

Parallel Circuit - A Modular Neural Network Architecture



Kien Tuong Phan
School of Computer Science
University of Nottingham

A thesis submitted for the degree of
Doctor of Philosophy

Feb 2018

This thesis is dedicated to
my mother and father

Acknowledgements

First and foremost, I would like to express my deep gratitude to my supervisor, Dr. Tomas Maul, for his continuous guidance throughout my PhD. journey. For years, he has provided countless comments, encouragement, and especially inspiration that motivated me during this long work. Needless to say, this thesis would never have been completed without his support. I would also like to convey my appreciation for Dr. Tuong Thuy Vu, my second supervisor, for his valuable feedback and shared experience.

I would like to acknowledge my family and friends, who have been parts of this chapter of my life. I deeply thank my parents and my little brother for all the unconditional love and support. I treasure all the moments with my amazing lab mates, Moataz, Haixia, Ijaz, Ibrahim, Mahmoud, and Sheena, to name but a few. For these years, I am grateful to have lived under the same roof with many wonderful people, especially Nhat. Finally, I would love to express my gratitude to Huong, for her endless conversation throughout this journey. I would never be able to withstand the pressure without her everyday humour.

Abstract

One of the obstacles that hinder the development of Artificial Neural Networks (ANNs) is the heavy computational cost of the training process. In an attempt to address this problem, I proposed a lightweight model named Parallel Circuits (PCs), with an emphasis on modularity. One of the key inspirations for the proposed model is the human retina, which consists of various cell types that only respond to particular visual stimuli. Similarly, conventional ANNs with high redundancy are decomposed into semi-independent modules, which is deemed to provide more efficient learning, both in terms of speed and generalizability.

Owing to the benefits of having fewer connections, the PC models were empirically shown to be considerably faster, especially when implemented in larger models. I also pursued the ability of automatic problem decomposition, and discovered that diversifying the learning process in each circuit strongly benefits the generalization of the proposed model. PC was shown to be advantageous in term of sparsity, which is highly correlated to modularity.

DropCircuit, a regularizer that targets circuits, was introduced to further enhance their specialities. Together with PCs, DropCircuit outperformed

models with dense connectivity in several experiments. The circuit-level DropCircuit also exhibited better performance compared to conventional DropOut in conjunction with both PC and non-PC configurations, demonstrating the benefits of modularity.

The modularity was further enhanced by imposing a set of biologically inspired constraints. Circuits are modelled as either excitatory or inhibitory types with contrastive properties. Modified PC networks were shown to discover sparse and part-based representations, showing further improvement in generalization.

The following publications were obtained during the research reported in this thesis.

- Phan, K.T., Maul, T.H., & Vu, T.T. (2015) A Parallel Circuit Approach for Improving the Speed and Generalization Properties of Neural Networks. In 11th IEEE International Conference on Natural Computation (ICNC). (pp. 1-7).
- Phan, K.T., Maul, T.H. & Vu, T.T. (2017) An Empirical Study on Improving the Speed and Generalization of Neural Networks Using a Parallel Circuit Approach. *International Journal of Parallel Programming*, 45: 780. <https://doi.org/10.1007/s10766-016-0435-4>
- Phan, K.T., Maul, T.H., Vu T.T. & Lai, W.K. (2016) Improving Neural Network Generalization by Combining Parallel Circuits with Dropout. In: Hirose A., Ozawa S., Doya K., Ikeda K., Lee M., Liu D. (eds) *Neural Information Processing. ICONIP 2016. Lecture Notes in Computer Science*, vol 9949. Springer, Cham
- Phan, K.T., Maul, T.H., Vu, T.T. & Lai, W.K. (2017) DropCircuit : A Modular Regularizer for Parallel Circuit Networks. *Neural Processing Letters*. <https://doi.org/10.1007/s11063-017-9677-4>

Nomenclature

ANN Artificial Neural Network

DL Deep Learning

GPU Graphics Processing Unit

ML Machine Learning

MLP Multi-Layer Perceptron

NMF Non-Negative Matrix Factorization

NNC Non-Negative Constraints

PA Precision Agriculture

PC Parallel Circuit

PCANN Parallel Circuit Artificial Neural Network

PL Penultimate Layer

SC Single Circuit

VC Vegetation Classification

Contents

1	Introduction	1
1.1	Research Background	1
1.1.1	Vegetation Classification	1
1.1.2	Deep Learning	3
1.2	Motivation	4
1.3	Objectives	5
1.4	Research Scopes	6
1.5	Proposed Approach	6
1.5.1	Parallel Circuits	7
1.5.2	DropCircuit	10
1.5.3	Constrained Network	11
1.6	Research Framework	13
1.7	Research Significance	14
1.8	Overview	15
2	Vegetation Classification	16
2.1	Crop Management	16
2.1.1	A Brief History of Crop Management	16

2.1.2	The task of Vegetation Classification	19
2.2	Vegetation Classification	19
2.2.1	Vegetation Classification Based on Remote Sensing	19
2.2.2	Data Characteristic	20
2.2.2.1	Spatial resolution	21
2.2.2.2	Temporal resolution	23
2.2.2.3	Spectral resolution	24
2.2.2.4	Radiometric resolution	25
2.2.3	Current Issues of Vegetation Classification Based on Remote Sensing	25
2.2.4	The Introduction of Machine Learning into Vegetation Classifi- cation	27
2.3	Conclusion	29
3	Comparison of Standard Machine Learning Approaches - A Case Study in Malaysian Region	30
3.1	Related Works	30
3.1.1	Overview	30
3.1.2	Baseline and Reference	32
3.1.3	Artificial Neural Networks	33
3.1.4	Decision Trees	35
3.2	Methodology	38
3.2.1	Dataset Creation	38

3.2.1.1	Data Source	38
3.2.2	Data Sampling	39
3.2.3	Classifier Implementation	42
3.2.4	Validation	43
3.3	Experiments Results	44
3.3.1	Traditional Classifier	44
3.3.2	Neural Network family	45
3.3.3	Decision Tree family	46
3.4	Discussion	48
3.5	Summary	50
4	A Study on Neural Network Design	51
4.1	The Hype of Representation Learning	51
4.2	To Go Deep or Wide?	55
4.2.1	Deep Networks	55
4.2.2	Wide Networks	59
4.2.3	Multibranch Architecture	63
4.3	Conclusion	63
5	Parallel Circuits	65
5.1	Motivations	65
5.2	Parallel Circuits	70
5.2.1	Biological Inspiration	70
5.2.2	Definition	70

5.2.3	Hypotheses	73
5.2.4	Learning Algorithm	74
5.3	Related Works	76
5.4	Proof of Concept	78
5.4.1	Methodology	78
5.4.2	Results	84
5.4.2.1	1000-node Networks	84
5.4.2.2	100-node Networks	85
5.4.3	Discussion	86
5.5	Sparsity Induction	90
5.5.1	Motivation	90
5.5.2	Sparsity Analysis	92
5.5.2.1	Methodology	92
5.5.2.2	Results and Discussion	93
5.5.3	Sparsity Enhancement	95
5.5.3.1	Methodology	95
5.5.3.2	Results and Discussions	97
5.6	Conclusion	98
6	Dropcircuit	100
6.1	Motivation	100
6.2	Related Works	101
6.3	Methodology	104

6.3.1	Node Dropout	104
6.3.2	DropCircuit	104
6.3.3	Experimental Setup	105
6.4	Experimental Results and Discussion	107
6.4.1	Dropout on Parallel Circuits versus on Single Circuits	107
6.4.2	Node Dropout versus DropCircuit	109
6.4.3	Insights into DropCircuit behaviour	110
6.5	Conclusion	115
7	Sparse Parallel Circuits	117
7.1	Motivation	117
7.2	Related Work	118
7.3	Constrained Network	119
7.4	Methodology	120
7.5	Result and Discussions	122
7.6	Conclusion	126
8	Conclusion	127
8.1	Main Contributions	127
8.1.1	Lightweight Parallel Circuits Architecture	127
8.1.2	Diversified Learning with DropCircuit	130
8.1.3	Enhance Sparsity with Constrained Network	131
8.2	Future Work	133
8.2.1	Learning Parallel Circuits via Pruning	133

8.2.2 Extensions of Parallel Circuits	134
8.3 Concluding Remarks	135
A Appendix for Chapter 5	137
B Appendix for Chapter 6	148
C Appendix for Chapter 7	152
Bibliography	152

List of Figures

1.1	Research Framework	13
3.1	Examples of opposing properties	40
3.2	Original image (a) Plant-based ground-truth (b) and Region-based ground-truth (c)	42
4.1	Evolution of Depth and Error in ImageNet Competition	56
4.2	Illustration of (a) ResNet and (b) DenseNet blocks	58
5.1	Illustration of ANN (a) without and (b) with PC architecture. Dis- carded connections are depicted as red dashed lines.	71
5.2	Illustration of (a) SC, (b) RC, (c) PC, and (d) VC designs. The number of nodes and circuits are simplified and do not follow precisely Table 5.1.	80
5.3	Test Accuracy on Leaf Dataset	83
5.4	Comparison of Different Numbers of Circuits on Accuracy (a) and Training Time (b) on the Breast Cancer Dataset.	86
5.5	Comparison of Different Sizes of Circuit on Accuracy (a), Stopping Epoch (b) and Training Time (c) on Breast Cancer Dataset.	88

5.6	Illustration of Single-Node Penultimate Layer Architecture. SNPL is Depicted in Dark Red Colour.	92
6.1	Simplified Diagram of PC with DropOut (The grey nodes have been dropped).	105
6.2	Simplified Diagram of PC with DropCircuit (The grey nodes have been dropped).	106
6.3	Mean test errors for different conditions and datasets (MN-MNIST, LF-Leaf, GL-Glass, BT-Breast Tissue, BC-Breast Cancer, AVG-Average of PCs)	108
6.4	Training and Testing Error across Epochs.	110
6.5	Mean Gradient Magnitude of the First Hidden Layer	111
6.6	Weight Distribution of (a) SC, (b) SCDO, (c) PC, (d) PCDO, and (e) PCDC	113
6.7	Features Learned in the First Layer of (a) SC and (b) PCDC	115
7.1	Training and Validation Error across Epochs	122
7.2	Weight Distribution of (a) PC, (b) PC-L2, (c) PC-C, (d) PC-C-ALL, and (e) PC-NNC	124
7.3	Features Learned in the First Layer of (a) PC-C, (b) PC-C-All, and (c) PC-NNC	125
A.1	Test Accuracy on Breast Cancer Dataset	138
A.2	Test Accuracy on Breast Tissue Dataset	138

A.3	Test Accuracy on Simulation Crash Dataset	139
A.4	Test Accuracy on Glass Dataset	139
B.1	Best (Lowest) Test Error on Breast Cancer Dataset with (a) 100 Node and (b) 1000 Node Network.	149
B.2	Best (Lowest) Test Error over 100 Trials on Breast Tissue Dataset with (a) 100 Node and (b) 1000 Node Network.	150
B.3	Best (Lowest) Test Error over 100 Trials on Glass Dataset with (a) 100 Node and (b) 1000 Node Network.	150
B.4	Best (Lowest) Test Error over 100 Trials on Leaf Dataset with (a) 100 Node and (b) 1000 Node Network.	151
B.5	Best (Lowest) Test Error over 10 Trials on MNIST Dataset with (a) 100 Node and (b) 1000 Node Network.	151
C.1	Weight distribution of PC variant	153

List of Tables

2.1	Spatial resolution ranges and level of recognition of some specific remote sensing materials	22
3.1	Specification of the dataset of each plant	42
3.2	Biased classification of Naive Bayes (bold indicates examples)	45
3.3	Problem of lengthy training time in the ANN family (the longest training duration is indicated in bold)	47
3.4	Comparison of the performance of RF and AB (better accuracy indicated in bold)	48
4.1	Details of ImageNet Winning Models	56
5.1	Network Abbreviations and Designs	79
5.2	Dataset Details	79
5.3	Training and Testing Set Generation Details	79
5.4	Performance of the 1000-Node Family across Five Datasets	83
5.5	Performance of the 100-Node Family across Five Datasets	85
5.6	Sparsity Benchmark on Breast Cancer Dataset (Bold for Better Sparsity).	94
5.7	Sparsity Benchmark on Leaf Dataset (Bold for Better Sparsity). . . .	94

5.8	Sparsity over Variants of PC Count on Breast Cancer Dataset (Higher is Better).	94
6.1	Training times (in seconds) for different network sizes (i.e. 100 and 1000 nodes), conditions (i.e. SCDO, PCDO, and PCDC) and datasets (i.e. BC, BT, GL, LF, and MN-MNIST). Note that each depicted PC condition represents the average of 2-, 5-, and 10-circuit conditions.	107
6.2	Test Errors for Different Network Sizes (i.e. 100 and 1000 nodes), Conditions (i.e. SCDO, PCDO, and PCDC) and Datasets (i.e. BC, BT, GL, LF, and MN). Note that each depicted PC conditions represents the average of 2-, 5-, and 10-circuit conditions. Detailed tables can be found in Appendix B.	109
7.1	Configuration of Constrained Model	121
7.2	Name, Description and Best Test Errors for Different Setups	121
A.1	Performance of 100-Node and 1000-Node Family on Breast Cancer	140
A.2	Performance of 100-Node and 1000-Node Family on Breast Tissue	140
A.3	Performance of 100-Node and 1000-Node Family on Simulation Crash	141
A.4	Performance of 100-Node and 1000-Node Family on Glass	141
A.5	Performance of 100-Node and 1000-Node Family on Leaf	142
A.6	Benchmark of Sparsity Induction on Weight (Fixed Coefficient).	142
A.7	Benchmark of Sparsity Induction on Activation Value (Fixed Coefficient).	143

A.8	Benchmark of Sparsity Induction on Penultimate Layer Activation Value (Fixed Coefficient).	144
A.9	Benchmark of Sparsity Induction on Weight (Varied Coefficient).	145
A.10	Benchmark of Sparsity Induction on Activation Value (Varied Coefficient).	146
A.11	Benchmark of Sparsity Induction on Penultimate Layer Activation Value (Varied Coefficient).	147
B.1	Detailed test errors for different network sizes (i.e. 100 and 1000 nodes), conditions (i.e. SCDO, PCDO and PCDC) and datasets (i.e. BC-Breast Cancer, BT-Breast Tissue, GL-Glass, LF-Leaf, MN-MNIST). Bold for the best.	149

Chapter 1

Introduction

1.1 Research Background

1.1.1 Vegetation Classification

Throughout many years of development, agriculture management has been transforming a great deal, from the basic experience-driven cultivation in the early ages of history to large-scale, uniform treatment during the industrial age. In particular, precision agriculture (PA) is considered as the most significant advancement in the past decades, which shifted the whole sector into site-specific management [37]. Decisions are taken based on the observations of multiple field variables, at much finer scales. Aside from crop-yield-related measurements, numerous analyses are now within the reach of agriculture practitioners, yielding them richer information to deal with unresolved problems of the past (e.g. nutrition value, environmental safety practice).

The task of classification various vegetation species (i.e. vegetation classification (VC)) plays an important role in a wide range of applications [144]. In today's site-specific management, it supports the monitoring of negative phenomena such as invasive species (e.g. weeds), soil degradation (i.e. the decline in soil condition), or desertification (i.e. the phenomenon in which fertile land losing its body of water,

becoming desert). In the context of this work, the task is critical to the research of underutilized crops, whose information is either limited or fragmented. Through proper identification, a better understanding of crop habitats and their geographical distribution could be extracted, revealing the possibility of cultivation.

In the past, traditional VC approaches such as map interpretation or in-situ investigation were both time- and labour-consuming [169]. Technological developments have been producing a vast amount of data (e.g. aerial and satellite imagery) and, inevitably, new challenges. Remotely sensed data, for instance, are acquired from multiple sources which are differentiated by various characteristics that determine their strengths and weaknesses. To date, these materials are considered as the primary resources for such classification tasks, providing specular observations of different scales, features, or visiting frequencies. However, remote sensing data thus requires comprehensive correlation to guarantee consistency [169]. Furthermore, the surplus of data also creates enormous overhead for the process, leading to the fact that traditional approaches (e.g. in-situ investigation, paper work) are no longer feasible, not to mention the lack of consistency stemming from their manual procedures. Therefore, it is necessary to develop automated processes that are capable to handle tasks in high volume and without much human interpretation.

Early works on incorporating Artificial Intelligence (AI) into the VC task are known as knowledge-based approaches, which attempt to encapsulate knowledge from experts of the field into computer-readable form. However, they face many challengers for the conversion, as human beings are subjective in expressing their own knowledge and modern problems involve a great variety of situations that sometimes have never

been experienced before [80]. Machine Learning (ML) models, on the contrary, mine a large set of data while trying to identify which features may be relevant. Such approaches possess the capability to derive the correlation between patterns and possible outputs. Therefore, owing to their capability of automating the whole process, they have become the dominant choice of the field.

1.1.2 Deep Learning

In order to select a model for building the vegetation classifier, a review on the past implementations of standard ML approaches in the field was conducted. To validate the comparison, in Chapter 3, experiments were conducted on a satellite imagery dataset that was specially designed for the area of interest. Eight different models were investigated, namely Naive Bayes, Support Vector Machine, ANN, Radial Basis Function network, Deep Belief Network, Decision Tree, Random Forest and AdaBoost.M1. The results demonstrated the relative superiority of ANN approaches, which were then selected for the classification task.

I further investigated the current trends in ANNs, a ML approach that was inspired from biological neural network, structuring into layers of nodes. The work primarily focused on the design of network topology (i.e. the architecture). The investment in depth (i.e. the number of layers), termed as Deep Learning (DL), is undoubtedly the most fruitful trend in both industry and academia, which has continuously broken records for different tasks in recent years [175]. However, its actual efficiency remains controversial and is the subject of much debate. Typically, DL enjoys many advantages in learning hierarchical representations (i.e. higher-level

features are built from lower-level features). In contrast, observations of extremely deep models have indicated a skip-through effect (i.e. layers providing minor to no contributions towards learning) on many layers [153]. The high degree of redundancy implicitly calls into question the necessity of using extremely deep networks.

Wider networks (i.e. having more nodes or feature maps at each layer) still prove their advantages with respect to certain aspects, especially in the effective flow of gradients across layers and the ease of boosting through parallelism. However, networks with excessively wide layers face multiple difficulties in optimization, caused mainly by the enormous number of parameters involved [173].

Whilst going to extremes of either width or depth does not necessarily result in the desired outcome, modularity has been shown to be a rising star in the field with multiple successes. However, past modular networks tend to require much human intervention (i.e. tailoring) or involve similar components in a sequential regime [155]. There are only a few works that explore the adoption of submodules in parallel. [29]

1.2 Motivation

The research study in this thesis is motivated by the challenges of ensuring food security. Reports claim that the world population is extremely dependent on the main crops, such as wheat, maize, and rice. Among 7000 species being cultivated, these three crops alone make up more than half of the world demand [48]. Therefore, it is important to seek alternative (i.e. underutilized) crops that help prevent catastrophic famines caused by various phenomena (e.g. droughts, floods, or diseases). However, most of the underutilized crops have not attracted much attention from academia

and industry. The lack of investment has resulted in not only knowledge gaps but also fragmented data that prevent further investigation.

Toward ANNs, it is inevitable that depth is the most fruitful investment in both industry and academia. However, it is still debatable whether wide networks would be a better trade-off, as they are more favorable to parallel paradigm and capable of having similar generalizability to existing deep models. Different from conventional studies that attempted to prove the ability of reproducing the generalizability of deep models, the work reported in this thesis emphasizes on computational expense, as the superiority of wider models.

1.3 Objectives

The initial objective of this study is to conduct a study on vegetation classification. However, as the project evolved, the primary goal switched towards constructing optimal neural network architecture. This can be formulated by the following sub-objectives:

- Study the tasks of vegetation classification to design the application.
- Conduct the case study over remote sensing imagery, within Malaysia region.
- Propose and analyze network architecture that reduces the expense on computation, supporting width over depth.
- Promote modularity in the proposed models.

1.4 Research Scopes

The project attempts to support these studies by closing the knowledge gap of under-utilized crops. The role of vegetation mapping in specific area is vital, especially in efforts to cultivate in unfamiliar territories. Thus, a case-study within Malaysian region was conducted, in which the objective is to build a model for classifying different crop species based on satellite imagery.

In this thesis, the primary focus is on the algorithm behind the mapping process. Instead of building a complete standalone solution, the design and optimization of the underlying models are emphasized, which in turn would be beneficial for the feasibility of the project. The work specifically explored the area of Artificial Neural Networks (ANNs), in which the gap of computational cost is identified and tackled. Due to the fact that vegetation datasets are costly in term of computational resources, the findings on model optimization are instead, evaluated on standard Machine Learning datasets.

1.5 Proposed Approach

From both literature and preliminary experiments, the problem of computational complexity in ANN models was highlighted, especially in larger models. In fact, various attempts have been made to tackle these problems from different perspectives [88, 42, 176, 129]. However, multiple gaps in past studies were identified. First, many approaches require additional or specially configured hardware, which usually imposes extra expense or work overhead. Second, in our everyday lives, we experience

different sorts of parallel processing. Such parallelism is beneficial in many cases, especially when facing multiple similar tasks. However, the trending deep networks which involve a massive sequential computation do not favour this regime. Being tied to the same parameter budget, wider but shallower networks provide a better environment for the parallel concept. However, wide networks have never beaten their deep counterparts in terms of preference, owing to the fact that DL is capable of discovering more interesting features. Finally, in deep networks with narrow layers, modules are stacked sequentially with similar characteristics to study progressive representations (i.e. based upon previous components). Modularity, which is represented by the variation between subcomponents, is thus rarely acknowledged with studies of independent features (hard coding might be an exception). Because modularity is implicitly included in parallel processing, it is logical to expect more efficient learning from boosting this property.

1.5.1 Parallel Circuits

Parallel Circuits (PCs) are the focal point of this thesis. The proposed architecture is inspired from paradigms in nature that favour parallelism and modularity, such as the human retina or brain. Instead of confronting the learning algorithm, this work tackles the optimization of network design. In particular, the proposed approach isolates different parts of the networks to form circuits, eliminating (i.e. pruning) the inter-circuit connections to force them to work semi-independently. This solution is based on two main hypotheses:

1. **A pruned network should result in a speed gain, or at least in lower-**

ing the computational cost. Various neural network implementations rely on the matrix computations that may explode if their dimensions becomes too large. Isolating the network into circuits would convert the large matrix operation into a combination of similar but smaller sub steps and thus, ease the computation. The reduced number of weight connections would also lower the memory requirement.

2. Modularity of the semi-dependent circuits should be beneficial to the generalizability of the model. Useful knowledge is memorised in the network weights. By removing some of its connections, the network is expected to experience a loss in capacity to certain extent, which might harm the model accuracy. However, modularity, which originates from the semi-independent processing of circuits, would produce more distinctive learning between them. Based on the divide and conquer theory, it is expected to achieve the ability of automatic problem decomposition, thus compensating the losses from the pruning process and even further improving the generalizability.

Initial (i.e. proof-of-concept) experimental results, as shown in Chapter 5, clearly demonstrated the advantages of PC over the baseline (i.e. fully connected) models under particular circumstances. There was a significant boost in training speed, whereas the reduction in accuracy was very modest. However, these improvements were observed to be restricted to considerably large networks. In fact, the proposed architecture exhibited significant variability in performance (of both speed and accuracy) when implemented in smaller networks. Although it is simple to generalize

the reduction in computational complexity as a function of circuit count, the experiments, in contrast, showed that performance did not closely follow the estimation. In fact, the training speed improvements were influenced by multiple dependencies, especially circuit-related hyper-parameters and thus, cannot be expressed as linear functions of the complexity.

Attempts were made to explain the failures in smaller networks by two main points. First, after destroying connections and forming circuits, there is a certain amount of additional overhead, as the subcomponents do not function completely independently. At particular points, they need to communicate before processing further. Because the sizes of the original (i.e. pre-pruned) networks are quite modest, the reduction in complexity would not be sufficient to compensate for the overhead. In other words, the networks need to be sufficiently wide for the speed boost to become noticeable. However, I believe my achievement is valuable, as the trending research is tending towards much larger networks. Second, I identified the issue of variability in accuracy within the second hypothesis. In an attempt to further investigate the causes, a set of experiments was conducted on model sparsity (i.e. the degree in which the network parameters are scattered throughout their ranges). The motivation for the evaluation of sparsity was to measure the differences in learning across circuits. Higher sparsity forces fewer network components to be active at the same time, creating variation (or specialisation) in learning that benefits modularity. The results showed that although the sparsity could be boosted, owing to the PC architecture alone, the degree of sparsity is still insufficient for the automatic problem decomposition property. Furthermore, this work attempted to improve the sparsity

by introducing penalty terms into the cost functions. However, even with better conditions, most of the enhanced models were unable to surpass the original fully connected network, except for a set of experiments on varying the coefficient of the sparsity constraints.

1.5.2 DropCircuit

Although the attempts to boost network sparsity did not work as well as expected, the result contributed to a valuable discovery. In particular, the PC approach performs significantly better when each circuit is treated as a single unit and by creating diversity among them (e.g. in this case, feeding them with different parameters: penalty weight). In literature, DropOut has notable effect on preventing the co-adaption of nodes, by simply disabling different nodes during training [163]. Inspired from both the previous findings and the DropOut concept, I scaled DropOut up to circuit level in the pursuit of specialized learning, called DropCircuit. Following a similar regime, DropCircuit treats each circuit as a single, unified unit and deactivates them in a random pattern.

Experiments were conducted to verify the efficiency of the proposed regularizer (see Chapter 6), in comparison with different DropOut-PC combinations as well as the baseline fully connected networks. A remarkable improvement was observed in which better accuracy was obtained regardless of network size, which was impossible in the set of experiments using PC alone. Furthermore, in the context of fusion with PC architecture, it was shown that that the proposed modification of DropOut (i.e. DropCircuit) worked better than the original DropOut.

1.5.3 Constrained Network

Despite the great success of DropCircuit implementations (as shown in Chapter 6), they resulted in a negative side-effect on the sparsity of the models. Owing to the importance of modularity, it is necessary to preserve the sparsity by seeking for alternative penalty terms. The proposed modification was inspired by implementations of Non-Negative Constrained (NNC) network that penalise the negative weights in networks, resulting in a capability to learn distinctive and part-based features [26, 76]. There exists similarities between these studies and other research on nature-inspired constraints. However, the NNC network studies consistently treated all of the nodes by the same constraints. It is worth mentioning that in biological brains, there are inhibitory and excitatory parts containing contrastive properties. Loosely defining parallel circuits as high-level and repeating brain modules and following the DropCircuit regime, a set of constraints is proposed, in which discriminate circuits into two distinctive types: inhibitory and excitatory. In general, these types of circuits are correlated with the main types of neurons in the brain, being constrained to produce either negative or positive output connections, respectively.

Experiments were conducted, as reported in Chapter 7, to compare between proposed constrained PC, the original (i.e. unconstrained) model, and the constrained model in the literature. It is important to note that all of the implementations were combined with DropCircuit, to emphasise the objective of being more compatible with it, compared with the conventional penalty. The results in Chapter 7 clearly demonstrate the dominance of my work, showing identically clear features and sparse

weights compared to the referenced constrained models, with even better accuracy.

1.6 Research Framework

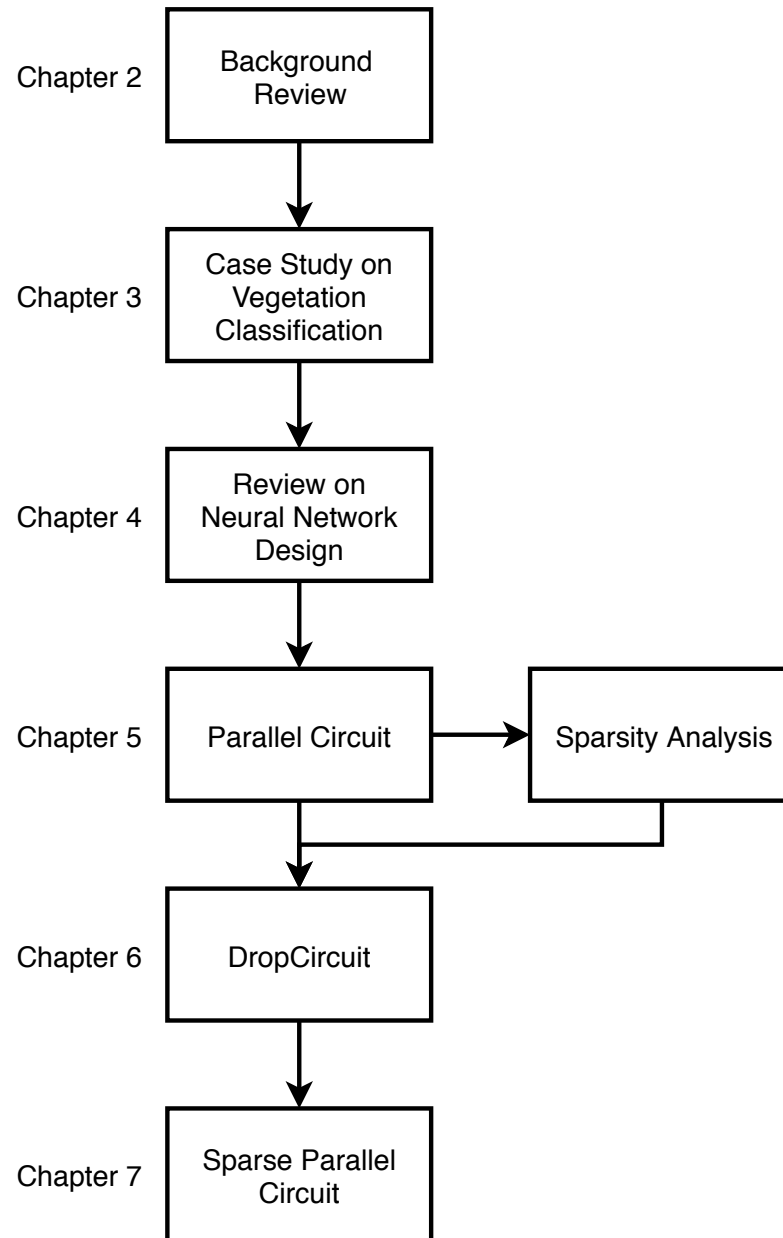


Figure 1.1: Research Framework

1.7 Research Significance

The research in this thesis has demonstrated the importance of modularity in designing ANN models. On one hand, large and densely connected models contain much higher computational complexity than models that are constructed from a pool of semi-connected subnetworks. On the other hand, although a portion of the generalization power is expected to be lost through the pruning process, it could be recovered by effective learning that benefits from the diversity among circuits.

The key contributions of my research are stated as follows:

1. a new dataset for the task of vegetation classification, which is collected inside Malaysian territory;
2. experiments comparing different standard classifiers on the new dataset, supporting the model selection;
3. an efficient, modular architecture called PC, which was proved to provide significant speed gain without harming the accuracy or requiring additional hardware;
4. proof-of-concept experiments showing the advantages of PC models on five datasets obtained from University of California Irvine (UCI) machine learning repository;
5. experiments investigating the sparsity of PC models, discovering the importance of diversifying the learning process in a circuit-wise fashion;
6. a novel circuit-level regularizer, DropCircuit, that eliminates the variability in performance of PC models, by diversifying the learning in individual circuits.

7. experiments that demonstrate the benefits of implementing DropCircuit on PC models, evaluated on MNIST and four of dataset used in (4).
8. a set of biology-driven constraints for PC models that cope well with DropCircuit whilst preserving model sparsity, including the separation of excitatory and inhibitory classes.
9. experiments proving the effectiveness of the constrained model, as well as proving my biologically inspired constraints are superior to those of similar works in the literature which do not differentiate the two types.

1.8 Overview

The remainder of this thesis is organised as follows: Chapter 2 provides a brief background on VC, including its roles and challenges as well as its developmental history. Chapter 3 reviews several standard machine learning approaches for the task. It also includes a description of my dataset and a preliminary benchmark on this particular problem. Chapter 4 reports the current trends in network design, focusing on the debates on the merits of width and depth. Chapter 5 is the focal point of my thesis, which introduces the novel PC architecture with implementation designs. The chapter presents a proof-of-concept experiment and another investigation on its sparsity. Chapter 6 introduces DropCircuit, a modular regularizer to further improve the performance of the PC model. Chapter 7 reports a set of biologically inspired constraints that help preserve sparsity. Finally, Chapter 8 summarises the main findings and contributions of the thesis.

Chapter 2

Vegetation Classification

2.1 Crop Management

2.1.1 A Brief History of Crop Management

Throughout human history, agriculture has been considered as one of the key economic sectors and the main source of livelihood. Prior to the emergence of agriculture, human societies were based on hunting and gathering. The advent of cultivation is a remarkable milestone in the history of human beings, allowing them to settle down in one place instead of leading nomadic lives. The productivity of intensive farming surpasses that of hunting and gathering, allowing the feeding of a greater density of population. Furthermore, excessive production leads to accumulation for off-season usage, thereby boosting prehistoric trading. In fact, agriculture is the backbone of the entire economic system. Its activities provide not only food but also fibre, medicine, fuel, and a vast variety of products that are essential for daily life or other economic sectors. Agricultural demand has been growing day by day, proportionally to the human population, with our quest for a better quality of life. Over 7000 plant species are cultivated by human beings, and yet only three of these (i.e. maize, wheat, and rice) supply half of the worlds demand [48]. Global food security is extremely depen-

dent on these main crops. With more than 7 billion people on Earth today, it is a fact that any yield loss, even of a very modest amount, may lead to severe damage at the individual, community, or even societal level [160]. Therefore, by studying the potential of underutilised crops, it is possible to find alternative supplies that can provide more flexible solutions to ensure the food security.

In order to meet the increasing demand with limited available resources, agriculture practitioners have to rely on cultivation techniques (i.e. crop management). By definition, crop management is the task of controlling and directing crop production; it enables the provision of useful products at reasonable prices, but with an acceptable margin of profit for farmers [160]. Initially, crop management attempted to successfully nurture the desired plant only. Through centuries of development, it has been maturing, striving to acquire the maximal crop yield or to improve the nutritional value of crops, while minimising the environmental impact. In the early days, most of the work was done manually and on a much smaller scale. Solutions were based on one's own experiences or passed down from generation to generation. One such example was the recognition that higher yields could be achieved in soil that contains a generous amount of manure and water. Knowledge obtained in the hunter-gatherer era allowed early farmers to determine which crops was best suited to each season. However, from time to time, wild crops were replaced by selectively breeding species. During the industrial age, with the wide introduction of large machinery, direct interaction between farmers and their plants has been sharply reduced, and larger fields have become the basic management unit. Intensive farming with advanced practices, such as crop rotation and artificial fertilizer, also contributes to the

massive improvement in yield rate, while allowing the soil to relax between seasons. In this era, uniform treatment is applied to the entire field simultaneously, resulting in a significant cut in human labor and time consumption. However, as a trade-off within this cultivation strategy, the treatments (of manure or herbicide) would be inefficiently applied [109]. This waste might result in not only financial problems but also a series of environmental issues.

Agriculture has transformed dramatically during recent decades, owing to the rapid technological development [121]. Innovations in computing, positioning, as well as monitoring by remote sensing have enabled the implementation of new analytical approaches. However, the most significant movement has been the shift towards precision, or so-called site-specific management [37]. Precision agriculture (PA) provides rich and timely information which enables farmers to effectively manage their fields, especially at finer scales. Decisions are made based on decision support systems that analyse the observations and measurements of field variables. Detailed investigation could help in estimating the fields potential productivity as well as detecting different plant stressors. Aside from crop yield, farmers have to manage the nutritional value and the safety of products, as well as considering the guidelines on the use of agricultural chemicals to minimise the environmental impacts. Information on soil, climate, or plant conditions (e.g. moisture, nitrogen levels, pH, and rainfall) are always in high demand owing to the emergence of these extra concerns. By adopting advanced positioning techniques (e.g. GPS), PA practitioners create precise spatial maps of a collection of field variables, which are crucial for this type of micro management.

2.1.2 The task of Vegetation Classification

Multiple analysis techniques have been introduced to satisfy the numerous requirements coming from PA, ranging from the identification of spectral signatures of plant species, water, or soil to the measurement of their thermal radiation. Among these, VC is one technique that has been actively investigated in both academia and industry in recent decades [18]. This trend can be explained by the fact that VC plays an important role in a wide range of applications, especially in modern cultivation practices [144]. For example, Xiao and Moody attempted to identify soil degradation and desertification areas based on VC analyses, and Yang et al. utilised VC to aid their crop yield prediction system [167, 171]. VC is also a vital component in systems that monitor invasive species (i.e. weeds), which have attracted much interest since the introduction of remote sensing resources. By keeping track of invasive species spreading direction and their status, one could implement appropriate techniques to partially control them or eliminate the threat entirely.

2.2 Vegetation Classification

2.2.1 Vegetation Classification Based on Remote Sensing

In the past, the task of VC was heavily dependent on field-based investigations and traditional methods such as field surveys, map interpretation, or literature reviews [169, 149, 83]. These methods inevitably exhibit many limitations at larger scale. They suffer from being subjective, time consuming, and excessively expensive. The fundamental cause is that the majority of works are conducted through manual procedures which lack consistency. Remote sensing has been shown to be able to provide more

timely and economical solutions to the problem [149]. Unlike traditional approaches, remote sensing investigations are capable of overcoming the issues of high cost and inconsistency, providing observations over a large area of interest and enabling automated processing [89]. These methods are advantageous for problematic tasks such as providing multiple observations at different scales or investigating data obtained over decades. Numerous research studies have been conducted on both global and regional scales, such as [113, 82]. A more complete list can be found in [111]. Some studies have extended the implementation of remote sensing into marine ecosystems, as shown in the works of [161, 162] or in the review of [133]. Moreover, remote sensing methods can be easily integrated with Geographic Information Systems (GIS) for the simultaneous processing of different coordinated data.

Owing to the rapid development of technology, remote sensing has become a dominant approach for both scientific research and real-life applications (e.g. map making and disease tracking). Studies have claimed that remote sensing data such as aerial photographs or multi spectral satellite imagery are now the primary sources for vegetation mapping, and these implementations have achieved mixed success [169]. For instance, a combination of helicopter-based and aerial photos was utilised to identify the distribution of *Melaleuca quinquenervia*, an aggressive species being targeted for eradication [105].

2.2.2 Data Characteristic

In the studies mentioned above, remotely sensed data are mostly acquired from multiple sources. Each source has its own advantages as well as limitations. In general,

people mostly characterise a remote sensing material based on its resolution. O'Neill and Dalsted defined the resolution of particular dataset as the quality of images or other spatial material [115]. In fact, there are four primary types of resolutions: spatial, temporal, spectral, and radiometric resolutions.

According to a number of studies, there are trade-off effects between the above criteria. Tsai and Chen stated that high spatial resolution imagery can provide data with the finest detail but at the same time, is only able to acquire within a limited wavelength band [148]. In contrast, high-spectral-resolution data may involve up to hundreds of signal channels but is relatively coarse in terms of spatial resolution. This limitation can be explained by the signal-to-noise ratio (SNR), which is affected by the calibration of sensors. Typically, the signal reflected from a target must have a sufficiently high SNR to be observed by the satellite. The ratios increase when observing either over larger area (i.e. lower spatial resolution) or over broader band (i.e. lower spectral resolution). A signal of simultaneously optimal spatial and spectral resolution would result in a low SNR and thus, would not be detectable. Furthermore, acquiring data of higher resolution (of any type) typically leads to a sharp increase in data volume. However, the data volume has to be balanced with the transmission capacity. An increase in one type of resolution would be compensated by decreases in the others.

2.2.2.1 Spatial resolution

Spatial resolution is a term referring to the quality or fineness of the spatial detail over a display unit (e.g. pixel) [115]. This is likely the first attribute that comes

Material	Approximate Range of Spatial Resolution (m)	General Level of Plant Discrimination
Common satellite images	30 (Landsat) 20 (SPOT Multispectral) 10 (SPOT Panchromatic)	Separation of extensive masses of evergreen versus deciduous forests
Small satellites	> 1 (Panchromatic) > 3 (Multispectral)	Recognition of large individual trees and of broad vegetative types
Airborne multi-spectral scanners	> 0.3	Initial identification of large individual trees and stand-level characteristics
Airborne video	> 0.04	Identification of individual trees and large shrubs
Digital frame camera	> 0.04	Identification of individual trees and large shrubs

Table 2.1: Spatial resolution ranges and level of recognition of some specific remote sensing materials

to mind when evaluating any dataset, particularly in mapping or classifying applications. Higher resolution provides better detail of each object, making it easier to discriminate. In 2006, Navulur proposed a rough guideline to define spatial resolution [112]. A dataset could be categorised based on a parameter termed the Ground Sampling Distance (GSD): low or coarse resolution for GSD of 30m or greater, medium resolution for GSD of around 2–30m, high resolution for GSD in the range of 0.5–2m, and very high resolution for pixel sizes less than <0.5m. The guidelines are summed up in Table 2.1. Although Landsat satellites are the sources that have the longest history and widest usage in the field of earth monitoring, their acquisition data are of medium or coarse resolution only [169]. It is worth mentioning that this intrinsic resolution can be degraded by many other factors, including improper focusing, atmospheric scattering and target motion.

2.2.2.2 Temporal resolution

Temporal resolution refers to the frequency of acquiring data from the same area, or the period of time for completing one entire orbit cycle. For remote sensing material, especially satellite imagery, the data is acquired autonomously and regardless of climate condition. Thus, there might be many unusable data, owing to negative effects of phenomena such as cloud or rain [115]. An appropriate temporal resolution will increase the opportunity to acquire high-quality (i.e. cloud-free) images from the sensors. A temporal resolution of one day is advantageous for vegetation mapping at multiple investigation scales [169]. As a typical example, it is possible to study large-scale dynamics of environmental changes or to monitor the growth and distribution of particular plants. However, for these applications, sun-synchrony is another important criterion for the data acquisition. The sun shading differs throughout the day. A scene may be completely different if it is not captured at similar time of the day. A previous study has also shown that the period of interest usually has a very limited window [169].

Regarding temporal resolution, we could categorise datasets into either multitemporal or single-date. Multitemporal data are acquired on more than one day, season, or year. This type of data has been used in many vegetation discrimination tasks [101, 106, 140]. There have been debates on whether using single-date or multitemporal classification would yield better results. In particular, Langley et al. performed a comparison of both approaches [89]. They indicated that multitemporal data had not been widely collected for their area of interest (i.e. semi-arid environment), and

claimed that such data would improve the accuracy, but they failed to prove their hypothesis. However, multitemporal data analysis requires additional correlation processing of the data to ensure consistency.

2.2.2.3 Spectral resolution

As mentioned above, a particular sensor is only capable of acquiring a fixed range of wavelength. Spectral resolution is the attribute that represents the intervals over which it can define a fine representation [115]. Based on this measurement, there are two main types: multispectral and hyperspectral.

Multispectral datasets usually consist of two to four bands. They have been used in many studies of different scales, from the regional to local level [103, 146, 166]. However, they still exhibit a number of limitations. According to Tsai and Chou, the medium spatial but limited spectral resolution are the two factors that make it very difficult to implement multispectral materials in species-level classification [149]. They are not specialised in any of the two resolutions, thus providing insufficient information for the tasks. In fact, data with fewer than four bands are commonly used for evaluating plants' health status rather than classifying them.

Hyperspectral material involves hundreds to thousands of bands in fine quality. For example, the Hyperion satellite has 220 channels, providing much better coverage ranging from visible to infrared wavelengths [148]. In a series of studies, Tsai and Philpot attempted to discriminate invasive plants in wetlands by analysing the in-situ hyper-spectral data [150, 147]. However, owing to the large number of bands captured, the data volume and dimensionality are usually very high [148]. Current hyperspectral

technology still has quite low ground resolution, limiting its performance in mapping tasks at small scale, which require high precision.

2.2.2.4 Radiometric resolution

Radiometric resolution refers to the ability of distinguishing the differences in intensity, or the number of grey levels between pure black and pure white. The higher it is, the more easily the system can observe or discriminate objects of the same colour, or from the same wavelength. For example, Landsat sensors can acquire data within 256 levels, whereas Ikonos-2 can measure up to 2048 grey levels.

2.2.3 Current Issues of Vegetation Classification Based on Remote Sensing

According to [149], accurate spatial information is vital for developing conservation or restoration strategies. However, increasing the accuracy of terrestrial vegetation analysis is a continuing challenge [125]. One of the most fundamental problems is the surplus of data. The number of species under observation is extremely high, especially when conducting mapping on a large scale. This is due to the fact that vegetation has to respond to a variety of climate conditions or natural disturbances. Throughout millions of years of existence, evolution has diversified them into an enormous number of species and habitats [169]. Each has its own spectral range and their features are visible within only this particular set of wavelengths. Observations thus require the use of appropriate sensors to acquire target plant information. In particular, vegetation data is usually captured in the red and near-infrared regions, because plants are very sensitive to illumination in this domain. Owing to the diversity of vegetation

species, finding a particular sensor which can satisfy all the requirements alone is considered impossible. In fact, some of the observation areas are very dense with objects of contrasting spectral characteristics. Researchers have to combine data from many sources, which requires an enormous amount of work. In general, as mentioned in [149], PA practitioners are facing two main struggles. On one hand, reducing the feature dimensions is critical as it enables us the use of a much simpler description of each species. On the other hand, because the vegetation databases involve different measurement techniques, it is necessary to derive a global metric for the sake of consistency. The vast data, both in terms of quantity and dimensionality, are challenging for the development of an appropriate information extraction framework to discriminate them.

In remote sensing applications, the sensor (or satellite) could be considered as the key device of the whole process [169]. Selecting the appropriate devices is vital in order to maximise the efficiency of data collection. Apart from the spectral range mentioned above, there are many other criteria, such as the mapping objective, image cost, and occlusion. The mapping objective refers to the target of interest and the expected detail level. For species-level observations, it is compulsory to have high-resolution data at hand. Aerial and multispectral remote sensing data is insufficient to produce a high-quality result with high confidence. However, it is not always true that the accuracy of vegetation mapping would be proportional to the resolution. In contrast, when performing tasks on a larger scale, high-resolution materials are not always required, owing to resource redundancy. Not all of the information is useful for the interpretation, to say nothing of noise. For example, Gougeon and Moore stated

that very-high-spatial-resolution data may cause difficulties for their forestry analysis [64]. Fortunately, some successful studies have proposed alternative approaches to overcome the above problem. Tolpekin et al. conducted a study on a contextual classification approach and achieved some success in improving object identification from low-resolution dataset [143]. Franklin et al. successfully improved the classification accuracy by adding image texture into context [50]. [34, 128] indicated that an alternative source with hyper-spectral imagery could improve the results (i.e. investment in spectral rather than spatial resolution). High cost is another important factor that prevents the use of these materials. Finer resolution imagery may be very expensive and thus, the cost needs to be considered accordingly for the sake of feasibility. Better data benefits the analysis accuracy in exchange for higher expense. Occlusion is another problem due to the fact that remote sensing materials are collected from above. There is a high probability that information of ground-based objects or lower canopies will be missed by capturing devices [165], particularly for dense areas. The climate conditions at the capture time also lead to the same issue, particularly cloud coverage which hide many features of the area of interest. In order to achieve a cloud-free image, we have to choose a sensor of appropriate traveling cycle, or combine multiple sources.

2.2.4 The Introduction of Machine Learning into Vegetation Classification

As mentioned above, VC is a vital part of multiple monitoring applications. Mapping in such applications requires the interpretation of both in-situ and remote sensing

data [27, 49]. Recently, much attention has been paid to complex land use, in which various types of landscape need to be characterised. However, the huge amount of diverse, inconsistent data has become a challenge for classical methods. Automated classification is an alternative technique that ensures the process to be objective and reusable for similar problems [41].

Expert systems (i.e. knowledge-based approaches) pioneered the trend of incorporating Artificial Intelligence (AI) into the process of VC. They act as a mechanism to hard-code the human knowledge into computer-usable form (i.e. in more formal language). Based on this, machines are able to derive their own reasoning through logical inference rules. However, these systems used to face a problem, so called the ‘knowledge acquisition bottleneck’ [80], which can be attributed to two main reasons. First, the process is time consuming as it requires the interaction of both the field experts and knowledge engineers over a long period. Second, most of the experts are capable of expressing knowledge subjectively (i.e. inconsistently), and some struggle to describe concepts with sufficient complexity. It is challenging to restructure these precious resources into sufficiently explicit forms to be used by computer applications. As mentioned in [63], the real challenge in AI is to deal with tasks that seem easy for humans to perform but difficult to formally abstract. In fact, the issues in building up expert system have already been recognised by many image analysis applications.

The difficulties with handcrafted knowledge leads to the emergence of a new need: the ability of the machine to acquire knowledge by itself, through learning from the raw data, i.e. Machine Learning (ML). Enslin et al. emphasised that it is a rising trend in the field to apply ML approaches to GIS-supported analysis [45].

2.3 Conclusion

This chapter briefly described the development of agriculture throughout our history, from basic cultivation to modern site-specific management. It highlighted the importance of VC in a wide range of applications, which motivates our research presented in this thesis.

The chapter also provided detailed insight of remote sensing data, which have been considered as the primary source for the VC tasks. Multiple properties that differentiate them were reviewed, as well as the challenges faced by VC. We concluded by discussing the importance of automating the process by utilising ML.

Chapter 3

Comparison of Standard Machine Learning Approaches - A Case Study in Malaysian Region

This chapter focuses on standard ML approaches, summarising their current implementation in land cover analysis applications. In addition to the review, a benchmark was conducted over Malaysian region to support the selection of the classifier.

3.1 Related Works

3.1.1 Overview

In the development of VC, several classifiers have been applied to both research and practice. In their remarkable review, Xie et al. categorised the commonly used algorithms into two groups: traditional and improved classifiers [169]. The traditional category includes classical algorithms, both supervised and unsupervised. Among the supervised classifiers that are based on statistical distributions, the most popular approach is the Maximum Likelihood Classifier (MLC) [135]. However, MLC is restricted by the assumption that the data has to follow Gaussian distributions, which is not necessarily true in complex cases [169]. The model accuracy is heavily dependent

on how well the data fit this assumption [170]. Unsupervised classifiers, however, are widely used in thematic mapping, especially vegetation cover mapping. These approaches have advantages in automatically converting raw data into useful information, as descriptors [151]. K-means and ISODATA are two typical examples for this group [169]. However, they also possess a significant drawback. As unsupervised classifiers do not have a mechanism for assigning new instances into existing classes, the whole process has to be repeated whenever new data points are introduced [21]. The result may be very different from; even members of predefined clusters might also be shifted away.

Improved classifiers refer to modern models such as Decision Tree (DT), ANN or Fuzzy Logic [169]. These classifiers usually exhibit better performance compared with the traditional group, especially when dealing with a high volume of data or complex problems [47, 108]. Many studies have been focused on improving the data extraction process. For example, combinations of expert knowledge and ancillary data have resulted in great improvements in accuracy [56, 132]. Peddle et al. performed a comprehensive comparison of three classifiers: ANNs, Evidential Reasoning and MLCs [120]. In their experiment of classifying alpine forest, ANN was proved to be superior. Hepner et al. also recommended using ANNs in a remote sensing context, because it is not dependent on any assumption of data distribution [71].

In this preliminary benchmark, we focus on the improved classifier category and implement eight different algorithms, namely Naive Bayes (NB), Multi-layer Perceptrons (MLP), Radial Basis Function networks (RBF), Deep Belief Networks (DBN), Decision Trees (DT), Random Forests (RF), AdaBoost.M1 (AB) and Support Vector

Machines (SVM). Most of these classifiers have been considered in different comparisons [19]. We arrange them into three sets of classifiers, i.e.: Neural Network family (MLP, RBF, and DBN), Decision Tree family (DT, AB, and RF), and baseline (NB and SVM). The first two are our main research focus, whereas the last one is primarily for reference purposes.

3.1.2 Baseline and Reference

NB is a classifier that learns the conditional probability of each attribute, given a particular class label [53]. However, NB is based on the assumption that all of the attributes must be conditionally independent which is, unfortunately, unrealistic for most real-life problems. Owing to this limitation, NB has never been widely used in the field of remote sensing. A comparison of multiple Bayesian classifiers, including NB, was conducted by Solarez and Sanz [136]. They evaluated the classifiers with respect to a remote sensing classification task, and as expected, NB exhibited limited generalizability compared with its counterparts.

SVMs represent a set of ML algorithms that were first introduced by Vapnik in 1995 [152]. Since then, SVMs have been implemented in many real-life problems, ranging from hand writing recognition to text categorisation. Recently, some works on land cover classification attempted to reveal the capabilities of SVMs with regards to remote sensing tasks [81, 180]. These studies shown that SVMs should be considered as competent classifiers within the same league as popular models such as MLC, DT, or ANN. In a similar study, Pal and Mather performed a brief comparison of SVM, MLC, and ANN for two land cover analysis problems [117]. Significantly, one of them

was to classify seven plant species from ETM+ data. They pointed out that SVM could achieve a better result than both MLC and ANN, especially when dealing with smaller training sets.

3.1.3 Artificial Neural Networks

By combining and alternating data features, some studies have already achieved promising improvements in classification tasks. In particular, Tso and Oslen succeeded in using both spectral and contextual data to set up a fundamental framework (Hidden Markov Models) for unsupervised learning [151]. The pooling of multiple data sources is especially favoured by ANNs and has been widely studied in the field. In 1999, Hilbert and van den Muysenberg created a model to characterise forests with 22 environmentally related features, separated into three categories (i.e. climate, soil, and terrain) [72]. They trained a classical MLP and reported that it was far superior compared to MLC. However, multiple ecological variables (e.g. Ellenberg values including water and nutrient availability, light, temperature, and pH) were used in another study by Ejrnæs [44]. A combination of both feature collections was also seen in [177, 21]. In a different application, Patra et al. proposed a semi-supervised MLP approach and compared it with three other classifiers in a change detection task [119]. Experiments showed that MLP easily outperformed standard techniques and obtained results comparable to state-of-the-art records.

Despite the popularity of MLPs in the field, they still exhibit several limitations such as slow convergence, the possibility of getting stuck in local minima, and the over-fitting problem [16]. RBF networks have been introduced as one of the alterna-

tive approaches to address the problem. Unlike MLPs, RBF networks contain only one hidden layer and each hidden node is associated with a particular kernel function. In spite of their simplicity, RBF networks have been shown to (1) have a fast training process, (2) deal well with nonlinear problems, and (3) be robust against missing regions. However, Bruzzone and his colleagues pointed out that the model accuracy is quite sensitive to the estimation of the kernel function’s centre and width [17]. The work of Bastos et al. could be considered as one of the pioneering attempts to apply RBF networks in the field [3]. According to their study, with appropriate configuration, RBF networks exhibited better accuracy than the popular MLC, especially in small training sets. In [17], a new approach for selecting RBF kernel functions was introduced and evaluated in a land cover analysis application. The proposed method was shown to have a lower error rate than the classical RBF networks as well as other commonly used classifiers such as MLPs or K-nearest neighbours. Recently, Yang et al. developed a regression model using RBF networks, aiming to relate between different vegetation indices and agronomic parameters [172]. They empirically proved the superiority of RBF networks over traditional methods.

The motivation behind the implementation of DBNs is mainly its capability to learn robust features automatically [24]. When dealing with small datasets, most of the classifiers mentioned above are proved to be sufficiently feasible by only utilising the raw features. However, with the increasing trend in both the number of samples and feature dimensions (i.e. the training set suffers from larger variance and noisier descriptions), studying features with higher levels of abstraction is necessary. DL approaches learn simple features at the lower layers, whereas latter layers are built

on top of the learned layers and could abstract features of considerable complexity [96]. The hidden nodes are components of Restricted Boltzmann Machines (RBMs), which in turn are the basic components of DBNs. The study of Chen et al. on aircraft detection could be considered as one of the pioneering works in this research area [24]. Recently, Qi et al. conducted a thorough investigation of DBNs within the application of urban land cover analysis [102]. They studied the performance of different DBN variants and compared them with some common classifiers, including SVMs and MLPs. In their experiments, DBNs outperformed the traditional MLPs in almost every category, and showed a considerable advantage over SVMs in terms of overall accuracy.

3.1.4 Decision Trees

According to Safavian and Landgrebe, DTs are known for their capability of breaking down complex problems into simpler tasks [127]. In particular, a DT consists of one root and a number of internal or terminal nodes, called splits and leaves respectively. As explained in [52], the tree recursively divides the data set (complete at the root node) into many subdivisions (branches). Unlike MLCs, DTs are nonparametric and do not require any assumption pertaining to the input data. In addition, they are said to be capable of revealing the nonlinear (or oddly, irrelevant) relationships between input instances in a natural manner. Owing to their simple tree structure, DTs are typically fast and easy to interpret, providing useful insights into the underlying reasoning. Towards land cover applications, DTs have already been implemented in many studies which utilise remote sensing resources. The model is well suited for

consistent, easy data sets (such as Earth science applications with explicit classification structure), enabling the construction based on the user expertise only [52]. For example, Running et al. designed a DT-based strategy for global classification of vegetation from NDVI values [125]. In 1998, DeFries and his colleagues proposed a global land cover classification by applying DT to Landsat images [41]. Friedl and Brodley attempted to follow a similar approach with Advanced Very-High-Resolution Radiometer data [52], showing the advantages of DTs over traditional MLCs. Similar findings comparing DTs and MLCs were reported in [57] and especially in [130].

In recent years, there has been a trend to use a combination of using multiple classifiers (i.e. ensemble) rather than one single model [58, 15]. Complex models such as ANNs or SVMs have advantages in gaining better generalization but suffer from lengthy training periods as well as strong dependence on fine tuning of parameters. Weaker but faster classifiers such as DTs are more desirable for ensemble classification, most probably owing to their lower computational complexity [22]. In general, ensemble classifiers generate several single classification units and decisions are made upon (either weighted or unweighted) popularity votes. The unit classifiers are constructed in a manner that fosters significant diversification amongst themselves. Numerous approaches were introduced with respect to this property, including different sampling in training sets, feature dimensions, target classes, or injecting randomness directly into the models [43]. In particular, Boosting and Bagging can be considered as the two methods that attract the most interest for both research and practical applications. In this work, we implement AB and RF, which are two typical representatives for the groups, respectively. They have shown excellence in the field,

exhibiting better accuracy and robustness to noise than traditional single classifiers [13, 43]. According to Breiman, RFs consist of a number of DTs, where each tree is generated with a random subset of features and instances (i.e. Bagging) from the original training set [14]. In the end, each would cast a vote towards one of the classes and the final ensemble output is typically obtained by a simple majority function. In a few ecological applications which implemented RFs, the performance was competitive with the best available classifiers and superior to most of the common methods [122, 91]. A typical RF study in this field is the work of Gislason et al. [59], in which they investigated the performance of RFs over multisource remote sensing datasets. RFs had a comparable result with other ensemble approaches but was significantly faster. Furthermore, Pal pointed out simplicity as an advantage of RFs over SVMs, while they show equal accuracy in a VC problem [116]. In an attempt to improve the performance of RFs for classification of hyperspectral data, Ham et al., claimed that the accuracy is related to the diversity within the ensemble and proposed a new approach to enhance it [68].

AB is another ensemble classifier that was introduced by Freund and Schapire [51]. In general, AB is a binary classifier that performs large-margin classification by iteratively summing the weighted output of a number of weak learners [9]. There have been multiple attempts to compare AB and RF. In the field of remote sensing, these include the works of Chan and Paelinckx [23], Chan and Canters [22] on ecotype mapping, and the study of Wei et al. on tree and vehicle classification [159]. These studies share the same findings: (1) both RF and AB exhibited better overall accuracy than common methods such as MLPs, (2) RFs are slightly better than ABs with

regards to accuracy, and have faster training speed.

3.2 Methodology

The objectives of the following section are defined as follows:

- Design the plant dataset for the benchmark.
- Introduce the experiment framework, including the evaluation metrics.
- Investigate the capabilities of the reviewed classifiers and discuss their performance.

3.2.1 Dataset Creation

3.2.1.1 Data Source

As mentioned in [32], many studies tend to use medium-resolution datasets for local and national scale analysis, because significant patterns could be recognised efficiently within this level. However, it is also true that remote sensing data is expensive, especially at this level of detail [65]. Data availability and quality are not always guaranteed, as rural and developing areas are generally not sufficiently profitable to justify the required coverage. As a consequence, medium-resolution datasets, not to mention higher-resolution ones, may be impractical for larger scales (i.e. regional or global). Lower-resolution resources are actually more preferable in these cases.

Recently, GoogleMaps (GM) has been proposed as an alternative source for many remote sensing applications, ranging from meteorology analysis and land cover classification to decision support for tourism [156, 54, 77]. In the work of Bernabe et al., it was indicated that high-resolution satellite images of many areas around the world

could be retrieved from GM without any expense [11]. In addition, many application interfaces as well as external libraries have been developed to assist the processing of retrieved GM data. The interactive mapping of GM represents a significant step forward in the field, as its application program interface (API) enables end users to freely access the satellite data and achieve clearer understanding of the areas of interest, significantly improving the functionality of applications [77]. Although much work still needs to be done to guarantee the quality of crowd-sourced features, labels or markers overlaid on the GM images could be an impressive boost for the evaluation of land cover classification [41, 31].

With regards to the project, I decided to collect data from GM owing to three main points:

1. I am tackling underutilised crops with fragmented resources; it is obvious that high-resolution or specialised materials are not always obtainable. GM provides us high-resolution imagery without any expense.
2. GM contains data obtained from multiple sources, which eases the difficulties facing specialised remote sensing data.
3. The main focus is on classification logic; processed data from GM is highly usable, which saves the cost of raw data correlation.

3.2.2 Data Sampling

The area of interest lies within the boundaries of Malaysia and external regions that are close to the border. The dataset consists of five sets of RGB images, corresponding

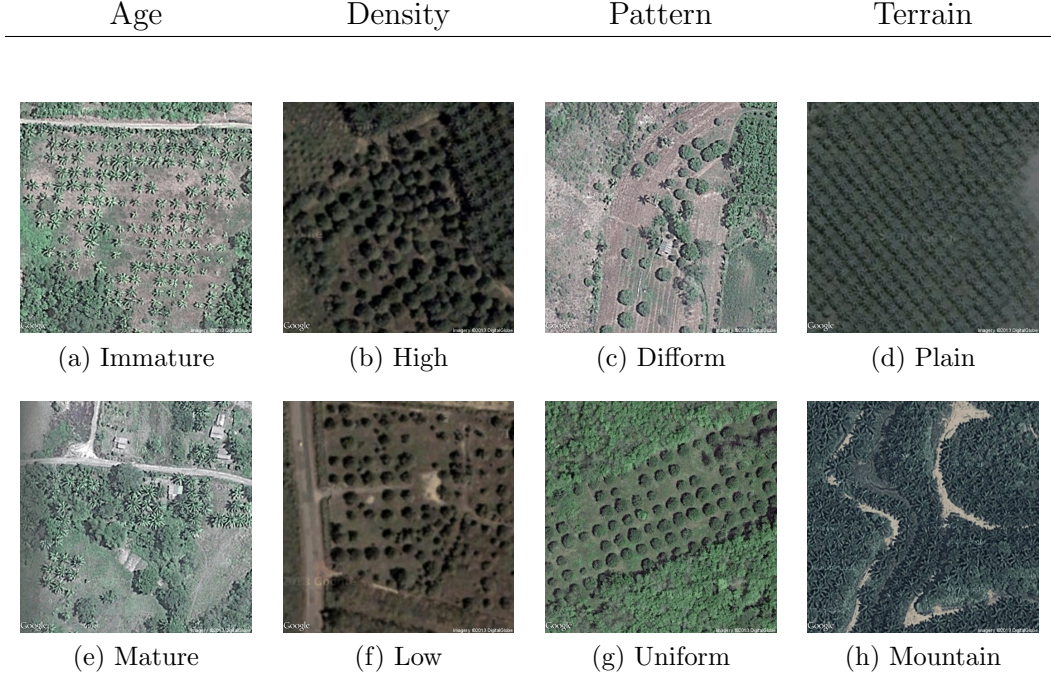


Figure 3.1: Examples of opposing properties

to five different plants, namely palm, rubber, coconut, durian, and mango. Each set is further divided into four subcategories with opposing properties, i.e.: age (immature vs. mature), density (low vs. high), pattern (uniform vs. difform) and terrain (plain vs. mountain). All images are of size 512×512 pixels and of .JPEG file type, as shown in Figure 3.1.

For each category, 10 different samples were selected to populate the set (i.e. five images for each opposing description). Moreover, these samples were selected from scattered locations to maximise the diversity within each group, which partially mirrors the properties of data acquired from heterogeneous remote sensing sources. The ground-truth images are divided into two types: region-based and plant-based, corresponding to the two types of classification as described in Figure 3.2. Both types are converted into binary images for easier interpretation. Plant-based ground-truth

consists of circular dots with different radii aiming to have the smallest fit of the tree canopy. In contrast, region-based images identify the tree area by accepting more errors, considering the small areas between trees as plant regions as well. In this experiment, results are restricted to the plant-based ground-truth, which in turns ignores the rubber subset. Because the space between rubber trees is very narrow and their canopies always overlap with that of others, it is infeasible to discriminate individual plants.

As the primary focus is the comparison between various classification algorithms, I extracted the simplest feature dimension, i.e.: colour intensity. Distinct training-testing sets were created for each plant. For each image (in grey-scale), 200 random pixels are selected in such a way that both classes, plant (of interest) and nonplant, are equally presented. Following this approach, it is expected to avoid the class imbalance as this can lead to a bias towards the majority class. At each selected location, three different convolutions were applied, with the sizes of 7×7 , 9×9 , and 11×11 pixels. As filter size increases, more contextual information can potentially be extracted from neighbouring pixels. These sizes were chosen so that each window would cover at least half of any plant canopy inside our sample images (constrained by the selected zoom level). The same convolution approach is applied to create the testing set. However, instead of sampling from multiple images, one random image is selected from the dataset (of 512×512 pixels) as the validation set. Due to the fact that the experiment is application-oriented, evaluation over an entire image (or an entire area on the map) is more meaningful, compared to over a set of discrete points. As the consequence, the validation set is much bigger than the training set, which is

Train sample	Test sample	Dimension			Class
		7×7	9×9	11×11	
7800 (39 images, 200 points each)	65536 (1 image)	49	81	121	(Plant / Non-Plant)

Table 3.1: Specification of the dataset of each plant

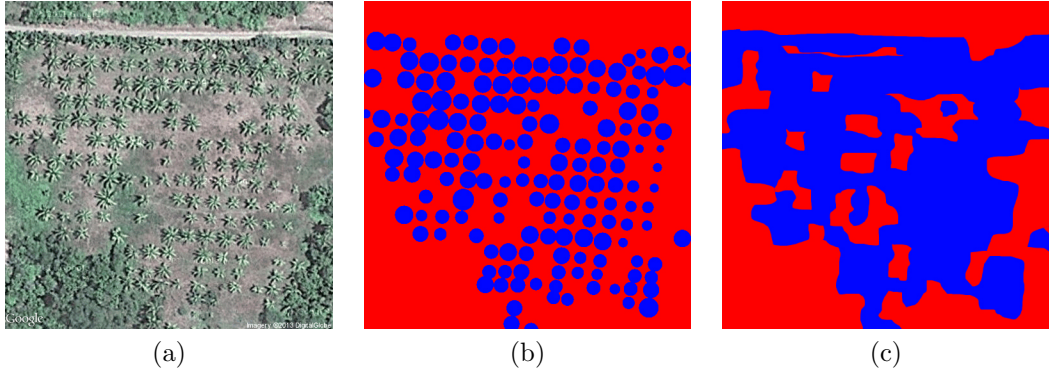


Figure 3.2: Original image (a) Plant-based ground-truth (b) and Region-based ground-truth (c)

different from conventional cases. The specification of the VC dataset is presented Table 3.1.

3.2.3 Classifier Implementation

For simplicity, the majority of the experiments were conducted using the Weka platform, an open source collection of data mining algorithms [67]. In particular, NBs, SVMs, RBFs, DTs, RFs, and ABs were evaluated under Weka’s default configuration. For MLPs, I implemented a Matlab model of two hidden layers, with 100 nodes each. This is because ANNs typically require heavier computation, and thus a Java-based platform such as Weka is unsuitable for the task. DBNs were built based on the work of Hinton and Salakhutdinov [74], and its parameters were selected according to Hinton’s hints on RBMs [73].

The experiments were conducted on a system with Intel Xeon Processor E5420 @ 2.50 GHz and 8 GB of RAM.

3.2.4 Validation

Each classifier was trained on an identical dataset generated from one specific match (plant, window size). The accuracy was based their performance against a randomly chosen image of the corresponding plant subset. For validation, the F_1 score (weighted by the population of each class in the testing image) is used and being defined as:

$$F_1 = \sum_{Class} \frac{2 \times Recall \times Precision}{Recall + Precision} \times \frac{Class\ population}{Total\ population} \quad (3.1)$$

F_1 score, a harmonic mean of recall and precision, is chosen as the validation metric to ensure the fairness of the classification. It is the fact that attention was paid only on the Plant class (i.e. Non-Plant class has limited meaning/insight towards agriculture) whilst the population of Non-Plant samples used to outnumber the other by a large margin. Therefore, a weighted F_1 is implemented instead of plain F_1 .

In addition, Anderson's 10 criteria of a good VC, as indicated in his study, was revisited, especially [1]:

1. The accuracy of interpretation for the several categories should be approximately equal.
2. Repeatable or repetitive results should be obtainable from one interpreter to another and from one time of sensing to another.
3. The classification system should be applicable over extensive areas.

4. The classification system should be suitable for use with remote sensor data obtained at different times of the year.

3.3 Experiments Results

3.3.1 Traditional Classifier

As expected, NBs was the fastest classifier among the eight candidates. Regardless of the complexity of the dataset, the training time was always less than 0.3s, with an average of 0.175s. This record was not matched by any other classifier. In terms of accuracy, it achieved an average F_1 score of 0.717. However, NB was actually one of the most unstable classifiers. Its performance varied significantly, with the highest and lowest F_1 scores of 0.868 and 0.45, respectively. By observing the 7×7 subsets of Durian and Palm, it is obvious that NB exhibited the largest variance between the Plant versus Nonplant categories, demonstrating biased classification (see Table 3.2). In particular, for the Palm dataset, the Plant class exhibited a true positive (TP) rate of 100%, whereas the true negative (TN) rate (for Nonplant) was only 6.3%. In fact, none of the three filter size had a TN rate over 10%, and the average F_1 for the group was only 0.539. Further evidence could be found in the Durian set with contrasting figures, where 94.5% of the Nonplant class was correctly classified, whereas the corresponding result of the Plant class was only 14.8%.

In the experiment, SVMs exhibited relatively poor performance, as evidenced by the fact that it fell in the lower half of the ranking for half of the tests. On average, SVM achieved an F_1 score of 0.7218 (i.e. 4th overall place) and 70.94s with regards to training time. Even though this duration is much shorter than some of the classifiers

Naive Bayes		Nonplant	Plant	F-measure
Coconut	7×7	0.807	0.437	0.836
	9×9	0.780	0.666	0.768
	11×11	0.695	0.694	0.697
Durian	7×7	0.945	0.148	0.842
	9×9	0.817	0.730	0.802
	11×11	0.809	0.470	0.666
Mango	7×7	0.626	0.792	0.696
	9×9	0.846	0.308	0.814
	11×11	0.894	0.211	0.868
Palm	7×7	0.063	1.000	0.450
	9×9	0.084	0.932	0.582
	11×11	0.094	0.925	0.584

Table 3.2: Biased classification of Naive Bayes (bold indicates examples)

(e.g. MLPs and DBNs with 1859.36s and 7058.49s, respectively), SVMs lagged far behind compared with the other classifiers on the list (e.g. RBFs with only 2.78s). Moreover, SVMs exhibited biased classification, particularly in the Palm dataset. In all three filter sizes, the TN rate was approximately 10%, whereas the TP rate was close to 100%.

3.3.2 Neural Network family

MLP is the second most time-consuming algorithm, with the best record of 762.97s and in the worst case up to 3317.32s (see Table 3.3). In term of accuracy, MLP finished with a slightly low score (0.7209, 5th overall) but it did rank in the top three in some particular setups, especially in the Coconut dataset.

RBF networks are said to outperform MLPs in terms of both speed and accuracy. Indeed, the amount of time required for RBFs to finish training was only 2.78s on average, showing much improvement compared with 2009.27s for MLPs. However,

with respect to accuracy, RBFs were shown to be superior to MLPs in only one third of the tests. RBFs ranked last with 0.6661 on average. Compared to other classifiers outside the ANN family, RBFs tend to be weaker in terms of both speed and accuracy. RBF networks also exhibited biased classification, especially in the same datasets that NBs found problematic (i.e. Durian and Palm). The ratios between TP to TN were 99.6% to 24.8% and 12.1% to 93.9% for Palm and Durian (7×7) respectively. Having said that, the differences between the Plant/Non-plant classes were not as large as for the NBs.

The most significant observation one can first derive from the benchmark of DBNs is that they took an impressively long time to complete the training phase. In particular, an average of approximately 7058.49s was recorded, three times longer than that of MLPs (2009.27s) and thousands times longer than most of the other classifiers. In spite of the lengthy training of DBNs, their accuracy was undoubtedly ranked as the best. With an average F_1 score of 0.7459, DBNs were the champions of our experiments, followed by RFs with 0.7368. In almost every dataset, DBNs remained in the top three. Especially in the Coconut dataset, DBNs took the lead for the first two filter sizes and finished as the runner up in the largest (i.e. 11×11) set.

3.3.3 Decision Tree family

With an average score of 0.7013 and an training duration of 2.77s, DTs could be considered as a fairly poor choice. In fact, although they rarely exhibited outstanding results, DTs showed relatively good results in the hardest dataset (i.e. Palm). Achieving an F_1 score of 0.5767 for this particular plant, DTs took the 2nd place after

Duration		NB	MLP	RBF	DBN
Coconut	7×7	0.07	1029.95	1.90	6951.01
	9×9	1.39	762.97	3.73	7181.94
	11×11	1.34	767.6	5.51	7183.68
Durian	7×7	0.94	2198.78	2.32	6236.34
	9×9	1.11	2303.08	2.91	6507.06
	11×11	1.07	2361.55	2.85	6252.5
Mango	7×7	1.11	1876.27	1.47	6401.95
	9×9	1.34	1584.65	2.88	6492.80
	11×11	1.33	1643.84	3.29	6563.55
Palm	7×7	1.62	3088.84	1.90	9315.56
	9×9	1.88	3317.32	3.27	7898.96
	11×11	2.17	3171.39	4.32	7716.53

Table 3.3: Problem of lengthy training time in the ANN family (the longest training duration is indicated in bold)

DBNs. In particular, in the 9×9 subset, DTs eventually took the lead with 0.599.

With regards to RFs (see Table 3.4), the ensemble technique of DTs has been proved to be a competitive candidate, given its qualities in terms of both speed and accuracy. They achieved an average training period of 1.41s (i.e. 3rd place after NBs and ABs). In terms of accuracy, RFs exhibited an average score of 0.7368 (i.e. 2nd place after DBNs) and took the lead for two setups (i.e. Durian and Mango sets, both of 11×11). However, their performance on the Palm dataset was quite disappointing. With an average score of 0.5443, they took the 4th place among the eight compared approaches. In addition, presumably inherited from DTs, RFs are bias-free, regardless of the Plant/Nonplant ratios in the testing sets.

On the contrary, ABs showed fairly modest performance with moderate accuracy but fast training speed (3rd in terms of accuracy and 2nd in terms of speed). Significantly, unlike the other two members of the DT family, ABs exhibited bias in

Dataset		RF	AB
Coconut	7×7	0.872	0.825
	9×9	0.817	0.759
	11×11	0.768	0.657
Durian	7×7	0.849	0.859
	9×9	0.814	0.819
	11×11	0.693	0.667
Mango	7×7	0.728	0.721
	9×9	0.797	0.810
	11×11	0.870	0.888
Palm	7×7	0.569	0.686
	9×9	0.529	0.456
	11×11	0.535	0.560
Average		0.737	0.726

Table 3.4: Comparison of the performance of RF and AB (better accuracy indicated in bold)

classification, though it was not as pronounced as with NBs.

3.4 Discussion

As mentioned above, NB relies on the assumption of independence between the attributes. However, in the experiment, each attribute is the colour intensity of neighbouring pixels. By feeding them to the classifiers, it is expected that the relationships between pixels to be revealed as the process progressed, thus totally contradicting this assumption. As expected, in terms of overall performance, NBs failed in our experiment.

Across individual experiments, one of the most common problems is the biased classification (i.e. between Plant and Nonplant classes), which violates Andersons second criterion for a good vegetation classifier. This issue was particularly prominent in NBs, RBFs, ABs, and SVMs (i.e. four out of the eight classifiers being considered).

In some extreme cases, the accuracy was nearly 100% for the biased class and less than 5% for other. In consequence, the evaluation metric suffered significantly from this phenomenon. In particular, if the biased plant is of the majority class but not of our interest (e.g. of the Non-plant class and we are trying to locate the plant of interest), we achieve a classifier with high score (owing to weighting) but which is practically unusable. In my experiments, attempts were made to overcome this problem by creating balanced training sets with equal counts for each class. However, it is believed that the main culprit lies not within sample count but within the sampling technique. In the dataset, especially for nature sites with inter-cropping between plant species, noise occurs frequently and it is difficult to obtain a good (i.e. clear, noise-free) sample of the plant. Furthermore, as the seed coordinates were picked randomly, there was a high probability that the whole tree canopy was not included in the patch, and that there was a mixture of tree canopy, either of same or different species. Therefore, owing to the random sampling technique, the generalization was greatly reduced. However, preprocessing and feature extraction techniques help decrease this ambiguity. Working purely in the grey-scale domain is not an effective way to deal with noisy vegetation datasets, although it is considered to be necessary in this study to assess the raw potential of different classifiers. In unreported experiments, it is confirmed that using simple edge detection and enhancement techniques can indeed further improve the accuracy results.

Among the eight classifiers being evaluated, RFs and DBNs were shown to be the best candidates for the task. It should be emphasised that neither of them suffers from the biased classification mentioned above. In particular, DBNs are the most

accurate classifier, with a leading average F_1 score of 0.7459. However, they suffered from a serious problem with respect to training speed. RFs lagged slightly behind with a score of 0.7368, but was significantly faster than DBNs. RF is a nonparametric model, which is only concerned with the number of trees. Simply put, the more trees RF has, the better it performs, but it has to trade off performance for complexity. Additionally, owing to its random nature, RF shows little to no potential for further studies in the problem domain. On the contrary, there is an increasing trend of using DBN and DL in general, which have the capability to discover features on different levels of abstraction. Therefore, I decided to implement ANN variants as the chosen vegetation classifier.

3.5 Summary

In this benchmark study, several common approaches for the task of VC were reviewed, particularly reporting their current implementations in the field of remote sensing. A new dataset with regards to the area of interest (i.e. Malaysian territory) was proposed for the validation of these models. Overall, the results have showed the superiority of RFs and DBNs over other classifiers. RFs proved their superiority over ABs, which is an interesting finding, given that the two approaches are among the main competitors within the ensemble category. However, DBNs clearly demonstrated their potential in spite of lengthy training. Based on these lines of evidence, I have selected the ANN family, represented by DBN, as the classifier for the system.

Chapter 4

A Study on Neural Network Design

4.1 The Hype of Representation Learning

Representation learning, often equated with DL, is an emerging research area in the field of ML [5]. The empirical success of DL has contributed to many breakthroughs in various problems, especially in perceptual domains such as computer vision or audio processing. In some particular tasks, DL dramatically improved the state-of-the-art result and eventually achieved human-level performance [175, 28].

Many research studies argue that the main driver of its phenomenal success is the ability to derive better representations from the raw data. As mentioned in Chapter 2, classical approaches showed limited success in converting raw data into an organised, usable form (i.e. representation). In order to overcome this problem, they were fed with handcrafted features, which is both time- and labour-intensive. Owing to the emerging need for automated processing, robust models have become more preferable with their capability of building up the reasoning autonomously (i.e. ML approaches). Fortunately, the past few years have brought a surplus in data volume,

along with the introduction of more efficient algorithms and more powerful hardware. These foundations have leveraged the hype, making it feasible to learn directly from raw data, without much human interference in the process [126]. However, for many classical (early) ML models, their performance is heavily constrained by the representation of input data [63]. One possible explanation is that these models are restricted to exploring only the direct mapping from the input (i.e. raw) representation to the output. For example, upon being given an image, logistic regression may point out the correlation between each pixel and different outcomes but hardly yield any useful prediction. Many models perform very well with the right types of features but it is challenging to produce such representation. For those tasks in which humans have an advantage (e.g. digit recognition), we could easily break down the problem and abstract the critical features (e.g. dots, strokes) from experience. However, it is difficult for machines to define them in terms of raw pixel values. In general, classical ML approaches exhibit many weaknesses in encapsulating raw data into those meaningful features for the sake of better accuracy.

To address the problem, as mentioned in [63], it is a useful practice to expand the learning, not only in direct input-output mapping but also in forming immediate, higher representations on top of raw input, termed as DL. This new approach is claimed to (1) usually produce better accuracy than handcrafted representations, (2) be more robust to various tasks with limited human intervention. Briefly, a deep model is a composition of multiple progressive transformations, each of which is built on top of simpler representations in the previous layers. The increasing level of abstraction plays an important role in DL. More abstract representations

are potentially empowered with better generalizability whilst showing resistance to irrelevant local variations. Unlike conventional kernel machines which struggle when the functions involve varied phenomena (e.g. functions with more wrinkles, objects under different shading), deep models suffer less from such smoothness constraints [5].

In spite of these successes, the research on DL is still far from complete. Deep networks face many problems, both old and new (e.g. vanishing gradients, diminishing feature reuse, degradation, and lengthy training time):

- **Vanishing gradients:** One of the most challenging problems is that of vanishing gradients, which is exacerbated by increasing depth [8]. The term refers to the phenomenon by which error gradients attenuate significantly as they are propagating backwards. With the gradient magnitude approaching to zero, the evolutions in earlier layers become stuck and saturated. It was proved in [66] that the gradients themselves are smaller after travelling only one layer from the output and their variance also decreases along the path. The issue is believed to originate from the back-propagation algorithm rather than the ANN itself. The gradient computation of conventional activation functions, such as the hyperbolic tangent or sigmoid function, compresses a large range of input to a very narrow region that is usually close to zero. Therefore, the propagation to early layers, which includes the repetitive multiplications of tiny gradients with small weights, results in a rapid vanishing of the gradient magnitudes. For further insight into the issue, the reader is referred to the detailed analysis by

Glorot and Bengio [60].

- **Diminishing feature reuse:** In contrast to the vanishing problem, this term refers to the washing out of information when propagating forwards, which has the similar cause (i.e. repetitive multiplication) [78]. The issue prevents later layers from utilising the features that are computed at earlier layers.
- **Degradation:** In [164], Kaiming et al. identified a problem in which test accuracy saturated and degraded quickly as more layers were stacked in the model. According to the authors, as opposed to the commonly held explanation, overfitting was not seen as the actual cause, because it happens when the model fits the training set too well, whereas they observed that training errors were also increasing. The authors referred to this problem as the degradation problem and claimed that the difficulty in optimization is in fact the main cause.
- **Long training time:** Heavy computational overhead is a well-known problem for ANNs, which has become a serious concern as networks have become increasingly deeper and larger [66]. The state-of-the-art networks involve a very large number of nodes and connections, which is problematic not only for model implementation but also for optimising parameters. In fact, the computation in both the forwards and backwards phases scales linearly with the network depth [78].

The trends in ML are undergoing a shift from feature engineering to network engineering [168]. Traditional handcrafted features become infeasible in large-scale applications, most probably owing to the dependency on human intervention. Recent models

tend to be more flexible, with the capability to be transferred between different tasks which, in turn, lessens the importance of problem-specific features. Transfer learning, for example, is an approach that creates a learner for a target domain by transferring knowledge from another model trained on a related domain [118]. One typical application is to build a cross-language system (e.g. translator). A model trained on a widely supported language (e.g. English) would be transferred into similar systems of other languages which have fewer labelled data (i.e. uncommon domain) [99].

Recent efforts by neural network researchers have been accelerated in multiple directions, from efficiently enlarging current models to introducing novel architectures. New activation functions, with reduced tendencies to saturate, such as ReLU [61] and softsign [10], were introduced as alternatives to traditional functions such as the sigmoid and hyperbolic tangent. DL has become increasingly feasible with the support of specialised hardware systems (e.g. multicore central processing units (CPUs), graphics processing units (GPUs) , and high-performance computing). The hype in DL is expected to enjoy a steady increase in the coming years, in both industry and academia. In this work, my interest is in drawing representational power from network structural design, which involves several arguably controversial opinions.

4.2 To Go Deep or Wide?

4.2.1 Deep Networks

Deep networks have been around since the 1980s, but these model were rarely used owing to infeasibility of optimization. In particular, networks introduced by Ivakhnenko in 1971 were described to be eight layers deep [84]. Fukushima’s Neocognitron made

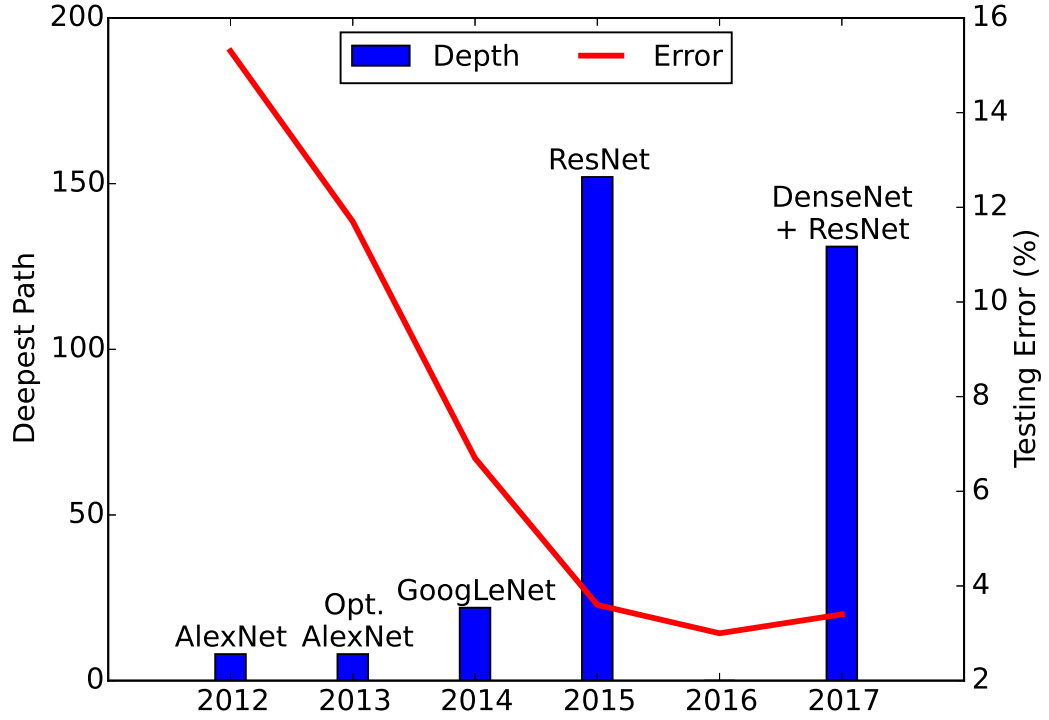


Figure 4.1: Evolution of Depth and Error in ImageNet Competition

Year	Models	Deepest path	Error Rate (%)
2012	AlexNet [88]	8	15.3
2013	Optimised AlexNet [174]	8	11.7
2014	VGG [46]	19	7.3
	GoogLeNet [142]	22	6.7
2015	ResNet [164]	152	3.6
2016	Ensemble of ResNets and Inceptions	N/A	3.0
2017	Dual Path Network [25]	131	3.4

Table 4.1: Details of ImageNet Winning Models

its first appearance in 1980 but with many similarities to modern deep architectures [55]. Actually, as argued in [63], the models of the 1980s worked quite well but were too computationally expensive for the hardware available at that time. The year 2006 was thus the beginning (or revival) of the DL era, when Hinton and colleagues introduced a breakthrough paradigm for training deep networks, thereby ending the criticism of their inefficient training [74]. From that point onwards, multiple models and algorithms were introduced, strengthening the superiority of deep networks over shallower counterparts. Around 2008, the trending research was on unsupervised pre-training [7]. Experiments showed that it helped to start supervised training closer to better local minima. Convolutional Neural Networks (CNNs), have become extremely popular in multiple domains (e.g. speech recognition [178], natural language processing [35]), and especially in computer vision with the notable success of AlexNet in 2012 [88]. In the ImageNet annual competition (i.e. one of the most competitive datasets in research), AlexNet is the pioneering deep model with only eight layers, which successfully improved the state-of-the-art record of 26.1% to 15.3%, details could be seen in Figure 4.1 and Table 4.1. Since then, progressively deeper CNNs have been investigated (e.g. VGG - 19 layers, GoogleNet - 22 layers) and have continuously broke the record (e.g. 6.7% in 2014) [46, 142]. A year later, the champion was Residual Network (ResNet) with a phenomenal depth of 152 layers. The latest competition (i.e. 2017) announced the final winning result of only 3.4% testing error, obtained with an ensemble of 98- and 131-layer deep networks.

In the breakthrough 2015 paper of ResNets, apart from the winning 152-layer model, the Microsoft Research team successfully trained an enormous 1202-layer net-

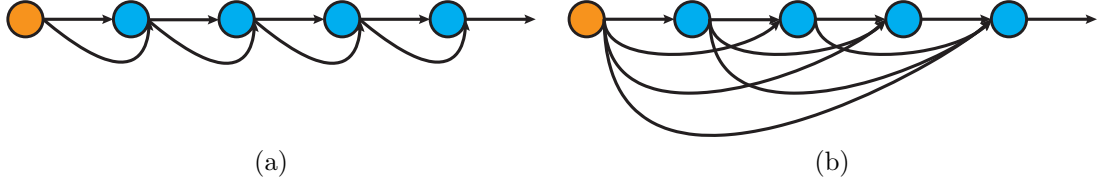


Figure 4.2: Illustration of (a) ResNet and (b) DenseNet blocks

work [164]. Despite the success from the perspective of training convergence, the model did not surpass the result (in terms of test accuracy), which they obtained with a shallower network of 110 layers. In fact, Highway Networks were the pioneering work on skip connections and the first to introduce a model with 100-layer depth [139]. ResNets followed the idea with only the identity shortcut. Motivated by the degradation problem, whereby performance degrades when stacking more layers in the network, the ResNet authors hypothesised that by stacking identity mapping layers on top of a learned shallow model, the resulting network should not produce any additional errors compared with the original one. Similarly, proposed model implemented a residual structure in which each micro-block involves both multiple nonlinear layers and an identity shortcut mapping the input straight to the element wise addition at the end. Therefore, once early blocks sufficiently fit the data, extended layers would simply bias their weights to embody identity pathways.

In late 2016, Huang et al. proposed DenseNet which was one core component of the winning model of ImageNet 2017 [78]. DenseNet produces comparable results to the state-of-the-art ResNet but was proved to be more computationally efficient [25]. Typically, DenseNet uses skip connections to bridge all layers directly together (see Figure 4.2 for comparison with ResNet). However, in order to preserve the feed-

forward nature, only paths that connect the current layer and its subsequent layers are allowed. Similar to ResNet, DenseNet takes advantage of shortcuts to enjoy the extra features learned in previous layers. In contrast to ResNet which uses summation as a merging function, DenseNet concatenates the incoming data [78]. In this way, it is said to produce the final decision based on the information learned at all layers. However, the obvious weakness of this approach is that the joint feature maps may grow linearly when the network goes deeper. This causes the number of parameters to explode, leading to enormous memory consumption. For this reason, the DenseNet feature maps used to be very narrow. Despite the above problem, feature reuse, as emphasised in the paper, is the key property of DenseNet that leads to its efficiency.

4.2.2 Wide Networks

- **Complexity:** Shallow networks are usually not expressive enough to encapsulate complex concepts [6]. The expressive power of a model is argued to be strongly correlated with network depth [66]. As mentioned above, the advantage of having multiple layers is that the network can learn features at various levels of abstraction. Deep models seem to be better at generalizing because they learn all the intermediate features between the raw data and output. However, standard shallow ANNs, such as MLPs, with single but sufficiently large hidden layers can theoretically approximate any function [94]. Increasing the width of networks is thus generally believed to improve accuracy significantly [134, 40]. However, this is also their main weakness. Fitting so well means that they become better at memorisation rather than generalization (i.e. more prone

to over-fitting). In other words, a very wide network may memorise all the possible input–output pairs but would not be able to make any proper prediction on unseen data. However, when being put into practice, shallower networks are harder to train compared to corresponding deeper networks, especially when being constrained by a parameter budget (which explodes exponentially to achieve similar generalizability) [173].

In fact, under similar parameter bounds, in the literature we can observe two contrasting trends: deep networks with narrow components and shallow networks with very wide layers. Contrary to conventional ResNets, the authors in [173] proposed an approach using networks that were wider rather than deeper. They observed that thin (bottle-neck) blocks are more favoured in the original version and tried to address the problem of identifying how wide the block should be. However, as shown in [40], the benefits of investing in widening a network decrease rapidly. Dauphin and Bengio tracked the number of errors removed per additional hidden unit (return of investment). They found that it would reach a plateau and even level off from that point onwards, which contradicts the arguably common expectation that it would at least remain constant. Despite that, the authors in [173] succeeded in training a competitive model with only 16 layers that outperforms all the deep (i.e. 152 layers) counterparts in the original ResNet paper.

- **Gradient flow:** In contrast to deep networks, shallower networks are believed to provide more efficient gradient propagation, of both the forward and back-

ward directions [78]. Therefore, these networks enjoy an advantage in execution and convergence speed. They are also less prone to the vanishing gradient problem, as there are fewer multiplication steps in computing the gradient at earlier layers.

- **Effective training:** Although recent breakthroughs in training very deep networks (e.g. more than 1000 layers) are remarkable achievements, whether deep networks are superior to wider (and shallower) models is still controversial. Since the introduction of VGG and AlexNet, many studies believed that deep networks with tiny kernels are the key to increasing the learning capability [25]. On the contrary, Veit et al., for instance, have denied the importance of very deep layers in ResNet [153]. They stated that the very deep architectures contradict the design principle found in biological systems, whereby a complex concept can be captured within just a few layers. Significantly, on a trained network, the authors experimented by removing random individual layers, and surprisingly found that the operation had a very modest effect on the performance. Instead, ResNet behaved smoothly corresponding to the number of layers being removed or shuffled, showing a loose connection between the paths as well as an ensemble-like characteristic. However, the study also pointed out the role of effective paths, which turn out to be the shorter paths in the model. The experiments also showed that the magnitude of the gradient decreases proportionally to the path length, with the result that long paths did not contribute much towards the learning (i.e. the vanishing gradient problem).

The identity shortcuts which allow the gradient to flow freely in the network are, at the same time, its weakness. ResNet is believed to be biased towards the identity pathways rather than the non-linear side. There is no guarantee that data would go through the residual block which means learning nothing valuable. As a consequence, it is possible that many blocks do not contribute much towards the final decision, showing a degree of redundancy. The statement is further strengthened through the investigation of Highway networks [139], which are the original form of ResNet, and where the probability of choosing the identity path is proven to be higher. Larsson et al. [90] proposed FractalNet, which is an alternative approach to ResNet, where they argued that the residual structure does not contribute much to the training of very deep networks, but rather the actual path length does. Instead of stacking similar blocks together, FractalNet emphasises modularity and introduces an approach to transform and enlarge networks that generates a fractal architecture (i.e. modularly enlarges the networks in both depth and width). Notably, in their experiments, the deepest column extracted from their model alone yielded very competitive results compared to the original ResNet. As mentioned above, the more effective paths in ResNets tend to be the shorter paths, whereas the performance of this deepest column (path) in FractalNet proves that it trains very deep networks more effectively.

- **Implementation:** Stacking up modules in a sequential fashion in ResNet does not bode well for parallelization and the utilization of GPU computation. There-

fore, wider components allow models to be more computationally effective by employing parallel computation solutions.

4.2.3 Multibranch Architecture

As stated by Xie et al., although improving performance by increasing capacity (both width and depth) is quite common, there are few methods that improve accuracy without creating extra complexity [168]. The study introduced an architecture based on the split-transform-merge strategy of Inception models [142], with an emphasis on simplicity, called ResNeXt. The authors argued that stacking modules in a simple vertical fashion reduces the choice of optimization. Thus, they split each module into subcomponents before aggregating at the end. This strategy has actually been adopted by several other studies [28, 155], which however required additional effort for augmentation or tailoring at each path (e.g. special preprocessing and pyramid resolution). As a result, ResNeXt achieved better accuracy compared to an equally large ResNet. In addition, the authors introduced cardinality, a new hyper-parameter to represent the number of subcomponents (i.e. transformations). The experiments showed that it is a better and more robust approach to improving accuracy by enlarging networks, rather than expanding only depth and width.

4.3 Conclusion

DL has contributed to multiple frontiers, achieving the state-of-the-art record in various different tasks. This chapter briefly reported the concept and the motivation behind it, as well as some of the current challenges in the field. In short, learning

different representations with increasing levels of abstraction is the key to the success of DL. Although DL achieved superhuman performance in a number of tasks, the research on DL is still far from completed, with multiple unresolved problems.

In addition, as discussed, the trend in DL research is moving towards network engineering, but the designing of better architectures is still a challenge. This chapter focused on the debate between the width and depth of the network, and the question of in which of these it is more beneficial to invest. Different opinions supporting both sides were reported together with recent advances and evidence. To sum up, within a parameter budget, a wider (and thus shallower) network provides a quicker answer, whereas the deeper counterpart enjoys better accuracy.

Chapter 5

Parallel Circuits

5.1 Motivations

In Chapter 4, DL was shown to be one of the most phenomenal research directions in AI today. The accuracy of DL models far surpasses classical approaches in multiple domains. Efforts in enlarging trainable networks, especially in terms of depth, have been paying dividends, but numerous studies have been conducted simultaneously to push forwards many frontiers [90]. In Chapter 3, I have compared various ML approaches over our specific VC datasets and proved that DL was the best candidate. Thus, this research is centred on DL or ANNs in general.

The contribution is motivated by the debates on network design, and directly by the problems I faced in the preliminary benchmark, particularly:

- **Reduce computational expense:** A serious problem was observed in the preliminary experiments, highlighting the fact that DL (and ANN approaches in general) took thousands of times longer than other models to finish training (see Chapter 3). Therefore, one of the primary research motivations was to address the lengthy training time of DL models. In addition, the under-

lying computation of DL is famous for the optimization of a massive number of parameters (see Chapter 4). Such tasks suffer from not only difficulties in fine-tuning but also heavy overheads of time and resources. In the scope of this work, I attempt to tackle the problem of computational expense rather than only the training duration.

- **Algorithm over hardware:** In the literature, there are a number of studies proposed to speed up the training, at both the hardware and algorithmic levels. The hardware-based solutions vary, ranging from GPU-accelerated computing [88] and distributed computing [42] to highly specialised hardware architectures such as field-programmable gate arrays [176] or neuromorphic computing [129]. However, the approaches that take advantage of modifications of learning algorithms tend to reduce the computational complexity. For example, Cong and Xiao modified the computation in convolution operations [36]. Jaderberg et al. [85] addressed the redundancy in feature maps, and Shi and Chu [131] attempted to remove the nonbeneficial zero operands. A combination of both types can be seen in [110], which traded the precision in arithmetic computation for speed. However, the former approaches usually require extra expenses and tailored implementations to match with the special components. Therefore, I decided to develop the solution without introducing any additional components.
- **Supporting parallelism:** Many research studies have been carried out to reduce the overhead. Hardware accelerators are one straightforward research avenues direction that have shown a lot of success. Taking advantage of GPU

computing with large tensors can speed up the computation by one or two orders of magnitude. The capabilities of computer hardware are no doubt increasing enormously, but the main computational power would be mostly drawn from parallelism techniques such as multi GPUs, multicore CPUs, or cluster computing. Unfortunately, the majority of state-of-the-art models (e.g. GoogLeNet, ResNet, DenseNet) are inherently sequential (i.e. each subsequent update requires the models to finish the computation on the current layer). As stated in [4], the current fashion in training (i.e. sequential with very deep networks) is becoming less efficient day by day. Parallelism is almost inevitable in DL. On one hand, the surplus of data, which is one of the main drivers to DL, is growing dramatically. On the other hand, it has been empirically shown that larger models could yield better performance. It is a fact that the capacity of a given hardware component (e.g. memory, CPU), or even a single computer is limited and would not be sufficient to fit either the model or data alone.

- **Width over depth:** There is still much debate over whether investing in width or depth would return more benefit. However, by its nature, a wider network would cope more efficiently with parallelism-based solutions owing to the ease of distributing the workload (i.e. simultaneously processing samples with nodes on separated devices). Deeper networks, in contrast, usually consist of very narrow components for the sake of complexity, which obviously does not favour parallelism. Nevertheless, from the results shown in [40], it has been proved that there exists wide networks with similar generalizability to their deeper

counterparts. However, one of the main critiques against wide networks is the difficulty in optimization. Usually, to achieve the same level of accuracy, wide networks require exponentially more components (or parameters). Therefore, in [40], as mentioned above, these wide networks are created by mimicking (i.e. approximating) the pre-trained deep networks, not by learning from scratch.

Despite the ground-breaking achievement of ResNets and its variants, the most effective guidelines for building these deep models have not yet been entirely determined. The literature has shown that the current models are not so efficient, as shown in Chapter 4. In ResNet particularly, a considerable proportion of residual modules do not contribute toward the final decision [153]. Therefore, it has been shown that deeper paths have little influence on the learning, while highlighting a degree of redundancy. Although later studies such as [90] pointed out that they could train the deepest path efficiently, the performance of very deep networks did not surpass that of the shallower models. In other words, over-investment in depth did not pay the dividends.

In general, the proposed model would:

1. Favour width more than depth to support the concept of parallelism and to reduce the degree of redundancy.
2. Not be limited to imitation approaches (i.e. mimicking pretrained models).
3. Not discourage the investment in depth as it has many helpful benefits apart from the narrow components. In fact, this would explode the model complexity, and thus an alternative solution to refine the model is

necessary.

- **Modularity** Recent research efforts on enlarging both the depth and width of networks have shown a continual improvement in different tasks. However, a gradual trend towards modularity is becoming more notable and showing much potential. Modularity also plays a significant role in parallelism, and thus it is expected to encourage this property in the model.

5.2 Parallel Circuits

Inspired by one of the nature’s solutions for heavy workloads, an alternative, more lightweight architecture was proposed, i.e. PC.

5.2.1 Biological Inspiration

The evidences for parallelism is revealed in various biological structures. For example, the human retina consists of multiple cell types, perceiving different perspectives of our surrounding (e.g. three types of cone cell correspond to three spectral regions in spectrum band or the rod cell is sensitive to low light levels) [157]. The projection of visual signals from our retina to our brain is also organised into various parallel routes. Each receptor is wired to a dedicated subcortical area, associated with a different visual feature (e.g. contour, brightness, colour, or motion direction) that processes simultaneously [62]. In fact, brain organisation, in general, seems to be guided by principles of parallelization and modularity. Although the advantages of these principles have long been discussed in both biological [123] and artificial domains [124], a complete understanding of how to implement them effectively remains elusive. This probably explains why these principles are not commonly found in competitive ANN benchmarks.

5.2.2 Definition

To form a PC architecture, a standard fully connected network (e.g. MLP) is divided vertically (i.e. layer by layer), forming a series of independent subnetworks called circuits. In general, Parallel Circuit ANNs (PCANNs) may consist of circuits with

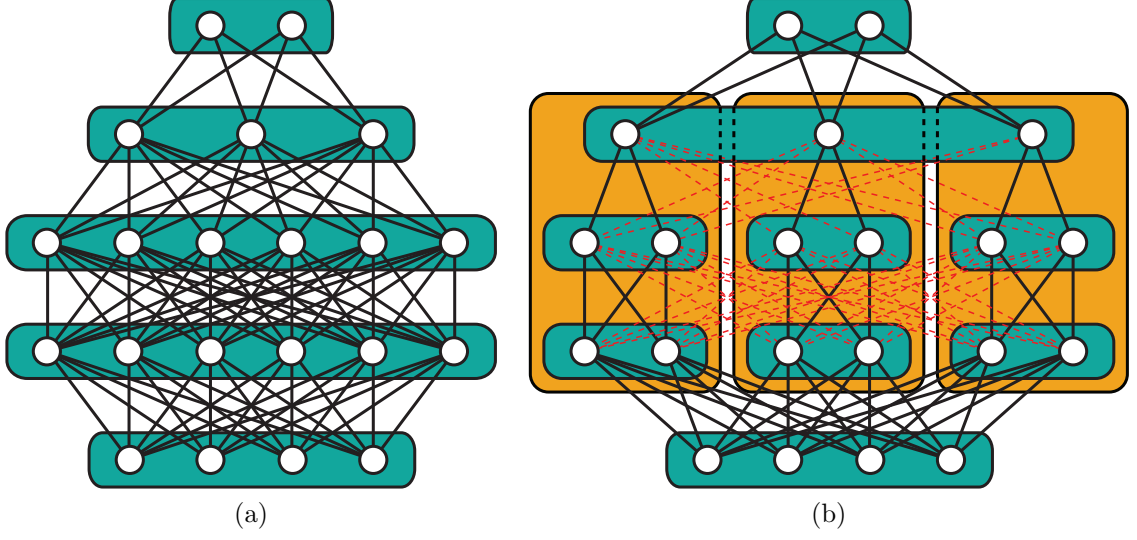


Figure 5.1: Illustration of ANN (a) without and (b) with PC architecture. Discarded connections are depicted as red dashed lines.

different widths and depths (refer to Figure 5.1 for a simplified diagram of a PCANN). It is worth mentioning that these circuits share the same input and output layers. In other words, the division is only applied to hidden layers. Thus, compared with MLPs with the same number of nodes, PC models reduce the number of connections in hidden layers by a factor of k , where k is the number of equal-sized circuits being used.

According to the definition of PC, for circuit k , the input sum for the hidden node i over layer l is computed as follows:

$$z_{k_i}^l = b_{k_i}^l + \sum_j^{n_k^{l-1}} y_{k_j}^{l-1} * \omega_{k_{ij}}^{l-1} \quad (5.1)$$

where n_k^{l-1} is the number of hidden nodes in layer $l-1$ of circuit k , $y_{k_j}^{l-1}$ represents the output of node j in circuit k and layer $l-1$, $\omega_{k_{ij}}^{l-1}$ represents the weight of the connection from node j in layer $l-1$ to node i in layer l , in circuit k , and $b_{k_i}^l$ refers to the bias of node i in layer l , in circuit k . The first hidden layer is actually a special

case, as all circuits share the same input layer ($l = 0$). Therefore,

$$y_{1j}^0 = y_{2j}^0 = \dots = y_{kj}^0 = x_j \quad (5.2)$$

The input for the output layer of nodes (referred to as ‘out’) is equivalent to a hidden layer that concatenates the last hidden layers of all circuits (penultimate layer or PL).

$$y^{out} = (z_0^{PL}, z_1^{PL}, \dots, z_k^{PL}) \quad (5.3)$$

Considering the case of a PCANN with two circuits, where the weight matrix for the last layer L is $\mathbf{W}$ and its input is X , we have at L:

$$X = [X_1, X_2] \quad (5.4)$$

$$W = [W_1, W_2] \quad (5.5)$$

where $[,]$ refers to horizontal concatenation. The element wise addition at L is then:

$$W^T X = [W_1, W_2]^T [X_1, X_2] = W_1^T X_1 + W_2^T X_2 \quad (5.6)$$

Thus, by splitting the network into two circuits, and performing the sum of element-wise additions of distinct weight matrices, it ensures that the concatenation of information coming from both circuits is done correctly, allowing the discrete computation of each circuit up to the last layer. Therefore, Eq. 5.1 for the output layer could be rewritten as:

$$z_i^{out} = b_i^{out} + \sum_k \sum_j^{n_k^{PL}} y_{kj}^{PL} \times \omega_{kij}^{outPL} \quad (5.7)$$

5.2.3 Hypotheses

In a two-hidden-layer network, denote the number of nodes in the input layer, output layer, 1st, and 2nd hidden layers as I , O , H_1 , and H_2 , respectively. Then, the number of free parameters (ignoring bias) are:

$$I \times H_1 + H_1 \times H_2 + H_2 \times O \quad (5.8)$$

For each circuit in a k -circuit model:

$$I \times \frac{H_1}{k} + \frac{H_1}{k} \times \frac{H_2}{k} + \frac{H_2}{k} \times O \quad (5.9)$$

Multiply by k , the total number of parameters is:

$$I \times H_1 + \frac{H_1 \times H_2}{k} + H_2 \times O \quad (5.10)$$

Based on Eq. (5.10), it is clear that the reduction in the number of connections takes place between the hidden layers. This fact also defines a crucial assumption of PCANNs, that the network should have at least two hidden layers. Otherwise, there would be no benefit (i.e. reduced complexity) from the subdivision of the networks .

In general, the research on PC architecture is centred on two main hypotheses:

1. **Speed boost:** It is reasonable to expect speed gains from the fact that PCs use fewer connections compared with the original fully connected networks. In addition, it is clear that PCs would be more optimised for parallel processing as the inter-circuit communication is highly restricted.
2. **Generalizability boost:** It is probably more controversial to hypothesise that PC architectures are advantageous in terms of generalization. In [104], a re-

ported ANN training protocol was characterised by problem decomposition (repeatedly switching between multiple goals, each with several subgoals), leading to the emergence of modularity. Provided that modularity is already achieved through the independence of PCs, I conversely hypothesise that this architecture should exhibit the property of automatic problem decomposition. Moreover, and in accordance with the divide and conquer principle, this automatic problem decomposition can further be hypothesised to improve generalization [86].

5.2.4 Learning Algorithm

There is no significant change in the control flow of the backpropagation (BP) algorithm (see Algorithm 1). In the forward phase, each training sample is pushed into the circuits through the shared input layer. As circuits are independent except for the input and output layers, most of the circuit computation can be entirely parallelised and performed separately. The forward phase stops as soon as information reaches the last hidden layer (penultimate layer - PL). To compute the final output, each circuit contributes its last hidden layer as part of the input for the output layers activation functions. Theoretically, there would be a concatenation operation before entering the last layer (see Eq. (5.3)). However, the parallel pipeline was maintained by multiplying each circuit with an associated submatrix and aggregating at the end (see Eq. (5.7)).

In the backward phase, the computed error and gradient of the whole network is duplicated to each circuit in such a way that they are seen as individual networks.

From that gradient, the delta values for both weights and biases are computed layer by layer for the updates. Similar to the forward phase, this process could be entirely parallelised across circuits.

Algorithm 1: Gradient Descent on Parallel Circuits

```

input: number of circuits  $n$ 
         weight matrix of reference MLP model  $\mathbf{W}$ 
         bias vector of reference MLP model  $b$ 

begin
  /* initialise */
  for each circuit  $i$  do
    for each layer  $k$  do
       $\mathbf{w}_i^k \leftarrow$  initialise as weight matrix of size  $\|\mathbf{W}^k\|/n$ 
       $b_i^k \leftarrow$  initialise as bias vector of size  $\|b^k\|/n$ 

     $\mathbf{w}^{out} \leftarrow$  initialise as weight matrix of size  $\|\mathbf{W}^{out}\|$ 
     $b^{out} \leftarrow$  initialise as bias vector of size  $\|b^{out}\|$ 

    /* main */
    for each training batch  $x^0$  do
      /* propagate forward */
      for each circuit  $i$  do
        input for first hidden layer  $x_i^{inp} \leftarrow x^{inp}$ 
        propagate across  $k$  layers
        calculate output of last hidden layer  $a_i^{pen}$ 

        input for output layer  $x^{out} \leftarrow \text{concat}(a_i^{pen})$ 
        output  $a^{out} \leftarrow h(b^{out} + \mathbf{w}^{out} x^{out})$ 

      /* propagate backward */
      calculate error term  $E$ 
      for each circuit  $i$  do
        imaginary output error term  $E_i \leftarrow E$ 
        backpropagate
        update

```

5.3 Related Works

The idea of parallelism is not new in the field of neural networks. Although past studies involve some form of modularity and parallelisation, they do not implement the exact type of parallel architecture we proposed. In fact, my approach might be interpreted in different ways. On one hand, it is closely related to techniques in *cluster computing*. Node parallelism [145], network parallel training [39] or vertical partitioning [141] are similar approaches that slice the network vertically, whereby each subnet is handled individually by a processing element (PE) of the cluster. Because the training sample is processed by all PEs in parallel, communication between them is necessary, which thus creates some extra overhead. Significantly, these models maintain the networks total connections and speed up processing by simultaneously updating these subnets weight matrix. In contrast, my approach eliminates some of the inter-circuit connections, reducing the computational load and consequently increasing the speed of training and classification.

On the other hand, the PC approach could be viewed as a form of *pruning*. In the literature, this technique has been widely used to simplify large networks, by identifying and removing redundant nodes and/or connections [20]. This pruning phase is typically followed by a retraining step to ensure the model can adapt to the reduction [69]. The proposed approach interprets the solution differently. It actually did prune the network, but not to eliminate weak nodes. As mentioned previously, I believe that one of the advantages of my approach pertains to automated problem decomposition, whereby each circuit learns to specialise on a subcomponent of the

given problem. My ‘model pruning’ separates circuits, forcing them to work semi-independently. Typically, both standard pruning techniques and my approach aim to reduce the computational cost without having a negative effect on generalization. In fact, it was argued in [79] that pruning techniques are not always practical and can often only be done through a computationally expensive process of trial and error. However, this does not apply to our PC architecture, as the pruning is applied prior to the training.

Some notable exceptions include CNNs [93, 87], which also draw inspiration from the visual system’s highly parallel nature. One crucial difference from this approach is that it does not enforce weight sharing, but instead, focus on the semi-independent learning across circuits. However, these studies support the hypothesis with respect to modularity and generalizability. In fact, the approaches described in [107, 30] are more closely related to my assumption. Mitra and his colleagues created an ensemble of deep CNNs with model averaging [107]. Their computational unit was a column (deep CNN), which was designed to be independent, and where each column handled a duplicate of the training sample or a version distorted by different filters. For the work on the Lidar dataset [30], they implemented a hybrid approach involving ANNs, SVMs and RBFs. The key aspect of their design was that they structured the ANNs to process information in parallel before feeding into SVMs for a final decision. The same training data was used for all ANNs but it underwent different encodings. In general, these findings show that we can improve generalization by dividing models into modules. However, these modules should work independently (or at least semi-independently) in order to specialise on different feature dimensions. One crucial

difference between my work and [107] is that I attempt to unify the architectural and training details of this modular approach into a single architecture consisting of semi-independent PCs with shared input and output layers.

Both the motivation and underlying concepts of ResNeXt [142] are closely related to my PCs. We differ in two main points. First, ResNeXt aims at exploring another dimension (i.e. cardinality) when the returns from width and depth investment start to level off, whereas my work tackled lengthy training times. However, both studies have the same focus on the effective use of parameters. ResNeXt improves accuracy without increasing complexity, whereas I trimmed the model without harming its performance, not to mention that it is expected for an accuracy gain through more effective learning. Lastly, this work approached more general model, MLP, instead of ResNet to ensure the gain in generalization is rooted in the architecture.

5.4 Proof of Concept

5.4.1 Methodology

The experiments reported in this section are designed to be the proof of concept for PCs. The performance of the PCANN approach are compared with the traditional (single-circuit and fully connected) MLPs, in an attempt to prove or disprove the two hypotheses regarding performance boost stated above (see Section 5.2.3).

The baseline model is a MLP with fully connected adjacent layers. More specifically, MLPs were evaluated using two architectures with different sizes, each with two hidden layers. They are differentiated in the design: in one architecture, both hidden layers consist of 50 nodes, whereas the other has 500 nodes in each layer. For

Family	Abbr.	Architecture	Circuit	Remark	Type
100 nodes	SC1	[50-50]	1	Full connectivity	Non PC
	RC1	[50-50]	1	Reduced connectivity (10%)	
	PC1	[5-5]	10	10 Circuits	PC
	VC1	[30-30] [15-15] [5-5]	3	Variable-sized Circuits	
1000 nodes	SC2	[500-500]	1	Full connectivity	Non PC
	RC2	[500-500]	1	Reduced connectivity (10%)	
	RC2a	[500-500]	1	Reduced connectivity (50%)	
	PC2	[50-50]	10	10 Circuits	PC
	EPC	[5-5]	100	100 Circuits	
	VC2	[300-300] [150-150] [50-50]	3	Variable-sized Circuits	

Table 5.1: Network Abbreviations and Designs

Name	Abbreviation	Attributes	Classes
Wisconsin Breast Cancer	BC	30	2
Breast Tissue	BT	9	6
Climate Model Simulation Crashes	SC	18	2
Glass Identification	GL	9	6
Leaf	LF	14	30

Table 5.2: Dataset Details

Dataset	Training	Testing	Total
BC	200	369	569
BT	50	56	106
SC	250	290	540
GL	100	114	214
LF	150	190	340

Table 5.3: Training and Testing Set Generation Details

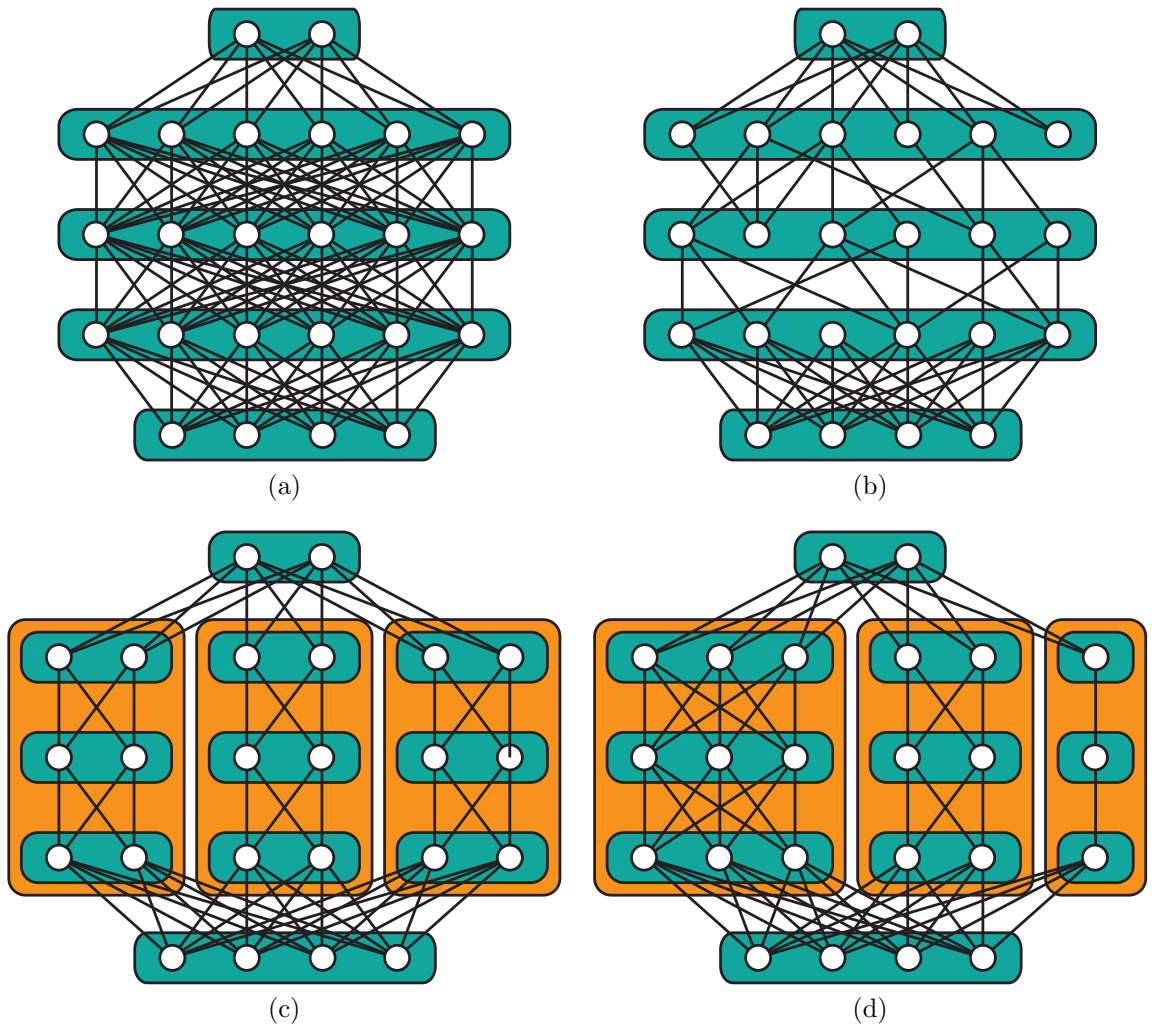


Figure 5.2: Illustration of (a) SC, (b) RC, (c) PC, and (d) VC designs. The number of nodes and circuits are simplified and do not follow precisely Table 5.1.

the sake of convenience, they are notated as the ‘100-node’ and ‘1000-node’ families, respectively.

For the PC approach, three versions were implemented, i.e.: (i) 10 equal-sized circuits, (ii) 100 equal-sized circuits (only for the 1000 node family), and (iii) three variable-sized circuits. When comparing traditional and PC architectures, the total number of nodes for each layer (across circuits) is always set to be the same across conditions. In order to further enhance the fairness of our comparison, I included two modified MLPs, which were not only controlled by the number of nodes, but also by the number of connections: one with 10% retained connections (i.e. equivalent to 10 circuits), and the other with 50% connections (i.e. two circuits), relative to the fully connected configuration. The reduced connectivity networks (RCs) are generated by randomly trimming the connections by 90% (or 50%) prior to the training. It is worth mentioning that the connections are trimmed by applying masks to the hidden layers only. The masks also remain unchanged during the training. In comparison with the RC network, we attempt to prove that the proposed PC approach is different from network pruning and the performance gain (in both speed and accuracy) are achieved from the modular structure rather than purely weight pruning.

Regarding the training data, five datasets from the UCI machine learning repository [20] were chosen to be used in the comparison, namely Wisconsin Breast Cancer (BC), Breast Tissue (BT), Climate Model Simulation Crash (SC), Glass Identification (GL), and Leaf (LF). The training and testing sets are randomly populated from the original sample pool. The sizes were defined to be comparable between the two sets and the training sets are the multipliers of batch size. The network designs are

shown in Table 5.1 and Figure 5.2, whereas the details of these datasets can be seen in Table 5.2 and Table 5.3.

All of the experiments in the comparison were carried out under the same parameter conditions: 0.01 learning rate, 0.4 momentum, and within 100 training epochs. The hyper-parameters were defined according to our experiments, for best average result. Because the performance was expected to manifest a high degree of variance, caused by random initialisation and training stochasticity, 100 runs were executed for each network configuration/dataset pair. The performance measures adopted were obtained by averaging the results over these runs. The evaluation metrics include:

- Mean Accuracy: Average test accuracy over 100 runs.
- Best Accuracy: Highest test accuracy achieved among 100 runs.
- Mean Training Time: Average training duration over 100 runs.
- Best Training Time: Lowest training duration achieved among 100 runs.
- Mean Stopping Epoch: Average epoch over 100 runs at which the training stops due to meeting early stopping conditions (e.g. no significant improvement for many epochs).
- Best Stopping Epoch: Earliest epoch at which the training stops among the 100 runs.

The experiments were executed on an Intel Xeon E5420 @ 2.50 GHz with 8 GB RAM.

Model	Mean Training Time (s)	Best Training Time (s)	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC2	19.824	11.914	0.690	0.777	76.344	47.0
RC2	25.744	18.556	0.731	0.802	84.610	64.0
RC2a	27.021	18.219	0.708	0.774	79.594	59.4
PC2	10.683	7.998	0.724	0.802	83.944	66.4
EPC	71.639	55.334	0.703	0.790	84.194	62.4
VC2	11.843	7.953	0.707	0.788	80.332	55.8

Table 5.4: Performance of the 1000-Node Family across Five Datasets

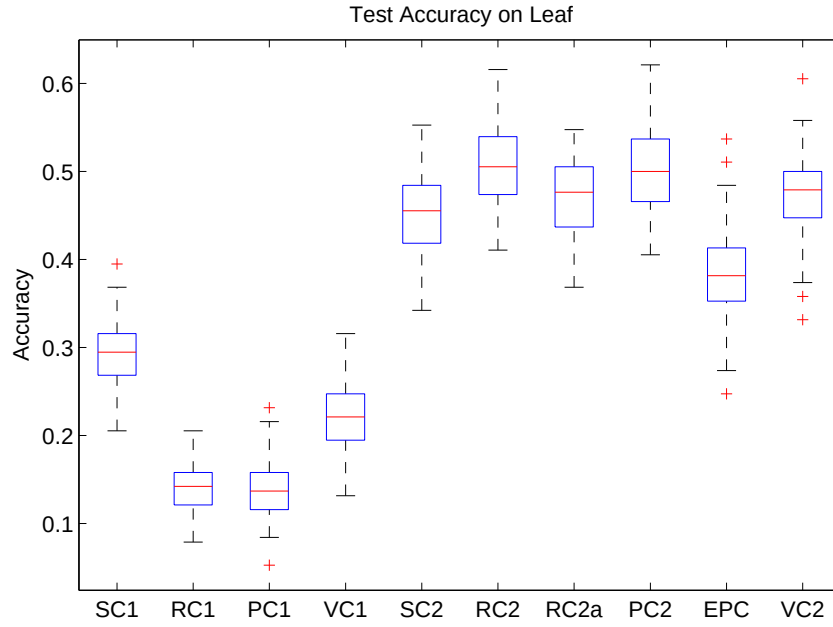


Figure 5.3: Test Accuracy on Leaf Dataset

5.4.2 Results

5.4.2.1 1000-node Networks

With an average test accuracy of 71.1% across five datasets, the 1000-node family exhibited much better performance than the small networks (i.e. 64.9%). In fact, multiple PC implementations (PC2, EPC, and VC2) of this particular family exhibited an overall robust performance. PC2, EPC, and VC2 achieved 72.44%, 70.28%, and 70.71% on average, respectively, whereas SC2 only scored 68.97% (see Table 5.4). In all the datasets, the mean accuracy of most PC classifiers was higher than that of SC2 (i.e. fully connected single-circuit condition). The only exception was on the EPC–Leaf setup, which yielded an accuracy rate of 38.28%, compared with 44.87% for SC2. Taking extreme measurements into account (i.e. 25–75% percentile, median, best performance), the dominant result was unchanged (as seen in Figure 5.3). Except for the case mentioned above, all other boxes representing the 25–75% percentile and medians for PC implementations were above the level of its corresponding SC2. The upper whiskers (i.e. best performance), with an average of 80.2%, 79%, and 78.78% (PC2, EPC, and VC2 respectively), were also higher than for SC2 with only 77.69%.

Regarding training time, there is a consistent trend across all datasets, showing that the PC2 and VC2 architectures confirm our speed hypothesis. For either of these two implementations (i.e. PC2 and VC2), there was an average of approximately 40% reduction in training time compared to the non-PC approaches (i.e. SC2 and RC2/RC2a), whereas PC2 tended to be slightly faster than VC2 (see Table 5.4).

Model	Mean Training Time	Best Training Time	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC1	1.411	1.204	0.692	0.755	88.212	80.4
RC1	1.540	1.011	0.633	0.699	90.058	69.8
PC1	7.865	5.711	0.617	0.695	90.232	69.8
VC1	2.910	2.499	0.657	0.729	87.902	79.0

Table 5.5: Performance of the 100-Node Family across Five Datasets

However, for the remaining PC implementation (i.e. EPC), the training duration unexpectedly increased taking approximately three times longer than the SC2 version. This trend occurred in all five datasets, where EPC exhibited an average training time of 71.638s compared with 19.824s for SC2. Further investigations are presented in the following Discussion section, and more details on EPC are included in Appendix A.

5.4.2.2 100-node Networks

In contrast, the 100-node networks depicted PC architectures in a different light. BC was the only dataset for which the SC1 exhibited worse accuracy than the PC variants. However, the difference between SC1 (i.e. 95.58%) and the nearest PC implementation (i.e. VC1: 95.64%) is very minimal, approximately 0.06%. In the remaining datasets, SC1 totally dominated the tests. Both the mean and the best performances of SC1 were higher than the average of the four PC classifiers and any of them individually.

Taking training time into account, it is obvious that our hypotheses were disproven in this case. Across all five datasets, the fastest classifier was SC1, followed by RC1 with 1.411s and 1.54s, respectively. It took the PC1 approach approximately six

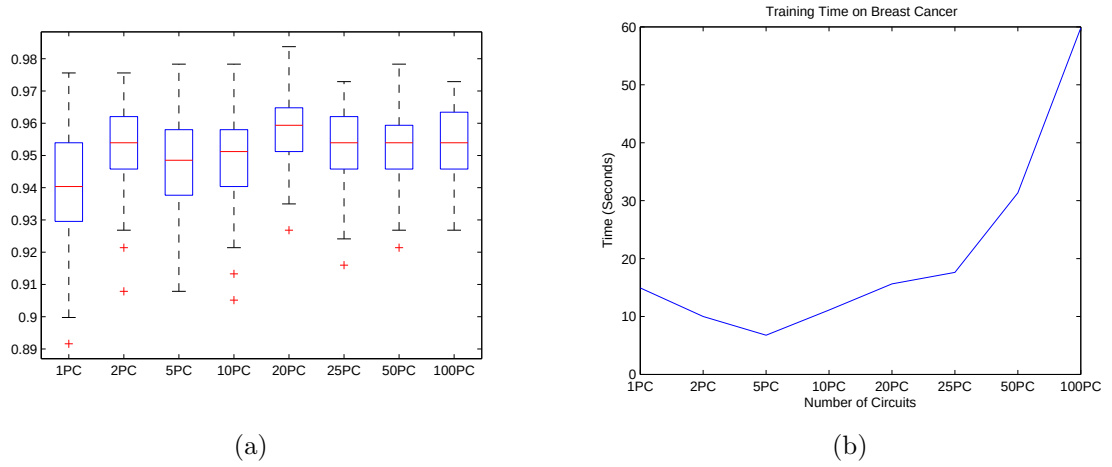


Figure 5.4: Comparison of Different Numbers of Circuits on Accuracy (a) and Training Time (b) on the Breast Cancer Dataset.

times longer than SC1 to finish its training. Regarding the convergence rate, VC1 performed best with an average of 87.9 epochs needed to reach a plateau. SC1 was the runner up with a slightly larger average number of epochs (i.e. 88.21). However, examine the results in more detail, only for BC did SC1 take fewer epochs than the other PC implementation (i.e. PC1) to reach a stable state. For all other datasets, PC1 surpassed it.

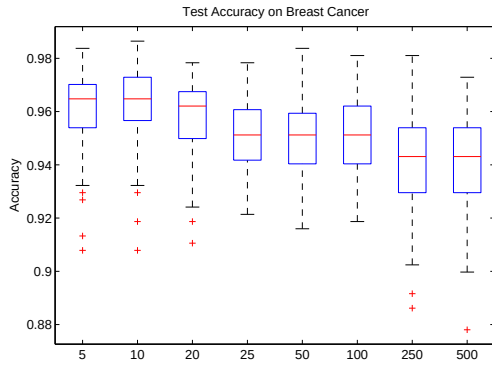
5.4.3 Discussion

The results in Table 5.5 and 5.4 provide positive evidence in support of my hypotheses, particularly in the context of large networks (i.e. 1000-node family). They showed that the PC approach provides a good trade-off. PC2 achieved the 2nd place in terms of test accuracy and the 1st place in terms of speed. Although the champion in accuracy is a non-PC approach (i.e. RC2), PC2 showed an average close behind with a difference of only 1% and a similar record for the best accuracy. Across all datasets, two of three PC implementations (PC2 and VC2) showed consistently better

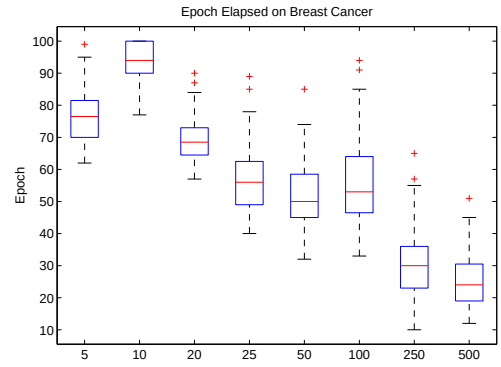
performance in both accuracy and training time. Unfortunately, EPC did not perform well, especially in terms of speed.

Owing to the observations of EPC, another experiment was conducted in which the number of circuits were controlled, in order to investigate the reason behind EPC’s speed underperformance. The number of nodes was maintained at 1000, but divided into 1 (i.e. single-circuit (SC) or fully connected MLP), 2, 5, 10, 20, 25, 50, and 100 circuits. With respect to accuracy, the results showed that PC variants outperformed SC, regardless of the number of circuits (Figure 5.4a). Although the differences are consistent, they are not very large, most probably because the easy dataset (BC) was chosen for the experiment. In terms of training speed, the interesting observation here is that the training time was optimal (minimal) at five circuits, beyond which point, training time increased, and even surpassed the SC case, from 20 circuits onwards (Figure 5.4b). Note that the training time for the PC condition is better than that for the SC condition, for the 2-, 5-, and 10-circuit setups.

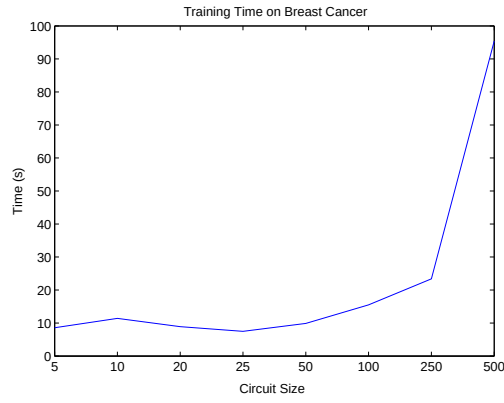
Because none of the PC implementations demonstrated any clear advantages for relatively small networks (i.e. 100-node family), I further investigated this issue with an additional experiment. In contrast with the previous experiment, the number of circuits was kept constant at 10, and varied the size of hidden layers (i.e. [5-5], [10-10], [20-20], [25-25], [50-50], [100-100], [250-250], and [500-500]). The results show that the best performance was obtained for the 10-node condition (96.3% mean performance and 98.64% for the best network). However, from that point onwards, the accuracy slowly reduced, to 94.02% for 500 nodes per layer (see Figure 5.5a). Regarding the training time, although it fluctuated slightly between 5 and 50 nodes, the general



(a)



(b)



(c)

Figure 5.5: Comparison of Different Sizes of Circuit on Accuracy (a), Stopping Epoch (b) and Training Time (c) on Breast Cancer Dataset.

trend was increasing (see Figure 5.5c). However, as the layer size increased, despite the fact that it cost more time to train, the results show that the number of epochs needed to converge decreased. At 10 nodes per layer, networks needed an average of 93.86 epochs to converge, which is also the highest value for this measurement. For 500-node networks, only 25.83 epochs were needed on average for convergence to occur (see Figure 5.5b). However, as the accuracy dropped, this phenomenon might be explained by 500-node networks saturating early, without converging to a good minima.

Combining the results of the two experiments, one might conclude that the reduction in training complexity cannot be represented linearly but rather with a function with peaks and troughs. As the network size was controlled in the first experiment, the more circuits we adopted, the smaller each circuit was. Thus, it is clear that PC design requires an appropriate size for the circuit to work properly (e.g. to decrease training time). Instead of using a ‘one-shot’ large matrix calculation, we implemented multiple computations of small matrices, which involves additional overhead (e.g. for separate iterations). This is an implementation issue rather than an algorithmic one. In particular, conventional propagation includes only a few operations (e.g. multiplication and addition) at each layer, with large operands (e.g. weight matrix or input/output vector). However, in the case of the PC approach, the order of these tensors is increased by 1, adding the circuit index. Owing to the implementation, at each layer, although the task is lighter with much smaller operands, it requires extra, duplicated work (e.g. saving and fetching data) for iterating the computation through each circuit. This overhead is obviously increased with the number of circuits, and

eventually nullifies any advantages accrued from a smaller number of connections. According to Figure 5.5c, the training times for layer sizes from 5 to 50 are not very different, as manifested by the fluctuating trend. This might partially contribute towards an explanation of the underperformance of the 100-node family. Because the original network size was already small, reducing it by a factor of k (as in the formula in the Methodology section) equates to only a small number of connections, regardless of the value of k . Thus, the absolute difference in both size and computing complexity between fully connected single-circuit and PC designs is not too large. On one hand, the reduction in the number of connections is not large, while on the other hand, we incur extra cost for the above-mentioned overhead. This may be one of the main causes for the relatively poor speed performance of PC networks with 100 nodes. In other words, subnetworks/circuits should be large enough to compensate for the cost of this multicircuit overhead.

5.5 Sparsity Induction

5.5.1 Motivation

One of the main hypotheses is that the proposed network topology would encourage automatic problem decomposition, in a manner analogous to biological neural systems. This assumption is critical, as the model capacity has been reduced after pruning the connections. I believe the modularity property would help the model to learn more efficiently, and thus recover the generalization power. However, the above experiments have shown that the PC networks performance is somewhat unpredictable, especially in smaller networks. Although the gaps in training speed was

explained by the trade-off with system overhead, the insights pertaining to generalization performance were insufficient. For further insight on this performance, an investigation on the sparsity of the PL was conducted, which is the concatenation of the circuits' final layers.

The relationship between modularity and sparsity has been mentioned multiple times in past studies. In [12], they mentioned that a highly modular network could be divided into subgraphs with strong inner connections but few to no interactions between them. Clune et al. supported the above statement with the finding of a negative correlation between the network density and modularity [33]. In other words, densely/fully connected networks might actually harm the network modularity. On the contrary, sparse networks would have fewer active connections, similar to PC architectures with pruned connections and thus, favouring the aforementioned property. Livingstone et al. stated that the sparsity of a network would be improved as the number of connections is reduced, leading to an increase in modularity [100]. More details on the relationship between modularity and sparse/dense networks can be seen in their paper.

Similar to the idea behind DropOut, the semi-independent processing of PCs should prevent the co-adaption of nodes, especially at the PL, where they contribute together to the final output. As in the paper of Baldi and Sadowski, the authors explained that DropOut eliminates excessive dependencies between any two neurons [2]. Therefore, the sparsity measurement could give us an insight pertaining to inner layer dependencies, i.e. whether they are evolving together or not.

The experiment consists of two parts: an analysis of sparsity of PC architecture

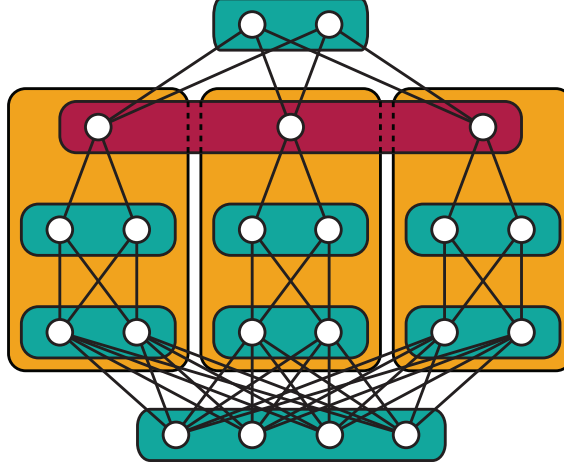


Figure 5.6: Illustration of Single-Node Penultimate Layer Architecture. SNPL is Depicted in Dark Red Colour.

and a benchmark of different sparsity induction techniques to enhance the condition.

5.5.2 Sparsity Analysis

5.5.2.1 Methodology

In the benchmark, the sparsity of the model are measured by computing Kurtosis value over the PL nodes. However, the common sparsity metrics used to be applied to a set of nodes instead of circuits, which means the measurement will not relate the sparsity between circuits well, but rather between the nodes of the PL. Therefore, I modified the configurations used in previous experiments by introducing a special hidden layer ahead of the final output layer, where each hidden node is associated with a particular PC in the network. This feature was called the single-node penultimate layer (SNPL), which potentially acted as a switch-like output for the PC (see Figure 5.6). The sparsity metrics (i.e. Kurtosis) would be then applied to this layer and each node would be the representation of its associated circuit. According to [114],

the Kurtosis value could be computed as follows:

$$k = \frac{1}{n} \sum_{i=1}^n \frac{(x_i - \bar{x})^4}{\sigma^4} - 3 \quad (5.11)$$

where n is the number of nodes, x_i is the output of node i , whereas $\bar{x}$ and σ are the mean and standard deviation, respectively. Kurtosis is actually a metric to analyse the peakedness in a distribution. A positive value would represent a distribution with a very high peak and heavier tail. In other words, a large proportion of the nodes are inactive, showing a high degree of sparsity.

The experiment was conducted over a set of networks similar to the previous benchmark (i.e. 100-node and 1000-node families), including both normal architectures and those with SNPL. For the PC condition, only models with 10 circuits were taken into consideration. The benchmark was evaluated over two datasets (i.e. BC: easy and LF: hard).

Furthermore, another test was performed in which we varied the number of PCs (i.e. 2, 5, 10, 20, 50, and 100 respectively) to see how the PC architecture affects the sparsity of models. In this set of experiments, all networks belonged to the 1000-node family, including the SNPL architectures.

5.5.2.2 Results and Discussion

The results in Table 5.6 and 5.7 clearly justify our theory, where the PC approach produces better sparsity than SC in the majority of setups, regardless of the topology and dataset. With respect to the variation of PC count, as shown in Table 5.8, sparsity is shown to improve generalization, in accordance with the ascending number of circuits. It could be concluded that PC design is a type of regularization similar

Architecture	Circuit	Accuracy	Sparsity	Remark
[100 100]	1	0.8958	−1.6901	Small SC
[10 10]	10	0.7661	−1.3782	Small PC
[100 100 10]	1	0.9047	−1.6938	Small SC + SNPL
[10 10 1]	10	0.8869	−1.6158	Small PC + SNPL
[1000 1000]	1	0.8955	−1.1647	Large SC
[100 100]	10	0.8831	−1.1971	Large PC
[1000 1000 10]	1	0.9091	−1.3932	Large SC + SNPL
[100 100 1]	10	0.9012	−1.1708	Large PC + SNPL

Table 5.6: Sparsity Benchmark on Breast Cancer Dataset (Bold for Better Sparsity).

Architecture	Circuit	Accuracy	Sparsity	Remark
[100 100]	1	0.362	−1.6438	Small SC
[10 10]	10	0.6196	−1.6	Small PC
[100 100 10]	1	0.1435	−1.6244	Small SC + SNPL
[10 10 1]	10	0.037	−1.2539	Small PC + SNPL
[1000 1000]	1	0.3557	−1.8981	Large SC
[100 100]	10	0.485	−1.6521	Large PC
[1000 1000 10]	1	0.1262	−1.7089	Large SC + SNPL
[100 100 1]	10	0.1357	−1.6393	Large PC + SNPL

Table 5.7: Sparsity Benchmark on Leaf Dataset (Bold for Better Sparsity).

Architecture	Circuit	Accuracy	Sparsity
[500 500 1]	2	0.8644	−2.75
[200 200 1]	5	0.8924	−1.8178
[100 100 1]	10	0.8989	−1.6543
[50 50 1]	20	0.902	−1.5932
[20 20 1]	50	0.9019	−1.3026
[10 10 1]	100	0.9086	−1.011

Table 5.8: Sparsity over Variants of PC Count on Breast Cancer Dataset (Higher is Better).

to DropOut. Its nodes are trained with high sparsity, forcing only a limited number of nodes to be activated at a time.

Despite the fact that the above benchmark has shown the sparsity advantage of PC over SC, the contribution to improving classification accuracy is still minimal. By observing the sparsity of the PL, it could be recognized that Kurtosis values are still negative. Ideally, one would expect them to be positive (i.e. the distribution would have a high peak at 0, which means most of the nodes would be inactive). Thus, although PC structure alone can improve the sparsity of a network, it is still insufficient to prevent co-adaption. In order to tackle this problem, I conduct experiments to further amplify the sparsity of the network.

5.5.3 Sparsity Enhancement

5.5.3.1 Methodology

Owing to the low degree of sparsity observed in plain PC models, attempts were made to improve the condition by implementing two classical penalties (i.e. L1 and L2). Several experiments were conducted to determine which regularizer was the most suitable implementation for the PC design. Apart from the two types of regularization, I also varied the configurations by penalising on either weights or activation values. A modification was included to restrict the sparsity constraint (i.e. penalty) to only the PL layer. The network architectures were similar to the eight setups in the previous benchmark that analysed the sparsity of plain PC.

L1-norm and L2-norm are regularization terms that are added into the loss func-

tion, defined as follows:

$$L1 = \lambda \sum_{i=1}^k |x_i| \quad (5.12)$$

$$L2 = \lambda \sum_{i=1}^k (x_i)^2 \quad (5.13)$$

where x_i might be either the weight or activation value of node i , depending on the experiment; k is the number of nodes; and λ is the penalty coefficient.

Taking advantage of the modularity of PC, I attempted to improve the sparsity conditions by varying regularization with random penalty coefficients (i.e. λ as in Eq. (5.12) and Eq. (5.13)) for each circuit, instead of using a constant. In particular, classical approaches introduce into the loss function a sparsity term (i.e. L1-norm or L2-norm), weighted by the λ coefficient. By adjusting this coefficient, it can determine how sparse the network is (i.e. determine how strictly the networks to be bounded to the sparsity constraints). Usually, this coefficient is unique and is multiplied after computing the gradient of the magnitude (L1) or squared value (L2) for each weight or activity, over the whole network. However, by varying the λ coefficient in the computation of the penalty at each circuit, I can manipulate the network to show different sparsity in different PCs. PCs that have large coefficients would have bigger penalties, becoming strictly sparse, whereas other PCs with lower λ might allow their nodes to evolve together. In fact, this type of sparsity penalty is bounded to each individual circuit and only manipulates the sparsity inside the circuit. Therefore, by applying to the SNPL structure with the ‘switches’ of circuits, it was casted to control the sparsity of all the circuits.

Regarding the implementation of the random coefficient, Eq. (5.12) and Eq. (5.13)

could be rewritten as:

$$L1 = \sum_{i=1}^k (\lambda_i \sum_{j=1}^{n_i} |x_i^j|) \quad (5.14)$$

$$L2 = \sum_{i=1}^k (\lambda_i \sum_{j=1}^{n_i} (x_i^j)^2) \quad (5.15)$$

where k is the number of circuits, n_i is the number of nodes in circuit i , x_i^j might be either the weight or activation value of node j in circuit i , depending on the experiment, and λ_i is the penalty coefficient associated with circuit i . Note that for penalties that are strictly on the SNPL, n_i corresponds to only the ‘switch’ node.

5.5.3.2 Results and Discussions

Obviously, regarding memory usage and computation, the dimensions of the weight matrices are much larger than that of activation values. Therefore, owing to lower computational cost, it is easy to explain why the activation value regularizer (especially on the PL) is approximately 20% faster in terms of training, compared with its counterpart on weights. It also represents a considerably good trade-off, whereby the average test errors are quite similar (5.872%, 5.822%, and 5.828%).

However, taking effectiveness into consideration, the performance of the whole experiment is somewhat disappointing, given that none of the PC approaches could beat their SC counterpart. The speed advantage is still maintained for large networks, but for most of the cases, it requires more epochs to reach the best validation score (i.e. slower convergence).

The implementations based on random coefficients exhibited more positive performance. In contrast with the previous (i.e. constant penalty coefficient) experiments, the majority of the PC classifiers (66.67%) achieved lower test error than their SC

counterparts (Appendix A). One remarkable finding is that failures in achieving better accuracy (compared to SC) tended to occur in classifiers with low PC count (i.e. 50% of failed setups consisted of two circuits) and especially in large networks (i.e. 1000 nodes per layer). A possible explanation for this observation might be that, owing to the small pool of circuits (i.e. two), each circuit controls a large proportion of network nodes, corresponding to half of the memory/generalization power. Because of the characteristic of sparsity, in most of cases, one of the circuits would be turned off, cutting down the capability of the network. Moreover, as can be seen from previous analysis (see Figure 5.5), an extremely large circuit size does not benefit the PC architecture.

5.6 Conclusion

To conclude, in this chapter, I have proposed PC, a simple but effective architecture to address the heavy computational cost of deep neural networks. Inspired by parallel structures in nature, I hypothesised that (1) networks can be speed up by pruning connections and (2) modularity from semi-independent circuits would provide more diverse learning, regaining the classification power lost due to model reduction.

In order to prove the benefit of the PC design, experiments were conducted in comparison with MLPs. The results have justified our hypotheses in multiple cases. However, from the failures in small networks, the dependency of PC performance on several different conditions (e.g. number of circuits and circuit size) was discovered. These findings suggest that the efficiency of the proposed approach cannot be expressed linearly, but rather through a function with peaks and troughs. In other

words, the reduction in pruning should be large enough to compensate for the cost of extra communication.

In an attempt to provide better insight on the accuracy instability problem, a benchmark of sparsity was conducted to evaluate our hypothesis on modularity. The results show that networks are sparser under PC architectures, leading to a better performance in most cases. The degree of sparseness increases according to number of PCs. However, the measurement (i.e. Kurtosis) also pointed out that the sparsity gained from implementing PCs did not meet the expectation. It was expected to exhibit a large positive value denoting a high proportion of inactive nodes. On the contrary, the Kurtosis values obtained were negative in all cases.

Owing to these observations, additional experiments to enhance the sparsity have been performed. The chapter examined multiple conditions of different penalty terms and targets. It is surprising that PCs architecture did not cope well with common sparsity penalty terms. However, I did achieve success to some extent with a modification based on the modularity of PC architectures. By varying the penalty coefficient associated with each circuit, I was able to diversify their degrees of sparsity, thereby achieving some improvements in terms of accuracy.

Chapter 6

Dropcircuit

6.1 Motivation

The preliminary experiments showed that the impact of PCs was still debatable, as the proposed architecture exhibited variable performance across its configurations, especially when the size of the original ANN was typically small. Whereas the slow training might be explained by the fact that the speed gain from pruning does not compensate for the extra communication overhead, it is still unclear why there is also a reduction in accuracy.

As mentioned above, one of the main hypotheses is that the attempt to achieve automatic problem decomposition via PCs, should lead to an improvement in performance. However, as observed from the sparsity benchmark, it is believed that the degree of modularity remains insufficient. Therefore, one straightforward approach is to improve these properties. In other words, we have to diversify the learning taking place in each module. The last experiment on random sparsity coefficients shed some light for my technique: the circuits should be treated as a single unit and modularity could be obtained by varying their conditions.

DropOut is a recent technique for regularization that has been improving neural network accuracy in many applications. By randomly dropping nodes in the network, DropOut successfully prevents their co-adaptation [163]. In addition, thanks to DropOut, the over-reliance on specific input nodes is reduced [2]. In particular, it helps preventing the domination of some nodes (i.e. their weights can be excessively large such that the weighted sum tends to be considerably biased). Each node tends to have a more equal contribution and is forced to focus on its own training.

Taking advantage of this valuable property, I propose a modification to DropOut, whereby I scale it up to the circuit level (i.e. DropCircuit). In this chapter, I hypothesise that DropOut and DropCircuit should help circuits work more independently (i.e. achieving sparser perspectives of the problem domain), thus helping reap of the benefits of PC modularity. The following studies report on the performance of different types of DropOut/DropCircuit and PC combinations.

6.2 Related Works

As mentioned above, DropOut is a regularization technique that has been shown to be effective for a broad variety of neural networks and datasets. Some studies, such as [75, 2], have interpreted DropOut as an ensemble method similar to bagging. By randomly dropping nodes, many thinned networks with shared weights are implicitly trained, which when recombined during classification with appropriate scaling, approximate an ensemble of averaged thinned networks [138]. Moreover, in the situation in which DropOut is also applied to the input layer, it can be seen as a form of data augmentation whereby noise is added to the input patterns.

Probably owing to both its effectiveness and ease of implementation, DropOut has attracted significant attention from researchers in the field. Multiple modifications have been developed for either specialised or general-purpose models. DropConnect is a successful descendent of DropOut, showing even better performance on a range of datasets [154]. In the context of CNNs, as one of the most state-of-the-art ANNs for vision-related problems, DropOut has been investigated in different parts of the model. For example, Hinton pioneered the trend of implementing the technique in the final fully connected layers [75], whereas Wu and Xu extended the application of DropOut to pooling layers [163]. Motivated by the success of ResNet, many studies have been carried out to analyse and extend the architecture. Unlike the original DropOut, algorithm which drops individual nodes, the techniques proposed in [66] and [78] target network layers as units for dropping, owing partly to the modularity of the ResNet architecture. Huang et al. pioneered this approach, calling it Stochastic Depth (SD) [78]. The authors introduced a switch (drop) term to the nonlinear path while leaving the identity path untouched. They claimed that the motivation for SD is different from that of DropOut. Rather than preventing the co-adaption of nodes, SD aims to reduce the effective length of networks, because shallower networks drive information more efficiently. In general, whereas DropOut makes the network thinner during training, SD makes it shorter. In spite of its simplicity, SD has been shown to consistently save training time and achieve competitive results when compared to the original ResNet model for multiple datasets (e.g. CIFAR and ImageNet). Significantly, with SD, the authors successfully trained the 1202-layer ResNet to achieve better performance than the shallower 110 layer network, which was unachievable

in the original study. Depth DropOut (DD), proposed by Guo and Gould, follows a similar approach to SD [66], although the authors primarily addressed the problem of lengthy training times. The authors did not rebalance activations during the testing phase (as opposed to DropOut and SD), claiming that their unique trainable shortcuts would cancel out the effect itself. SwapOut (SO) reverted back to the original DropOut, in which it targeted the node instead of a block of nodes [134]. Because the residual functions in ResNets involve two components, applying dropping terms to both paths results in four possible scenarios: (i) drop the block, (ii) skip the block (i.e. taking the identity path), (iii) nonlinear block (i.e. feedforward), and (iv) residual. Therefore, the method expands on the variety of network instances that are implicitly used in an ensemble. Unlike DD, SO supports SD and emphasises the importance of having different stochastic training patterns (i.e. different survival probabilities) between layers. The authors of FractalNet also contributed another variants of DropOut, namely DropPath (DP), which aims to cope with the macro-scale structure of FractalNet. DP attempts to prevent the co-adaption of parallel paths by randomly dropping one of the two components for joining layers, similarly to SO, although in contrast to the latter, it guarantees that at least one path will survive. I extended DropOut to PCANNs (ANN featuring a PC architecture), whereby entire circuits, rather than nodes, are dropped (DropCircuit), and which I explain further in the section below.

6.3 Methodology

6.3.1 Node Dropout

Node DropOut (ND) is an implementation of the standard DropOut technique, where dropping is applied at the node level (refer to Fig. 6.1). In the PC context, it is worth recalling that each circuit was treated independently. For any single circuit, whenever a training sample is introduced, a distinct mask is generated according to its predefined probability. This favours the attempt to enhance sparsity across sub-networks. Thus, for the PC case, the input sum in Eq. (5.1) might be rewritten as follows:

$$m_{kj}^{l-1} = \text{Bernoulli}(p_k^{l-1}) \quad (6.1)$$

$$y'_{kj}{}^{l-1} = y_{kj}{}^{l-1} \times m_{kj}^{l-1} \quad (6.2)$$

$$z_{ki}^l = b_{ki}^l + \sum_j^{n_k^{l-1}} y'_{kj}{}^{l-1} \times \omega_{kij}^{l-1} \quad (6.3)$$

Here, m_{kj}^{l-1} stands for component j of the mask, corresponding to circuit k and layer $l - 1$, and the remaining elements are defined as in Eq. (5.1).

6.3.2 DropCircuit

The PC concept assumes that each circuit should produce a unique perspective of the problem, in line with the notion of problem decomposition. Therefore, it is critical to prevent circuits from adapting together. When extending the standard DropOut technique from nodes to circuits (DropCircuit), we could consider the technique as a

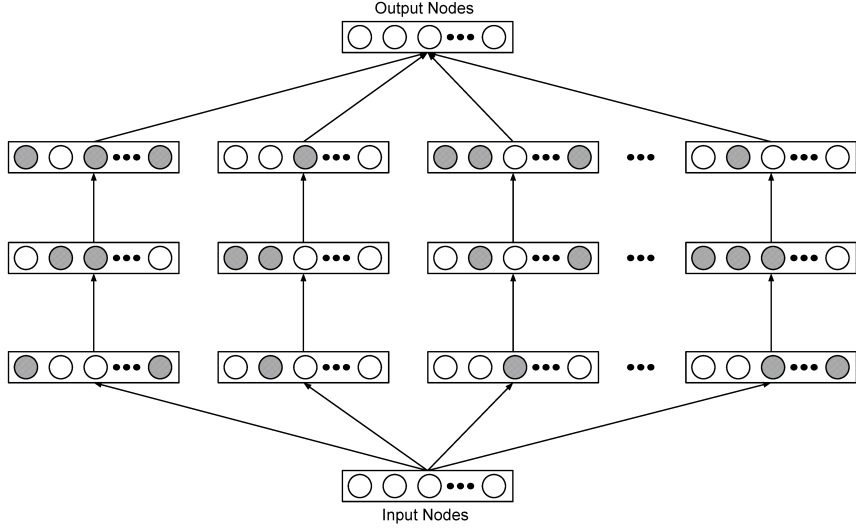


Figure 6.1: Simplified Diagram of PC with DropOut (The grey nodes have been dropped).

type of ‘uniform’ Node Dropout, whereby particular nodes stick together (in terms of their drop status) for the entire training process. Once one of them is dropped, all the nodes tied to it will also be dropped (refer to Fig. 6.2). To implement this, I create a mask m_k describing the circuit k ’s drop status. Every node belonging to the circuit inherits the mask and standard node Dropout is applied accordingly.

$$m_k^0 = m_k^1 = \dots = m_k^n = m_k = \text{Bernoulli}(p) \quad (6.4)$$

$$y'_{kj}{}^{l-1} = y_{kj}{}^{l-1} * m_k \quad (6.5)$$

6.3.3 Experimental Setup

The underlying objectives of the experiments reported in this chapter were:

1. To prove that DropOut-aided PCANNs can improve generalization performance compared to corresponding DropOut-aided SCANNs (i.e. standard fully con-

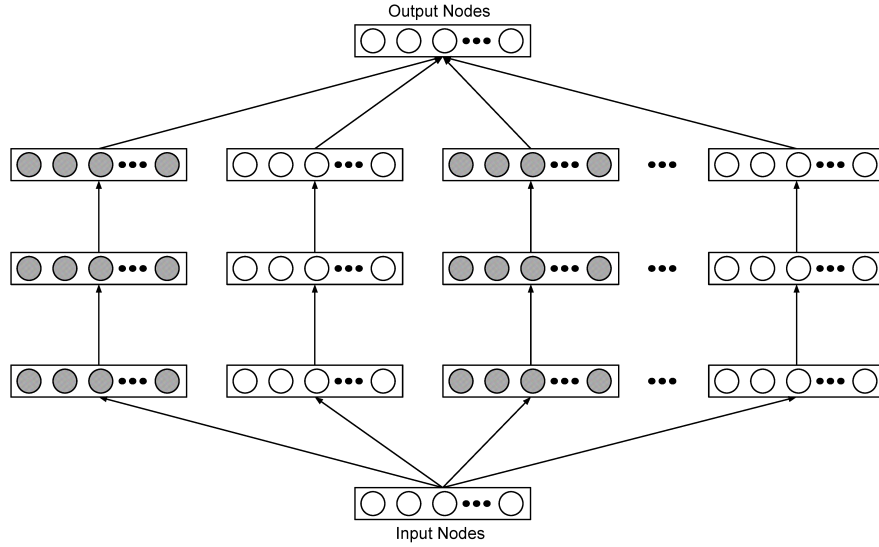


Figure 6.2: Simplified Diagram of PC with DropCircuit (The grey nodes have been dropped).

nected MLPs), thus proving that PCs are not only useful for boosting training speeds but also have an impact on accuracy.

2. To determine the most effective type of DropOut for PCANNs by comparing different variants.

The datasets adopted include MNIST and four smaller ones obtained from the UCI ML repository (i.e. BC, BT, GL, and LF; refer to Table 5.2) [98]. Two network structures were considered, one with 100 hidden nodes per layer and the other with 1000 nodes, both of which had two hidden layers. For each network structure, one SC and two PC implementations were examined (corresponding to ND and DropCircuit). The PC versions were further divided into three subcategories of 2, 5 and 10 circuits. In total, there were seven variants for each network size. The models were trained with 0.1 learning rate, 0.4 momentum, 0.0001 L2 sparsity penalty, and the chosen activation function was the hyperbolic tangent (tanh). Under all DropOut-aided

	Dataset	SCDO	PCDO	PCDC	PC Average
100 nodes	BC	2.293	4.723	4.957	4.840 (211.08%)
	BT	0.386	0.826	0.857	0.842 (218.01%)
	GL	0.814	1.707	1.767	1.737 (213.39%)
	LF	1.490	2.938	3.111	3.025 (202.99%)
	MN	362.914	424.498	412.547	418.523 (115.32%)
1000 nodes	BC	89.073	29.930	29.606	29.768 (33.42%)
	BT	16.605	5.258	5.020	5.139 (30.95%)
	GL	34.542	10.539	10.346	10.443 (30.23%)
	LF	53.541	18.392	18.316	18.354 (34.28%)
	MN	6475.711	4060.208	3958.507	4009.358 (61.91%)

Table 6.1: Training times (in seconds) for different network sizes (i.e. 100 and 1000 nodes), conditions (i.e. SCDO, PCDO, and PCDC) and datasets (i.e. BC, BT, GL, LF, and MN-MNIST). Note that each depicted PC condition represents the average of 2-, 5-, and 10-circuit conditions.

conditions, the retaining probability was 0.5. For the four smaller datasets, each condition was run for 100 trials, whereas for MNIST, 10 trials per condition were used. This large number of trials explains the choice of 100 training epochs per training session.

6.4 Experimental Results and Discussion

6.4.1 Dropout on Parallel Circuits versus on Single Circuits

This experiment focused on two different integrations of DropOut with PC: standard DropOut (i.e. PCDO) and DropCircuit (i.e. PCDC). Thus, in the comparison, it was necessary to include a single-circuit architecture implemented with standard DropOut (SCDO) as baseline. For obvious reasons, it is not possible to plainly apply DropCircuit to SCANNs.

Regarding the training duration, the problem of PC inefficiency on small networks

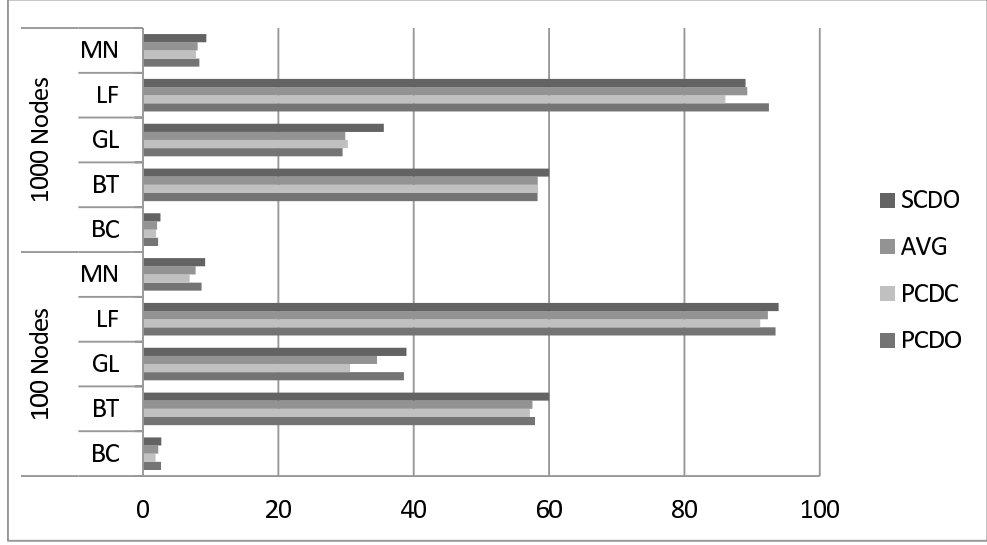


Figure 6.3: Mean test errors for different conditions and datasets (MN-MNIST, LF-Leaf, GL-Glass, BT-Breast Tissue, BC-Breast Cancer, AVG-Average of PCs)

is still unsolved. As mentioned above, the complexity reduction in small networks is not sufficient to compensate for the computational overhead involved in separating circuits. Therefore, in excessively small networks, PC actually increases the total training time to some extent. However, considering larger networks (e.g. 1000 nodes), it is observed that implementing PCs gives at least a 30% reduction in training time for all cases (Table 6.1). Because one of the mainstream strategies of DL is to improve performance by enlarging models, I believe that the reduction is significant.

Figure 6.3 illustrates the test errors for both model sizes, on different datasets, and across DropOut-aided conditions. It is clear that the results favour PCs more than SCs with generally lower average test errors (38.209% versus 40.127%). The average test error across PC conditions (i.e. PCDO and PCDC) is generally lower, except for the case in which large networks (i.e. 1000 nodes) were used to classify the LF dataset. However, the best individual condition for this case was still a PC

	Dataset	SCDO	PCDO	PCDC
100 nodes	BC	2.710	2.655	1.842
	BT	60.000	57.917	57.167
	GL	38.938	38.546	30.579
	LF	93.908	93.464	91.208
	MN	9.178	8.650	6.881
1000 nodes	BC	2.575	2.235	1.917
	BT	60.000	58.292	58.375
	GL	35.575	29.492	30.258
	LF	89.033	92.489	86.064
	MN	9.356	8.317	7.827

Table 6.2: Test Errors for Different Network Sizes (i.e. 100 and 1000 nodes), Conditions (i.e. SCDO, PCDO, and PCDC) and Datasets (i.e. BC, BT, GL, LF, and MN). Note that each depicted PC conditions represents the average of 2-, 5-, and 10-circuit conditions. Detailed tables can be found in Appendix B.

approach (i.e. PCDC). In fact, PCs always have at least one of their implementations (usually PCDC) performing better than SC conditions. Significantly, in this context of generalization performance, PCs completely surpass SCs for small networks (i.e. 100 nodes), which was unachievable in the proof-of-concept experiment.

6.4.2 Node Dropout versus DropCircuit

According to Figure 6.3, we can see that PCs on average (AVG) obtained only a slight advantage over SCs (usually less than 1% in difference) with the largest gap of 4% for the GL dataset. However, upon closer inspection, we can see that the performance of the two PC conditions varied significantly.

PCDO achieved quite similar performance to SC, with a slightly lower average test error (39.206% versus 40.123%). However, taking each dataset into account individually, it is clear that PCDO surpassed SCDO in multiple cases, except the LF dataset on 1000 node networks (see Figure B.1 to B.4, in Appendix A).

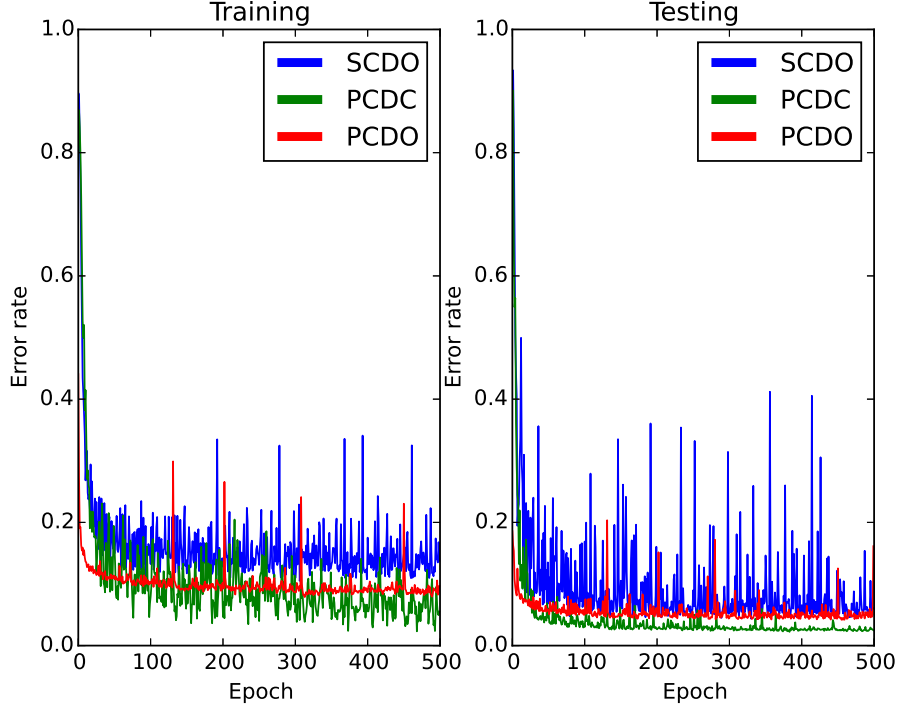


Figure 6.4: Training and Testing Error across Epochs.

PCDC, however, is clearly the best performing condition. In every setup (even in the problematic LF case), PCDC always outperformed SCDO (37.212% versus 40.127% on average) and displayed a difference of up to 8.5% for the GL dataset. In this benchmark, with the aid of DropCircuit, the same small-network PC (i.e. 100 nodes) setup achieved the best generalization performance for all five datasets. For the BC dataset, it showed an error rate of approximately 2.375%, compared with 3.66% in the earlier tests. Thus, we can conclude that DropCircuit is a useful tool for balancing the speed/generalization trade-off in ANNs that incorporate PCs.

6.4.3 Insights into DropCircuit behaviour

In Section 6.4.2, I showed that the PC approach could reduce training times. To test whether this result scaled to larger datasets, I compared conditions using MNIST, but

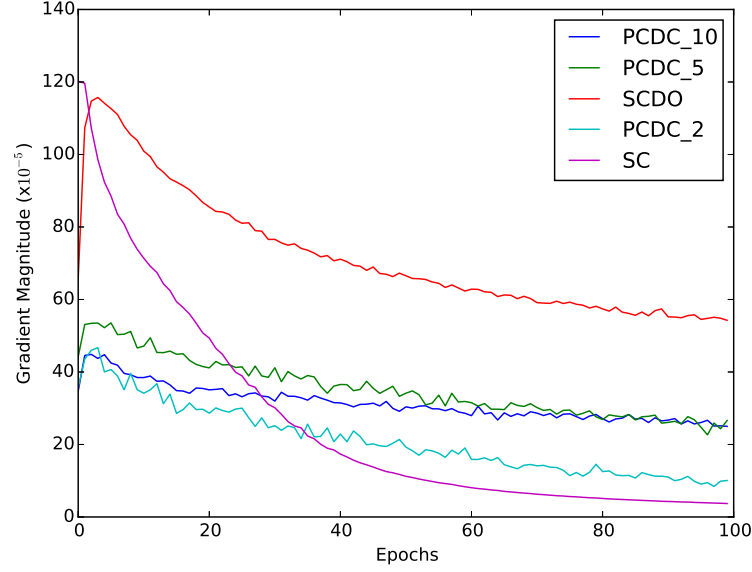


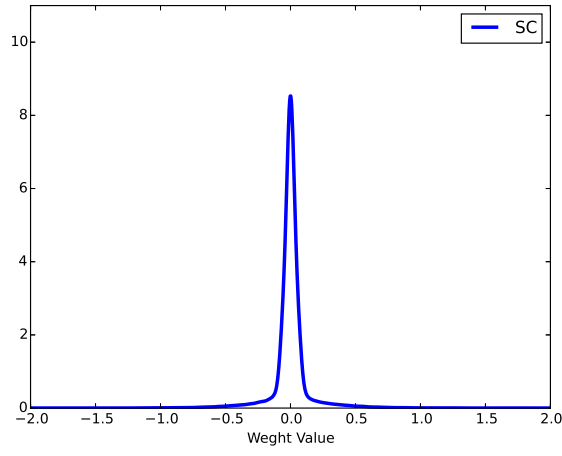
Figure 6.5: Mean Gradient Magnitude of the First Hidden Layer

this time extended the training to 500 epochs using a [1000-500] hidden-layer structure with a much higher learning rate (i.e. 1) to favour DropOut. The other parameters were kept unchanged. Figure 6.4 shows that both of the PC approaches converge faster, reaching a plateau around epoch 20 (especially PCDO) compared to epoch 60 for SCDO. Significantly, in this period, both PC approaches reached lower plateaus than SCDO. Both of the PC approaches exhibited the normal pattern of DropOut implementations, with error fluctuations throughout training. Especially in PCDC, the degree of fluctuation was even higher than PCDO. Finally, PCDO achieved a slightly lower error rate than SCDO (best at 4.19% and 4.38% respectively), whereas PCDC reached 2.22%. This revealed that: (i) PC approaches are more efficient (i.e. with smaller numbers of parameters, they achieve similar test errors within smaller epoch numbers) and (ii) PCDC performs better than PCDO.

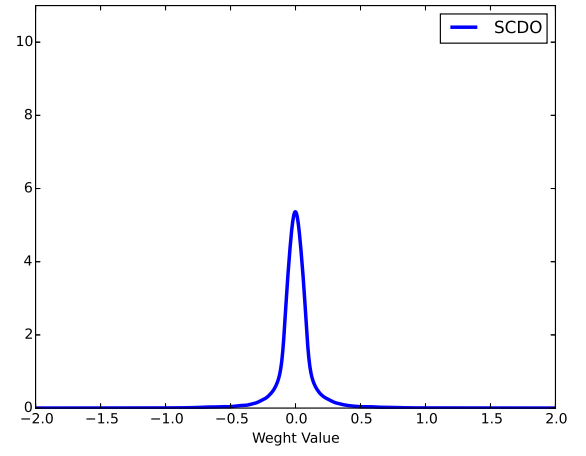
The evolution of the mean gradient (magnitude) is tracked in the first hidden layer,

in five configurations: three PCs (i.e. namely PCDC_2, PCDC_5, and PCDC_10 with 2, 5 and 10 circuits respectively) and two SCs (i.e. with and without DropOut: SC and SCDO), as shown in Fig. 6.5. It is observable that the PC approaches lead to significantly smaller gradients in the early stages of the training but maintaining larger magnitude in the end. The change in gradient magnitude is quite stable, in contrast with the conditions of SCs, especially without DropOut (encoded as SCs). The gradient magnitude of SC decreased rapidly to a low value, leading to the saturation of feature detectors. SCDO increased significantly in the first few epochs and then steadily converged. However, its magnitude never came close to those of either PCDC or SC, indicating that the training did not fit well owing to the interference of noise from DropOut. The reason why the gradient magnitudes of PCDCs were lower than those of SCDOs might come from the fact that DropCircuit eliminates the nodes in a more structural regime. While the reason why the gradients of PC conditions start at relatively low values is still unclear, it is also unclear whether starting with smaller gradients might not actually be advantageous. Although it is a speculative proposition at this point, a slower start with more steady learning might actually contribute to reducing over-fitting.

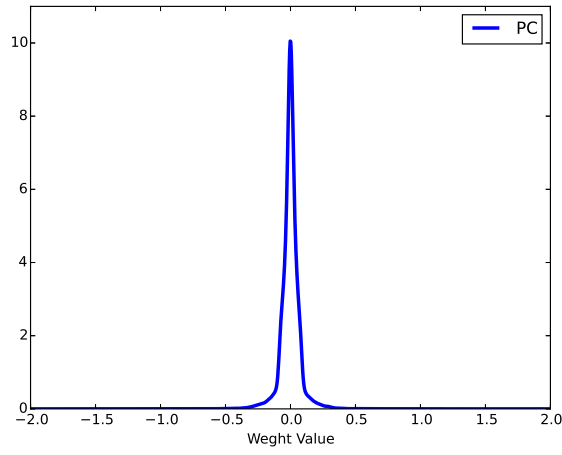
According to Figure 6.6, the sparsest distribution belongs to PC (plain, without DropOut or DropCircuit), with a high peak, indicating that most of the weights are concentrated around zero. Both PC and PCDO exhibited better sparsity than their corresponding counterparts (i.e. SC and SCDO). These observations indicated that the PC architecture has a very positive effect towards sparsity, strengthening the statements in Chapter 5. As PCANNs introduced thinner networks with less connec-



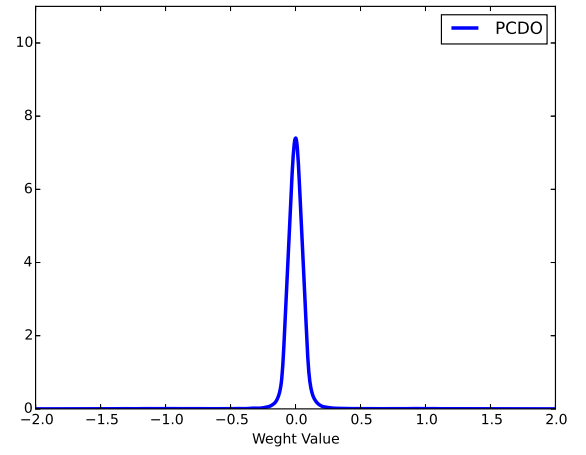
(a)



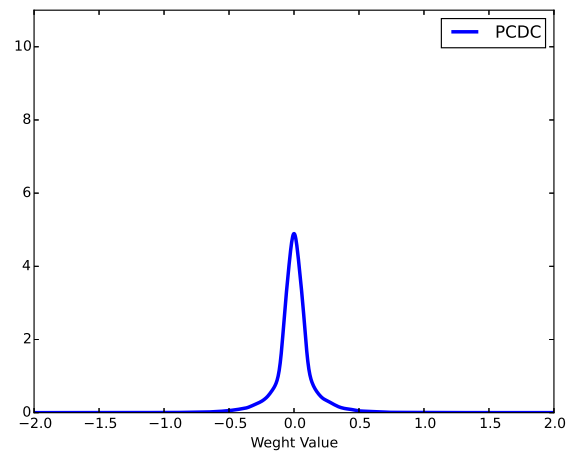
(b)



(c)



(d)



(e)

Figure 6.6: Weight Distribution of (a) SC, (b) SCDO, (c) PC, (d) PCDO, and (e) PCDC

tivity, this is believed to be the main driver of the above effect. However, the fact that both SCDO and PCDO are less sparse than SC and PC proved that DropOut does not cope well with my regularizer. The phenomenon is even worse in PCDC, in which the distribution had the same shape but a much lower peak and wider body. In other words, the network lost a considerable portion of weights that are closest to zero. A possible justification is that the model was under multiple regularizers with duplicating effect. PCANNs were constrained by sparsity penalties, in a similar manner to pruning (i.e many connections or nodes are forced to be inactive). Furthermore, I masked out considerable portions of nodes and/or weights when implementing DropOut/DropCircuit over PCANNs. This most likely interferes with the computation of penalty terms and their updates, as dropped units are unchangeable during the iteration, which causes the ‘cancellation’ effect on the regularizer. Unlike DropCircuit or DropOut, which are hard constraints, the penalties are, on other hand, soft constraints. Therefore, they are the ones that can effectively be cancelled out.

For further insight, I captured the features learned in the first layer of both SC and PC. Figure 6.7 shows that the nodes in both of these are very sparse. However, the observation is clearer in SC, where majority of nodes have flat weight magnitudes (blank cells). On the contrary, in the PC case, blank cells involve partly noise but the number of distinctive features seems to have increased dramatically.

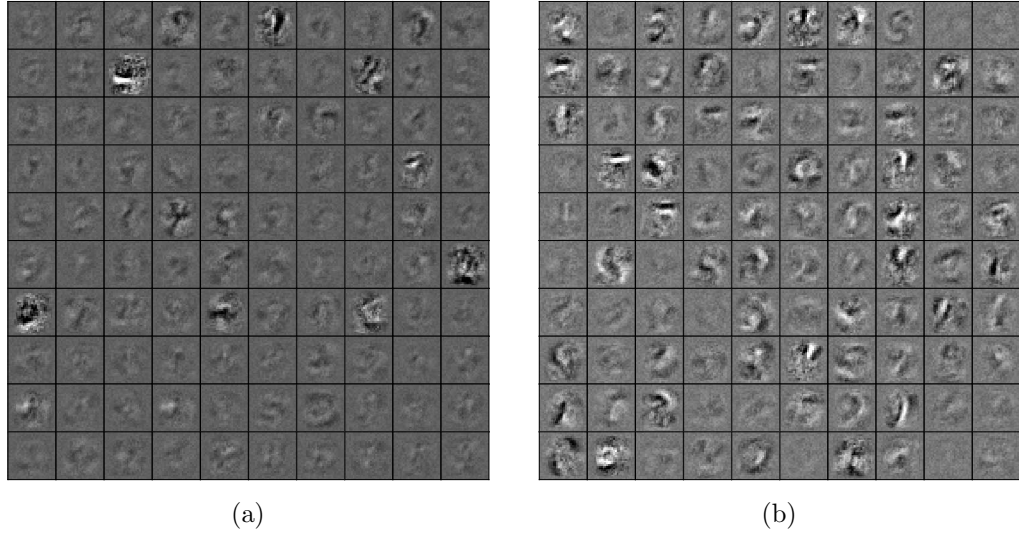


Figure 6.7: Features Learned in the First Layer of (a) SC and (b) PCDC

6.5 Conclusion

In this chapter, an improvement targeting the generalization was proposed by diversifying the learning in each circuit. The proposed DropCircuit is a variant of DropOut that is scaled up to the circuit level.

A benchmark was performed to prove the benefit of applying DropOut on PCANN. The experiments showed that combining PCANN with either DropOut or DropCircuit not only reduces training times but also enhances generalization performance in most cases. On one hand, this achievement is a long step forward in my work, as I was unable to produce such results with PC alone in the previous experiment. On the other hand, PC models cope with DropOut better than SC counterparts.

Furthermore, my work includes a comparison of different combinations. I found that DropCircuit leads to the best overall improvements in generalization performance and was able to provide more distinctive feature detectors. In addition, the advantage

in terms of training speed was maintained for large networks. Unfortunately, through investigation of the weight distribution, both DropOut and DropCircuit are discovered to actually have counter effects relative to sparsity enhancement by penalty terms, which may potentially lead to a degradation in performance.

Chapter 7

Sparse Parallel Circuits

7.1 Motivation

It was revealed in the previous chapter that PCs combined with DropCircuit (PCDCs) have negative effects on the weight control of the sparsity regularizer. The distribution on the PL layer has much shorter peak at 0, clearly showing that there are fewer inactive nodes. However, the visualisation of the first layer nodes also exhibits noisier features.

Further investigation strengthened the suspicion of its consequences for network performance. I compared PCDC with and without penalty (abbreviated as PC and PC-L2 in this set of experiments). As can be seen from Figure 7.1, PC (i.e. PCDC in previous experiment) alone converged relatively well, and performed well on the validation set. However, with the sparsity penalty, the validation error on PC-L2 started to grow considerably larger from the 50th epoch onwards.

As mentioned in Chapter 5, network modularity is highly correlated with its sparsity and thus, conserving the sparseness is a vital objective of my PC approach. However, because DropCircuit is the key driver to my recent achievements, it is necessary

to seek alternative sparsity induction approaches that match better with PCDCs.

7.2 Related Work

Chorowski and Zurada pioneered the use of Non-Negative Constraints (NNC) in ANNs [26]. Their model is motivated by the ability of Non-negative Matrix Factorization (NMF) to learn distinctive, part-based representations. In [95], the authors argued that the problem of conventional approaches such as Principle Component Analysis (PCA) is that they tolerate the co-existence of both negative and positive weights. For this reason, the eigenfaces in PCA would be cancelled out in the linear combination. Therefore, the basis images look alike and do not have a clear representation of what they are interpreting. NMF, however, only permits non-negative weighting (i.e. only additive combination allowed). Therefore, because there is no way to subtract in the combined output, both the weight (i.e. fewer basis images are used in combination) and representation (i.e. fewer features are represented in each basis images) are much sparser.

We could correlate the basis images of NMF to the features learned in each node. NNC network concepts are quite similar to NMF. In [26], the authors built a model using a hard constraint, by projecting every negative weight back to zero after each training epoch. The result turned out to be very promising with clear and sparse features. However, as stated by the authors, the accuracy was lower than that of the unconstrained network. One possible reason might be that the optimization plane was restricted by the constraints. One year later, the same group proposed a similar concept but applied to autoencoders [76]. This time, they switched from using a hard

constraint to introducing a new term in the loss function. Instead of projecting the negative weights, a penalty loss was created by summing up the squared values of the negative weights.

In fact, NMF has a strong biological motivation, as stated in [95]; the biological brain is known to have two properties:

1. The firing rates of nodes should never be negative.
2. The sign of synaptic strength is unchangeable.

7.3 Constrained Network

This chapter is motivated by the sparse feature representation of NNC networks, which fits the concept of automatic problem decomposition in PCs. Given that most biological neurons are either excitatory or inhibitory (and therefore not all non-negative) [137], I propose extending these concepts further. Therefore, in the implementation, all of the outgoing connections originating from a single node all have the same sign, based on the type of the node. Moreover, the sign of these weights should be consistent throughout the training process. The authors in [137] implemented a similar approach to Recurrent Neural Networks on a set of cognitive tasks. Their primary objective was to successfully train a network behaving similarly to biology. In this work, focussing on the computational properties of PCs, I consider each circuit, rather than each node, as an individual module, with the characteristic of being either excitatory or inhibitory at its output. Thus, as in [137], the implementation includes a penalty based on a set of constraints:

- All of the outgoing weights from each circuit are fixed as either negative or positive. In other words, the connection weights leaving the last layer of PC nodes is constrained. The type of the circuit is defined prior to training with the ratio of 4:1 (i.e. four positive circuits for every negative circuit).
- All of the incoming weights into circuits are non-negative. In other words, the weights to the first hidden PC layer are constrained. This is based on the observation that long-range connections between different areas of the brain tend to be excitatory.
- I did not constrain the remaining weights (intra-connections) to avoid over-restricting the optimization process.

7.4 Methodology

In order to justify the approach, an experiment was conducted with model specifications as below:

- **PC:** PC with DropCircuit only.
- **PC-L2:** PC with both DropCircuit and sparsity constraint, similar to the problematic model in the previous experiment.
- **PC-C:** My proposed model, which is characterised by all the 3 properties mentioned above (i.e. unique sign outgoing weights, non-negative incoming connections and unconstrained inner circuit connections).
- **PC-C-All** A modification of PC-C to justify the 3rd property. It differentiates

Architecture	[1000-1000]
Number of Circuit	10
Learning rate	0.1
Momentum	0.4
Drop rate	0.5
Number of epoch	200
Coefficient (Small)	0.0001
Coefficient (Large)	0.03
Number of experiment	10

Table 7.1: Configuration of Constrained Model

Name	Description	Best Testing Error	
		Large	Small
PC	None	2.18	2.18
PC-L2	L2	30.25	2.66
PC-C	Proposed constrained regularizer	1.97	2.18
PC-C-All	Proposed regularizer with all hidden units affected	5.23	2.31
PC-NNC	Non-negative constraint on units	5.84	2.50

Table 7.2: Name, Description and Best Test Errors for Different Setups

at having all of the node constrained to the type of their associated circuit, instead of only the PL. In other words, all the inner circuit connections have the same sign as the circuit’s outgoing connection.

- **PC-NNC** Model similar to [26], in which all nodes are constrained to be non-negative, instead of being categorised as excitatory or inhibitory.

All of the above variants were trained on the MNIST dataset, under the configuration in Table 7.1. Each used two setups, corresponding to penalty coefficients 0.03 and 0.0001. For each setup, 10 trials were run, and the results were averaged.

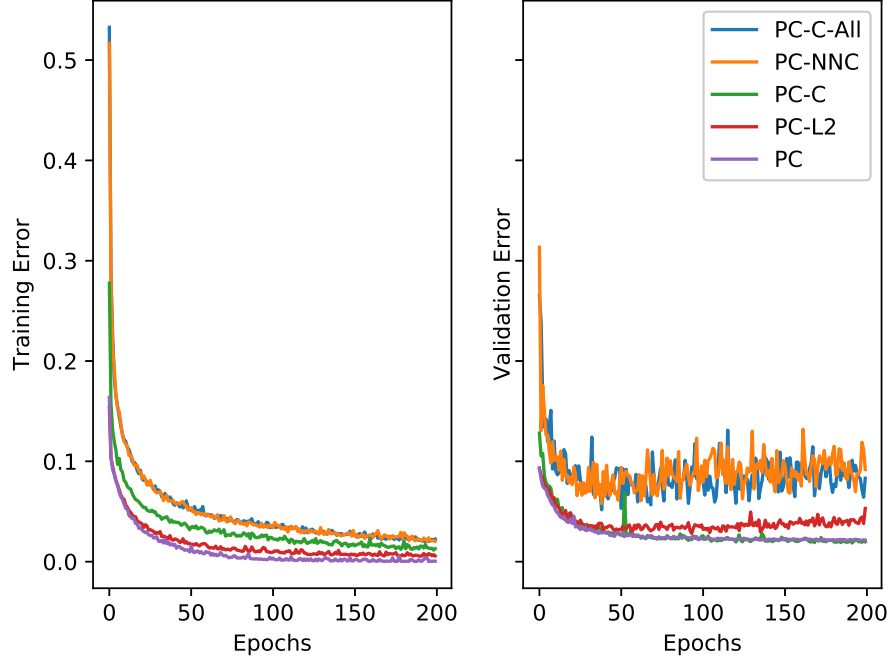


Figure 7.1: Training and Validation Error across Epochs

7.5 Result and Discussions

The experiment demonstrated that the proposed constraints for the network achieved the best performance in both conditions (i.e. 1.97% and 2.18%). The small coefficient did not constrain the networks well (see Table 7.2), and thus the features learned involved a large amount of noise as in plain PC. It should be noted that in the first test (i.e. large penalty coefficient), PC-L2 could not converge, even to below 10%, and completed the training with the best test error of 30.25%.

As can be seen from Figure 7.1, all of the constraint variants had higher training errors than PC-L2 and PC (as shown in the training curve), which is consistent with a stronger regularization effect. However, only the proposed constraint performed well on the validation set. The fact that PC-C performed better than the other two variants confirmed my suspicion that hard constraints applied over the whole network

are too heavy-handed.

According to Figure 6.6, the weight distributions in all three constrained variants (i.e. PC-C, PC-C-All, and PC-NNC) are very similar. In contrast, the distribution in PC was very different with a wide body and very flat peak. PC-L2 showed the benefit of applying the regularizer with a much higher peak and thinner body compared to that of plain PC. Significantly, unlike PC and PC-L2 that had their distributions centred on zero, the three constrained models are biased to positive values and their negative sides are extremely small, especially in PC-NNC (see Figure 7.2). This observation could be explained by PC-C, PC-C-ALL, and PC-NNC being constrained the first layer, which contains the majority of the weight population (i.e. in my configuration: 87.7%), in a similar manner. The same explanation could be used for the latter observation, as negative weights, only exist inside the circuits (i.e. small portion of weight population).

However, the high peaks at zero indicate that the weights in the constrained networks are even sparser than that of PC-L2, especially PC-NNC (Figure C.1). In other words, the set of proposed constraints perform much better than the L2-regularizer in terms of sparsity enhancement.

Figure 7.3 illustrates the representations learned in the first hidden layer of the three constrained variants. It is clear that PC-C and PC-C-ALL learned similar representations to that of PC-NNC, which contributed to the success of [26]. At each node, the features learned are mostly black (i.e. very close to zero) with only a small portion being white (i.e. positive), showing a part-based representation. The fact that only a few nodes contained those critical features also demonstrates a degree of

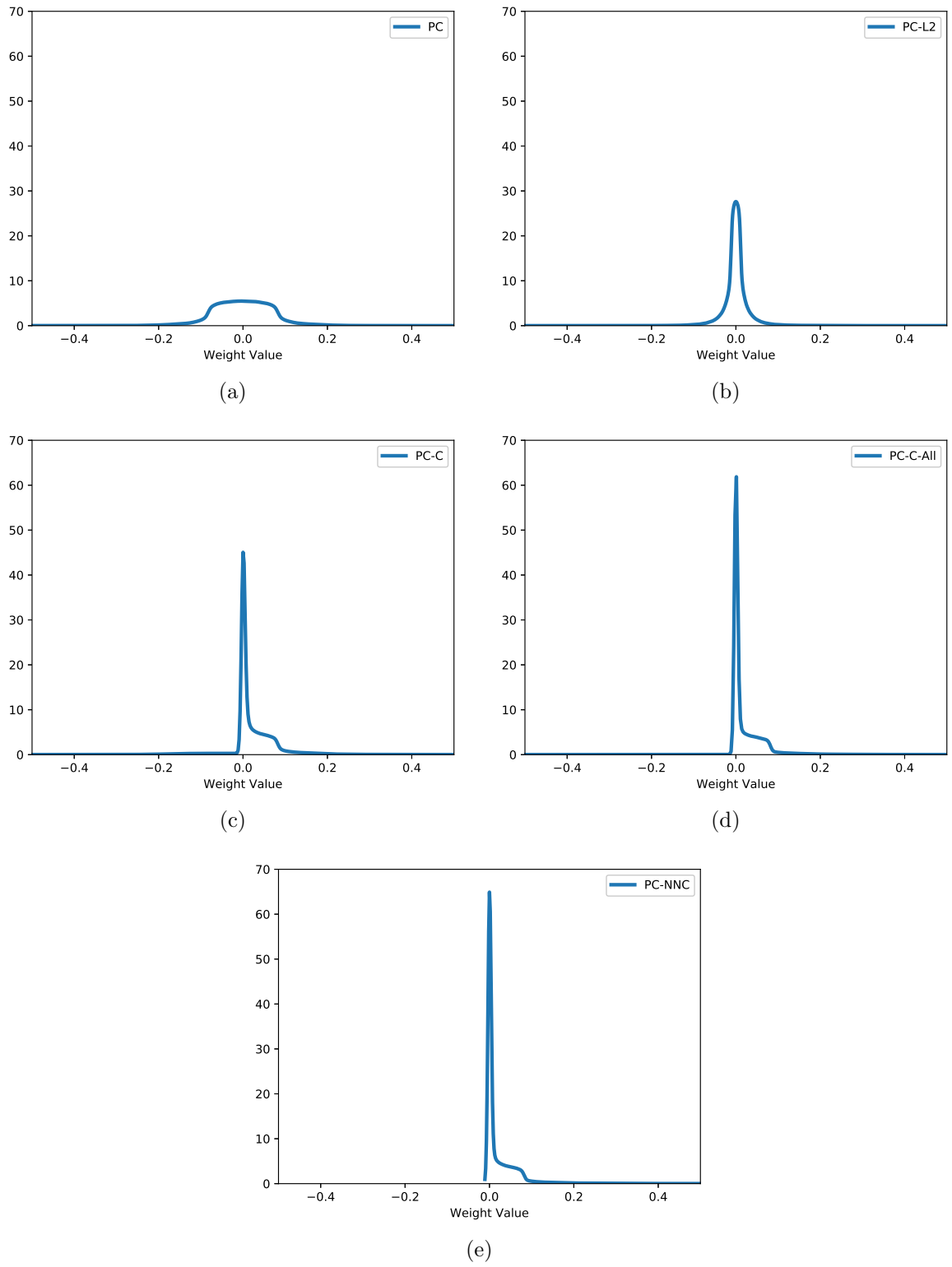
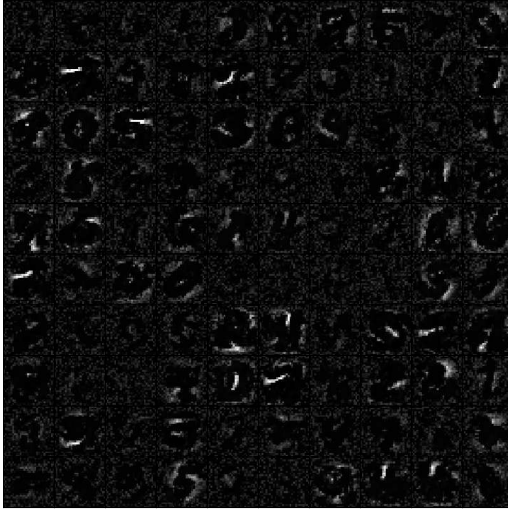
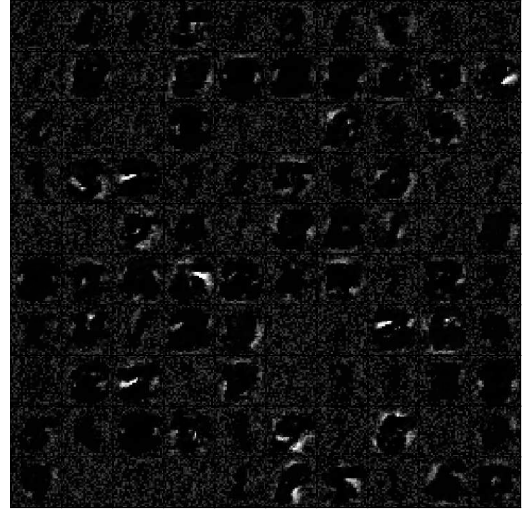


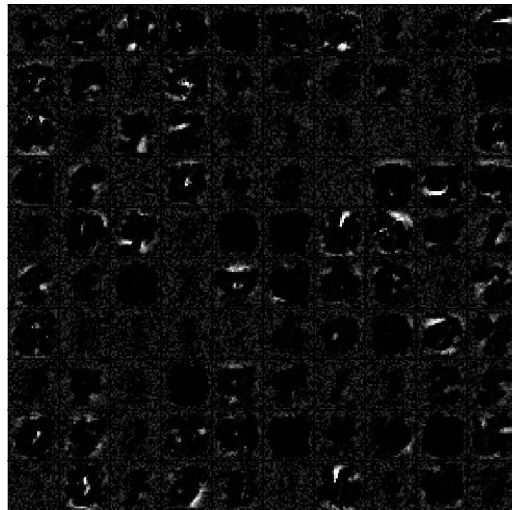
Figure 7.2: Weight Distribution of (a) PC, (b) PC-L2, (c) PC-C, (d) PC-C-ALL, and (e) PC-NNC



(a)



(b)



(c)

Figure 7.3: Features Learned in the First Layer of (a) PC-C, (b) PC-C-All, and (c) PC-NNC

sparsity. In addition, with the evidence of being the best approach in the two setups, it supersedes both the conventional sparsity penalties and NNC network [26] when combined with PCDC.

7.6 Conclusion

In this chapter, I further investigated the phenomenon of losing sparsity owing to the co-implementation of both DropCircuit and a penalty term. The consequence was clear with evidence of a degradation in performance. I addressed the problem by adopting the biological separation of excitatory and inhibitory nodes. I constrained the outgoing weights from each circuit, to be either positive or negative, and also restricted the incoming weights to being non-negative, in an attempt to produce sparsity.

The experimental results were very positive, showing that my proposed constrained model is much sparser than using the common sparsity penalties. My model contains both the advantages of NMF: (1) sparse weights and (2) sparse representations. Significantly, I obtained better accuracy than the referenced constrained model in the literature.

Chapter 8

Conclusion

8.1 Main Contributions

In Chapter 2, a study on current site-specific agriculture was presented. The chapter also described the motivation of this work, with a focus on the role, development, and challenges of VC. Chapter 3 reviewed various ML approaches in conjunction with a benchmark on my dataset of interest. The results clearly demonstrated the dominance of ANNs and their variants with regards to the task, but also showed evidence of a high computational cost. Chapter 4 in particular reported the current trends in network design along with multiple controversial opinions. This chapter summarizes the main contributions of the thesis, which were presented in Chapters 5, 6, and 7. The major contributions include: (i) proposing a lighter-weight neural architecture, (ii) improving generalization through diversified learning, and (iii) introducing biologically inspired constraints to enhance network sparsity.

8.1.1 Lightweight Parallel Circuits Architecture

In Chapter 5, I proposed PCs as a lightweight architecture to tackle the problem of heavy computation, with a focus on modularity. I supported the architecture by

hypothesising that: (i) the reduction in connections should provide speed gains and (ii) modularity created from the isolation of circuits should lead to the ability of automatic problem decomposition, thus improving the accuracy.

The proof-of-concept results were very promising, especially in the case of large networks. An average reduction in training time of approximately 40% has been achieved, while gaining a slight advantage on accuracy compared with the original fully connected networks. However, taking randomly pruned networks (RC) into account, PC only maintained the speed advantage, which is likely to originate from its advantages in implementation. Although the two models theoretically have the same number of parameters, the computation of RC in practice consisted of large matrix operations (i.e. weight matrices had the same shape of those in fully connected models) owing to the sparse connectivity, as mentioned in [97]. This evidence highlighted the computational effectiveness of the PC model, comparable to the conventional pruned networks. In terms of accuracy, both RC and PC achieved better performance than the original network. For this reason, one may argue that the main driver to the success is not necessarily the modularity of PCs (as in the 2nd hypothesis) but rather the sparser connections (i.e. lower redundancy) that exist in both PC and RC. In the proof-of-concept experiment, PC with deterministic pruning (i.e. aiming to isolate circuits) was, in fact, overshadowed by RC with random pruning (i.e. for reducing the connection count). However, it should be noted that the main purpose of the involvement of RCs is to justify the speed gain of PCs. The experimental results successfully verified my statement.

In fact, there was evidence against the hypotheses whereby PC models did not

cope well with small networks, in terms of both speed and accuracy. The observation was explained by the fact that the PC structure involves extra overhead because it replaced one-shot, large operations by a combination of multiple smaller operations. Because the reduction through pruning is proportional to network size, for the case in which the original network becomes too small, speed gains on PC cannot compensate the overhead. Unfortunately, there was no clear explanation for the accuracy drop. However, I believed that it might be due to the fact that the width of each circuit becomes too small to capture any meaningful information. In particular, the PC did not exhibit proper performance when the width is on either the small or large extreme. Such arguments could explain the observations of PC hyperparameters, showing that performance gain (especially in speed) cannot be expressed in a linear function but rather through peaks and troughs. Although the issues in small networks persisted throughout this research, the contributions of PC approach were relatively solid. Owing to the fact that larger networks have been empirically shown to achieve significantly better performance, the state-of-the-art models tend to be quite enormous rather than having only a few units as in the problematic case.

To address the uncertainty in explaining the accuracy drop, a further investigation was performed to examine the possibility of it originating from the second hypothesis. Considering the high correlation between sparsity and modularity, a highly sparse PC architecture is argued to result in a mixture of diversified circuits. Therefore, by measuring the sparsity, I attempted to evaluate the variation in the evolution of different circuits. Although PC did prove advantageous over the original network, this analysis clearly demonstrated that the degree of sparsity was much lower than

expectation. Attempts were made to improve the condition by introducing sparsity penalty terms. Unfortunately, none of these experiments returned positive results except for an experiment on varying penalty coefficients, of which the primary idea was to diversify the circuits by varying their sparsity constraints. Since then, it is observable that network-wide soft constraints such as conventional sparsity penalties did not provide as much benefit for the modularity as enforcing circuits with different conditions. However, this argument is only empirically demonstrated in this study. One possible justification might be that the direct interference constrained the circuits much more efficiently than the soft constraints, which were shown to cancel each other out in later experiments.

8.1.2 Diversified Learning with DropCircuit

Continuing the previous set of experiments, PC models is proved to worked better when each circuit was treated differently, as an individual unit. This shed light on the proposed DropCircuit approach, which is a uniform, circuit-level modification of DropOut that is expected to prevent the co-adaption of circuits.

Experiments were conducted to: (i) prove the effectiveness of DropCircuit and (ii) find the best combination with PC models. The results clearly showed the advantages of DropCircuit, where I finally obtained better accuracy in both large and small networks. The success of DropCircuit is a significant milestone, where the second hypothesis was proved. Recall that similar (small) network configurations failed in the proof-of-concept experiments (Chapter 5). Modularity, as represented by the distinctive learning in each circuit, has been shown to actually benefit generalization

(i.e. more effective learning). Results also demonstrated that the version of DropOut (i.e. DropCircuit) worked even better than the original DropOut when implemented in PC networks. However, there was a major drawback that this observation was not proved theoretically. DropCircuit is actually a special case of DropOut in which the nodes of the same circuit are bound together. Therefore, it is arguable that DropCircuit actually acted as a form of selective DropOut, reducing the large number of thinner networks that DropOut creates to a much smaller subset of more critical networks.

Further insights on DropCircuit were also reported in Chapter 6. The analysis showed that under the effect of DropCircuit, the first layer produced steady gradients with low magnitude, showing a resistance against the vanishing gradient problem. While the steady gradients might be explained by DropOut's nature, the reason for the much lower magnitude of PC's gradient than the original network is still vague.

It is worth mentioning that DropCircuit exhibited negative effects towards sparsity. It is believed that the sparsity regularizer (i.e. soft constraint) has been cancelled out because DropCircuit is a hard constraint that makes the network thinner during training. Feature visualisations also indicated that the features learned are less sparse (i.e. having fewer nodes being inactive or learning empty representations) but more distinctive (i.e. features learned at different nodes are not similar).

8.1.3 Enhance Sparsity with Constrained Network

I investigated the possible consequence of DropCircuit's negative effects on sparsity and found that it leveraged the error considerably. As DropCircuit was the key com-

ponent for previous success, I sought an alternative approach to enhance the model’s sparsity. The ability of Non-Negative Factorization to provide distinctive representations caught my interest. Owing to the fact that only additive combinations are allowed, the mutual cancellation of features learned in different nodes are eliminated, which leads to sparser representations. NNC networks adopted the idea and implemented it by penalising negative weights. However, because biological neurons are separated into excitatory and inhibitory types, I declined to constrain all of them in the same way (i.e. non-negative). The concept was implemented by assigning a type (i.e. excitatory/inhibitory) to each circuit and introducing soft constraints accordingly (i.e. non-negative for excitatory and non-positive for inhibitory).

Throughout the experiment, the model has exhibited a higher degree of sparsity compared to conventional penalty terms. From the biological perspective, the result supported the theory that long-range incoming synapses, represented by the connections to the first hidden layers of circuits, should be excitatory. In this model, the modularity of the network was beneficial from the sparse representation learned at the ‘receptor’ layer. However, the fact that I obtained higher accuracy compared with the NNC network has two significant meanings. First, although forcing the whole network to be non-negative resulted in sparser models, it is too heavy-handed and it actually hindered the learning. The observation was explained by the representational capacity being excessively restricted in NNC networks, whereas in my model, negative connections are tolerated in latter layers. Second, the above above-mentioned gain in accuracy could arguably be attributed to the diversification between circuits that it obtained by assigning excitatory/inhibitory types. With the support from

experiments in both Chapters 5 and 6, I strongly believe in the potential of further research on the characterisation (i.e. constraining) of individual circuits.

8.2 Future Work

8.2.1 Learning Parallel Circuits via Pruning

At the current stage of my research, PC exhibits a tight dependency on human design, especially in structural design. Chapter 5 demonstrated that both the speed and accuracy of PCs are correlated with circuit-related parameters such as circuit count and circuit size. Although with DropCircuit I did address the accuracy drop in small networks (i.e. small circuit size), the explanation of why the same circuit size did not work with plain PCs is not yet clear. Owing to the fact that those parameters are predefined by human design, the performance is vulnerable in many cases.

As mentioned in Chapter 5, PCs have some similarities with network pruning. The design of a PCANN actually could be considered as pruning before training. Such operations might result in general circuits with different width and depth. Therefore, it is logical to deterministically construct the network structure through pruning, ideally in the first epochs. It is recommended to start the training with a fully connected layer to analyse. After pruning, weak connections are eliminated, forming circuits that learn only the critical features. Conventional pruning techniques such as Optimal Brain Damage [38] and Optimal Brain Surgeon [70] have been discussed for a long time but as Li et al. [97] suggested, these approaches create sparse connections. In particular, because there is no specific pattern for detecting weak connections, they exist in the network at random locations. Pruning these weights would result

in a sparse, messy connectivity. However, the PC approach requires more uniform connectivity (i.e. to construct the circuits). Structural (i.e. group-wise) pruning as in [92, 97] would be more suitable for my case. A remarkable work is [158], in which the authors attempted to cluster nodes into communities. They hypothesised that highly correlated nodes (i.e. belonging to the same community) would have the same probability to connect to another particular cluster. Connections between clusters could be then represented as strong bundles, whereas the rest would be eliminated.

8.2.2 Extensions of Parallel Circuits

The work reported in this thesis implemented PC in its most basic form. There are multiple improvements that could practically be made. First, the advantage of PC's potential in parallel processing was not taken into account. The computation on PCs are highly parallel and thus, could be assigned to different processing units. Unlike similar works mentioned in Chapter 5, PCs only require communication on the output layer, significantly reducing the overhead. Moreover, I believe that with proper configuration, the problematic case of implementing PCs on small networks can be resolved.

In the experiments, the underlying structures that was used to create PC models corresponded to MLPs. More sophisticated structures belonging, for instance, to CNNs and ResNets, have been proven to exhibit much better performance, thus being part of the mainstream in ANN research. Migrating the PC approach towards these models would likely result in several breakthroughs in the research. It is worth mentioning that there are similar works, such as [97] that prunes feature maps (i.e. for

CNN) or [90] that separates models into paths with different depth (i.e. for ResNet). In terms of DL, considering each circuit as a unit, it is clear that my models only have a single layer of processing blocks (i.e. circuits). As I already proved the advantages of the model in terms of parallel stacking, it would be beneficial for the research to pursue a model with multiple layers of PCs.

Chapters 5, 6, and 7 supported the advantages of diversified learning between circuits. However, it is clear that the experiments have only just scratched the surface of the approach’s potential. Biological neurons tend to be exclusively excitatory or inhibitory, with rare exceptions. A typical example of a sophisticated approach to model these neuron types is [179]. Different types of biological neurons are characterised by different shapes, properties, and roles, which provides guidance for modelling different computational neural types. It is not necessary to strictly follow the inspiration in neuroscience but it is logical to believe that increasing the variety of circuit ‘types’ should benefit network modularity. Apart from the explicit sparsity in Chapter 5 or its implicitly inclusion via DropCircuit, any variation in hyperparameter selection or activation function will affect network behavior. However, without mimicking the structural design of the brain, creating an effective but deterministically constructed model would require a comprehensive study.

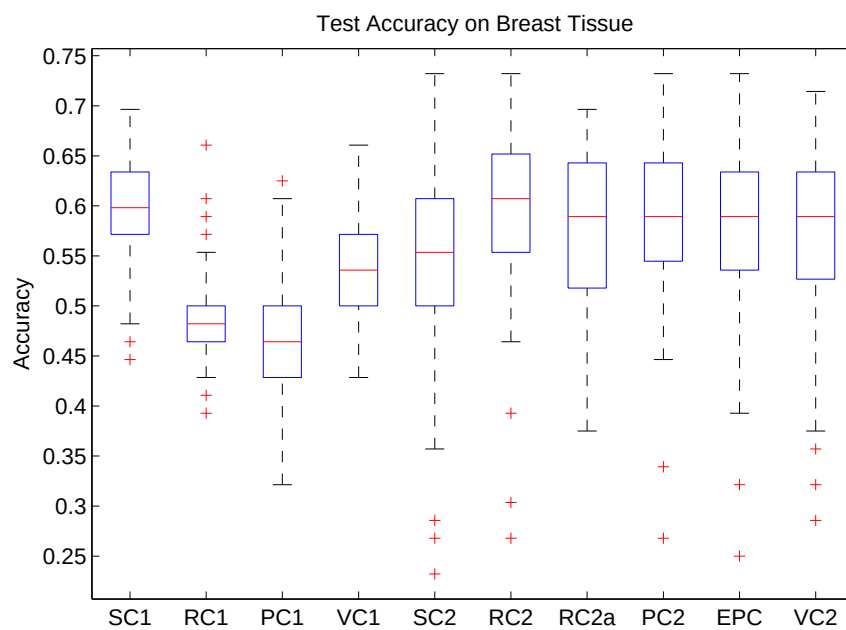
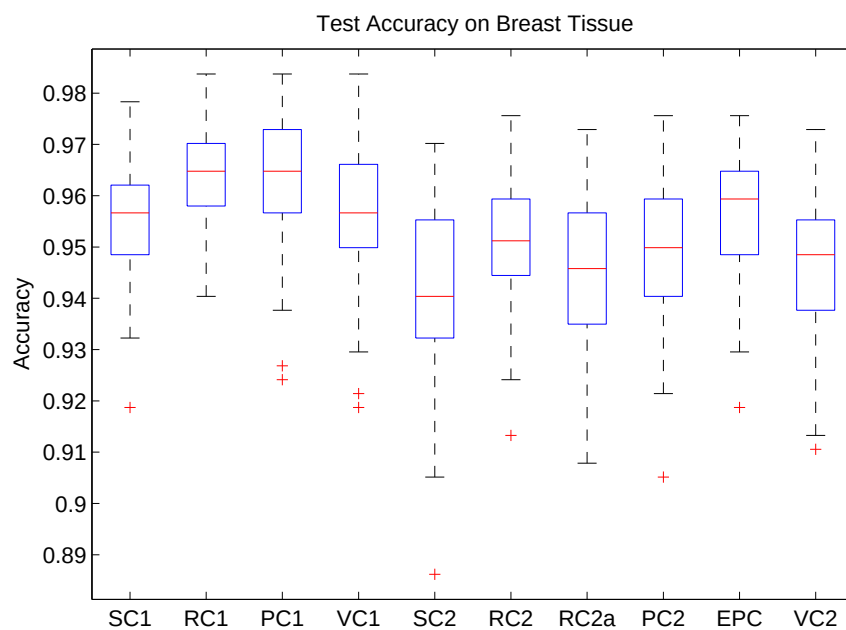
8.3 Concluding Remarks

The work in this thesis aimed at revealing the advantages of PC architectures for ANNs. I argued that sparsely connected networks would be more beneficial than the commonly used models with dense connectivity, in terms of both speed and gener-

alization. In Chapter 5, the proposed model, PC, was empirically shown to have a speed advantage, with a significant reduction in training duration, compared with a fully connected network of the same size. However, only after discovering the benefit of diversifying circuits, highlighting the importance of modularity, I was finally able to claim the second argument. The proposed DropCircuit gave PCs an accuracy advantage, claiming the leading position in term of generalization for various setups. In the end, I proposed the modelling of circuits by biologically inspired classes: excitatory and inhibitory neurons to further improve sparsity, which is highly correlated with modularity. The experiments demonstrated the advantages of my work, with an increase in accuracy as well as significantly sparse feature learning.

Appendix A

Appendix for Chapter 5



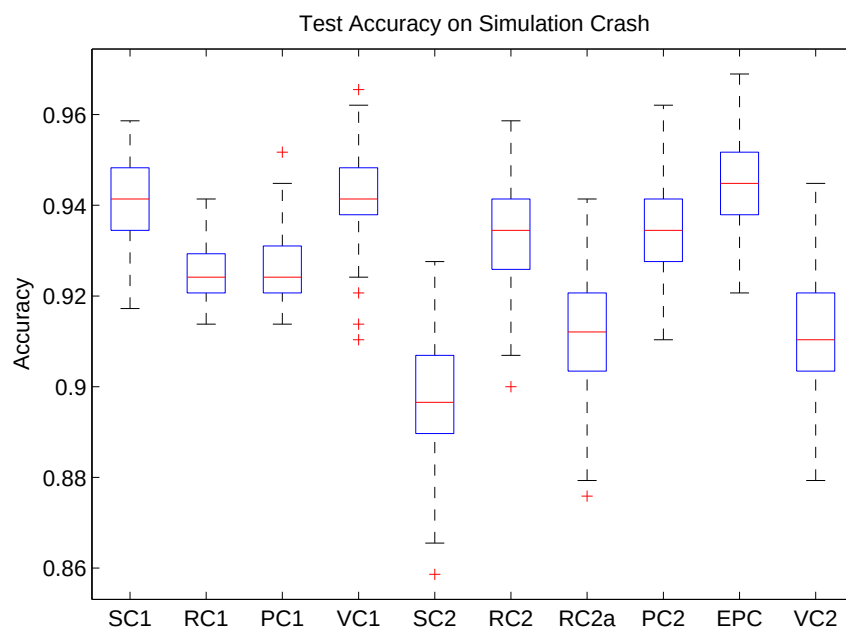


Figure A.3: Test Accuracy on Simulation Crash Dataset

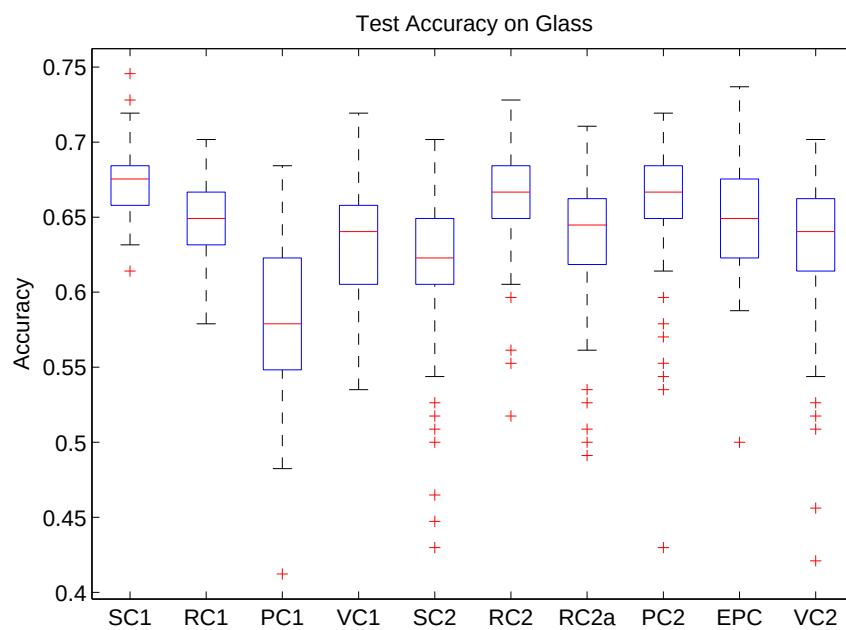


Figure A.4: Test Accuracy on Glass Dataset

Model	Mean Training Time (s)	Best Training Time (s)	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC1	1.3074	0.9587	0.9558	0.9783	62.86	48
RC1	1.6566	1.2792	0.9633	0.9837	77.72	61
PC1	8.8225	6.9766	0.9634	0.9837	77.81	62
VC1	2.6639	1.9837	0.9564	0.9837	64.88	49
SC2	14.4596	7.0719	0.9419	0.9702	40.07	20
RC2	21.3168	11.6859	0.9515	0.9756	51.98	29
RC2a	20.9014	12.2573	0.9441	0.9729	42.36	23
PC2	8.965	5.6693	0.9484	0.9756	50.19	32
EPC	60.6598	42.5088	0.9561	0.9756	54.13	38
VC2	12.6751	5.5444	0.9467	0.9729	44.33	23
Average	15.34281	9.59358	0.95276	0.97722	56.633	38.5

Table A.1: Performance of 100-Node and 1000-Node Family on Breast Cancer

Model	Mean Training Time (s)	Best Training Time (s)	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC1	0.3934	0.2624	0.5961	0.6964	78.23	56
RC1	0.3777	0.2984	0.4875	0.6607	77.07	63
PC1	2.172	1.6333	0.4727	0.625	77.14	58
VC1	0.7411	0.5633	0.5329	0.6607	74.74	57
SC2	7.7227	3.447	0.5459	0.7321	87.94	40
RC2	9.8562	6.1396	0.5989	0.7321	98.07	62
RC2a	12.7725	8.2068	0.5754	0.6964	96.98	63
PC2	4.0224	2.4271	0.5841	0.7321	96.16	59
EPC	19.0597	9.3228	0.5812	0.7321	69.24	34
VC2	4.7048	2.9322	0.5725	0.7143	95.64	57
Average	6.18225	3.52329	0.55472	0.69819	85.121	54.9

Table A.2: Performance of 100-Node and 1000-Node Family on Breast Tissue

Model	Mean Training Time (s)	Best Training Time (s)	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC1	2.4866	2.3538	0.9406	0.9586	99.97	98
RC1	2.4134	1.123	0.9255	0.9414	97.95	47
PC1	13.8171	6.7076	0.9267	0.9517	98.46	48
VC1	5.0177	4.8789	0.9412	0.9655	100	100
SC2	26.6597	14.2459	0.8976	0.9276	60.46	33
RC2	36.8932	23.4294	0.9337	0.9586	73.38	47
RC2a	38.9588	21.3629	0.9109	0.9414	60.74	35
PC2	15.6358	9.8694	0.9349	0.9621	73.45	47
EPC	136.2521	102.7331	0.9441	0.969	98.35	74
VC2	15.9524	9.3687	0.9117	0.9448	63.38	38
Average	29.40868	19.60727	0.92669	0.95207	82.614	56.7

Table A.3: Performance of 100-Node and 1000-Node Family on Simulation Crash

Model	Mean Training Time (s)	Best Training Time (s)	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC1	1.0662	0.916	0.6717	0.7456	100	100
RC1	1.1635	0.7849	0.6479	0.7018	97.55	78
PC1	5.564	4.5864	0.5834	0.6842	97.75	81
VC1	2.5699	1.9437	0.6351	0.7193	99.89	89
SC2	20.18	11.7684	0.6143	0.7018	94.25	59
RC2	24.4418	19.3928	0.664	0.7281	99.62	82
RC2a	25.1617	16.7611	0.6366	0.7105	97.89	76
PC2	9.0627	8.0953	0.6528	0.7193	99.92	94
EPC	55.7713	36.8802	0.6497	0.7368	99.25	66
VC2	9.9414	6.4624	0.6319	0.7018	98.36	66
Average	15.49225	10.75912	0.63874	0.71492	98.448	79.1

Table A.4: Performance of 100-Node and 1000-Node Family on Glass

Model	Mean Training Time (s)	Best Training Time (s)	Mean Accuracy	Best Accuracy	Mean Stopping Epoch	Best Stopping Epoch
SC1	1.8012	1.5283	0.2935	0.3947	100	100
RC1	2.0907	1.5699	0.1416	0.2053	100	100
PC1	8.9503	8.652	0.1379	0.2316	100	100
VC1	3.5561	3.1235	0.221	0.3158	100	100
SC2	30.0991	23.0352	0.4487	0.5526	99	83
RC2	36.2114	32.1325	0.5079	0.6158	100	100
RC2a	37.3126	32.5088	0.4712	0.5474	100	100
PC2	15.7276	13.9275	0.5018	0.6211	100	100
EPC	86.4461	85.2226	0.3828	0.5368	100	100
VC2	15.9411	15.4579	0.4726	0.6053	99.95	95
Average	23.81362	21.71582	0.3579	0.46264	99.895	97.8

Table A.5: Performance of 100-Node and 1000-Node Family on Leaf

Network	Type	Error	Iteration	Duration
100-100	SC	6.340	1620.400	1.561
	PCx10	8.350	1226.400	3.143
	PCx5	7.735	1331.000	2.006
	PCx2	6.910	1415.200	1.485
Average		7.334	1398.250	2.049
1000-1000	SC	5.525	1891.200	69.925
	PCx10	5.985	1803.200	12.900
	PCx5	6.095	1796.200	17.420
	PCx2	5.645	1895.000	35.001
Average		5.813	1846.400	33.812
100-100-10	SC	4.700	1268.200	1.497
	PCx10	5.730	1540.200	4.282
	PCx5	5.560	1545.400	2.694
	PCx2	4.995	1438.000	1.686
Average		5.246	1447.950	2.540
1000-1000-10	SC	4.575	1236.000	69.413
	PCx10	5.670	1568.400	14.583
	PCx5	5.250	1558.400	18.395
	PCx2	4.880	1406.000	36.753
Average		5.094	1442.200	34.786

Table A.6: Benchmark of Sparsity Induction on Weight (Fixed Coefficient).

Network	Type	Error	Iteration	Duration
100-100	SC	6.185	1638.200	1.570
	PCx10	7.780	1367.600	3.087
	PCx5	7.540	1341.600	1.836
	PCx2	6.915	1456.000	1.589
Average		7.105	1450.850	2.020
1000-1000	SC	5.485	1882.200	58.135
	PCx10	5.995	1820.600	11.400
	PCx5	6.035	1842.600	13.653
	PCx2	5.685	1857.000	34.020
Average		5.800	1850.600	29.302
100-100-10	SC	4.710	1239.800	1.405
	PCx10	5.875	1443.800	3.873
	PCx5	5.600	1472.000	2.293
	PCx2	5.090	1388.400	1.706
Average		5.319	1386.000	2.319
1000-1000-10	SC	4.405	1237.000	61.131
	PCx10	5.640	1588.400	13.135
	PCx5	5.240	1544.000	14.212
	PCx2	4.975	1360.200	32.752
Average		5.065	1432.400	30.308

Table A.7: Benchmark of Sparsity Induction on Activation Value (Fixed Coefficient).

Network	Type	Error	Iteration	Duration
100-100	SC	6.160	1613.400	1.281
	PCx10	8.125	1300.000	2.758
	PCx5	7.590	1254.600	2.039
	PCx2	7.015	1444.600	1.473
Average		7.223	1403.150	1.888
1000-1000	SC	5.485	1854.400	57.599
	PCx10	5.980	1838.200	10.929
	PCx5	5.980	1856.600	13.054
	PCx2	5.645	1882.800	30.064
Average		5.773	1858.000	27.911
100-100-10	SC	4.950	1243.800	1.380
	PCx10	5.415	1567.800	2.263
	PCx5	5.415	1567.800	2.263
	PCx2	4.860	1420.600	1.576
Average		5.160	1450.000	1.870
1000-1000-10	SC	4.575	1170.600	57.510
	PCx10	5.690	1559.400	4.135
	PCx5	5.310	1570.200	13.751
	PCx2	5.045	1363.800	31.023
Average		5.155	1416.000	26.605

Table A.8: Benchmark of Sparsity Induction on Penultimate Layer Activation Value (Fixed Coefficient).

Network	Type	Error	Iteration	Duration
100-100	SC	6.340	1620.400	1.561
	PCx10	6.750	1344.800	3.741
	PCx5	6.735	1327.200	2.514
	PCx2	5.660	1377.600	2.014
Average		6.371	1417.500	2.458
1000-1000	SC	5.525	1891.200	69.925
	PCx10	5.745	1421.800	16.546
	PCx5	5.375	1362.800	21.868
	PCx2	4.360	1071.200	53.237
Average		5.251	1436.750	40.394
100-100-10	SC	4.700	1268.200	1.497
	PCx10	4.505	1554.000	5.301
	PCx5	4.005	1629.200	3.172
	PCx2	3.295	1632.800	2.302
Average		4.126	1521.050	3.068
1000-1000-10	SC	4.575	1236.000	69.413
	PCx10	4.035	1533.000	19.105
	PCx5	3.270	1561.800	23.503
	PCx2	13.550	1184.400	53.444
Average		6.358	1378.800	41.366

Table A.9: Benchmark of Sparsity Induction on Weight (Varied Coefficient).

Network	Type	Error	Iteration	Duration
100-100	SC	6.185	1638.200	1.570
	PCx10	5.495	1489.200	3.465
	PCx5	4.705	1504.600	2.312
	PCx2	4.440	1560.800	1.786
Average		6.371	1548.200	2.283
1000-1000	SC	5.485	1882.200	58.135
	PCx10	3.985	1635.400	13.321
	PCx5	3.750	1583.000	14.689
	PCx2	4.190	1546.600	36.784
Average		5.251	1661.800	30.732
100-100-10	SC	4.710	1239.800	1.405
	PCx10	5.160	1539.400	4.983
	PCx5	4.760	1522.000	2.979
	PCx2	4.650	1440.200	2.082
Average		4.820	1435.350	2.862
1000-1000-10	SC	4.405	1237.000	61.131
	PCx10	4.475	1561.000	15.598
	PCx5	4.545	1528.400	15.969
	PCx2	13.890	1448.800	37.804
Average		6.829	1443.800	32.626

Table A.10: Benchmark of Sparsity Induction on Activation Value (Varied Coefficient).

Network	Type	Error	Iteration	Duration
100-100	SC	6.160	1613.400	1.281
	PCx10	5.080	1424.800	3.388
	PCx5	4.415	1534.400	2.304
	PCx2	4.035	1452.400	1.745
Average		4.923	1506.250	2.180
1000-1000	SC	5.485	1854.400	57.599
	PCx10	3.870	1445.600	13.028
	PCx5	3.830	1446.600	14.831
	PCx2	6.075	1005.400	35.290
Average		4.815	1438.000	30.187
100-100-10	SC	4.950	1243.800	1.380
	PCx10	4.635	1401.200	4.911
	PCx5	4.315	1426.600	2.932
	PCx2	6.225	1400.800	2.018
Average		5.031	1368.100	2.810
1000-1000-10	SC	4.575	1170.600	57.510
	PCx10	4.080	1461.800	15.099
	PCx5	10.160	1394.400	15.590
	PCx2	40.090	866.800	36.020
Average		14.726	1223.400	31.055

Table A.11: Benchmark of Sparsity Induction on Penultimate Layer Activation Value (Varied Coefficient).

Appendix B

Appendix for Chapter 6

	Dataset	SCDO	PCDO			PCDC		
			2	5	10	2	5	10
100 nodes	BC	2.710	2.720	2.590	2.655	2.040	1.560	1.925
	BT	60.000	59.100	57.400	57.250	57.025	56.825	57.650
	GL	38.938	38.925	38.538	38.175	31.338	30.625	29.775
	LF	93.908	93.317	93.483	93.592	92.792	91.108	89.725
	MN	9.178	8.507	8.698	8.746	7.297	6.586	6.758
1000 nodes	BC	2.575	2.290	2.220	2.195	1.945	1.470	2.335
	BT	60.000	59.500	57.475	57.500	57.600	57.525	60.000
	GL	35.575	30.363	29.075	29.038	31.550	29.788	29.438
	LF	89.033	90.783	92.500	94.183	91.767	85.208	81.217
	MN	9.356	9.135	8.031	7.784	9.157	7.357	6.967

Table B.1: Detailed test errors for different network sizes (i.e. 100 and 1000 nodes), conditions (i.e. SCDO, PCDO and PCDC) and datasets (i.e. BC-Breast Cancer, BT-Breast Tissue, GL-Glass, LF-Leaf, MN-MNIST). Bold for the best.

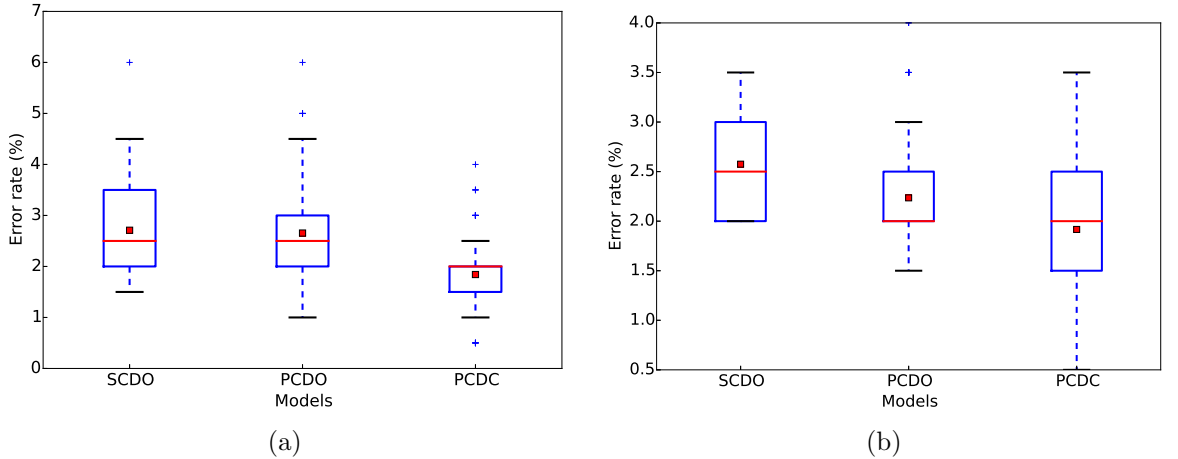


Figure B.1: Best (Lowest) Test Error on Breast Cancer Dataset with (a) 100 Node and (b) 1000 Node Network.

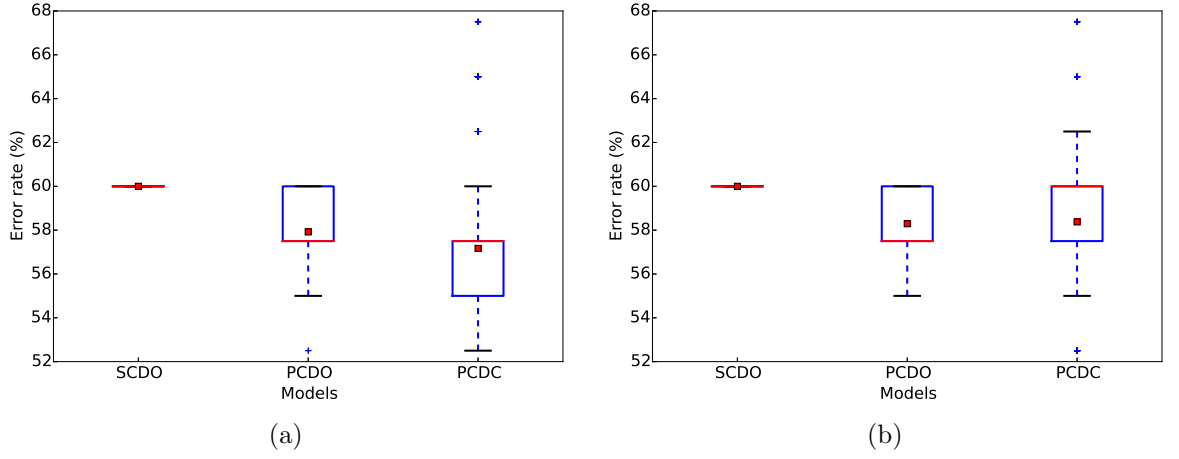


Figure B.2: Best (Lowest) Test Error over 100 Trials on Breast Tissue Dataset with (a) 100 Node and (b) 1000 Node Network.

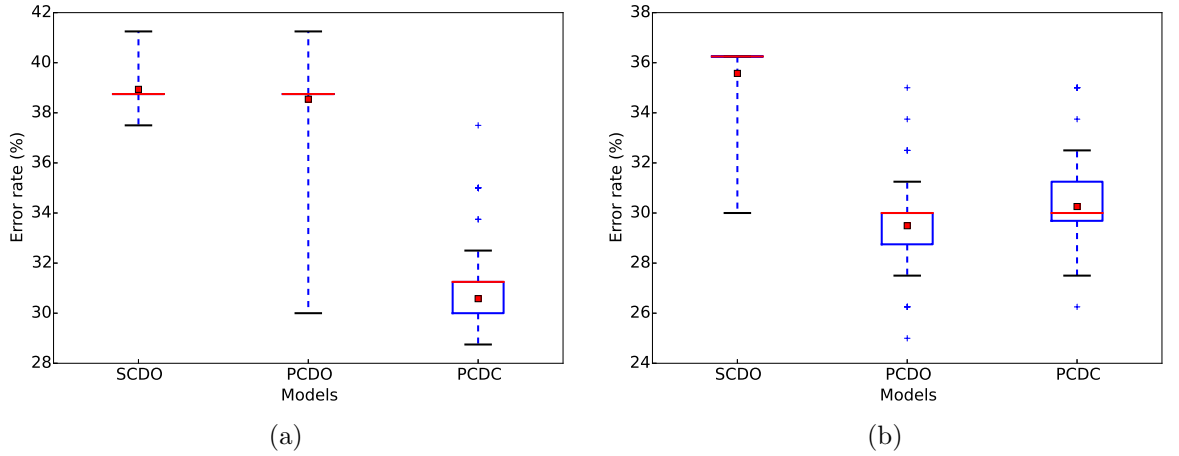


Figure B.3: Best (Lowest) Test Error over 100 Trials on Glass Dataset with (a) 100 Node and (b) 1000 Node Network.

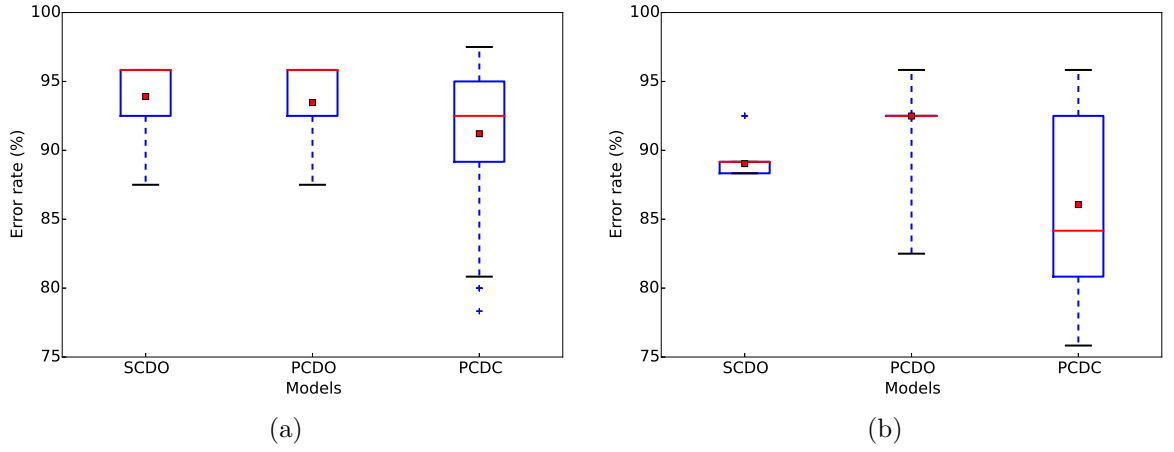


Figure B.4: Best (Lowest) Test Error over 100 Trials on Leaf Dataset with (a) 100 Node and (b) 1000 Node Network.

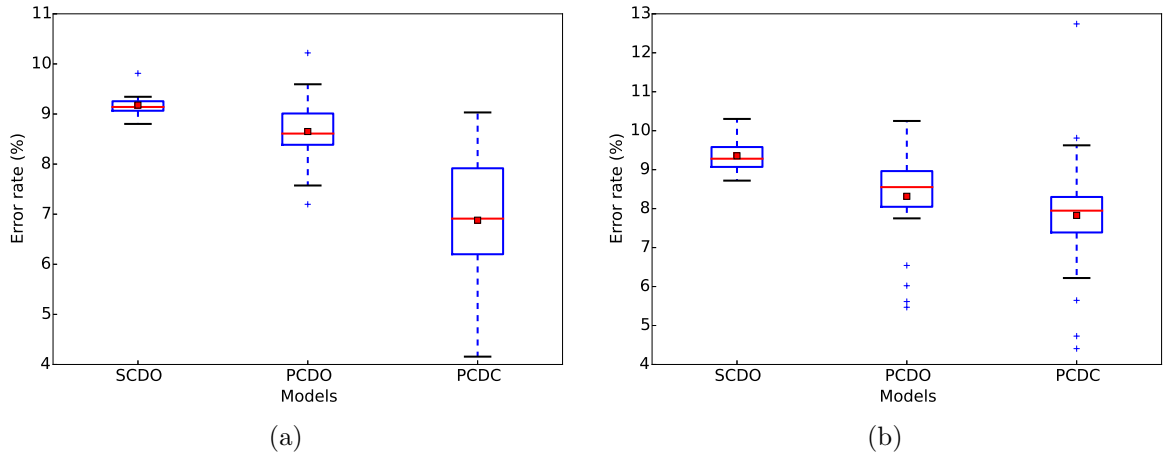


Figure B.5: Best (Lowest) Test Error over 10 Trials on MNIST Dataset with (a) 100 Node and (b) 1000 Node Network.

Appendix C

Appendix for Chapter 7

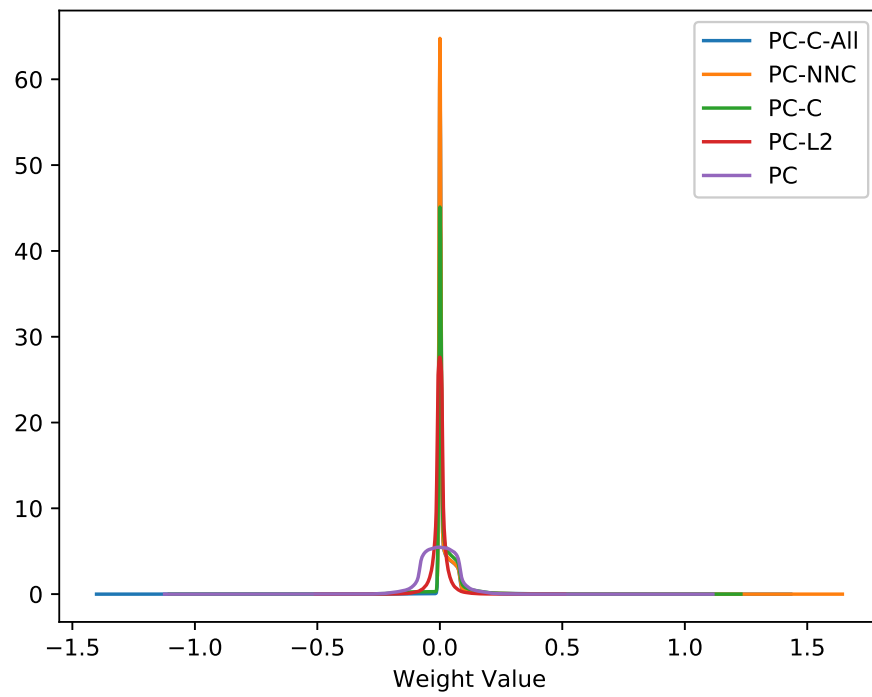


Figure C.1: Weight distribution of PC variant

Bibliography

- [1] James Richard Anderson. *A land use and land cover classification system for use with remote sensor data*, volume 964. US Government Printing Office, 1976.
- [2] Pierre Baldi and Peter Sadowski. The dropout learning algorithm. *Artificial Intelligence*, 210(1):78–122, 2014.
- [3] L.C. Bastos, R.C. Bastos, and W. Nishida. Radial basis function for classification of remote sensing images. *IJCNN’99. International Joint Conference on Neural Networks. Proceedings (Cat. No.99CH36339)*, 3:1959–1962, 1999.
- [4] Yoshua Bengio. Deep learning of representations: Looking forward. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7978 LNAI:1–37, 2013.
- [5] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.
- [6] Yoshua Bengio and Olivier Delalleau. On the expressive power of deep architectures. *Lecture Notes in Computer Science (including subseries Lecture Notes in*

- Artificial Intelligence and Lecture Notes in Bioinformatics*), 6925 LNAI:18–36, 2011.
- [7] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy Layer-Wise Training of Deep Networks. *Advances in Neural Information Processing Systems*, 19(1):153, 2007.
 - [8] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning Long-Term Dependencies with Gradient Descent is Difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994.
 - [9] James Bergstra, Norman Casagrande, Dumitru Erhan, Douglas Eck, and Balazs Kegl. Aggregate features and ADABOOST for music classification. *Machine Learning*, 65(2-3):473–484, jun 2006.
 - [10] James Bergstra, Guillaume Desjardins, Pascal Lamblin, and Yoshua Bengio. Quadratic polynomials learn better image features. Technical report, Technical Report 1337, Département d’Informatique et de Recherche Opérationnelle, Université de Montréal, 2009.
 - [11] Sergio Bernabé, Antonio Plaza, Prashanth Reddy Marpu, and Jon Atli Benediktsson. A new parallel tool for classification of remotely sensed imagery. *Computers and Geosciences*, 46:208–218, sep 2012.
 - [12] Anton Bernatskiy and Josh C Bongard. Exploiting the Relationship Between Structural Modularity and Sparsity for Faster Network Evolution. pages 0–3, 2015.

- [13] Leo Breiman. Bagging Predictors. *Machine Learning*, 24(421):123–140, 1996.
- [14] Leo Breiman. Random Forests. *Technical Report 567*, 567(September 1999):1–29, 1999.
- [15] Gunnar Jakob Briem, Jon Atli Benediktsson, and Johannes R. Sveinsson. Multiple classifiers applied to multisource remote sensing data. *IEEE Transactions on Geoscience and Remote Sensing*, 40(10):2291–2299, 2002.
- [16] D.S. Broomhead and David Lowe. Radial Basis Functions, Multivariable Functional Interpolation and Adaptive Networks. (4148), 1988.
- [17] Lorenzo Bruzzone and D. F. Prieto. A technique for the selection of kernel-function parameters in rbf neural networks for classification of remote-sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 37(2 II):1179–1184, 1999.
- [18] Gail A. Carpenter, Marin N. Gajda, Sucharita Gopal, and Curtis E. Woodcock. Artificial neural networks for remote sensing: vegetation classification from landsat tm and terrain Data. *IEEE Transactions on Geoscience and Remote Sensing*, 35(2):308–325, mar 1997.
- [19] Rich Caruana and Alexandru Niculescu-Mizil. An empirical comparison of supervised learning algorithms. *Proceedings of the 23rd International Conference on Machine Learning*, C(1):161–168, 2006.

- [20] Giovanna Castellano, Anna Maria Fanelli, and Marcello Pelillo. An iterative pruning algorithm for feedforward neural networks. *IEEE Transactions on Neural Networks*, 8(3):519–531, 1997.
- [21] Lenka Cerna and Milan Chytrý. Supervised classification of plant communities with artificial neural networks. *Journal of Vegetation Science*, 16(4):407–414, 2005.
- [22] JCW Chan and Frank Canters. Ensemble classifiers for hyperspectral classification. *Proceedings 5th EARSeL Workshop on . . .*, (2):1–11, 2007.
- [23] Jonathan Cheung Wai Chan and Desiré Paelinckx. Evaluation of Random Forest and Adaboost tree-based ensemble classification and spectral band selection for ecotope mapping using airborne hyperspectral imagery. *Remote Sensing of Environment*, 112(6):2999–3011, jun 2008.
- [24] Xueyun Chen, Shiming Xiang, Cheng Lin Liu, and Chun Hong Pan. Aircraft detection by deep belief nets. In *Proceedings - 2nd IAPR Asian Conference on Pattern Recognition, ACPR 2013*, pages 54–58, 2013.
- [25] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual Path Networks. pages 1–11, 2017.
- [26] Jan Chorowski and Jacek M. Zurada. Learning understandable neural networks with nonnegative weight constraints. *IEEE Transactions on Neural Networks and Learning Systems*, 26(1):62–69, 2015.

- [27] J. Cihlar. Land cover mapping of large areas from satellites: Status and research priorities. *International Journal of Remote Sensing*, 21(6):1093–1114, 2000.
- [28] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, aug 2012.
- [29] Dan Ciregan, Ueli Meier, and Jurgen Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.
- [30] Dan Cirezan, Ueli Meier, and Jürgen Schmidhuber. Multi-column deep neural networks for image classification. *Computer Vision and Pattern Recognition*, (February):3642–3649, 2012.
- [31] Matthew L. Clark and T. Mitchell Aide. Virtual interpretation of Earth Web-Interface Tool (VIEW-IT) for collecting land-use/land-cover reference data. *Remote Sensing*, 3(3):601–620, mar 2011.
- [32] Matthew L. Clark, T. Mitchell Aide, H. Ricardo Grau, and George Riner. A scalable approach to mapping annual land cover at 250 m using MODIS time series data: A case study in the Dry Chaco ecoregion of South America. *Remote Sensing of Environment*, 114(11):2816–2832, nov 2010.
- [33] Jeff Clune, Jean-Baptiste Mouret, and Hod Lipson. The evolutionary origins of modularity. 2012.

- [34] M. a. Cochrane. Using vegetation reflectance variability for species level classification of hyperspectral data. *International Journal of Remote Sensing*, 21(10):2075–2087, jan 2000.
- [35] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.
- [36] Jason Cong and Bingjun Xiao. Minimizing Computation in Convolutional Neural Networks. *International Conference on Artificial Neural Networks*, pages 281–290, 2014.
- [37] National Research Council. *Precision Agriculture in the 21st Century: Geospatial and Information Technologies in Crop Management*. The National Academies Press, Washington, DC, 1997.
- [38] Yann Le Cun, John S Denker, and Sara a Solla. Optimal Brain Damage. *Advances in Neural Information Processing Systems*, 2(1):598–605, 1990.
- [39] George Dahl, A McAvinney, and T Newhall. Parallelizing neural network training for cluster systems. *Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks*, (6):220—225, 2008.
- [40] Yann N. Dauphin and Yoshua Bengio. Big Neural Networks Waste Capacity. *arXiv*, (2011):1–5, 2013.
- [41] R. S. De Fries, M. Hansen, J. R. G. Townshend, and R. Sohlberg. Global land cover classifications at 8 km spatial resolution: The use of training data

- derived from Landsat imagery in decision tree classifiers. *International Journal of Remote Sensing*, 19(16):3141–3168, jan 1998.
- [42] Jeffrey Dean, Greg S Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V Le, Mark Z Mao, Marc Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large Scale Distributed Deep Networks. *Nips*, pages 1–11, 2012.
- [43] Thomas G. Dietterich. Experimental comparison of three methods for constructing ensembles of decision trees: bagging, boosting, and randomization. *Machine Learning*, 40(2):139–157, 2000.
- [44] Rasmus Ejrnæs, Erik Aude, Bettina Nygaard, and Bernd Münier. Prediction of Habitat Quality Using Ordination and Neural Networks PREDICTION OF HABITAT QUALITY USING ORDINATION. 12(4):1180–1187, 2014.
- [45] Tonand J. Enslin, W.R. and A.K. Jain. Land cover change detection using a gis-guided feature-based classification of landsat thematic mapper data, 1987.
- [46] Alejandro Gustavo Farji-Brener, Federico Chinchilla, Mar??a Natalia Uma??a, Mar??a Elena Ocasio-Torres, Alexander Chauta-Mellizo, Diana Acosta-Rojas, Sof??a Marinaro, M??nica De Torres Curth, and Sabrina Amador-Vargas. Branching angles reflect a trade-off between reducing trail maintenance costs or travel distances in leaf-cutting ants. *Ecology*, 96(2):510–517, 2015.

- [47] G.M. Foody. Papers Presented at a Workshop on Fuzzy Logic in Ecological Modelling, Kiel, Germany, 12-13 October, 1993. *Ecological Modelling*, 85(1):3–98, feb 1996.
- [48] Crop for the Future Research Centre. Crop for the future research centre: Beyond food security, 2011.
- [49] S Franklin and M Wulder. Remote sensing methods in medium spatial resolution satellite data land cover classification of large areas. *Progress in Physical Geography*, 26(2):173–205, jun 2002.
- [50] S. E. Franklin, R. J. Hall, L. M. Moskal, a. J. Maudie, and M. B. Lavigne. Incorporating texture into classification of forest species composition from airborne multispectral images. *International Journal of Remote Sensing*, 21(1):61–79, 2000.
- [51] Yoav Freund and Robert E Schapire. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [52] M Friedl and C Brodley. Decision tree classification of land cover from remotely sensed data. *Remote Sensing of Environment*, 61(3):399–409, 1997.
- [53] Nir Friedman, Dan Geiger, Moises Goldszmidt, G Provan, P Langley, and P Smyth. Bayesian Network Classifiers *. *Machine Learning*, 29:131–163, 1997.
- [54] Steffen Fritz, Ian McCallum, Christian Schill, Christoph Perger, Linda See, Dmitry Schepaschenko, Marijn van der Velde, Florian Kraxner, and Michael

- Obersteiner. Geo-Wiki: An online platform for improving global land cover. *Environmental Modelling and Software*, 31:110–123, may 2012.
- [55] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics*, 36(4):193–202, 1980.
- [56] Sabreen Gad and Timothy Kusky. Lithological mapping in the Eastern Desert of Egypt, the Barramiya area, using Landsat thematic mapper (TM). *Journal of African Earth Sciences*, 44(2):196–202, 2006.
- [57] M K Ghose, Ratika Pradhan, and Sucheta Sushan Ghose. Decision Tree Classification of Remotely Sensed Satellite Data using Spectral Separability Matrix. *International Journal of Advanced Computer Science and Applications*, 1(5):93–101, 2010.
- [58] G Giacinto and F Roli. Ensembles of Neural Networks for Soft Classification of Remote Sensing Images. *European Symposium on Intelligent Techniques*, pages 166–170, 1997.
- [59] P O Gislason, J A Benediktsson, and J R Sveinsson. Random forest classification of multisource remote sensing and geographic data. *Proceedings 2004 IEEE International Geoscience and Remote Sensing Symposium (IGARSS 2004)*, 00(C):1049–1052, 2004.

- [60] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 9:249–256, 2010.
- [61] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. *AISTATS '11: Proceedings of the 14th International Conference on Artificial Intelligence and Statistics*, 15:315–323, 2011.
- [62] Tim Gollisch and Markus Meister. Eye Smarter than Scientists Believed: Neural Computations in Circuits of the Retina, 2010.
- [63] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- [64] F Gougeon and T Moore. Individual tree classification using MEIS-II imagery. *Igarss*, 2(August):927, 1988.
- [65] Gsdi. *the Spatial Data Infrastructure Cookbook*, volume 18. 2004.
- [66] Jian Guo and Stephen Gould. Depth Dropout: Efficient Training of Residual Convolutional Neural Networks. In *2016 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2016*, 2016.
- [67] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations*, 11(1):10–18, 2009.

- [68] Ji Soo Ham, Yangchi Chen, Melba M. Crawford, and Joydeep Ghosh. Investigation of the random forest framework for classification of hyperspectral data. *IEEE Transactions on Geoscience and Remote Sensing*, 43(3):492–501, 2005.
- [69] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems*, NIPS’15, pages 1135–1143, Cambridge, MA, USA, 2015. MIT Press.
- [70] Babak Hassibi, David G. Stork, and Gregory J. Wolff. Optimal brain surgeon and general network pruning. In *IEEE International Conference on Neural Networks - Conference Proceedings*, volume 1993-Janua, pages 293–299, 1993.
- [71] George F Hepner, Thomas Logan, Niles Pitter, and Nevin Bryant. Artificial Neural Network Classification Using a Minimal Training Set : Comparison to Conventional Supervised Classification. *Photogrammetric engineering and remote sensing*, 1989.
- [72] David W. Hilbert and Jeroen Van Den Muyzenberg. Using an artificial neural network to characterize the relative suitability of environments for forest types in a complex tropical vegetation mosaic. *Diversity and Distributions*, 5(6):263–274, 1999.
- [73] Geoffrey E. Hinton. A Practical Guide to Training Restricted Boltzmann Machines. *Computer*, 9(3):599–619, 2012.

- [74] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comp.*, 18(7):1527–1554, 2006.
- [75] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *ArXiv e-prints*, pages 1–18, 2012.
- [76] Ehsan Hosseini-Asl, Jacek M. Zurada, and Olfa Nasraoui. Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints. *IEEE Transactions on Neural Networks and Learning Systems*, PP(99):1–13, 2015.
- [77] Fang Ming Hsu, Yu Tzeng Lin, and Tu Kuang Ho. Design and implementation of an intelligent recommendation system for tourist attractions: The integration of EBM model, Bayesian network and Google Maps. *Expert Systems with Applications*, 39(3):3257–3264, feb 2012.
- [78] Gao Huang, Zhuang Liu, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. *arXiv preprint*, pages 1–12, 2016.
- [79] Guang Bin Huang, P. Saratchandran, and Narasimhan Sundararajan. A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation. *IEEE Transactions on Neural Networks*, 16(1):57–67, 2005.
- [80] Xueqiao Huang and John R Jensen. A Machine-Learning Approach to Automated Knowledge-Base Building for Remote Sensing Image Analysis with GIS Data. *Photogramm. Eng. Remote Sens.*, 63(10):1185–1194, 1997.

- [81] Huang, C., L. S. Davis and J. R. G. Townshend. An assessment of support vector machines for land cover classification. *International Journal of remote sensing*, 23.4(December 2012):725–749, jan 2002.
- [82] Alfredo R. Huete. Vegetation Indices, Remote Sensing and Forest Monitoring. *Geography Compass*, 6(9):513–532, 2012.
- [83] Juha Hyypä, Hannu Hyypä, Mikko Inkinen, Marcus Engdahl, Susan Linko, and Yi Hong Zhu. Accuracy comparison of various remote sensing data sources in the retrieval of forest stand attributes. *Forest Ecology and Management*, 128(1-2):109–120, 2000.
- [84] A. G. Ivakhnenko. Polynomial Theory of Complex Systems. *IEEE Transactions on Systems, Man and Cybernetics*, 1(4):364–378, 1971.
- [85] Max Jaderberg, Andrea Vedaldi, and Andrew Zisserman. Speeding up Convolutional Neural Networks with Low Rank Expansions. 2014.
- [86] Nadav Kashtan and Uri Alon. Spontaneous evolution of modularity and network motifs. *Proceedings of the National Academy of Sciences of the United States of America*, 102(39):13773–13778, 2005.
- [87] V.R. Khare, Xin Yao, Xin Yao, B. Sendhoff, Yaochu Jin, Yaochu Jin, and H. Wersing. Co-evolutionary modular neural networks for automatic problem decomposition. *2005 IEEE Congress on Evolutionary Computation*, 3:2691–2698, 2005.

- [88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, pages 1–9, 2012.
- [89] Susan Kathleen Langley, Heather M. Cheshire, and Karen S. Humes. A comparison of single date and multitemporal satellite image classifications in a semi-arid grassland. *Journal of Arid Environments*, 49(2):401–411, oct 2001.
- [90] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. FractalNet: Ultra-Deep Neural Networks without Residuals. *arXiv preprint*, pages 1–9, 2016.
- [91] Joshua J. Lawler, Denis White, Ronald P. Neilson, and Andrew R. Blaustein. Predicting climate-induced range shifts: Model differences and model reliability. *Global Change Biology*, 12(8):1568–1584, aug 2006.
- [92] Vadim Lebedev and Victor Lempitsky. Fast ConvNets Using Group-wise Brain Damage. pages 2554–2564, 2015.
- [93] Y LeCun and Y Bengio. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(April 2016):255–258, 1995.
- [94] Yann LeCun, Bengio Yoshua, and Hinton Geoffrey. Deep learning. *Nature*, 521(7553):436–444, 2015.
- [95] D D Lee and H S Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–91, 1999.

- [96] Honglak Lee, Peter Pham, Y Largman, and AY Ng. Unsupervised feature learning for audio classification using convolutional deep belief networks. *Advances in neural information ...*, pages 1–9, 2009.
- [97] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. (2016):1–13, 2016.
- [98] M. Lichman. Uci machine learning repository.
- [99] Xiao Ling, Gui-Rong Xue, Wenyuan Dai, Yun Jiang, Qiang Yang, and Yong Yu. Can chinese web pages be classified with english data source? *The 17th international conference on World Wide Web (WWW2008)*, page 969, 2008.
- [100] Nicholas Livingston, Anton Bernatskiy, Kenneth Livingston, Marc L. Smith, Jodi Schwarz, Joshua C. Bongard, David Wallach, and John H. Long. Modularity and Sparsity: Evolution of Neural Net Controllers in Physically Embodied Robots. *Frontiers in Robotics and AI*, 3(December):1–16, 2016.
- [101] R. S. Lunetta and E. B. Mary. Application of multi-temporal Landsat 5 TM imagery for wetland identification. *Photogrammetric Engineering & Remote Sensing*, 65(11):1303–1310, 1999.
- [102] Qi Lv, Yong Dou, Xin Niu, Jiaqing Xu, Jinbo Xu, and Fei Xia. Urban Land Use and Land Cover Classification Using Remotely Sensed SAR Data through Deep Belief Networks. 2015.

- [103] Bruno D V Marino, Paul Geissler, Brian O’Connell, Nina Dieter, Tony Burgess, Cary Roberts, and Jonathan Lunine. Multispectral imaging of vegetation at Biosphere 2. *Ecological Engineering*, 13(1-4):321–331, 1999.
- [104] Masakazu Matsugu, Katsuhiko Mori, Yusuke Mitari, and Yuji Kaneda. Subject independent facial expression recognition with robust face detection using a convolutional neural network. *Neural Networks*, 16(5-6):555–559, 2003.
- [105] C.M. McCormick. Mapping exotic vegetation in the Everglades from large-scale aerial photographs. *Photogrammetric Engineering and Remote Sensing*, 65(2):179–184, 1999.
- [106] J.G. Mickelson, D.L. Civco, and J.A. Silander. Delineating forest canopy species in the northeastern united states using multi-temporal tm imagery. *Photogrammetric engineering and remote sensing*, 64:891–904, 1998.
- [107] Vikramjit Mitra, Chia Jiu Wang, and Satarupa Banerjee. Lidar detection of underwater objects using a neuro-SVM-based architecture. *IEEE Transactions on Neural Networks*, 17(3):717–731, 2006.
- [108] Doug Muchoney and James Williamson. A Gaussian adaptive resonance theory neural network classification algorithm applied to supervised land cover mapping using multitemporal vegetation index data. *IEEE Transactions on Geoscience and Remote Sensing*, 39(9):1969–1977, 2001.

- [109] D J Mulla and J S Schepers. Key processes and properties for site specific soil and crop management. *The state of site specific management for agriculture*, pages 1–18, 1997.
- [110] Taesik Na and Saibal Mukhopadhyay. Speeding up Convolutional Neural Network Training with Dynamic Precision Scaling and Flexible Multiplier-Accumulator. *Proceedings of the 2016 International Symposium on Low Power Electronics and Design - ISLPED '16*, pages 58–63, 2016.
- [111] Harini Nagendra, Richard Lucas, João Pradinho Honrado, Rob H G Jongman, Cristina Tarantino, Maria Adamo, and Paola Mairota. Remote sensing for conservation monitoring: Assessing protected areas, habitat extent, habitat condition, species diversity, and threats. *Ecological Indicators*, 33:45–59, 2013.
- [112] K. Navulur. Automated geospatial image mosaic generation with multiple zoom level support, July 19 2016. US Patent 9,396,391.
- [113] Maj L. Nordberg and J. Evertson. Vegetation index differencing and linear regression for change detection in a Swedish mountain range using Landsat TM?? and ETM+?? imagery. *Land Degradation and Development*, 16(2):139–149, 2005.
- [114] Bruno A. Olshausen and David J. Field. Sparse coding of sensory inputs. *Current Opinion in Neurobiology*, 14(4):481–487, aug 2004.
- [115] M. O'Neill and K. Dalsted. *Obtaining Spatial Data*. 2011.

- [116] M. Pal. Random forest classifier for remote sensing classification. *International Journal of Remote Sensing*, 26(1):217–222, jan 2005.
- [117] M Pal and P M Mather. Support vector machines for classification in remote sensing. *International Journal of Remote Sensing*, 26(5):1007–1011, mar 2005.
- [118] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, 2010.
- [119] Swarnajyoti Patra, Susmita Ghosh, and Ashish Ghosh. Semi-supervised Learning with Multilayer Perceptron for Detecting Changes of Remote Sensing Images. pages 161–168, 2007.
- [120] Derek R. Peddle, Ellsworth F. Ledrew, Giles M. Foody, Aironc Zhang, and Steven E. Franklin. Multi-source image classification II: An empirical comparison of evidential reasoning and neural network approaches. *Canadian Journal of Remote Sensing*, 20(4):396–407, 1994.
- [121] Paul J Pinter, Jerry Hatfield, James Schepers, Edward Barnes, M Susan Moran, Craig Daughtry, and Dan Upchurch. Remote Sensing for Crop Management. *Photogrammetric Engineering and Remote Sensing*, 69(6):647–664, 2003.
- [122] Anantha M. Prasad, Louis R. Iverson, and Andy Liaw. Newer classification and regression tree techniques: Bagging and random forests for ecological prediction. *Ecosystems*, 9(2):181–199, mar 2006.
- [123] Christoph Redies and Luis Puelles. Modularity in vertebrate brain development and evolution. *BioEssays*, 23(12):1100–1111, dec 2001.

- [124] Project Report. An Artificial Brain ATR ' s CAM-Brain Project Aims to Build / Evolve an Artificial Brain with a Million Neural Net Modules Inside a Trillion Cell Cellular Automata Evolutionary Systems Dept ., w The Genetic Programming of Cellular Automata Trails. 12(1994):215–221, 2001.
- [125] Steven W. Running, Thomas R. Loveland, Lars L. Pierce, R. R. Nemani, and E. R. Hunt. A remote sensing based vegetation classification logic for global land cover analysis. *Remote Sensing of Environment*, 51(1):39–48, 1995.
- [126] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.
- [127] S. Rasoul Safavian and David Landgrebe. A Survey of Decision Tree Classifier Methodology. *IEEE Transactions on Systems, Man and Cybernetics*, 21(3):660–674, 1991.
- [128] K. S. Schmidt and A. K. Skidmore. Spectral discrimination of vegetation types in a coastal wetland. *Remote Sensing of Environment*, 85(1):92–108, apr 2003.
- [129] Catherine D. Schuman, Thomas E. Potok, Robert M. Patton, J. Douglas Birdwell, Mark E. Dean, Garrett S. Rose, and James S. Plank. A Survey of Neuro-morphic Computing and Neural Networks in Hardware. pages 1–88, 2017.

- [130] Richa Sharma, Aniruddha Ghosh, and P. K. Joshi. Decision tree approach for classification of remotely sensed satellite data using open source support. *Journal of Earth System Science*, 122(5):1237–1247, 2013.
- [131] Shaohuai Shi and Xiaowen Chu. Speeding up Convolutional Neural Networks By Exploiting the Sparsity of Rectifier Units. 2017.
- [132] Dhruba Pikha Shrestha and J Alfred Zinck. Land use classification in mountainous areas: integration of image processing, digital elevation data and field knowledge (application to Nepal). *International Journal of Applied Earth Observation and Geoinformation*, 3(1):78–85, 2001.
- [133] Thiago S F Silva, Maycira P F Costa, John M. Melack, and Evlyn M L M Novo. Remote sensing of aquatic vegetation: Theory and applications. *Environmental Monitoring and Assessment*, 140(1-3):131–145, 2008.
- [134] Saurabh Singh, Derek Hoiem, and David Forsyth. Swapout: Learning an ensemble of deep architectures. *arXiv preprint*, (Nips):1–9, 2016.
- [135] Youngslnn Sohn and N Sanjay Rebello. Supervised and unsupervised spectral angle classifiers. *Photogrammetric Engineering and Remote Sensing*, 68(12):1271–1280, 2002.
- [136] Cristina Solares and AM Sanz. Different Bayesian network models in the classification of remote sensing images. . . . *Data Engineering and Automated Learning-IDEAL* . . . , pages 10–16, 2007.

- [137] H. Francis Song, Guangyu R. Yang, and Xiao Jing Wang. Training Excitatory-Inhibitory Recurrent Neural Networks for Cognitive Tasks: A Simple and Flexible Framework. *PLoS Computational Biology*, 12(2):1–30, 2016.
- [138] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15(2):1929–1958, 2014.
- [139] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Training Very Deep Networks. *Nips*, pages 1–9, 2015.
- [140] D M Stoms, M J Bueno, F W Davis, K M Cassidy, K L Driese, and J S Kagan. Map-guided classification of regional land cover with multi-temporal AVHRR data. *Photogrammetric Engineering and Remote Sensing*, 64(8):831–838, 1998.
- [141] S. Suresh, S. N. Omkar, and V. Mani. Parallel implementation of back-propagation algorithm in networks of workstations. *IEEE Transactions on Parallel and Distributed Systems*, 16(1):24–34, 2005.
- [142] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 07-12-June:1–9, 2015.
- [143] Valentyn A Tolpekin, Juan Pablo Ardila, and Wietske Bijker. Super-Resolution Mapping for Extraction of Urban Tree Crown Objects From Vhr Satellite Im-

- ages. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXVIII-4/C7, 3:1–6, 2010.
- [144] J. Torres-Sánchez, J. M. Peña, A. I. de Castro, and F. López-Granados. Multi-temporal mapping of the vegetation fraction in early-season wheat fields using images from UAV. *Computers and Electronics in Agriculture*, 103:104–113, apr 2014.
- [145] Jim Torresen. *Parallelization of Backpropagation Training for Feed-Forward Neural Networks*. PhD thesis, 1996.
- [146] Philip A Townsend and Stephen J Walsh. Remote Sensing of Forested Wetlands : Application of Multitemporal and Multispectral Satellite Imagery to Determine Plant Community Composition and Structure in Southeastern Stable URL : <http://www.jstor.org/stable/20051167> and sensing of forested wetland. *Plant Ecology*, 157(2):129–149, 2001.
- [147] F Tsai and William D Philpot. A derivative-aided hyperspectral image analysis system for land-cover classification. *IEEE Transactions on Geoscience and Remote Sensing*, 40(2):416–425, 2002.
- [148] Fuan Tsai and Chi-Fan Chen. Detecting Invasive Plants Using Hyperspectral and High Resolution Satellite Images. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 35(B7):71–76, 2004.

- [149] Fuan Tsai and MingJhong Chou. Texture augmented analysis of high resolution satellite imagery in detecting invasive plant species. *Journal of the Chinese Institute of Engineers*, 29(4):581–592, jun 2006.
- [150] Fuan Tsai and William Philpot. Derivative analysis of hyperspectral data. *Remote Sensing of Environment*, 66(1):41–51, 1998.
- [151] B. Tso and R. C. Olsen. Combining spectral and spatial information into hidden Markov models for unsupervised image classification. *International Journal of Remote Sensing*, 26(10):2113–2133, 2005.
- [152] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
- [153] Andreas Veit, Michael Wilber, and Serge Belongie. Residual Networks Behave Like Ensembles of Relatively Shallow Networks. *arXiv*, pages 1–9, 2016.
- [154] Li Wan, Matthew Zeiler, Sixin Zhang, Yann LeCun, and Rob Fergus. Regularization of neural networks using dropconnect. *Icml*, (1):109–111, 2013.
- [155] Mingming Wang. Multi-path Convolutional Neural Network for Complex Image Classification. 2015.
- [156] Yansen Wang, Giap Huynh, and Chatt Williamson. Integration of Google Maps/Earth with microscale meteorology models and data visualization. *Computers and Geosciences*, 61:23–31, dec 2013.

- [157] Heinz Wässle. Parallel processing in the mammalian retina. *Nature reviews. Neuroscience*, 5(10):747–57, oct 2004.
- [158] Chihiro Watanabe, Kaoru Hiramatsu, and Kunio Kashino. Modular Representation of Layered Neural Networks. pages 1–29, 2017.
- [159] Y. Wei, W. Yao, J. Wu, M. Schmitt, and U. Stilla. Adaboost-Based Feature Relevance Assessment in Fusing Lidar and Image Data for Classification of Trees and Vehicles in Urban Scenes. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, I-7(September):323–328, 2012.
- [160] M.V. Wiese. Crop Management by Comprehensive Appraisal of Yield Determining Variables. *Annual Review of Phytopathology*, 20(1):419–432, 1982.
- [161] David J. Williams, Nancy B. Rybicki, Alfonso V. Lombana, Tim M. O’Brien, and Richard B. Gomez. Preliminary investigation of submerged aquatic vegetation mapping using hyperspectral remote sensing. *Environmental Monitoring and Assessment*, 81(1-3):383–392, 2003.
- [162] Peter T. Wolter, Carol A. Johnston, and Gerald J. Niemi. Mapping submergent aquatic vegetation in the US Great Lakes using Quickbird satellite data. *International Journal of Remote Sensing*, 26(23):5255–5274, 2005.
- [163] Haibing Wu and Xiaodong Gu. Towards dropout training for convolutional neural networks. *Neural Networks*, 71:1–10, 2015.
- [164] Songtao Wu, Shenghua Zhong, and Yan Liu. Deep residual learning for image steganalysis. *Multimedia Tools and Applications*, pages 1–17, dec 2017.

- [165] Mike Wulder. Optical remote-sensing techniques for the assessment of forest inventory and biophysical parameters. *Progress in Physical Geography*, 22(4):449–476, oct 1998.
- [166] B. K. Wylie, D. J. Meyer, L. L. Tieszen, and S. Mannel. Satellite mapping of surface biophysical parameters at the biome scale over the North American grasslands a case study. *Remote Sensing of Environment*, 79(2-3):266–278, 2002.
- [167] Jingfeng Xiao and Aaron Moody. A comparison of methods for estimating fractional green vegetation cover within a desert-to-upland transition zone in central New Mexico, USA. *Remote Sensing of Environment*, 98(2-3):237–250, oct 2005.
- [168] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. *arXiv:1611.05431 [cs.CV]*, 2016.
- [169] Yichun Xie, Zongyao Sha, and Mei Yu. Remote sensing imagery in vegetation mapping: a review. *Journal of Plant Ecology*, 1(1):9–23, apr 2008.
- [170] Min Xu, Pakorn Watanachaturaporn, Pramod K. Varshney, and Manoj K. Arora. Decision tree regression for soft classification of remote sensing data. *Remote Sensing of Environment*, 97(3):322–336, 2005.

- [171] Chenghai Yang, James H. Everitt, and Joe M. Bradford. Comparison of Quick-Bird satellite imagery and airborne imagery for mapping grain sorghum yield patterns. *Precision Agriculture*, 7(1):33–44, mar 2006.
- [172] X H Yang, J F Huang, J W Wang, X Z Wang, and Z Y Liu. Estimation of vegetation biophysical parameters by remote sensing using radial basis function neural network. *Journal of Zhejiang University-Science A*, 8(6):883–895, may 2007.
- [173] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. *Arxiv*, pages 1–15, 2016.
- [174] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8689 LNCS(PART 1):818–833, 2014.
- [175] Yusen Zhan, Haitham Bou Ammar, and Matthew E. Taylor. Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. *IJCAI International Joint Conference on Artificial Intelligence*, 2016-Janua(7540):2315–2321, 2016.
- [176] Chen Zhang, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, and Jason Cong. Optimizing FPGA-based Accelerator Design for Deep Convolutional Neural Networks. *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays - FPGA '15*, pages 161–170, 2015.

- [177] Lianjun Zhang, Chuangmin Liu, Craig J. Davis, Dale S. Solomon, Thomas B. Brann, and Lawrence E. Caldwell. Fuzzy Classification of Ecological Habitats from FIA Data. *Forest Science*, 50(1):117–127, 2004.
- [178] Ying Zhang, Mohammad Pezeshki, Philemon Brakel, Saizheng Zhang, Cesar Laurent Yoshua Bengio, and Aaron Courville. Towards End-to-End Speech Recognition with Deep Convolutional Neural Networks. *arXiv preprint arXiv:1701.02720*, (November):410–414, 2017.
- [179] Guibo Zhu, Zhaoxiang Zhang, Xu Yao Zhang, and Cheng Lin Liu. Diverse neuron type selection for convolutional neural networks. *IJCAI International Joint Conference on Artificial Intelligence*, pages 3560–3566, 2017.
- [180] Guobin Zhu and Dan G. Blumberg. Classification using ASTER data and SVM algorithms: The case study of Beer Sheva, Israel. *Remote Sensing of Environment*, 80(2):233–240, 2002.