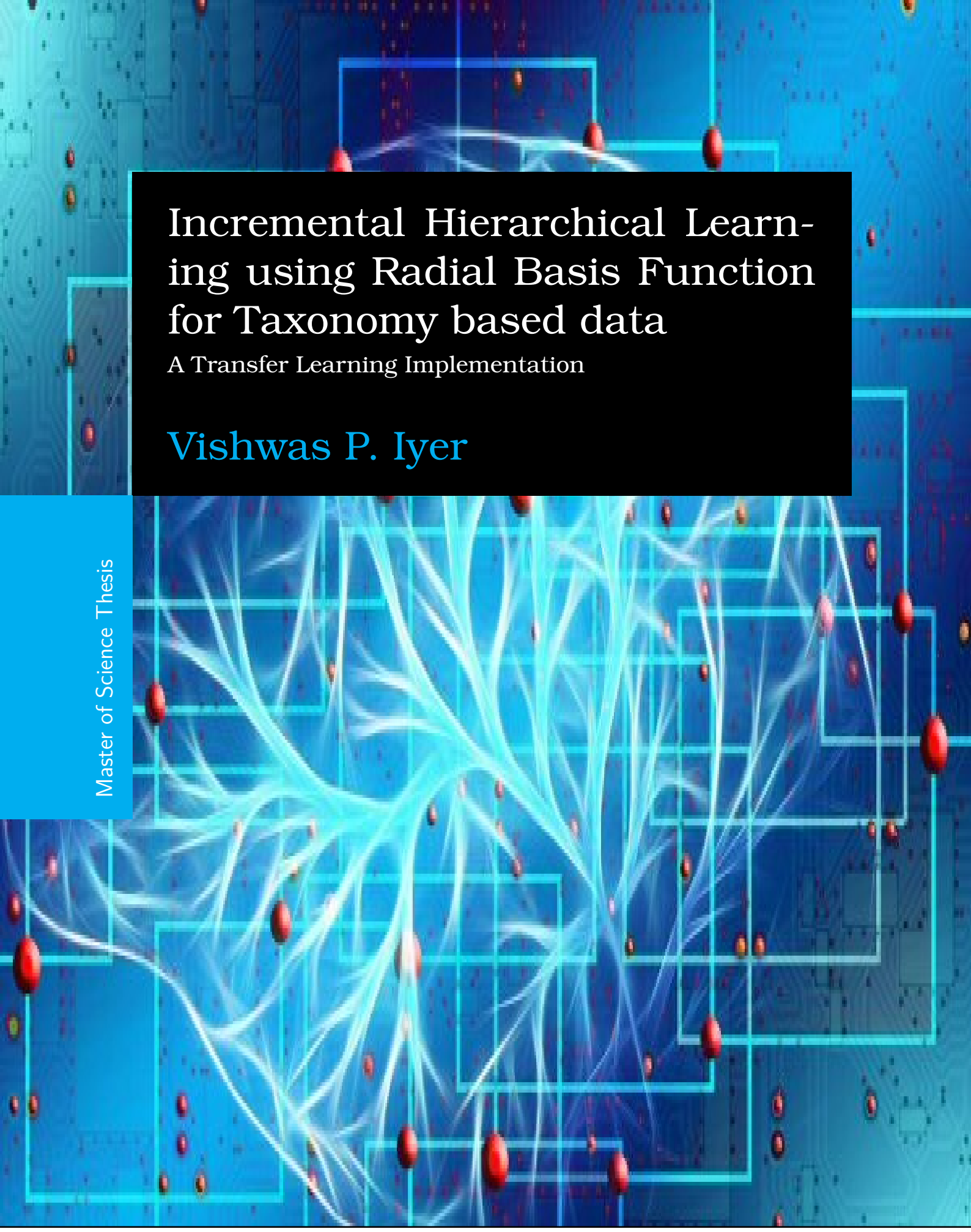

Incremental Hierarchical Learning using Radial Basis Function for Taxonomy based data

A Transfer Learning Implementation

Vishwas P. Iyer

Master of Science Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

Incremental Hierarchical Learning using Radial Basis Function for Taxonomy based data

A Transfer Learning Implementation

Master of Science Thesis

For the degree of Master of Science in Systems and Control at Delft University of Technology

Vishwas P. Iyer

January 19, 2021

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Delft University of Technology
Department of
Delft Center for Systems and Control (dcsc)

The undersigned hereby certify that they have read and recommend to the Faculty of Mechanical, Maritime and Materials Engineering (3mE) for acceptance a thesis entitled

Incremental Hierarchical Learning using Radial Basis Function for Taxonomy based data

by

Vishwas P. Iyer

in partial fulfillment of the requirements for the degree of

Master of Science Systems and Control

Dated: January 19, 2021

Supervisor(s):

dr.ir. J. Sijs

Reader(s):

Prof.dr.ir. B. De Schutter

dr. C. Smith

Abstract

Significant work has been done in the field of computer vision focusing on learning and clustering methods. The use of improved learning methods has paved a way forward for researches to explore various theories to improve existing methods. One among various learning methods is *Hierarchical learning* which has showed impressive benefits and performance over traditional sequential learning approaches. In general, machine learning models require a lot of data for every new scenario which is not always possible and if so, is very expensive. Transfer learning, which focuses on transferring knowledge across trained machine learning models, is a promising machine learning methodology for solving the above problem.

In this thesis, we propose an end-to-end neural network architecture on the NM500 neuromorphic chip using an incremental hierarchical learning approach. We first design a hierarchical representation of a taxonomy, develop a batch of pre-classifiers and use their output to construct a custom feature vector that is the input to the front-end network which learns the taxonomy. In other words, the taxonomy is embedded in the clustering method and not trained by a backpropagation algorithm. The custom feature vector has been structured to accurately incorporate the taxonomy based on the *Manhattan distance norm*. The structure has been proven mathematically and validated using experiments.

A Radial Basis Function (RBF) is used for learning and a combination of RBF and K-Nearest Neighbors (KNN) for classification. The applicability of the proposed framework has been demonstrated on a road sign classification problem which is represented as a taxonomy. The ability of the framework to incrementally learn new categories and update the taxonomy online has also been shown. Lastly, we show a case of transfer learning where the entire back-end networks is used as a starting point to learn new features without significantly forgetting prior knowledge. This transfer learning framework showed comparable performance to the standard learning method in terms of accuracy while using significantly less labelled data. This work paves a way forward for researchers to develop transfer learning frameworks and more importantly explore neuromorphic hardware for machine learning tasks.

Table of Contents

Acknowledgements **xiii**

1 Introduction **1**
1-1 Research motivations . . . 1
1-2 Main contributions . . . 3
1-2-1 Minor Contributions . . . 4
1-3 Thesis outline . . . 4

2 Background & Motivations **5**
2-1 Classification Problem . . . 5
2-1-1 Hierarchical Classification . . . 6
2-2 Road Sign Classification Problem . . . 8
2-2-1 Taxonomy . . . 9
2-3 Transfer Learning . . . 11
2-3-1 Fine-tuning . . . 11
2-4 Summary . . . 12

3 Neuromorphic Hardware **13**
3-1 NeuroShield . . . 13
3-2 Building the decision space . . . 13
3-2-1 Active Influence Field (AIF) . . . 14
3-2-2 Maximum Influence Field (MAXIF) . . . 14
3-2-3 Minimum Influence Field (MINIF) . . . 15
3-2-4 Global Context Register (GCR) . . . 15
3-2-5 Distance Norm . . . 16
3-3 Classification . . . 17
3-3-1 RBF . . . 17
3-3-2 KNN . . . 17
3-4 Summary . . . 19

4 End-to-End Framework **21**
4-1 Network Structure . . . 21
4-2 Summary . . . 24

5 Back-end Networks **25**
5-1 Extraction . . . 25
5-2 Image Pre-processing . . . 26
5-2-1 Shape Features Pre-processing . . . 26
5-2-2 Colour Features Pre-processing . . . 27
5-2-3 Descriptive Features Pre-processing . . . 27
5-2-4 Final Features Pre-processing . . . 29
5-3 Back-end Networks . . . 29
5-3-1 Learning . . . 29
5-3-2 Classification . . . 30
5-4 Catastrophic Forgetting . . . 33
5-5 Summary . . . 35

6 Front-end Network - Taxonomy **37**
6-1 Proposed Clustering Approach . . . 37
6-2 System Specific parameters and constraints . . . 45
6-3 Front-End Network . . . 48
6-3-1 Learning . . . 48
6-3-2 Classification . . . 49
6-4 Summary . . . 51

7 Results **53**
7-1 Evaluation Metrics . . . 53
7-2 Back-end Networks . . . 54
7-2-1 Shape Feature Extraction Network . . . 54
7-2-2 Colour Feature Extraction Network . . . 55
7-2-3 Descriptive Feature Extraction Network . . . 56
7-2-4 Final Level Feature Extraction Network . . . 58
7-2-5 Front-end Network . . . 59
7-3 Online Incremental Learning . . . 63
7-4 Transfer Learning . . . 64

8 Conclusion **69**
8-1 Summary . . . 69
8-2 Answers to Research Questions . . . 70
8-3 Future Work . . . 71

A Proofs **73**
A-1 Proof of infinite geometric progression . . . 73

Bibliography **75**

Glossary **79**

List of Acronyms 79

List of Figures

2-1 An example of a Directed Acyclic Graph (DAG) structure (left) and a tree structure (right) . . . 6
2-2 Local Classifier per node . . . 7
2-3 Local Classifier per parent node . . . 7
2-4 Local Classifier per level . . . 8
2-5 Global Classifier . . . 8
2-6 A possible representation of the road sign classification problem as a taxonomy. Each level of the taxonomy represents a feature of the road sign. . . . 10

3-1 Effects of MAXIF on the decision space, Adapted from [1]. The coloured circles represent one committed neuron for the respective category and the diamond shapes represent their AIF. . . . 14
3-2 Effects of MINIF on the decision space, Adapted from [1].The coloured circles represent one committed neuron for the respective category and the diamond shapes represent their AIF. . . . 15
3-3 Decision space mapped using RBF classifier. Gray area represents unmapped decision space. The colours represent the category associated with neurons, diamond shapes represent the AIF of the committed neurons and circles represent committed neurons. . . . 16
3-4 An example showing two different input feature vectors in the modelled decision space. Yellow cross represents an input feature vector which does not fall into the influence field of any neurons and black cross represents an input feature vector which falls under the influence filed of a neuron with category red. . . . 18
3-5 Decision space mapped using KNN classifier. Entire decision space is mapped. Colours represent the category associated with the committed neurons, shapes represent the AIF of the committed neurons and circles represent committed neurons. 18
3-6 An example showing two different input feature vectors in a decision space interpreted using a KNN mode. Colours (except yellow and black crosses) represent the category associated with the committed neurons, shapes represent the AIF of the committed neurons and circles represent committed neurons. . . . 18

4-1 General End-to-end Network structure . . . 22

4-2 Online Incremental Learning 23
4-3 Transfer Learning - Fine tuning 24

5-1 Example of Erosion, Dilation and Morphological Gradient 27
5-2 Image segmentation using colour spaces 28
5-3 Image pre-processing pipeline 28
5-4 Final feature pre-processing 29
5-5 An example of a decision space with two neurons. Red and blue colours indicate two different categories for each neuron and their influence fields are indicated by the diamond shapes. The circles cross marks are committed neurons and the green cross mark is the input vector 32
5-6 Example of an unknown input sample in the decision space 33
5-7 Classification Logic 34

6-1 A general taxonomy structure indicating different levels and taxonomical features. The light grey circles represent classes and each dotted rectangle represents a separate level of the taxonomy. The dark grey circle with a label 'Type' is the overall type of the taxonomy (for example, road signs). The arrows represent taxonomical features. 38
6-2 The expected output of our proposed distance based clustering approach. The two dotted cloud-like structures represent clusters of 2 taxonomical features at level l, dotted circles represent clusters of taxonomical features at level $(l-1)$ and the solid-lined ellipses represent clusters of taxonomical features at level $(l-2)$. Note that the dotted-lined figures represent virtual clusters while the solid-lined ellipses represent the final actual clusters. 39
6-3 The expected output of n-dimensional feature space for our proposed distance based clustering approach for the case 1 and case 2. The two solid-lined cloud-like structures represent the case when one class is recognised and all others are not recognized while the two dotted cloud-like structures represent the case when one of the classification outcomes is uncertain and the rest are not recognised. . . . 42

6-4 The expected output of the network for the test input vector (red dot). Level l has the highest influence on clustering and hence the uncertainty on level $(l-1)$ does not affect the decision at level l. The uncertainty at level $(l-1)$ is observable as the sample neither lies within Feature A cluster nor within Feature B cluster. The sample is however placed closer to Feature A than B because of an uncertain outcome at level $l-1$ and also because Feature R is recognised at level $(l-2)$. But since the influence of level $(l-1)$ on clustering is more than level $(l-2)$, the sample lies outside class A in spite of being recognised as Feature R. 43
6-5 The expected output of the network for the input vector. Level l has the highest influence on clustering and hence the recognition outcome on level $(l-1)$ and level $(l-2)$ does not affect the decision at level l. The sample is placed almost at the border of Feature 2 cluster because the lower 2 classes are closer to Feature B and Feature S clusters which are part of Feature 1 cluster. But since the influence of level (l) on clustering is more than level $(l-1)$ and $(l-2)$ combined, the sample lies within Feature 2 cluster in spite of being recognised as Feature B and Feature S respectively at lower levels. 44
6-6 The test sample is incorrectly classified here. This shows the output of the network when Equation $(6\text{-}13)$ is violated. On violation of equation $(6\text{-}13)$, level l will have a lower influence on clustering than level $(l-1)$ and $(l-2)$ combined. Hence, in spite of being recognised as Feature 2, the sample is placed outside the cluster. This is because, the sum of distances of the sample from Feature 2 is more than its distance from Feature B and Feature S. 44

6-7 The updated taxonomy as per the classification outcome shown in Figure 6-5. New nodes Feature B and Feature S have been introduced at level $l-1$ and $l-2$ respectively. 44
6-8 The updated decision space as per the classification outcome shown in Figure 6-5. New nodes Feature B and Feature S have been introduced at level $l-1$ and $l-2$ respectively. 44
6-9 A decision space with no threshold on MAXIF. 49
6-10 A decision space with a threshold on MAXIF. 49

7-1 Learning Curve for the shape feature extraction network 55
7-2 Neuron Distribution Histogram for shape feature extraction network 55
7-3 Confusion Matrix for the shape feature extraction network. 55
7-4 Learning Curve for the colour feature extraction network 56
7-5 Neuron Distribution Histogram for colour feature extraction network 56
7-6 Confusion Matrix for the colour feature extraction network 56
7-7 Learning Curve for the descriptive feature extraction network 57
7-8 Neuron Distribution Histogram for descriptive feature extraction network 57
7-9 Confusion Matrix for the descriptive feature extraction network 57
7-10 An example of a decision space with too many overlaps/uncertainty. The colours represent distinct features, circles represent committed neurons, the coloured shapes represent the influence fields of each neuron and the yellow cross is an input sample. 58
7-11 Learning Curve for the final level feature extraction network 59
7-12 Neuron Distribution Histogram for final feature extraction network 59
7-13 Confusion Matrix for the final level feature extraction network 59
7-14 Decision Space 1 61
7-15 Decision Space 2 61
7-16 Decision space modelling using an uncertain sample 9 61
7-17 Decision space before the incremental update 63
7-18 Decision space after incrementally updating the taxonomy 63
7-19 Updated decision space with new categories 9,10 and 11 67

List of Tables

6-1 Parameter values for different levels of the taxonomy 46

7-1 Sample Confusion Matrix 54
7-2 Categories 60
7-3 All Categories Learnt 62
7-4 6 closest neurons for sample 9 63
7-5 7 closest neurons for sample 10 63
7-6 Shape Network Performance 65
7-7 Descriptive Network Performance 65
7-8 Last Layer Network Performance 65
7-9 Network performance after transfer learning 66

Acknowledgements

First, I would like to thank my thesis supervisor, Dr. Joris Sijs for giving me an opportunity to work with you. You have always guided me in the right direction and provided constant motivation in spite of all the pitfalls throughout this journey. I am grateful to you for always encouraging me to explore new ideas while also suggesting challenging and innovative scenarios. Last but not the least, thank you for enlightening me about NeuroShield and Transfer Learning, which laid a foundation for this thesis.

Of course, I would like to thank my parents without whom I would have never made it here and also all the other people who always had faith in me and my choices. Mom, Dad, I will never be able to express all my gratitude to you. Even though it was hard sometimes, you always put my needs before yours, always wanted the best for me, and always let me follow my dreams. I would also like to thank one of my best friends Shreyash Palande, who has been a constant support and source of encouragement. He always stood by me and helped me in every way he could. I can say with assurance that my journey at TU Delft would have been a 100 times more difficult and incomplete without your support.

Finally, I want to thank all the amazing friends I made during these two years at TU Delft. I will always cherish these beautiful memories and I thank everyone for being a part of this extraordinary adventure.

Delft, University of Technology
January 19, 2021

Vishwas P. Iyer

"You have to dream before your dreams can come true."
— *Dr. APJ Abdul Kalam*

Chapter 1

Introduction

This introductory chapter focuses on the motivations behind this research, posing the fundamental questions that this thesis addresses and the main contributions of this work. The chapter is then concluded with an outline of the document.

1-1 Research motivations

Classification is a machine learning technique used to predict the class of an input sample. Classification problems have gained substantial attention in the last decade and continuous efforts are being made to incorporate these methods into real life problems. They have found significant applications ranging from speech recognition to image segmentation and DNA sequence classification [2]. Traditional Machine learning methods have some limitations for certain real-world scenarios. The ideal scenario of machine learning is that there is abundant labeled training data, which have the same distribution as the test data. However, collecting sufficient training data is often expensive, time-consuming, or even unrealistic in many scenarios. Semi-supervised learning [3] can partly solve this problem by relaxing the need of mass labeled data. A semi-supervised approach only requires a limited number of labeled data, and it uses a large amount of unlabeled data to improve the learning accuracy. But in many cases, unlabeled data is also difficult to collect, making the resultant traditional models unsatisfactory.

Traditionally, most of the multi-class classification problems (i.e. problems where you want to predict the class of a sample from a set of possible classes) focus on a small number of possible predictions. For certain applications like classifying a given e-mail as Spam/Legitimate , classifying an image based on colour, classifying music genre of some song playing on the radio, etc. But if we want to be more granular, and for instance are trying to predict the artist playing the music, there are too many of them to consider at once. But at the same time, we know that there are some shared characteristics (based on voice, style of music, etc.) among them which can be grouped together and their relationships can be exploited. These

methods take into account the data taxonomy and provide information about each level of the taxonomy.

Hierarchical learning methods have proven to show noteworthy results in classification problems. Wang et al. in their work address a road sign classification problem using multiple classifiers to distinguish between features of different levels of the hierarchy [4] an accuracy of 99.52% on the German Traffic Sign Recognition Benchmark (GTSRB). Sengar et al. proposed a hierarchical classifier in which the number of coarse classes is automatically determined [5]. Their approach has an advantage of dedicated classifiers trained for classes which are more difficult to distinguish. Many of these algorithms including the work done in [5] use neural networks (more dominantly Convolutional Neural Network (CNN)s) as a platform to implement hierarchical learning algorithms. This is primarily because CNNs are known to provide modular architecture which give them a significant advantage over other Artificial Neural Networks (ANN)s. Not much work has been done in this domain using other ANNs like Radial Basis Function (RBF) networks, Spiking Neural Network (SNN)s, etc. These methods have shown performance equivalent to CNNs in many classification and regression applications but there is a need explore these methods further as they have the potential to compete with state-of-the-art CNN algorithms.

Even though conventional hierarchical learning methods show impressive results, they still require a lot of data for training. Moreover, these methods can not cope with new or unseen classes of data. To overcome this problem, methods like incremental learning and transfer learning were explored in this work. Incremental learning has been used for quite some years now and there are several state-of-the-art algorithms available. Transfer learning on the other hand is not as widely explored. Both these methods aim to reduce data dependency but serve different purposes. Incremental learning aims at learning new data without forgetting the previous knowledge. This means that the network can be trained in sequence for different tasks without forgetting previously learnt tasks. While transfer learning is based on the assumption that when a network is trained for one task, it can be reused for any other similar task with minimal or no additional tuning.

With the aim to significantly reduce training time and computational costs, researchers developed neuromorphic chips with dedicated machine learning algorithms and neural network accelerators. Several neural network accelerators are available which use Field Programmable Gate Array (FPGA)s in their hardware architecture for communication and allow developing a modular neural network architecture. Most popular existing neural network hardware include CPUs or GPUs. GPUs are known to outperform CPUs in most cases especially ones with large datasets [6], [7]. GPU's are quite expensive and bulky compared to dedicated neuromorphic chips. These neuromorphic chips can outperform GPUs by reducing the learning time, recognition time while also making training more efficient and energy consumption [8]. This can act as one of the rationale for choosing neuromorphic chip over conventional neural network hardware. Recently developed neuromorphic hardware have shown that combinations of ANNs or a dedicated SNN chip can be used to develop almost any type of modular neural network architecture. State-of the-art neuromorphic chips like IBM's TrueNorth chip, Intel's Loihi chip have achieved tremendous parallelism and have outperformed CNNs. Some work done using neuromorphic accelerators include voice recognition [9], LIDAR image classification [10], face and voice recognition [11], Text image recognition [12] etc.

Most developments in classification problems using neural networks have been with respect

to CNNs. Not much work has been done in this field using other ANNs or hardware neuromorphic chips. These chips have shown tremendous performance over most existing hardware like CPUs and GPUs and can be a very suitable candidate for applications that require very quick on board decision making and classification. Blouw et al. in their work compare the performance of the Inter Loihi chip with CPU and GPU based neural networks in terms of inference speed, dynamic power consumption, and energy cost per inference [13]. Their results show that Loihi outperforms all of these alternatives on an energy cost per inference basis while maintaining equivalent inference accuracy. We believe all these advantages need to be utilised to their full potential and hence there is a need to exploit such neuromorphic chips with state-of-the-art learning methods.

In this work, we have developed an end-to-end neural network architecture on the NM500 chip using an incremental hierarchical learning approach. The output of a batch of developed pre-classifiers is used to construct a custom feature vector that is the input to the front-end network which learns the taxonomy. In other words, the taxonomy is embedded in the clustering method and not trained by a backpropagation algorithm. The proposed network can classify almost any taxonomy based data using information available at various levels of the taxonomy. It is capable of classifying unknown classes by using existing knowledge. In case when information about some levels of the taxonomy is not available, the network can use the available information to predict an approximate position of the input sample in the decision space. Moreover, it is able to incrementally update the taxonomy in cases when an unseen category of data is available. Lastly, we show a transfer learning framework where the back-end networks are used as a starting point to learn new features using minimal labelled data and learn the remainder of the dataset using labels predicted by the network. To the best of our knowledge, no work has been done on incremental learning and transfer learning using such neuromorphic chips. The framework is tested on a road sign classification problem which is represented in the form of a taxonomy. To conclude, the research motivations led to formulation of the following questions that this work addresses:

1. ***Can dedicated hardware such as NM500 neuromorphic chip show comparable or better performance than conventional hardware like CPUs and GPUs?***

2. ***Can a transfer learning framework help improve performance as compared to conventional learning methods?***

1-2 Main contributions

The main contributions of this work are detailed below:

- ***Custom feature vector structure which can represent an entire taxonomy.***

  We have developed a custom feature vector structure which can represent almost any taxonomy. This custom feature vector is partitioned into the number of levels in a taxonomy and elements of each partition represent the recognition outcome of a feature belonging to the corresponding level of the taxonomy.

- ***Introducing uncertainty into the neural network and build a decision making logic around it.***

  Instead of a binary classification, we have a third outcome which gives information about the input vector possibly belonging to a class in the form of uncertainty. The advantage of having such an uncertainty is to get additional information about an input vector in spite of no neurons firing i.e. the network not recognising the input sample.

- ***Incremental Learning Framework***

  We have shown an incremental learning implementation where new combinations of features corresponding to the same taxonomy can be learnt online incrementally without forgetting the previously learnt data. The taxonomy can thus be updated online.

- ***Transfer Learning Framework using fine-tuning***

  We have shown a transfer learning implementation which allows learning unseen classes of data. This framework is also developed with an intention of being able to learn a new but similar taxonomy using the previously learnt taxonomy.

1-2-1 Minor Contributions

This contribution is related to the NeuroShield API and does not have a direct impact on the work done in this thesis. However, we feel it is worthy of a mention.

- ***Adapting various in-built functions from the NeuroShield API to Python***

  Although the API is available in Python, various useful functions like storing the learnt knowledge, loading the saved knowledge, changing recognition modes, etc. were only written in C or encoded as a binary executable file. We have rewritten the existing C functions in python to make them more user-friendly.

1-3 Thesis outline

This document is organised as follows. Chapter 2 covers the main topics important to understand this thesis. The general classification problem and its types are discussed initially. It also addresses the road sign classification problem and finally some light is thrown on transfer learning. Chapter 3 completely focuses NeuroShield which is the neuromorphic hardware used in this thesis. In Chapter 4 we discuss the end-to-end network design followed by different learning and classification scenarios that the proposed approach must handle. In Chapter 5 we discuss in detail the pre-processing pipeline, back-end networks and the learning and classification logic for these back-end networks. In chapter 6, we discuss the custom feature vector and the front-end networks. We present our hypothesis mathematically and also prove it using induction. In chapter 7 we have validated the work done in this thesis. The performance of the network is evaluated in the general learning framework, incremental learning framework and transfer learning framework. Chapter 8 concludes the work presented in this document, and it summarises the answers to the research questions previously mentioned. Besides, we have also included some recommendations for future research in this direction.

Chapter 2

Background & Motivations

In this chapter we present a short survey on some of the most commonly used classification methods and their limitations, followed by introducing the road sign classification problem. This chapter is aimed at providing readers with a background on the addressed problem for easier and better understanding of the upcoming chapters. Section 2-1 discusses briefly the types of classification problems and their limitations. It also discusses hierarchical classification in little detail. Section 2-2 discusses the road sign classification problem, the motivations behind addressing it and introduces the proposed taxonomy. Lastly in Section 2-3, we briefly introduce the concept of transfer learning and fine-tuning.

2-1 Classification Problem

A classification problem involves predicting which class a sample belongs to. Some classifiers are binary, resulting in a yes/no decision. Others are multi-class, able to categorize an item into one of several classes. Classification is a very common use case of machine learning used to solve problems like email spam filtering, document categorization, speech recognition, image recognition, etc. Classification approaches can be categorised into the following types:

- Flat Classification: This approach is the simplest amongst all approaches and it completely ignores the structure of the data. Flat classification includes methods like binary classification, multi-class classification [14]. Basically, such classification methods only recognise leaf nodes of a decision tree.
- Hierarchical Classification: This approach uses a predefined data taxonomy to create a hierarchy of classes. It is a well structured approach which provides more details on classification [14].

Flat classification methods are relatively simpler and are also computationally less expensive. But since it does not consider the hierarchy of data (partial information), essential information is lost. Let us consider the road sign classification problem which will be addressed in this

thesis. Flat classification method will only detect the leaf node of the hierarchy. This means that, if the information about it being a prohibition sign is part of the leaf node, it will classify it as a prohibition sign or if the leaf node has information about it being a 30 km/hr speed sign, it will classify it as that. On the contrary, hierarchical classification methods are highly intuitive and use the complete hierarchy information. They will classify it as both i.e. a prohibitory sign and a 30 km/hr speed sign irrespective of whether there is a leaf node specified or not. The only major drawback of these methods is error propagation, where an error at one level of the hierarchy could affect all of the following ones.

The focus of this thesis is on hierarchical classification because this type of classification can provide partial information even if a sample is not completely classified which can be useful in critical applications such as autonomous cars.

2-1-1 Hierarchical Classification

As mentioned earlier, hierarchical classification method takes into account the data taxonomy and provides information about each level of the taxonomy. According to Freitas and de Carvalho [15], hierarchical classification methods differ by three criteria which are as follows:

Type of hierarchical structure: The structure is based on the problem and it is generally a `tree` or a `Directed Acyclic Graph (DAG)`. The major difference between them is that a node in `tree` structure can not have more than one parent while in a `DAG` structure it can.

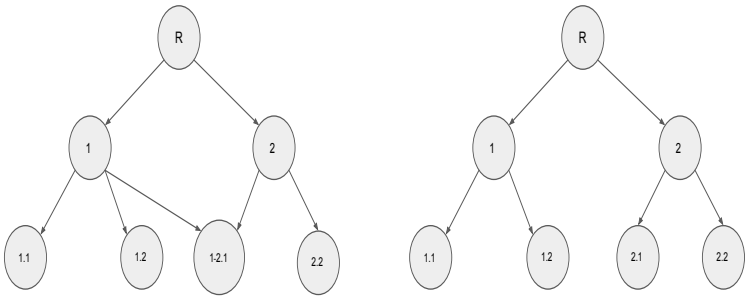

Figure 2-1: An example of a DAG structure (left) and a tree structure (right)

Depth of classification: This criterion is related to how deep in the hierarchy must the classification be performed. If classification method is implemented such that it will only classify a leaf node, it is referred to as `virtual category tree` [16] or `mandatory leaf-node prediction` [15] and if it can stop the classification at any node at any level of the hierarchy, it is referred to as `category tree` [16] or `non-mandatory leaf-node prediction` [15].

Exploration type: This criterion deals with how the hierarchical structure is explored. In literature, two approaches are used for exploration which are discussed below:

1) Local Classifier Approach or Top-Down Approach: In this approach, the hierarchy is taken into account by using a local information perspective. Based on how this local information is used to build classifiers, these approaches can be divided into 3 types:

- *Local classifier per parent node:* Each parent node in the taxonomy is trained as a multi-class classifier. This means that every node at every level of the hierarchy is a classifier except the leaf nodes. To avoid any inconsistencies, each parent node is only trained to classify it's corresponding child nodes. Hence, this algorithm is not directly applicable to DAG structure as a given class node can have multiple parent nodes. An example is shown in Figure 2-3.

- *Local classifier per node:* In this method, a binary classifier is trained for each node in the hierarchy. Hence, every node except the root node is a binary classifier. This method can be directly used in the `tree structure` and `DAG structure`. An example is shown in Figure 2-2.

- *Local classifier per level:* This approach involves training one multi-class classifier for each level of the hierarchy. The outcome of each classifier is used to predict the final classification. This method is the least computationally expensive amongst all the above methods. However, it is prone to class-membership inconsistency. Class-membership inconsistency means a conflict between the outcomes of 2 levels. However, there are methods to overcome this problem [17], [18]. An example of such a classifier is shown in Figure 2-4.

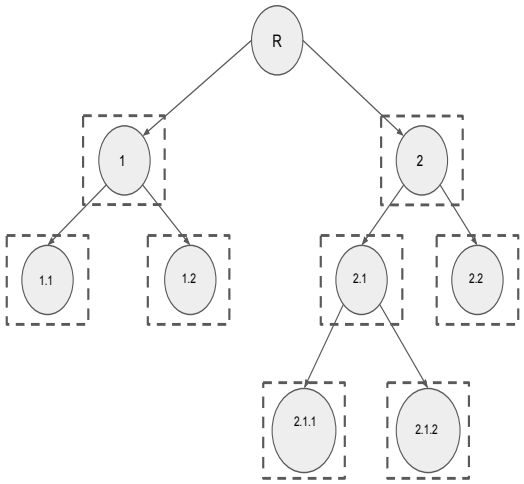

Figure 2-2: Local Classifier per node

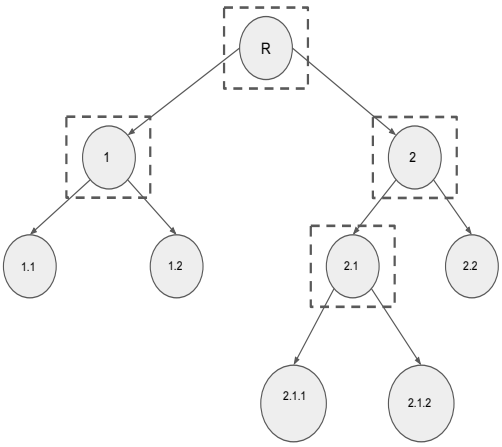

Figure 2-3: Local Classifier per parent node

2) Global Classifier Approach or Big-Bang Approach: This approach consists of learning a single global model for all classes taking into account class hierarchy as a whole. The main advantage of this approach is that there is no need to train a large number of classifiers and to deal with the inconsistency in the prediction of classes. Even though it does not incorporate every level of the hierarchy explicitly, they can be taken into account in a natural way. For example, if a road sign is classified as a *50 speed limit* sign, it automatically belongs to a prohibitory sign which thereby can be differentiated as circular in shape and red in colour. For the example shown in Figure 2-5, if the sample is classified as `2.1.1`. it automatically belongs to `2.1` and `2` as well.

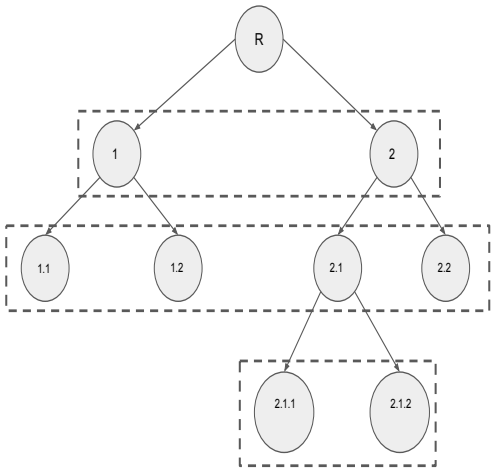

Figure 2-4: Local Classifier per level

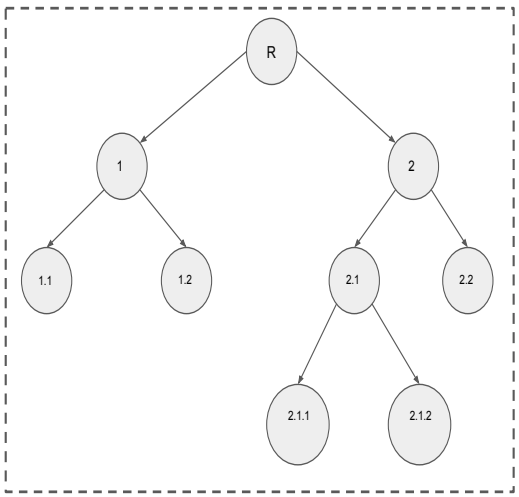

Figure 2-5: Global Classifier

Figure 2-2 through Figure 2-4 show the different exploration types based on local classifiers. The circles represent classes and each dashed rectangle encloses the classes predicted by a multi-class classifier. Figure 2-5 shows a global classifier based exploration that learns a global classification model about the whole class hierarchy. For this work, a combination of *local classifiers per level* and *global classifiers* is used which makes the network less sensitive to error propagation. The relevance to this will be clear in Chapter 4 where the network structure is explained.

2-2 Road Sign Classification Problem

Road sign classification is a highly researched field and numerous methods have been proposed over the years to tackle this problem. In this era of autonomous driving, a lot of research is being done to improve safety and reliability of these cars. Advanced Driver Assistance Systems (ADAS) is a major field in autonomous driving within which Automatic Traffic Sign Detection and Recognition (TSDR) is an important research focus [19]. Traditional classification algorithms for TSDR require enormous amount of data and are also limited to recognising road signs within the dataset. This does not work in real-life scenarios where a situation might occur when a critical road sign is not seen completely. In conventional networks, this might cause a problem but using a hierarchical approach (based on a certain taxonomy) it can give partial information such as it being a warning sign, etc. Also when a new road sign is available, conventional models will fail to recognise them since they were not trained on these signs. They will have to be re-trained every time new data is available. Considering an autonomous car, this simply means that if the car is trained with the road signs in the Netherlands, it will not be able to recognize most of the road signs in Switzerland (considering road signs are different in both the countries). This can be a dangerous situation.

To avoid this, several incremental learning algorithms have been proposed. Incremental learning methods have the capability of updating their knowledge when a new data class is available. These methods overcome the problem of retraining from scratch on availability of new

data. Traditional incremental algorithms however require some amount of data from the new dataset for training to achieve acceptable accuracy. These methods have been highly accepted and have shown great performance. But, one of the major drawbacks of incremental learning especially using artificial networks is catastrophic forgetting. Catastrophic forgetting is the tendency of an artificial neural network to completely and abruptly forget previously learned information upon learning new information [20]. The Learning without Forgetting (LwF) algorithm for Convolutional Neural Network (CNN)s proposed in [21] adds new nodes to an existing network for a new task only in the fully connected layers and this approach demonstrated to preserve the performance on old tasks without accessing training data for the old tasks. SeNA-CNN [22] is another algorithm which adds convolutional and fully connected layers of the new tasks to an existing model. SeNA-CNN has shown to have a better capability of learning new problems than LwF because they train a series of convolutional and fully connected layers instead of only training the added nodes in the fully connected layer. It must be noted that most of these incremental learning methods which aim to avoid catastrophic forgetting are focused on CNNs only. Another notable algorithm is Progressive Neural Networks (PNN) [23] which primarily focuses on reinforcement learning methods. It must be noted that dealing with forgotten data is not the only motivation for this work but is one of the aspects that will be considered while evaluating network performance.

In spite of so many state-of-the-art algorithms, there is always room for improvement and more research is being done to explore better alternatives. The best application to TSDR is for ADAS and these systems require very fast computation and decision making as the slightest delay can lead to unwanted circumstances. As mentioned earlier, neuromorphic chips have very low training and decision making times. In our opinion, such chips would be ideal candidates for on-board processing and real-time decision making in ADAS. To the best of our knowledge, no prior work has been done on realising a TSDR problem on a neuromorphic chip.

Based on the motivations so far and also knowing the fact that road signs can be represented as a taxonomy (explained in Section 2-2-1) we decided to test our framework on the road sign classification problem.

2-2-1 Taxonomy

Taxonomy is the practice and science of classification of things or concepts, including the principles that underlie such classification [24].

So far, we know that our framework will have a hierarchical classification structure along with incremental learning. The type of hierarchical structure is problem dependent. In our case, we have some nodes which have common parents but their following child nodes are different. In such cases, it is very difficult to incorporate them using a DAG structure. To precisely include all classes, we decided to use the tree structure. One could argue that the number of nodes per level can significantly increase in a tree structure; we will at a later stage show that the number of nodes per level has no effect on the proposed framework. The only important factor that is considered are the number of distinct features per level of the taxonomy. Depth of classification and exploration type will be discussed in a later chapter where we explain the functional architecture of the proposed framework. Readers will have a better understanding

of the choices made for the these methods once they know the functional architecture of the system.

Based on the decisions made so far, a taxonomy can be developed for the road sign classification problem. The proposed taxonomy is shown in Figure 2-6

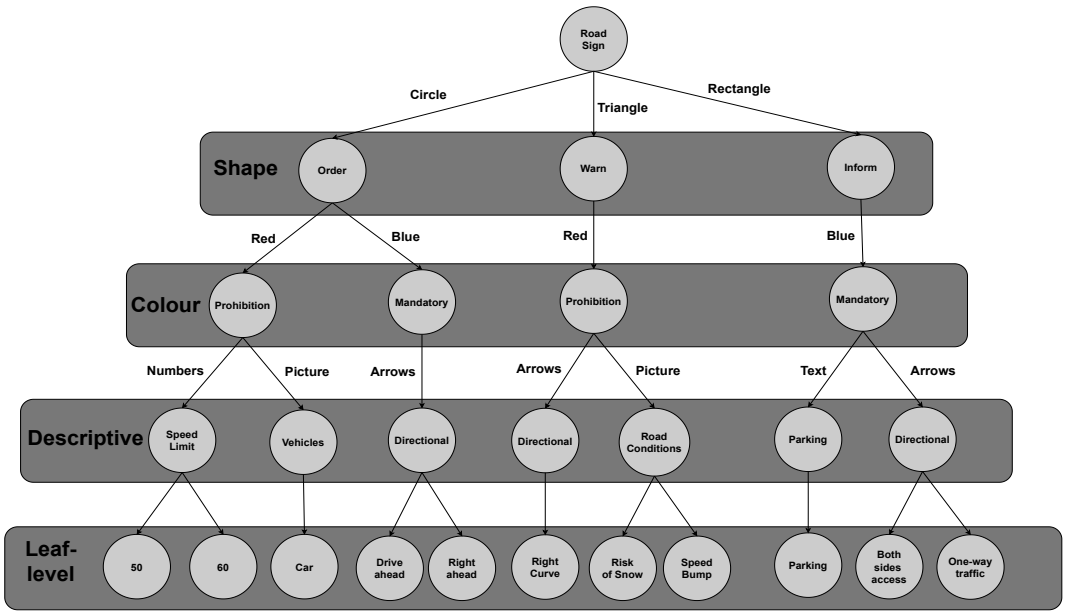

Figure 2-6: A possible representation of the road sign classification problem as a taxonomy. Each level of the taxonomy represents a feature of the road sign.

Figure 2-6 shows how a road sign classification problem can be represented by a taxonomy, making it an ideal problem for hierarchical learning. Each level gives a semantic meaning to the features of a road sign. In the above taxonomy, a circular shape and red colour would mean that it is a road sign which orders a prohibition. Such information is not available in case of a flat classification approach. It must be noted that this is not a complete road sign taxonomy and there are several exceptions to this taxonomy. The purpose of this thesis is not to completely solve the road sign classification taxonomy but to show how any data that can be represented as a taxonomy can be used for fruitful classification. Also, the semantic meaning of these features change in different regions of the world and hence it is difficult to have a general taxonomy for road signs. This taxonomy has been adapted from the guidelines set up by the UK Department for Transport [25]. Before proceeding further, following are a few terminologies that will be used throughout this document.

- **Class:** Each circle/node in Figure 2-6 is a class. It must be noted that these classes may or may not be unique.
- **Taxonomical Feature**: The arrows are taxonomical features that define the relationship between the classes of two levels.
- **Category:** It is a combination of classes and taxonomical features from the highest to lowest level of the taxonomy. An example of a category is the following chain:
  Road Sign $\xrightarrow{\text{Circle}}$ Order $\xrightarrow{\text{Red}}$ Prohibition $\xrightarrow{\text{Picture}}$ Vehicles $\rightarrow$ Car

This entire chain represents 1 category.

It is important to note this difference as they can lead to confusion while understanding the concepts in the upcoming chapters. In the next section, we introduce another element to the problem which is *transfer learning*.

2-3 Transfer Learning

Humans have a natural ability to transfer knowledge across different tasks. The knowledge gained while learning about one task is utilized to solve related tasks. The more related the tasks, the easier it is for us to transfer, or cross-utilize our knowledge. A simple example would be: if a person knows how to play a flute, then using this and some general knowledge about musical notes he can adapt and quickly learn how to play a trumpet. In the literature survey done prior to this thesis, transfer learning was explained in detail. The basic principle, transfer learning strategies and its advantages over conventional machine learning techniques were also discussed. A general transfer learning approach involves the following steps [26]:

- Take layers from a previously trained model.
- Freeze them, so as to avoid destroying any of the information they contain during future training rounds.
- Add some new, trainable layers on top of the frozen layers. They will learn to turn the old features into predictions on a new dataset.
- Train the new layers on your dataset.

A last step is *fine tuning*, which consists of unfreezing the entire model i.e. after training the new layers (or part of it), and re-training it on the new data with a very low learning rate. This is an optional step but will be used in this work.

2-3-1 Fine-tuning

Fine-tuning, in general, means making small adjustments to a process to achieve the desired output or performance. Fine-tuning involves using weights of a previously trained learning algorithm (in this case, using weights of a previously learnt neural network model) for programming another similar learning process. Some of the prominent fine-tuning frameworks include Keras [27], TensorFlow [28], MxNet [29], etc.

Fine-tuning can be used only when the dataset of an existing model and the new learning model are similar to each other. This is due to the way fine-tuning works. Since we also aim to test new classes belonging to the same or similar taxonomy, fine-tuning is an appropriate choice for this work. For this work, a general transfer learning framework with fine-tuning involves the following steps:

- **Step 1:** Load the network with the previously trained model.

- **Step 2:** Freeze some network(s) to avoid destroying the learnt model.
- **Step 3:** Train the remaining networks with a small portion of new data.
- **Step 4:** Learn the remainder of the dataset using predicted labels i.e. the network uses its old and new(small portion of the new dataset) knowledge to predict the labels for the remaining new dataset.
- **Step 5:** Unfreeze the frozen network(s) and update their existing model.

Finer details on how the above steps are adapted to our network are discussed in Chapter 4.

2-4 Summary

This chapter discussed the classification problem which can be divided into 2 types i.e. flat classification and hierarchical classification. The two approaches were compared and a decision was made regarding the method to be used based on the problem to be addressed and the functionality of the two methods. Next, the road sign classification problem was introduced. We presented an example of how the road sign classification problem can be formulated as a hierarchical learning problem by representing it in the form of a taxonomy. In Section 2-3, we introduced the concept of transfer learning and briefly discussed fine-tuning which is an approach to transfer learning.

Chapter 3

Neuromorphic Hardware

In this chapter, we discuss the hardware neuromorphic chip used in this thesis. Section 3-1 introduces NeuroShield. Section 3-2 discusses how the decision space is built and how we will be structuring it for this work. In section 3-3, we discuss the classification algorithms in NeuroShield.

3-1 NeuroShield

NeuroShield is a trainable pattern recognition board for Internet of Things (IoT) and smart appliances featuring the NeuroMem NM500 chip with 576 neurons which can interface with Arduino boards, Raspberry Pi or a PC. NeuroShield is developed by nepes corporation. The NM500 features NeuroMem technology which is developed by General Vision Inc. [30]. All board specific details and configurations can be found in the user manual [31]. The NM500 chip is a fully parallel silicon neural network: it is a chain of identical elements (i.e. neurons) addressed in parallel and which have their own "genetic" material to learn and recall patterns [32]. A NeuroMem network can build a non-linear decision space following a Radial Basis Function (RBF) model generator and more specifically a Restricted Coulomb Energy (RCE) neural network. They can also behave as a simple K-Nearest Neighbors (KNN) classifier. Hence, there are two stages involved i.e. *building the decision space* and *classification*.

3-2 Building the decision space

NM500 has a built-in RBF model generator for the learning phase. RBF networks were discussed in detail in the literature study conducted prior to this thesis and will hence not be explained here. The decision space is built by teaching a series of examples and labelling them with a category. Each time a feature vector is presented with a category to the network, the neurons first verify if one of them already recognizes it as belonging to an existing category. If this is the case, no action is taken. On the contrary, if the feature vector is not recognized by

any neuron, a new one is created to store it as a reference pattern with the assigned category. In addition, the new neuron inherits an influence field which defines its area of attraction or similarity domain. As more examples are taught, the decision space gets built accommodating smaller neurons which model the outliers. A neuron is represented in the decision space by the point (X, Y) surrounded by a dotted area which represents the influence field of the neuron and its shape is determined by the norm used for distance calculation.

There are five parameters/registers which contribute to shaping the decision space namely Maximum Influence Field (MAXIF), Minimum Influence Field (MINIF), Global Context Register (GCR), Distance Norm and Active Influence Field (AIF). Out of these four, AIF is set through back-propagation by the model generator while MAXIF, MINIF, distance norm and GCR are user dependent parameters. These can be set randomly or set by some user-defined optimisation algorithm.

3-2-1 AIF

The Active Influence Field of a neuron defines the amplitude of its similarity domain. If an input vector is similar to the prototype stored in the neuron and their distance is less than the neuron's AIF, the neuron will fire to acknowledge recognition of the input vector. The AIF of a neuron is automatically reduced when a learning operation requires that the neuron shrinks its similarity domain to leave some decision space for a new neuron [1]. This is a dynamic parameter which is adjusted during backpropagation by the model generator. As new neurons are committed, the decision space gets filled up forcing the previous neurons to shrink in order to accommodate new neurons. AIF is adjusted as per Equation (3-1).

$$\text{AIF} = \min(m, d_n) \tag{3-1}$$

where m is MAXIF and d_n is the distance to the closest neuron.

3-2-2 MAXIF

It is used to decide the conservatism of the network. Lower the value of MAXIF, higher is the conservatism. Hence, neurons with high MAXIF tend to over-generalise and recognise patterns with a high throughput.

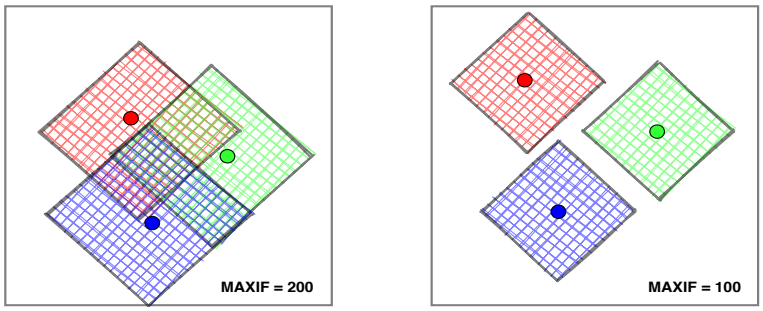

Figure 3-1: Effects of MAXIF on the decision space, Adapted from [1]. The coloured circles represent one committed neuron for the respective category and the diamond shapes represent their AIF.

Applications which have very small tolerance to errors shall have a smaller MAXIF value and vice versa. Figure 3-1 shows how different MAXIF values can shape the decision space and also tackle uncertainty[1].

3-2-3 MINIF

It is the minimum value below which the neuron's AIF can not reduce. The areas of uncertainty can be modulated with MINIF. If the AIF of a neuron becomes limited to this minimum value, it means that the prototype stored in the neuron lies close to the boundary of another category, and may be overlapped by another neuron of a different category. The higher the MINIF, the bigger the extent of *Uncertainty* zones in the decision space. It can be seen in Figure 3-2 how MINIF can enforce regions of uncertainty. The neurons in the decision space with MINIF = 100 are all included in influence fields of multiple categories which creates a state of uncertainty and hence will hamper the accuracy in the classification phase.

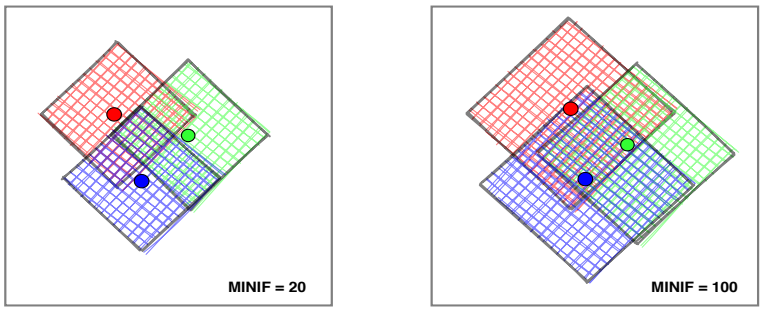

Figure 3-2: Effects of MINIF on the decision space, Adapted from [1].The coloured circles represent one committed neuron for the respective category and the diamond shapes represent their AIF.

3-2-4 GCR

The context allows you to partition the physical NeuroMem network into multiple virtual networks [33]. Each of these virtual networks can be trained to learn and recognise different objects. A context is selected by writing a context value to the GCR. The neurons belonging to a particular context have a separate decision space. This means that if a network has N different contexts, it contains N different feature spaces. When a context is set, neurons corresponding to only that context participate in learning and classification. Neurons belonging to other contexts turn idle.

However, if an application requires to access the entire decision space, the context value can be set to 0. A GCR equal to 0 activates all the neurons regardless of their context value. Context 0 is generally used in the classification phase when an input vector needs to be compared to the entire decision space.

[1] shared/overlapping decision spaces between categories

3-2-5 Distance Norm

The distance norm determines how to calculate the distance between the reference pattern or prototype stored in a neuron (C) and an input vector (X). If the calculated distance is less than the influence field of the neuron, the vector X is considered as similar to the prototype C. There are two norm options in NM500 which are discussed below.

1) L1 Norm (Manhattan Distance): It emphasizes the drift of the sum of the all components between X and C. The norm can be mathematically represented as shown in Equation (3-2)

$$d(X,C) = ||X - C||_1 = \sum_{i=1}^{n} |X_i - C_i| \tag{3-2}$$

where n is the number of elements in the prototype C and input vector X. In a two dimensional space, the points X such that their distance D(X,C) to a reference point C is less than or equal to L describe the area of a diamond shape centered in X and with a side equal to $\sqrt{2}L$.

2) Lsup Norm: It emphasizes the largest drift of the same component between X and C. The norm can be mathematically represented as shown in Equation (3-3)

$$d(X,C) = ||X - C||_{sup} = \max |X_i - C_i| \tag{3-3}$$

In a two dimensional space, the points X such that their distance D(X,C) to a reference point C is less than or equal to a constant C describe the area of a square centered in X with a side equal to $2L$.

In this work, we are using feature vectors to discriminate between different features at different levels of the taxonomy. This means that all the features in a corresponding feature vector contribute towards a decision. For this reason, we cannot use the *Lsup* norm as it considers the largest drift between features of the prototype and reference thereby leading to a decision based on a single feature. On the contrary, *L1* norm considers all the features and hence is the ideal choice of distance norm for our approach.

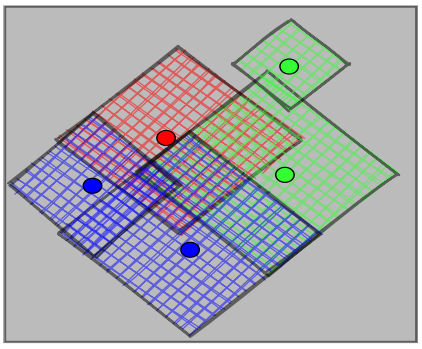

Figure 3-3: Decision space mapped using RBF classifier. Gray area represents unmapped decision space. The colours represent the category associated with neurons, diamond shapes represent the AIF of the committed neurons and circles represent committed neurons.

An example of a decision space mapped using an RBF network is shown in Figure 3-3. The decision space is utilized as effectively as possible to ensure enough space for new neurons. A fully modelled decision space does not necessarily imply that more neurons can not be committed. The decision space is partially mapped and certain zones are left as unmapped.

3-3 Classification

Classification of a prototype consists of evaluating if it is recognised by any neuron i.e. if it falls in the influence field of one or more neurons modelling the decision space. When an input vector is presented to the network, all neurons calculate their distance between the input vector and the prototype stored in their memory. If the distance of a neuron is greater than its influence field, the neuron excludes itself from the list of neurons recognizing the vector. Otherwise it fires to indicate that it somewhat recognizes the vector. The similarity range is expressed with the distance value. In classification, there are three possible outcomes:

- If all firing neurons have the same category, the input vector is positively identified.
- If the firing neurons do not have the same category, the input vector is recognized but with uncertainty.
- If no neuron fires, the input vector is unknown.

During the classification phase, the classifier outputs the following parameters for each firing neuron:

- Neuron ID: The ID of the firing neuron in the decision space.
- Distance: Distance of the input vector to the firing neuron.
- Category: The label associated to the firing neuron.

There are two built in classifiers in the NM500 chip which are explained below.

3-3-1 RBF

The response can be refined when using this classifier as uncertainty and unknown classes can also be modelled. RBF classifier takes into account the influence field of the neurons i.e. when using an RBF classifier, a neuron will only fire if the input feature vector falls in the AIF of any neuron modelling the decision space. An example is shown in Figure 3-4. In the figure, 2 input feature vectors are presented for classification out of which the yellow cross represents a feature vector which did not fall in the AIF of any neuron and the black cross lies in the influence field of a neuron with category red. In such a case, no neuron will fire when the yellow feature vector is presented for classification since it does not fall in the AIF of any neuron. It is simply classified as unknown with no information about closest neurons in the decision space. The position of this feature vector in Figure 3-4 is just for explanation but in reality, its position will be unknown due to lack of information about its closest neurons.

3-3-2 KNN

KNN classifier completely ignores the notion of influence field and it always returns K responses which are the first K closest matches based on shortest distance. If KNN mode is active while building the decision space, only one neuron per category can be committed

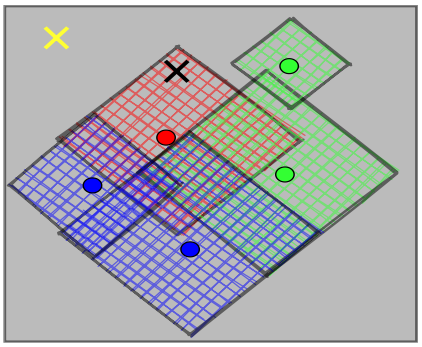

Figure 3-4: An example showing two different input feature vectors in the modelled decision space. Yellow cross represents an input feature vector which does not fall into the influence field of any neurons and black cross represents an input feature vector which falls under the influence filed of a neuron with category red.

which is why it is not used as a model generator and only as a classifier. An example of a decision space mapped using KNN mode is shown in Figure 3-5. As can be seen in Figure 3-5, the entire decision space is mapped which is why it is not used as a model generator. Figure 3-6 shows the same two feature vectors as in the RBF example now presented to the KNN classifier for classification. But in this case, the top k neurons will definitely fire giving out the distance of the yellow cross feature vector from the firing neuron. According to the figure, neuron with red category will be the closest to the yellow feature vector.

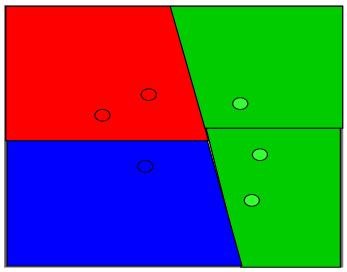

Figure 3-5: Decision space mapped using KNN classifier. Entire decision space is mapped. Colours represent the category associated with the committed neurons, shapes represent the AIF of the committed neurons and circles represent committed neurons.

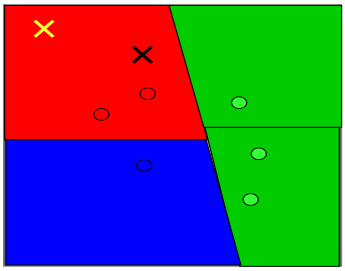

Figure 3-6: An example showing two different input feature vectors in a decision space interpreted using a KNN mode. Colours (except yellow and black crosses) represent the category associated with the committed neurons, shapes represent the AIF of the committed neurons and circles represent committed neurons.

For this work, a combination of both classifiers is used. Each input sample will be classified using an RBF classifier initially. In case of an unknown outcome, it is still essential to know which neurons are closest to the presented input sample. This information will be essential for any decision making logic used. Since an RBF classifier will not search for classes if the input vector does not fall within the influence field of any neuron, we will switch to a KNN

classifier to search the entire decision space. The classification algorithm used is explained in detail in Chapter 5.

3-4 Summary

In this chapter, we described the neuromorphic hardware that will be used in this work. Section 3-1 briefly introduced NeuroShield, which is a pattern recognition board featuring the NM500 chip. Section 3-2 discussed in detail how the decision space was modelled and various important parameters that affected the decision space modelling. In Section 3-3, we discussed the two in-built classifiers i.e. RBF and KNN and the key differences between them. Towards the end of this section, we concluded that a combination of both these classifiers would be beneficial for our proposed framework.

Chapter 4

End-to-End Framework

In this chapter we discuss the overall end-to-end framework. There are three important components: image pre-processing pipeline, back-end networks and the front end network. In this chapter we will discuss the general structure of the network and the three cases implemented in this work. Image pre-processing pipeline and back-end networks will be explained in detail in Chapter 5 and the front-end network is explained in Chapter 6.

4-1 Network Structure

The end-to-end framework proposed in this thesis is shown in Figure 4-1. From the figure, one might argue that it is a complete feedforward network, that is however not the case here. This is just a general structure showing individual components of this framework. The update direction and feedback mechanism will be explained in detail in Chapter 5 and Chapter 6. The back-end networks consist of 4 individual feature extraction networks. Each of these networks extract key features per corresponding level of the taxonomy. It can be inferred that each feature extraction network acts as one level of the taxonomy. The final network is a separate network and is also trained separately. Each back-end network is a separate pre-classifier and hence function as *local classifiers per level*. The output of these networks is transformed into a custom feature vector which is the input to the front-end network. This networks learns this taxonomy information taking into account the hierarchy as a whole. Hence, the front-end network is a *global classifier*. Overall, the end-to-end network is a combination of *local classifiers per level* and a *global classifier*. Due to such a structure, this network is less sensitive to error propagation. This is because of the ability of the front-end networks to completely or partially classify any custom feature vector.

All four networks are trained separately and have their own decision space. Each of these networks have a unique context value written to the Global Context Register (GCR) and hence form independent decision spaces. Neurons corresponding to one context have no connection to neurons in other contexts. Using separate decision spaces ensures that each of these

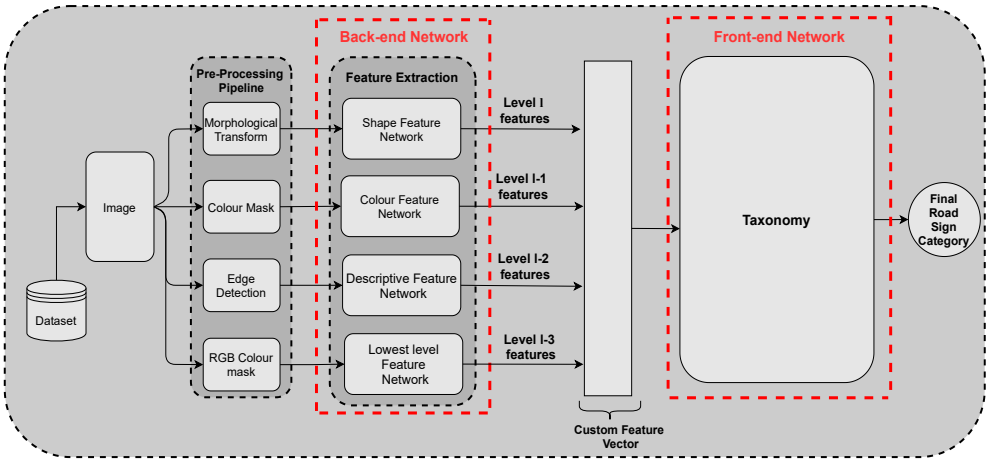

Figure 4-1: General End-to-end Network structure

networks precisely learn only the required features while discarding all unwanted features. This increases the accuracy of these networks thereby leading to better classification results.

The entire approach is divided into 3 frameworks as follows:.

Learning Phase: The *Learning Phase* involves training the neural network with relevant data to build a decision space. Figure 4-1 shows the regular learning framework. This phase can be seen as training four individual networks, but parallel to one another. Each network is trained with the same image dataset but they receive a particular representation of the input image after pre-processing. We want each network to represent one level of the taxonomy and hence each network must output the features belonging to only its corresponding level. The output of these back-end networks are converted into relevant values of the custom feature vector (Explained in detail in Chapter 6). The custom feature vectors act as feature inputs for the front-end network.

Online Incremental Learning: The back-end networks are completely unaware about the taxonomy and so when an input sample is presented, they can not decide if it aligns with the taxonomy or not. They are trained to extract certain features without any knowledge of the taxonomy. When a test sample that does not align with the taxonomy is presented with features that are part of the learnt dataset, these networks will classify them correctly and form the custom feature vector. But, the front-end network will not be able to recognise this sample as it does not align with the taxonomy. In such a scenario, the network will give an unknown output. A decision has to be made whether to learn it as one of the existing taxonomical features or introduce a new feature (Discussed in detail in Chapter 6). In either case, the network has to learn this sample and update the taxonomy. This learning happens online and is referred to as incremental learning. This scenario is shown in Figure 4-2

Transfer Learning: The above two scenarios show a regular learning phase and an online incremental learning phase. In the incremental learning phase, no new taxonomical features are involved but unknown combinations of existing features are learnt incrementally. This leads us to think about situations when an unknown features are observed. With respect to the taxonomy shown in Figure 2-6, an unknown feature could be a road sign with a yellow colour or with a rhombus shape and also a combination of these. Learning these new taxonomical features generally result in loss of previously learnt data and therefore the entire

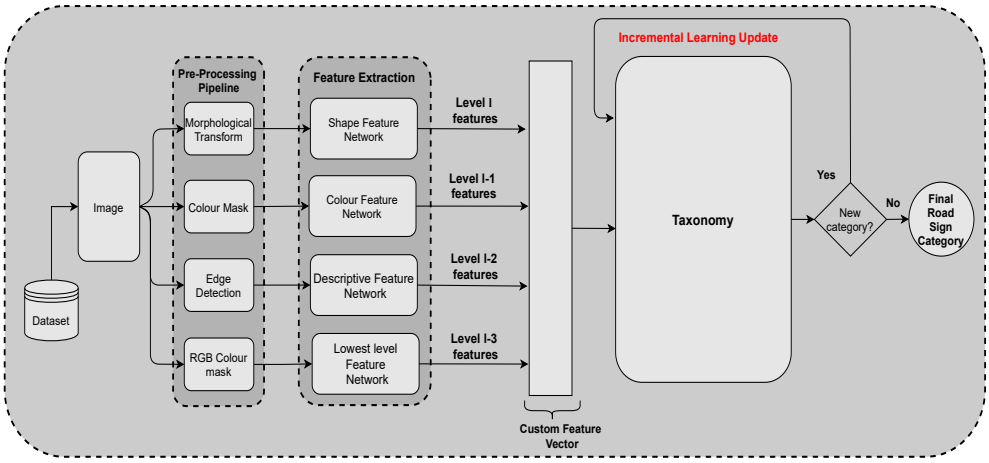

Figure 4-2: Online Incremental Learning

network needs to be re-learnt. To avoid re-learning from scratch, we use a transfer learning approach which is described below.

Fine-Tuning: Fine-tuning involves re-training an existing network on a new dataset with a very low learning rate. This can potentially achieve meaningful improvements, by incrementally adapting the pre-trained taxonomical features to the new data. The fine-tuning network structure is shown in Figure 4-3.

Finer details on the transfer learning steps mentioned in Section 2-3-1 with reference to the framework shown in Figure 4-3 are mentioned below. First, the front-end and back-end networks are loaded with the previously learnt neural network model. Next, we freeze[1] the front-end network which will avoid destroying the existing taxonomy modelled in the network. In step 3, the back-end networks are trained on 5% of the new dataset in a supervised manner. Remainder of the dataset is presented to the network for classification. For every unknown recognition outcome, the category of the sample is predicted and a new neuron is committed with the predicted category; which is step 4. Prediction of these categories is done using the classification logic explained in section 5-3-2. The weight i.e. the Active Influence Field (AIF) is updated automatically by the model generator in the NeuroShield. The generated custom feature vectors are saved so that they could be used later while training the front-end network.

Step 5 includes unfreezing and updating the front-end network by writing new elements to the existing neurons. These new elements correspond to new taxonomical features that are learnt during transfer learning and hence the size of the custom feature vector will change due to increased elements. Since the new custom feature vector has a different size, all other previously learnt neurons must be adapted to this size in order to represent the new taxonomy. This is not equivalent to re-learning the entire network. In this case, we are using the existing knowledge as it is and just updating the taxonomy by inserting a new element(s) in the learnt feature vectors. Re-learning the whole network involves having a new dataset and completely replacing the old knowledge with a new one, which is not the situation in our case. It must be noted that the front-end network is updated offline since the contents of all committed

[1] weights of the network are not updated

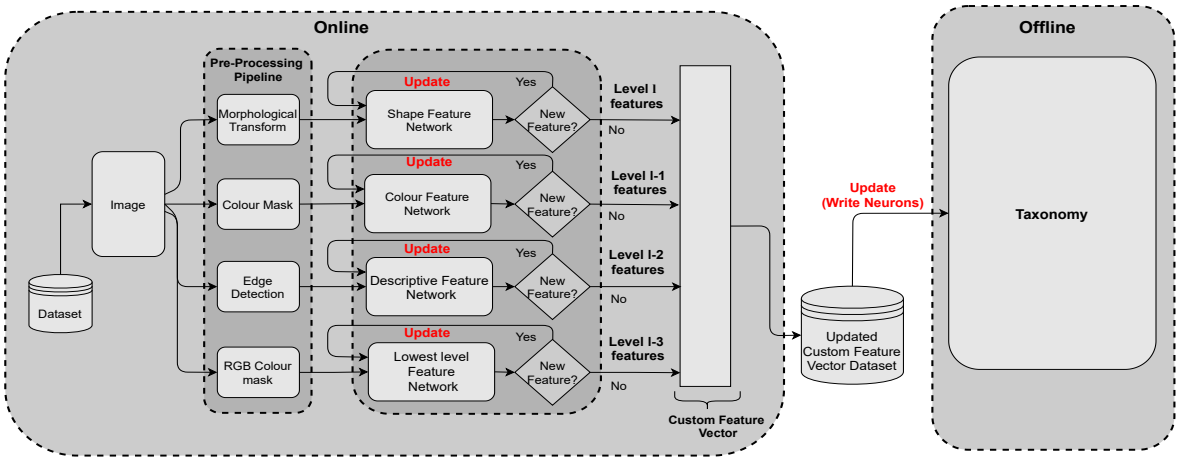

Figure 4-3: Transfer Learning - Fine tuning

neurons are first read and then new elements are added to the existing content. For the previously committed neurons, new elements corresponding to newly learnt features will have a non-recognition outcome value since they already recognise 1 feature at every taxonomy level. Once these new elements are written, we can safely say that the taxonomy structure has been updated in the front-end network i.e. the network is aware about new features at corresponding taxonomy levels. Lastly, the front-end network is trained on the new categories with its input being the saved custom feature vectors generated by the back-end networks for these new features. The performance of this network is shown in Section 7-4.

4-2 Summary

In this chapter we very briefly discussed the structure of the end-to-end framework. We discussed the learning phase which is involves training the end-to-end framework with a basic dataset. We then discussed the incremental learning phase which is performed on a new dataset which consists of different combinations of existing features. Next, we discussed transfer learning and fine-tuning which can deal with new taxonomical features and unseen data. Transfer learning along with fine-tuning involved training the back-end networks with a small portion of the new dataset while keeping the front-end network frozen. Lastly the front-end network is updated by writing new element(s) to the existing neurons in order to incorporate the updated taxonomy. This way, none of the networks are completely re-learnt. In the next chapter, we will discuss in detail the back-end networks.

Chapter 5

Back-end Networks

The overall network design shown in Figure 4-1 is primarily divided into two sections — the back-end network and the front-end network. In this chapter we will discuss the road sign detection problem, pre-processing pipeline and then present the internal structure of each back-end network and discuss in detail the classification and decision making logic. Section 5-1 very briefly discusses about how the road sign is extracted from an image. Section 5-2 discusses about the pre-processing pipeline for each of the feature extraction network. Section 5-3 focuses on the training and classification of these networks followed by various recognition outcomes. This chapter will be concluded with a short summary.

5-1 Extraction

In an image, there are several features apart from the ones desired. This work does not focus on detecting road signs but rather on classifying the detected ones. But since we do not have cropped road sign images, we have a created a simulation environment in Gazebo. To precisely extract road signs, we use colour thresholding. It is used to extract parts of the image that fall within the specified colour range.

Firstly, the image is converted into Hue, Saturation, Value (HSV)[1] colour space. The advantage of using HSV colour space is that it separates the image intensity from the colour information. Secondly, it is less noisy. It is easier to recognise different shades of colours due to intensity information being available. Road signs are primarily red, blue and yellow in colour. For this road sign extraction, we use colour filters to detect red, blue and yellow colours.

After converting to HSV colour space, a mask is created for corresponding colour values in the HSV colour space. A mask separates the desired colour from the original image by converting the desired colour into white and everything else into black colour. Finally, a bitwise AND operation is performed over the original and the masked image. One could argue that this

[1] HSV is a model to represent the colour space similar to the Red, Green, Blue (RGB) color model [34]

might not always work on a real dataset since the colours are not so bright and they fade away with time. Colour detection and segmentation have been used by many researchers and has proven to be an efficient method for road sign extraction. Hasan Fleyeh proposed a few methods for colour segmentation to detect road signs [35], Supreeth et al. in their work showed that colour segmentation methods achieved a very high accuracy during different environment and lighting conditions [36]. It is known that colour segmentation methods work pretty well on real datasets and any of these methods can easily be adopted to our case. For this thesis, we just want to get the back-end networks working so that we can evaluate the our proposed custom feature vector and the front-end network. For this reason we adopt simple existing methods available in literature for road sign extraction and image pre-processing. Some of these existing methods have been slightly modified to suit achieve desired pre-processing results for our dataset.

5-2 Image Pre-processing

Image pre-processing is required in order to improve an image and remove any unwanted features. Removal of unwanted features helps the network model a better decision space and also significantly enhances the classification accuracy. As seen in the previous chapter, we have 4 levels in our taxonomy and hence 4 feature extraction networks. For each of these feature extraction networks, an image pre-processing step is performed.

5-2-1 Shape Features Pre-processing

To learn the shape of the road sign, we only need to send features relevant to it's shape. We can discard any external noise, the contents within the road sign, and other unwanted features. Firstly, we remove unwanted noise by applying a 7×7 kernel median filter. Median filtering algorithm will scan the entire image, using a small matrix/kernel, and recalculate the value of the center pixel by taking the median of all of the values inside the matrix [37]. The Gaussian filter is also very good for noise removal but it does not preserve edges in the image. Since we need to perform a morphological transformation later, we will require the edge information to obtain the desired shape features. Hence, we choose to apply a median filter for noise removal.

Morphological Transformations have been effectively used in literature for shape detection in images. Dartmouth college explains the importance of morphological operation in shape detection in their course on *Image Processing in IDL* [38]. Darlis Herumurti et al. [39] and W.K.I.L. Wanniarachchi1 et al. [40] use morphological transformations in their work for shape manipulation to detect road signs. Morphological transforms can either be applied to binary or grayscale images. We apply a binary threshold on the image to convert it to binary format. Binary thresholding is simply applying a same threshold value for an image. If the pixel value is smaller than the threshold, it is set to 0, else set to a maximum value. Since we are only interested a black and white image, the threshold is set to 0 and the maximum value is set to 255.

To avoid unwanted background information, it is required to crop only the road sign from the entire image. For this, we perform contour detection using OpenCV. We are concerned with

the shape of the detected road sign and hence we only detect external and closed contours. A bounding box around the detected contour is approximated. This rectangular box is cropped out of the image and now we have an image with only the road sign. Finally, a morphological gradient of the image is taken and then resized to 16×16 to make it NeuroShield compatible. A morphological gradient is the difference between dilation and erosion of a given image. Erosion is a concept similar to soil-erosion. It erodes away the boundaries of foreground object. The kernel slides through the image (as in 2D convolution). A pixel in the original image (either 1 or 0) will be considered 1 only if all the pixels under the kernel is 1, otherwise it is eroded (made to zero) [41]. Dilation is just opposite of erosion. Here, a pixel element is '1' if at least one pixel under the kernel is '1'. So it increases the white region in the image or size of foreground object increases [41]. An example of erosion, dilation and morphological gradient operations applied to an image is shown in Figure 5-1. A morphological transform gives best results when applied to binary images which is primarily why we applied a binary threshold before a morphological transform. The entire pre-processing pipeline is shown in Figure 5-3.

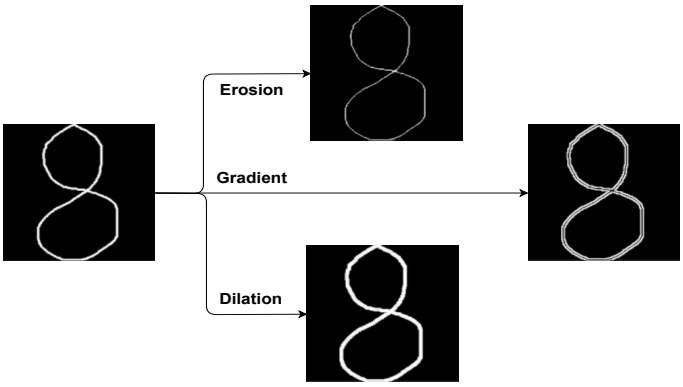

Figure 5-1: Example of Erosion, Dilation and Morphological Gradient

5-2-2 Colour Features Pre-processing

Blue, Green, Red (BGR) image format supported by OpenCV is used in this work. Extracting colour information is necessary when dealing with road signs as it is an important feature to discriminate between different them. In this work, we focus only on red and blue road signs. We apply a colour threshold for these two colours in the HSV space and mask out all other colours. Now, our image primarily consists of one of these two colours depending on the road sign as can be seen in Figure 5-2. The concept of dominant colour can now be used and so we simply use the mean values of each B, G and R channels of the image. This is sufficient to get an idea of the dominant colour in an image. Hence, the input vector for the colour network only consists of 3 features which are the mean values of B, G and R channels respectively.

5-2-3 Descriptive Features Pre-processing

Descriptive features correspond to features like numbers, arrows, pictures of cars, bicycles, etc. All the pre-processing steps are similar to the ones mentioned in section 5-2-1 except for

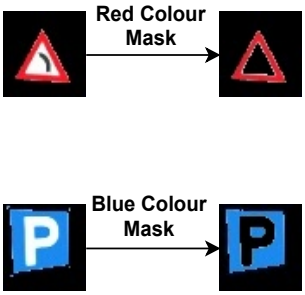

Figure 5-2: Image segmentation using colour spaces

the last one. After the bounding box is cropped out of the image, we calculated the sobel gradient of the image. Inspired by the work done in [42], [43] and [44], we decided to use sobel operator for edge detection. The sobel operator is an edge detection algorithm that uses two 3×3 kernels which are convolved with the original image to calculate approximation of the derivatives. One kernel is used for horizontal changes and the other for vertical changes [45]. It emphasizes regions of high spatial frequency that correspond to edges [46]. Suppose we have an input image I, and G_x and G_y are two images which at each point contain horizontal and vertical derivative approximations respectively, then the computation is as follows [46]:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * I \quad and \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * I \tag{5-1}$$

The kernels are designed to respond maximally to edges. In this work, we combine the two gradients to get the absolute magnitude of the gradient at each point. The gradient magnitude is given by:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{5-2}$$

The result of applying a sobel operator on an image is shown in Figure 5-3.

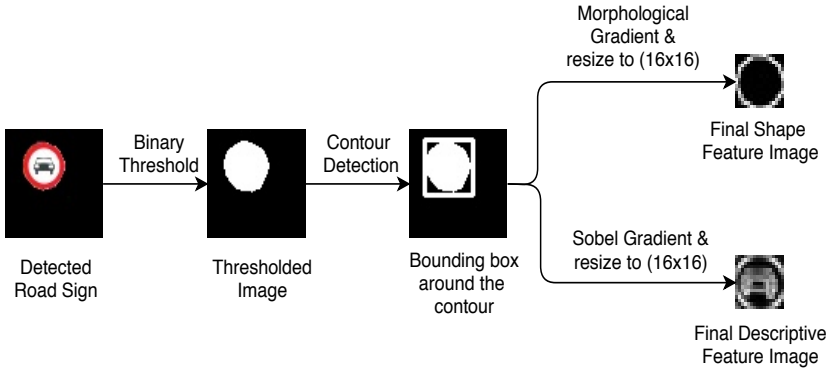

Figure 5-3: Image pre-processing pipeline

5-2-4 Final Features Pre-processing

These are the lowest level features which represent the finest details of the image. These features include further details of the numbers, arrows, etc. Examples of these features include the value in case of a number (50, 60, etc.), direction in case of an arrow (straight, left, etc.). Hence, we want to send only that information to the network so that these features can be distinguished precisely. We perform a similar pre-processing step as in 5-2-2 to extract blue, and red colours. We then perform image subtraction with the original image to only obtain the white and black portions of the road signs which are the final features. This method has been adapted from the work done by Keser et al. [47]. An example of this operation is shown in Figure 5-4.

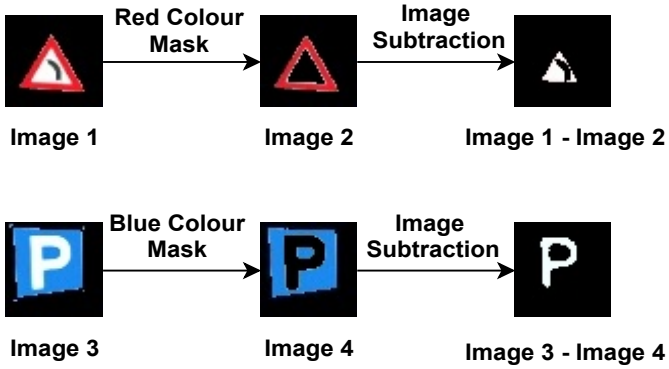

Figure 5-4: Final feature pre-processing

5-3 Back-end Networks

Back-end networks are those networks which are trained to classify different features of the corresponding level in the taxonomy based on pre-processed information of the image. Back-end networks are shown in Figure 4-1. According to Figure 4-1, there are four back-end networks, one for each level of the taxonomy. Each of these networks is trained with different feature vectors corresponding to the features at respective taxonomy levels. The four networks are named as colour feature network, shape feature network, descriptive feature network and lowest level feature network respectively. As the name suggests, they are trained to learn colour features, shape features, descriptive features and the lowest level features respectively. The learning and classification logic for all back-end networks is the same and is explained in the following sections.

5-3-1 Learning

The pre-processing pipeline removes all the unwanted features and produces an image with only important features. As mentioned earlier, the colour features are not sent in the form of an image but instead as a feature vector of three elements with the mean values of BGR channels respectively. On the other hand, the shape and descriptive networks receive as an input the entire pre-processed image as shown in Figure 5-3.

As mentioned earlier, we will use NeuroShield board with NM500 as our neuromorphic chip and Radial Basis Function (RBF) as our neural network activation function. It has so far been mentioned that there are four of these back-end networks, but actually that is not the case. NeuroShield has a special register known as the Global Context Register (GCR) which allows partitioning of a network into multiple virtual networks. Each of these back-end networks correspond to a different context value and hence form virtual networks each creating different decision spaces. There is just one big network divided into virtual networks thereby creating a modular neural network architecture. GCR is explained in detail in Section 3-2-4. The learning logic for all the back-end networks is explained in the following paragraph.

An iterative learning method is used to learn the feature vectors for every network. From chapter 3, we know that the existing neurons tend to shrink their influence fields when new neurons are committed. The shape of the decision space depends on the sequence of training examples. This temporal dependency is not ideal and can lead to building a poor decision space. This is not a problem when the number of examples to be learnt are very few. But in real life machine learning problems, this is not the case. Having a large dataset and many different categories to learn, the number of neurons required will be significant. In such a case, as more neurons get committed, the existing neurons closest to the newly committed neurons will shrink in order to accommodate them. This results in forgetting i.e. some of the previously learnt samples will no longer be recognised by these committed neurons. To avoid this, we opted for an iterative learning approach.

The learning is done over a few epochs to model an accurate decision space. The training samples are repeatedly learnt over epochs until the stopping criterion is reached. The stopping criterion is reached when no more neurons are committed. One might argue that in cases where training samples are too many, this process will be very time consuming and computationally expensive. However, this is not the case. In the NeuroShield, modelling the decision space is triggered by writing the CAT register[2] which takes a fixed number of cycles irrespective of the number of committed neurons. There is no significant time increase when learning over several epochs. Using the above logic, a new neuron is committed for all the samples that are no more recognised after the previous epoch. The decision space is modelled in an iterative manner and when no more neurons are committed, learning is complete. Learning curves shown in Chapter 7 will give the reader further insight on iterative learning.

5-3-2 Classification

Like learning, every network has the same classification logic. Classification is done to assign a category to an input feature vector. Figure 5-7 shows a flowchart explaining the classification logic. There are two different segments in Figure 5-7 namely Classification and Custom Feature Vector Generation. The output of classification segment is used to generate a custom feature vector which act as input features for the front-end network. Custom Feature vector generation and front-end network are discussed in detail in Chapter 6. In this chapter, we will focus on the classification segment only.

When an input vector is presented to the network, a broadcast command is executed which raises a recognition status flag. In case of recognition, there are 2 possibilities: *Identified*

[2]Register which learns the category of the input vector

and *Uncertain*. In the case when the input vector is not recognised, the recognition status is *Unknown*.

Recognition

A recognition flag is set when the input vector falls within the influence field of a single neuron or multiple neurons. If it falls within the influence field of a single neuron i.e. only one neuron recognises the input vector, the recognition status is *identified* and if it falls within the influence field of multiple neurons i.e. more than one neurons recognise the input vector, the recognition status is *uncertain*. Since we already have one or multiple neurons recognising the input vector, there is no need to search the entire decision space. Hence for both these cases, we choose to further classify the input vector using an RBF classifier.

When the recognition status is *identified*, only one neuron has recognised the input vector. Clearly, the input vector belongs to the category corresponding to this firing neuron. The classifier outputs the neuron ID, distance to the firing neuron and the category of the firing neuron. Classifier outputs have been discussed in Chapter 3. Using the neuron ID from the classifier output, the corresponding neuron contents are read to obtain its Active Influence Field (AIF). A confidence is calculated for the input vector which indicates how similar it is to the firing neuron. Confidence is calculated as shown in Equation (5-3)

$$confidence = \left(\frac{distance\ from\ firing\ neuron}{\text{AIF}} \right) \tag{5-3}$$

Following can be interpreted from different values of the confidence metric:

- $confidence = 0$: The input sample and the neuron are identical.
- $0 \leq confidence \leq 1$: The input sample lies within the influence field of a neuron.
- $confidence = 1$: The input sample lies on the boundary of the neuron's influence field.
- $confidence > 1$: The input sample does not fall within the influence field of any neuron i.e. it is not recognised.

When recognition status is *uncertain*, multiple neurons associated with different categories recognise the input sample. This creates a state of confusion and the classifier cannot decide which neuron is the input vector associated to. To resolve this confusion, we calculate the confidence. It is calculated using the same formula mentioned in Equation (5-3). Confidence of the input sample corresponding to each firing neuron is calculated and compared. Lower the value of confidence, better the match. Hence, the lowest value of confidence is the best possible neuron for the input sample. Therefore, a decision is made to assign the input sample with the category associated to this neuron. This is a strict decision and could sometimes discard neurons with almost equal confidence. To soften it, a better approach would be to have a threshold determining which amongst the two most confident neurons should the sample be classified as. Let $confidence_1$ and $confidence_2$ denote the confidence values of the most and second most confident neuron respectively, then for the sample to be associated with category of the most confident neuron the following condition must be satisfied.

$$confidence_1 - confidence_2 < threshold \tag{5-4}$$

This threshold value could be set to a small value in the range $[0.01, 0.05]$. We did not focus much on realising this condition since the focus of this work was primarily on the taxonomy and front-end network and the prior method achieved satisfactory results. In case of multiple firing neurons, there will be situations when both neurons belong to the same categories. If so, there is ideally no need to calculate the confidence but we will still use the same method to decide which amongst the firing neurons is the most suitable. Although this won't have any impact on the classification outcome, but this can be essential information for further analysis like improving the decision space, understanding which neurons fired the most, etc. Analysing all these factors can be very useful to understand how much each neuron contributes to the decision space and how removing minimally used neurons from the decision space would affect the network performance. This analysis is not within the scope of this thesis but is certainly important for developers and probably neuromorphic chip manufacturers for debugging and performance analysis.

We use this confidence metric for decision because it indicates how much inside the input sample is within the AIF of the firing neuron. One could argue that the classifier already outputs the distance to the firing neurons and hence there is no need for this extra evaluation. But it is important to note that this is the distance form the input vector to the center of the neurons AIF. There can be situations when the distance of the input vector is less but it still lies almost on the boundary of the neuron's AIF, while it might lie pretty much inside the boundary of a second neuron with a higher AIF. In such cases, the input vector is more likely to belong to the category corresponding to the second neuron than the first. This is depicted in Figure 5-5. In the case shown in Figure 5-5, the input vector lies closer to the center of the

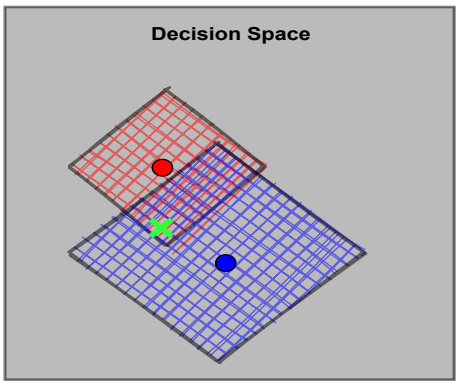

Figure 5-5: An example of a decision space with two neurons. Red and blue colours indicate two different categories for each neuron and their influence fields are indicated by the diamond shapes. The circles cross marks are committed neurons and the green cross mark is the input vector

red neuron. But when the confidence will be calculated, the blue neuron will have a lower confidence value as the input vector lies further inside the blue neuron than the red one. The input vector is almost on the boundary of the red neuron and will be forgotten even if the red neuron shrinks slightly. It is more likely to be part of the blue neuron than being an outlier to the red neuron. Hence, we decided to use the confidence metric for decision making instead of the minimum distance to the neuron center.

Unknown

Recognition status is unknown when none of the neurons recognise the input vector. In such a case, RBF classifier is of no use since we know that the input vector does not fall within the AIF of any neuron. Hence we switch to a K-Nearest Neighbors (KNN) classifier to explore the entire decision space. Using a KNN classifier, the top k closest neurons fire. For each of these firing neurons, we calculate the confidence as shown in Equation (5-3) and sort them in increasing order of confidence value i.e. lowest value of confidence first. The category corresponding to the most confident neurons is classified as uncertain and all others are classified as not recognised. Let us consider an example to get a better idea about the confidence metric in case of an unknown outcome. Figure 5-6 shows a simple decision space with three neurons represented as blue and red circles (colours indicate the category associated to the neurons) and an input sample shown as a yellow cross mark. The coloured shapes indicate the influence field of these neurons. Let's assume that the influence fields of neurons 1,2 and 3 are 100, 200 and 160 respectively and the distance of the input sample from each of these neurons is 120, 370 and 170 respectively. The the confidence values for each neuron would respectively be 1.2, 1.85 and 1.06. In this case, the unknown sample will be classified or learnt with a category corresponding to neuron 3. This is a strict decision and could lead to learning some neurons with incorrect categories. But we found no such cases while testing this network on our dataset. Since we only want to get the back-end networks working to test the front-end network, we decided to continue with this decision making condition.

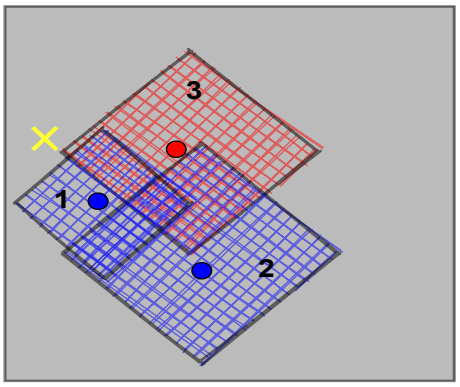

Figure 5-6: Example of an unknown input sample in the decision space

5-4 Catastrophic Forgetting

Catastrophic forgetting or catastrophic interference was first recognized by McCloskey and Cohen [48]. They found that, when training on new tasks or categories, a neural network tends to forget the information learned in the previous trained tasks. This usually means a new task will likely override the weights that have been learned in the past, and thus degrade the model performance for the past tasks. Prominent examples of catastrophic forgetting are incremental learning and transfer learning using deep neural networks. In a typical transfer learning scenario, the weights of the old network are adapted to the new task resulting in

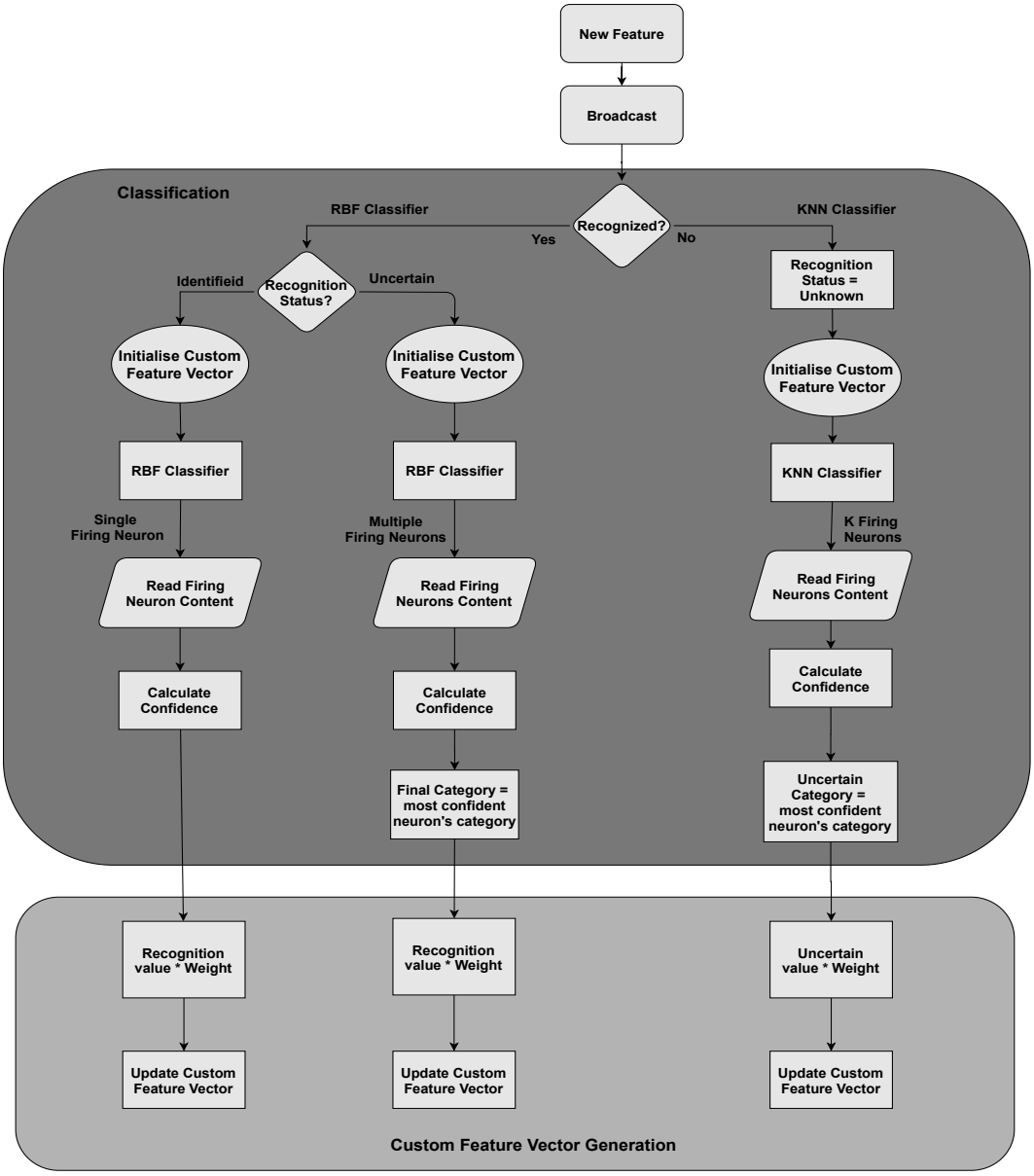

Figure 5-7: Classification Logic

disruption of weights learned for the old task [49]. In our case, catastrophic forgetting can be seen as neurons shrinking their influence fields in order to accommodate new neurons i.e. some of these neurons will no longer recognise certain samples which they were able to recognise earlier. We define two types of catastrophic forgetting based on uncertainty and unknown recognition.

- **Partial Forgetting:** It is defined as the percentage increase of samples with an uncertain recognition outcome after an incremental or transfer learning update. We refer to this as partial forgetting because these samples are still recognised by the network but with multiple neurons. It could be argued that this does not refer to forgetting

but according to us, if a neuron that had a recognition outcome as recognised in the old network is now being confused with multiple categories, then the network tends to have lost some information about that sample. Let $uncertain_{new}$ denote the number of samples with an uncertain recognition outcome after an incremental or transfer learning update, $uncertain_{old}$ denote the number of samples with uncertain recognition outcome in the network trained only on the old data and *total samples* denote the number of samples in the old dataset i.e. without new features, then partial forgetting is calculated according to

$$partial\ forgetting = \frac{uncertain_{new} - uncertain_{old}}{total\ samples} \tag{5-5}$$

- **Complete Forgetting:** It is defined as the percentage increase of samples with an unknown recognition outcome after an incremental or transfer learning update. These samples are no longer recognised after learning new data thereby concluding that the network has completely forgotten these samples. Let $unknown_{new}$ denote the number of samples with an unknown recognition outcome after an incremental or transfer learning update, $unknown_{old}$ denote the number of samples with unknown recognition outcome in the network trained only on the old data and and *total samples* denote the number of samples in the old dataset, then complete forgetting is calculated according to

$$complete\ forgetting = \frac{unknown_{new} - unknown_{old}}{total\ samples} \tag{5-6}$$

Catastrophic forgetting is defined in this chapter because it was essential for the reader to know about concepts like recognition outcomes, incremental and transfer learning, etc prior to understanding how catastrophic forgetting applies to this work.

5-5 Summary

In this chapter, we discussed the entire workflow of the back-end networks. We started with road sign extraction problem where the road signs are cropped from the image using a colour filter. These cropped road signs form our basic dataset and the input to the pre-processing pipeline. Next, we discussed the pre-processing pipeline which had a separate pre-processing approach for each of the back-end networks. Pre-processing pipeline was used to remove all the irrelevant features from the image and prepare it for training. These pre-processed images were converted to a 1D vector of 8-bit unsigned integers so that they can be processed by NeuroShield. Next, we discussed the learning and classification scenarios of all the back-end networks. We saw how iterative learning helped model a better decision shape and avoid forgetting. We also discussed in detail the classification logic used in case of a *Recognition* status and *Unknown* status and how we use both RBF and KNN classifiers respectively. The classification algorithm was shown in Figure 5-7. Lastly, we discussed two types of catastrophic forgetting which were used to analyse the how much of the previous knowledge the network had partially or completely forgotten after learning new data.

Chapter 6

Front-end Network - Taxonomy

In this chapter, the scientific aspect of this thesis is discussed. The custom feature vector and the final network is discussed in detail where we mainly focus on explaining how we aim to control the clustering. We have formulated our problem mathematically and proved the necessary equations. In Section 6-1, we describe our proposed clustering approach with an example of a general 3-level taxonomy. We also formulate the problem mathematically and show with the same example the expected output of the modelled decision space. In Section 6-2 we derive the necessary parameters which are specific to our system. We also prove the distance criterion formulated in Section 6-1 for our system i.e. NeuroShield. In Section 6-3, we discuss the learning and classification phase for the front-end network followed by a simple example. Section 6-4 concludes the chapter with a short summary.

6-1 Proposed Clustering Approach

We concluded the previous chapter with the classification phase of the back-end networks. Figure 5-7 has another component `Custom Feature Vector Generation` which will be discussed in this chapter. The objective is to represent an entire taxonomy in the form of a trainable feature vector. Hence, we propose a feature vector structure which is virtually divided into segments, where each segment represents one level of the taxonomy and every element in that segment represents a classification outcome of a feature of that corresponding taxonomy level. The proposed general structure of the custom feature vector is shown in (6-1).

$$\left[\underbrace{x_1, x_2, ..., x_{m_l}}_{l}, \underbrace{x_{m_l+1}, x_{m_l+2}, ..., x_{m_l+m_{l-1}}}_{l-1}, ..., \underbrace{x_{m_1+...+m_{l-(z-1)}+1}, ..., x_{m_{l-(z-1)}+m_{l-z}}}_{l-z}\right] \quad (6\text{-}1)$$

where l and $(l-z)$ correspond to the highest and lowest level of the taxonomy respectively and $z \in \mathbb{N}$, x_n is any element in the feature vector and $n \in [1, m_l + m_{l-1} + ... + m_{l-z}]$, $m_l, m_{l-1}, ..., m_{l-z}$ are the total number of features in the corresponding taxonomy level.

For better understanding of the equations in the remainder of the chapter we represent Definition (6-1) as follows:

$$\left[\underbrace{x_{l,1}, x_{l,2}, ..., x_{l,m_l}}_{l}, \underbrace{x_{l-1,1}, x_{l-1,2}, ..., x_{l-1,m_{l-1}}}_{l-1}, ..., \underbrace{x_{l-z,1}, ..., x_{l-z,m_{l-z}}}_{l-z}\right] \tag{6-2}$$

where $x_{l,1} = x_1$, $x_{l-1,1} = x_{ml+1}$ and so on. For simplicity, we will denote Definition (6-1) as follows:

$$\left[\underbrace{\phi_l}_{l}, \underbrace{\phi_{l-1}}_{l-1}, ..., \underbrace{\phi_{l-z}}_{l-z}\right] \tag{6-3}$$

where:

- $\phi_l = x_1, x_2, ..., x_{m_l}$
- $\phi_{l-1} = x_{m_l+1}, x_{m_l+2}, ..., x_{m_l+m_{l-1}}$
- $\phi_{l-z} = x_{m_{l-(z-1)}+1}, x_{m_{l-(z-1)}+2}, ..., x_{m_{l-(z-1)}+m_{l-z}}$

Definition (6-1) can be understood from Figure 6-1. It must be noted that $(l - z)$ from Definition (6-1) has been replaced by $(l - 2)$ in the figure since there are only 3 levels in the taxonomy.

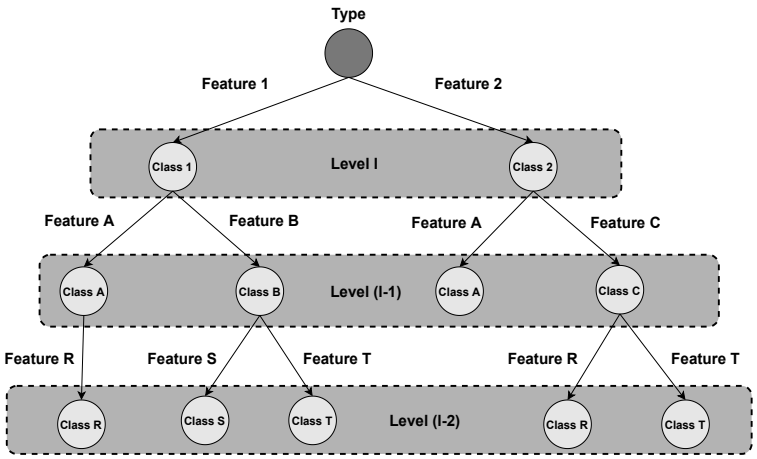

Figure 6-1: A general taxonomy structure indicating different levels and taxonomical features. The light grey circles represent classes and each dotted rectangle represents a separate level of the taxonomy. The dark grey circle with a label 'Type' is the overall type of the taxonomy (for example, road signs). The arrows represent taxonomical features.

Figure 6-1 shows a three-level taxonomy with various classes at each level. For the above mentioned taxonomy, the general custom feature vector shown in Equation (6-1) will look like:

$$\left[\underbrace{x_1, x_2}_{l}, \underbrace{x_3, x_4, x_5}_{l-1}, \underbrace{x_6, x_7, x_8}_{l-2}\right] \tag{6-4}$$

Each element in the feature vector corresponds to a recognition outcome of a feature at respective taxonomy levels.

Recognition outcome is the output of the back-end networks. As mentioned earlier in this chapter, each network can have 3 possible outcomes for each feature namely recognition, uncertain and non-recognition. In this work, we will be focusing on clustering by distance, particularly using the *Manhattan distance*[1]. In order to compute this distance, these recognition outcomes of an element are represented as a numerical value, which will be discussed later in this chapter. Let p be the input feature vector and q be a feature vector learnt by a neuron, then the Manhattan distance (shown in (6-5)) between two vectors is defined as the sum of absolute differences of their elements [50].

$$d(p,q) = ||p-q||_1 = \sum_{i=1}^{n} |p_n - q_n| \tag{6-5}$$

where p_i and q_i are the i^{th} input features and learnt features of a unit respectively. The clustering algorithm will use the *Manhattan distance metric* to differentiate between different classes. A simple example of how we want the clustering to happen for the taxonomy is shown in Figure 6-2. As can be seen in the figure, a cluster of neurons that have learnt a different feature at the highest level (level l) are the furthest apart. The distance between subsequent lower level clusters keep decreasing within the recognised class at the higher level.

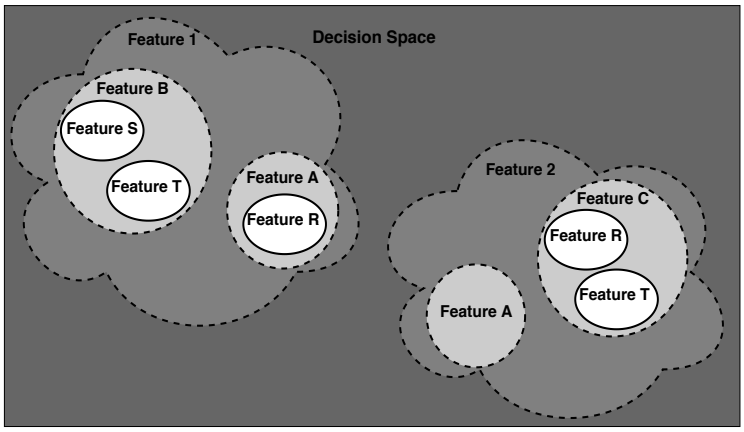

Figure 6-2: The expected output of our proposed distance based clustering approach. The two dotted cloud-like structures represent clusters of 2 taxonomical features at level l, dotted circles represent clusters of taxonomical features at level $(l-1)$ and the solid-lined ellipses represent clusters of taxonomical features at level $(l-2)$. Note that the dotted-lined figures represent virtual clusters while the solid-lined ellipses represent the final actual clusters.

It can be inferred that taxonomical features at higher levels have more dominance/weight than those at lower levels and hence they have a higher impact on clustering. This distance is what we will refer to as *intra-class* distance in the remainder of this document. It must however be noted that the above logic will work only in case of a correct recognition. To get

[1]It is also referred to as taxicab distance or l_1 norm

a better understanding of the proposed clustering approach, we will use a couple of examples later in this chapter.

Everything mentioned until now in this chapter has been under the assumption that if one taxonomical feature is recognized, all others are not recognised. But we have mentioned earlier that we have a third recognition outcome which is *uncertain*. There can be various possible combinations for these recognition outcomes. For this work, we will consider 2 cases which are:

1) One taxonomical feature per level is *recognized* and all the others are not recognised: This is the ideal case where only 1 feature is recognized at each level and the remaining are not recognized. An example of the expected clustering output for level l is shown in Figure 6-3. It can be observed that the two clusters representing this case (dark coloured, solid-lined, cloud-like structures) are far apart and are in accordance with the expected output shown in Figure 6-2. A similar clustering trend shall also be observed for other levels of the taxonomy. Before formulating this scenario mathematically, let us first understand a few terms that will be used in the remainder of the chapter. Best Matching Unit (BMU) is a neuron with a learned feature vector that are recognised at the corresponding level when an input sample is presented for classification. It must be noted that no BMU has neurons with multiple recognition outputs or uncertain outputs. A set of BMUs in the neural network is represented by Q and each BMU within the set is represented by q.

A general BMU for the highest level (l^{th} level) can be represented as shown in (6-6). This represents a set of all BMUs in the decision space and can be considered as a universal set.

$$Q_l = \{\phi_l, \phi_{l-1}, ..., \phi_{l-z} | X_{l,i} = \{r\}, X_{l,j} = \{nr\}, i \neq j\} \tag{6-6}$$

where $X_{l,i}$ is a singleton set i.e. it contains only one element with a value corresponding to a *recognised* outcome, $X_{l,j}$ is a set of all elements with a value corresponding to a *not recognised* outcome, i is 1 element from $[1, m_l]$ and j is $\{1, ..., m_l\} \setminus \{i\}$. The "\" operator indicates a set subtraction.

In this case, we have one recognised outcome at each level. Sets of BMUs for level l can be represented as follows:

$$Q_{l,1} = \{\phi_l, \phi_{l-1}, ..., \phi_{l-z} | X_{l,1} = \{r\}, X_{l,j} = \{nr\}, Q_{l,1} \subset Q_l\} \tag{6-7a}$$

$$Q_{l,2} = \{\phi_l, \phi_{l-1}, ..., \phi_{l-z} | X_{l,2} = \{r\}, X_{l,j} = \{nr\}, Q_{l,2} \subset Q_l\} \tag{6-7b}$$

$$\vdots$$

$$Q_{l,m_l} = \{\phi_l, \phi_{l-1}, ..., \phi_{l-z} | X_{l,m_l} = \{r\}, X_{l,j} = \{nr\}, Q_{l,m_l} \subset Q_l\} \tag{6-7c}$$

For lower levels, the BMU is dependent on all the levels above it i.e. based on which taxonomical feature is recognised at the higher level, the BMUs will differ. The general definition for BMU for level $l-1$ is:

$$Q_{(l-1),i|l,k} = \{\phi_l, \phi_{l-1}, ..., \phi_{l-z} | X_{l,k} = \{r\}, X_{l-1,i} = \{r\}, X_{l-1,j} = \{nr\}, Q_{l-1,1|l,1} \subset Q_l\} \tag{6-8}$$

where $Q_{(l-1),i|l,k}$ represents a BMU whose parent refers to $x_{l,i} = r$, $i \neq j$, $k \neq j$ and k is 1 element in $[1, m_l]$. k is the value of i corresponding to the BMU of the parent level i.e. $Q_{l,i}$. Variable k has been used here to avoid confusion. Therefore, $Q_{l-1,i|l,k} \subset Q_{l,i}$.

The following set of equations represent sets of BMUs for level $l-1$.

$$Q_{l-1,1|l,1} = \{\phi_l, ..., \phi_{l-z})|X_{l,1} = \{r\}, X_{l-1,1} = \{r\}, X_{l-1,j} = \{nr\}, Q_{l-1,1|l,1} \subset Q_l\} \quad (6\text{-}9a)$$

$$Q_{l-1,1|l,2} = \{\phi_l, ..., \phi_{l-z})|X_{l,2} = \{r\}, X_{l-1,1} = \{r\}, X_{l-1,j} = \{nr\}, Q_{l-1,1|l,2} \subset Q_l\} \quad (6\text{-}9b)$$

$$\vdots$$

$$Q_{l-1,1|l,m_l} = \{\phi_l, ..., \phi_{l-z})|X_{l,m_l} = \{r\}, X_{l-1,m_{l-1}} = \{r\}, X_{l-1,j} = \{nr\}, Q_{l-1,1|l,m_l} \subset Q_l\} \quad (6\text{-}9c)$$

A similar set of equations can represent BMUs for all subsequent lower levels and hence we will not mention them here. As mentioned earlier, we want the higher levels to dominate the clustering and hence we introduced a term *intra-class* distance. Let's assume that $q_{b,l}$ in $Q_{l,2}$ fired when an input feature vector p_r was presented. Then the distance between p_r and any $q_{a,l} \in Q_{l,i}$ $(i \neq 2)$ should be greater than the distance between p_r and any $q_{b,l} \in Q_{l,2}$

$$|p_{r,l} - q_{a,l}| > |p_{r,l} - q_{b,l}| \quad (6\text{-}10)$$

where $p_{r,l}$ correspond to few elements (belonging to level l) of p_r and $p_r = (p_{r,l}, p_{r,l-1}, ..., p_{r,l-z})$.

Similarly, for any $q_{b,l-1} \in Q_{l-1,2|l,2}$ (this implies that feature number 2 is recognized both levels l and $l-1$) and any $q_{a,l-1} \in Q_{l-1,i|l,i}$, condition (6-10) can be rewritten for level $l-1$ as follows:

$$|p_{r,l-1} - q_{a,l-1}| > |p_{r,l-1} - q_{b,l-1}| \quad (6\text{-}11)$$

2) One taxonomical feature per level is *uncertain* and all the others are *not recognised*: This is a case where the network is in a state of uncertainty. It gives an uncertain recognition outcome for one taxonomical feature as the sample lies closer to that taxonomical feature but isn't close enough to be recognized. The other taxonomical features are not recognized because they are further apart than the uncertain feature. Hence, the presented sample must lie closer to the uncertain feature. Figure 6-3 depicts this scenario where the two clusters representing this case (light coloured, dotted-lined, cloud-like structures) are closer to their respective uncertain classes than the non-recognized classes. It must be noted that none of the BMUs will have an uncertain outcome value as they are learnt feature vectors with different recognised classes at each level.

Now, we have input feature vectors $p_{u,l}$ where we have 1 uncertain outcome at level l and all other non-recognised output. The distance criterion similar to (6-10) is shown in (6-12).

$$|p_{u,l} - q_{a,l}| > |p_{u,l} - q_{b,l}| \quad (6\text{-}12)$$

for any $q_{b,l} \in Q_{l,2}$ and any $q_{a,l} \in Q_{l,i}$

For the 2 cases discussed above, one could argue that the number of possible cases based on the classification outcomes must be more than 2 since all combinations of these classification outcomes can represent a different scenario. In section 5-3-2, we have shown that there will be no cases where a feature is recognized and there is an uncertainty in another. The decision logic used, forces the network to come to a conclusion where it removes the uncertainty and classifies it as a non-recognition outcome. Also, when there is a case of multiple feature

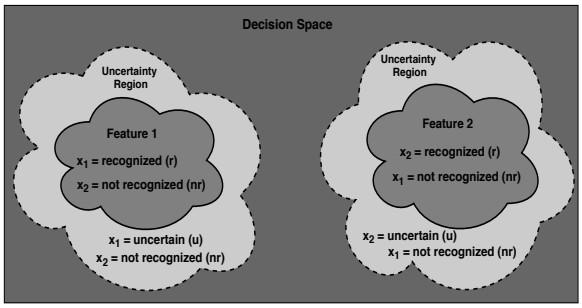

Figure 6-3: The expected output of n-dimensional feature space for our proposed distance based clustering approach for the case 1 and case 2. The two solid-lined cloud-like structures represent the case when one class is recognised and all others are not recognized while the two dotted cloud-like structures represent the case when one of the classification outcomes is uncertain and the rest are not recognised.

recognition, the neural network makes a decision as to which is the most appropriate feature based on confidence (Also mentioned in section 5-3-2).

We have so far seen how BMUs are defined at each level. A final classification outcome is based on the combination of BMUs at each level. The final condition for clustering is shown in Equation (6-13).

$$|p_r - q_{a,l}| > |p_{r,l} - q_{b,l}| + |p_{r,l-1} - q_{b,l-1}| + ... + |p_{r,l-z} - q_{b,l-z}| \tag{6-13}$$

where $p_{r,l}$, $p_{r,l-1}$, $p_{r,l-z}$ are a few elements of p_r indicating elements of corresponding levels and $p_r = (p_{r,l}, p_{r,l-1}, ..., p_{r,l-z})$.

This will ensure that if a taxonomical feature is recognized at a higher level, then the corresponding lower taxonomical features will be closer to the recognised higher level taxonomical feature than the lower taxonomical features corresponding to the non-recognised higher level taxonomical features. This applies for subsequent layers as well.

We will now explain our clustering approach with two examples and show that Equation (6-13) is correct. Later, we will also prove Equation (6-13). Let our taxonomy be as shown in Figure 6-1 and the corresponding feature vector structure be as represented in Equation (6-4).

Example 1: We have a custom feature vector generated from the outputs of the back-end networks which with respect to Equation (6-4) is

$$[\underbrace{r, nr}_{l}, \underbrace{u, nr, nr}_{l-1}, \underbrace{r, nr, nr}_{l-2}] \tag{6-14}$$

This corresponds to case 1 for level l and $l-2$ i.e. Feature 1 and Feature R are recognised and case 2 for level $l-1$ i.e. Feature A is uncertain. Since we have an uncertainty at level $(l-1)$, initially the network comprehends it as not belonging to any feature at this level. According to the proposed approach, we must be able to ensure that it always lies closest to

the correctly recognised taxonomical features. In this case, the sample will be placed within the Feature 1 cloud with it being closer to Feature A cluster than Feature B cluster as shown in Figure 6-4. In terms of Equation (6-13), we can say that the distance of the sample from all neurons with Feature 2 at level l is more than all neurons with Feature 1 at level l. This will ensure that when a higher level feature is recognised, the closest firing neurons are always the ones with the same higher level feature.

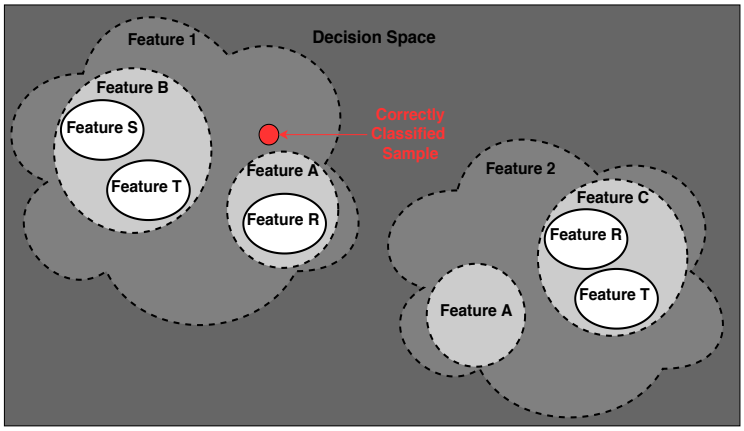

Figure 6-4: The expected output of the network for the test input vector (red dot). Level l has the highest influence on clustering and hence the uncertainty on level $(l-1)$ does not affect the decision at level l. The uncertainty at level $(l-1)$ is observable as the sample neither lies within Feature A cluster nor within Feature B cluster. The sample is however placed closer to Feature A than B because of an uncertain outcome at level $l-1$ and also because Feature R is recognised at level $(l-2)$. But since the influence of level $(l-1)$ on clustering is more than level $(l-2)$, the sample lies outside class A in spite of being recognised as Feature R.

From the figure, we can conclude that the classification is correct and is in accordance with Equation (6-13).

Example 2: Now we have a custom feature vector generated from the outputs of the back-end networks which with respect to Equation (6-4) is

$$[\underbrace{nr, r}_{l}, \underbrace{nr, r, nr}_{l-1}, \underbrace{nr, r, nr}_{l-2}] \tag{6-15}$$

This corresponds to case 1 where only one feature per level is recognised (i.e Feature 2, Feature B and Feature S) and all others are not recognised. The expected classification output is shown in Figure 6-5. One could argue for the above example that the classification outcome of level l could be incorrect and that of the lower 2 levels could be correct. But for this work, we are not concerned with the accuracy of the feature extraction network. The classification accuracy for feature extraction networks is shown to be decent and we will assume that such errors will rarely occur. However, if such errors do occur, it shall lead to a mis-classification. Figure 6-6 shows the classification outcome in the case when Equation (6-13) is violated. This is clearly incorrect as according to the output, the sample does not belong to any feature at any level in the taxonomy and hence the network will make a decision to introduce a new feature; which is unnecessary. Now, based on the classification outcome shown in Figure 6-5,

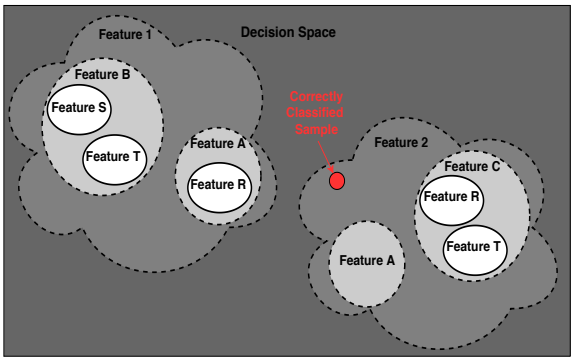

Figure 6-5: The expected output of the network for the input vector. Level l has the highest influence on clustering and hence the recognition outcome on level $(l-1)$ and level $(l-2)$ does not affect the decision at level l. The sample is placed almost at the border of Feature 2 cluster because the lower 2 classes are closer to Feature B and Feature S clusters which are part of Feature 1 cluster. But since the influence of level (l) on clustering is more than level $(l-1)$ and $(l-2)$ combined, the sample lies within Feature 2 cluster in spite of being recognised as Feature B and Feature S respectively at lower levels.

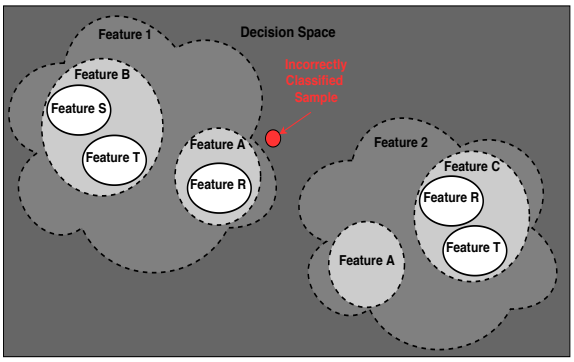

Figure 6-6: The test sample is incorrectly classified here. This shows the output of the network when Equation (6-13) is violated. On violation of equation (6-13), level l will have a lower influence on clustering than level $(l-1)$ and $(l-2)$ combined. Hence, in spite of being recognised as Feature 2, the sample is placed outside the cluster. This is because, the sum of distances of the sample from Feature 2 is more than its distance from Feature B and Feature S.

we can conclude that the sample belongs to Feature 2 and two new features need to be added namely Feature B and Feature S. The updated taxonomy and decision space are shown in Figure 6-7 and Figure 6-8 respectively.

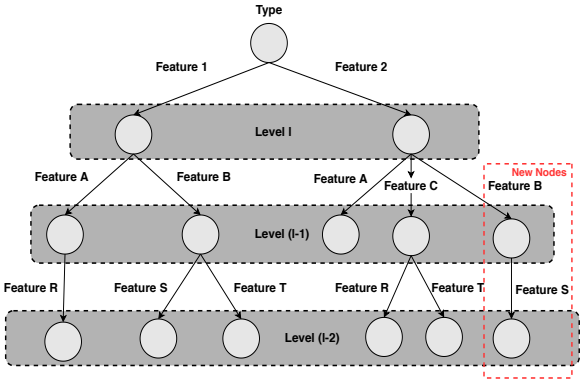

Figure 6-7: The updated taxonomy as per the classification outcome shown in Figure 6-5. New nodes Feature B and Feature S have been introduced at level $l-1$ and $l-2$ respectively.

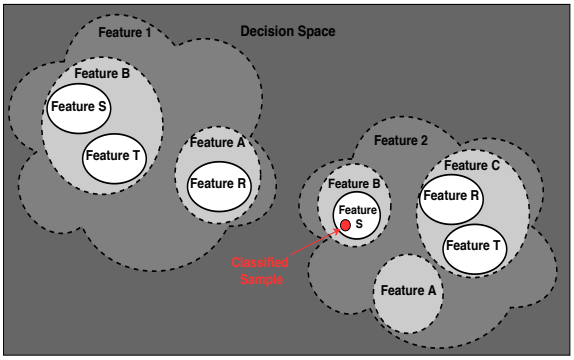

Figure 6-8: The updated decision space as per the classification outcome shown in Figure 6-5. New nodes Feature B and Feature S have been introduced at level $l-1$ and $l-2$ respectively.

Using Example 1 and Example 2, we have shown how Equation (6-13) affects and controls

the clustering. We have also shown that Equation (6-13) is a necessary condition to assure the network clusters the samples in accordance with our requirement and when new features and classes should be added.

6-2 System Specific parameters and constraints

NeuroShield has certain constraints on the data that can be fed into it. As our approach depends on a customized feature vector, we present a few parameters in this section which can be adapted to any such hardware or even software neural networks. NeuroShield only accepts 8-bit unsigned integer data type and each feature vector is limited to a maximum length of 256 elements with each element being any 8-bit unsigned integer i.e. the range of the space is [0,255]. Following are representations for several parameters which will be used throughout the remainder of this document.

- Let the maximum value of range of the feature space be denoted by $R = 255$.
- Let total levels in the taxonomy be denoted by L.
- Let the set denoting classification outcomes be $C = \{r, u, nr\}$.
- Let the weight for any level of the taxonomy be denoted by W_L. Weight is a factor by which the values of recognition, non-recognition and uncertainty decreases with decreasing taxonomy levels.

Here, R is taken into account since the size of a feature element is limited on the NeuroShield. To the best of our knowledge, there is no neuromorphic chip which offers indefinite memory. Most neuromorphic chips are limited to 8-bit or 16-bit data types. In case this range is not known, one can suitably assume since the taxonomy is known in advance. For this work, we have fixed the general values of r, u and nr to 0,1 and 2 respectively. The simple reason behind using these values is that *multiplying* them with decreasing weights will decrease the difference between two classification outcomes at different levels, for instance, difference between non-recognition and recognition outcomes for level l must be more than that for level $l-1$. These are the base values for these classification outcomes. These values are multiplied to the corresponding weights (W_L) to obtain the true values for each level of the taxonomy. Hence, using fixed values for recognition outcomes and dynamic weights for different levels of the taxonomy, we can ensure unique and decreasing values for the classification outcomes for every level of the taxonomy. One would argue that the value of recognition is 0 and multiplying it by any weight would always result in 0. This exactly the case and will be better understood as one progresses through this section. The values for these recognition outcomes can be seen in Table 6-1.

Now, since the maximum value of an element in the feature vector NeuroShield can accept is 255 and we want to utilise the entire range space, the weight for the highest level must be $R/2$. So then the highest value for a non recognition outcome is $2 \times R/2 = R = 255$. We know that the custom feature vector is virtually divided into number of segments equal to the number of levels in the taxonomy. So, the objective is to divide a limited range space into many virtual segments by ensuring that every successive segment is smaller than the

previous one. Hence we perform successive division with exponents of 2 where the exponent of 2 represents the level of the taxonomy. Let our taxonomy have $l-z$ levels. Then the weight (W_{l-z}) is represented mathematically as:

$$\lfloor W_{l-z} \rfloor = \frac{R}{2 \times 2^z} \tag{6-16}$$

where $\lfloor W_{l-z} \rfloor$ represents the floor function defined as the greatest integer less than or equal to W_{l-z}. From here on, W_{l-z} will indicate the floor function value. In definition (6-16), a division by two is made to ensure that we have decreasing weights for each decreasing level. Now, each recognition outcome value can be calculated as follows:

$$r_{l-z} = r \times W_{l-z} \tag{6-17a}$$

$$u_{l-z} = u \times W_{l-z} \tag{6-17b}$$

$$nr_{l-z} = nr \times W_{l-z} \tag{6-17c}$$

Substituting $z = 0$ in Equations (6-16), (6-17a), (6-17b) and (6-17c) we have,

$$\begin{aligned} W_l &= \frac{255}{2} = 127.5 = 127 \\ r_l &= 0 \times W_l = 0 \\ u_l &= 1 \times W_l = 1 \times 127 = 127 \\ nr_l &= 2 \times W_l = 2 \times 127 = 154 \end{aligned} \tag{6-18}$$

Similarly, the values of the above parameters for different values of z (i.e. different levels of the taxonomy) can be obtained. Table 6-1 shows the values of these parameters for all the four levels of our taxonomy.

Table 6-1: Parameter values for different levels of the taxonomy

z	W_{l-z}	r_{l-z}	u_{l-z}	nr_{l-z}
0	127	0	127	254
1	63	0	63	126
2	31	0	31	62
3	15	0	15	30

Substituting the values for each outcome in Equation (6-14) and Equation (6-15), the final custom feature vectors will respectively be as follows:

$$\begin{aligned} &[\underbrace{0, 254}_{l}, \underbrace{127, 126, 126}_{l-1}, \underbrace{0, 62, 62}_{l-2}] \\ &[\underbrace{254, 0}_{l}, \underbrace{126, 0, 126}_{l-1}, \underbrace{62, 0, 62}_{l-2}] \end{aligned} \tag{6-19}$$

Looking at these vectors, one might argue that there will be many taxonomical features at the same distance from some taxonomical features. When visualised in 2D, this is indeed the case. But the feature space is not 2D and each element in the feature vector represents a

separate dimension in the feature space. The feature vectors mentioned in Equation (6-19) are 8-dimensional vectors in the feature space.

We will now prove that condition (6-13) is always satisfied with the choice of our system parameters. Let us consider the extreme case. Suppose we have the following:

- input feature vector $p_{r,L} = \{\underbrace{r, nr, ..., nr}_{l}, \underbrace{r, nr, ..., nr}_{l-1}, ..., \underbrace{nr, r, ..., nr}_{l-z}\}$
- BMU $q_{b,L} = \{\underbrace{r, nr, ..., nr}_{l}, \underbrace{nr, nr, ..., r}_{l-1}, ..., \underbrace{r, nr, ..., nr}_{l-z}\}$
- closest non firing neuron $q_{a,L} = \{\underbrace{nr, r, ..., nr}_{l}, \underbrace{r, nr, ..., nr}_{l-1}, ..., \underbrace{nr, r, ..., nr}_{l-z}\}$

It must be noted that the BMU and the input feature vector only have same feature recognition at level l while the closest non firing neuron and the input feature vector have same feature recognition at all other levels. This is an extreme case and according to the proposed method, the input feature vector must still lie closer to $q_{b,L}$ than $q_{a,L}$. Left Hand Side (L.H.S) of condition (6-13) is:

$$\begin{aligned} \text{L.H.S} &= |p_{r,l} - q_{a,l}| \\ &= |r_l - nr_l| + |nr_l - r_l| + 0... + 0 \\ &= 2nr_l \qquad\qquad \because r_l = 0 \end{aligned}$$

Substituting nr_{l-z} from (6-17c) with z = 0

$$\text{L.H.S} = 2nrW_l \tag{6-20}$$

The Right Hand Side (R.H.S) of condition (6-13) is:

$$\begin{aligned} \text{R.H.S} &= |p_{r,l} - q_{b,l}| + |p_{r,l-1} - q_{b,l-1}| + ... + |p_{r,l-z} - q_{b,l-z}| \\ &= \underbrace{(0)}_{l} + \underbrace{(|r_{l-1} - nr_{l-1}| + 0 + ... + |nr_{l-1} - r_{l-1}|)}_{l-1} + ... + \underbrace{(|r_{l-z} - nr_{l-z}| + |nr_{l-z} - r_{l-z}| + ... + 0)}_{l-z} \\ &= 0 + 2nr_{l-1} + ... + 2nr_{l-z} \qquad \because r_L = 0 \end{aligned}$$

Substituting nr_{l-z} from (6-17b):

$$\text{R.H.S} = 0 + 2nrW_{l-1} + ... + 2nrW_{l-z}$$

Rewriting in terms of W_L from (6-16)

$$\begin{aligned} \text{R.H.S} &= \frac{2nr}{2}\left[W_l + \frac{W_l}{2^1} + ... + \frac{W_l}{2^{z-1}}\right] \\ &= nr(2W_l) \end{aligned}$$

The sum $\left[W_l + \frac{W_l}{2^1} + ... + \frac{W_l}{2^{z-1}}\right] = 2W_l$ for infinite z. The proof is shown in Appendix A-1. But in our case, the number of levels in a taxonomy can never be infinite and hence z is finite. Therefore,

$$\text{R.H.S} < nr(2W_l) \tag{6-21}$$

From equations (6-20) and (6-21), L.H.S > R.H.S. We can thus say that condition (6-13) has been proven and is a mandatory condition to control the clustering. In the following sets of equations, the values for weights and all recognition outcomes are calculated for level l.

6-3 Front-End Network

The final network of the end-to-end framework shown in Figure 4-1 is the front-end network. This network takes in as input the custom feature vectors generated by the output of back-end networks and outputs a final category of the taxonomy. In short, this network learns the taxonomy and classifies the custom feature vectors into respective categories.

6-3-1 Learning

The learning phase is almost identical to the back-end networks. Instead of an image, this network takes as input the custom feature vector. A Radial Basis Function (RBF) model generator is used for learning with a context value different from all the back-end networks. Learning is done in an iterative manner to avoid forgetting any learnt samples. The Active Influence Field (AIF) is tuned automatically and the Minimum Influence Field (MINIF) is set to the default minimum value of 2. The Maximum Influence Field (MAXIF) parameter plays an important role in tuning the AIF and hence is not randomly set. It is better to apply a certain threshold on MAXIF. This is particularly useful in cases when there is very less data for a particular class i.e. very few neurons are committed for one class as compared to the others. These neurons will generally have a very high MAXIF and will hence over generalise.

Figure 6-9 and Figure 6-10 show a possible decision space when no threshold is applied on MAXIF and when a threshold is applied on MAXIF respectively. The yellow colour class in Figure 6-9 has very few samples and hence commits only one neuron. Since there is no threshold on MAXIF, it is automatically set using the formula mentioned in Equation (3-1). Hence, in spite of the center being away from remaining classes, its AIF extends till the center of the closest neuron (blue in Figure 6-9). It can also be observed that there are a lot of overlapping influence fields which depict uncertainty between corresponding neurons. Applying a threshold reduces areas of uncertainty as can be seen in Figure 6-10. Also due to a smaller MAXIF, the yellow class is modelled with 2 neurons instead of one which are comparatively conservative. This happens when a new example belonging to the yellow class is presented, and it does not fall in the influence field of the existing neurons containing information about the yellow class. Since a very small percentage of examples belonging to yellow class were used to build the decision space, it is not enough to recognise all samples. So that neuron did not fire and the network finds a need to commit a new neuron and improve the decision space. Had the MAXIF been higher, then this neuron would have fired and the presented sample would have been recognised as already part of the decision space.

We want to avoid over generalising the decision space and so the MAXIF is set to a value less than the difference between the non-recognition and uncertain outcome value of the lowest level of the taxonomy. Doing so has 3 major advantages:

- Prevents over-generalisation of neurons leading to a better decision space as shown in Figure 6-10.
- No neuron fires in the RBF mode when the input sample only differs at the lowest level feature from any neuron in the decision space i.e. if we have an input sample corresponding to a new category where all taxonomical features at higher levels match with an existing neuron except the lowest level, no neuron must fire as it does not belong to any existing category.
- When new neurons are committed for new categories, none of the existing neurons will shrink their AIFs since we avoid confusion by setting a threshold on the MAXIF. We know that if an input sample does not belong to any existing category, the minimum distance of this input sample from any neuron will be equal to the value of an uncertain outcome at the lowest level. Hence according to Definition (3-1), the AIF will always be set to m (MAXIF) which in our case is always less than this minimum distance d_n.

Learning stops when no more neurons are committed in the following epoch, which is the stopping criterion.

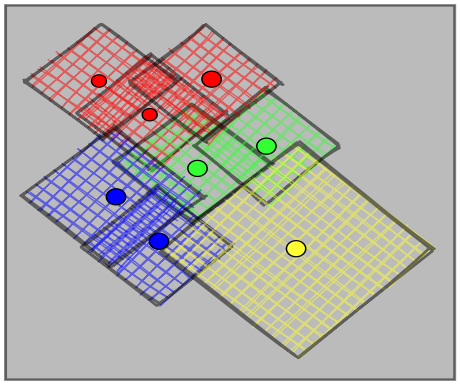

Figure 6-9: A decision space with no threshold on MAXIF.

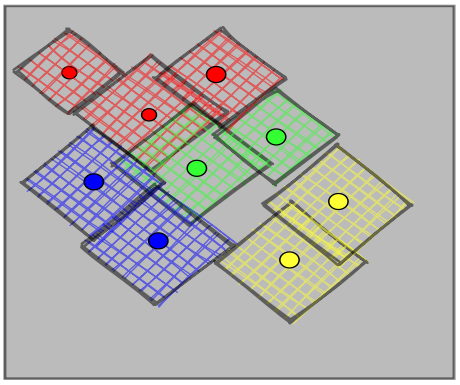

Figure 6-10: A decision space with a threshold on MAXIF.

6-3-2 Classification

Classification approach is exactly the same as that for back-end networks (Section 5-3-2) which is shown in the classification segment of Figure 5-7. The only difference is the output of the network. The front-end network outputs the category and not individual features since a category has information about all the levels of the taxonomy. The virtual features are defined as labels for these higher classes. It is important to note that the back-end networks do not have any information about the taxonomy and each of them classify input vectors independently. Since none of the back-end networks classify with a 100 % accuracy, there will be cases when each of these networks recognise certain features but the resulting combination

might not be in line with the taxonomy. For instance, the three back-end networks recognise Feature 1, Feature B and Feature R respectively. But according to the taxonomy shown in Figure 6-1, no such combination of features exist. There are two possible reasons as to why such a situation can occur.

1. Incorrect Classification: When testing the network with a dataset without any new classes, we know before hand that if the input vector does not align with the taxonomy, one or more of the back-end networks have made a classification error. But since these networks have no information about the taxonomy, the generated custom feature vector will be sent to the front-end network for final classification. Since the custom feature vector does not align with the taxonomy, the front-end network must ideally not recognise it and classify it as *unknown* but still place it accurately based on partially correct information. To check which back-end network made an incorrect classification, we came up with a fault detection technique.

Since recognition outcome values for every level is 0, we know with certainty that the distance of the input vector to the BMU must be 0. If p_r is the input vector and $q_{b,L}$ is the BMU and $L \in \{l, l-1, ..., l-z\}$ then, if taxonomical features at all levels are recognised we will have:

$$|p_{r,l} - q_{b,l}| + |p_{r,l-1} - q_{b,l-1}| + ... + |p_{r,l-z} - q_{b,l-z}| = 0 \tag{6-22}$$

where each one of $|p_{r,L} - q_{b,L}|$ is equal to 0. Whichever term amongst these is not equal to 0, the corresponding level is where an incorrect classification occurred. If Equation (6-22) is satisfied, we can assure that none of the back-end networks produced an incorrect classification. This method is label independent and will hence be very effective in cases where we have non-labelled test data. This will also handle cases where the test data was labelled incorrectly. An example to understand the condition mentioned in Equation (6-22) is as follows:

- Let the the generated feature vector be $[\underbrace{0, 254}_{l}, \underbrace{126, 0, 126}_{l-1}, \underbrace{0, 62, 62}_{l-2}]$
- Let the BMU be $[\underbrace{0, 254}_{l}, \underbrace{126, 0, 126}_{l-1}, \underbrace{62, 0, 62}_{l-2}]$ or $[\underbrace{0, 254}_{l}, \underbrace{126, 0, 126}_{l-1}, \underbrace{62, 62, 0}_{l-2}]$.

Then the value of Equation (6-22) for this generated vector would be 126 for both the BMUs. These values correctly indicate that there is an incorrect classification occurring in level $l-2$ of the taxonomy.

2. New Classes: The input feature vector represents a new category which was not learnt during the learning phase. Such a case will appear during an incremental learning phase. When new categories (i.e unknown combinations of the known taxonomical features) are presented to the network for classification, the back-end networks shall ideally identify them as either uncertain or recognised since these taxonomical features are known. The final network will give an uncertain output since this is a new category. In such a case, it will place this input vector according to our classification logic. Based on the classification outcome, a decision is made as to whether a new category needs to be introduced or if it can be clubbed with one of the existing ones. Example 2 discussed previously in this chapter discusses the case when a new class is introduced in the taxonomy and is also shown in Figure 6-7. The network incrementally updates the taxonomy online and there is no need to re-learn any

previous knowledge. However, it must be noted that such an incremental update is only possible if the taxonomical features are known to the back-end networks. If there are new features, a transfer learning framework is used to learn these new classes in the back-end networks. The taxonomy is updated by writing new element(s) to the existing neurons of the front-end network. This entire concept is explained in Section 4-1.

6-4 Summary

In this chapter, we discussed about the custom feature vector and the front-end network. The custom feature vector is the input to the front-end network which classifies it based on the learnt taxonomy. We discussed in detail the proposed clustering approach which involved mathematically formulating the custom feature vector, visual representation and explanation of the expected clustering output. Next, we introduced and derived system specific parameters based on some assumptions and constraints which were needed to implement our proposed approach on the NeuroShield. Using these parameters, we proved that the intra-class distance condition mentioned in Equation (6-13) holds for our system in all the cases. Next, we saw how learning and classification phase takes place in the front-end network and the importance of iterative learning to build a better decision space. Lastly, we introduced a simple fault detection condition to check which level of the back-end networks made a classification error in case there was one.

Chapter 7

Results

7-1 Evaluation Metrics

- **Learning Curve:** The learning curve reports how many neurons are committed as input patterns are learned [51]. An asymptotic curve reflects that the neurons can model the training set while over-generalizing and converging.
- **Neuron Distribution Histogram:** It shows how many neurons were required to model a certain taxonomical feature and how good the generalisation across all taxonomical features is.
- **Confusion Matrix:** It gives a summary of prediction results on a classification problem [52]. It is one of several methods to evaluate the performance of a neural network. For a binary classification, a confusion matrix can have the following outcomes (referring to Table 7-1):
  - **True Positive (TP):** Input belonging to class 1 is classified as class 1
  - **True Negative (TN):** Input belonging to class 0 is classified as class 0
  - **False Positive (FP):** Input belonging to class 0 is classified as class 1
  - **False Negative (FN):** Input belonging to class 1 is classified as class 0

  Using these outcomes, different metrics can be used to compare different solutions. The most intuitive metric is *accuracy*, according to

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{7-1}$$

  Having an accuracy of 1 means that all inputs are correctly classified, while an accuracy of 0 means that all inputs are misclassified. This metric is very easy to understand, but it also can be very deceiving. For example, the number of inputs belonging to each class might not be balanced. E.g. if there are 500 samples of class 1 and 5 of class 0,

then classifying every input as class 1 yields and accuracy of $\frac{500+0}{500+0+0+5} = 99\%$, while the network is effectively doing nothing useful. Furthermore, the accuracy metric assumes that each outcome has the same weight. That is, misclassifying class 1 carries the same consequences as misclassifying class 0, which is not always the case. Another popular metric is the *F1-score*. The F1-score is more robust against class imbalance and is calculated using precision and recall, according to

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{7-2}$$

where:

$$precision = \frac{TP}{TP + FP} \tag{7-3}$$

$$recall = \frac{TP}{TP + FN} \tag{7-4}$$

Table 7-1 shows an example of a confusion matrix.

Table 7-1: Sample Confusion Matrix

Prediction \ Target	Class 1	Class 0
Class 1	True Positive	False Positive
Class 0	False Negative	True Negative

7-2 Back-end Networks

In this section we evaluate the performance of the back-end networks. The performances of each of these networks is evaluated the metrics mentioned in the previous section.

7-2-1 Shape Feature Extraction Network

In this section, the performance of the shape feature extraction network is discussed. Figure 7-1 shows the learning curve for this network. It can be observed that only 3 epochs were required to model the decision space. Most neurons are committed during the first epoch since the decision space is empty. The non-linear model generator in NM500 is known to generalise very quickly and this can be seen in Figure 7-1. As examples are learnt, some neurons reduce their Active Influence Field (AIF) in order to incorporate more neurons. Shrinking of neurons can result in forgetting which is why we perform iterative learning. More neurons are committed in the second and third epochs to re-learn the forgotten knowledge.

Figure 7-3 shows the confusion matrix after classifying the test dataset. The colour gradient in the colour bar indicates the distribution of samples per taxonomical feature. It must be noted that we have a multi-class classification in this case which still does not change the fundamental definitions of the outcomes in the confusion matrix. For example, the outcomes for *Circle* with respect to confusion matrix shown in Figure 7-3 are:

- TP is $(1, 1)$
- TN is $(2, 2) + (2, 3) + (3, 2) + (3, 3)$
- FP is $(1, 2) + (1, 3)$
- FN is $(2, 1) + (3, 1)$

These outcomes can similarly be calculated for all remaining features. The shape network achieves an accuracy of 99.82%, an F1-score of 0.9977 and a recognition time of 174.49ms per sample. An observation that can be made from the confusion matrix is that both the incorrect classifications were made for the *Square* shape for which we had minimum training samples and hence less committed neurons (see Figure 7-2).

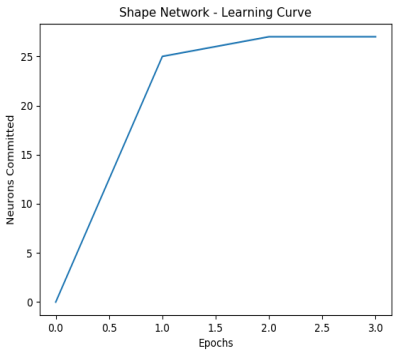

Figure 7-1: Learning Curve for the shape feature extraction network

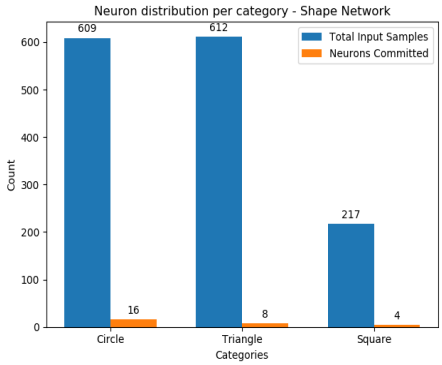

Figure 7-2: Neuron Distribution Histogram for shape feature extraction network

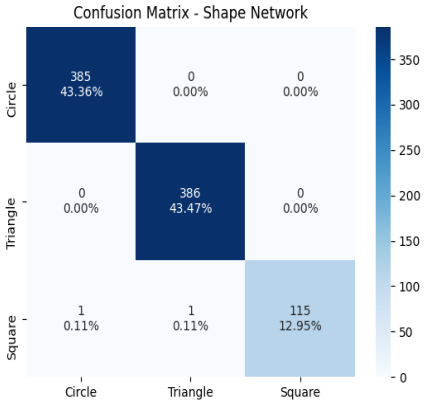

Figure 7-3: Confusion Matrix for the shape feature extraction network.

7-2-2 Colour Feature Extraction Network

In this section, the performance of the colour feature extraction network is discussed. As mentioned earlier, the feature vector that is learnt here consists of the mean values of blue,

green and red channels of the image respectively. Hence, only 3 elements form the feature vector for 1 image. Since we only learn 2 colours which are themselves standard colours (Red and Blue), it is fairly easy for the network to discriminate between these features. The results come with no surprise as the mean values for the three channels for a red road sign is completely different than that of a blue road sign. The network generalises in the first epoch itself (Figure 7-4) as no neuron is committed in the second epoch and requires only 3 neurons to model the decision space (Figure 7-5). The colour network achieved an accuracy of 100%, an F1-Score of 1 and a recognition time of 4.3ms per sample. Recognition time is significantly lower in this case as it also depends on the length of the input feature vector which is 3 in this case. While for all other back-end networks the feature vector length is 255. Hence, the recognition time corresponding to these networks will be the upper limit as we can not have a feature vector longer than 255.

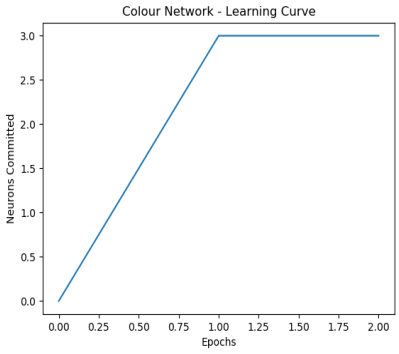

Figure 7-4: Learning Curve for the colour feature extraction network

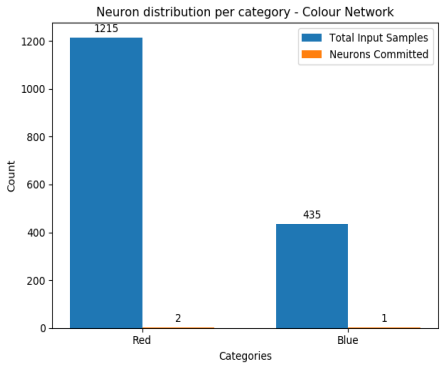

Figure 7-5: Neuron Distribution Histogram for colour feature extraction network

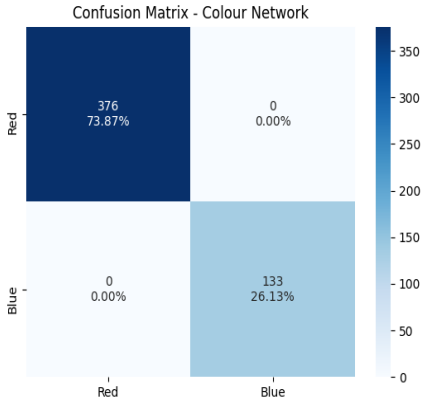

Figure 7-6: Confusion Matrix for the colour feature extraction network

7-2-3 Descriptive Feature Extraction Network

In this section, the performance of the descriptive feature extraction network is discussed. The network converged in 3 iterations (Figure 7-7) and required 53 neurons to model the decision

space Figure 7-8. The dataset is not even as there are way too many samples corresponding to classes *Picture* and *Arrow*. But as can be observed in Figure 7-8, more neurons were required to model those features which is self explanatory.

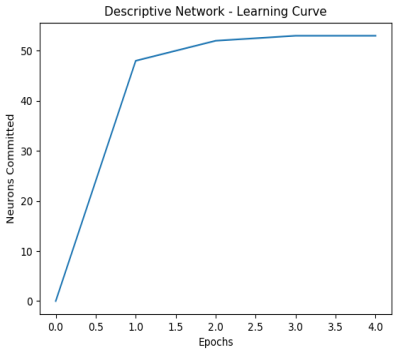

Figure 7-7: Learning Curve for the descriptive feature extraction network

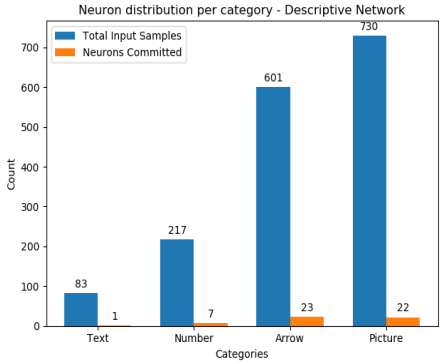

Figure 7-8: Neuron Distribution Histogram for descriptive feature extraction network

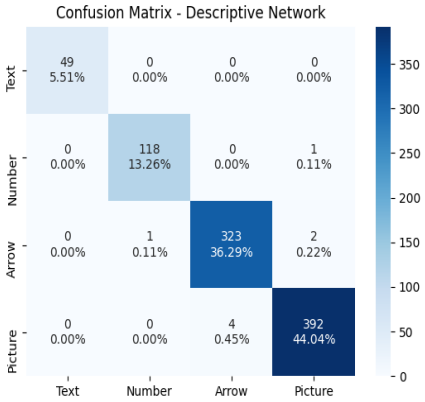

Figure 7-9: Confusion Matrix for the descriptive feature extraction network

From the confusion matrix shown in Figure 7-9, maximum confusion is observed between *Picture* and *Arrow*. Although there are more samples learnt for both these features, the neurons seemed to have over-generalised resulting in large AIF for multiple neurons. The decision logic used has shown to be very effective in resolving such confusion but this is a case where multiple neurons have similar overlaps i.e. multiple neurons corresponding to these classes have overlaps in the decision space. This would cause the sample to be further inside the incorrect neuron's influence field resulting in an incorrect decision and hence an incorrect classification. Since it was not possible to visualise the decision space modelled by the NeuroShield, Figure 7-10 is used to visualise the above statement. Let the yellow coloured "X" mark be the input sample corresponding to the red feature and it is assumed that it is placed as shown in Figure 7-10. In this case, the decision making logic which is based on the confidence will lead to the sample being classified as a blue feature instead of red. This is because, the confidence metric checks how much inside the sample is from the boundary of the neuron's influence field. In this case, the sample is almost halfway inside the blue

neuron's influence field while being almost at the boundary of the red neuron's influence field. In our experiments, such a case was rarely observed and we believe that such a situation can be avoided by further tuning the modelling parameters, especially the Maximum Influence Field (MAXIF) parameter. The descriptive network achieved an overall accuracy of 99.27%, an F1-Score of 0.991 and a recognition time of 174.83ms per sample

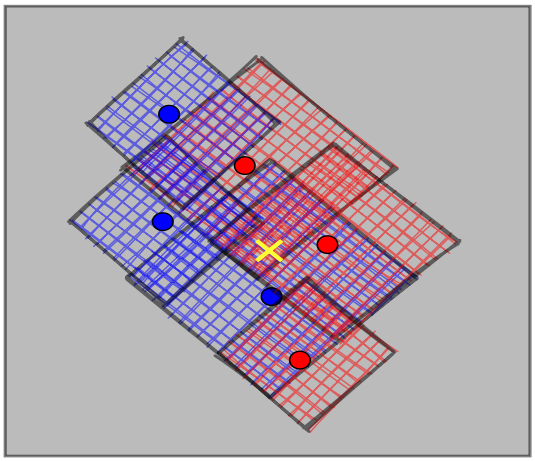

Figure 7-10: An example of a decision space with too many overlaps/uncertainty. The colours represent distinct features, circles represent committed neurons, the coloured shapes represent the influence fields of each neuron and the yellow cross is an input sample.

7-2-4 Final Level Feature Extraction Network

In this section, the performance of the final level feature extraction network is discussed. The model generator converges in 2 epochs Figure 7-11 while committing 37 neurons (Figure 7-12). The categories are labeled from 1 to 14 in Figure 7-12 for better visualisation, but they are actually similar to the ones shown in Figure 7-13 where 1 = 50, 3 = Car and so on. There is some uneven distribution of neurons observed in Figure 7-12 as more neurons are required for feature 1 and feature 2 than feature 5,7 and 12 in spite of having less training samples. This is not surprising as commitment of neurons depends on how different the training samples are and how large is the influence field of the committed neuron. A neuron with a large influence field will consider many samples as similar to the learnt sample while a neuron with smaller influence field will not recognise many different samples. This network achieved an overall accuracy of 99.84%, F1-score of 0.989 and a recognition time of 174.1ms per sample.

It must be noted that the recognition times obtained in this work are quite high from NM500's standards. The claimed recognition times are in micro-seconds. The reason for quite high recognition times in our case is because we communicate with the NeuroShield via a raspberry pi which creates a delay. If data is directly written to the registers of the NeuroShield using the software provided by nepes, we could achieve recognition times in the range of microseconds.

These results were obtained on an artificial dataset and hence these numbers are quite high. Although some noise was introduced in these images and also different illuminations were used to try and replicate a real dataset, these numbers are expected to decrease on a real dataset. We don't expect the accuracy and F1-score to reduce much but model generator will

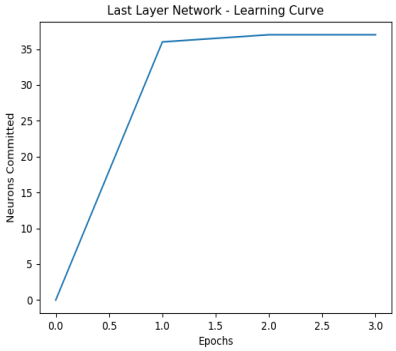

Figure 7-11: Learning Curve for the final level feature extraction network

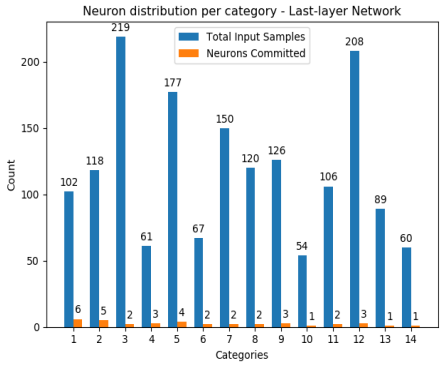

Figure 7-12: Neuron Distribution Histogram for final feature extraction network

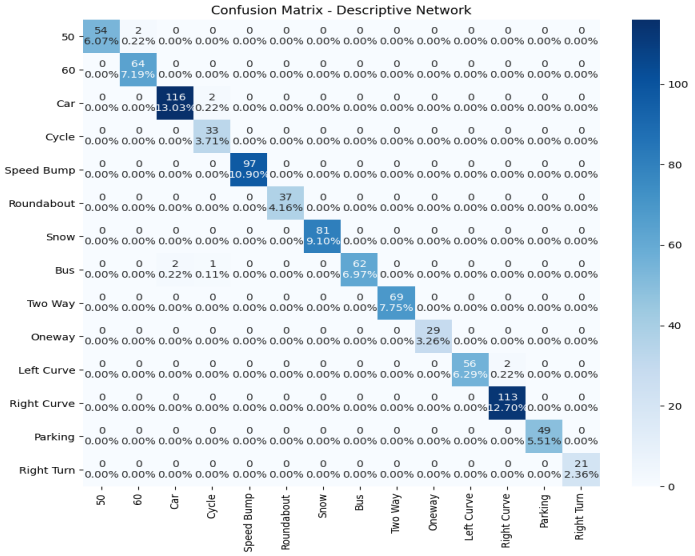

Figure 7-13: Confusion Matrix for the final level feature extraction network

definitely commit more neurons to model the decision space. The focus of this thesis is not on these back-end networks and the pre-processing pipeline. We have just used some available resources and used this evaluation as a proof of concept.

7-2-5 Front-end Network

In this section, the performance of the front-end network is discussed.

Figure 7-14 and Figure 7-15 show the decision spaces with different randomness. A decision space modelled by the NM500 could not be plotted as the software required to visualise the decision space is not open source. Hence, these plots were generated using the Somoclu library which is a parallel implementation of self-organizing maps [53]. The objective of these plots is to show that the proposed custom feature vector structure indeed incorporates the

entire taxonomy information and the neurons with different taxonomical features recognised at higher levels are further apart from one another than the ones having same features at the highest levels but different ones at lower levels. Table 7-2 show the actual categories of the numbers shown in both the figures.

Table 7-2: Categories

Neuron ID	True Categories
1	Circle → Red →Number → 50
2	Circle → Red → Picture → Car
3	Triangle → Red → Picture → Speed Bump
4	Triangle → Red → Arrow → Roundabout
5	Circle → Blue → Picture → Bus
6	Square → Blue → Arrow → Two Way
7	Circle → Blue → Arrow → Drive Ahead
8	Square → Blue → Text → Parking

The objective of the custom feature vector was to incorporate information of the entire taxonomy. In Figure 7-14, we can clearly see that neurons with different higher level features are further apart than the the ones having same higher level taxonomical features but common lower level features. Neurons 1,2,5,7 are are separated from 3,4 and 6,7 by a colour corresponding to a higher value in the colour bar as they have different taxonomical features at the highest level i.e. level l. Neurons 1,2 and 3,4 are separated with a colour corresponding to a lower value in the colour bar as they have the same taxonomical feature recognised at the highest level but have different colour features. A distinction at the descriptive level i.e. level $l-2$ can also be observed between neurons 1 and 2 and also between neurons 3 and 4. It is observable that the value of the colour separating neurons decreases as the taxonomy level decreases. Similar observations can also be made for neurons 6 and 8.

Unfortunately, information about the last level of the taxonomy could not be captured in these plots as somoclu does not consider such low distances between neurons and the user has no control over parameters to control the distance threshold. Somoclu also does not let the user control the randomness parameter and hence different plots are obtained every time. Figure 7-15 is included just for assurance that irrespective of the shape of the decision space, the taxonomy is always retained. An important thing to observe in Figure 7-15 is that neurons 1,2 and 5,7 are at different ends. The decision space in Self Organizing Map (SOM)s can cross to the other border meaning that neurons 5,7 and neurons 1,2 belong to the same region. We believe these plots provide sufficient proof to support our hypothesis and will hold true for all levels if we have control over distance threshold parameters. Through these results we have also shown that our method works on different clustering methods based on the Manhattan distance.

In Figure 6-4, we showed how we would want the decision space to be modelled if the back-end networks output an uncertain outcome for a taxonomical feature at any level $l-1$ of the taxonomy. Figure 7-16 proves that the assumed output of the decision space is correct and samples with an uncertain outcome for a certain taxonomical feature are indeed placed closer to the neuron recognising that taxonomical feature in the decision space.

Two input feature vectors were presented to the network with uncertain outcome at level

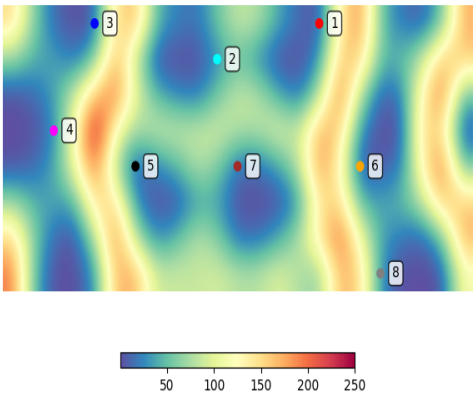

Figure 7-14: Decision Space 1

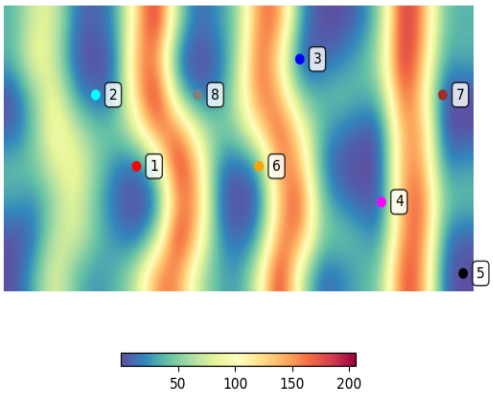

Figure 7-15: Decision Space 2

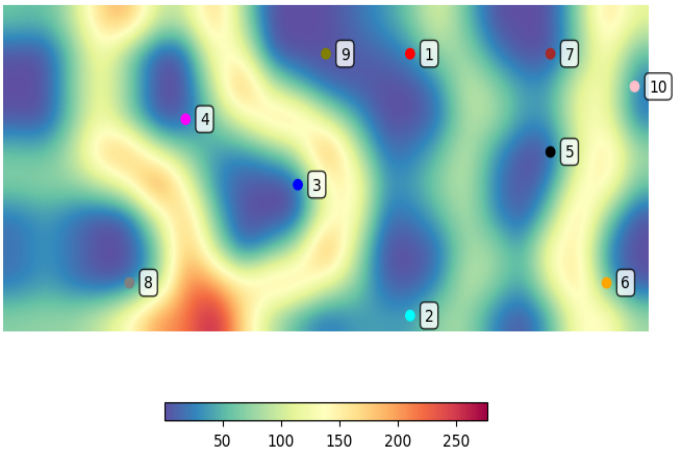

Figure 7-16: Decision space modelling using an uncertain sample 9

l (sample 10) and at level $l-3$ (sample 9). Sample 9 is a road sign with speed limit 40. So all back-end networks recognise the corresponding features except the last network which has only learnt speed sings with numbers 50 and 60. Hence it gives an uncertain output for taxonomical feature 50 (since the back-end network sets 40 closest to a firing neuron with 50 as a recognised taxonomical feature). It can be seen in Figure 7-16 that sample 9 separated from 1 by a colour with a very low value in the colour bar which is exactly as expected. Sample 10 has an uncertain outcome for square shape and has recognised outcomes for blue colour, text and number 50 for the lower levels. It can be observed that sample 10 is separated from neuron 6 and 8 (both correspond to square shape recognition outcomes) by a colour with lower colour bar value than neurons recognising other shapes.

The above results show that the proposed approach works correctly and the network can classify unknown categories quite well using the information from the recognised taxonomical features at various levels of the taxonomy.

NeuroShield Results

The results shown above provide visual proof for the claims made in Chapter 6. Since visual results for the decision space modelled by the NeuroShield could not be produced, we present some quantitative results. The categories mentioned in Table 7-2 do not include all the classes we learn. Table 7-3 shows all the 14 categories that were learnt during the training phase. When samples 9 and 10 were presented to the network for classification, the 6 closest neurons from the position of the samples in the decision space are shown in Table 7-4 and Table 7-5 respectively.

Table 7-3: All Categories Learnt

Neuron ID	True Categories
1	Circle → Red →Number → 50
2	Circle → Red →Number → 60
3	Circle → Red → Picture → Car
4	Circle → Red → Picture → Cycle
5	Triangle → Red → Picture → Speed Bump
6	Triangle → Red → Arrow → Roundabout
7	Triangle → Red → Picture → Snow and Ice
8	Circle → Blue → Picture → Bus
9	Square → Blue → Arrow → Two Way
10	Circle → Blue → Arrow → Oneway
11	Triangle → Red → Arrow → Left Curve
12	Triangle → Red → Arrow → Right Curve
13	Square → Blue → Text → Parking
14	Circle → Blue → Arrow → Right Turn

As can be seen in Table 7-4, sample 9 (speed limit 40) is placed closest to category 1 and then category 2. Both of these are speed signs but the feature extraction output gave an uncertain outcome for 50 and a non-recognition outcome for 60 and hence it is placed closer to 50 than 60. It must be noted that the 5^{th} and 6^{th} closest neurons correspond to blue circles and not any red road sign. This also shows that the taxonomical feature at higher levels have more dominance than the lower level ones. A similar observation can be made for the closest neurons for sample 10 shown in Table 7-5. The 2 closest neurons correspond to square shape recognition at the highest level. 3^{rd}, 4^{th} and 5^{th} neurons correspond to different shapes with blue colour since sample 10 also has blue colour recognised at the second level. Noteworthy are the 6^{th} and 7^{th} neurons which recognise a number at the third level and so does sample 10. Neuron with category 1 is closer to sample 10 than the one with category 2 because category 1 corresponds to 50 at the last level while category 2 corresponds to 60.

These quantitative results from the NeuroShield also show how the taxonomy is taken into account while classification. This also shows that our proposed custom feature vector indeed

Table 7-4: 6 closest neurons for sample 9

Closest Neurons		
No.	Distance	Category
1	15	1
2	45	2
3	169	3
4	169	4
5	421	8
6	421	10

Table 7-5: 7 closest neurons for sample 10

Closest Neurons		
No.	Distance	Category
1	311	9
2	311	13
3	565	8
4	565	10
5	565	14
6	633	1
7	693	2

incorporates information about all levels of the taxonomy accurately.

7-3 Online Incremental Learning

The results shown in this section correspond to the online incremental learning case explained in Section 4-1. We present a small dataset of a new road sign namely sample 11 (Square → Blue → Number → 50) to the network and each of the back-end networks successfully classify this since the features are already known to them. The generated custom feature vector is sent as an input to the front-end network and no neurons recognise this sample since we have modelled a slightly conservative decision space. Since the sample is not recognised, a neuron is committed with a new category. Since each of these committed neurons represent an entire chain in the taxonomy, committing a new neuron is equivalent to adding a new chain to the taxonomy. In this manner, the taxonomy can be incrementally updated without affecting the previous knowledge i.e. no catastrophic forgetting occurred. Figure 7-17 and Figure 7-18 show the decision space before and after updating the taxonomy respectively.

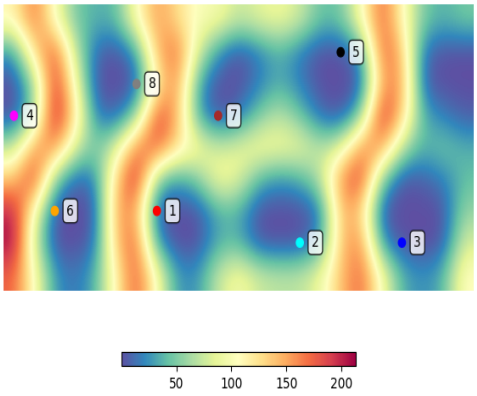

Figure 7-17: Decision space before the incremental update

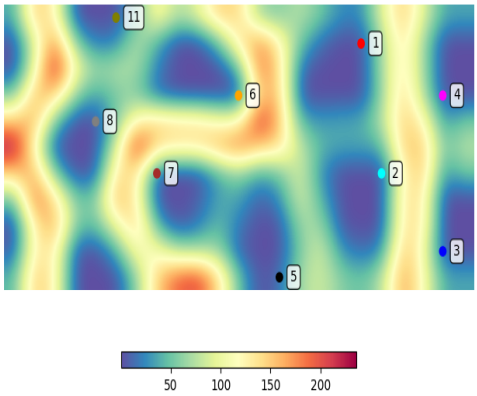

Figure 7-18: Decision space after incrementally updating the taxonomy

It can be observed in Figure 7-18 that a new neuron associated with category 11 has been committed and is correctly placed with neurons recognising square shape i.e. neurons 6 and 8. We can also get a general idea that the influence field of neurons were not affected when comparing Figure 7-17 and Figure 7-18.

7-4 Transfer Learning

In this section we evaluate the performance of the network in a transfer learning framework. It must be noted that this experiment is just to show that transfer learning is indeed possible using the NM500 chip and using the advantages of the built-in non-linear model generator, noteworthy results can be achieved. We fine-tune all back-end networks except the colour feature extraction network since we have no new colour feature in the new dataset. We have 3 completely new road signs for this experiment which are:

- **40 speed limit sign:** 40 is the new feature at the lowest level.
- **Yield sign:** Inverted triangle is a new shape and it has no descriptive features and hence we shall have a new feature section which has a label *featureless*.
- **Road Closed sign:** It is a circular red sign with no descriptive features.

We compare the network performance after fine-tuning with standard learning scenarios where 70% of the new dataset is learnt in the standard learning case while only 5% of the new dataset is learnt and then fine-tuned in the transfer learning case.

It is important to understand the following terms to have a better understanding of the results in the following section.

- **Recognition Accuracy:** It refers to the number of samples for which 1 or multiple neurons fire i.e. it is either recognised or has uncertainties (multiple neurons fire). It is calculated according to

$$Recognition\ Accuracy(\%) = \frac{recognised + uncertain}{totalsamples} \times 100 \tag{7-5}$$

- **Uncertain Classification Accuracy (U_1):** It is the number of uncertain samples correctly labelled using the confidence metric i.e. confusions correctly resolved. It is calculated according to

$$U_1(\%) = \frac{correctly\ labelled\ uncertain}{total\ uncertain} \times 100 \tag{7-6}$$

- **Unknown Classification Accuracy (U_2):** It is the number of unknown samples correctly labelled using the confidence metric. It is calculated according to

$$U_2(\%) = \frac{correctly\ labelled\ unknown}{total\ unknown} \times 100 \tag{7-7}$$

- **Overall Classification Accuracy (O):** It is the overall accuracy which includes all the correctly classified samples i.e. recognised, uncertain and unknown. It is calculated as

$$O = \frac{recognised + correctly\,labelled\,uncertain + correctly\,labelled\,unknown}{total\ samples} \times 100 \tag{7-8}$$

Tables 7-6, 7-7 and 7-8 compare the performance of the shape, descriptive and final level feature extraction networks after standard learning and fine-tuning respectively.

It can be observed that performance of all the networks after fine-tuning is comparable to the case where these networks learn 70% or 60% of the new dataset. This clearly shows that fine-tuning can give equivalent or even better performance in comparison to traditional learning algorithms where a labelled dataset will be required to learn. Fine-tuning can be done online with only a very small dataset learnt offline which is a tremendous advantage. It must also be noted that the number of unknowns are more in the fine-tuning network which will always be the case since it has only been trained on 5% of the new dataset. However, all of the unknown samples were correctly classified leading to a very comparable overall classification accuracy.

Table 7-6: Shape Network Performance

New Learnt Dataset	Total Test Samples	Recognition Accuracy	Total Uncertain	Uncertain Classification Accuracy	Total Unknown	Unknown Classification Accuracy	Overall Classification Accuracy of recognised samples
70%	75	97.33	11	100	2	100	97.33
5% + Fine Tuning	237	97.89	7	85.71	5	100	96.20

Table 7-7: Descriptive Network Performance

New Learnt Dataset	Total Test Images	Recognition Accuracy	Total Uncertain	Uncertain Classification Accuracy	Total Unknown	Unknown Classification Accuracy	Overall Classification Accuracy
60%	60	98.66	5	100	1	100	96.66
5% + Fine Tuning	142	95.07	7	100	7	100	97.88

Table 7-8: Last Layer Network Performance

New Learnt Dataset	Total Test Images	Recognition Accuracy	Total Uncertain	Uncertain Classification Accuracy	Total Unknown	Unknown Classification Accuracy	Overall Classification Accuracy
60%	60	98.33	5	100	1	100	95.0
5% + Fine Tuning	142	92.95	7	100	10	100	96.47

There was no threshold set on the AIF of the feature extraction networks and hence we can expect some forgetting to occur since new neurons have been committed after fine-tuning. Table 7-9 shows the network performance after transfer learning based on how much data was forgotten from the previous dataset and the accuracy. It can be observed that all the

networks have undergone more complete forgetting than partial forgetting. This is because commitment of new neurons causes some already committed neurons to shrink their influence fields. This possibly results in a decrease in overlapping areas while also leaving some empty spaces between the previously committed neurons. These empty spaces result in more samples being not recognised thereby leading to complete forgetting.

Table 7-9: Network performance after transfer learning

Network	Partial Forgetting (%)	Complete Forgetting (%)	Classification Accuracy (%)
Shape	0.7	1.69	99.09
Descriptive	1.797	2.47	98.31
Last Layer	0	1.34	99.43

As mentioned earlier, the numbers shall vary when a real dataset is used. In this experiment, we expect a very minor impact on the classification accuracy of the network after fine-tuning on a real dataset. However, there will definitely be more unknowns initially and the recognition accuracy could be significantly lower. We expect a similar trend to follow on a real dataset where the number of unknowns reduce as the decision space gets modelled. Using these results, we can say that a transfer learning framework can indeed be implemented on the NeuroShield and it has also shown to be at par with traditional learning methods in terms of accuracy. However, it must noted that these methods have a tremendous advantage of learning new features online while traditional learning methods do not.

Although one might argue that in both the cases the network is learning the new dataset but what we try to show here is that fine-tuning required only a small portion of labelled data which in real life scenarios will not be very difficult to get. In fact, samples from another dataset or samples taken from simulation can also be used as labelled data and performance after fine-tuning can be evaluated. Training the network on a small portion of an artificial dataset could potentially be enough to learn new similar features in real time.

As a last step, the updated custom feature vectors are written to the *Component* register of the NeuroMem neurons. Figure 7-19 shows the updated decision space after these new categories are learnt by the front-end network. Neurons 9, 10, 11 represent categories *40 speed limit sign*, *yield sign*, *road closed sign* respectively. It can be observed in the figure that neuron 10 is separated from other neurons having different features at the highest levels by a colour corresponding to a high value in the colour bar. While neurons 9 and 11 are placed closer to neurons corresponding to red circles. Neurons 1 and 9 are separated by a colour with a very low value in the colour bar since they only differ by a feature corresponding to the lowest level of the taxonomy (i.e. 50 and 40 respectively).

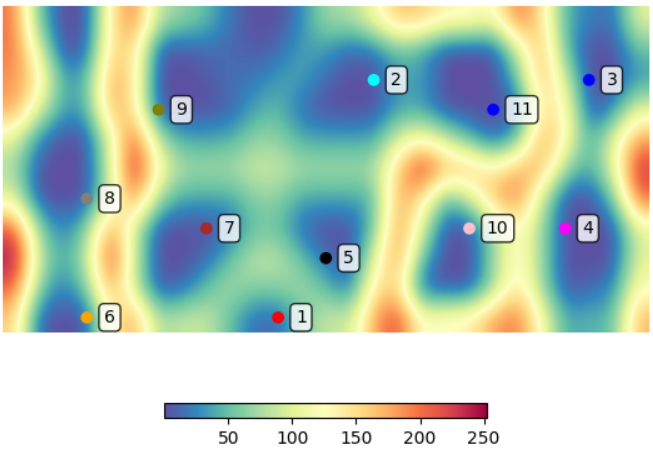

Figure 7-19: Updated decision space with new categories 9,10 and 11

Chapter 8

Conclusion

This conclusive chapter summarises the content presented in this work, and provides answers to the initially posed research questions. Besides, some guidelines for future work are also mentioned.

8-1 Summary

In this work, we propose an end-to-end neural network architecture on the NM500 neuromorphic chip using an incremental hierarchical learning approach. We first design a hierarchical representation of a taxonomy, develop a batch of pre-classifiers and use their output to construct a custom feature vector that is the input to the front-end network which learns the taxonomy. The actual starting point of this work was to exploit a neuromorphic chip and implement one of the many ongoing research algorithms. Having NeuroShield motivated us to look into classification problems as it contains inbuilt classifiers. Analysing state-of-the-art classification methods led us to hierarchical classification and also the road sign classification problem. On board road sign classification is a very critical task and in spite of decades of research, problems regarding decision making time, lack of training data, etc. still exist. Very quick learning and decision making ability of NM500 chip and the fact that road signs can be represented as a taxonomy motivated us to test our framework on this problem.

Some further literature survey led us to a conclusion that not much work has been done on hierarchical classification on neuromorphic chips. The aim now was to exploit NM500 and the Radial Basis Function (RBF) to develop a general end-to-end hierarchical learning framework which could classify taxonomical data. This led to the idea of developing a trainable feature vector which could control modelling the decision space (clustering) based on the taxonomy information. This developed custom feature vector is intended to be capable of representing information about all levels of the taxonomy. Learning one feature vector is equivalent to learning one entire chain in a taxonomy (i.e. from top level to the lowest leaf level).

Considering the drawbacks of many road sign classification algorithms of huge data requirement, inability to learn new features over the existing knowledge, we decided to implement

incremental and transfer learning frameworks and evaluate the performance of the proposed system. Chapter 4 introduces the end-to-end network structure which is primarily divided into 3 segments. Pre-processing pipeline deals with image pre-processing to ensure that the right features are sent to the back-end networks. Back-end networks are feature extraction networks each of which learn dedicated features corresponding to one taxonomy level. The outputs of these networks are transformed into the developed custom feature vector structure which is the input to the front-end network. The front-end network learns these custom feature vectors and models the decision space to represent the entire taxonomy. Chapter 5 discusses in detail the pre-processing pipeline and the back-end networks. In Chapter 6 we discuss the custom feature vector and formulate the problem mathematically. We also prove the hypothesis and provide various examples to validate the correctness of the hypothesis. In Chapter 7, we validate the performance of all the networks in the regular, incremental and transfer learning frameworks. The proposed framework has shown to represent the entire taxonomy and it also models the decision space as per our expectations. The proposed custom feature vector structure works on other Manhattan distance based clustering methods as was shown by plots generated using Somoclu. We were also able to successfully update the taxonomy using both incremental and transfer learning frameworks and lastly, transfer learning framework showed comparable performance to the regular framework in spite of learning a very small portion of the new dataset.

8-2 Answers to Research Questions

The outcome of this work led to the following answers to the initially posed research questions.

Can dedicated hardware such as NM500 neuromorphic chip show comparable or better performance than conventional hardware like CPUs and GPUs?

NM500 chip uses fixed clock cycles for each learning and recognition operation based on the size of the input feature vector. The results have shown that the network can recognise samples within milliseconds which is very low compared to several state-of-the-art existing algorithms. We also know from [32] that recognition times can be much lower if the chip is directly programmed on its registers and that it is very energy efficient. Figure 7-1, Figure 7-4, Figure 7-7 and Figure 7-11 show that the model converges in 2-4 epochs while CNNs trained for road sign classification on CPUs and GPUs require significantly more. Saha et al. show in their work that their deep hierarchical convolutional neural network model which is based on residual convolutional blocks achieves better overall scores than state-of-the-art architectures [54]. Their model required 30 epochs on a 8 Gigabyte Nvidia GeForce GTX 1070 GPU. Another state-of-the-art method Mask R-CNN is used in for the road sign recognition problem where the authors required 95 epochs to train their network on multiple GPUs [55]. Although these have been trained on different datasets with more training samples, we have observed in our work that NM500 reaches above 90% convergence in the first epoch and requires a few more to reach the stopping criterion. We believe this trend shall be observed on any dataset irrespective of the number of images. Hence we can state that NM500 chip shows better performance in terms of fast model convergence and low power consumption and comparable performance in terms of low recognition times to state-of-the-art methods using conventional hardware.

Can a transfer learning framework help improve performance as compared to conventional learning methods?

Transfer learning framework involved learning a very small portion of the new dataset and using this new knowledge and knowledge of the previous dataset to model the decision space for the remainder of the new dataset. The results shown in tables 7-6, 7-7 and 7-8 show that the transfer learning framework achieved comparable performance in terms of accuracy in spite of learning just 5% of the new dataset. Apart from learning this new dataset, it also retains the old knowledge which conventional learning models can not. Results in Table 7-9 show that some catastrophic forgetting was observed in the back-end networks and this was very likely to happen since nothing was done to prevent it. On the contrary, a threshold was applied to the Maximum Influence Field (MAXIF) of the front-end network neurons which led to the network retaining 100% of its previous knowledge. Hence we can state that transfer learning enhanced the existing network performance by learning new categories with minimal loss of previous knowledge while still maintaining comparable classification performance on the old data.

8-3 Future Work

The trainable custom feature vector structure developed in this thesis was accurately able to represent a taxonomy. We were also successfully able to update the taxonomy online and implement a transfer learning framework. However, many ideas and improvements are possible some of which we highlight below.

Although this work focuses on primarily on the front-end networks, these networks are heavily dependent on the accuracy of the back-end networks. In this work, we have used an artificial dataset for testing our framework. It would be interesting to test it on a real dataset. As an addition to this, it would be interesting to test the speed of the chip on a real vehicle and check if it is indeed a potential candidate to replace existing software in Advanced Driver Assistance Systems (ADAS) applications.

An interesting idea would be to be able to realise such a framework without a neuromorphic chip using an RBF and develop a similar modular neural network architecture. A proper comparison could be made with a neuromorphic chip in terms of training time, recognition time, energy consumption, etc.

Appendix A

Proofs

A-1 Proof of infinite geometric progression

Proof. An infinite element geometric progression can be represented as:

$$a + ar + ar^2 + ... + ar^n + ...\infty \tag{A-1}$$

with first term a and common ratio r, where -1 $< r <$ 1 i.e., $|r| < 1$.

The sum of n terms of this Geometric Progression in given by:

$$S_n = a\frac{1-r^n}{1-r} = \frac{a}{1-r} - \frac{ar^n}{1-r}... \tag{A-2}$$

Since - 1$< r <$ 1, therefore r^n decreases as n increases and r^n tends to zero as n tends to infinity i.e., $r^n \to 0$ as $n \to \infty$. Therefore

$$\frac{ar^n}{1-r} \to 0 \quad as \quad n \to \infty$$

Hence from Equation (A-2), the sum of an infinite Geometric Progression ig given by:

$$S = \lim_{x\to 0} S_n = \lim_{x\to\infty}\left(\frac{a}{1-r} - \frac{ar^n}{1-r}\right) = \frac{a}{1-r}, \quad if \ \ |r| < 1 \tag{A-3}$$

According to our terminology $a = W_l$ and $r = \frac{1}{2}$ and $n = l - z$. Substituting these values in Equation (A-3) we have,

$$\frac{W_l}{1-\frac{1}{2}} = 2W_l \tag{A-4}$$

□

Bibliography

[1] Nepes, *CogniMem Reference Guide, Parallel neural network for pattern recognition.* Nepes.

[2] "Classification problems in real life," last accessed on 12 December 2020. https://online.stat.psu.edu/stat508/lesson/1a/1a.5.

[3] O. Chapelle, B. Schölkopf, and A. Zien, "Semi-supervised learning. adaptive computation and machine learning," *MIT Press, Cambridge, MA, USA. Cited in page (s)*, vol. 21, no. 1, p. 2, 2010.

[4] G. Wang, G. Ren, Z. Wu, Y. Zhao, and L. Jiang, "A hierarchical method for traffic sign classification with support vector machines," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–6, IEEE, 2013.

[5] V. Sengar, R. M. Rameshan, and S. Ponkumar, "Hierarchical traffic sign recognition for autonomous driving.," in *ICPRAM*, pp. 308–320, 2020.

[6] K. Aida-Zade, E. Mustafayev, and S. Rustamov, "Comparison of deep learning in neural networks on cpu and gpu-based frameworks," in *2017 IEEE 11th International Conference on Application of Information and Communication Technologies (AICT)*, pp. 1–4, IEEE, 2017.

[7] E. BUBER and D. Banu, "Performance analysis and cpu vs gpu comparison for deep learning," in *2018 6th International Conference on Control Engineering & Information Technology (CEIT)*, pp. 1–6, IEEE, 2018.

[8] Z. Du, D. D. B.-D. Rubin, Y. Chen, L. Hel, T. Chen, L. Zhang, C. Wu, and O. Temam, "Neuromorphic accelerators: A comparison between neuroscience and machine-learning approaches," in *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pp. 494–507, IEEE, 2015.

[9] S. Sardar and K. A. Babu, "Hardware implementation of real-time, high performance, rce-nn based face recognition system," in *2014 27th International Conference on VLSI*

Design and 2014 13th International Conference on Embedded Systems, pp. 174–179, IEEE, 2014.

[10] Y. Cai, Y. Hu, M. Siegel, S. J. Gollapalli, A. R. Venugopal, and U. Bardak, "Onboard feature indexing from satellite lidar images," *IEEE IWADC, Perugia, Italy*, 2003.

[11] M. Suri, V. Parmar, A. Singla, R. Malviya, and S. Nair, "Neuromorphic hardware accelerated adaptive authentication system," in *2015 IEEE Symposium Series on Computational Intelligence*, pp. 1206–1213, IEEE, 2015.

[12] Q. Qiu, Z. Li, K. Ahmed, W. Liu, S. F. Habib, H. H. Li, and M. Hu, "A neuromorphic architecture for context aware text image recognition," *Journal of Signal Processing Systems*, vol. 84, no. 3, pp. 355–369, 2016.

[13] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *Proceedings of the 7th Annual Neuro-inspired Computational Elements Workshop*, pp. 1–8, 2019.

[14] C. N. Silla and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[15] E. Costa, A. Lorena, A. Carvalho, and A. Freitas, "A review of performance evaluation measures for hierarchical classifiers," in *Evaluation Methods for machine Learning II: papers from the AAAI-2007 Workshop*, pp. 1–6, 2007.

[16] A. Sun, E.-P. Lim, W.-K. Ng, and J. Srivastava, "Blocking reduction strategies in hierarchical text classification," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 10, pp. 1305–1308, 2004.

[17] B. C. Paes, A. Plastino, and A. A. Freitas, "Improving local per level hierarchical classification," *Journal of Information and Data Management*, vol. 3, no. 3, pp. 394–394, 2012.

[18] J. Wang, X. Shen, and W. Pan, "On large margin hierarchical classification with multiple paths," *Journal of the American Statistical Association*, vol. 104, no. 487, pp. 1213–1223, 2009.

[19] S. B. Wali, M. A. Hannan, A. Hussain, and S. A. Samad, "Comparative survey on traffic sign detection and recognition: a review," *Przeglad Elektrotechniczny*, vol. 1, no. 12, pp. 40–44, 2015.

[20] M. McCloskey and N. J. Cohen, "The psychology of learning and motivation," 1989.

[21] Z. Li and D. Hoiem, "Learning without forgetting," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 12, pp. 2935–2947, 2017.

[22] A. S. Zacarias and L. A. Alexandre, "Overcoming catastrophic forgetting in convolutional neural networks by selective network augmentation," *CoRR*, 2018.

[23] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[24] "Taxonomy," last accessed on 20 December 2020. https://en.wikipedia.org/wiki/Taxonomy.

[25] "Know your traffic signs, official edition," last accessed on 20 December 2020. https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/519129/know-your-traffic-signs.pdf.

[26] "Transfer learning & fine-tuning," last accessed on 10 January 2021. https://keras.io/guides/transfer_learning/.

[27] "Keras for beginners: Building your first neural network," last accessed on 8 January 2021. https://victorzhou.com/blog/keras-neural-network-tutorial/.

[28] "Tensorflow," last accessed on 8 January 2021. https://www.tensorflow.org/.

[29] "mxnet," last accessed on 9 January 2021. https://mxnet.apache.org/versions/1.7.0/.

[30] "Technology," last accessed on 10 December 2020. https://www.general-vision.com/technology/.

[31] Nepes, *NM500 User's Guide*. Nepes.

[32] Nepes, *NeuroShield User's Guide*. Nepes.

[33] G. Vision, *NeuroMem Technology Reference Guide*. General Vision.

[34] "Hsl and hsv," last accessed on 5 December 2020. https://en.wikipedia.org/wiki/HSL_and_HSV.

[35] H. Fleyeh, "Color detection and segmentation for road and traffic signs," in *IEEE Conference on Cybernetics and Intelligent Systems, 2004.*, vol. 2, pp. 809–814, IEEE, 2004.

[36] H. Supreeth and C. M. Patil, "An approach towards efficient detection and recognition of traffic signs in videos using neural networks," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, pp. 456–459, IEEE, 2016.

[37] "Median filtering with python and opencv," last accessed on 26 November 2020. https://medium.com/@florestony5454/median-filtering-with-python-and-opencv-2bce390be0d1.

[38] "Analyzing image shapes," last accessed on 10 January 2021. https://northstar-www.dartmouth.edu/doc/idl/html_6.2/Analyzing_Image_Shapes.html.

[39] D. Herumurti, K. Uchimura, G. Koutaki, and T. Uemura, "Urban road extraction based-on morphological operations and radon transform on dsm data," *ITE Transactions on Media Technology and Applications*, vol. 2, no. 3, pp. 277–286, 2014.

[40] W. Wanniarachchi, D. Sonnadara, and M. Jayananda, "Detection and extraction of road traffic signs," 2008.

[41] "Morphological transformations," last accessed on 26 November 2020. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html#morphological-gradient.

[42] H. Vishwanathan, D. L. Peters, and J. Z. Zhang, "Traffic sign recognition in autonomous vehicles using edge detection," in *Dynamic Systems and Control Conference*, vol. 58271, p. V001T44A002, American Society of Mechanical Engineers, 2017.

[43] Y. Zhu and Z. Jiang, "A sub-pixel edge detection method based on sobel operator," *Journal of Nanchang Institute of Aeronautical Technology (Natural Science)*, vol. 19, no. 2, pp. 100–102, 2005.

[44] H. Fitriyah, E. R. Widasari, and G. E. Setyawan, "Traffic sign recognition using edge detection and eigen-face: Comparison between with and without color pre-classification based on hue," in *2017 International Conference on Sustainable Information Engineering and Technology (SIET)*, pp. 155–158, IEEE, 2017.

[45] "Sobel operator," last accessed on 4 December 2020. https://en.wikipedia.org/wiki/Sobel_operator.

[46] "Sobel edge detector," last accessed on 4 December 2020. http://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm.

[47] T. Keser and I. Dejanović, "Traffic sign shape detection and classification based on the segment surface occupancy analysis and correlation comparisons," *Tehnički vjesnik*, vol. 25, no. Supplement 1, pp. 23–31, 2018.

[48] M. McCloskey and N. J. Cohen, "Catastrophic interference in connectionist networks: The sequential learning problem," in *Psychology of learning and motivation*, vol. 24, pp. 109–165, Elsevier, 1989.

[49] Z. Chen and B. Liu, "Lifelong machine learning," *Synthesis Lectures on Artificial Intelligence and Machine Learning*, vol. 12, no. 3, pp. 55–75, 2018.

[50] "Taxicab geometry," last accessed on 15 November 2020. https://en.wikipedia.org/wiki/Taxicab_geometry.

[51] G. Vision, *NeuroMem Knowledge Builder*. Nepes.

[52] "What is a confusion matrix in machine learning," last accessed on 9 January 2021. https://machinelearningmastery.com/confusion-matrix-machine-learning/.

[53] "Somoclu - introduction," last accessed on 9 January 2021. https://somoclu.readthedocs.io/en/stable/index.html.

[54] S. Saha, S. A. Kamran, and A. S. Sabbir, "Total recall: Understanding traffic signs using deep hierarchical convolutional neural networks," *arXiv preprint arXiv:1808.10524*, 2018.

[55] D. Tabernik and D. Skočaj, "Deep learning for large-scale traffic-sign detection and recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 4, pp. 1427–1440, 2019.

Glossary

List of Acronyms

RBF	Radial Basis Function
KNN	K-Nearest Neighbors
DAG	Directed Acyclic Graph
ADAS	Advanced Driver Assistance Systems
TSDR	Traffic Sign Detection and Recognition
HSV	Hue, Saturation, Value
IoT	Internet of Things
RGB	Red, Green, Blue
BGR	Blue, Green, Red
MAXIF	Maximum Influence Field
MINIF	Minimum Influence Field
AIF	Active Influence Field
RCE	Restricted Coulomb Energy
BMU	Best Matching Unit
PNN	Progressive Neural Networks
GCR	Global Context Register
LwF	Learning without Forgetting
CNN	Convolutional Neural Network
SNN	Spiking Neural Network
FPGA	Field Programmable Gate Array
L.H.S	Left Hand Side
R.H.S	Right Hand Side
TP	True Positive

TN	True Negative
FP	False Positive
FN	False Negative
SOM	Self Organizing Map
ANN	Artificial Neural Networks