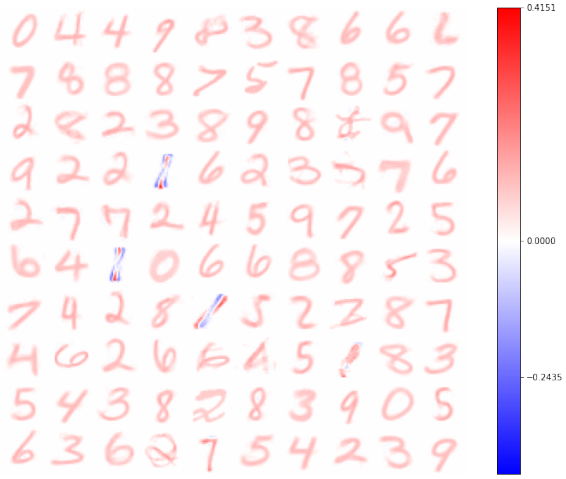


(c) Noise (mean=0, std dev=0.5).

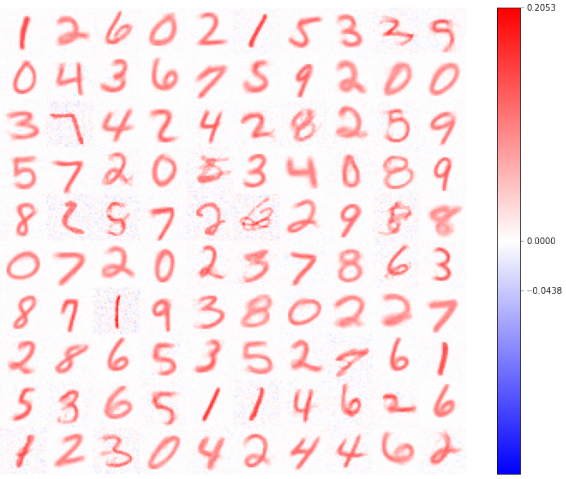


(d) Noise (mean=0, std dev=1).

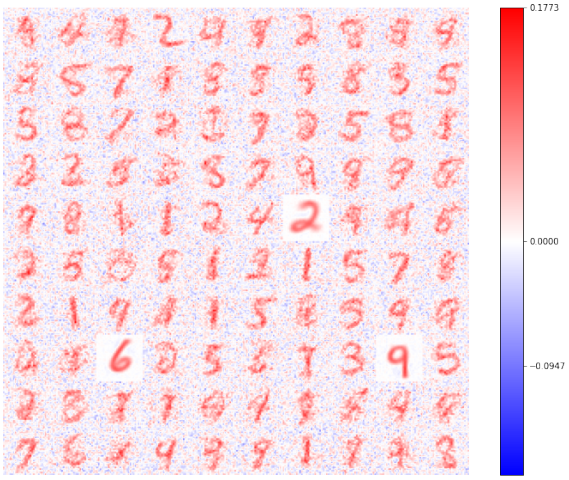
Figure 4.8: CIFAR-10 training data image (32x32x3 pixels) injected with noise from gaussian distributions.



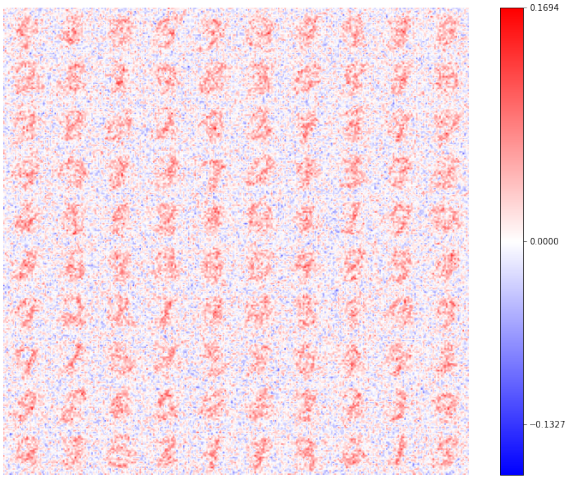
(a) No noise.



(b) Noise (mean=0, std dev=0.1).

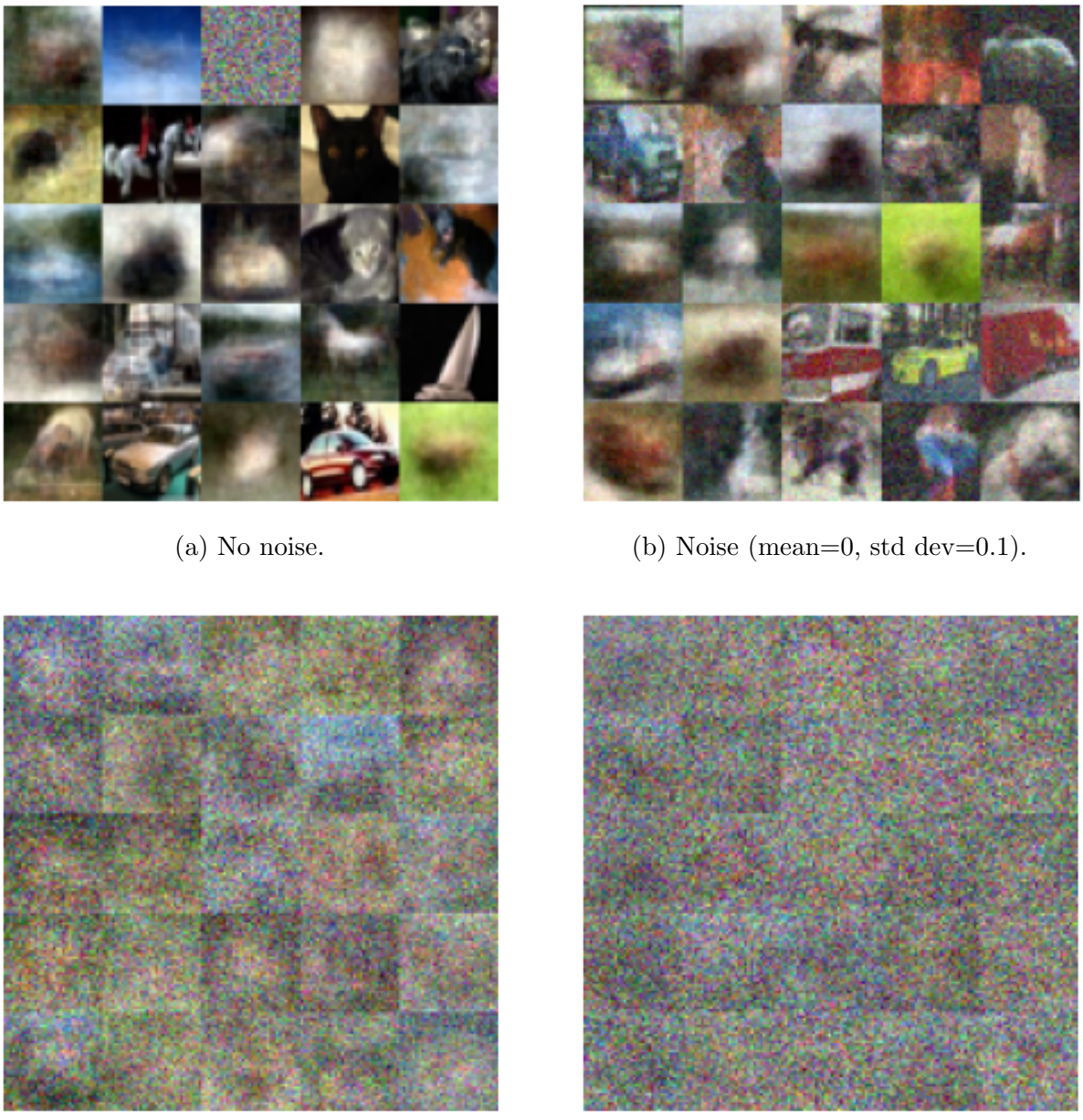


(c) Noise (mean=0, std dev=0.5).



(d) Noise (mean=0, std dev=1).

Figure 4.9: A hundred randomly selected feature detectors out of 2000 learned by the KH model for MNIST classification with noise.



(a) No noise.

(b) Noise (mean=0, std dev=0.1).

(c) Noise (mean=0, std dev=0.5).

(d) Noise (mean=0, std dev=1).

Figure 4.10: 25 randomly selected feature detectors out of 2000 learned by the KH model for CIFAR-10 classification with noise.

Chapter 5

Conclusion

This chapter briefly discusses knowledge learned from the experiments and the contribution of the work presented in this thesis. This discussion is followed by a description of the possible directions of future research.

5.1 Summary and Reflections

"Biologically plausible" learning algorithms are alternatives to the biologically unfeasible end-to-end backpropagation learning algorithm. There is a large body of research dedicated to designing such algorithms [16; 20; 21; 59; 60; 61]. Krotov and Hopfield propose one such algorithm incorporating concepts of Hebbian synaptic plasticity [3] which they have applied for various classification tasks [2; 4].

The main aim of this project is to investigate if a neural network trained with Krotov and Hopfield's biologically plausible learning algorithm [2] (KH model) can overcome limitations of modern ANN's trained with end-to-end backpropagation (Section 2.1.3). This work specifically focuses on the KH model [2] rather than other models because of the simplicity of its network architecture.

5.1.1 Achievements

I implemented a PyTorch version of the KH Model [2] and evaluated it's generalization performance on MNIST and CIFAR-10 classification. This KH model achieved generalization accuracy of 97.39% on MNIST compared to 98.52% with the classic NN model. Moreover, this KH model achieved generalization accuracy of 48.62% on CIFAR-10 compared to 55.15% with the classic NN model. Moreover, my KH model implementation learns feature detectors (Section 4.2.3) very similar to what the KH model from the original paper learns [2].

The results of the data scarcity experiments (Section 4.3.1) indicate that the KH model performs better (using generalization accuracy as a metric) than the classic NN model under circumstances of limited labelled training data. One potential reason that can explain this improved performance is that the KH model is able to generate an effective latent representation of the input images (Section 4.2.3) via its unsupervised component (Section 3.2.2). This unsupervised mechanism results in a nice set of initial weights that serve as feature detectors (Section 4.2.3) which are used by the supervised process in the KH model. Hence, this unsupervised process may give the KH model an advantage to provide a better generalization performance than the classic NN model when working with a restricted amount of labelled training data.

The results of the network robustness experiments (Section 4.3.2) indicate that the KH model is not as robust to random noise as the classic NN model. Specifically, even small random perturbations in the input cause a noticeable degradation in generalization performance of the KH Model. The robustness experiments run are very basic and more comprehensive analysis in this domain is recommended to be performed (Section 5.2).

5.1.2 Drawbacks and Limitations

The first drawback in this work is that my implementation of the KH model results in sub-optimal performance compared to what Krotov and Hopfield achieved in their paper [2] when all labelled training data are available. This gap is present for both MNIST and CIFAR-10 classification but is particularly visible with CIFAR-10. Multiple experiments using cross validation with all combinations of hyperparameters were attempted but could not achieve further improvements in the generalization performance for either dataset.

In spite of this difference in performance, I still obtained useful results for the data scarcity experiments. However, because the performance of bio NN's is not better than classic NN's when all the data is used, future work would require that performance to be improved.

A second limitation is the short time frame available to work on this project. Ten weeks worth of work is presented in this report. While progress has been made especially towards the first two goals outlined at the beginning of the project (Section 1.1), this time period was not adequate enough to setup and run a comprehensive set of tests pertaining to evaluating the stability and robustness of the KH model. I ran a set of experiments which introduced random Gaussian noise into input into the KH model (Section 4.3.2), but this base ought to be built upon in future work.

5.2 Directions for Future Research

Based on the drawbacks of the current work presented in this report, future work in this domain ought to be directed to address these limitations.

The first major improvement would be to overcome the discrepancy in generalization performance between my implementation of the KH model and the original implementation in TensorFlow [2]. The key may lie in using more appropriate hyperparameters for the unsupervised and supervised phases, or in carefully evaluating the implementation for any subtle issues. However, additional deep analysis needs to be performed to identify a clear scope for improvement. Moreover, contacting Krotov, one of the original authors of the algorithm [2], may be a potentially useful resource as he has kindly provided some help on work presented in this report.

Another limitation of the present work is that all of the experiments were performed on MNIST and CIFAR-10 datasets. Another major next step would be to apply these experiments to more complicated datasets like ImageNet. This progression would be similar to how Krotov and Hopfield build on their first paper in this subject [2], and then go on to evaluate the semi-supervised learning algorithm performance on the ImageNet dataset in their subsequent paper [4]. Moreover, investigating performance with a more complex dataset is appealing because of the question raised when analyzing my experimental results: "Do more complicated datasets result in the KH model being even more effective at classification when labelled data is limited?" (Section 4.3.1).

In a similar vein to the previous point, it is not clear what kind of improvements will result (if any) when the semi-supervised algorithm is applied iteratively in deeper architectures. Krotov et al. [4] evaluate performance using CNN's but not on deeper architectures which would be an interesting avenue to explore.

Another direction of future work would be to comprehensively explore the robustness of the KH model against noisy input and adversarial attacks. The origin of adversarial perturbation lies in the speckled-ness of decision boundaries [39]. Since the unsupervised "biological" learning algorithm results in less speckled feature detectors (Section 4.2.3), intuition suggests that the KH model may be more robust against both noisy input and adversarial attacks than the classic NN model [39]. The results from experiments (Section 4.3.2) indicate otherwise; however, to test this hypothesis more thoroughly, a series of comprehensive tests ought to be performed that establish rigorous benchmarks for image classifier robustness as established by Hendrycks and Dietterich [6].

Bibliography

[1] Werbos PJ. The Roots of Backpropagation : From Ordered Derivatives to Neural Networks and Political Forecasting. New York: John Wiley Sons; 1994.

[2] Krotov D, Hopfield JJ. Unsupervised learning by competing hidden units. Proceedings of the National Academy of Sciences of the United States of America. 2019;116(16):7723–7731.

[3] Hebb DO. The organization of behavior: A neuropsychological theory. New York: Wiley; 1949.

[4] Grinberg L, Hopfield J, Krotov D. Local Unsupervised Learning for Image Analysis. 2019;p. 1–11. Available from: http://arxiv.org/abs/1908.08993.

[5] Amini A, Soleimany A. MIT Intro to Deep Learning Course 6S191; 2020. Retrieved from: http://introtodeeplearning.com.

[6] Hendrycks D, Dietterich T. Benchmarking Neural Network Robustness to Common Corruptions and Perturbations. Proceedings of the International Conference on Learning Representations. 2019;.

[7] Rumelhart, David E , Hinton, Geoffrey E & Williams RJ. Learning representations by back-propagating errors. Nature. 1986;(323):533–536.

[8] Nokland A. Direct feedback alignment provides learning in deep neural networks. Advances in Neural Information Processing Systems. 2016;.

[9] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436–444.

[10] Rosenblatt F. The Perceptron - A Perceiving and Recognizing Automaton. Buffalo, NY, USA: Cornell Aeronautical Laboratory; 1957.

[11] Minsky M, Papert SA. Perceptrons. The MIT Press; 1969.

[12] Werbos P. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Harvard University. Cambridge, MA; 1974.

[13] Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. Nature. 2016 Jan;529(7587):484–489.

[14] Silver D, Schrittwieser J, Simonyan K, Antonoglou I, Huang A, Guez A, et al. Mastering the game of Go without human knowledge. Nature. 2017 Oct;550:354–. Available from: http://dx.doi.org/10.1038/nature24270.

[15] Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. Nature. 2019;575(7782):350–354. Available from: https://doi.org/10.1038/s41586-019-1724-z.

[16] Whittington JCR, Bogacz R. Theories of Error Back-Propagation in the Brain. Trends in Cognitive Sciences. 2019;23(3):235–250. Available from: https://doi.org/10.1016/j.tics.2018.12.005.

[17] Rao RPN, Fairhall A. Computational Neuroscience course; 2020. Retrieved from Coursera: https://www.coursera.org/learn/computational-neuroscience.

[18] Ravichandran NB, Lansner A, Herman P. Brain-like approaches to unsupervised learning of hidden representations – a comparative study. 2020;800999:1–12. Available from: http://arxiv.org/abs/2005.03476.

[19] Bartunov S, Santoro A, Hinton GE, Richards BA, Marris L, Lillicrap TP. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. Advances in Neural Information Processing Systems. 2018;2018-Decem(Nips):9368–9378.

[20] Lillicrap TP, Cownden D, Tweed DB, Akerman CJ. Random synaptic feedback weights support error backpropagation for deep learning. Nature Communications. 2016;7:1–10. Available from: http://dx.doi.org/10.1038/ncomms13276.

[21] Lee DH, Zhang S, Fischer A, Bengio Y. Difference target propagation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). 2015;9284:498–515.

[22] Illing B, Gerstner W, Brea J. Biologically plausible deep learning — But how far can we go with shallow networks? Neural Networks. 2019;118:90–101. Available from: https://doi.org/10.1016/j.neunet.2019.06.001.

[23] Tapia G. PyTorch Implementation of “Unsupervised learning by competing hidden units” MNIST classifier; 2019. Retrieved from: https://picnet.com.au/blog/pytorch-

implementation-of-unsupervised-learning-by-competing-hidden-units-mnist-classifier/.

[24] Goodfellow I, Bengio Y, Courville A. Deep Learning. The MIT Press; 2017.

[25] Rumelhart DE, Zipser D. Feature discovery by competitive learning. Cognitive Science. 1985;9(1):75–112.

[26] Recht B, Roelofs R, Schmidt L, Shankar V. Do CIFAR-10 Classifiers Generalize to CIFAR-10? 2018;p. 1–25. Available from: http://arxiv.org/abs/1806.00451.

[27] Azulay A, Weiss Y. Why do deep convolutional networks generalize so poorly to small image transformations? Journal of Machine Learning Research. 2019;20:1–25.

[28] Krizhevsky A, Sutskever I, Hinton GE. ImageNet Classification with Deep Convolutional Neural Networks Alex. 2012;.

[29] He K, Zhang X, Ren S, Sun J. Deep residual learning for image recognition. 2015;.

[30] Xie S, Girshick R, Dollár P, Tu Z, He K. Aggregated residual transformations for deep neural networks. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017. 2016;.

[31] Szegedy C, Zaremba W, Sutskever I, Bruna J, Erhan D, Goodfellow I, et al. Intriguing properties of neural networks. 2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings. 2014;p. 1–10.

[32] Carlini N, Wagner D. Adversarial examples are not easily detected: Bypassing ten detection methods. AISec 2017 - Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, co-located with CCS 2017. 2017;p. 3–14.

[33] Carlini N, Wagner D. Defensive Distillation is Not Robust to Adversarial Examples. 2016;0:1–3. Available from: http://arxiv.org/abs/1607.04311.

[34] Hendrycks D, Mazeika M, Dietterich T. Deep anomaly detection with outlier exposure. 7th International Conference on Learning Representations, ICLR 2019. 2019;p. 1–18.

[35] Hendrycks D, Gimpel K. A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks. 2017;.

[36] Liu S, Garrepalli R, Dietterich TG, Fern A, Hendrycks D. Open category detection with PAC guarantees. 35th International Conference on Machine Learning, ICML 2018. 2018;7:4985–5016.

[37] Steinhardt J, Koh PW, Liang P. Certified defenses for data poisoning attacks. Advances in Neural Information Processing Systems. 2017;2017-December(i):3518–3530.

[38] Hendrycks D, Mazeika M, Wilson D. Using trusted data to train deep networks on labels corrupted by severe noise DataPoisoning. 2018;Available from: http://papers.nips.cc/paper/8246-using-trusted-data-to-train-deep-networks-on-labels-corrupted-by-severe-noise.

[39] Krotov D. MIT 6.S191 (2019): Biologically Inspired Neural Networks (IBM); 2019. Retrieved from https://www.youtube.com/watch?v=4lY-oAY0aQU.

[40] Oja E. Simplified neuron model as a principal component analyzer. Journal of Mathematical Biology. 1982;15(3):267–273.

[41] Linsker R. From basic network principles to neural architecture: Emergence of spatial-opponent cells. Proceedings of the National Academy of Sciences of the United States of America. 1986;83(19):7508–7512.

[42] Linsker R. From basic network principles to neural architecture: Emergence of orientation-selective cells. Proceedings of the National Academy of Sciences of the United States of America. 1986;83(21):8390–8394.

[43] Linsker R. From basic network principles to neural architecture: Emergence of orientation columns. Proceedings of the National Academy of Sciences of the United States of America. 1986;83(22):8779–8783.

[44] Carpenter GA, Grossberg S. A massively parallel architecture for a self-organizing neural pattern recognition machine. Computer Vision, Graphics and Image Processing. 1987;37(1):54–115.

[45] Pehlevan C, Hu T, Chklovskii DB. A Hebbian/anti-Hebbian neural network for linear subspace learning: A derivation frommultidimensional scaling of streaming data. Neural Computation. 2015;27(7):1461–1495.

[46] Von der Malsburg C. Self-organization of orientation sensitive cells in the striate cortex. Kybernetik. 1973;14(2):85–100.

[47] Kohonen T. The Self-Organizing Map. Proceeding of the IEEE. 1990;78(9):1464–1480. Available from: https://ieeexplore.ieee.org/document/58325.

[48] Foldiak P. Forming sparse representation by local anti-Hebbian learning. Biological Cybernetics. 1990;170:165–170. Available from: http://www.rctn.org/vs265/foldiak90.pdf.

[49] Seung HS, Zung J. A correlation game for unsupervised learning yields computational interpretations of Hebbian excitation, anti-Hebbian inhibition, and synapse elimination. 2017;p. 1–16. Available from: http://arxiv.org/abs/1704.00646.

[50] Bengio Y, Courville A, Vincent P. Representation learning: A review and new perspectives. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2013;35(8):1798–1828.

[51] Chakravarthy A. Visualizing Intermediate Activations of a CNN trained on the MNIST Dataset; 2019. Retrieved from https://towardsdatascience.com/visualizing-intermediate-activations-of-a-cnn-trained-on-the-mnist-dataset-2c34426416c8.

[52] Dr Mönchmeyer R. A simple CNN for the MNIST dataset – IV – Visualizing the output of convolutional layers and maps; 2020. Retrieved from https://linux-blog.anracom.com/2020/06/28/a-simple-cnn-for-the-mnist-dataset-iv-visualizing-the-output-of-convolutional-layers-and-maps/.

[53] Krotov D. Biological_Learning; 2019. Retrieved from GitHub: https://github.com/DimaKrotov/Biological_Learning.

[54] PyTorch Tutorials; 2019. Retrieved from GitHub: https://github.com/python-engineer/pytorchTutorial.

[55] Tapia G. PyTorch Implementation of MNIST Bio Classifier; 2019. Retrieved from GitHub: https://github.com/gatapia/unsupervised_bio_classifier.

[56] Apparaju A. PyTorch Implementation of Krotov and Hopfield's Bio Classifier applied to MNIST and CIFAR-10; 2020. Retrieved from GitHub: https://github.com/votaju/apparaju_biological_learning.

[57] LeCun Y, Cortes C, Burges CJC. THE MNIST DATABASE of handwritten digits; 1999. Retrieved from http://yann.lecun.com/exdb/mnist/.

[58] Krizhevsky A. Learning Multiple Layers of Features from Tiny Images. MIT and NYU; 2009.

[59] Lillicrap TP, Cownden D, Tweed DB, Akerman CJ. Random feedback weights support learning in deep neural networks. 2014;p. 1–27. Available from: http://arxiv.org/abs/1411.0247.

[60] Hinton G. Can sensory cortex do backpropagation?; 2016. Retrieved from: https://www.youtube.com/watch?v=cBLk5baHbZ8.

[61] Bengio Y. Deep learning and backprop in the brain; 2016. Retrieved from: https://www.youtube.com/watch?v=FhRW77rZUS8.