Inducing diversity among subpredictors in an implicit deep ensemble

RP van der Spoel

orcid.org/ 0000-0003-4212-0011

Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor:	Dr R Rabe
Co-supervisor:	Dr MW Theunissen

Graduation: May 2026

Declaration

I, Ruan Pieter van der Spoel, hereby declare that the dissertation entitled "Inducing diversity among subpredictors in an implicit deep ensemble" is my own original work and has not already been submitted to any other university or institution for examination.



R.P. van der Spoel

Student number:

Signed on the 6th day of March 2026 at Johannesburg.

Acknowledgements

This research was performed within the MUST Deep Learning research group of the North-West University. I would like to thank the MUST Deep Learning group for the opportunity to be part of such a stimulating and supportive research environment. It has been a privilege to work with such an incredible group of researchers.

I would like to thank:

- My supervisors, Dr. Randle Rabe and Dr. Tian Theunissen whose guidance, insight, and patience have shaped every stage of this work.
- The members of the MUST group in the office for their camaraderie and daily support.
- The broad group of friends with whom I started and completed my Master's degree—especially our Friday "Draak" sessions where we shared both good and bad times.
- A special thanks to my friends, Tjaart and Arnold, for the many days of encouragement, laughter, and shared gym sessions that helped maintain balance during the course of the study.
- To my parents, for their unwavering love and support throughout this journey. Thank you for creating an environment in which hard work was valued, perseverance was expected, and curiosity was nurtured. Your constant support—whether emotional, practical, or simply the reassurance that you were always there—made it possible for me to pursue my studies with confidence.
- Finally, I wish to express my heartfelt appreciation to Christi, whose remarkable support, especially in the final stages leading up to submission, meant more than words can capture.

Abstract

Artificial neural networks (ANNs) have achieved exceptional performance across a wide range of tasks, however, a complete understanding of how their internal mechanisms relate to their ability to perform well on unseen data remains elusive. The work presented in this dissertation investigates the perspective that a *single* ANN can be interpreted as *implicit ensemble*, where individual nodes behave as members/subpredictors of said ensemble. We focus on the diversity among these members, a quantity known to improve model performance in explicit ensembles. By investigating how to measure and induce this diversity (and how it relates to model generalisation), we highlight this under-explored perspective on ANN models.

This study builds on a framework by Davel et al. [1], which treats each hidden node as an independent classifier derived from its activation patterns. We formalise this framework and provide a set of diversity measures that quantify how similarly (or dissimilarly) nodes behave within an ANN layer. We then conduct experiments across multiple datasets and architectures, investigating how training dynamics and regularisation techniques (such as Dropout, DropConnect, and varying batch size), influence the diversity within an ANN.

Our results show that regularisation methods can induce measurable changes in subpredictor diversity and that this change in diversity correlates positively with model generalisation—particularly when measured with the *Disagreement* and *Double-fault* diversity measures. This study lays the foundation for further investigations into the ensemble-like qualities of ANNs.

Keywords: *implicit ensembles, deep learning, neural networks, diversity, generalisation, ensembles, subpredictors*

Contents

List of Figures ix

List of Tables xii

List of Abbreviations xiii

Notation xiv

1 Introduction 1

1.1 Overview 1

1.2 Problem statement 3

1.3 Research objectives 3

1.4 Research scope 4

1.5 Methodology 4

1.6 Dissertation overview 5

1.7 Publications 6

2 Background 7

2.1 Overview 7

2.2 Artificial neural networks (ANNs) 8

2.2.1 Multilayer perceptrons (MLPs) 9

2.2.2 Convolutional neural networks (CNNs) . . . 10
2.3 Ensemble theory . . . 13
2.3.1 What is an ensemble? . . . 13
2.3.2 Methods for constructing ensembles . . . 15
2.4 Diversity in explicit ensembles . . . 16
2.4.1 Generalisation in ensembles . . . 16
2.4.2 Diversity measures . . . 17
2.4.3 Pairwise measures . . . 18
2.4.4 Non-pairwise measures . . . 24
2.5 Summary . . . 30

3 Implicit ensembles **31**

3.1 Overview . . . 31
3.2 Background . . . 32
3.2.1 Modularity . . . 32
3.2.2 Implicit Ensembles . . . 34
3.3 Implicit ensemble construction . . . 35
3.3.1 Node as a predictor . . . 36
3.3.2 Ensembling nodes with a majority vote . . . 39
3.3.3 Differences between explicit and implicit ensembles . . . 40
3.4 Experimental verification . . . 43
3.4.1 Node-level accuracy . . . 43
3.4.2 Discussion . . . 44
3.5 Summary . . . 45

4 Experimental setup **46**

4.1 Overview . . . 46
4.2 Datasets . . . 47
4.2.1 Modified National Institute of Standards and Technology (MNIST) 47
4.2.2 K-MNIST . . . 47
4.2.3 Fashion-MNIST . . . 49
4.2.4 CIFAR-10 . . . 49
4.2.5 Data pre-processing . . . 51
4.3 Experimental procedure . . . 51
4.3.1 Model architectures . . . 51
4.3.2 Model training protocol . . . 53
4.3.3 Diversity calculations . . . 54
4.4 Sensitivity analysis . . . 56
4.5 Discussion . . . 56

5 Diversity in a single neural network 58

5.1 Overview . . . 58
5.2 Layer-wise diversity in a three-layer network . . . 59
5.2.1 Experimental setup . . . 59
5.2.2 Diversity results . . . 61
5.2.3 Grouping diversity measures . . . 67
5.2.4 Discussion . . . 70
5.3 Effect of network capacity on diversity . . . 72
5.3.1 Experimental setup . . . 73
5.3.2 Performance results . . . 74
5.3.3 Diversity trends . . . 77
5.3.4 Discussion . . . 83

5.4 Diversity evolution during training 84
5.4.1 Experimental setup 85
5.4.2 Performance results 86
5.4.3 Diversity results 88
5.4.4 Discussion on diversity evolution during training 91
5.5 Conclusion 93

6 Inducing diversity in implicit deep ensembles 95

6.1 Overview 95
6.2 How to induce diversity in a network 96
6.2.1 Dropout 97
6.2.2 DropConnect 98
6.2.3 Batch-size variation 99
6.2.4 Summary 101
6.3 Dropout 101
6.3.1 Experimental setup 101
6.3.2 Dropout effects on performance 103
6.3.3 Dropout effects on diversity 106
6.3.4 Summary 111
6.4 DropConnect 112
6.4.1 Experimental setup: 112
6.4.2 DropConnect effects on performance 113
6.4.3 DropConnect effects on diversity 116
6.4.4 Summary 121
6.5 Batch size 122
6.5.1 Experimental setup: 122

6.5.2 Batch size effects on performance . . . 122
6.5.3 Batch size effects on diversity . . . 125
6.5.4 Summary . . . 130
6.6 Conclusion . . . 130

7 Conclusion **133**

7.1 Introduction . . . 133
7.2 Key findings . . . 133
7.3 Contributions . . . 137
7.4 Limitations and future work . . . 138
7.5 Final remarks . . . 140
References . . . 141

A Supplemental content **150**

A.1 Replication of prior work . . . 150
A.1.1 Overview . . . 150
A.1.2 Experimental setup . . . 150
A.1.3 Node-level class activation distributions . . . 151
A.1.4 Layer-wise accuracy under different estimates . . . 152
A.1.5 Discussion . . . 153
A.2 Sensitivity analysis . . . 154
A.2.1 Experimental setup . . . 154
A.2.2 Results . . . 155
A.2.3 Conclusion . . . 160
A.3 Capacity experiments . . . 160
A.3.1 Weight-change analysis . . . 160

List of Figures

2.1 MLP diagram 8

3.1 Node-level test accuracy heatmaps 44

4.1 MNIST sample images 48

4.2 K-MNIST sample images 49

4.3 Fashion-MNIST sample images 50

4.4 CIFAR-10 sample images 50

5.1 Fashion-MNIST baseline *Disagreement* 62

5.2 Fashion-MNIST baseline *Q-statistic* 64

5.3 CIFAR-10 baseline *Disagreement* 66

5.4 Grouped baseline measure scatter plots 67

5.5 Grouped baseline diversity trends across layers 68

5.6 Model and layer performance over varying model capacity 74

5.7 *Disagreement* across model capacity 77

5.8 *Double-fault* across model capacity 78

5.9 *Q-statistic* across model capacity 80

5.10 *Entropy* across model capacity 81

5.11 MNIST diversity versus model performance across model capacity 82

5.12 Fashion-MNIST diversity versus model performance across model capacity 83

5.13 Accuracy across training epochs 86

5.14 Diversity across epochs for all measures 89

5.15 Relationship of averaged diversity with model accuracy during the training process for MNIST 90

5.16 Relationship of averaged diversity with model accuracy during the training process for Fashion-MNIST 91

6.1 Dropout effects on layer performance 104

6.2 Dropout effects on model performance 105

6.3 Dropout effects on averaged model diversity 107

6.4 Dropout effects on the relationship between diversity and model performance 109

6.5 DropConnect effects on layer performance 114

6.6 DropConnect effects on model performance 115

6.7 DropConnect effects on averaged model diversity 116

6.8 DropConnect effects on the relationship between diversity and model performance 119

6.9 Batch size effects on layer performance 123

6.10 Batch size effects on model performance 124

6.11 Batch size effects on averaged model diversity 126

6.12 Batch size effects on the relationship between diversity and model performance 128

7.1 Key findings summary 137

A.1 Class activation distributions 152

A.2 Layer accuracy across layers 153

A.3 Sensitivity analysis - *Disagreement* 156

A.4 Sensitivity analysis - *Double-fault* 157

A.5 Sensitivity analysis - *Q-statistic* 158

A.6 Sensitivity analysis - Entropy 159

List of Tables

0.1 Notation conventions used throughout this dissertation. xiv

2.1 Pairwise contingency table 19

2.2 Summary of ensemble diversity measures, where ↑ (↓) indicates that the larger (smaller) the measurement, the higher the estimated diversity. . . . 30

3.1 Hyperparameter setup for node-as-subpredictor verification 44

4.1 CNN architecture setup 52

5.1 Hyperparameter setup for baseline model training 60

5.2 Diversity-layer accuracy correlations 63

5.3 Correlation-based diversity-layer accuracy correlations 63

5.4 Grouped baseline layer and diversity correlations over all datasets 69

6.1 Dropout CNN specifications 102

6.2 Hyperparameter setup for Dropout model training 103

6.4 Hyperparameter setup for DropConnect model training 113

6.3 DropConnect CNN specifications 132

6.5 Hyperparameter setup for varying batch size for model training 132

A.1 Weight change statistics for MNIST models of varying width. 162

A.2 Weight change statistics for Fashion-MNIST models of varying width. . . . 163

List of Abbreviations

Adam Adaptive moment estimation

ANN Artificial neural network

CNN Convolutional neural network

CIFAR-10 Canadian Institute for Advanced Research

DE Deep ensemble

DNN Deep neural network

GPU Graphics processing unit

KDE Kernel density estimator

K-MNIST Kuzushiji-MNIST

MLP Multilayer perceptron

MNIST Modified National Institute of Standards and Technology

ReLU Rectified linear unit

RGB Red-Green-Blue

SGD Stochastic gradient descent

Notation

Unless stated otherwise, we consistently make use of the notation described here.

Table 0.1: Notation conventions used throughout this dissertation.

Description	**Notation**	**Example**
a scalar	an upper- or lower-case letter	X or x
a vector	a bold lower-case letter	$\mathbf{x}$
a matrix	a bold upper-case letter	$\mathbf{X}$
a set	a calligraphic letter	$\mathcal{X}$
a number system	a blackboard-bold letter	$\mathbb{R}$

We use subscripts and superscripts to index elements of vectors and matrices:

- Superscripts in round brackets, such as $\mathbf{x}^{(i)}$, denote sample indices.
- Superscripts without brackets, such as $h^n_{(\ell)}$ denote node indices.
- Subscript and superscript pairs such as $W^{p,n}_{(\ell-1),(\ell)}$ denote connections between nodes in adjacent layers where the order of the sub- and superscript, separated by a comma, match.
- Subscripts without brackets, such as D_a denote identifiers for individual elements within a set or collection.

Chapter 1

Introduction

This chapter introduces the context, motivation, and problem statement for the study. It concludes with an overview of the dissertation structure and a summary of related publications.

1.1 Overview

Artificial neural networks (ANNs), often simply referred to as neural networks, have recently been adopted by and incorporated in nearly every field of modern science and technology, transforming the way that people interact and work with these fields. They have achieved unprecedented performance across a variety of application domains, such as computer vision [2], natural language processing [3], speech recognition [4], and reinforcement learning [5]. For specific machine learning applications such as autonomous driving, medical diagnosis, financial modelling, and even material discovery, deep learning has become one of the main approaches, solving some of the most complex tasks that were once considered infeasible with traditional methods. Our theoretical understanding of *why* ANNs perform so well—and how they generalise beyond the data they are trained

on—remains incomplete. Additionally, a lack of interpretability continues to limit their use in high-risk or safety-critical domains such as healthcare, insurance, and law, where decisions must be explainable and reliable [6, pp. 1–8],[7, pp. 421–435][8], [9].

The origin of modern deep learning can be traced back through several architectural breakthroughs that demonstrate the diverse application of these ANNs. LeNet-5 [10] was one of the first convolutional architectures that displayed practical success on image recognition tasks such as handwritten digit classification (e.g., MNIST [10]). AlexNet [2] also revolutionised large-scale image classification by leveraging Graphics processing unit (GPU) acceleration and deep convolutional architectures. Newer architectures such as VGG16/19 [11], ResNet [12], and Inception [13] introduced even deeper and more efficient network designs. Transformer-based models [3], [14] supplemented the recurrent and convolutional techniques (e.g., those used in architectures such as VGG16/19 and ResNet) with self-attention mechanisms [3], which led to state-of-the-art results in natural language processing, understanding, and generation.

Understanding why ANNs generalise well to unseen data has been a major focus of recent research. There are multiple perspectives that have been proposed to explain this phenomenon. Some studies attribute generalisation to implicit biases in the optimisation process [15], [16] (e.g., Stochastic gradient descent (SGD) tends to find "flat" minima [6, pp. 149–150, 275–279][7, pp. 83–88]). Others focus on regularisation—both explicit (e.g., weight decay [17], Dropout [18], batch normalisation [19]) and implicit (e.g., the favourable implicit biases of SGD [7], [20], [21, pp. 141–143])—suggesting that these also play a critical role in shaping generalisation behaviour [18], [22][1].

A complementary way of understanding generalisation is to examine the *internal structure* of ANNs. Recent work has proposed viewing networks as a system of interacting subcomponents—such as modules, clusters, sub-networks, or circuits—that cooperate to produce a final prediction [23]–[26]. Some insightful viewpoints treat an ANN as an *implicit ensemble* of smaller models or paths, where individual neurons or sub-networks act as members of sub-classifiers that contribute collectively to the overall model decision [1],

[1]Strictly speaking, regularisation refers to the addition of explicit terms to the loss function. However, in practice, the term is often used to refer to any strategy that improves generalisation [7, p. 138]

[27].

In the literature on *explicit* ensembles, an important concept is that of *diversity*, which aims to characterise how similarly (or dissimilarly) ensemble members make their predictions [28], [29]. A wide range of studies have shown that, when the individual members are sufficiently accurate, higher diversity among them often leads to improved generalisation performance.

While diversity in explicit ensembles has been studied extensively, much less is known about how diversity manifests within a *single* ANN viewed as an *implicit ensemble*, or how such diversity might relate to the network's generalisation performance.

1.2 Problem statement

Recent research has begun to explore concepts analogous to implicit ensembles as they relate to ANNs [1], [23]–[27], but the idea of viewing a single ANN as an implicit ensemble remains underexamined. In classical ensemble learning, *diversity* among ensemble members is known to play a crucial role in improving generalisation [28], [30], [31]. However, in the context of viewing a single ANN as an implicit ensemble, the role of diversity and its relationship with generalisation remains unexplored. Existing diversity measures were developed for explicit ensembles and have not been thoroughly evaluated for their relevance or interpretability when applied to individual layers or nodes within deep ANNs.

1.3 Research objectives

This study investigates the following question: How can diversity among the ensemble members of a single ANN (assumed to be an implicit ensemble) be quantified, interpreted, and actively induced, and how does this diversity influence the model's generalisation performance?

We approach this problem as follows:

- We first curate a set of measures suitable for measuring diversity.
- Using established methods from the literature, we treat a single ANN as an implicit ensemble.
- We then perform an exploratory study to characterise diversity within the network using our curated measures.
- Finally, we investigate how well-known regularisers affect diversity, and examine how these changes relate to model generalisation.

1.4 Research scope

In this study we focus on standard image classification tasks using the MNIST [10], Kuzushiji-MNIST [32], Fashion-MNIST [33], and CIFAR-10 [34] datasets. These datasets were chosen for their widespread use in empirical investigations of ANN behaviour and their well-understood characteristics which make them suitable for the analyses in the study. The ANN architectures used in the study include shallow Multilayer perceptrons (MLPs) and relatively small Convolutional neural networks (CNNs). While these models do not reflect the scale of modern state-of-the-art networks, they are sufficient to support our claims by capturing the trends and mechanisms that underlie our study.

1.5 Methodology

This study falls squarely under quantitative experimental research, in that we are investigating a hypothesis (that neural networks might be probed as implicit ensembles). We select fitting tools and experimental setups to manipulate various independent variables (model size, datasets, training protocols) while monitoring dependent variables (model performance, diversity metrics, etc.) in order to find promising regular patterns that

suggest an underlying causal effect (that diversity plays a role in why neural networks perform well).

1.6 Dissertation overview

The dissertation is structured as follows:

- Chapter 2 provides the necessary background information and discusses work that relates to this study.
- Chapter 3 describes the implicit ensemble framework and analysis procedure used to construct and evaluate our implicit ensembles.
- Chapter 4 describes the datasets, model architectures, and training protocols used in all experiments. It outlines the general experimental pipeline, including how diversity measures are computed and compared across different network configurations.
- Chapter 5 explores diversity under different conditions. The chapter investigates layer-wise diversity across networks, the effect of network capacity (width) on diversity and performance, and the change in diversity throughout training. It also investigates correlations between diversity and generalisation performance.
- Chapter 6 investigates methods for intentionally inducing diversity within ANNs through architectural and regularisation-based mechanisms. The effects of these methods are analysed and compared with the baseline results from Chapter 5.
- Chapter 7 summarises the main findings and contributions of the dissertation, discusses their implications for understanding diversity in deep learning, and suggests future research on implicit ensembles and inducing diversity.

1.7 Publications

Portions of this dissertation have been published in the conference paper titled "Investigating the relationship between diversity and generalization in over-parameterized deep neural networks" [35]. This work has been accepted for publication in the *Proceedings of Machine Learning Research (PMLR)* and will be presented as a poster at the Northern Lights Deep Learning Conference in Norway in January 2026.

The conference paper and poster draw upon content from Chapters 5 and 6, where diversity measures are applied to over-parameterized deep networks, including a three-layer MLP and a CNN. Specifically, the overlapping sections concern the analysis of diversity in standard ANNs and in networks where diversity is intentionally induced through regularisation and architectural changes.

Chapter 2

Background

This chapter provides a background on the main concepts and tools that will be employed in this study.

2.1 Overview

This chapter provides the necessary background to understand ANNs and other theoretical concepts used in this study. We first give a brief overview of ANNs, focusing on MLPs and how they are constructed. We then introduce the main components of CNNs—namely convolution and pooling layers—to call attention to the structural differences between MLPs and CNNs.

Following this, we introduce explicit ensemble models, including how they help improve generalisation by aggregating the outputs across multiple models. Finally, we focus on diversity and various measures of it in explicit ensembles.

2.2 Artificial neural networks (ANNs)

Artificial neural networks (ANNs) (hereafter referred to simply as neural networks) are computational models composed of interconnected processing units—often called neurons, nodes, or hidden units [6, pp. 164–165][7, pp. 33–35]. Typically, each node computes a non-linear transformation of a weighted sum of its inputs. The weights of the connections between nodes are trainable parameters, and learning involves updating these parameters to minimise some loss function. These nodes are often organised into layers: an input layer that receives raw data, hidden layers that apply transformations to the data, and an output layer that produces predictions [7, pp. 25–36].

In feedforward networks, information flows strictly from inputs to outputs—there are no recurrent connections or cycles [7, p. 35]. A simple example of such an architecture is shown in Figure 2.1.

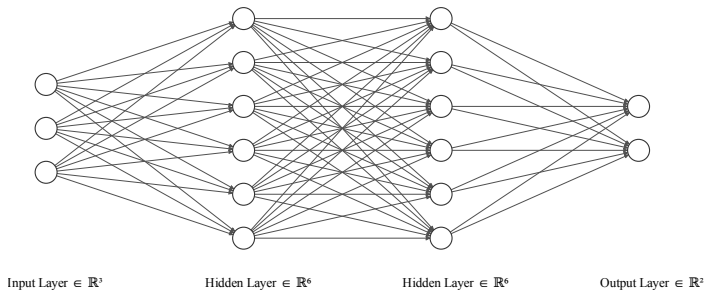


Figure 2.1: A fully connected neural network (MLP) with two hidden layers. The input layer feeds into the hidden layer nodes, which feed into the output layer. Each arrow represents a learned weight connecting the output of one node to the input of another. Activation functions (e.g. ReLU) are applied at hidden nodes, introducing non-linearity to the system [7, p. 29][6, p. 170]. Bias terms are ignored in this illustration.

2.2.1 Multilayer perceptrons (MLPs)

This study focuses on Multilayer perceptrons (MLPs) within a standard supervised learning setting, where the goal is to learn a function that maps input–output pairs using a dataset of labelled images as samples. Let

$$\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$$

be a dataset of input–output pairs sampled from an unknown joint distribution over $\mathcal{X} \times \mathcal{Y}$, where each input $\mathbf{x}^{(i)} \in \mathcal{X} \subset \mathbb{R}^{d_0}$ and each label $y^{(i)} \in \mathcal{Y} = \{1, 2, ..., C\}$.

An MLP is a type of feedforward network composed of multiple layers of nodes, where each node in one layer is connected to every node in the next—this is referred to as a *fully-connected* network. [7, p. 35]. The network is organised into an input layer, one or more hidden layers, and an output layer. Each hidden layer applies a linear transformation to its input, followed by a non-linear activation function (such as the Rectified linear unit (ReLU) function [36]) that introduces non-linearity into the network [7, pp. 45–49].

Formally, let $f_W : \mathcal{X} \to \mathbb{R}^C$ denote an MLP with L hidden layers, parameterised by a set of weights $\mathcal{W}$ and biases $\mathcal{B}$. f_W is defined recursively as follows: for $\ell \in \{1, \ldots, L\}$, we define the hidden layer $\mathbf{h}_{(\ell)} \in \mathbb{R}^{d_\ell}$ component-wise as

$$h_{(\ell)}^{n} = \sigma\left(b_{(\ell)}^{n} + \sum_{p=1}^{d_{\ell-1}} W_{(\ell-1),(l)}^{p,n} h_{(\ell-1)}^{p}\right), \tag{2.1}$$

where

- $h_{(\ell)}^{n}$ is the activation of node $n \in \mathcal{N}_{(\ell)} := \{1, \ldots, d_\ell\}$ in layer ℓ,
- $W_{(\ell-1),(l)}^{p,n}$ is the weight, from $\mathcal{W}$, connecting node p in layer $\ell - 1$ to node n in layer ℓ,
- $b_{(\ell)}^{n}$ is the bias term, from $\mathcal{B}$, connected to node n in layer ℓ,
- $\sigma(\cdot)$ is a non-linear activation function, and

- $h^p_{(\ell-1)}$ is the activation of node $p \in \mathcal{N}_{(\ell-1)} := \{1, \ldots, d_{\ell-1}\}$ in layer $\ell - 1$.

For $\ell = 1$, we define $\mathbf{h}_{(0)} = \mathbf{x}^{(i)}$. The final output of the network, i.e., the un-normalised class scores (logits) $\mathbf{y}' \in \mathbb{R}^C$, is computed after the last hidden layer (L). The output vector $\mathbf{y}'$ is the result of the final transformation

$$\mathbf{y}' = f_W(\mathbf{x}^{(i)}) = \mathbf{W}_{(out)}\mathbf{h}_{(L)} + \mathbf{b}_{(out)}. \tag{2.2}$$

where $\mathbf{W}_{(out)}$ is a $d_\ell \times C$ weight matrix and $\mathbf{b}_{(out)}$ refers to the the corresponding bias terms. The elements in these terms form part of $\mathcal{W}$ and $\mathcal{B}$, respectively. $\mathbf{y}'$ scores are usually passed through a softmax function to obtain predicted class probabilities [7, pp. 68–69].

In our study, we consider ReLU activation functions:

$$\sigma(z) = \max(0, z), \tag{2.3}$$

where z refers to the scalar pre-activation value that is fed into the activation function at any given node. This operation allows the network to learn piecewise affine functions [7, p. 25].

Training a neural network involves adjusting the parameters $\mathcal{W}$ and $\mathcal{B}$ to minimise a loss function based on $\mathbf{y}'$ and $y^{(i)}$—for classification problems, it is typically cross-entropy loss [7, pp. 71–72]. Parameters are adjusted with an *optimiser*. Optimisers usually use gradient-based methods such as gradient descent, Stochastic gradient descent (SGD), or Adaptive moment estimation (Adam) [7, pp. 77–91]. Gradients of the loss with respect to the parameters are computed using backpropagation [7, pp. 103–106], and the parameters are updated iteratively through training to reduce the prediction error and (ideally) improve generalisation to unseen data.

2.2.2 Convolutional neural networks (CNNs)

While standard MLPs (as described in Section 2.2.1) are powerful and effective function approximators in their own right, they often struggle to achieve high performance on complex image classification tasks without added data augmentation or a large amount of

training data. Modern deep learning models address this problem through using CNNs, which are specifically designed to exploit the spatial structure of image data [10].

CNN-based classifiers typically consist of two stages: a *convolutional stage*, which includes both convolutional and pooling layers, and a *fully-connected stage*, which is effectively an MLP responsible for the final classification [7, pp. 179–180].

Convolutional layers: As with the MLP, CNNs also consist of layers of hidden units that transforms the input. Following the definition from Prince [7, pp. 170–171] and the notation used in Section 2.2.1, let the input to layer ℓ be $\mathbf{H}_{(\ell-1)}$ as a tensor of height λ, width ω: $\mathbf{H}_{(\ell-1)} \in \mathbb{R}^{\lambda \times \omega}$ and let $\mathbf{W} \in \mathbb{R}^{K \times K}$ be a $K \times K$ convolutional kernel[2]. We determine the activation in a given layer ℓ at a spatial position (i, j) for any given output channel as:

$$H_{(\ell)}^{(i,j)} = \sigma\left(b_{(\ell)}^{(i,j)} + \sum_{\alpha=1}^{K}\sum_{\beta=1}^{K} W_{(\ell-1),(\ell)}^{(\alpha,\beta)} \cdot H_{(\ell-1)}^{(i+\alpha-[K-1],j+\beta-[K-1])}\right), \tag{2.4}$$

where

- $W_{(\ell-1),(\ell)}^{(\alpha,\beta)}$ is the kernel weight indexed by kernel position (α, β), and
- $b_{(\ell)}^{(i,j)}$ is a bias term.

This basically performs a weighted sum over a $K \times K$ patch of the input, which is then passed through a non-linear activation function $\sigma(\cdot)$. By shifting the kernel like a sliding window across all positions (i, j) of the input, the network computes a full two-dimensional output feature map for each kernel [7, pp. 170–171].

CNNs are well suited to processing images with multiple channels such as Red-Green-Blue (RGB) images, which consist of three colour channels—red, green, and blue—due to their ability to learn feature maps from the multi-channel inputs. When the input image

[2]A convolutional kernel refers to the set of shared weights $\mathbf{W}$ applied to each $K \times K$ patch of the input. Applying the kernel to a single patch produces one scalar output; sliding the kernel across the spatial dimensions produces a two-dimensional feature map, referred to as a channel. Multiple kernels therefore produce multiple output channels.

has ζ input channels, we denote it as $\mathbf{X} \in \mathbb{R}^{\lambda \times \omega \times \zeta}$, where $\zeta = 3$ for RGB images. When applying a convolutional kernel to this input, the kernel needs to be applied to all ζ input channels in parallel, and the channel-wise results summed to produce the final activation value. Consequently, each kernel requires learnable weights of shape $K \times K \times \zeta$ [7, pp. 170–171].

To produce multiple output channels (or feature maps), this process is repeated in parallel using different kernel weights for each output channel. For each output channel, the kernel combines information from all ζ input channels, so each output channel requires $\zeta \times K \times K$ weights and a single bias. In total, the full convolutional layer then contains $\zeta_{\text{out}} \times \zeta \times K \times K$ learnable weights and ζ_{out} biases [7, pp. 170–171].

Pooling layers: CNNs often include pooling layers, which usually perform downsampling to reduce the size of two-dimensional input representations [7, pp. 171–172]. The pooling operations function similar to the convolutions as a fixed-size window slides over each input channel. However, unlike convolutions, pooling contains no learnable parameters.

The two most common types of pooling are max pooling and average pooling. Max pooling selects the maximum value within each sliding window, while average pooling computes the mean of each window [37]. Pooling is important for two reasons: it reduces the scale of the input representations (the spatial dimensionality of the data) which lowers the computational cost of subsequent layers, and it introduces local invariance to small shifts or distortions in the input [37].

Stride and padding: Two additional hyperparameters that influence both convolutional and pooling operations are the *stride* and *padding*. The *stride* determines the step size by which the kernel window is shifted—larger stride values reduce the dimensions of the output [7, pp. 164–165]. For example, a two-dimensional input of size $N \times N$, kernel size $K \times K$, with padding P applied on each side, and stride s used in both spatial dimensions, will have an output feature map with a spatial dimension of $N_{\text{out}} \times N_{\text{out}}$,

where

$$N_{\text{out}} = \left\lfloor \frac{N - K + 2P}{s} \right\rfloor + 1.$$

The floor operator accounts for the case where $\frac{N-K+2P}{s}$ is not an integer—the final kernel placement that would extend beyond the padded input is omitted. Padding refers to the addition of extra values (typically zeroes) around the border of the input, allowing control over the size of the output feature maps. Without padding, repeated convolutions progressively shrink the spatial dimensions, whereas appropriate padding can preserve the input height and width [7, p. 164].

Final classification via MLP: After several convolutional and pooling layers, the output feature maps are flattened into a one dimensional vector. This vector is then passed into a set of fully-connected layers which form the final MLP stage of the CNN. This MLP stage operates identically to the MLP formulation described in Section 2.2.1. This stage is responsible for producing the final class predictions.

2.3 Ensemble theory

This section introduces the essential concepts of ensemble theory and forms the basis for their application and interpretation in the remainder of this study.

2.3.1 What is an ensemble?

Ensemble learning in machine learning is when multiple models—called ensemble members or base learners—are trained to solve the same task, and their predictions are combined or aggregated together to get a final prediction or output [38]. By combining multiple models, the goal is to produce a system or an *ensemble model* that has better generalisation performance than any individual member of the ensemble. Empirical and theoretical results consistently show that ensembles often outperform single models, particularly when

the individual learners are moderately accurate and diverse in their predictions [30], [39], [40].

In the context of deep learning, a Deep ensemble (DE) refers to an ensemble consisting of multiple Deep neural networks (DNNs). These DNNs, which are trained independently from each other, may share the same architecture and training procedure but differ due to variations in initialisation, data sampling, or training noise [30], [41]. We mention different methods for ensembling in Section 2.3.2.

There are different ways of aggregating the results of ensembles, and the method of choice depends on the task.

- **Classification:** Common methods for combining classification predictions include: *majority voting, soft voting* (averaging class probabilities), and *Bayesian model averaging* [30], [42, pp. 111–150].
- **Regression:** Outputs are typically combined by *mean* or *median* averaging due to the nature of the type of output [30].

According to the Jury Theorem [40]—which states that, under certain assumptions, aggregating the decisions of multiple independent and competent voters increases the probability of a correct collective decision—an ensemble will have improved accuracy if two key conditions are met. From this point forward, we focus specifically on the classification setting and refer to ensemble members simply as *classifiers*.

1. **Accuracy:** Each classifier must be better than random guessing (e.g., has accuracy > 0.5 on a balanced binary classification task).
2. **Diversity:** The classifiers must make *uncorrelated* or *diverse* errors.

If the classifiers within an ensemble are highly correlated (i.e., they make similar errors), then combining them adds little value. On the other hand, if classifiers are diverse and make different mistakes, the errors can be averaged out through aggregation [28], [39].

2.3.2 Methods for constructing ensembles

Ensembling refers to the process of combining multiple models to produce a single, typically more accurate predictor—which in our study is a classifier. The effectiveness of an ensemble depends on how its constituent models are constructed or trained so that they make less- or uncorrelated errors. There are various methods for generating such ensembles in the literature, each encouraging diversity through different methods.

One such group of methods use data-centric methods like *bagging* (bootstrap aggregating) [43], and *boosting* [44], [45] to generate subsets or variations of the ensemble's training data. Related approaches such as *stacking* [46] instead combine models by integrating their predictions.

In bagging, multiple new training sets are generated by randomly sampling—usually with replacement—from the original training set. Each model is trained on one of these different training sets which, due to the differences in the data each model receives, leads to classifiers whose errors differ. This enables the aggregated prediction to "average out" individual overfitting behaviours [38], [43], [47].

Instead of training models independently on re-sampled data, boosting trains models sequentially. Successive models' training then focus on examples that previous models found difficult, i.e., that they failed on more frequently. This produces an ensemble of different models whose strength compensate for each other's weaknesses [44], [45].

Stacking, as the name implies, involves combining models by taking the predictions of the classifiers within an ensemble, and feeding them to another "meta-model". This meta-model is trained to integrate the outputs in such a way that overall performance is maximised [46].

2.4 Diversity in explicit ensembles

Understanding why diversity is important for ensemble performance requires examining how ensembles improve generalisation and how diversity contributes to this effect. In this section, we outline generalisation in ensembles before discussing why diversity is a necessary component for achieving improved performance. We then explore different diversity measures from the literature.

2.4.1 Generalisation in ensembles

Ensembles are used as a method to improve generalisation performance. As with other regularisation techniques (e.g., Dropout, weight decay, data augmentation [7, pp. 144–155]), the strength of ensembling lies in its ability to mitigate overfitting—which is when a model captures patterns that are likely unique to a particular training set (possibly due to noise), typically leading to reduced test set performance. While an individual model may overfit to particular structures or noise in the training set, multiple models trained with different initialisations may overfit *differently*. When these models are combined, their errors may be uncorrelated and tend to "cancel out", leading to improved test set performance [30], [39], [48].

This effect can be formalised through the bias-variance decomposition: ensembles often reduce variance without significantly increasing bias [38]. For detailed examples of such decompositions, we point the reader to the work by Geman et al. [49], Domingos et al. [50], Wood et al. [51], and Brown et al. [52]. DNNs tend to have high variance, and may fluctuate significantly depending on the training data or initialisation. By averaging across such models, DEs stabilise the predictions and reduce the likelihood of poor generalisation due to any one model's high variance [41].

However, for an ensemble to be effective, the classifiers within the ensemble should not only be accurate, but should also make different or *diverse* errors. Diversity refers to producing uncorrelated predictions. Several decomposition frameworks, such as the classical bias-

variance decomposition [49], [50], the bias-variance-covariance decomposition [30], [52], and the more recent bias-variance-diversity decomposition [51], all support the notion that ensemble diversity directly contributes to improving generalisation performance. These perspectives motivate the study of explicit measures of diversity in ensembles, which we discuss next.

2.4.2 Diversity measures

There are numerous metrics in the literature that aim to capture different aspects of diversity in ensembles. In this study, we evaluate nine well-known diversity measures in a classification setting. These are: the *Q-statistic*, *Correlation coefficient*, *Kappa-statistic*, *Disagreement*, *Kohavi–Wolpert variance*, *Double-fault*, *Generalised diversity*, *Coincident failure diversity*, and *Entropy* [53, pp. 105–109], [28], [42, pp. 300–306]. We also introduce a variant of *Entropy* that is computed from binary correct/incorrect predictions, which we simply refer to as *Binary Entropy*.

The diversity measures used in this study were selected following a literature review of ensemble learning. The focus of this review was on established approaches for measuring and quantifying the diversity of an ensemble. Diversity is commonly characterised in terms of disagreement among classifiers in an ensemble, and the chosen metrics each quantify specific aspects of agreement or disagreement—either between pairs of classifiers or across the entire ensemble. Our choice of measures are based off of the well-cited work by Kuncheva [28], [42] and Zhou [53].

A summary of the diversity measures and how to interpret their values—specifically what indicates higher or lower diversity—can be found in Table 2.2.

These measures can be grouped into two broad categories: *pairwise* and *non-pairwise* metrics.

- **Pairwise**: Evaluates diversity between pairs of classifiers within the ensemble. Measures under this category generally quantify how often the pairs agree or disagree in

their predictions. The results of these pairwise comparisons are summarised using a 2×2 contingency table, as shown in Table 2.1.

- **Non-pairwise:** These measures evaluate the diversity of the ensemble as a whole by considering the collective behaviour of all classifiers on a given input.

2.4.3 Pairwise measures

In this section, we define each of the pairwise diversity measures used in this study, together with a description of their value ranges and the interpretations associated with higher or lower diversity.

As outlined previously, pairwise diversity measures assess the diversity between pairs of classifiers in an ensemble. To evaluate the diversity of the entire ensemble under a measure, the standard approach is to compute the average values for that measure across all unique classifier pairs [53, p. 105].

Calculating the contingency table: To compute any of the pairwise diversity measures, we construct a 2×2 contingency table that summarises the classification outcomes between a pair a classifiers. Assuming a C-class classification problem, let D_a and D_b be two classifiers from an ensemble. Given a dataset $\left\{(\mathbf{x}^{(i)}, y^{(i)})\right\}_{i=1}^{N}$, we define the following four frequency metrics regarding the prediction outcomes of D_a and D_b:

- N^{11}: Number of samples for which both D_a and D_b are correct on the same sample.
- N^{10}: Number of samples for which D_a is correct, and D_b is incorrect on the same sample.
- N^{01}: Number of samples for which D_a is incorrect, and D_b is correct on the same sample.
- N^{00}: Number of samples for which both D_a and D_b are incorrect on the same sample.

Since every sample must fall into exactly one of these four categories, the sum of these four frequencies must equal the total number of samples (N) in the dataset:

$$N = N^{11} + N^{10} + N^{01} + N^{00}$$

Table 2.1: 2×2 contingency table for pairwise measures. See text for details.

	D_b correct (1)	D_b incorrect (0)
D_a correct (1)	N^{11}	N^{10}
D_a incorrect (0)	N^{01}	N^{00}

Limitations of using binary outcomes: We acknowledge that by operating on binary outcomes derived from the multi-class predictions (as per the formulation of the classical ensemble diversity measures here-after), we discard information contained in the full multi-class output distribution. In particular, differences in confidence and near-miss predictions are not captured by binary agreement statistics. Extending diversity analysis to operate directly on multi-class predictions constitutes a valid and potentially informative alternative. Such extensions are beyond the scope of the present study but represent a promising direction for future work.

Q-statistic **(Q):** The *Q-statistic*, originally proposed by Yule [54, p. 271], is a classical measure of association between two binary variables. While Yule originally developed the statistic for general statistical association, it has been adapted for the context of ensemble learning where it quantifies the similarity of predictions between two classifiers, D_a and D_b, based on their correctness [28].

Using the 2×2 contingency table (Table 2.1), the *Q-statistic* is defined as:

$$Q_{a,b} = \frac{(N^{11}N^{00} - N^{10}N^{01})}{(N^{11}N^{00} + N^{10}N^{01})} \tag{2.5}$$

The numerator reflects the difference in the frequency of agreement versus disagreement, while the denominator normalises the result to lie within $[-1, 1]$. A value of $Q = 0$ indicates that there is no association between the classifiers (they are independent). Positive

values (e.g., $Q > 0$) indicates a tendency to make the same predictions, whereas negative values ($Q < 0$) suggests that when one classifier is correct, the other is likely to be incorrect. Values of $Q = +1$ or $Q = -1$ indicate perfect agreement or perfect disagreement, respectively.

In ensemble learning, as the ensemble benefits from classifiers making different errors on the same inputs, a lower Q value signifies greater diversity (especially values approaching -1, as this is when classifiers always make different errors). The *Q-statistic* depends on the joint counts of correct and incorrect predictions of the pairwise classifiers. Q may indicate high agreement (low diversity) simply because both classifiers perform well, conversely, if both classifiers are weak, Q may also suggest high agreement due to shared failures. Thus, we can expect the *Q-statistic* to be sensitive to the accuracies of the classifiers.

The *Q-statistic* remains a widely used measure for diversity in explicit ensembles due to its simplicity and interpretability when quantifying pairwise diversity [28], [53, p. 105].

To assess the diversity of the entire ensemble, we need to compute the *average Q-statistic* across all unique pairs of classifiers. For an ensemble of T classifiers, with $\frac{T(T-1)}{2}$ distinct pairs, the average is given by:

$$Q_{av} = \frac{2}{T(T-1)} \sum_{a=1}^{T-1} \sum_{b=a+1}^{T} Q_{a,b} \tag{2.6}$$

***Correlation coefficient* (ρ):** The *Correlation coefficient* ρ, originally introduced by Sneath et al. [55], can be used as a metric to assess the similarity between pairs of classifiers D_a and D_b using the contingency table (Table 2.1) in a similar fashion to the *Q-statistic* [28].

The *Correlation coefficient* is defined as:

$$\rho_{a,b} = \frac{(N^{11}N^{00} - N^{10}N^{01})}{\sqrt{(N^{11} + N^{10})(N^{01} + N^{00})(N^{11} + N^{01})(N^{10} + N^{00})}}, \tag{2.7}$$

where ρ lies within $[-1, 1]$. Similar to the *Q-statistic*, $\rho = 1$ indicates a perfect positive correlation, $\rho = -1$ indicates a perfect negative correlation, and $\rho = 0$ implies there is no

linear correlation.

The value of $\rho_{a,b}$ is influenced by the accuracies of the individual classifiers. This can be seen through the contingency table entries (Table 2.1), where the terms $(N^{11} + N^{10})$ and $(N^{11} + N^{01})$ represent the total number of correct predictions made by classifiers D_a and D_b, respectively. As classifier accuracies increase, the value of N^{11} grows, which increases the product $N^{11}N^{00}$ relative to $N^{10}N^{01}$ in the numerator of Equation (2.7). Increased N^{11} tends to increase $\rho_{a,b}$, reflecting stronger agreement between classifiers. However, if classifiers make errors on different samples, the terms N^{10} and N^{01} increase, which raises the product $N^{10}N^{01}$ and reduces $\rho_{a,b}$. We can therefore assume that the *Correlation coefficient* captures both classifier accuracy and the extent to which their correct and incorrect predictions coincide.

We calculate the average correlation ρ_{av} across all T ensemble members using the same formulation as in Equation (2.6) for Q_{av}.

***Disagreement* measure (dis):** The *Disagreement* measure, introduced by Skalak et al. [56] and later used by Ho et al. [57], quantifies the relationship between the number of observations in which classifier D_a is correct, while D_b is incorrect or vice versa, divided by the total number of observations [28], [53, pg. 105].

Formally, the *Disagreement* between two classifiers D_a and D_b is defined as:

$$\text{dis}_{a,b} = \frac{N^{10} + N^{01}}{N}. \tag{2.8}$$

We define the *Disagreement* between two classifiers D_a and D_b specifically based on their state of prediction correctness on a sample $\mathbf{x}^{(i)}$. In other words, the *Disagreement* metric measures the proportion of instances where the classifiers exhibit a difference in their correctness status (N^{10} or N^{01}). Since $N^{10} + N^{01}$ represents the number of times the two classifiers disagreed, and the numerator can never be greater than N, the measure is bounded between $[0, 1]$.

A *Disagreement* value of dis $= 0$ indicates that D_a and D_b always agree on the same

instances—both are always correct, or always incorrect—and dis = 1 is when D_a and D_b never agree, indicating a maximum level of diversity. A dis = 1 is ideal if we want to determine which classifiers complement each other the most—in other words, which classifiers' errors overlap the least.

The value of $\text{dis}_{a,b}$ is influenced by the accuracies of the individual classifiers through the contingency counts N^{10} and N^{01}. When both classifiers have high accuracy and tend to succeed on the same inputs, N^{11} increases while N^{10} and N^{01} decrease, leading to smaller *Disagreement* values. If the classifiers make errors on different values, N^{10} and N^{01} increase, which raises $\text{dis}_{a,b}$. When one classifier is substantially more accurate than the other, *Disagreement* may also increase because the weaker classifier will make more errors on many samples where the stronger classifier is correct, increasing N^{10} and N^{01}.

The average *Disagreement* (dis_{av}) across all T ensemble members is calculated using Equation (2.6) for Q_{av}.

***Double-fault* measure (df):** Giacinto et al. [29] made use of this measure to assess the diversity of their classifiers by simply counting how often two classifiers made the same error on the same instance. If both classifiers fail frequently on the same instances (they are both misclassifying the same input), they are considered less diverse. The double-fault measure is represented as:

$$\text{df}_{a,b} = \frac{N^{00}}{N}, \tag{2.9}$$

where N^{00} is where both classifiers make an error on the same instance, and N is the total number of instances [28], [53, pg.106]. The *Double-fault* measure is useful as it directly quantifies how often classifiers fail together, which is a practical (and interpretable) indicator of whether combining them in an ensemble will lead to a significant error reduction. Classifiers that are less likely to fail on the same instances are more likely to provide complementary strengths to the ensemble.

Double-fault is bounded within $[0, 1]$, where df = 0 indicates *maximum* diversity (the classifiers never make the same error), and df = 1 indicates *no diversity* (the classifiers always fail on the same instances). This behaviour contrasts with measures such as

Disagreement, where larger values correspond to higher diversity. The value of $\mathrm{df}_{a,b}$ is influenced by the accuracies of the individual classifiers through the contingency count N^{00}, which represents the number of samples on which both classifiers are incorrect. As the accuracies of the classifiers increase, N^{00} typically decreases, resulting in smaller values of $\mathrm{df}_{a,b}$. In contrast, if the classifiers are less accurate, or if the errors tend to occur on the same samples, N^{00} increases and $\mathrm{df}_{a,b}$ becomes larger.

The average df_{av} across all T ensemble members can be calculated with Equation (2.6) for Q_{av}.

***Kappa-statistic* (κ_p):** Cohen's *Kappa-statistic* is a measure designed to assess the level of agreement between two classifiers while considering the possibility of *agreement by chance* [58]. The measure was introduced to address a limitation of simple agreement-based measures where they do not account for agreement that could occur purely by coincidence. The *Kappa-statistic* addresses this by comparing the observed agreement to the expected agreement by chance.

κ_p is calculated as:

$$\kappa_p = \frac{(P_{o_{a,b}} - P_{e_{a,b}})}{(1 - P_{e_{a,b}})}, \tag{2.10}$$

where $P_{o_{a,b}}$ is the observed agreement (the proportion of instances on which the classifiers agree), and $P_{e_{a,b}}$ is the expected agreement by chance [28], [53, pg. 106]. These two quantities can be determined using the contingency table (Table 2.1):

$$P_{o_{a,b}} = \frac{(N^{11} + N^{00})}{N}, \tag{2.11}$$

$$P_{e_{a,b}} = \frac{[(N^{11} + N^{10})(N^{11} + N^{01}) + (N^{01} + N^{00})(N^{10} + N^{00})]}{N^2} \tag{2.12}$$

The denominator, $1 - P_{e_{a,b}}$, represents the proportion of instances where we would expect disagreement by chance. It therefore normalises the difference between the observed agreement and the expected agreement. The numerator captures how much the observed agreement exceeds what would be expected by chance alone.

κ_p attains a maximum value of 1 under perfect agreement. A value of 0 indicates agree-

ment no better than chance. Negative values indicate agreement worse than would be expected by chance, implying some systematic disagreement. Although the *Kappa-statistic* is often described as ranging in $[-1, 1]$, its minimum attainable value depends on the marginal distributions of the classifiers and is not necessarily equal to -1 in all cases. These marginal distributions correspond to the row and column totals (e.g., $N^{11} + N^{10}$ and $N^{11} + N^{01}$), which represent the individual accuracies of the classifiers.

From the expressions for $P_{o_{a,b}}$ and $P_{e_{a,b}}$, it is evident that both terms depend directly on the marginal distribution of the classifiers. As classifier accuracy increases, N^{11} may increase while N^{10}, N^{01}, and N^{00} decrease. This increases the observed agreement $P_{o_{a,b}}$. The expected agreement $P_{e_{a,b}}$ is also affected because it is computed from the marginal totals of the contingency table. These marginals correspond to the individual correctness frequencies of the classifiers, namely $(N^{11} + N^{10})$ and $(N^{01} + N^{00})$ for classifier D_a, and $(N^{11} + N^{01})$ and $(N^{10} + N^{00})$ for classifier D_b. Since $P_{e_{a,b}}$ is calculated from products of these marginal probabilities, changes in classifier accuracy alter the level of agreement expected under independence. Consequently, κ_p may vary due to shifts in the marginal performance of the classifiers. Therefore, although κ_p adjusts for chance agreement, it remains intrinsically coupled to the classifiers' accuracies.

By measuring the agreement (or disagreement) between individual classifiers in an ensemble, κ_p helps to gauge whether the classifiers are making independent errors. For use in ensembles, ensemble members/classifiers with lower κ_p scores indicate higher diversity.

We obtain an averaged *Kappa-statistic*, $\kappa_{p_{av}}$ across all T ensemble members using Equation (2.6) for Q_{av}.

2.4.4 Non-pairwise measures

In this section, we define the non-pairwise measures which evaluate the ensemble's diversity as a whole, rather than through pairwise comparisons. These measures consider the distribution of predictions across the entire ensemble for each input sample.

***Kohavi–Wolpert variance* (KW):** The *Kohavi–Wolpert variance* [59] originates from the bias-variance decomposition for a *single classifier* model under a zero-one loss. It quantifies the variability of predicted class labels for a given input $\mathbf{x}^{(i)}$ across multiple training sets.

For a classification task with C classes, let $P\big(c \mid \mathbf{x}^{(i)}\big)$ denote the probability assigned to class $c \in \{1, \ldots, C\}$ for sample $\mathbf{x}^{(i)}$. Then the variance for a given input $\mathbf{x}^{(i)}$ is given by:

$$\text{variance}_{\mathbf{x}^{(i)}} = \frac{1}{2}(1 - \sum_{c=1}^{C} P(c \mid \mathbf{x}^{(i)})^2), \tag{2.13}$$

where $\sum_{c=1}^{C} P(c \mid \mathbf{x}^{(i)})^2$ represents the concentration of the classifier's predicted probabilities, with higher values indicating lower variance. Note that $P(c \mid \mathbf{x}^{(i)})$ is obtained as an average across different datasets.

Kuncheva et al. [28] extended the concept of variance from Kohavi et al. [59], adapting it to classifier ensembles instead of individual classifiers. In particular, Kuncheva et al. [28] consider the variance over the set of classifiers $\{D_1, \ldots, D_T\}$ given an input $\mathbf{x}^{(i)}$. To achieve this, Kuncheva et al. [28] reduce the (potentially) multi-class problem to a binary problem by considering two classifier outputs, namely, correct ($c = 1$) and incorrect ($c = 0$). Then to estimate $P\big(c = 1 \mid \mathbf{x}^{(i)}\big)$ and $P\big(c = 0 \mid \mathbf{x}^{(i)}\big)$, Kuncheva et al. [28] uses an average over $\{D_1, \ldots, D_T\}$:

$$\hat{P}\big(c = 1 \mid \mathbf{x}^{(i)}\big) = \frac{l(\mathbf{x}^{(i)})}{T} \tag{2.14}$$

$$\hat{P}\big(c = 0 \mid \mathbf{x}^{(i)}\big) = \frac{T - l(\mathbf{x}^{(i)})}{T} \tag{2.15}$$

where $l(\mathbf{x}^{(i)})$ is the number of classifiers in the ensemble that correctly classify input $\mathbf{x}^{(i)}$. The hat notation ($\hat{P}$) indicates that these are estimated probabilities.

These estimated probabilities are then substituted into Equation (2.13):

$$\text{variance}_{\mathbf{x}^{(i)}} = \frac{1}{2}\left(1 - \hat{P}\big(c = 1 \mid \mathbf{x}^{(i)}\big)^2 - \hat{P}\big(c = 0 \mid \mathbf{x}^{(i)}\big)^2\right) \tag{2.16}$$

which then gives:

$$\text{variance}_{\mathbf{x}^{(i)}} = \frac{1}{2}\left(1 - \left(\frac{l(\mathbf{x}^{(i)})}{T}\right)^2 - \left(\frac{T - l(\mathbf{x}^{(i)})}{T}\right)^2\right) \tag{2.17}$$

To compute a measure of diversity for the entire dataset, Kuncheva et al. [28] averaged this variance over all inputs $\mathbf{x}^{(i)} \in N$, where N is the number of samples. The resulting measure, known as the *Kohavi–Wolpert variance* (KW), is:

$$\mathrm{KW} = \frac{1}{NT^2} \sum_{i=1}^{N} l(\mathbf{x}^{(i)})(T - l(\mathbf{x}^{(i)})). \tag{2.18}$$

Kuncheva et al. [28] noticed that their diversity measure KW is a scaled version of the averaged *Disagreement* measure:

$$\mathrm{KW} = \frac{T-1}{2T} dis_{av} \tag{2.19}$$

For this measure, higher values indicate more disagreement/diversity among classifiers. KW has a range of $\mathrm{KW} \in [0, 0.5]$. $\mathrm{KW} = 0$ when all classifiers always agree (which means no diversity), and reaches its maximum $\mathrm{KW} = 0.5$ when $T/2$ classifiers are correct for each instance (which is the maximum diversity) [28].

Since KW is a scaled version of the averaged *Disagreement*, it inherits the same intrinsic coupling to classifier performance and must therefore be interpreted in a similar manner.

Generalised diversity **(**GD**):** The *Generalised diversity* measure was introduced by Partridge et al. [60]. Rather than examining pairwise classifier disagreement, the *Generalised diversity* measure considers how often an input is misclassified by a specific number of classifiers. In other words, it helps quantify the independence of failure between classifiers (how much more likely it is for one classifier to be correct, while the other is incorrect).

It is formally defined as:

$$\mathrm{GD} = \frac{p(1) - p(2)}{p(1)}, \tag{2.20}$$

where $p(1)$ represents the probability that a single randomly chosen classifier fails on a randomly chosen sample and $p(2)$ represents the probability that two randomly chosen classifiers both fail on a randomly chosen sample. The expression $p(1) - p(2)$ quantifies how independent the failures of the classifiers are.

The probabilities $p(1)$ and $p(2)$ are estimated empirically by counting, for each input sample $\mathbf{x}^{(i)}$, the number of classifiers that misclassify it. This forms a corresponding empirical frequency distribution following the procedure described by Kuncheva [28], [42]. Let m_i denote this count on each sample $\mathbf{x}^{(i)}$, so that $m_i \in \{0, \ldots, T\}$. Empirically, the probability p_m that exactly m classifiers fail on an input can be estimated as

$$p_m = \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}\{m_i = m\}, \tag{2.21}$$

where $\mathbf{1}\{\cdot\}$ is an indicator function that equals 1 when the condition holds and 0 otherwise. Thus, p_m represents the fraction of samples for which exactly m classifiers misclassify the input.

Given the distribution $\{p_m\}_{m=0}^{T}$, we can then calculate the probabilities $p(1)$ and $p(2)$:

$$p(1) = \sum_{m=1}^{T} \frac{m}{T} p_m, \tag{2.22}$$

$$p(2) = \sum_{m=2}^{T} \frac{m}{T} \frac{(m-1)}{(T-1)} p_m \tag{2.23}$$

It is important to note the difference between the probability p_m and the probabilities $p(1)$ and $p(2)$. p_m is the probability that *exactly* m classifiers fail on a sample. $p(1)$ is the probability that a *randomly* selected classifier fails on a randomly selected sample, and $p(2)$ that two *randomly* selected classifiers both fail on a randomly selected sample[3].

GD is a value that lies in $[0, 1]$, where a GD $= 0$ represents the minimum diversity ($p(2) = p(1)$), and GD $= 1$ corresponds to maximum diversity, where $p(2) = 0$, meaning no two classifiers fail on the same instance. Since $p(1) = \sum_{m=1}^{T} \frac{m}{T} p_m$, as classifier accuracy increases, the distribution $\{p_m\}$ shifts toward lower values of m, thereby reducing $p(1)$. Simultaneously, $p(2) = \sum_{m=1}^{T} \frac{m}{T} \frac{(m-2)}{(T-1)} p_m$ also decreases, since joint failures become less frequent when overall misclassification counts decline. In the extreme case where all classifiers are perfectly accurate, we obtain $p_0 = 1$, implying $p(1) = p(2) = 0$, and hence GD is interpreted as exhibiting no failure-based diversity. Conversely, when classifiers

[3]The classifiers are selected *without replacement*. This is reflected in the term $\frac{m-1}{T-1}$ in the definition of $p(2)$.

exhibit moderate error rates allow GD to attain larger values. Thus, the accuracy of the classifiers affects the diversity as measured by the *Generalised diversity* measure.

Coincident failure diversity **(**cfd**):** Also by Partridge et al. [60], this measure is a modification of *Generalised diversity* that places stronger emphasis on how classifier errors coincide across the entire ensemble.

We define the *Coincident failure diversity* as:

$$\text{cfd} = \begin{cases} 0, & p_0 = 1; \\ \frac{1}{1-p_0} \sum_{m=1}^{T} \frac{T-m}{T-1} p_m, & p_0 < 1 \end{cases} \tag{2.24}$$

Here, p_0 refers to the proportion of input samples on which none of the classifiers in the ensemble make an error. When cfd $= 1$, it is at its maximum, meaning that each classifier makes mistakes on unique instances. When all classifiers give the same predictions then cfd $= 0$ and the diversity is at a minimum.

Similar to *Generalised diversity*, cfd depends directly on the distribution $\{p_m\}_{m=0}^{T}$, and in particular on p_0. As classifier accuracies increase, m shifts toward 0, increasing p_0 and reducing the contribution of coincident failures in the summation term. Consequently, improvement

Entropy **($\mathbf{Ent_{CC}}$):** *Entropy*, proposed as a diversity measure for classifier ensembles by Cunningham et al. [61], quantifies the uncertainty or randomness of predictions made by classifiers within the ensemble. Unlike previous measures that are based on oracle outputs (i.e., whether predictions are correct or incorrect), *Entropy* considers the raw multi-class predictions made by individual classifiers—regardless of whether they are correct.

To formally define *Entropy* in the ensemble setting, consider a sample $\mathbf{x}^{(i)}$. Let $P(c \mid \mathbf{x}^{(i)})$ denote the proportion of classifiers in the ensemble that assign $\mathbf{x}^{(i)}$ to class c, where $c \in \{1, \ldots, C\}$. This can be calculated as:

$$P(c \mid \mathbf{x}^{(i)}) = \frac{1}{T} \sum_{m=1}^{T} \mathbf{1}\{D_m(\mathbf{x}^{(i)}) = c\}, \tag{2.25}$$

where T is the number of classifiers, $D_m(\mathbf{x}^{(i)})$ is the class predicted by the m-th classifier for the input $\mathbf{x}^{(i)}$, and $\mathbf{1}\{\cdot\}$ is an indicator function which equals 1 if the condition inside is true, and 0 otherwise. The per-sample entropy is then calculated as:

$$H(\mathbf{x}^{(i)}) = -\sum_{c=1}^{C} P(c \mid \mathbf{x}^{(i)}) \log P(c \mid \mathbf{x}^{(i)}). \tag{2.26}$$

Using Equation (2.26), we can calculate the average per-sample entropy over the dataset:

$$\text{Ent}_{\text{CC}} = \frac{1}{N} \sum_{i=1}^{N} H(\mathbf{x}^{(i)}). \tag{2.27}$$

In this context, *Entropy* acts as a proxy for disagreement; high entropy indicates that the classifiers are highly varied in their predictions—i.e., greater diversity among classifier predictions—while low entropy suggests that the classifiers are largely in agreement—lower diversity [28].

Entropy is computed using the base-2 logarithm and is bounded between $[0, \log C]$, where C is the number of classes. For the image classification tasks that are employed in this study, $C = 10$, which means that *Entropy* is bounded by $[0, 3.322]$.

Binary Entropy **(Ent_{bin}):** The *Binary Entropy* measure uses the same formulation as the standard *Entropy* measure, but instead of considering multi-class predictions, it employs binary oracle outputs—where each classifier's prediction for a sample is reduced to being either correct or incorrect. For a given input $\mathbf{x}^{(i)}$, we compute the proportion of correct and incorrect predictions across the ensemble (as in Equation (2.25)), and then substitute these values into Equation (2.26). To determine the average entropy over the dataset, we use Equation (2.27). Similar to the multi-class *Entropy*, higher values indicate greater diversity among ensemble members.

The range for *Binary Entropy* is $[0, \log 2]$, which is then $[0, 1]$.

Table 2.2: Summary of ensemble diversity measures, where ↑ (↓) indicates that the larger (smaller) the measurement, the higher the estimated diversity.

Diversity Measure	**Symbol**	↑ / ↓	**Pairwise**
Disagreement	dis	↑	Yes
Q-statistic	Q	↓	Yes
Correlation coefficient	ρ	↓	Yes
Kappa-statistic	κ_p	↓	Yes
Double-fault	df	↓	Yes
Kohavi–Wolpert variance	KW	↑	No
Generalized diversity	gd	↑	No
Coincident failure	cfd	↑	No
Entropy (C&C's)	Ent_{CC}	↑	No
Binary Entropy	Ent_{bin}	↑	No

2.5 Summary

In this chapter, we introduced the foundational concepts necessary to understand the experiments and analyses that follow. By examining the structure of both MLPs and CNNs, we outlined how these networks operate on a node-level. We then presented key ideas from ensemble learning where we noted that generalisation can be improved by combining multiple diversity classifiers within an ensemble. This led to a discussion on several diversity measures that are widely used when studying ensembles. We discussed how metrics capture different aspects of disagreement or correlation among classifiers within an ensemble, and summarised these in Table 2.2.

In the next chapter, we extend the ideas of ensembles and ensemble diversity to the notion of *implicit ensembles*, where substructures or subcomponents inside a *single* neural network are treated as members of an ensemble.

Chapter 3

Implicit ensembles

This chapter introduces the implicit ensemble view of neural networks—motivated by ideas of modularity, circuits, and clustering—and formalises how individual nodes can be treated as classifiers following the framework of Davel et al. [1].

3.1 Overview

A complete characterisation of the mechanisms behind neural network generalisation is still an open problem. One approach is to analyse networks in terms of their internal subcomponents. In this chapter, we adopt this perspective by viewing a *single* neural network as an *implicit ensemble*, following the work by Davel et al. [1].

We begin by outlining several related works and perspectives—modularity, circuits, and clustering—that all suggest trained networks contain meaningful internal structure. We then formalise our implicit ensemble construction, describing how node-level subpredictors are obtained and how their outputs are combined at the layer level. Finally, we present a preliminary experimental verification showing that many nodes achieve accuracy above

chance, supporting their feasibility as subpredictors.

3.2 Background

In this section, we discuss several of the approaches used to study the internal substructures of *single* neural networks. Although these works do not represent the full breadth of research on internal network structure, it captures the main ideas that are relevant to this study.

3.2.1 Modularity

In neural networks, modularity refers to the presence of distinct structural or functional subnetworks (i.e., modules) within a model. These modules emerge when nodes are grouped together, either because they specialise in different sub-tasks (functional modularity), or because their internal connectivity is stronger than their connectivity to the rest of the network (structural modularity). In principle, if subnetworks implement *reusable functions*, a network could gain benefits like compositionality (recombining simple functions to solve complex tasks), improved interpretability, and reduced interference between tasks.

Multiple studies investigate the idea of modularity from either the structural or the functional decomposition of the neural network.

Structural modularity: Watanabe et al. [62]–[66] and Filan et al. [67] examine modularity from a *structural* perspective, demonstrating that trained neural networks show community-like connection patterns—characterised by strong connections within a community and weaker connections between communities—such that the network can be decomposed into groups of nodes, or *modules* that can be extracted and analysed. These modules can then be used for tasks such as pruning (making neural network smaller by

removing unimportant connections or nodes) without affecting generalisation error.

Structural modularity is also sometimes referred to as "clustering" [25], [68] where these two terms are often used interchangeably in literature.

Functional modularity: Other works examine functional rather than structural modularity, i.e., they identify subnetworks that are responsible for specific functions, although how they identify these subnetworks differ. For example, Csordás et al. [24] identify these functional subnetworks using learned binary masks that isolate the minimal set of connections required for a task; Zhang et al. [69] and Patil et al. [70] similarly identify subnetworks based on modular task structures. Patil et al. [70] identify hierarchically organised functional modules by iteratively pruning connections and then clustering nodes within layers that have strong connections to each other. They then merge clusters with considerable overlap in connectivity across layers into functional modules.

Another way of describing functional modularity is through "circuits", which describe small interconnected groups of nodes that collectively implement a specific feature or sub-task. Olah et al. [23] introduced the "circuits" concept in interpretability research, stating that features connect via weights to form circuits within a network. These circuits are essentially sub-graphs of the network that can be isolated and studied to understand how the network processes information.

Discussion: These studies all point to the idea that neural networks tend to develop a latent modular structure, whether measured by structural connectivity patterns—as in community detection or clustering—or identifiable subcomponents with a task specific function—where subnetworks specialise in particular tasks. This modular behaviour is useful for interpretability research and for understanding why neural networks work so well. By recognising that there are multiple methods for decomposing neural networks into meaningful substructures, we motivate further investigation into alternative frameworks—such as implicit ensembles—that may reveal additional insights into a neural network's internal structure and workings.

3.2.2 Implicit Ensembles

Where modularity focuses on *structural* or *functional* subnetworks within a model, the implicit ensemble perspective offers a conceptually different interpretation of the substructures within neural networks. Using this perspective, neural networks can be understood as an *ensemble of many internal subnetworks* whose outputs are aggregated to determine a final prediction. In this section, we discuss several formulations of implicit-ensemble-like ideas.

Ensemble-like behaviour in residual networks: Veit et al. [27] examine residual networks (ResNets) and show that they behave like an ensemble of many shallow subnetworks due to their skip connections—i.e., they implicitly act like ensembles. Each path within the network acts like a separate (relatively shallow) subnetwork, and the final prediction of the network is an aggregation of the outputs of these subnetworks.

Implicit boosting and self-averaging of neural networks: Sun et al. [71] explore a theoretical comparison of how a neural network trained with Dropout [18] functions in a similar ensemble-like manner to AdaBoost [44] (often referred to simply as "boosting"). They argue that neural networks trained with Dropout behave like *self-averaging ensembles* of many different "sub-classifiers". As mentioned by Srivastava et al. [18], the network's forward passes effectively correspond to sampling from different subnetworks, and Sun et al. [71] argue that training encourages these subnetworks to cooperate in a boosting-like manner.

Collaborating feature differentiators: Theunnisen et al. [72] study how neural networks can interpolate noisy training data without it negatively impacting the generalisation performance of the network. They argue that neural networks are structured as collaborative substructures, where hidden nodes act as individual "feature differentiators" trained only on the subset of the data for which it is active. By adopting this viewpoint, they argue that each layer then acts as a mixture model that fits samples based on their

representations in the previous layer. These specialised substructures (nodes) behave like an implicit ensemble—each substructure is focused on different parts of the data, but they cooperate in a layer-wise manner to produce a combined output.

Layer-wise ensembles of hidden nodes: Davel et al. [1] propose interpreting neural networks as layered collections of cooperating classifiers, where each hidden node estimates class probabilities based on discrete and continuous activation behaviours. They show that deeper nodes tend to become highly class-selective during training and that entire layers can be viewed as *implicit ensembles* of cooperating nodes in that layer.

Discussion: These papers support the idea that neural networks could be viewed as implicit ensembles of different subnetworks or components. Although the formulation for these perspectives differ, each perspective reinforces the notion that neural networks rely on several subcomponents cooperating in an ensemble-like fashion to produce better model generalisation. This implicit ensemble interpretation provides another viewpoint on how a neural network can be decomposed into substructures that could aid interpretability and can potentially lead to increased understanding of their inner workings.

3.3 Implicit ensemble construction

In this section, we clarify how we view a neural network as an implicit ensemble. While several approaches exist for decomposing neural networks into substructures that can potentially be interpreted as subpredictors of an ensemble (see Section 3.2), we adopt a simple and direct approach from the implicit ensemble literature.

For our purposes, we use the method from Davel et al. [1]. We define their approach in the next section, but justify our choice here:

- **Simplicity:** The method requires minimal additional assumptions or approximations to define what constitutes an ensemble member within a neural network.

- **Conceptual alignment:** This approach characterises ensemble members in a similar manner to how explicit ensembles are typically conceptualised—i.e., as homogeneous classifiers that receive the same input and whose outputs are combined through a common aggregation rule.
- **Conservativeness:** Given that few studies have directly investigated the ensemble-like nature within neural networks, we thought it prudent to select this simple approach as opposed to more complex alternatives that may introduce unnecessary confounding factors. We leave extensions to more elaborate approaches for future work.
- **Familiarity:** This method was developed within our research group, providing access to established tools, frameworks, and implementation insights.

A complete formal definition of the implicit ensemble construction is provided in the subsection that follows.

3.3.1 Node as a predictor

Why use node-based subpredictors? We adopt the approach from Davel et al. [1], which frames each hidden node within the neural network as an individual classifier. This method offers a fine-grained perspective on the network's underlying subcomponents. Unlike approaches that decompose the model into larger architectural subcomponents (e.g., modules or clusters, as discussed in Section 3.2), a node-level decomposition allows an analysis of the network using one of its most fundamental computational elements.

While a single node is unlikely to constitute a very powerful classifier, a core advantage of decomposing the model at the node level is the resulting increase in *ensemble size*. By treating every hidden node as a subpredictor, we significantly increase the total number of "ensemble members" within the network when viewed as an implicit ensemble. A larger ensemble enhances the *potential* for diversity among the subpredictors, which is a key factor for improving generalisation in traditional ensemble learning [30], [39], [48].

Another advantage of using this method, is that it can be applied across various neural network architectures, including both MLPs and CNNs, provided the network contains standard fully-connected layers (as described in Section 2.2.1) with hidden node activations as defined in Equation (2.1).

Challenges in viewing nodes as subpredictors: Despite these advantages, it is important to acknowledge the conceptual challenges of this interpretation. A hidden node does not—by definition—output class predictions, but rather some representation of the input. Its function is to perform a specific non-linear transformation on the weighted inputs it receives, thereby computing one component of the internal representation of the input $\mathbf{x}^{(i)}$ at some hidden layer. Any "prediction" derived from an individual node must therefore be constructed indirectly—such as mapping the node's activation pattern to class labels using a simple estimator or local classifier, as proposed by Davel et al. [1]. Moreover, hidden nodes exhibit a high degree of interdependence—i.e., their activations are determined by shared connections with other preceding and successive nodes, meaning that an individual node's contributions is meaningful in combination with the activations of other nodes [1], [18], [73], [74].

A further (and important) caveat is that modern neural networks often learn distributed representations in which individual nodes do not need to correspond to disentangled or specialised factors of variation. Disentanglement is typically not identifiable without strong inductive biases on the model and/or data [75]. Related work on modularity and systematic generalisation also suggests that specialisation is not a default outcome of standard training, and often requires architectural constraints or specific data properties [76]–[79].

Operational meaning of "node-as-subpredictor": In this study, the term *node-as-subpredictor* is therefore used in an *operational* sense rather than implying semantic monosemanticity or true modular independence. Specifically, we define a node-level subpredictor by probing for a class decision based on the activation value of a node. Under this viewpoint, analysing node accuracies and inter-node agreement does not assume that

neurons are disentangled "experts"; instead, it quantifies how class information and redundancy are distributed across nodes in the layer [73], [74].

Constructing node-level subpredictors: Taking note of the fact that individual nodes do not natively produce class predictions, we define a method of constructing the node-based classifiers from their pre-activations using the framework proposed by Davel et al. [1]. Given a hidden node, our aim is therefore to construct a classifier associated with that node.

Let $h^n_{(\ell)}$ denote the activation of node n in layer ℓ, as defined in Equation (2.1), for an input sample $\mathbf{x}^{(i)}$ to an MLP. Using the same notation, we define the pre-activation value for this node as $z^n_{(\ell)}(\mathbf{x}^{(i)}) = b^n_{(\ell)} + \sum_{p=1}^{d_{\ell-1}} W^{p,n}_{(\ell-1),(l)} h^p_{(\ell-1)}$, so that $h^n_{(\ell)} = \sigma\left(z^n_{(\ell)}(\mathbf{x}^{(i)})\right)$.

Following Davel et al. [1], we construct a continuous probability estimate $P\left(c \mid z^n_{(\ell)}(\mathbf{x}^{(i)})\right)$ for a class c given a pre-activation $z^n_{(\ell)}(\mathbf{x}^{(i)})$. To do this, we first estimate $P\left(z^n_{(\ell)}(\mathbf{x}^{(i)}) \mid c\right)$. We model this class-conditional distribution of pre-activations $z^n_{(\ell)}(\mathbf{x}^{(i)})$ over the training set using a Gaussian Kernel density estimator (KDE)[4]. This therefore yields a continuous estimate of the likelihood of observing a certain pre-activation value at node n when the true class is c.

In order to get $P(c \mid z^n_{(\ell)}(\mathbf{x}^{(i)}))$ we combine $P(z^n_{(\ell)}(\mathbf{x}^{(i)}) \mid c)$ with a class prior[5] $P(c)$ using Bayes' rule:

$$P(c \mid z^n_{(\ell)}) = \frac{P(z^n_{(\ell)} \mid c)\, P(c)}{\sum_{c'=1}^{C} P(z^n_{(\ell)} \mid c')\, P(c')}. \tag{3.1}$$

$P\big(c \mid z^n_{(\ell)}(\mathbf{x}^{(i)})\big)$ is then node n in layer ℓ's predicted probability for class c when given input $\mathbf{x}^{(i)}$. The node's class prediction is then simply computed as

$$\hat{y}^n_{(\ell)}(\mathbf{x}^{(i)}) = \arg\max_c \; P\big(c \mid z^n_{(\ell)}(\mathbf{x}^{(i)})\big),$$

which we refer to as a *hard vote* because the node outputs only the single most likely class, rather than its full probability distribution.

[4] In practice we do this by gathering $z^n_{(\ell)}(\mathbf{x}^{(i)})$ for all training samples where $y^{(i)} = c$ and fitting an out-of-the box KDE to it.

[5] Class priors are taken as the empirical class frequencies in the training data.

This hard-vote step is also the key difference in our subpredictor construction versus the formulation by Davel et al. [1]; in their work, the continuous class-probability estimates are combined at the layer level using a soft aggregation method (multiplying class-probabilities across nodes in a layer).

It is important to note that node-classifiers (i.e., KDEs) are "fit" using the training set and then only *used* on the test set—we do not fit the nodes to the test set.

We extend this analysis to CNNs. In this case, node-level predictions are computed *only* for the *fully-connected layers*. Considering subpredictors within a convolutional layer is beyond the scope of this study and will be reserved as a task for future work.

3.3.2 Ensembling nodes with a majority vote

For each hidden layer ℓ, we treat the collection of node-level subpredictors—constructed as described in Section 3.3.1—as an implicit ensemble. Each node n provides a hard class prediction $\hat{y}^n_{(\ell)}(\mathbf{x}^{(i)})$ for an input $\mathbf{x}^{(i)}$, and the layer's overall prediction is obtained by aggregating from these subpredictors. To ensemble the outputs of all nodes in a layer ℓ, we employ a simple *majority-vote rule*: the layer-level prediction for sample $\mathbf{x}^{(i)}$ is the class receiving the most votes among its constituent nodes. This majority vote directly defines the layer-wise prediction which can then be used to determine the layer accuracy.

We adopt a *hard majority vote* because the diversity measures used in this study (e.g., *Disagreement*, *Double-fault*, and related contingency table metrics) are defined on *discrete class decisions*. Concretely, they require the per-node predicted label $\hat{y}^n_{(\ell)}(\mathbf{x}^{(i)}) \in 1, \ldots, C$ for each sample.

An alternative would be to compute a soft layer-level prediction by first aggregating each node's per-class scores (or estimated class-posterior probabilities) across the layer—e.g., by summing log-probabilities over nodes as in Davel et al. [1]—and then taking an arg max to obtain the layer's predicted class. While this soft aggregation can be used to compute accuracy, it would introduce an inconsistency: diversity would still be computed from hard

node predictions, whereas performance would be computed from a score-based combiner.

By using majority vote for the layer prediction, we ensure that both (i) the diversity measures and (ii) the layer-wise accuracy are derived from the same underlying *discrete node predictions*, so the diversity–performance relationship is evaluated on a consistent basis. Additionally, majority voting yields an interpretable estimate of "how many subpredictors support a class," aligning with our implicit ensemble viewpoint.

We do note that the results could be dependent on this choice and that future work will investigate this further.

3.3.3 Differences between explicit and implicit ensembles

Interpreting a neural network as an implicit ensemble provides a useful analytical lens, but it is important to note that this construction is fundamentally different from explicit ensembles used in classical ensemble learning.

Some differences include:

- **Different training procedure:** Explicit ensembles typically train their members separately, whereas neural networks are trained end-to-end using gradient descent [7, pp. 83–88] and backpropagation [80], which simultaneously updates all parameters in the network.
- **Different training data:** In explicit ensembles, members often see re-sampled or re-weighted versions of the training data (e.g., bagging, boosting). In contrast, nodes within the same layer of a neural network are all trained on the same training data at each iteration of training (this is assuming a fully connected layer configuration). However, due to non-linear activations (such as ReLU [7, p. 25]), individual nodes may effectively "ignore" certain samples; if a node remains inactive for a specific input, it receives no gradient update for that sample.
- **Layer-dependent data input:** Deeper layers of neural networks operate on trans-

formed representations of the inputs of earlier layers, meaning that during training, each layer effectively sees entirely different input compared to the input layer.

- **Differences during early training:** Unlike explicit ensembles whose members are trained on meaningful data from the start, subpredictors in the deeper layer of a neural network may initially learn from low-quality data from preceding layers.

Classical results in ensemble theory provide useful intuition but rely on assumptions that are not automatically satisfied in our implicit ensemble viewpoint. For example, Condorcet-style jury theorems assume that individual voters have better than random guessing accuracy under majority voting; under these conditions the probability of a correct majority vote increases with ensemble size (see also Section 2.3.1) [40]. In contrast, node-level subpredictors within a trained network are strongly dependent due to shared parameters and shared representations, and their individual competence varies and can be near chance. As such, we do not treat such theorems as explicit guarantees for implicit ensembles. Instead, we borrow diversity measures from the explicit-ensemble literature as empirical descriptors of agreement/disagreement or shared failures computed on a common output space, and we interpret them together with node- and layer-level accuracy to evaluate how information and redundancy are distributed across the network.

Assumptions regarding classical diversity measures: Most classical diversity measures (such as those described in Section 2.4.2) assume that ensemble members solve the same prediction task and produce decisions in a common output space, so that their predictions can be compared on the same input samples [28]. They also assume that predictions are expressed as discrete class decisions (most measures assume binary correct/incorrect input) allowing classifier behaviour to be summarised through contingency tables that count agreements, disagreements, and joint failures (as with the pairwise measures in Section 2.4.2). In addition, these measures implicitly assume that ensemble members are not identical predictors and possess at least some level of predictive competence, since diversity among purely random predictors would not provide useful information about ensemble performance [28], [42]. Importantly, however, these measures

do not require strict independence between classifiers; rather, they quantify the degree to which classifier errors are correlated. In the implicit ensemble setting considered in this study, the first two assumptions hold by construction because each node-level subpredictor is mapped to the same class labels and produces a discrete decision. The remaining assumptions are only partially satisfied: node predictors may vary substantially in competence and are strongly dependent due to shared parameters. Consequently, the diversity measures used in this study should be interpreted as identifying patterns of agreement and error correlation between node-level predictors, rather than as indicators of independence between separately trained ensemble members.

Assumptions for treating node-level subpredictors as an ensemble: The node-as-subpredictor framework does not assume that nodes within a layer are redundant predictors of the same function. In practice, different nodes may respond differently to distinct aspects of the input. As a result, nodes form a distributed representation rather than a collection of identical predictors.

The ensemble perspective becomes applicable only after constructing the node-level predictors that map each node's activation to a prediction in a shared output space. Once this mapping is defined, every node-level subpredictor produces a discrete class prediction for the same classification task, which allows ensemble-style measures of agreement, disagreement, and joint failure to be computed across nodes. In this formulation, the ensemble is formed by the node-level subpredictors derived from node activations, rather than by the nodes themselves acting as redundant classifiers.

Conclusion: These differences and assumptions indicate that, although the implicit ensemble perspective may allow us to analyse a neural network in terms of subpredictors, the resulting "ensemble" differs structurally and procedurally from an explicit ensemble. The comparison remains useful as a conceptual framework, but the mechanisms around learning and variability among subpredictors are fundamentally different.

3.4 Experimental verification

Using the construction described in Section 3.3.1, we empirically investigate whether individual nodes can act as feasible subpredictors within the implicit ensemble framework. We regard a node as a feasible subpredictor if its classification accuracy exceeds random-chance performance. To test this, we conduct a preliminary experiment in which we evaluate the accuracy of each node's predictions across all hidden layers (Section 3.4.1).

In addition to this preliminary verification, we also replicate the experiments of Davel et al. [1] to ensure that our implementation of their node-based prediction mechanism behaves as expected. The reproduced results are provided in Appendix A.1.

3.4.1 Node-level accuracy

Experimental setup: To evaluate whether individual nodes can serve as meaningful subpredictors, we examine the per-node classification accuracy across all hidden layers of a three-hidden-layer network, each containing 512 nodes. We train the network to interpolation—i.e., 100% training accuracy or a near-zero training loss—using the hyper-parameters settings shown in Table 3.1. We apply the node-classifier construction method outlined in Section 3.3.1 to the training data, before computing the accuracy of each node on a stratified subset of the MNIST test set comprising 5 000 samples, stratified by class to preserve the original class distributions. The justification for this 5 000 sample size choice, along with the corresponding sensitivity analysis, is provided in Appendix A.2.

Results: The resulting per-node accuracies are presented in Figure 3.1. Across all three layers, we observe that most nodes consistently achieve accuracies above random guessing—which, for MNIST, would be 10%. Specifically, the majority of per-node accuracies fall between 15% and 20%. Given the substantial number of nodes that outperform random chance, it is reasonable to interpret these nodes as viable subpredictors.

Table 3.1: Summary of hyperparameters for the MLP architecture.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	0.0003
Batch size	256
Epochs (max)	1 000
Dropout/DropConnect	None
BatchNorm	None
Learning rate scheduler	{Step size = 1, gamma = 0.99}

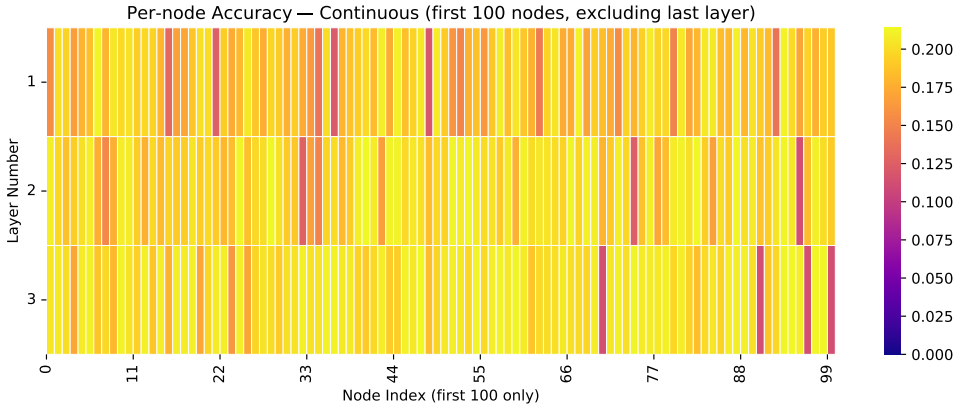


Figure 3.1: Accuracy heatmap across network layers. The heatmap displays node-wise accuracy per layer, with layers on the y-axis and nodes on the x-axis. Brighter regions indicate higher accuracy. Results were obtained from the MNIST test data set. The heatmap was obtained from a single 3-layer network visualising the first 100 nodes per layer.

3.4.2 Discussion

This verification step establishes that the node-level predictors constructed in Section 3.3.1 possess sufficient predictive capacity to justify exploring them as subpredictors within the implicit ensemble viewpoint. Although individual nodes are weak classifiers, many achieve accuracies meaningfully above chance, indicating they capture non-trivial structure in the data. Furthermore, the successful replication of Davel et al.'s [1] results (Appendix A.1) provides confidence that our implementation aligns with prior work, ensuring that subsequent analyses build on a reliable methodological foundation.

3.5 Summary

In this chapter, we explored several perspectives that all point toward an underlying theme: neural networks contain meaningful internal structure that can be analysed in terms of smaller functional components. Research on modularity shows that networks can, in principle, contain specialised sub-networks, even if standard architectures do not naturally form strongly connected modular decompositions. The literature on circuits demonstrates that groups of neurons collaborate to implement features or sub-tasks, revealing functional pathways embedded within the larger model. Clustering further shows that trained networks often organise their nodes into groups with strong internal connectivity, suggesting the emergence of semi-independent substructures.

We next introduced the implicit ensemble interpretation we find in Davel et al. [1] and Veit et al. [27]. Building on Davel et al.'s [1] interpretation, in which each hidden node is treated as a classifier contributing to a layer-level ensemble, we detailed how to construct node-level subpredictors using activation-based density estimation and how to aggregate their predictions via majority voting. While this ensemble interpretation could conceptually be useful, we also mention its key differences from explicit ensembles.

Finally, we conducted a preliminary empirical verification to gauge whether individual nodes possess sufficient predictive ability to be considered feasible subpredictors. Our experiment confirms that many nodes achieve accuracies above random chance, and our replication of Davel et al.'s [1] results validates our implementation. These steps lay the groundwork for subsequent chapters, where we study diversity, accuracy, and their relationship to each other under varying training conditions.

Chapter 4

Experimental setup

We provide an overview of the datasets, protocols, and techniques used for model setup and training.

4.1 Overview

This chapter outlines the experimental setup used throughout this study to train and evaluate neural networks under multiple controlled conditions. We begin by introducing the datasets used in our experiments, including their structure and their pre-processing steps (Section 4.2). We then give a high-level description of the model architecture used across the different experiments (Section 4.3.1), followed by a detail of the training protocol followed for all models (Section 4.3.2). Finally, we describe how diversity is measured within these trained models (Section 4.3.3), providing the computational procedures used for calculating all diversity metrics mentioned in Section 2.4.2.

4.2 Datasets

We create a controlled environment for experimentation by using well-known, and commonly used image-classification datasets, namely: Modified National Institute of Standards and Technology (MNIST) [10], Kuzushiji-MNIST (K-MNIST) [32], Fashion-MNIST [33], and Canadian Institute for Advanced Research (CIFAR-10) [34]. These datasets are widely used as benchmark datasets due to the good quality data that is publicly available.

4.2.1 MNIST

The MNIST [10][6] dataset is a standard benchmark in computer vision and deep learning, with a set of 70 000 grayscale images of digits (0-9) handwritten by high school students and employees of the US Census Bureau. The data is split into a 60 000 sample training set, and a testing set of 10 000 samples. Each image is 28x28 pixels, with a single channel. Pixel values are in the range $[0, 255]$, but we normalise them with *zero-one scaling* which scales the input features to $[0, 1]$.

MNIST is a simple but effective dataset for testing and analysing novel techniques or methods. Characterised by its low classification complexity, the MNIST dataset is particularly well-suited for initial prototyping and rapid experimental tests when computational resources are limited (as mentioned by Xiao et al. [33]). Figure 4.1 illustrates two samples of different classes that can be found in the dataset.

4.2.2 K-MNIST

The K-MNIST [32] dataset is a drop-in replacement for the MNIST dataset, also consisting of 70 000 28x28 grayscale images, provided in the same format as the original MNIST dataset. This dataset was designed to be a more challenging version of MNIST, and was

[6]The MNIST dataset is often called the "Hello World" of machine learning due to how much it has been studied.

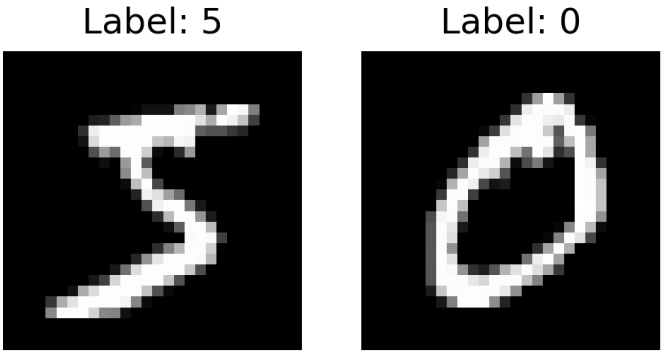


Figure 4.1: Sample images of the MNIST dataset, illustrating a sample from class five and class zero respectively.

introduced in 2018 by Clanuwat et al. [32] as part of the larger Kuzushiji dataset project.

Instead of digits, the classes represent 10 categories of Hiragana characters in the cursive Kuzushiji script. In modern Hiragana, the classes, in order, are as follows: 'o', 'ki', 'su', 'tsu', 'na', 'ha', 'ma', 'ya', 're', and 'o' (these 'o' classes are used in different contexts in Japanese texts).

K-MNIST maintains the same structure and size as MNIST but is supposed to be more challenging to classify. While the characters exhibit more intricate strokes, this structural complexity primarily affects models such as CNNs. For fully-connected architectures such as the MLPs used in this study, a more relevant factor is the larger number of active pixels and the greater overlap between classes (intra-class similarity) [32]. This dataset should prove useful for evaluating whether methods that perform well on MNIST transfer to more difficult recognition tasks. See Figure 4.2 for samples from two different classes in the dataset.

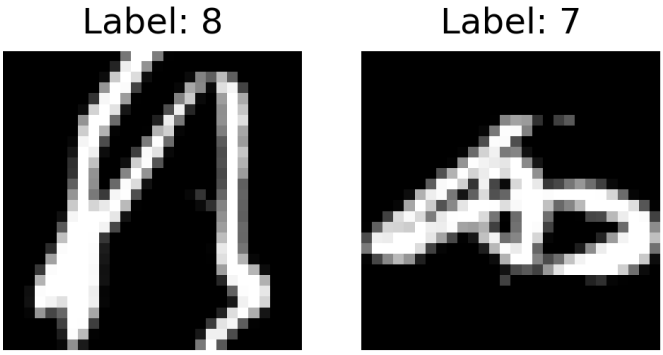


Figure 4.2: Sample images of the K-MNIST dataset. Label 8 refers to the 're' character class, and Label 7 refers to the 'ya' character class. These examples clearly showcase the added complexity compared to the MNIST dataset.

4.2.3 Fashion-MNIST

Fashion-MNIST is another benchmark dataset that was introduced as a drop-in replacement for MNIST by Xiao et al. [33] in 2017. It shares the same image size and structure, with the same training and test splits as the original MNIST dataset, however, the images represent classes of fashion items instead of digits with the following classes in order: 'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot' (see Figure 4.3 for example images).

Again, this dataset provides increased complexity compared to MNIST, as the clothing items are less distinct than digits [33]. Fashion-MNIST is also widely used in deep learning research to test whether results on MNIST generalise to more realistic visual data.

4.2.4 CIFAR-10

Introduced by Krizhevsky and Hinton [34] in 2009, the CIFAR-10 dataset is another widely used image classification benchmark dataset. CIFAR-10 consists of 60 000 colour images, each 32x32 pixels in size with 3 RGB channels. The 60 000 images are split into a 50 000

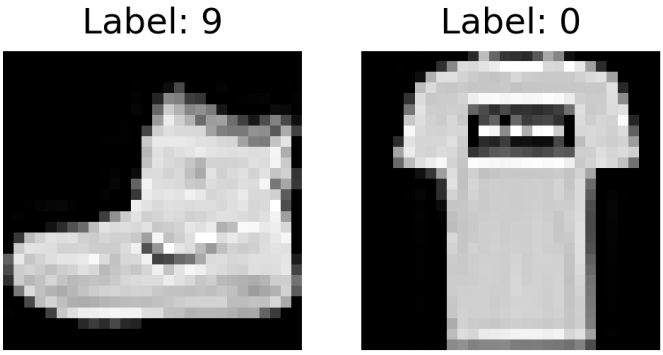


Figure 4.3: Sample images of the Fashion-MNIST dataset. We see an example of the first and the last classes (class 0 and class 9), with class 9 being an ankle boot, and class 0 being a T-shirt.

training set and 10 000 testing set. It also consists of 10 different object classes, namely: 'Airplane', 'Automobile', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog', 'Horse', 'Ship', 'Truck'.

CIFAR-10 is a much more complex dataset compared to the different MNIST variants, and requires more computational resources due to higher dimensionality and added colour channels [34].

This dataset serves as a more realistic and complex dataset that will be useful for testing whether results scale to higher-dimensional, more natural images.



Figure 4.4: Sample images of the CIFAR-10 dataset showing class 'frog' and class 'truck'.

4.2.5 Data pre-processing

Before training, all datasets undergo pre-processing to ensure consistent input scaling and dataset partitioning. For all experiments, the input features are normalised using *zero-one scaling*, where pixel intensity values are rescaled to the range $[0, 1]$. The MNIST, K-MNIST, and Fashion-MNIST datasets are initially divided into 50 000 training, 10 000 validation, and 10 000 test samples. A stratified subset of 2 000 samples are then taken from the 50 000 sample training set and used for training (see Section 4.3.2 for more details). CIFAR-10 is split into 40 000 training, 10 000 validation, and 10 000 test samples. A separate training set is also created for CIFAR-10 as a stratified subset with 8 000 samples.

4.3 Experimental procedure

A strict experimental procedure is followed throughout this study to ensure consistency and reproducibility. All experiments are performed using at least one of the datasets described in Section 4.2. All models are trained using the training protocol mentioned in Section 4.3.2.

4.3.1 Model architectures

MLP model setup: With the datasets having varying degrees of complexity, we employ different model architectures to ensure relatively good performance across settings. We use fully-connected MLPs for the MNIST, K-MNIST, and Fashion-MNIST datasets. We systematically vary the configurations of architectural depth and width to align with the specific goals of each experiment. The precise architectural parameters used for the network in each setting will be detailed in the corresponding section.

CNN model setup: For models trained on CIFAR-10, we employ CNNs consisting of multiple convolutional and pooling layers followed by fully-connected layers (see Table 4.1). These networks are structured in a VGG16-like [11] fashion, with minor adjustments to the input layer dimensions and sizes of the fully-connected layers.

Table 4.1: CNN specification (Convolutional bias=True, activation=ReLU, BatchNorm=False). "$\times n$" indicates repeated blocks.

Stage	Layers	Kernel / Stride / Pad	Output size (for 32×32 input)
Input			32×32×3
S1	Conv 64 ×2	$(3, 1, 1)$	32×32×64
	MaxPool	$(2, 2, 0)$	16×16×64
S2	Conv 128 ×2	$(3, 1, 1)$	16×16×128
	MaxPool	$(2, 2, 0)$	8×8×128
S3	Conv 256 ×3	$(3, 1, 1)$	8×8×256
	MaxPool	$(2, 2, 0)$	4×4×256
S4	Conv 512 ×3	$(3, 1, 1)$	4×4×512
	MaxPool	$(2, 2, 0)$	2×2×512
S5	Conv 512 ×3	$(3, 1, 1)$	2×2×512
	MaxPool	$(2, 2, 0)$	1×1×512
Head	Flatten → MLP(3×512) → Dense(C)		C

Dataset-specific architectural choices The choice of architecture per dataset was guided by both methodological and practical considerations. For the MNIST variants, MLPs are sufficient to achieve strong performance [10]. CIFAR-10 consists of higher resolution, colour images, for which convolutional architectures are known to be more appropriate [34]. Training MLPs on CIFAR-10 without extensive data augmentation typically results in poor generalisation performance, which would confound the diversity-generalisation analysis.

Additionally, this study requires running a large number of experiments across multiple seeds and intervention settings. To ensure that the experiments remain computationally feasible, architectures are selected that provide reliable baseline performance for each dataset without requiring extensive augmentation or hyperparameter tuning.

4.3.2 Model training protocol

All models are trained using the PyTorch [81] framework which is an open-source machine learning framework for building and training deep learning models.

To ensure a fair and consistent basis for our experiments, we adopt a controlled training protocol that can be used for both the simple and more complex datasets. This protocol is designed to highlight the effect of inducing diversity, while keeping the experimental setup reproducible and interpretable.

MLP training protocol: For MLP models trained on MNIST, Fashion-MNIST, and K-MNIST, we make use of a reduced training set rather than the full dataset. MLPs can fit these datasets with relative ease—usually reaching test accuracies above 98% on MNIST—when using the full training set, which results in near-saturated performance with little variation across models. To counteract the little variation, and to better illustrate the impact of diversity-inducing methods, we instead sample a small stratified subset of 2 000 images from the training set. Due to stratified sampling, we preserve the original class distribution while significantly reducing the training data available to the model—significantly reducing the time needed for training. We continue training the models until they reach interpolation (i.e., 100% training accuracy). Using this training setup, all models fit the subset essentially perfectly, and should exhibit varying degrees of generalisation on the test set, allowing us to investigate how diversity interacts with model performance.

CNN training protocol: When training models on CIFAR-10, we use CNNs with a structure as shown in Table 4.1. We follow a more conventional approach to dataset size, where the entire training set of 40 000 images are used due to the increased complexity of the CIFAR-10 dataset. Section 6.5 discusses the use of the 8 000 sub-sampled training set.

Loss function and stochasticity control: All models are trained using the cross-entropy loss function [7, pp. 71–72]. Batch normalisation is deliberately not used across all experiments to avoid possible effects it might have on the diversity and generalisation. Each experiment is repeated with five independent random seeds with different weight initialisations (default PyTorch uniform initialisation). Unless mentioned otherwise, results correspond to the mean and standard deviation[7] across these five seeded runs, ensuring that conclusions are made based on reproducible outcomes.

Hyperparameter configuration: Specific hyperparameter configurations are provided alongside corresponding experiments. We emphasize that *no formal hyperparameter search was conducted*. The primary objective of this study was not to maximise test accuracy but rather to understand the trends we see, and establish a consistent and controlled experimental setting in which we can investigate the effects of inducing diversity within single neural networks. Consequently, the chosen hyperparameters reflect commonly used default values from established literature that are known to produce stable training across MLPs and CNNs. This approach ensures that any observed differences in performance can be attributed primarily to manipulations made during experimentation—rather than to extensive hyperparameter tuning.

4.3.3 Diversity calculations

As mentioned in Section 2.4.2, to calculate the diversity of the subpredictors' predictions, we first construct *contingency tables* (as in Table 2.1) for each pair of nodes in a given layer ℓ. [8] To populate these tables, we make use of a binary accuracy matrix $\mathbf{B}$ for each layer.

For each node $n \in \mathcal{N}_{(\ell)}$, where $\mathcal{N}_{(\ell)}$ denotes the set of nodes within layer ℓ and input sample $\mathbf{x}^{(i)}$, we determine whether the node's predicted class $\hat{y}_\ell^n(\mathbf{x}^{(i)})$ matches the true

[7] The reported standard deviations were computed across the $i = 5$ runs using divisor i (rather than Bessel's correction with $i - 1$).

[8] The contingency tables are computed exclusively for node pairs within the same layer, as nodes in the same layer operate on a shared input representation (similar to classical ensembles).

class $y^{(i)}$. The binary accuracy matrix is defined as:

$$\mathbf{B}^n_{(\ell)}(\mathbf{x}^{(i)}) = \begin{cases} 1, \text{ if } \hat{y}^n_{(\ell)}(\mathbf{x}^{(i)}) = y^{(i)} \\ 0, \text{ otherwise} \end{cases} \tag{4.1}$$

Thus, $\mathbf{B}^n_{(\ell)}(\mathbf{x}^{(i)})$ indicates whether node n correctly classified input $\mathbf{x}^{(i)}$.

Using $\mathbf{B}_{(\ell)}$, we populate the contingency table for each pair of nodes in a layer ℓ as follows:

$$N^{11} = \sum_{i=1}^{N} \mathbf{1}(\mathbf{B}^{n_1}_{(\ell)}(\mathbf{x}^{(i)}) = 1 \wedge \mathbf{B}^{n_2}_{(\ell)}(\mathbf{x}^{(i)}) = 1) \tag{4.2}$$

$$N^{00} = \sum_{i=1}^{N} \mathbf{1}(\mathbf{B}^{n_1}_{(\ell)}(\mathbf{x}^{(i)}) = 0 \wedge \mathbf{B}^{n_2}_{(\ell)}(\mathbf{x}^{(i)}) = 0) \tag{4.3}$$

$$N^{10} = \sum_{i=1}^{N} \mathbf{1}(\mathbf{B}^{n_1}_{(\ell)}(\mathbf{x}^{(i)}) = 1 \wedge \mathbf{B}^{n_2}_{(\ell)}(\mathbf{x}^{(i)}) = 0) \tag{4.4}$$

$$N^{01} = \sum_{i=1}^{N} \mathbf{1}(\mathbf{B}^{n_1}_{(\ell)}(\mathbf{x}^{(i)}) = 0 \wedge \mathbf{B}^{n_2}_{(\ell)}(\mathbf{x}^{(i)}) = 1) \tag{4.5}$$

where $\mathbf{1}(\cdot)$ denotes the indicator function and N is the number of test samples.

Half of the diversity measures in this work are pairwise-based, meaning they are computed for each pair of nodes within a layer. Once the pairwise diversity values are obtained, we calculated the average diversity for the layer by averaging over all possible node pairs.

The pairwise diversity values form a two-dimensional $\mathcal{N} \times \mathcal{N}$ matrix that is symmetric by definition. Consequently, only one triangular half (upper or lower) needs to be considered when computing the average, as the other half contains identical duplicate entries.

4.4 Sensitivity analysis

When applying our constructed implicit ensemble view to the models used in this study, we evaluate each network on a stratified subset of 5 000 samples drawn from the corresponding test set. This subset size was determined through a sensitivity analysis conducted on the MNIST and Fashion-MNIST datasets (Section A.2), where we observed that diversity metrics stabilised around the 5 000 sample threshold. Based on this analysis, we assume that a similarly sized subset is sufficient to produce stable, representative results for the K-MNIST and CIFAR-10 datasets as well.

Using a reduced test subset allows the analysis to remain computationally feasible—an important consideration given the large number of models and node-level calculations involved—while still yielding results that closely match those obtained when using the full test set. This approach offers a practical balance between computational efficiency and result reliability.

4.5 Discussion

The experimental setup presented in this chapter establishes a controlled and reproducible foundation for analysing the relationship between diversity and generalisation in *single* neural networks. By combining well-characterised benchmark datasets with carefully constrained model architectures and a standardised training protocol, we aim to isolate the effects of induced diversity from confounding factors such as dataset scale, model capacity (when capacity is not the variable under study), and hyperparameter optimisation. The use of stratified sampling for MLP experiments enables meaningful variation in performance across seeds and experiments, while the larger CNN setting for CIFAR-10 provides a complementary, but higher complexity setting in which the same principles can be evaluated.

The procedures described for computing diversity—particularly the construction of node-level contingency matrices and layer-wise aggregation of pairwise measures—create a con-

sistent framework through which diversity can be quantified across architectures, datasets, and experimental conditions. This controlled framework ensures that differences observed in later chapters can be attributed to changes in model behaviour rather than to inconsistencies in the training process.

Chapter 5

Diversity in a single neural network

We analyse the behaviour of diversity in implicit deep ensembles, examining layer-wise diversity in trained networks, and exploring how it changes with network capacity and training progress

---

5.1 Overview

Following the development of the implicit ensemble perspective in Chapter 3, this chapter undertakes an *exploratory* analysis aimed at understanding how diversity manifests within a *single* neural network. The implicit ensemble picture of a neural network allows us to study the neural network using tools from ensemble theory. Specifically, with our notion of a subpredictor within a neural network, we can apply explicit ensemble diversity measures to quantify how diverse these internal components are.

In this chapter, we investigate the diversity that emerges within implicit ensembles formed by individual subpredictors (nodes) in a *single* neural network. Using the framework introduced in Chapter 3, we extend the analysis to quantify how diversity manifests across

layers and how it relates to layer-wise and overall model accuracy. We begin by presenting the diversity results obtained from a three-hidden-layer network, examining how various diversity measures behave across layers and datasets. Thereafter, we conduct several *exploratory* studies to deepen our understanding of how diversity operates within neural networks. First, we analyse how diversity changes with network capacity by varying the width of a single-hidden-layer network. Next, we measure diversity dynamically during training by analysing checkpoints saved during training—saved after a certain number of epochs—including an evaluation of diversity at model initialisation. These experiments should provide a comprehensive overview of how diversity develops, changes, and contributes to performance within a single neural network.

5.2 Layer-wise diversity in a three-layer network

We begin our analysis by addressing two key questions: (1) whether well-known ensemble diversity measures can be applied to the implicit ensemble case, and (2) if so, how does diversity manifest across layers in a fully connected network. This experiment provides a direct view of how the diversity among subpredictors changes with network depth and how it relates to layer-level performance.

5.2.1 Experimental setup

Model architectures: To analyse diversity within single neural networks in an implicit ensemble setting, we use a fully connected neural network with three hidden layers, each containing 512 nodes. As described in Section 4.3.1, MLP architectures were used for the MNIST, K-MNIST, and Fashion-MNIST datasets. For CIFAR-10, we use a CNN architecture (see Section 4.3.1 and Table 4.1 for details on convolutional and pooling setup), followed by a fully connected stage that mirrors the depth and width used in the MLP models. This ensures architectural consistency during node-level analysis.

Model training protocol, hyperparameter setup, and analysis: Using the above architectures, models are trained on their corresponding datasets following the protocol defined in Section 4.3.2. The specific hyperparameters used in the experiment include the use of the Adam optimiser with a learning rate of 0.0003, a batch size of 256, a maximum of 1 000 epochs, no Dropout, DropConnect, or Batch normalisation, and a step learning-rate scheduler with a step size of 1 and a decay factor (γ) of 0.99. These hyperparameters are summarised in Table 5.1.

Table 5.1: Summary of hyperparameters for MLP and CNN architectures. The baseline configuration uses a batch size of 256 with no explicit regularisation applied.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	0.0003
Batch size	256
Epochs (max)	1 000
Dropout/DropConnect	None
BatchNorm	None
Learning rate scheduler	{Step size = 1, gamma = 0.99}

After training, node-level predictions are extracted for a stratified subset of 5 000 test samples (as discussed in Section 4.4) using our construction of nodes as subpredictors (Section 3.3.1). These predictions form the basis for all diversity calculations following the procedures outlined in Section 4.3.3.

Diversity computation: For each hidden layer ℓ, we compute diversity using the full set of pairwise and non-pairwise diversity measures introduced in Section 2.4.2. These include *Disagreement*, *Double-fault*, *Q-statistic*, *Correlation coefficient*, *Kappa-statistic*, *Coincident failure diversity*, *Generalised diversity*, *Kohavi-Wolpert variance*, *Entropy*, and *Binary Entropy*.

For pairwise measures, diversity values are averaged across all unique node pairs within a layer. For each metric, we report both the mean and standard deviation across five independently seeded model initialisations. The mean and standard deviation of the

diversity values allow us to characterise how diversity evolves from shallow to deeper layers and how it varies across different datasets.

5.2.2 Diversity results

We now evaluate whether classical ensemble-based diversity measures provide meaningful insights for a *single* neural network that we view as an *implicit ensemble*. Specifically, we investigate whether the diversity among node-level predictions exhibits any noticeable trends across layers and whether these trends relate to layer accuracy. For this experiment, we consider only the accuracy of layers (recall that we ensemble the node predictions in a layer using majority voting—see Section 3.3.2).

Fashion-MNIST, MNIST, and K-MNIST

Relationship between diversity and layer performance: Figure 5.1 illustrates the behaviour of the *Disagreement* measure across the three hidden layers of a fully connected network trained on Fashion-MNIST. We observe a clear upward trend in both diversity and layer accuracy—the mean disagreement increases progressively with layer depth, closely mirroring the improvement in layer accuracy. The scatter (right) quantifies this relationship, showing a strong positive correlation between the level of disagreement between subpredictor predictions and accuracy across layers. This pattern suggests that deeper layers—while achieving higher layer accuracy—also have a greater degree of variability or independence among nodes.

Consistency across measures: The increasing trend with depth is not unique to the *Disagreement* metric. As shown in Table 5.2, several of the classical diversity measures display similar behaviour. *Binary Entropy*, *Coincident failure diversity* (*CFD*), *Generalised diversity* (*GD*), *Kohavi–Wolpert variance* (*KW-variance*), and *Entropy* all rise monotonically across layers, suggesting that node-level predictions become more diverse with network depth. The *Double-fault* (*DF*) measure exhibits the same pattern in an inverse

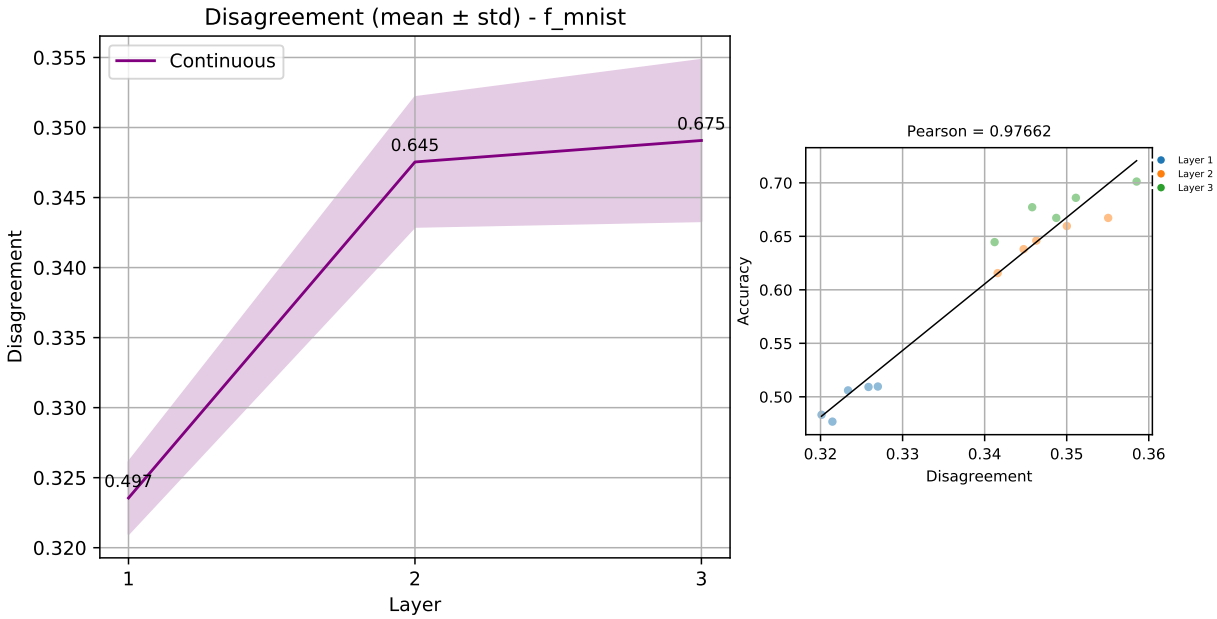


Figure 5.1: *(left) Disagreement* of node-level predictions across layers of a three-layer MLP trained on Fashion-MNIST, averaged over five seeds, and *(right)* the correlation between the disagreement and layer accuracy for the five seeds. The annotated values correspond to the averaged layer accuracies, which rise in tandem with *Disagreement*. The right panel confirms this relationship, showing a strong positive correlation (Pearson $r = 0.98$) between *Disagreement* and layer accuracy across seeds and layers.

form—its value decreases with depth, reflecting fewer simultaneous misclassifications—which in turn reflects an increase in diversity with depth.

The increase in diversity is strongly and positively correlated with layer accuracy for all these metrics—besides *DF* with an inverse, negative correlation—with absolute Pearson correlations exceeding $r = 0.95$ in every case (Table 5.2).

These results suggest a consistent relationship between diversity and layer performance: as layers progress, the node ensemble becomes both more accurate, and more diverse.

Divergent behaviour of correlation-based metrics: The second group of diversity measures—namely the *Correlation coefficient*, the *Kappa-statistic* and the *Q-statistic*—show a different trend. These measures (shown in Table 5.3 and Figure 5.2) show slight

Table 5.2: Per-layer diversity (mean ± std over seeds) and correlation with layer accuracy for Fashion-MNIST.

Measure	**Layer 1**	**Layer 2**	**Layer 3**	**Pearson r**
Bin Entropy	0.710 ± 0.004	0.752 ± 0.008	0.754 ± 0.011	0.970
CFD	0.296 ± 0.001	0.315 ± 0.003	0.322 ± 0.004	0.956
Disagreement	0.324 ± 0.003	0.348 ± 0.005	0.349 ± 0.006	0.977
DF	0.543 ± 0.001	0.512 ± 0.004	0.505 ± 0.005	-0.987
Entropy	2.622 ± 0.013	2.705 ± 0.015	2.706 ± 0.020	0.952
GD	0.230 ± 0.001	0.252 ± 0.002	0.257 ± 0.004	0.995
KW Variance	0.161 ± 0.001	0.173 ± 0.002	0.174 ± 0.003	0.977

decreases in magnitude initially, followed by an increase in magnitude in the deepest layer—there is no monotonic increase with layer depth. Importantly, these metrics show a strong *negative correlation* with layer accuracy (Pearson $r \approx -0.8$). This behaviour is due to how these measures quantify *agreement* between node predictions. Since agreement is conceptually the inverse of diversity, lower values imply that node predictions are more independent (and thus more diverse). Consequently, as the layer's predictive ability improves, node outputs become less correlated with each other—hence the observed negative relationship.

Table 5.3: Per-layer diversity (mean ± std over seeds) and correlation with layer accuracy for negatively correlated measures on Fashion-MNIST.

Measure	**Layer 1**	**Layer 2**	**Layer 3**	**Pearson r**
Correlation Coefficient	0.223 ± 0.007	0.196 ± 0.010	0.202 ± 0.014	-0.786
Kappa	0.222 ± 0.007	0.193 ± 0.010	0.199 ± 0.014	-0.799
Q-statistic	0.450 ± 0.012	0.395 ± 0.019	0.407 ± 0.025	-0.815

MNIST and K-MNIST: Similar trends were observed for the K-MNIST and the simpler MNIST datasets, both of which exhibited steadily increasing diversity and layer accuracy with layer depth, along with strong correlations between diversity and layer accuracy across nearly all measures (see Figure 5.5 and Table 5.4 for MNIST and K-MNIST diver-

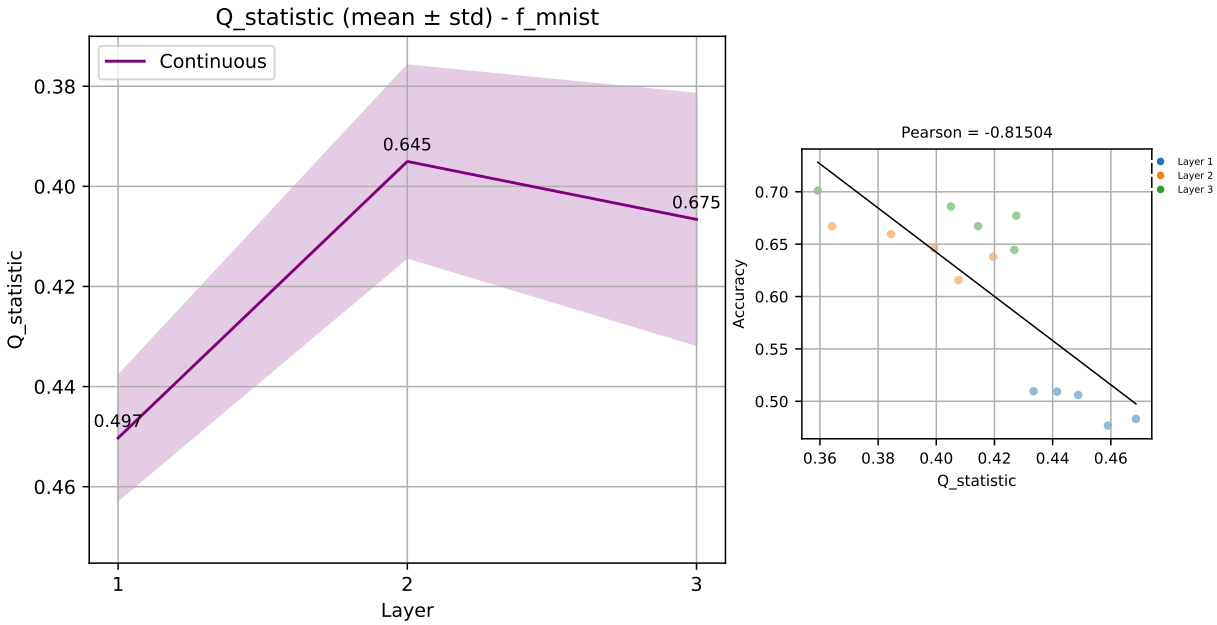


Figure 5.2: *(left) Q-statistic* of node-level predictions across layers of a three-layer MLP trained on Fashion-MNIST, averaged over five seeds, and *(right)* the correlation between the *Q-statistic* and layer accuracy. The annotated values correspond to the averaged layer accuracies. The right panel illustrates the relationship between *Q-statistic* and layer accuracy, showing a strong negative correlation (Pearson $r = -0.815$). Note that the y-axis has been inverted so that increases in diversity are reflected as upward trends in the plot.

sity trends and correlations). These additional results confirm the relationship between diversity and layer performance observed in Fashion-MNIST, reinforcing the robustness of the trends and patterns we see, prior to extending the analysis to CIFAR-10.

Interpretation: The results demonstrate that classical diversity measures—those originally designed for explicit classifier ensembles—remain interpretable when applied to a single neural network's node-wise prediction estimates. Measures consistently indicate that diversity is closely linked to the layer accuracy, and most follow the layer accuracy's trend of increasing with depth.

The divergence between measure groups underlines a possible limitation—not all classical metrics behave monotonically in the implicit setting, and the sign of their correlation with

layer performance depends on how "diversity" is defined mathematically. In later sections and chapters, we extend this analysis to architectures with differing capacity (Section 5.3) and regularised models (Chapter 6) to evaluate whether these relationships persist under varying training conditions.

CIFAR-10

Layer performance with a CNN: The performance pattern observed for the CNN's trained on CIFAR-10 deviates significantly from that of the MLP models. In the MLPs, layer accuracy consistently increases with depth, exhibiting a relatively clear distinction between the accuracy of different layers (as shown in the right-hand scatter plot of Figure 5.1). However, within the convolutional network, no clear monotonic relationship is observed between layer depth and layer performance (see the scatter plot in Figure 5.3, where some Layer 1 accuracies exceed those of Layer 3). This pronounced deviation from the established MLP trend is likely due to the fundamental architectural differences between the fully-connected networks and the convolutional networks

Relationship between diversity and layer performance: Figure 5.3 presents the averaged layer accuracies and corresponding layer diversity for the CNN trained on CIFAR-10. Although there is no consistent pattern between depth and layer accuracy—evidenced by the spread of predictions across layers, such as the wide variation in Layer 1—there remains a strong relationship between diversity and performance. Layers with lower accuracy tend to show lower diversity, while those with higher accuracy tend to be more diverse. This proportional tracking between the two metrics is further supported by the strong positive correlation ($r = 0.93$) across all layers and seeds. We conclude that even though layer accuracy does not systematically increase with depth in the CNN, the *correlation* between diversity and layer performance is still present. See Table 5.4 for results concerning other datasets.

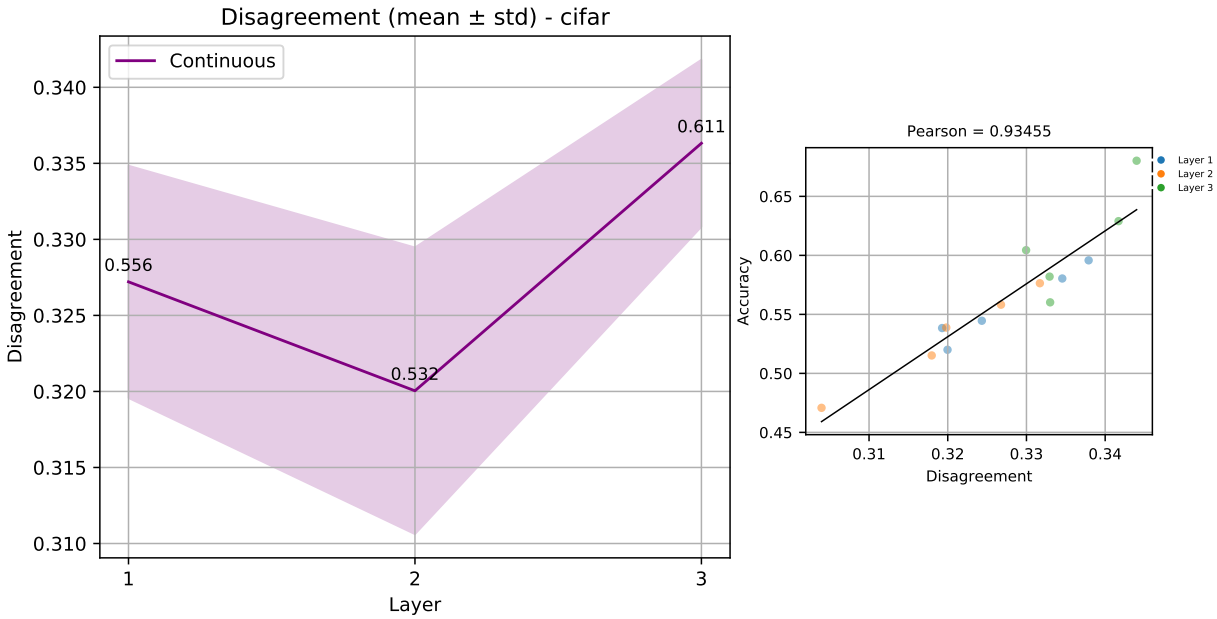


Figure 5.3: *(left) Disagreement* of node-level predictions across layers of a three-layer MLP stage of a CNN trained on CIFAR-10, averaged over five seeds, and *(right)* the correlation between the disagreement and layer accuracy. The annotated values correspond to the averaged layer accuracies. The right panel illustrates the relationship between disagreement and layer accuracy across seeds and layers with a strong Pearson correlation of $r = 0.93$.

Interpretation: The CIFAR-10 results reinforce one of the main findings: although the layer-wise performance differs from that of the simpler MLP architecture, the *relationship* between layer diversity and layer performance remains consistent. Even in the absence of a clear depth-accuracy trend, diversity continues to track layer performance closely.

The persistence of these relationships across both datasets and architectures implies that the various diversity measures are capturing related underlying trends. This motivates a closer examination of how these metrics relate to one another—specifically, which measures exhibit similar trends or redundancies—leading to the grouping analysis presented next.

5.2.3 Grouping diversity measures

When comparing the behaviour of the various diversity measures, we see some interesting parallels between certain metrics. Although each measure defines 'diversity' differently, many produce nearly identical trends when evaluated layer-wise and correlated with layer accuracy. These similarities suggest that the measures can be meaningfully grouped according to their underlying mathematical formulation and empirical behaviour.

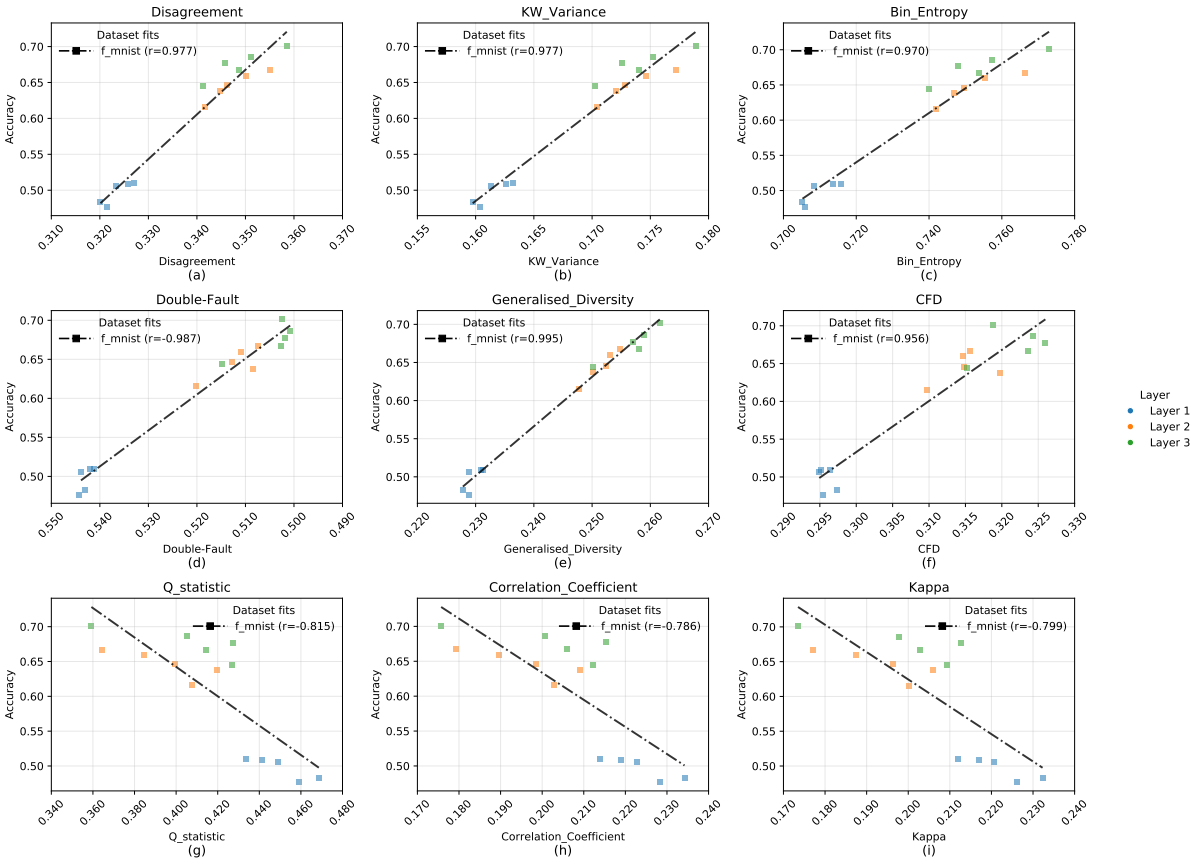


Figure 5.4: Scatter plots of layer accuracy vs. diversity across layers of a three-layer MLP trained on Fashion-MNIST for varying diversity metrics. The x-axis for the *DF* measure has been flipped for visual consistency with other metrics in the same group.

Grouping disagreement-based measures: The first group consists of *Disagreement*, *KW-variance*, and *Binary Entropy*, which exhibit almost indistinguishable patterns across all experiments (see Figures 5.4.(a-c) and 5.5.(a-c), and Table 5.4). Each of these metrics quantifies variability among node predictions by considering the proportion of pairwise

disagreements or output variance across the ensemble of subpredictors. This leads to them producing near-identical diversity curves and correlations with layer accuracy, typically showing a strong positive relationship. This is further supported by the similarity in the different measures scatter plots, with the only difference being scale or factor differences (Figure 5.4).

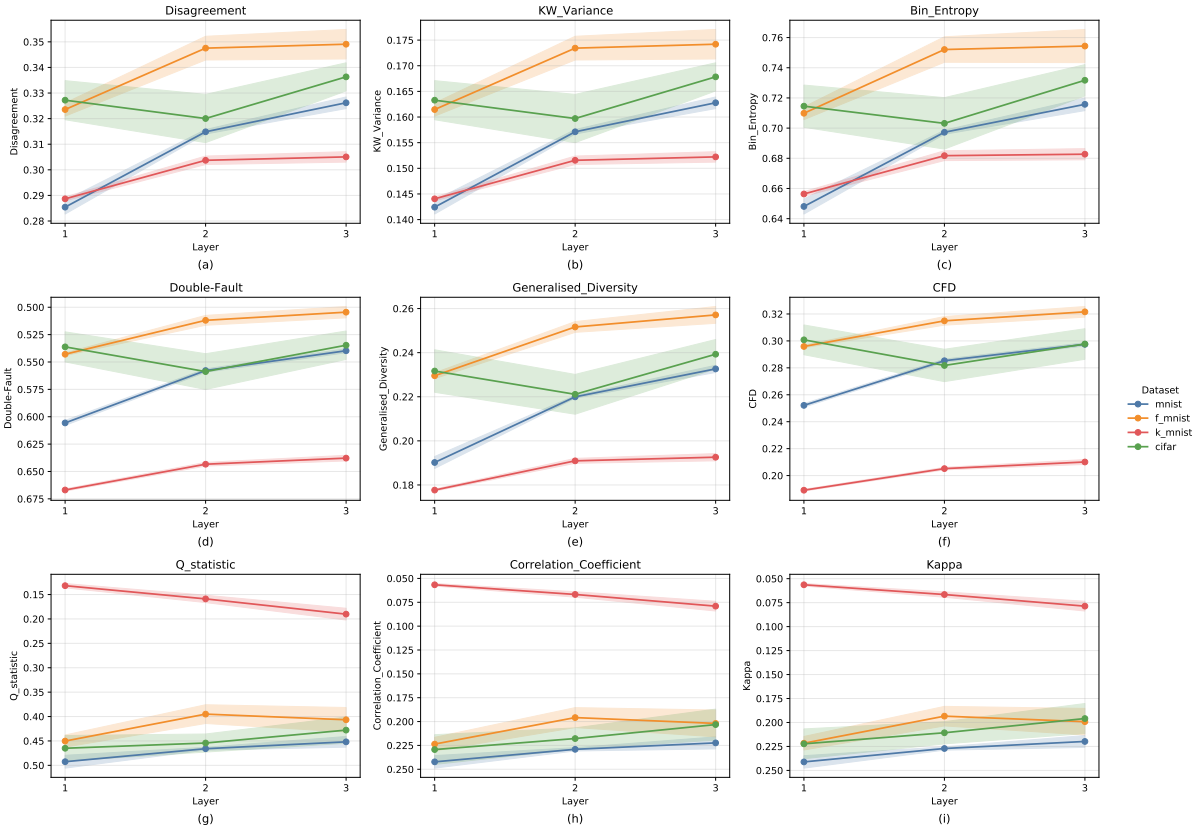


Figure 5.5: Layer-wise diversity values for all datasets (MNIST, K-MNIST, Fashion-MNIST, and CIFAR-10) computed using each of the diversity measures introduced in Section 2.4.2 besides *Entropy*. Each sub-plot illustrates how a specific metric changes across the three hidden layers.

Grouping joint-error-based measures: A second group, consisting of *DF*, *GD*, and *CFD* (Table 5.4), reflects a similar situation. These measures focus on the degree to which nodes fail on the same sample instead of overall disagreement. As such they have the same qualitative trends as the first group, with *DF* having inverted directionality—a higher *GD* and *CFD* reflect higher diversity, while with *DF*, a higher diversity is indicated

Table 5.4: Pearson correlation (r) between layer accuracy and diversity measures across datasets. Values represent the strength and direction of correlation for each metric, averaged over all layers and seeds.

Measure	MNIST	Fashion-MNIST	K-MNIST	CIFAR-10
Disagreement-based measures				
Bin Entropy	0.994	0.970	0.928	0.934
Disagreement	0.995	0.977	0.912	0.941
KW Variance	0.995	0.977	0.912	0.941
Joint error-based measures				
CFD	0.994	0.956	0.826	0.824
GD	0.994	0.995	0.894	0.866
DF	-0.996	-0.987	-0.856	-0.842
Agreement and correlation measures				
Correlation Coefficient	-0.860	-0.786	0.488	-0.638
Kappa	-0.877	-0.799	0.490	-0.648
Q-statistic	-0.879	-0.815	0.500	-0.672
Uncertainty/Entropy measure				
Entropy	0.969	0.952	0.029	0.757

through a lower *DF* value. The high consistency of similarity within this group indicates that all three metrics describe the same underlying aspect of diversity.

Grouping agreement/correlation-based measures: A third group is formed by the *Q-statistic*, *Correlation coefficient*, and Cohen's *Kappa statistic* (Table 5.4). These measures differ from the previous groups in that they explicitly quantify *agreement* rather than *disagreement*. Consequently, their correlation with accuracy are typically negative—layers with higher performance display lower inter-node agreement values. The three measures are tightly aligned with one another, both numerically and in trends, motivating putting them together in a family of correlation-based diversity measures.

***Entropy* as a distinct case:** Finally, *Entropy* (Table 5.4) forms a group of its own. Unlike other metrics, it is computed directly from the distribution of the predicted classes

rather than from pairwise node relationships. This formulation captures a different notion of diversity—it reflects the uncertainty or the spread of the prediction distribution at the layer level. This results in trends and correlation patterns that do not align with any of the other groups.

Representative measures: To avoid redundancy in subsequent analyses, only one representative measure from each group is used for further discussion (see Table 5.4 for all measure correlations): *Disagreement*, *DF*, *Q-statistic*, and *Entropy*. These four measures capture essential diversity dynamics observed across all metrics while minimising repetition. Throughout the remainder of this study, the term *diversity* will refer to the general behaviour reflected by these representative measures, unless otherwise specified.

Diversity metric inconsistencies: While most measures behave consistently across all four datasets, two groups exhibit notable exceptions. For the correlation-based measures (*Correlation coefficient*, *Q-statistic*, and *Kappa-statistic*), which typically show strong negative correlations with layer accuracy, the K-MNIST results deviate from this pattern—displaying positive correlations instead (see the K-MNIST column in Table 5.4). Similarly, *Entropy* exhibits a near-zero correlation with layer accuracy ($r = 0.029$) on K-MNIST, in contrast to the strong positive correlations seen in the other datasets.

5.2.4 Discussion

This section set out to evaluate whether classical diversity measures can provide meaningful insight when applied to the implicit ensemble viewpoint formed within hidden layers of neural networks. A three-layer fully connected network was trained independently on three datasets—MNIST, K-MNIST, and Fashion-MNIST—using five random seeds per configuration. A separate convolutional network was trained on the CIFAR-10 dataset. For each trained model, node-level predictions were extracted, and layer accuracies were compared with several classical diversity measures (introduced in Section 2.4.2). This allowed us to examine how diversity evolves across layers and whether it correlates with

layer performance in a consistent and interpretable way.

Across all datasets and model architectures, a clear relationship between layer diversity and layer performance was found: layers achieving higher accuracy also displayed higher diversity. For the MLP models, both layer accuracy and diversity increased monotonically with depth (with some exceptions among the diversity metrics), with strong correlations (typically $r > 0.90$) for most measures. Metrics based on agreement or correlation, such as the *Q-statistic*, *Kappa-statistic*, and *Correlation coefficient*, also showed relatively strong but negative correlations, reflecting that lower pairwise agreement among node outputs corresponds to higher effective diversity.

Extending the analysis to a convolutional architecture trained on CIFAR-10 revealed that the relationship between *depth* and *layer performance* no longer followed a clear monotonic trend. Here, the shallow layers achieved comparable or even higher accuracies than the intermediate or deep layers, possibly due to architectural differences between CNNs and MLPs. Nevertheless, diversity continued to match layer performance closely; layers with lower accuracy also had lower diversity, while higher-performing layers had higher diversity, still resulting in strong correlations (e.g., $r = 0.93$ for *Disagreement*).

When comparing results across all four datasets, some measures behave similarly, forming distinct groups based on the similarity of their trends and correlations. *Disagreement*, *Binary Entropy*, and *KW-variance* showed near-identical relationships with accuracy, while *DF*, *CFD*, and *GD* followed similar (but in *DF*'s case, inverted) patterns due to their similar joint-error-based formulation. Correlation-based metrics (*Q-statistic*, *Correlation coefficient*, and *Kappa-statistic*) formed another group characterized by their mostly negative correlation with accuracy, whereas *Entropy* stood apart as a distinct measure derived from uncertainty in multi-class predictions rather than inter-node disagreement.

Not all measures, however, behaved consistently across datasets. The K-MNIST dataset introduced some notable deviations: the correlation-based measures displayed *positive* rather than negative correlations with layer accuracy, and *Entropy* showed an almost negligible relationship ($r = 0.029$). These deviations show that while diversity measures

are generally robust, their performance might be dataset/measure dependent (some measures might be more consistent than others). A compelling direction for future research involves an investigation into the reason as to why we see divergent trends with some measures on certain datasets. Determining the specific factors that cause these measures to deviate from expected trends would enhance our understanding of diversity within single neural networks.

Overall, the analysis confirms that classical diversity measures, originally developed for explicit ensembles, can meaningfully characterise diversity in a *single neural network* that we view as an implicit ensemble. The observed inconsistencies in certain metrics illustrate the importance of measure selection. These insights motivate the grouping and reduction of redundant metrics in subsequent analyses, where representative measures—*Disagreement*, *DF*, *Q-statistic*, and *Entropy*—are used to further investigate diversity behaviour under varying training conditions.

5.3 Effect of network capacity on diversity

To better understand the concept of diversity within a *single* neural network, we continue our *exploratory* study by systematically investigating the effect that width-wise capacity has on diversity. For neural networks, subpredictors in deeper layers share weights—as they are all computed from the same set of activations in the preceding layer—which likely impacts the diversity of the subpredictors. By increasing the width, the number of shared weights increases as well. To study the effect of capacity on diversity without this complication, we are motivated to consider a single hidden layer neural network. Notably, the subpredictors in the hidden layer of such a network do not share weights. Adjusting the width directly changes the number of subpredictors that make up the implicit ensemble, thereby altering not only the effective "ensemble" size, but also the representational capacity—the ability to model complex functions and patterns in data—of the network. Increasing the number of nodes allows for a larger set of activation patterns. This larger set of activation patterns may have an impact on the diversity of

the neural network depending on the degree to which nodes specialise. By examining diversity across a range of widths, we can characterise how the structure and behaviour of these subpredictors evolve as model capacity increases.

5.3.1 Experimental setup

Model training protocol, hyperparameter setup, and analysis: Models are trained on the MNIST and Fashion-MNIST datasets following the protocol defined in Section 4.3.2. The hyperparameters used in this experiment are the same as those described in Section 5.2.1 and shown in Table 5.1.

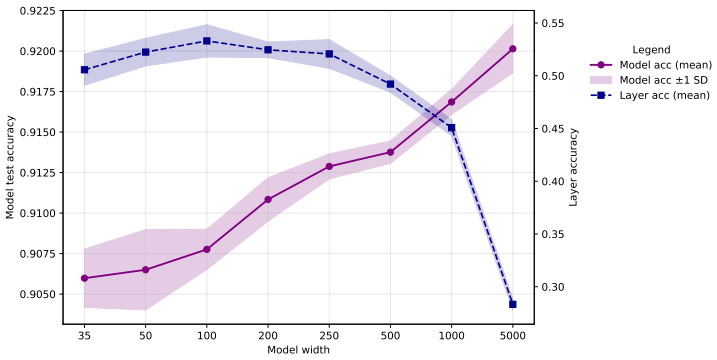
Model architectures: As mentioned, we will be investigating diversity within a single neural network using models with only a single hidden-layer. Before conducting the main analysis, a preliminary experiment was performed to identify the interpolation threshold for each dataset based on the training protocol and hyperparameter setup being employed. This threshold represents the minimum number of hidden nodes required for the model to perfectly fit the training data across different random seeds. For MNIST, interpolation was reached at approximately 35 nodes, while for Fashion-MNIST, the higher complexity of the dataset required a wider network, with interpolation occurring around 300 nodes. These thresholds were then set as the minimum widths for the following experiments, ensuring that every model considered in the analysis was trained on—and capable of learning from—the same training set, thereby maintaining comparability across configurations.

Using these baseline thresholds, we trained models with progressively larger and arbitrarily chosen hidden layers: for MNIST, widths of $\{35, 50, 100, 200, 250, 500, 1\ 000, 5\ 000\}$ nodes, and for Fashion-MNIST, widths of $\{300, 350, 400, 500, 800, 1\ 000, 2\ 000, 5\ 000\}$ nodes.

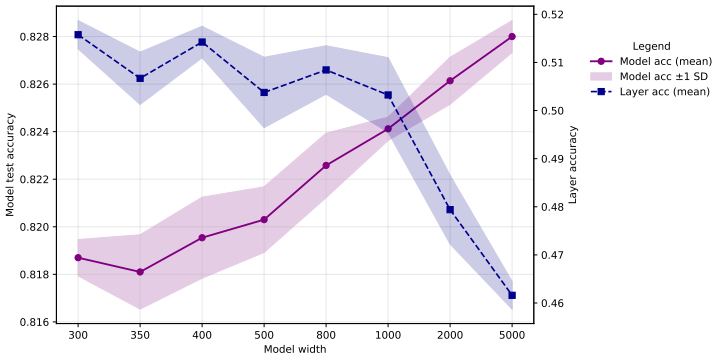
Diversity measure computations: Diversity measures were calculated using the procedures discussed in Section 4.3.3. For each trained model, the diversity among node-level

predictions was computed using the same set of classical measures as in Section 5.2, although only the chosen representative measures (as mentioned in Section 5.2.3) will be analysed for discussion.

5.3.2 Performance results



(a) MNIST



(b) Fashion-MNIST

Figure 5.6: Model and layer accuracy versus network width for single-layer MLPs trained on MNIST and Fashion-MNIST. *(a)* MNIST and *(b)* Fashion-MNIST results showing the mean (± std) model test accuracy (solid line, left axis) and mean layer accuracy (dashed line, right axis) for increasing model widths.

Model performance: Model performance increases consistently as the capacity of a single-layer network is increased. Figure 5.6 shows that for both the MNIST *(a)* and Fashion-MNIST *(b)* datasets, accuracy improves as the number of hidden nodes increases.

On MNIST, the average test accuracy rises from approximately 90.6% at a width of 35 nodes to around 92.1% at 5 000 nodes. Fashion-MNIST displays a similar trend, with accuracy increasing from roughly 81.9% at 300 nodes to 82.8% at 5 000 nodes. These results indicate that increasing the model capacity yields measurable improvements in generalisation performance for both datasets.

Layer performance: For the single-layer networks trained on the MNIST and Fashion-MNIST datasets, increasing model capacity does not lead to a direct improvement in layer-wise performance. Instead, we observe an initial period of fluctuation followed by a pronounced decline once the width becomes sufficiently large (Figure 5.6). In MNIST, layer accuracy rises slightly from roughly 50.6% to 53.3% when increasing the width from 35 to 100 nodes, and then fluctuates up to around 250 nodes before dropping sharply to approximately 28.3% at a width of 5 000 nodes. Fashion-MNIST exhibits a similar pattern, although the variation in accuracy is less pronounced: accuracies range from 51.6% at 300 nodes to 50.3% at 1 000 nodes, followed by a clear decline from 2 000 nodes onward, reaching about 46.2% at 5 000 nodes.

These results indicate that increasing the capacity (width) of a single-layer network does *not* necessarily improve layer accuracy. Despite the larger number of subpredictors within the layer, excess width ultimately leads to reduced performance at the layer level.

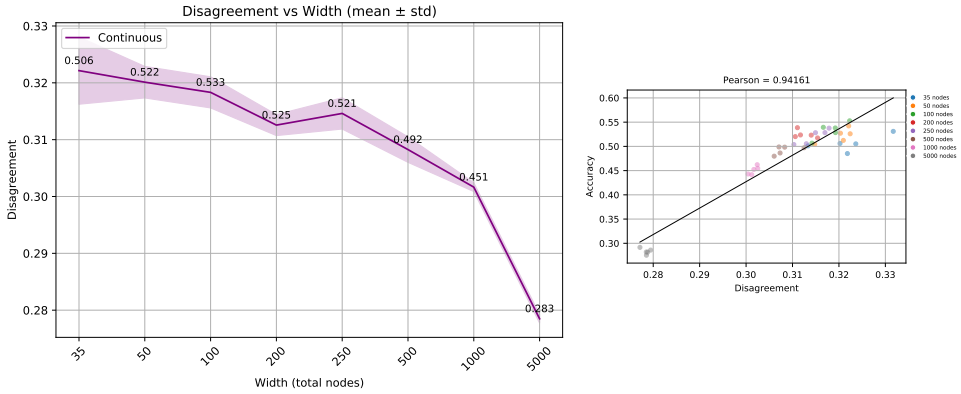
Discussion: These results reveal that while increasing model width reliably improves overall model performance, it has the opposite effect on our layer-level implicit ensemble: layer accuracy systematically decreases as the number of subpredictors grows. This increasing gap between model and layer performance suggests that hidden-layer subpredictors as we define them, do not become stronger classifiers as capacity increases, even though the full network becomes more expressive. We hypothesize that this behaviour is linked to the optimisation dynamics of wide networks, in which the first layer weights tend to move more slowly from their initialisation as width increases. If the first layer weights remain close to their initialisation, the resulting node-level classifiers would behave more like random classifiers and therefore be less accurate, explaining the observed decline in

layer performance. To investigate this hypothesis, we performed an additional experiment (see Appendix A.3.1) analysing the displacement of weights across widths, and find that the empirical results are consistent with our hypothesis.

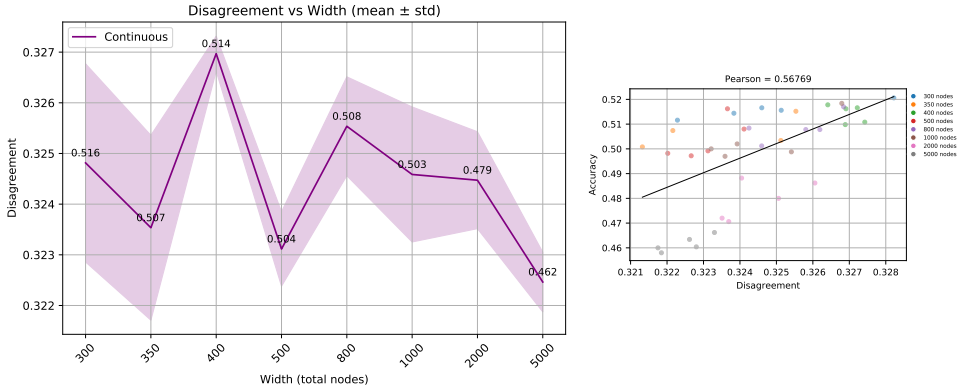
Our analysis does reveal a critical point concerning our model capacity and generalisation results. Even the smallest capacity model achieves interpolation (near-zero loss) on the small subset of training data. This confirms that our experiments are operating in the over-parameterised regime.

This context is important for interpreting the observed relationship we observe between capacity and model performance. In the under-parameterised regime, when model capacity is very small, the model underfits the data and exhibits poor generalisation. As the capacity increases, the generalisation error decreases until reaching a minimum point. Beyond this point, further increases in model capacity causes the generalisation error to increase again as the interpolation threshold is approached [82]. At the interpolation threshold, the model has enough capacity to perfectly fit the training data. Increasing model capacity beyond the interpolation threshold (entering the over-parameterised regime) may lead to overfitting, resulting in degraded generalisation performance [82]. However, since all the trained models have interpolated on the training data—which is a small 2 000 sample subset—even from the smallest model, the continued increase in model generalisation performance as we increase the capacity is reminiscent of the "double-descent" phenomenon [82], [83]. This phenomenon demonstrates an increase in test performance when the model is pushed far into the over-parameterised regime [82], [83]. Consequently, as all models are in the over-parameterised regime and model performance is increasing with capacity, it provides evidence that they have entered the "second descent phase" of the double descent curve.

5.3.3 Diversity trends



(a) MNIST

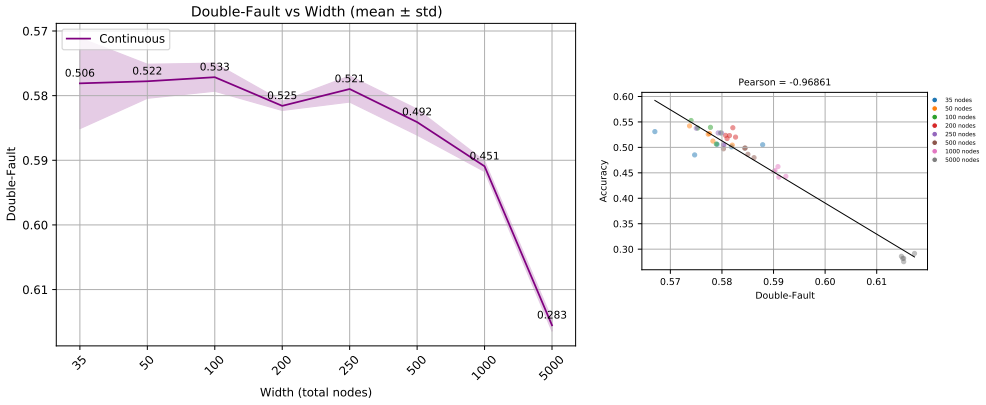


(b) Fashion-MNIST

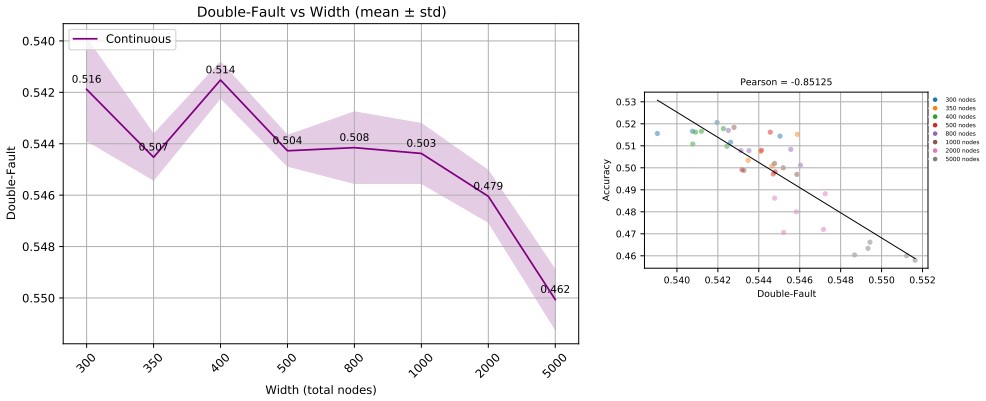
Figure 5.7: *Disagreement* versus network width for single-layer networks trained on MNIST and Fashion-MNIST. *(left)* The mean (± std) *Disagreement* values across five seeds for increasing model widths. Annotated values indicate corresponding mean layer accuracies. *(right)* The relationship between layer accuracy and *Disagreement* across all models.

***Disagreement* across model widths:** Across both MNIST and Fashion-MNIST, we observe a positive correlation between the *Disagreement* measure and layer performance (Figure 5.7). For MNIST, this relationship is strong (Pearson $r = 0.942$), whereas for Fashion-MNIST it is noticeably weaker but still positive ($r = 0.568$). Although the visual trends differ between the datasets, the quantitative results consistently show that higher

Disagreement values correspond to higher layer accuracy. Importantly, both *Disagreement* and layer accuracy decrease as network width increases, indicating that wider networks tend to produce weaker and less diverse node predictors. These findings suggest that, despite some dataset-specific variation in magnitude, *Disagreement* tracks the decline in layer performance as model capacity grows.



(a) MNIST

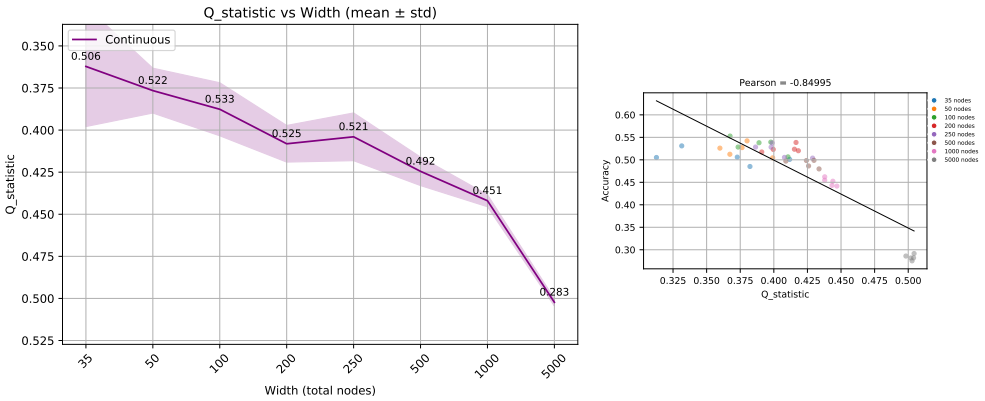


(b) Fashion-MNIST

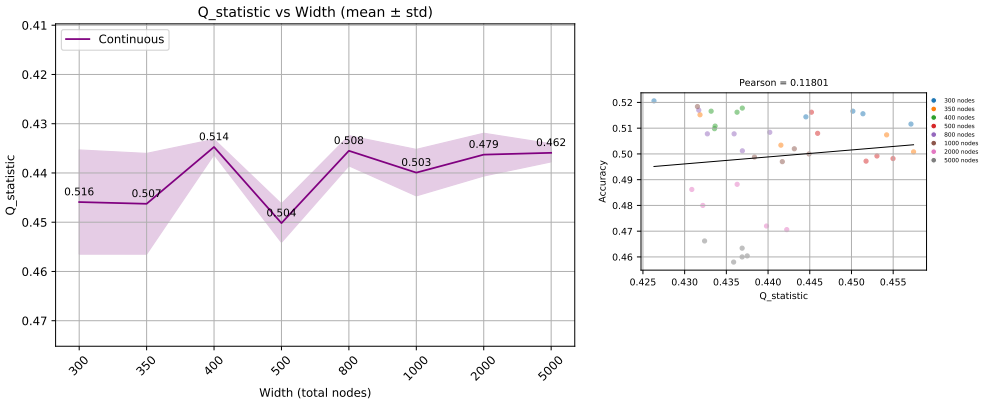
Figure 5.8: *Double-fault* versus network width for single-layer networks trained on MNIST and Fashion-MNIST. *(left)* The mean (± std) *DF* values across five seeds for increasing model widths. Annotated values indicate corresponding mean layer accuracies. Note that the y-axes have been inverted for visual consistency with other diversity measures (higher means more diversity). *(right)* The relationship between layer accuracy and *DF* across all models.

Double-fault across model widths: Figure 5.8 illustrates how the *Double-fault* (DF) measure changes as the network capacity is varied. We observe a generally decreasing trend in diversity as width is increased, reflected by rising DF values. This decrease in diversity mirrors the declining layer performance we see in Figure 5.6 with strong negative correlations for both MNIST and Fashion-MNIST datasets ($r = -0.969$ and $r = -0.851$, respectively). These results indicate that as larger widths lead to poorer layer-level classification, the corresponding node predictions become more prone to failing on the same samples—demonstrating a clear link between layer performance and reduced diversity when measured via DF.

***Q-statistic* across model widths:** The *Q-statistic* shows inconsistent behaviour across model widths when comparing the MNIST and Fashion-MNIST datasets (Figure 5.9). On MNIST, the *Q-statistic* follows the layer-performance trend reasonably well, both visually and in terms of correlation ($r = -0.850$). In contrast, the Fashion-MNIST results are irregular—fluctuating initially before remaining flat—exhibiting only a weak *positive* correlation with layer accuracy ($r = 0.118$). These conflicting patterns suggest that the *Q-statistic* does not provide a stable or interpretable indicator of diversity behaviour when model width is varied, and its relationship with layer accuracy remains inconclusive in this capacity setting.



(a) MNIST

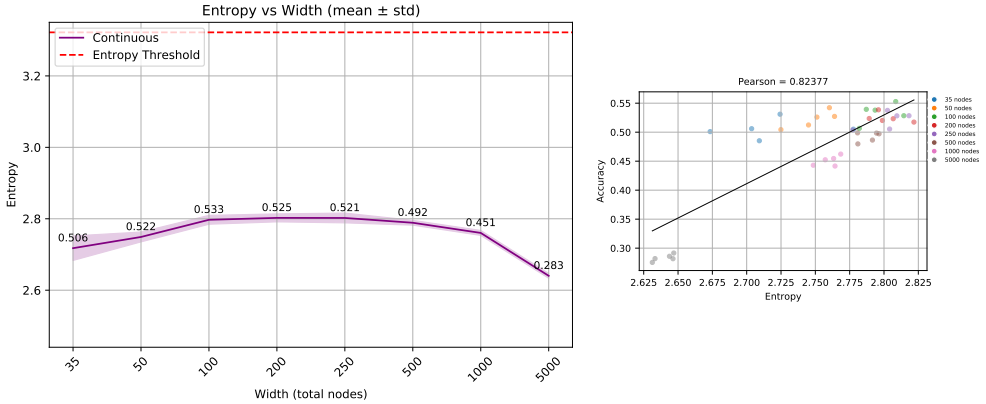


(b) Fashion-MNIST

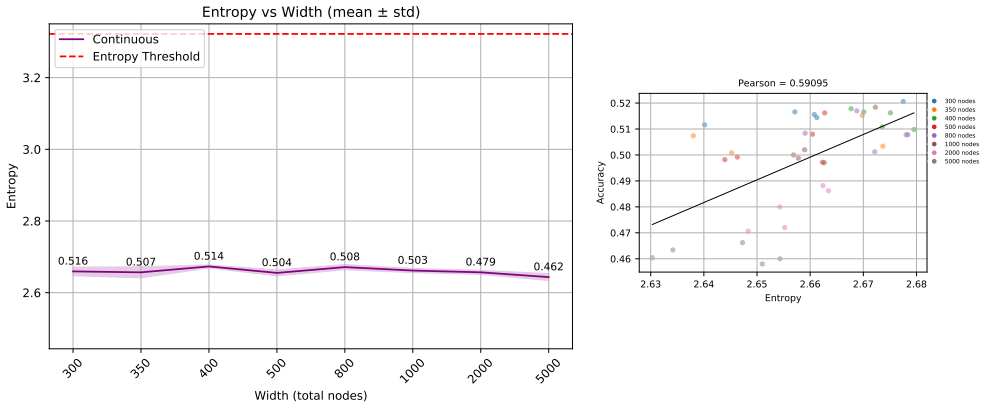
Figure 5.9: *Q-statistic* versus network width for single-layer networks trained on MNIST and Fashion-MNIST. *(left)* The mean (± std) *Q-statistic* values across five seeds for increasing model widths. Annotated values indicate corresponding mean layer accuracies. Note that the y-axes have been inverted for visual consistency with other diversity measures (higher means more diversity). *(right)* The relationship between layer accuracy and *Q-statistic* across all models.

***Entropy* across model widths:** The *Entropy* measure also shows some differences in behaviour across the MNIST and Fashion-MNIST datasets (Figure 5.10). *Entropy* shows a strong positive correlation with layer accuracy for MNIST at $r = 0.824$, whereas the relationship for Fashion-MNIST is noticeably weaker ($r = 0.591$) but still positive. However, the Fashion-MNIST values change only marginally across width, especially

when compared to the clearer variation seen in MNIST. Despite the marginal changes on Fashion-MNIST, the results still suggest that *Entropy* captures the overall trend in layer performance as network width increases.



(a) MNIST



(b) Fashion-MNIST

Figure 5.10: *Entropy* versus network width for single-layer networks trained on MNIST and Fashion-MNIST. *(left)* The mean (± std) *Entropy* values across five seeds for increasing model widths. Annotated values indicate corresponding mean layer accuracies. *(right)* The relationship between layer accuracy and *Entropy* across all models.

Diversity versus model performance: Figures 5.11 and 5.12 show how each diversity measure relates to the model performance as network width is varied. Across both datasets, the *Disagreement*, *Q-statistic*, and *Entropy* metrics show inconsistent behaviour.

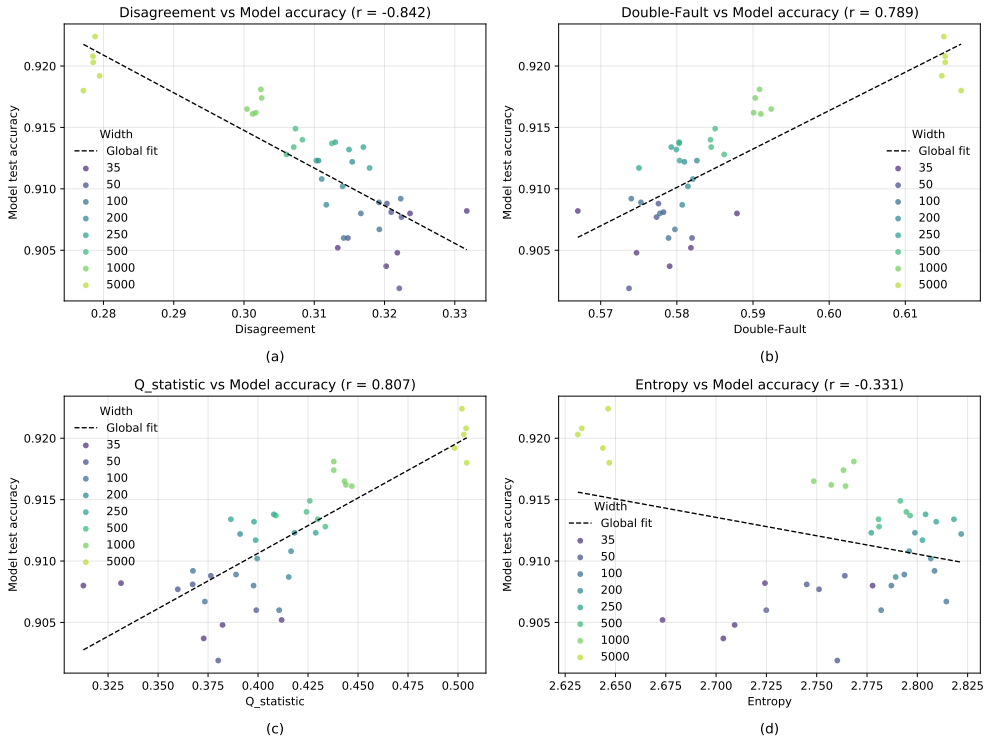


Figure 5.11: Relationship between model test accuracy and diversity across different network widths for the MNIST dataset. Each subplot illustrates the relationship between model performance and a different diversity measure.

While these measures correlate reasonably well with layer- and/or model performance on MNIST (though the correlation for *Entropy* is weak), their relationships significantly weaken on Fashion-MNIST. Notably, the correlation direction for the *Q-statistic* even reverses on the Fashion-MNIST dataset. In contrast, the *DF* measure exhibits a strong and consistent correlation with model performance on both datasets ($r = 0.789$ for MNIST and $r = 0.662$ for Fashion-MNIST). This suggests that while most diversity measures do not provide stable cross dataset relationships in the capacity setting, *DF* remains reliable: as model performance increases, *DF* also increases, indicating that reduced diversity (higher *DF*) aligns with improved model accuracy in this capacity-changing context.

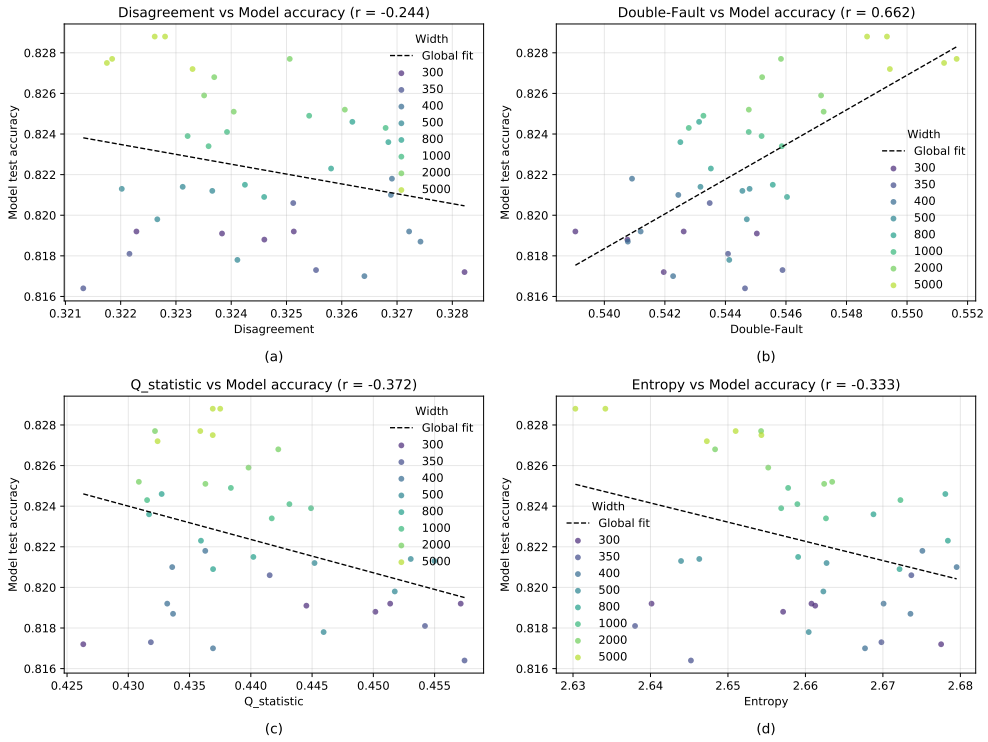


Figure 5.12: Relationship between model test accuracy and diversity across different network widths for the Fashion-MNIST dataset. Each subplot illustrates the relationship between model performance and a different diversity measure.

5.3.4 Discussion

Overall, the results in this section show that increasing the capacity of a single-layer neural network has a complex and non-uniform effect on diversity. While model-level performance increases steadily with width, layer-level accuracy declines sharply once the network becomes sufficiently wide, indicating that the subpredictors within the layer do not benefit from additional capacity in the same way the full model does. This increasing disconnect between model- and layer-level behaviour suggests that additional nodes do not develop into stronger or more independent classifiers; instead, they appear to become increasingly redundant or weakly specialised.

The diversity measures provide further insight into this phenomenon. Most metrics behave inconsistently across MNIST and Fashion-MNIST. Measures such as *Disagreement*, *Q-statistic*, and *Entropy* correlate reasonably well with MNIST, but fail to with Fashion-

MNIST—often showing weak or contradictory trends.

In contrast, the *Double-fault* measure stands out as the only metric that consistently aligns with layer and model behaviour across datasets. As the network widens, DF increases, indicating that nodes become more likely to fail on the same samples. This trend matches the observed decline in layer accuracy and continues to hold even as overall model performance improves. This suggests that, in the capacity-setting, DF captures a meaningful aspect of how subpredictors weaken as their number increases.

This outcome is notably opposite to what one would expect from increasing the number of subpredictors in a traditional ensemble. In typical ensemble settings, adding more members can improve ensemble accuracy by reducing variance, although accuracy gains depend on the degree of diversity among members of the ensemble. In our case however, increasing the number of subpredictors (nodes) leads to *worse* layer-level performance and *reduced* diversity, even as overall model performance continues to improve. This divergence underscores that implicit ensembles within neural network as we define them, do not behave like explicit ensembles in all cases: widening the network enhances the model's expressive power, but does not necessarily produce more competent or more diverse subpredictors.

5.4 Diversity evolution during training

To gain a deeper understanding of how diversity evolves within a *single* neural network, we analyse the dynamics of diversity throughout the training process. By evaluating the network at multiple checkpoints during training, we obtain a finer-grained view of how diversity changes as the model progressively learns the training data. This allows us to identify when diversity increases or decreases, how these changes relate to model performance, and whether key transitions occur early or late in training.

5.4.1 Experimental setup

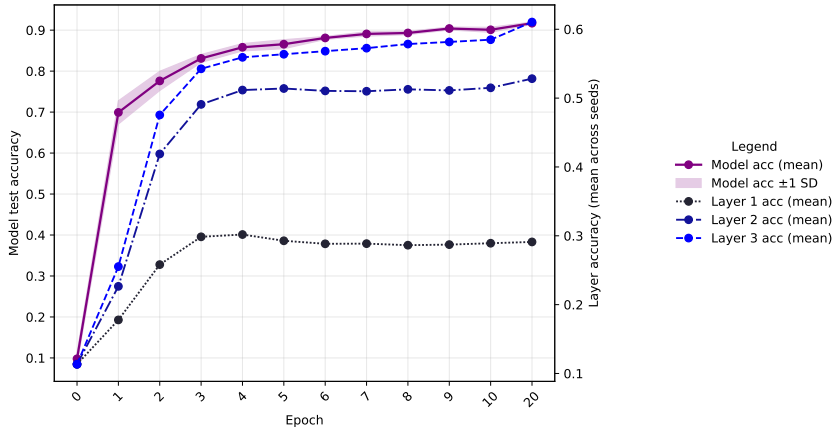
Model architecture and hyperparameter setup: The model architectures used in this section follow the same configuration described in Section 5.2.1. As the analysis requires evaluating diversity across possibly hundreds of checkpoints over different models, we restrict the analysis to the MNIST and Fashion-MNIST datasets to keep the computational cost and time tractable while still capturing meaningful diversity dynamics. All hyperparameters match those listed in Table 5.1.

Training protocol: Models are trained using the same protocol outlined in Section 4.3.2. For this experiment, however, we introduce periodic check-pointing to capture the model at different stages of training—where we are particularly interested in the early stages of training. During each run, models are saved at the following points:

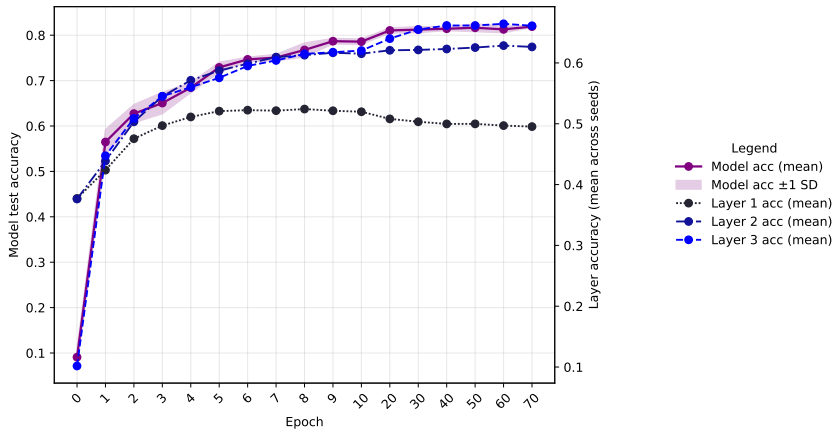
- **Early training phase:** epochs $[0, 10]$ (step size of 1 epoch).
- **Intermediate training phase:** epochs $[20, 100]$ (step size of 10 epoch).
- **Late training phase:** epochs $[200, 1\ 000]$ (step size of 100 epoch).
- **Interpolated model:** final trained model (at 100% training accuracy).

At each checkpoint, test-set performance is recorded; this evaluation is purely diagnostic and does not affect the training trajectory. The full layer-level and diversity analyses described in Sections 3.3 and 4.3.3 are then applied independently to every saved checkpoint, allowing us to track how diversity develops throughout the training process.

5.4.2 Performance results



(a) MNIST



(b) Fashion-MNIST

Figure 5.13: Model and layer test accuracy across training epochs for the MNIST and Fashion-MNIST datasets. The purple lines shows the model-level accuracy *(left-axis)* and the dashed and dotted blue and black lines show layer-level accuracy *(right-axis)* for each layer. The x-axis limit is set by the largest shared epoch achieved across all models.

Layer performance over training: Layer-level performance increases steadily as training progresses initially. Figure 5.13 shows how the performance of all three layers evolves during training for MNIST and Fashion-MNIST. On MNIST (Figure 5.13*(a)*), layer accuracies begin near random-guessing accuracy (10%) at initialisation, then rapidly increase within the first few epochs—most notably up to epoch 3. After this early rapid improve-

ment, the growth in layer accuracy slows and eventually plateaus as training continues, remaining relatively constant until the model reaches interpolation with layer accuracies around 29%, 54%, and 62% for Layer 1-3 respectively.

Fashion-MNIST exhibits a similar pattern (Figure 5.13*(b)*), although with a slight deviation with Layer 1 and 2 accuracies at initialisation. All layers show monotonic improvement early in training before reaching their respective plateaus at different points: Layer 1 plateaus around epoch 5, Layer 2 around epoch 8, Layer 3 noticeably later, around epoch 30. These results indicate that as the network learns, all hidden layers become progressively more accurate in their predictions, with most of the improvement occurring in early training before converging to relatively constant performance levels.

Model performance over training: Figure 5.13 shows how model test accuracy evolves throughout training. For both MNIST and Fashion-MNIST, model performance increases sharply at the start of training, rising from random-guessing accuracy ($\sim 10\%$) at initialisation to approximately 70% on MNIST and 57% on Fashion-MNIST by the first epoch. After this rapid initial improvement, accuracy continues to climb more gradually, ultimately reaching around 92% for MNIST[9] and 82% for Fashion-MNIST. These results confirm the expected behaviour observed widely in the literature [6, pp. 108–112]: as training progresses, the model steadily improves its generalisation performance—given enough network capacity, proper regularisation or optimiser settings etc.—exhibiting a consistently increasing accuracy over training epochs.

Discussion: The performance results show that both layer-level and model-level accuracy increase steadily as training progresses, with the most substantial improvements occurring within the first few epochs, after which the performance increases more gradually before stabilising for both the layer- and model-performances. By the time the models reach interpolation, the layers have generally reached a plateau in their accuracy levels. These results confirm that, under stable training conditions, both the network as a whole

[9]The relatively low MNIST accuracy is primarily due to the reduced training set used in this study (2 000 samples instead of the standard 60 000) as well as the absence of hyperparameter tuning aimed at maximising classification performance.

and the individual layers consistently improve their predictive ability over the course of training.

Although model accuracy continues to improve gradually throughout training, layer accuracy tends to stabilise much earlier. This could suggest that the hidden layers acquire most of their discriminative structure in the early stages of learning.

Finally, we note the deviation in initial layer accuracies between MNIST and Fashion-MNIST which we will explore in future work.

5.4.3 Diversity results

Diversity changes with network training: We observe a clear and consistent pattern in how diversity evolves during training: for both MNIST and Fashion-MNIST, diversity increases initially across all layers and across all diversity measures (Figure 5.14). For MNIST, *Disagreement* begins near zero at initialisation (values of 0.00, 0.04, and 0.16 for Layer 1–3) and rises sharply within the first few epochs, reaching approximately 0.26, 0.28, and 0.29 by epoch 2, followed by a gradual increase to final values of around 0.28, 0.32, and 0.33. Fashion-MNIST shows a similar upward trend in *Disagreement*, and the remaining diversity measures—*DF*, *Q-statistic*, and *Entropy*—exhibit comparable monotonic increases in diversity over training for both datasets. Although Fashion-MNIST shows higher initial diversity at initialisation, this does not affect the overall trend: as training progresses, diversity consistently increases across all layers. These results demonstrate that diversity develops alongside training, with most of its growth occurring early in training, and continuing to increase steadily as the model learns.

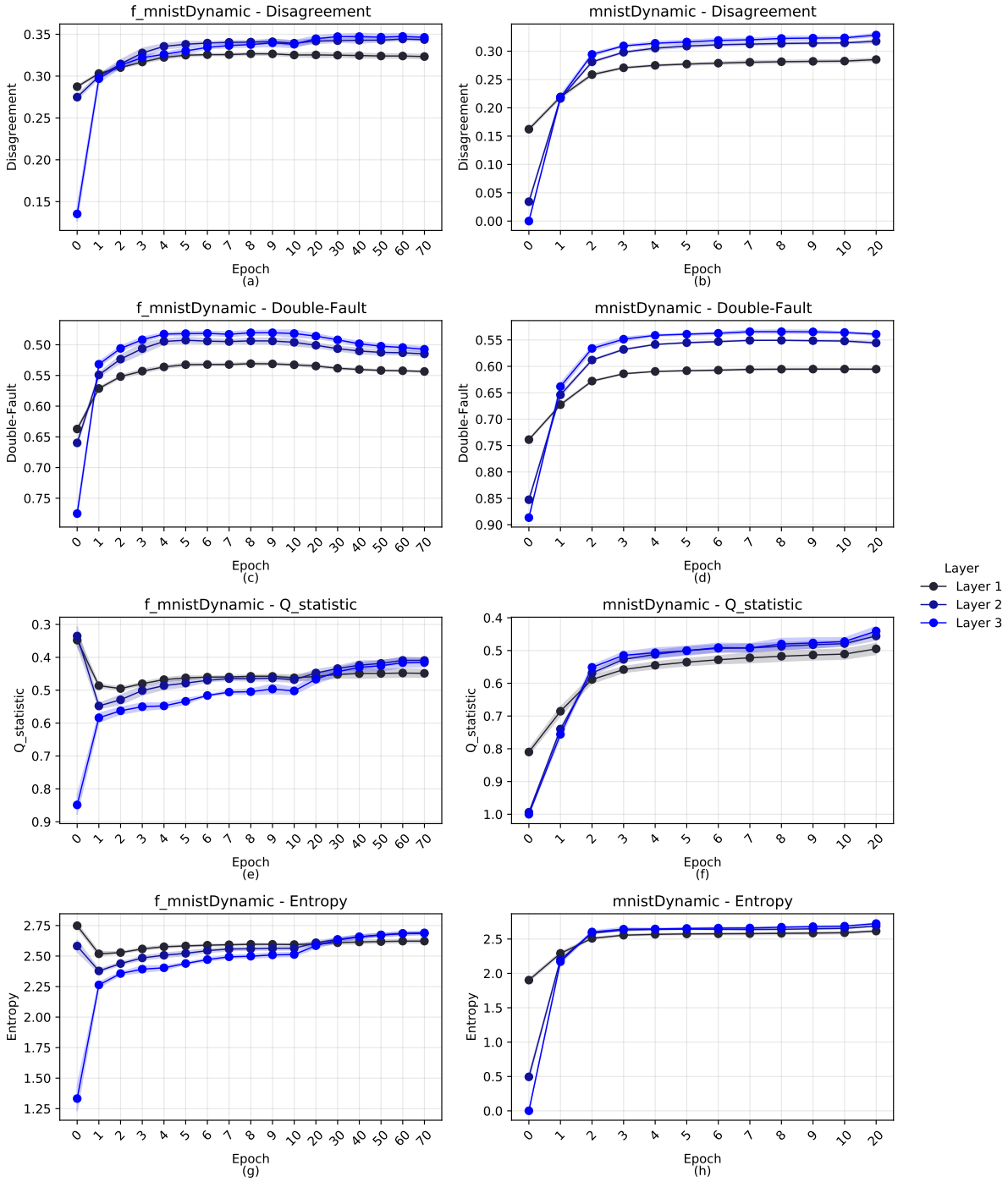


Figure 5.14: Diversity across training epochs for both MNIST *(right)* and Fashion-MNIST *(left)*, shown for all diversity measures and for all three hidden layers. Each subplot illustrates how diversity evolves from initialisation to interpolation with colours indicating the different layers.

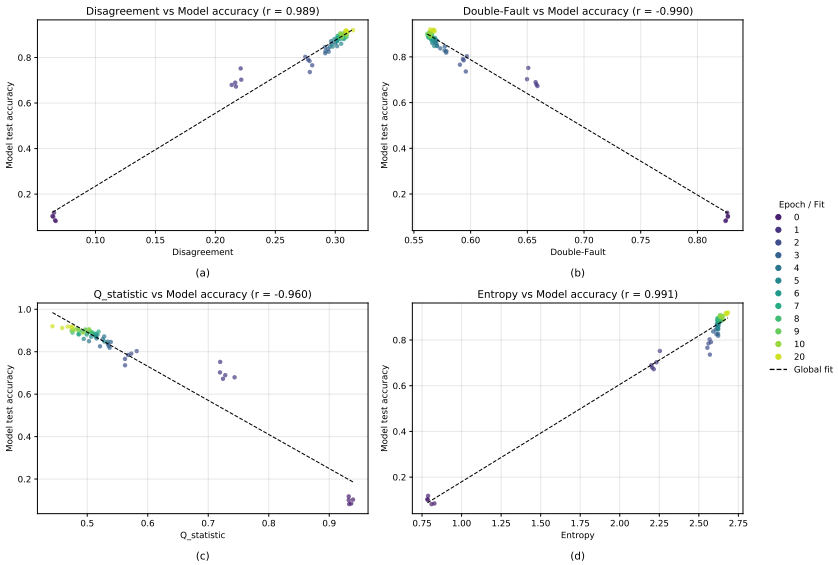


Figure 5.15: Relationship between averaged layer-wise diversity and model test accuracy across training checkpoints for MNIST. Each subplot shows how a specific diversity measure correlates with overall model performance as training progresses.

Relationship between diversity and model performance: Figures 5.15 and 5.16 show how averaged diversity across layers relates to model accuracy throughout training. For both MNIST and Fashion-MNIST, we observe strong correlations between diversity and model performance across all diversity measures. On MNIST, the correlations are $r = 0.989$, -0.990, -0.960, and 0.991 for *Disagreement*, *DF*, *Q-statistic*, and *Entropy*, respectively. Fashion-MNIST exhibits similarly strong relationships ($r = 0.984$, -0.922, -0.592, and 0.916), with only the *Q-statistic* showing a slightly weaker correlation due to lower and less stable values near initialisation. These consistent trends across datasets and metrics suggest that as the model becomes more accurate during training, the overall diversity within the network increases correspondingly.

Interpretation: Overall, the results show that diversity increases steadily as the network trains. Early in training, node predictions change rapidly as the model begins to separate classes, which leads to a sharp increase in diversity. As training continues, diversity keeps increasing, although more gradually, alongside improvements in model accuracy. The strong correlations between diversity and performance indicate that when-

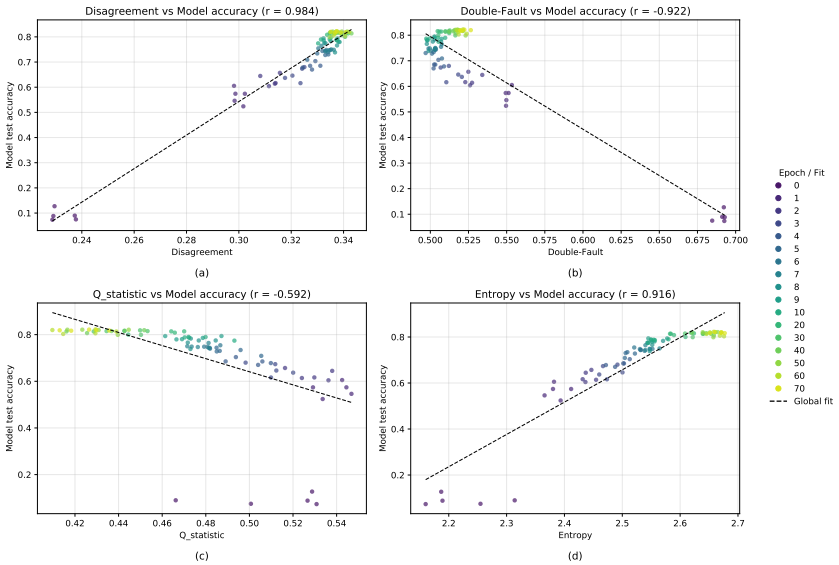


Figure 5.16: Relationship between averaged layer-wise diversity and model test accuracy across training checkpoints for Fashion-MNIST. Each subplot shows how a specific diversity measure correlates with overall model performance as training progresses.

ever the model learns a task better, the nodes within each layer also become more varied in their predictions.

5.4.4 Discussion on diversity evolution during training

The results in this section show a coherent picture of how diversity emerges and evolves during training. Both datasets show the same overall behaviour: diversity generally begins low at initialisation and then increases rapidly during the first few epochs—mirroring the period where both model accuracy and layer accuracy improve most sharply. After this initial phase, diversity continues to increase more gradually as training progresses, ultimately stabilising once the model reaches interpolation. This pattern is consistent across all diversity measures considered, indicating that the increase in diversity is not dependent on a particular metric or dataset, but reflects a fundamental shift in how the nodes within each layer behave as learning progresses.

Although diversity trajectories are reported over the full training process, this analysis

serves a diagnostic rather than purely evaluative purpose. Tracking diversity across epochs allows us to identify distinct points during training in which the metrics might behave differently. In the early stages of training, node predictors are typically near chance; diversity measures in this regime are therefore dominated by random disagreement and do not reliably reflect the differences in learned representations of the subpredictors. As optimisation progresses and the network approaches convergence, node predictors become more competent and prediction patterns stabilise. Diversity measures computed in this late-stage regime are therefore more meaningful, as they reflect systematic relationships between subpredictors rather than stochastic variation.

Because the diversity measures used in this study are computed from discrete node decisions, their magnitude is inherently coupled to node competence. As training progresses, predictions may progressively become more consistently correct, creating opportunities for disagreement- and joint-error to diminish, which can reduce the measured diversity.

Diversity is therefore not treated as an unconditional objective. Increases in diversity can reflect beneficial complementary error patterns among competent predictors, but can also arise from weakened or unstable predictors.

The full-epoch analysis is thus used to determine when diversity metrics become interpretable indicators of predictor relationships and generalisation behaviour. All diversity–performance analyses in subsequent sections are based on these late-stage converged models.

When relating these trends to model performance, we observe strong and consistent correlations across both datasets for all diversity measures. Higher diversity measures reliably correspond to higher model accuracy, suggesting that diversity and performance grow together throughout training. Although some measures (such as the *Q-statistic*) show slightly weaker correlations early in training—primarily due to the instability at initialisation—the overall relationship remains clear: improvements in model performance are accompanied by increases in diversity.

5.5 Conclusion

This chapter presented an *exploratory* investigation into how diversity emerges, develops, and behaves within a *single* neural network when viewed as an implicit ensemble of nodes as subpredictors. Through three complementary investigations—layer-wise diversity in a fixed-depth network, the effect of network capacity on diversity, and the evolution of diversity during training— we developed an understanding of how subpredictors (nodes) differ, specialise, or become redundant under different modelling conditions.

Our layer-wise analysis showed that classical diversity measures—originally designed for explicit ensembles—can meaningfully characterise differences between layers in an implicit ensemble. In fully connected networks, both layer accuracy and diversity increased consistently with depth across datasets, and most diversity measures correlated strongly with layer performance. Even in a convolutional architecture, where layer accuracy no longer followed a monotonic trend with depth, diversity continued to mirror performance closely. These results demonstrate that diversity captures important distinctions in the discriminative strength of different layers.

In contrast, examining diversity under varying network capacity revealed behaviour very different from what one might expect from traditional ensemble theory. Increasing the width of a single-layer network consistently improved overall model accuracy but *reduced* both layer-level accuracy and layer-level diversity. Most diversity measures behaved inconsistently across datasets in this setting, with only *Double-fault* showing stable, interpretable trends that aligned with changes in layer- and model performance. These findings show us that widening a network by adding more hidden nodes does not lead to increased implicit ensemble (layer) performance, which tells us that implicit ensembles as we define them, do not behave like explicit ensembles when capacity is increased.

Finally, the dynamic training analysis showed that diversity increases naturally throughout training, especially during the early epochs where model accuracy also improves most rapidly. This trend was consistent across datasets and metrics, and strong correlations between diversity and performance suggest that diversity grows in tandem with the network's

ability to perform the task. This indicates that diversity is neither fixed at initialisation nor solely a structural property, but rather develops as a consequence of optimisation.

These insights motivate the next question: if diversity naturally emerges under certain conditions, and collapses under others, can we *deliberately induce* diversity through architectural or regularisation strategies to improve network behaviour?

Chapter 6

Inducing diversity in implicit deep ensembles

This chapter explores how controlled interventions—Dropout, DropConnect, and batch size—can be used to deliberately induce diversity in neural networks and how these techniques affects the relationship between layer- and model-level performance and diversity.

6.1 Overview

Building on earlier chapters that established diversity as a measurable property of single neural networks viewed as implicit ensembles, we now shift focus from measurement to intervention. That is, instead of passively observing diversity as an emergent feature, we actively attempt to manipulate it using regularisation mechanisms. We investigate whether deliberately inducing diversity through controlled interventions results in improved generalisation, and whether the relationship between diversity and performance becomes more consistent under such settings.

In this chapter, we investigate methods for inducing diversity within neural networks under the implicit ensemble framework. Specifically, we examine how induced diversity influences both the correlation between layer performance and diversity, and how it relates to the overall model performance. Three techniques are explored for inducing diversity: Dropout, DropConnect, and variations in batch sizes. Each of these methods alters the degree to which subpredictors interact, depend on, and co-adapt with one another—effectively modifying their independence—and is therefore expected to influence the diversity of their predictions. The experiments in this chapter aim to determine whether these techniques can meaningfully induce diversity within an implicit ensemble and how such induced diversity impacts model generalisation. By testing multiple regularisation strengths and batch sizes, we analyse how varying the level of regularisation affects both network performance and internal diversity.

6.2 How to induce diversity in a network

In classical ensemble theory, combining multiple classifiers is only beneficial if those predictors are not perfectly correlated—that is, if they make *different* errors on the same input. Hansen and Salamon [48] formally demonstrated that the ensemble's generalisation error decreases when its members exhibit uncorrelated errors, provided the average accuracy of the members remain above chance. Consequently, methods that increase the independence or heterogeneity of the ensemble members tend to improve generalisation, assuming they do not excessively degrade individual accuracy.

While these statements were originally derived for *explicit ensembles* of separate models, the same principles can be extended to *implicit ensembles* within a deep neural network. In the framework we discussed in Chapter 3, each hidden node is interpreted as an individual classifier. The hidden layer therefore functions as a collection of subpredictors whose outputs can be combined to form an implicit ensemble. Encouraging diversity among these subpredictors might increase the network's ability to generalise. We then ask *how we can deliberately induce such diversity* through controlled modifications to the training

process.

We examine three different techniques in this chapter: Dropout, DropConnect, and variation in batch size. Each of these regularisation methods is well-established in the literature for improving a neural network's generalisation performance. Since our models are trained in the over-parameterised regime—interpolating on a small, stratified subset of the training data for the MLP models—we expect that introducing regularisation will have a positive outcome on the generalisation performance. The main questions we address are: (1) how do these regularisers influence diversity within a single neural network?; and (2) can changes in diversity be linked to changes in generalisation performance?

6.2.1 Dropout

Dropout randomly deactivates a subset of nodes during each training iteration [18]. For a hidden node computing an activation, given a specific input sample $\mathbf{x}^{(i)}$, the activation of node $n \in \mathcal{N}$ in a layer ℓ is computed as in Equation (2.1):

$$h^n_{(\ell)} = \sigma\left(b^n_{(\ell)} + \sum_{p=1}^{d_{(\ell-1)}} W^{p,n}_{(\ell-1),(\ell)}\ h^p_{(\ell-1)}\right),$$

where $h^p_{(\ell-1)}$ is the activation of node p in the previous layer $\ell - 1$ for an input $\mathbf{x}^{(i)}$, $W^{p,n}_{(\ell-1),(\ell)}$ is the weight connecting node p in layer $\ell - 1$ to node n in layer ℓ, $b^n_{(\ell)}$ is the bias for node n, and $\sigma(\cdot)$ is a non-linear activation function (e.g., ReLU).

Under Dropout, a binary mask $r^n_{(\ell)} \sim \text{Bernoulli}(p)$ is applied to the output of each node $n \in \mathcal{N}$ in a layer ℓ, yielding

$$\tilde{h}^n_{(\ell)} = r^n_{(\ell)} h^n_{(\ell)},$$

where p is the probability that a node remains active during training. This stochastic masking forces each node to operate with varying subsets of its inputs, reducing reliance on specific activation patterns and preventing co-adaptation [18]. One could argue that this resembles an implicit ensemble, where each unique Dropout mask corresponds to a different sub-network. Training with Dropout effectively averages predictions over an

exponentially large set of these sub-networks [18]. Within our implicit ensemble view (Section 3.3), this randomness encourages node-level specialisation, which we believe should promote diversity among subpredictors. As a result, increased Dropout is expected to yield greater diversity—up to a point where too much Dropout disrupts learning dynamics or harms overall model performance.

It is important to note that the definition of p used in this study differs from the original Dropout formulation. While Srivastava et al. [18] define p as the probability of a node *remaining active* during training, we adopt the opposite convention: p refers to the probability of a node *being dropped*. This change aligns with common deep learning frameworks such as PyTorch, where the Dropout parameter specifies the drop probability. For consistency with our implementation, all references to p in this study follow this convention, meaning higher values of p correspond to stronger regularisation. Furthermore, this inverted Dropout convention as used in PyTorch, scale the activations during training by a factor of $\frac{1}{(1-p)}$ to preserve their expected magnitude; consequently, no additional scaling is required at test time when Dropout is disabled.

6.2.2 DropConnect

DropConnect [84] is a stochastic regularisation method similar to Dropout, but instead of randomly deactivating nodes, it randomly deactivates individual weights during training. Given a weight matrix $\mathbf{W}_{(\ell-1),(\ell)} = [W^{p,n}_{(\ell-1),(\ell)}]$, a binary mask matrix $\mathbf{M}_{(\ell-1),(\ell)} = [M^{p,n}_{(\ell-1),(\ell)}]$ is sampled independently where each $M^{p,n}_{(\ell-1),(\ell)} \sim \text{Bernoulli}(p)$. These binary mask values are applied element-wise to the weight matrix $\mathbf{W}_{(\ell-1),(\ell)}$, producing effective weights:

$$\bar{W}^{p,n}_{(\ell-1),(\ell)} = M^{p,n}_{(\ell-1),(\ell)} W^{p,n}_{(\ell-1),(\ell)},$$

yielding a new forward activation at node n in layer ℓ of:

$$h^n_{(\ell)} = \sigma\left(b^n_{(\ell)} + \sum_{p=1}^{d_{(\ell-1)}} \bar{W}^{p,n}_{(\ell-1),(\ell)}\ h^p_{(\ell-1)}\right).$$

This means that instead of each node receiving a clean, consistent input vector, it receives

a version where connections are randomly dropped—leading to stochastic and noisy pathways. This disruption discourages reliance on specific connections in a similar fashion to Dropout.

In the implicit ensemble framework, DropConnect can be interpreted as promoting diversity not by masking entire subpredictors, but by randomly modifying the structure of their interconnections. This induces variability in how subpredictors form and operate across training iterations. As with Dropout, moderate DropConnect regularisation has been shown to improve generalisation by weakening co-adaptation among nodes [84]. Accordingly, we expect to see a correlation between DropConnect probability and both diversity and model performance.

6.2.3 Batch-size variation

Besides using explicit regularisation techniques, we aim to exploit the inherent randomness of Stochastic gradient descent (SGD) [6, pp. 149–150][7, pp. 141–143]—via the batch size—as a source of *implicit* regularisation [7, pp. 141–143], which should have an affect on the generalisation performance of the model.

How is batch size connected to regularisation? Smith et al. [20] shows that stochastic gradient descent (SGD) can be interpreted as optimising a loss function that includes an additional regularisation term arising from the stochasticity of mini-batches. Comparing SGD with discrete time-steps to the continuous-time version, Smith et al. [20] demonstrates that continuous SGD is equivalent to the discrete time-step case if the loss function $J[\phi]$ is modified with two penalty terms [7, pp. 141–143], namely,

$$\tilde{J}_{SGD}[\phi] = J[\phi] + \frac{\eta}{4}\left\|\frac{\partial J}{\partial \phi}\right\|^2 + \frac{\eta}{4S}\sum_{j=1}^{S}\left\|\frac{\partial J_j}{\partial \phi} - \frac{\partial J}{\partial \phi}\right\|^2, \tag{6.1}$$

where J is the full-dataset loss, J_b is the loss computed on the b-th mini-batch, η is the learning rate, $\phi \in \{\mathcal{W}, \mathcal{B}\}$ denotes learnable parameters, and S is the number of mini-batches. The final term in Equation (6.1) is proportional to the *variance* of the per-batch

gradients. This variance acts as an implicit regulariser because SGD favours regions of parameter space where all batches agree on the slope—i.e., where the gradient variance is small. Such regions typically respond to flatter, more stable solutions that fit all data points reasonably well, rather than memorising subsets of the data with highly variable gradients.

Prince [7, pp. 141–143] mentions that this implicit regularisation becomes stronger when the batch size is small: with fewer samples per batch, the gradient variance increases, and the optimiser is pushed towards parameters that minimise this variance.

Although the batch size argument is framed in terms of SGD, the underlying mechanism is the stochasticity of mini-batch gradients, which applies equally to Adam. Adam's adaptive moments are estimated from mini-batch gradients and therefore remain sensitive to batch size [85]; decreasing batch size increases gradient noise, and may reduce subpredictor correlation. Hence, varying batch size under Adam is still expected to have a similar qualitative impact as under SGD.

How does batch size affect generalisation? A smaller batch size is argued to yield better generalisation than larger ones. This trend is usually attributed to the additional gradient noise introduced by mini-batch sampling, which helps the optimiser escape sharp or poorly generalising minima. Small-batch SGD naturally leads to flatter, better generalising solutions [16], [86].

How do we expect batch size to affect diversity? Given its relation to an implicit regulariser, we expect the batch size to influence the diversity of the subpredictors within a network. Smaller batch sizes introduce higher gradient noise, which could encourage different hidden nodes to specialise in slightly different ways. In other words, with noisier updates, the subpredictors receive more varied training signals, making it less likely that they show correlated behaviours. Consequently, if diversity indeed relates to generalisation performance, then decreasing the batch size should increase subpredictor diversity, mirroring the improvement in model-level generalisation commonly associated

with smaller batches.

6.2.4 Summary

Each of these techniques introduces stochasticity into training which we expect should influence the diversity of subpredictor predictions. Dropout operates at the activation level, DropConnect at the connection level, and batch-size variation at the gradient-update level. Together, they provide a spectrum of different methods for encouraging independency within layers. The following sections evaluate the empirical effects of these mechanisms on layer-wise and averaged diversity (averaged over layers) and their correlation with both layer and model performance.

6.3 Dropout

6.3.1 Experimental setup

Model architectures: To investigate the effect that Dropout has on model performance and diversity in a single neural network, we again use fully connected neural networks with three hidden layers, each with 512 nodes per layer—as in Section 5.2.1. The CNN architecture follows the same general structure as in Section 5.2.1, except that Dropout layers are added before each pooling layer (see Table 6.1) and at each layer in the final MLP stage.

Table 6.1: CNN specification (conv bias=True, activation=ReLU, BN=False) with Dropout being applied. "$\times n$" indicates repeated blocks.

Stage	Layers	Kernel / Stride / Pad	Output size (for 32×32 input)
Input			32×32×3
S1	Conv 64 ×2	$(3, 1, 1)$	32×32×64
	Dropout		
	MaxPool	$(2, 2, 0)$	16×16×64
S2	Conv 128 ×2	$(3, 1, 1)$	16×16×128
	Dropout		
	MaxPool	$(2, 2, 0)$	8×8×128
S3	Conv 256 ×3	$(3, 1, 1)$	8×8×256
	Dropout		
	MaxPool	$(2, 2, 0)$	4×4×256
S4	Conv 512 ×3	$(3, 1, 1)$	4×4×512
	Dropout		
	MaxPool	$(2, 2, 0)$	2×2×512
S5	Conv 512 ×3	$(3, 1, 1)$	2×2×512
	Dropout		
	MaxPool	$(2, 2, 0)$	1×1×512
Head	Flatten		512
	Dense 512		512
	Dropout		
	Dense 512		512
	Dropout		
	Dense 512		512
	Dropout		
	Dense(C)		C

Model training protocol, hyperparameter setup, and analysis: Models are trained on their corresponding datasets using the protocol described in Section 4.3.2. We use the Adam optimiser with a learning rate of 0.0003, a batch size of 256, a maximum of 1 000 epochs, and a learning-rate scheduler with a step size of 1 and a decay factor (γ) of 0.99. No DropConnect or Batch normalisation is applied. Dropout is used as the primary form of regularisation in both MLP and CNN architectures. For MLPs, Dropout is applied with drop probabilities of $p = \{0.10,\ 0.20,\ 0.30,\ 0.40,\ 0.50\}$. For CNNs, Dropout is applied at

both the convolutional and fully connected stages using $p = \{0.10,\ 0.15,\ 0.20,\ 0.25,\ 0.30\}$. A summary of the hyperparameters is provided in Table 6.2.

Table 6.2: Summary of hyperparameters for MLP and CNN architectures. This configuration uses a batch size of 256 with Dropout as the only regularisation mechanism applied.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	0.0003
Batch size	256
Epochs (max)	1 000
Dropout (MLP)	$p = \{0.10,\ 0.20,\ 0.30,\ 0.40,\ 0.50\}$
Dropout (CNN)	$p = \{0.10,\ 0.15,\ 0.20,\ 0.25,\ 0.30\}$
DropConnect	None
BatchNorm	None
Learning rate scheduler	{Step size = 1, gamma = 0.99}

After training, node-level predictions are extracted for a stratified subset of 5 000 *test* samples (as discussed in Section 4.4) using the our construction of nodes as subpredictors (Section 3.3.1). These predictions form the basis for all diversity calculations following the procedures outlined in Section 4.3.3.

Diversity measure calculations: Diversity is calculated using the same procedures discussed in Section 4.3.3. Only the chosen representative diversity measures discussed in Section 5.2.3 are included in the analysis.

6.3.2 Dropout effects on performance

Effect of Dropout on layer performance: Figure 6.1 shows the effect of applying Dropout during training on layer performance across all datasets. In general, increasing the Dropout probability leads to improved layer accuracy, although this trend varies somewhat by dataset and layer.

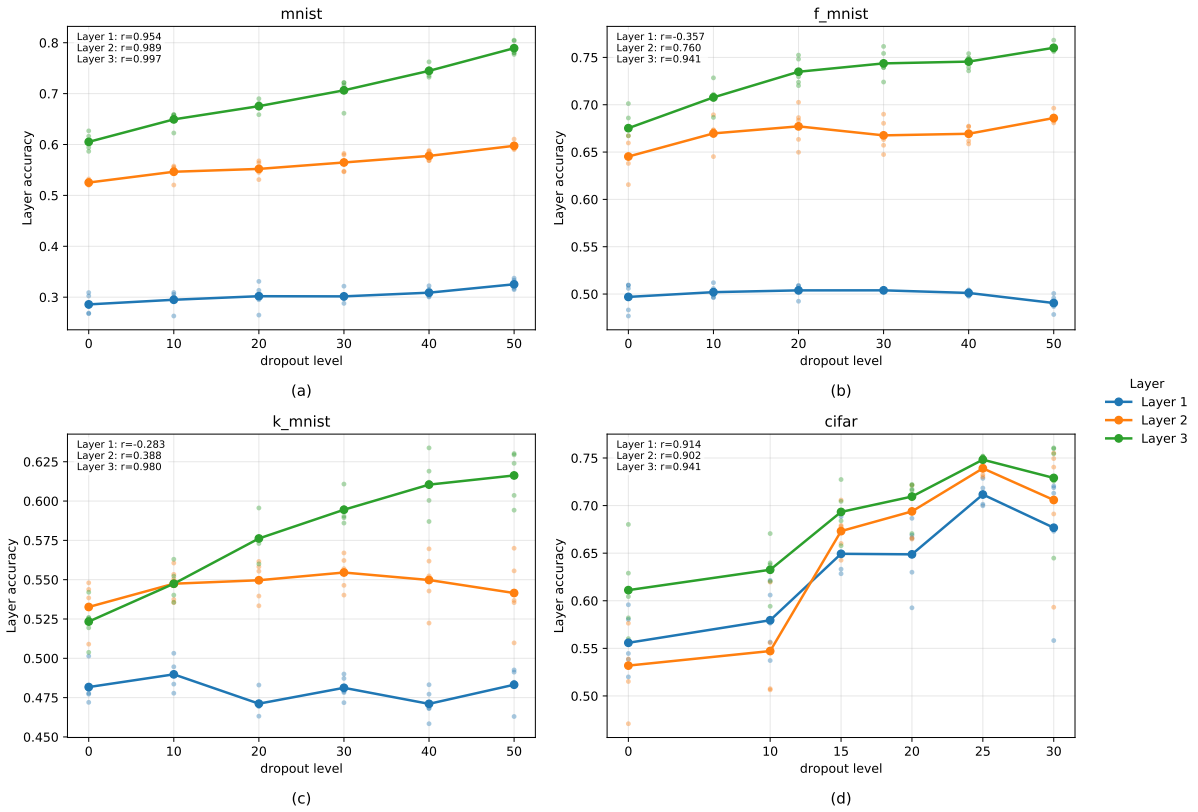


Figure 6.1: Layer test accuracy versus Dropout probability across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Accuracy generally improves as Dropout probability is increased.

For MNIST, a strong positive correlation is observed between Dropout level and layer accuracy across all three layers, with Pearson correlation coefficients of $r = 0.954$, 0.989, and 0.997 for Layer 1–3, respectively. CIFAR-10 follows a similar pattern, showing consistently positive correlations ($r = 0.914$, 0.902, and 0.941), with a slight dip in accuracy at $p = 0.30$.

In contrast, K-MNIST and Fashion-MNIST show less consistent behaviour. In both cases, the first layer exhibits a weak negative correlation with Dropout ($r = -0.283$ for K-MNIST and $r = -0.357$ for Fashion-MNIST). The second layer show weaker positive correlations ($r = 0.388$ and 0.760, respectively), and the third layers display a strong positive correlation.

These results suggest that, overall, applying higher levels of Dropout tends to improve layer performance, particularly in deeper layers.

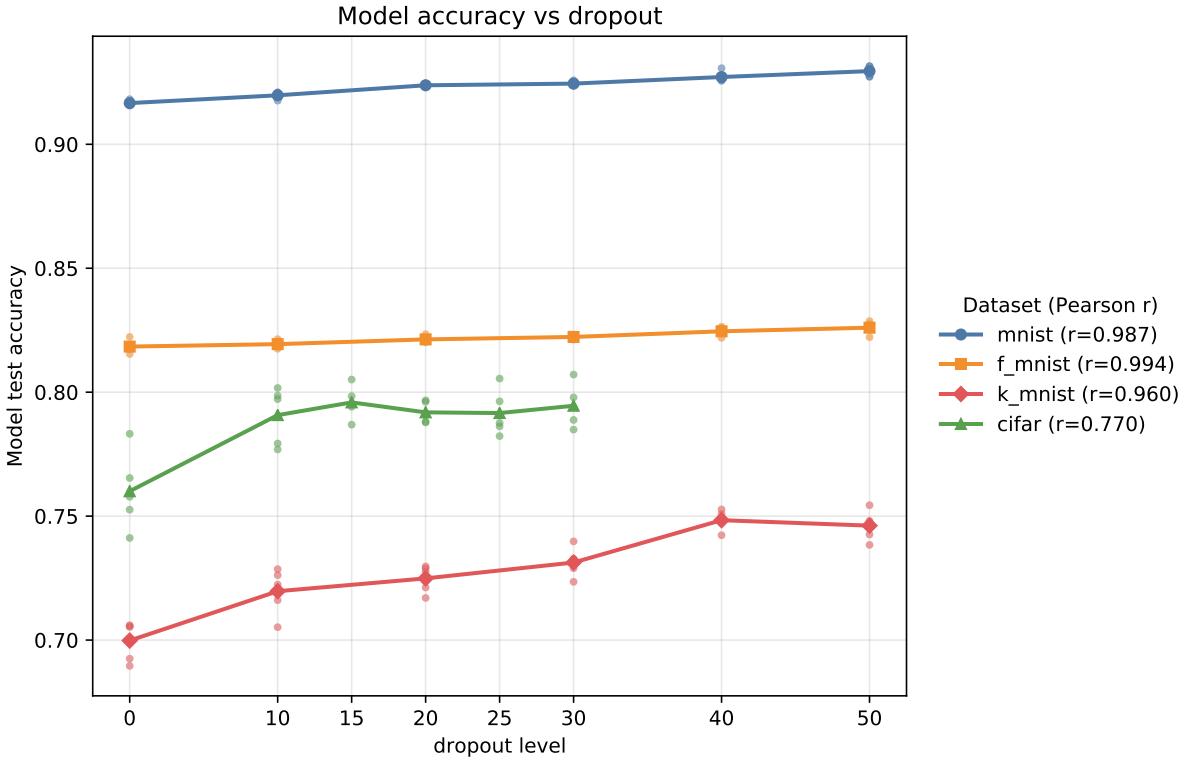


Figure 6.2: Model test accuracy versus Dropout probability across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Accuracy generally improves as Dropout probability is increased.

Effect of Dropout on model performance: As shown in Figure 6.2, increasing the Dropout probability generally leads to improved model test accuracy across all datasets, though the strength of this improvement differ between datasets.

For MNIST and Fashion-MNIST, the relationship is clear and monotonic. For MNIST we observe an increase in accuracy from 91.7% with no Dropout to 93% at $p = 0.50$. Fashion-MNIST follows a similar pattern, improving from 81.8% to 82.6% as Dropout is increased. K-MNIST displays a comparable upward trend up to $p = 0.40$, after which performance slightly declines at $p = 0.50$. CIFAR-10, by contrast, shows an initial improvement with Dropout, but then fluctuates across the remaining range of probabilities.

Pearson correlation coefficients further quantify these trends: $r = 0.987$ for MNIST, 0.994 for Fashion-MNIST, 0.960 for K-MNIST, and slightly weaker 0.770 for CIFAR-10. These results show that Dropout, when applied appropriately, acts as a strong regulariser that enhances generalisation in both MLP and CNN models, although the magnitude of the effect can depend on the dataset and model architecture.

Interpretation: The results show that introducing Dropout during training generally improves both model- and layer-level performance, though the extent of improvement varies across datasets and layers. For MNIST and CIFAR-10, the effect of Dropout is consistently positive across all layers and the overall model. In contrast, K-MNIST and Fashion-MNIST display some differences, with weaker or even negative relationships in the early layers, yet improved performance in the deeper layers and at the model level. This discrepancy may reflect differences in co-adaptation dynamics within early network layers.

Overall, these findings show that Dropout has a beneficial effect on both the layer- and model-level performance in the networks.

6.3.3 Dropout effects on diversity

Effect of Dropout on *Disagreement* and *Double-fault* (*DF*): As the Dropout probability increases, we observe a consistent increase in diversity as measured by both *Disagreement* and *DF* across all datasets (Figure 6.3*(a-b)*). The two metrics display similar trends in diversity: *Disagreement* values rise, while *DF* values decline, indicating a greater degree of independence among node predictions.

For MNIST, Fashion-MNIST, and K-MNIST, this relationship holds across the entire range of Dropout probabilities, with diversity increasing monotonically as p is increased. This is further supported by the strong correlations between averaged diversity and increased p for each of these datasets. In the case of CIFAR-10, diversity initially increases, dips slightly at $p = 0.20$, then rises again at $p = 0.25$. A noticeable decline occurs at

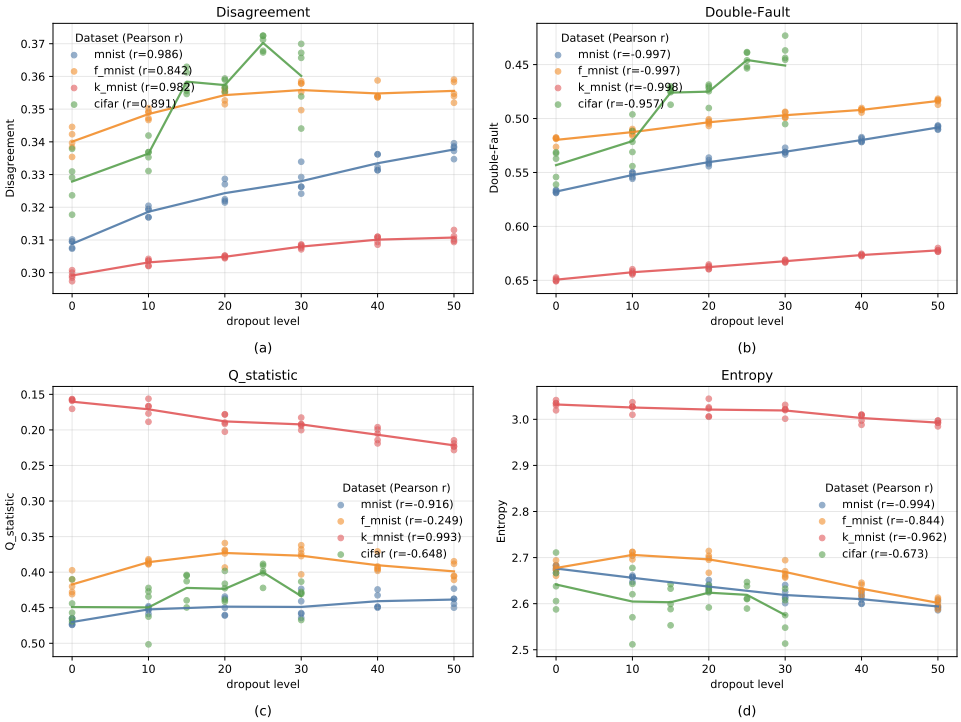


Figure 6.3: Averaged model diversity (averaged over layers) versus dropout probability across all datasets. Each line in the plot represents the mean diversity, calculated across five independently trained seeds—each shown as a scatter point. Diversity generally improves for most diversity measures as dropout probability is increased. Note that for *Double-fault* and *Q-statistic*, we have inverted the y-axis to indicate increasing diversity.

$p = 0.30$, although this point also shows high variability, as reflected by the large standard deviation. Despite these fluctuations, CIFAR-10 still displays a strong correlation between diversity (*Disagreement* and *DF*) and the level of Dropout applied.

Overall, the results suggest that increasing Dropout tends to increase the diversity of node-level predictions as measured by *Disagreement* and *DF*, supporting the idea that Dropout encourages more independent behaviour among subpredictors within a single neural network.

Effect of Dropout on *Q-statistic*: The *Q-statistic* displays inconsistent behaviour across datasets, particularly in the cases of Fashion-MNIST and K-MNIST (Figure 6.3*(c)*). For MNIST and CIFAR-10, we observe a generally increasing trend in diversity—reflected

by a decrease in *Q-statistic* values (noting that the y-axis is inverted). These trends are supported by the moderate to strong negative correlations between Dropout probability and *Q-statistic*: $r = -0.916$ for MNIST, and $r = -0.648$ for CIFAR-10.

Fashion-MNIST follows the same general direction, but with a much weaker correlation ($r = -0.249$), suggesting a limited relationship. In contrast, K-MNIST displays a strong *positive* correlation, indicating an opposite trend in which increased Dropout is associated with decreased diversity according to the *Q-statistic*.

While three of the four datasets (MNIST, Fashion-MNIST, and CIFAR-10) maintain a negative relationship between Dropout and the *Q-statistic*, the inconsistency in correlation strength—and the reversal seen for K-MNIST—suggests that this metric may be unreliable for drawing broad conclusions about how Dropout affects diversity.

Effect of Dropout on *Entropy*: Figure 6.3*(d)* shows how *Entropy* changes with increasing Dropout probability. Across all datasets, we observe a general *decrease* in *Entropy*—indicating a reduction in uncertainty—as Dropout is increased. This trend is supported by *negative* correlations across all datasets: $r = -0.994$ for MNIST, -0.844 for Fashion-MNIST, -0.962 for K-MNIST, and -0.673 for CIFAR-10.

These findings suggest that with higher levels of Dropout, individual nodes tend to produce less uncertain outputs. In this sense, while Dropout increases diversity from a disagreement and error perspective, it appears to reduce diversity when measured by output uncertainty.

Correlation of *Disagreement* and *Double-fault* metrics with model performance across Dropout rates: Figure 6.4*(a-b)* illustrates how diversity—measured via *Disagreement* and *DF*—relates to model accuracy as Dropout probability is increased. Across all datasets, we observe moderate to strong correlations between *Disagreement* and model performance, ranging from $r = 0.642$ for CIFAR-10, to $r = 0.705$ for Fashion-MNIST, $r = 0.916$ for MNIST, and $r = 0.932$ for K-MNIST. A similar pattern is observed

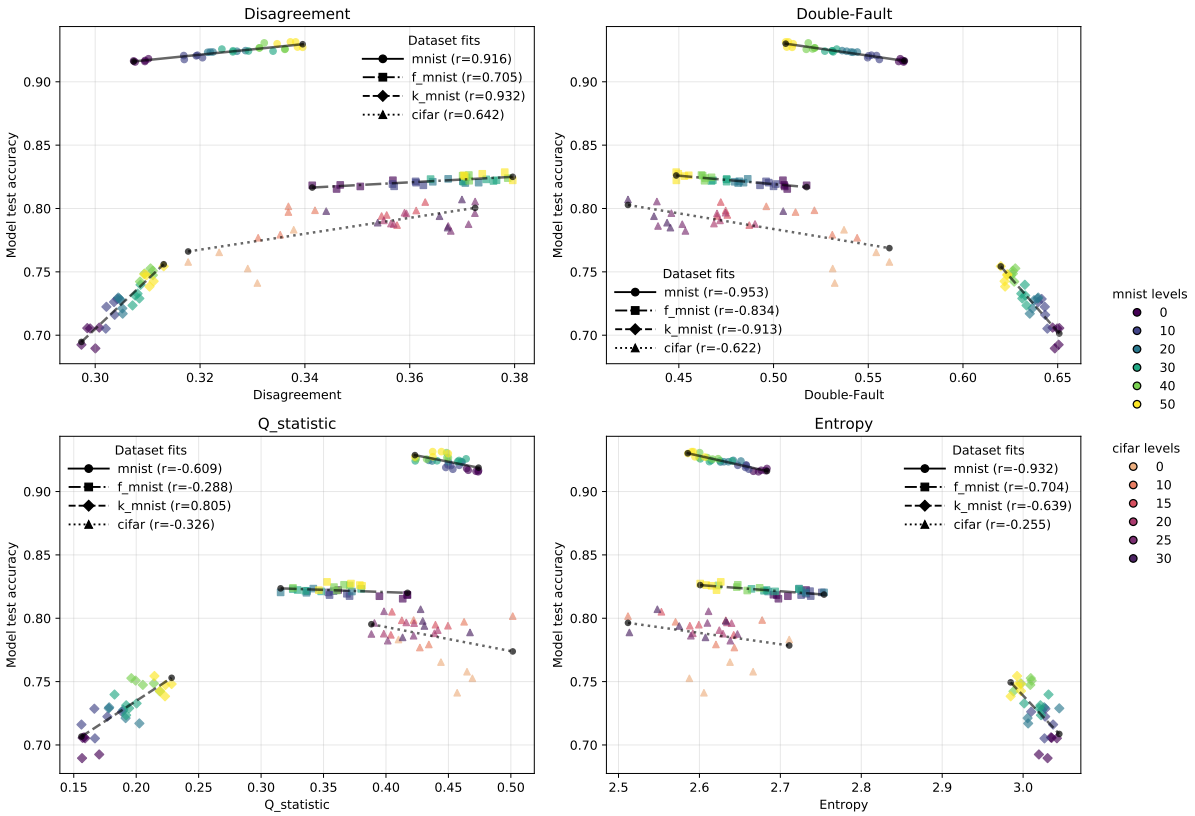


Figure 6.4: Correlation between model test accuracy and averaged diversity across datasets and metrics under induced diversity conditions using Dropout.

for the *DF* measure, though the correlations are negative by construction, reflecting that lower *DF* values correspond to higher diversity. These results indicate that higher model accuracy is generally associated with greater subpredictor independence—higher *Disagreement* and lower simultaneous failure.

Taken together, the findings reinforce the idea that increased Dropout not only improves model generalisation, but also promotes diversity within the network—at least in the context of *Disagreement* and *DF*.

Relationship between *Q-statistic* and model performance across Dropout rates: In line with earlier observations (see Section 6.3.3 on the effect of Dropout on *Q-statistic*), the *Q-statistic* shows inconsistent behaviour across datasets when examining its relationship with model performance under varying Dropout rates (Figure 6.4*(c)*). Although

MNIST, Fashion-MNIST, and CIFAR-10 all exhibit negative correlations between *Q-statistic* values and model accuracy—as would be expected if increased diversity supports better generalisation—these correlations are relatively weak, with the strongest being $r = -0.609$ for MNIST.

Notably, K-MNIST deviates again from this pattern, displaying a strong *positive* correlation between *Q-statistic* and model accuracy—directly contradicting the anticipated trend. This inconsistency across datasets, particularly the reversal seen in K-MNIST, suggests that the *Q-statistic* may be less reliable or may even be dataset-specific in its results. Compared to other metrics like *Disagreement* or *DF*, the *Q-statistic* appears to offer less interpretable and less consistent insights into how diversity relates to model performance when Dropout is applied.

Relationship between *Entropy* and model performance across Dropout rates: *Entropy* shows a consistent inverse relationship with model performance as Dropout probability increases, though the strength of this relationship varies across datasets. For MNIST and Fashion-MNIST, the correlations are strong at $r = -0.932$ and $r = -0.704$, respectively, while K-MNIST shows a moderately strong correlation at $r = -0.639$ (see Figure 6.4*(d)*). In contrast, CIFAR-10 exhibits only a weak relationship, with a correlation of $r = -0.255$.

These results still suggests that as model accuracy improves, the uncertainty in node-level predictions tends to decrease. This aligns with the interpretation of *Entropy* as capturing the confidence or sharpness of class probability distributions, rather than inter-node disagreement. Compared to *Disagreement* or *DF*, *Entropy* reflects a different perspective of diversity—one rooted in uncertainty of individual subpredictors' collective consensus rather than their disagreement with one another.

Interpretation The results across all four diversity measures show that Dropout has a measurable and often consistent impact on the diversity of node-level predictions in single neural networks. *Disagreement* and *DF* both exhibit reliable trends across datasets:

as Dropout probability increases, node predictions become more diverse, with higher *Disagreement* and lower *DF* values indicating greater independence among subpredictors. These measures also correlate well with improved model performance, reinforcing the view that diversity—when understood as reduced agreement or co-failure among nodes—benefits generalisation.

In contrast to these findings, *Entropy* suggests an alternative interpretation. While model performance increases with Dropout, *Entropy* decreases, suggesting that subpredictors are becoming less uncertain in their class predictions. This reflects a complementary aspect of diversity: while nodes disagree more with each other (as captured by *Disagreement*), their individual predictions become less uncertain (lower *Entropy*). This dual behaviour indicates that diversity is multi-faceted, and different measures capture different characteristics—some related to inter-node variation, others to prediction uncertainty.

The *Q-statistic* however, remains the least stable of the four. Despite showing expected trends for some datasets, its inconsistent behaviour—especially the reversed trend on K-MNIST—raises questions about its reliability in this context. It appears sensitive to dataset-specific dynamics and may not be as suited for studying diversity under induced regularisation as the other measures.

Together, these findings suggest that Dropout does not merely regularise networks in the traditional sense, but actively impacts the internal diversity within a single neural network. This induced diversity, particularly captured by *Disagreement* and *DF*, appears strongly connected to improved model accuracy.

6.3.4 Summary

In this section, we investigated how Dropout affects performance and diversity within a single neural network. Using both MLP and CNN architectures across four datasets, we systematically varied Dropout probability and evaluated its impact on both model- and layer-level behaviour.

From a performance perspective, we observed that increasing Dropout generally improves model generalisation across datasets, with the strongest and most consistent gains seen in MNIST and Fashion-MNIST. Layer-level accuracy also improved with higher Dropout rates, especially in deeper layers, though the first layers of K-MNIST and Fashion-MNIST showed weaker or negative correlations.

From a diversity standpoint, Dropout consistently increased diversity as measured by *Disagreement* and *DF*, both of which were also strongly correlated with model accuracy (*Disagreement* with a positive correlation, and *DF* with a negative correlation). *Entropy* showed the opposite trend, decreasing as Dropout increased, suggesting a drop in prediction uncertainty even as the disagreement among nodes rose. The *Q-statistic* showed the least stable behaviour, with inconsistent or contradictory trends—especially for K-MNIST—making it less reliable for interpreting diversity in this context. Future work will focus on an in-depth mechanistic investigation to determine the precise cause for the divergent trends observed between the *Q-statistic* and *Entropy*, thereby validating or refuting our current conjectures regarding their distinct relationships with model performance.

Finally, these findings confirm that Dropout not only improves model performance, but also induces meaningful internal diversity within the network. This diversity, particularly when captured through disagreement- and co-failure based measures, appears to contribute directly to generalisation gains.

6.4 DropConnect

6.4.1 Experimental setup:

Model architectures: This section employs the same model architectures used in Section 6.3.1, with the key difference being the replacement of Dropout with DropConnect. The MLPs remain unchanged except for using DropConnect as regulariser. In the CNN models, DropConnect is applied after each convolutional block, as detailed in Table 6.3.

Model training protocol, hyperparameter setup, and analysis: The training protocol and evaluation methodology follow those described in Section 6.3.1. However, in this experiment, DropConnect is applied instead of Dropout, using the same set of drop probabilities. No other regularisation methods, such as BatchNorm or additional Dropout, are used. Table 6.4 summarises the hyperparameters used throughout this section.

Table 6.4: Summary of hyperparameters for MLP and CNN architectures. This configuration uses a batch size of 256 with DropConnect as the only regularisation mechanism applied.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	0.0003
Batch size	256
Epochs (max)	1 000
DropConnect (MLP)	$p = \{0.10,\ 0.20,\ 0.30,\ 0.40,\ 0.50\}$
DropConnect (CNN)	$p = \{0.10,\ 0.15,\ 0.20,\ 0.25,\ 0.30\}$
Dropout	None
BatchNorm	None
Learning rate scheduler	{Step size = 1, gamma = 0.99}

6.4.2 DropConnect effects on performance

Effect of DropConnect on layer performance: Figure 6.5 shows the effect of applying DropConnect during training on the layer performance across all datasets. In general, increasing the DropConnect probability leads to improved layer accuracy, although we do see a discrepancy in the first layer trend of K-MNIST.

For MNIST, a strong positive correlation is observed across all three layers, with correlations of $r = 0.984$, 0.994, and 1.000 for Layer 1–3, respectively. CIFAR-10 and Fashion-MNIST follows a similar trend, also showing consistently positive correlations (for CIFAR-10: $r = 0.754$, 0.789, and 0.841 and for Fashion-MNIST: $r = 0.830$, 0.984, and 0.985), although CIFAR-10 does exhibit a slight decline in layer accuracy between $p = 0.20$ and $p = 0.25$.

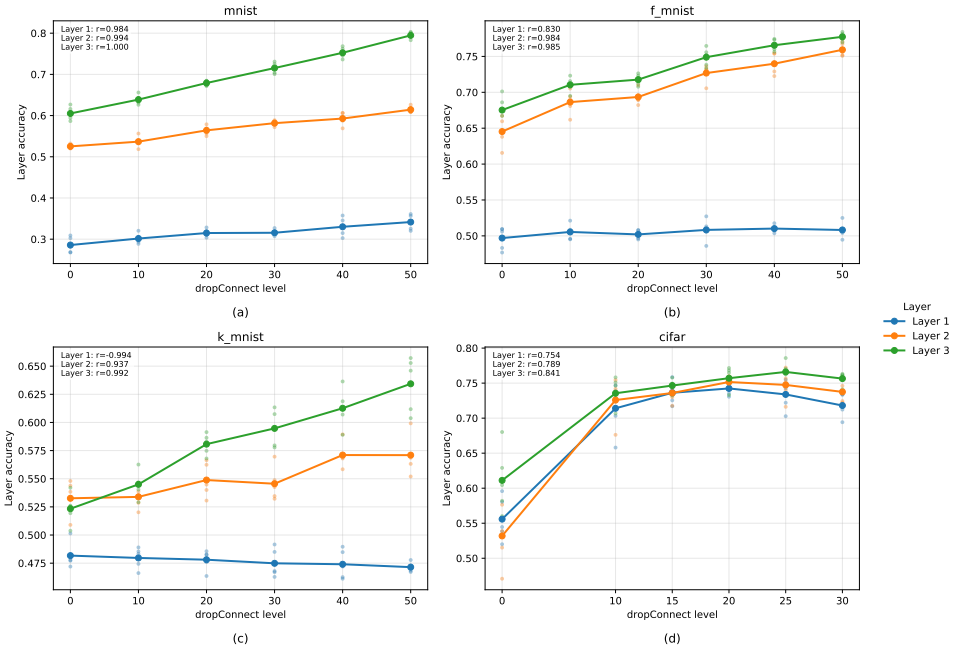


Figure 6.5: Layer test accuracy versus DropConnect probability across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Accuracy generally improves as DropConnect probability is increased.

K-MNIST has contrasting results for the first layer, with a strong *negative* correlation ($r = -0.994$) between layer accuracy and DropConnect probability. Despite this deviation, our results suggest that, in general, higher levels of DropConnect being applied, leads to improved layer performance—with the deeper layers having more pronounced improvements.

Effect of DropConnect on model performance: In Figure 6.6 shows the effect of applying DropConnect during training on model performance across all datasets. In every case—MNIST, Fashion-MNIST, K-MNIST, and CIFAR-10—increasing the DropConnect probability leads to consistent and monotonic improvements in test accuracy—at least up to $p = 0.25$.

Specifically, model accuracy improves from 91.6% to 93.5% on MNIST, from 81.8% to 83.1% on Fashion-MNIST, from 70.0% to 74.9% on K-MNIST, and from 76.0% to 79.1%

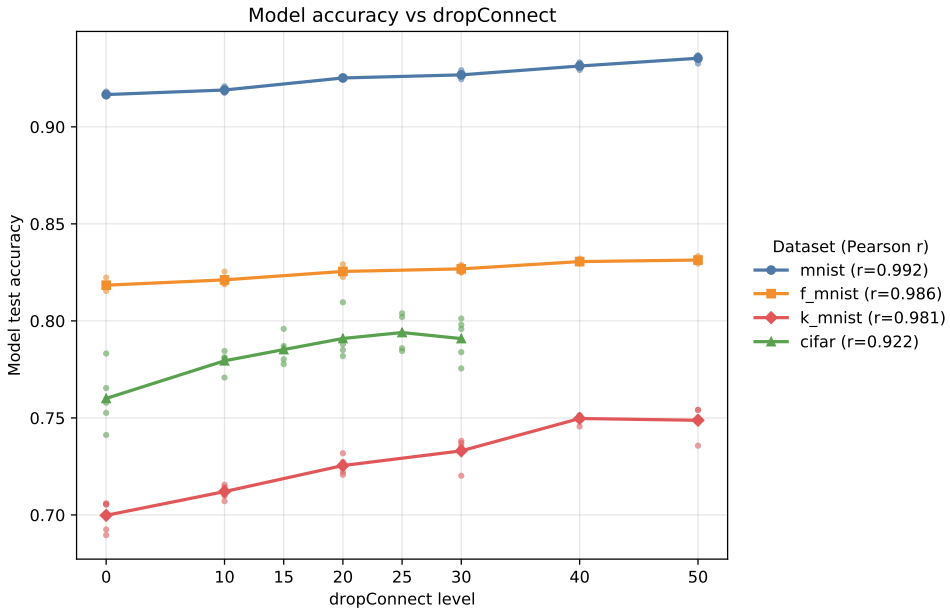


Figure 6.6: Model test accuracy versus DropConnect probability across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Accuracy generally improves as DropConnect probability is increased.

on CIFAR-10 (with the highest accuracy for CIFAR-10 (79.4%) at $p = 0.25$). These improvements are supported by the corresponding Pearson correlation coefficients that are all strongly positive with $r > 0.9$ in all cases.

These results indicate that DropConnect acts as an effective regulariser, improving generalisation across both MLP and CNN architectures.

Discussion of DropConnect effects on performance: The results indicate that applying DropConnect during training improves not only *model* accuracy but also improves the accuracy of individual layers. MNIST and Fashion-MNIST exhibit consistently strong positive correlations between DropConnect probability and layer accuracy for all layers, while CIFAR-10 shows similarly positive, though slightly weaker, trends.

An exception to this general trend appears in the first layer of K-MNIST, where accuracy decreases significantly as DropConnect increases, showing a strong negative correlation.

Despite this outlier, the remaining K-MNIST layers show increasing performance, aligning with the broader pattern.

Overall, these findings support the view that DropConnect, like Dropout, improves both model and internal layer performance by encouraging more independent learning across nodes.

6.4.3 DropConnect effects on diversity

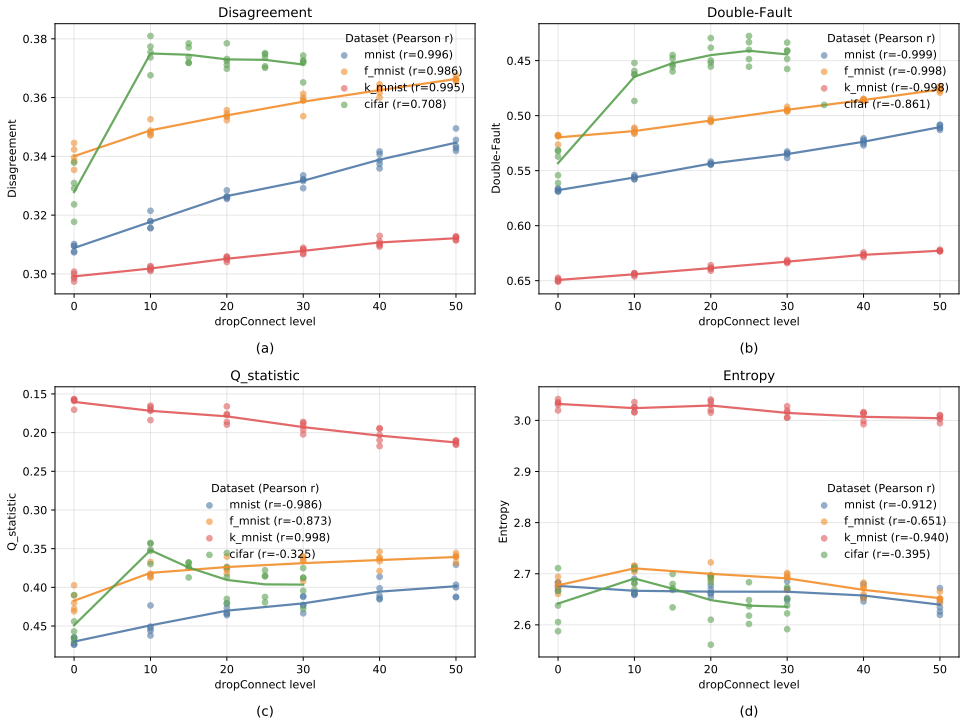


Figure 6.7: Averaged model diversity versus DropConnect probability across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Diversity generally improves for most diversity measures as DropConnect probability is increased. Note that for *Double-fault* and *Q-statistic*, we have inverted the y-axis to indicate increasing diversity.

Effect of DropConnect on *Disagreement*: As the DropConnect probability increases, we observe a consistent and monotonic increase in diversity (*Disagreement*) across

MNIST, Fashion-MNIST, and K-MNIST with correlations of $r = 0.996$, 0.986, and 0.995, respectively (Figure 6.7*(a)*). CIFAR-10 exhibits a more complex pattern: *Disagreement* increases sharply from the baseline to $p = 0.10$, but then steadily declines as the DropConnect probability continues to increase. Despite this decline, the overall correlation remains strongly positive at $r = 0.708$.

These results indicate that DropConnect typically increases *Disagreement*, supporting the view that it promotes independence among nodes.

Effect of DropConnect on *Double-fault* (*DF*): *DF* values decrease consistently with increasing DropConnect across all datasets, reflecting a rise in diversity as subpredictors fail less frequently on the same inputs (Figure 6.7*(b)*). This trend is especially strong for MNIST, Fashion-MNIST, and K-MNIST, with nearly perfect negative correlations of $r = -0.999$, -0.998, -0.998, respectively. CIFAR-10 follows a similar pattern, albeit with a slightly weaker correlation of $r = -0.861$.

These strong negative correlations suggest that DropConnect reliably reduces co-failure among subpredictors. This supports the interpretation that DropConnect encourages more independent learning within the network.

Effect of DropConnect on *Q-statistic*: As with Dropout, the *Q-statistic* exhibits inconsistent trends in response to DropConnect across datasets (Figure 6.7*(c)*). For MNIST and Fashion-MNIST, we observe strong negative correlations between *Q-statistic* and DropConnect probability ($r = -0.986$ and $r = -0.873$), respectively), indicating an increase in diversity. CIFAR-10 also follows this trend but with much weaker correlation ($r = -0.325$), suggesting only a modest connection between DropConnect and inter-node agreement.

In contrast, K-MNIST again deviates significantly, showing a strong *positive* correlation of $r = 0.998$, implying that DropConnect actually increases agreement among subpredictors—i.e., reduces diversity—on this dataset. This mirrors the reversed trend observed in the

Q-statistic under Dropout and raises further questions about the metric's dataset sensitivity. While some datasets follow the expected pattern, the inconsistency limits the *Q-statistic*'s reliability for assessing diversity in this context.

Effect of DropConnect on *Entropy*: Figure 6.7*(d)* illustrates how *Entropy* changes as DropConnect is applied. Across all datasets, we observe a general *decrease* in *Entropy*—indicating a reduction in uncertainty—as DropConnect is increased. This trend is supported by *negative* correlations across all datasets: $r = -0.912$ for MNIST, -0.651 for Fashion-MNIST, -0.940 for K-MNIST, and -0.395 for CIFAR-10.

Although the strength of these correlations varies, the consistent direction across datasets suggests that DropConnect leads to less uncertain node-level predictions. This aligns with earlier findings from Dropout: while diversity increases when measured by *Disagreement* or *DF*, it decreases when quantified via prediction uncertainty. Thus, we believe that *Entropy* captures a different, complementary aspect of diversity.

Correlation of *Disagreement* and *Double-fault* metrics with model performance across DropConnect rates: Figure 6.8*(a-b)* illustrates how *Disagreement* and *DF* relate to model accuracy across DropConnect levels. In all datasets, we observe moderate to strong positive correlations between *Disagreement* and accuracy: $r = 0.959$ for MNIST, $r = 0.907$ for Fashion-MNIST, $r = 0.944$ for K-MNIST, and $r = 0.714$ for CIFAR-10. *DF* shows a mirrored pattern, with strong negative correlations due to its inverse formulation.

These results reinforce the interpretation that DropConnect leads to increased subpredictor independence, leading to both improved generalisation and increased diversity. *Disagreement* and *DF* remain reliable indicators in this setting, consistently capturing diversity trends that align with improved performance.

Relationship between *Q-statistic* and model performance across DropConnect rates: As with previous observations (see Section 6.3.3 regarding the *Q-statistic* results

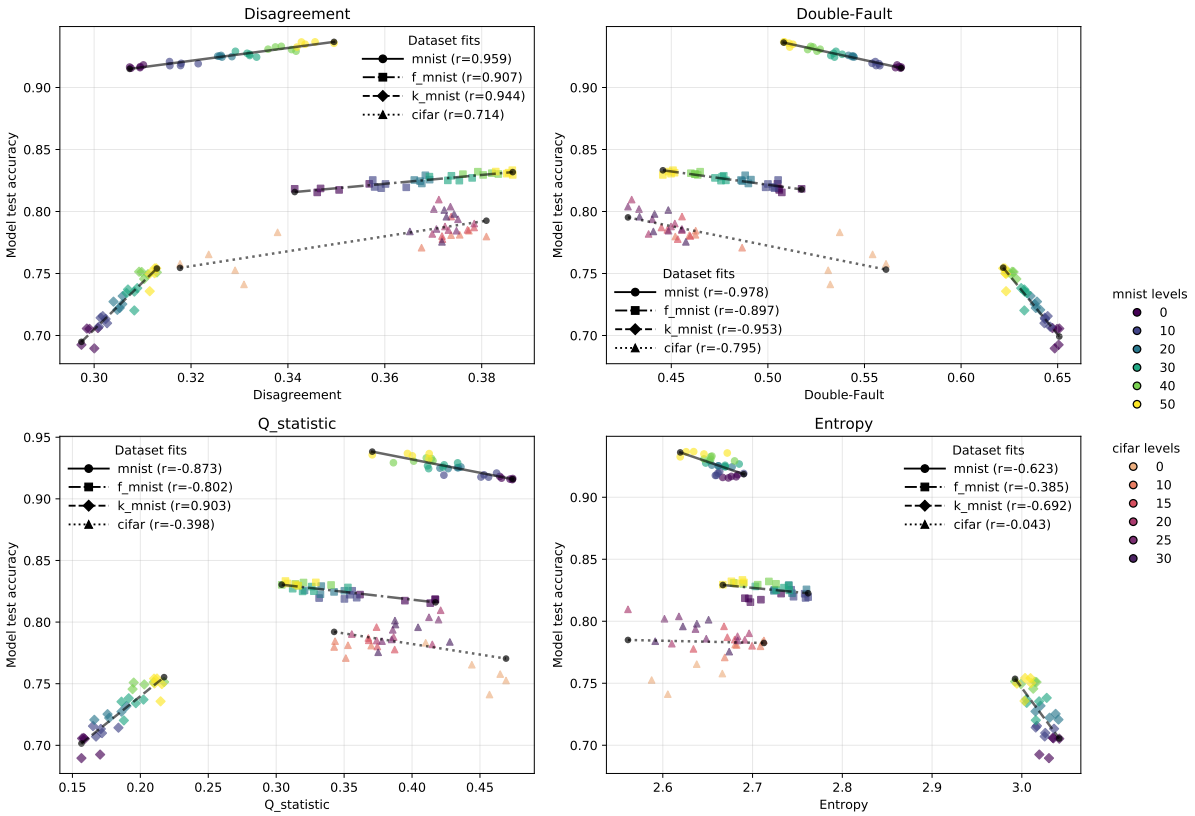


Figure 6.8: Correlation between model test accuracy and averaged diversity across datasets and metrics under induced diversity conditions using DropConnect.

for Dropout, as well as Section 6.4.3's effect of DropConnect on *Q-statistic*), the *Q-statistic* shows inconsistent behaviour across datasets when relating diversity to model performance under varying DropConnect levels (Figure 6.8*(c)*). For MNIST and Fashion-MNIST, the expected negative correlation is observed, and is stronger than in the Dropout case, with correlations of $r = -0.873$ and $r - 0.802$, respectively. CIFAR-10 also exhibits a negative correlation, though much weaker at $r = -0.398$, indicating a less clear relationship.

K-MNIST once again diverges from this pattern, showing a strong *positive* correlation of $r = 0.903$ between *Q-statistic* and model performance. This suggest that, for K-MNIST, increased agreement among subpredictors is correlated with improved generalisation—contrary to what would be expected.

Relationship between *Entropy* and model performance across DropConnect rates: *Entropy* shows a relatively consistent inverse relationship with model performance as DropConnect probability increases, although this relationship is generally weaker than what was observed in the Dropout case. For MNIST and K-MNIST, the correlations are moderately strong at $r = -0.623$ and $r = -0.692$, respectively, while Fashion-MNIST shows only a weak correlation of $r = -0.385$ (see Figure 6.8*(d)*). CIFAR-10, in contrast, displays an almost negligible relationship, with a correlation of $r = -0.043$.

While most datasets follow the negative trend—where improved model accuracy is accompanied by lower *Entropy*—the inconsistency and weakness of the correlations, particularly for CIFAR-10, make it difficult to draw strong conclusions. By generalising the results, it suggests that *Entropy* could still be an indicator of the link between model performance and diversity under DropConnect, though a less reliable one.

Discussion on DropConnect effects on diversity: The results demonstrate that DropConnect consistently influences diversity within neural networks, although the nature of this effect depends on the specific diversity metric and dataset. Both *Disagreement* and *DF* exhibit reliable and interpretable trends across all datasets. As DropConnect probability increases, *Disagreement* rises and *DF* decreases, indicating improved subpredictor independence and reduced co-failure among nodes. These trends are strongly correlated with model performance, suggesting that increased diversity—understood through these metrics—is beneficial for generalisation.

Entropy, which captures predictive uncertainty rather than inter-node variation, exhibits a consistent but weaker inverse relationship with DropConnect (as compared to Dropout). As DropConnect is increased, *Entropy* tends to decrease, suggesting that nodes become less uncertain in their outputs. While this reduction in uncertainty may seem at odds with rising diversity from other metrics, it likely reflects a complementary perspective: nodes become more certain, but in different and less correlated ways.

In contrast, *Q-statistic* continues to show inconsistent and dataset-specific behaviour. Although MNIST, Fashion-MNIST, and CIFAR-10 show the expected negative relationship

between *Q-statistic* and both DropConnect and model accuracy, K-MNIST displays a strong positive trend, suggesting increasing agreement among subpredictors with higher DropConnect.

6.4.4 Summary

This section investigated the effect of DropConnect on model performance and diversity within single neural networks. Using both MLP and CNN architectures, we applied DropConnect at varying probabilities across four datasets to assess its role as a regularisation method and its effects on diversity.

Overall, DropConnect improved model accuracy consistently. Performance gains were seen across all datasets, with notable improvements in layer accuracies, especially in the deeper layers where DropConnect's regularisation appears most beneficial. The only exception to this trend was the first layer of K-MNIST models which showed a decline in accuracy.

Looking at the diversity, DropConnect reliably increased *Disagreement* and decreased *DF* values across datasets, indicating greater independence between subpredictors and fewer shared predictions errors. These two metrics were also strongly correlated with model accuracy, reinforcing their usefulness as diversity indicators under this induced regularisation setting. *Entropy*, which measures prediction uncertainty, consistently decreased with higher DropConnect, suggesting the subpredictors became more certain in their outputs—even as they remained diverse in terms of error and disagreement. The *Q-statistic*, however, showed inconsistent behaviour, particularly in the K-MNIST datasets, making it a less reliable measure for interpreting diversity trends in this context.

In summary, DropConnect not only improves generalisation, but also meaningfully influences the internal diversity of a network. *Disagreement* and *DF* emerge as reliable and interpretable indicators of this diversity, while *Entropy* and *Q-statistic* seem to reflect some dataset-dependent nuances. These results further confirm that specific regularisation techniques can be used to modulate and improve diversity in single neural networks,

viewed as implicit ensembles.

6.5 Batch size

6.5.1 Experimental setup:

Model architectures: This section uses the same model architectures described in Section 5.2.1. For CNN models, the architecture configuration corresponds to the specification outlined in Table 4.1.

Training protocol, hyperparameter configuration, and analysis: The training and evaluation procedures are consistent with those described in Section 5.2.1. In this experiment, we vary two aspects of the training configuration: the batch size and, for CIFAR-10, the effective size of the training set. Batch size is treated as the sole regularisation technique applied, with no additional techniques such as Batch normalisation, Dropout, or DropConnect applied. The full set of hyperparameters is summarised in Table 6.5.

Due to time constraints, especially at smaller batch sizes requiring more parameter updates, we used a stratified subset of 8 000 training samples for CIFAR-10 instead of the full dataset. This adjustment ensures training remains tractable while mostly preserving class balance.

6.5.2 Batch size effects on performance

Effect of varying batch size on layer performance: Figure 6.9 illustrates how training with different batch sizes affects layer performance across all datasets. Overall, increasing the batch size tends to reduce layer accuracy, though the strength and direction of this trend vary by dataset and layer.

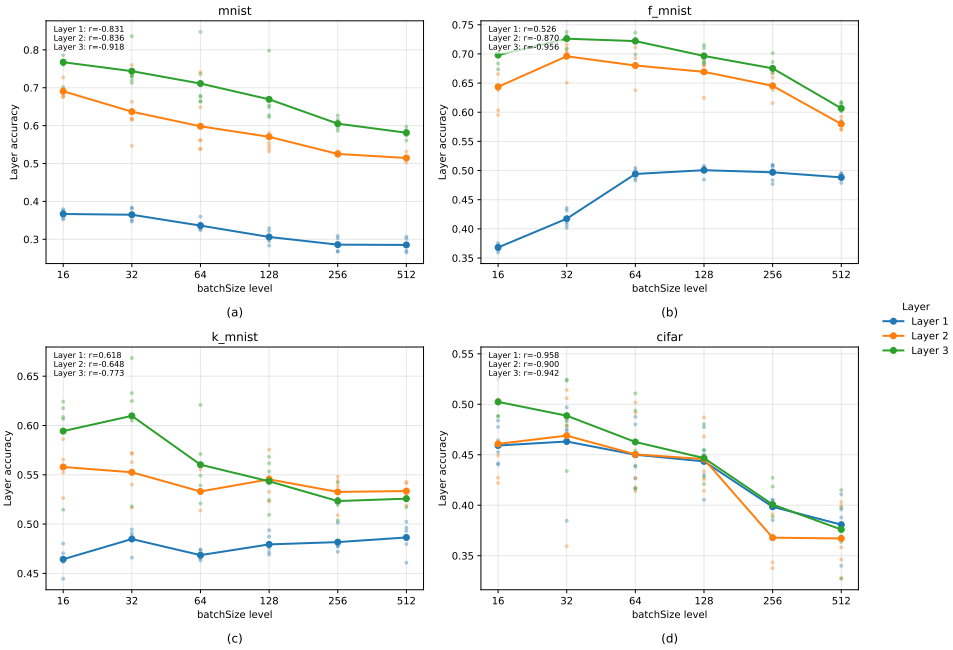


Figure 6.9: Layer test accuracy versus batch size across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Accuracy generally improves as batch size is reduced.

For MNIST and CIFAR-10, there is a clear and consistent decline in accuracy with larger batch sizes across all three layers, with correlations of $r = -0.831$, -0.836, and -0.918 for MNIST Layers 1-3, and $r = -0.958$, -0.900, and -0.942 for CIFAR, respectively.

In contrast, Fashion-MNIST and K-MNIST exhibit mixed behaviour. For both datasets, the first layer shows a moderate positive correlation with larger batch size ($r = 0.526$ for Fashion-MNIST and $r = 0.618$ for K-MNIST). However, deeper layers follow the general downward trend: the second layers show moderate to strong negative correlations ($r = -0.870$ and $r = -0.648$, respectively), and the third layers continue with strong negative relationships.

The findings indicate that larger batch sizes are generally related to a degradation in layer performance—especially in deeper layers. Performance in the initial layers, however, displays mixed or inconsistent behaviour.

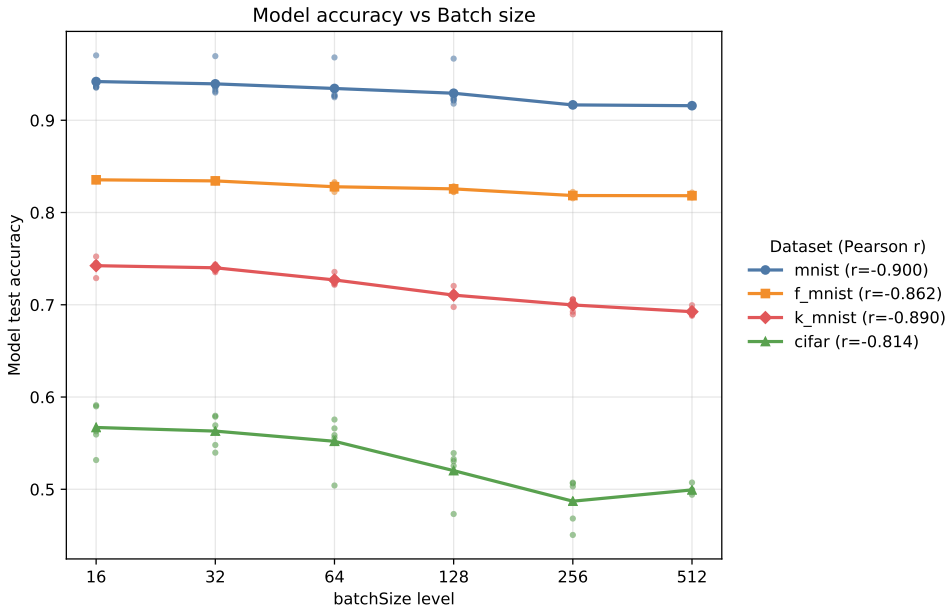


Figure 6.10: Model test accuracy versus batch size across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Accuracy generally improves as batch size is reduced.

Effect of varying batch size on model performance: In general, increasing the batch size leads to a decline in test accuracy. This is shown in Figure 6.10, which illustrates how batch size influences model performance across all datasets. For MNIST, Fashion-MNIST, and K-MNIST, the relationship is clear and monotonic. Accuracy for MNIST drops from 94.2% with a batch size of 16 to 91.6% at 512. Similarly, Fashion-MNIST decreases from 83.5% to 81.8%, while K-MNIST decreases from 74.2% to 69.3%. CIFAR-10 also shows a downward trend in accuracy with increasing batch size, although a slight rebound is observed at 512.

The observed relationships are supported by strong negative correlations across all datasets, ranging from $r = -0.814$ to $r = -0.900$. These results support the interpretation that smaller batch sizes improve generalisation performance.

Interpretation: The results demonstrate that batch size has a meaningful impact on both layer and model performance in a single neural network. Smaller batch sizes con-

sistently yield better accuracy across all datasets, especially better layer accuracy in the deeper layers. This improvement in the model's ability to generalise could potentially be due to the noisier gradient estimates during training (see Section 6.2.3). While some variability is observed in the first layers on Fashion-MNIST and K-MNIST—where performance seems to *improve* with larger batches—the deeper layers conform to the general trend of declining accuracy as batch size increases. At the model level, we observe the same inverse relationship: smaller batches improve accuracy consistently across all datasets. These patterns suggests that batch size is not only a parameter that helps with training efficiency, but also a significant regularisation factor that influences the generalisation performance of a model.

6.5.3 Batch size effects on diversity

Effect of varying batch size on *Disagreement* and *Double-fault* (*DF*): As batch size increases, we observe a general reduction in diversity as measured by both *Disagreement* and *DF* across most datasets (Figure 6.11*(a-b)*). For MNIST, K-MNIST, and CIFAR-10, *Disagreement* values decline with larger batch sizes, supported by strong negative correlations: $r = -0.892$, $r = -0.882$, and $r = -0.962$, respectively.

Fashion-MNIST exhibits a less straightforward pattern. Although the overall correlation is weakly negative ($r - 0.288$), visual inspection reveals a non-monotonic trend—*Disagreement* peaks at intermediate batch sizes and decreases at both ends, forming a mild arc-shaped pattern.

The *DF* metric mirrors these findings. MNIST, K-MNIST, and CIFAR-10 again show strong positive correlations with batch size, consistent with reduced diversity. Fashion-MNIST displays a similar but weaker trend compared to the other datasets.

These trends indicate that subpredictors generally become more similar in their predictions under larger batch sizes (lower *Disagreement* and higher *DF* values). The results suggest that increasing batch size typically leads to decreased diversity in subpredictor predictions, introducing the idea that smaller batch sizes may encourage more indepen-

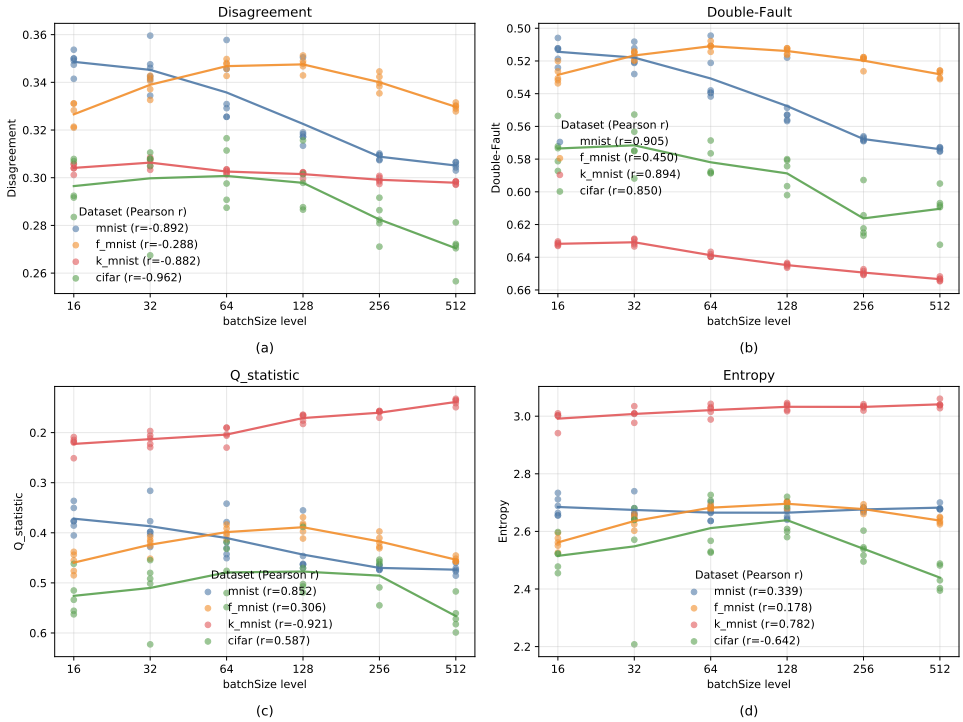


Figure 6.11: Averaged model diversity versus batch size across all datasets. Each line in the plot represents the mean accuracy, calculated across five independently trained seeds—each shown as a scatter point. Diversity generally improves for most diversity measures as batch size is reduced. Note that for *Double-fault* and *Q-statistic*, we have inverted the y-axis to indicate increasing diversity.

dent behaviour among subpredictors as more gradient noise is introduced.

Effect of varying batch size on *Q-statistic*: The *Q-statistic* once again displays inconsistent trends across datasets (see *Q-statistic* results in Sections 6.3.3 and 6.4.3), particularly due to the reversed pattern observed in K-MNIST (see Figure 6.11*(c)*). For MNIST, Fashion-MNIST, and CIFAR-10, we observe positive correlations between *Q-statistic* values and increasing batch size, suggesting reduced diversity with larger batches. However, the strength of these relationships varies considerably: MNIST shows a strong correlation ($r = 0.852$), while Fashion-MNIST and CIFAR-10 only have weak to moderate correlations ($r = 0.306$ and $r = 0.587$, respectively).

K-MNIST presents a strong negative correlation ($r = -0.921$), indicating that diversity—

as measured by the *Q-statistic*—increases with larger batch sizes. This is in direct opposition to the pattern observed in the other datasets, as well as measures such as *Disagreement* and *DF*.

These findings reinforce our earlier concerns about the *Q-statistic*'s reliability as a cross-dataset diversity metric. Its sensitivity to what seems to be dataset-specific dynamics limits its utility for drawing strong conclusions across settings.

Effect of varying batch size on *Entropy*: Figure 6.11*(d)* shows how *Entropy* varies with increasing batch size, revealing a mixed set of trends across datasets. K-MNIST exhibits a strong positive correlation ($r = 0.782$), indicating that subpredictor uncertainty increases as batch size grows. MNIST and Fashion-MNIST also show positive correlations, but these are weak ($r = 0.339$ and 0.178, respectively), suggesting only a mild increase in uncertainty. In contrast, CIFAR-10 displays a moderately strong *negative* correlation, with *Entropy* decreasing as batch size increases.

Taken together, the results suggest that while larger batch sizes may lead to higher uncertainty in some datasets, this effect is neither consistent nor reliable across all datasets. The mixed trends and weak correlations make it difficult to draw a general conclusion about how batch size impacts *Entropy* as a measure of diversity.

Correlation of *Disagreement* and *Double-fault* metrics with model performance across varying batch sizes: Figure 6.12*(a-b)* illustrates the relationship between *Disagreement*, *DF*, and model performance across different batch sizes. Across all datasets, *Disagreement* is positively correlated with model accuracy: $r = 0.844$ for MNIST, $r = 0.557$ for Fashion-MNIST, $r = 0.837$ for K-MNIST, and $r = 0.601$ for CIFAR-10. The *DF* measure shows a mirrored trend, with strong negative correlations reflecting its inverse nature as a diversity metric.

These results reinforce the connection between generalisation and internal diversity: as batch size decreases, both model performance and diversity increase. Among diversity

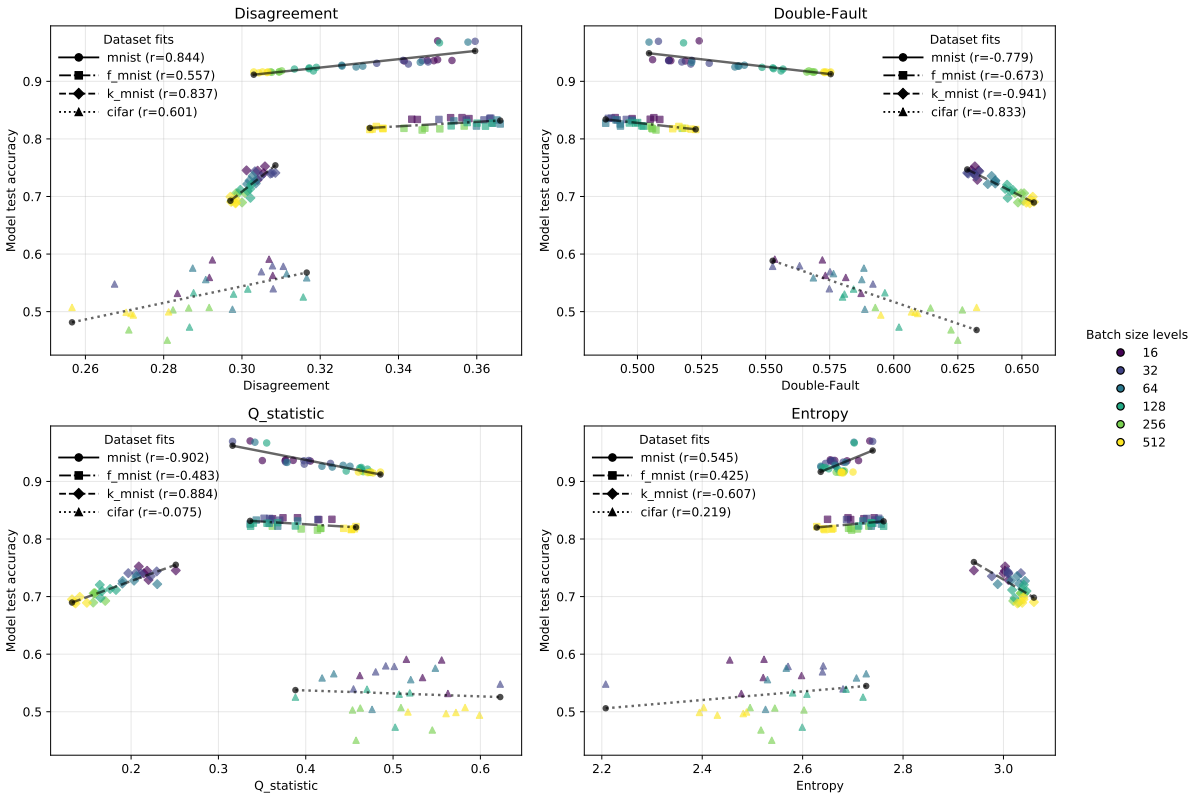


Figure 6.12: Correlation between model test accuracy and averaged diversity across datasets and metrics under induced diversity conditions when varying batch size.

measures, *Disagreement* and *DF* continue to demonstrate reliability and interpretability, consistently reflecting trends that align with improved test performance.

Relationship between *Q-statistic* and model performance across varying batch sizes: In Figure 6.12*(c)*, we see the continued inconsistency of the *Q-statistic* in capturing diversity trends across datasets. For MNIST, we observe a strong negative correlation ($r = -0.902$), aligning with previous results. However, Fashion-MNIST and CIFAR-10 show weak or negligible correlations ($r = -0.483$ and -0.075, respectively), while K-MNIST again exhibits a reversed trend with a positive correlation.

These conflicting results again reinforce earlier concerns about the *Q-statistic* as a reliable measure of diversity across datasets. Its sensitivity to what we assume to be dataset characteristics makes it difficult to interpret consistently. As such, it remains the least dependable and relatable among the diversity metrics considered.

Relationship between *Entropy* and model performance across varying batch sizes: *Entropy* displays another deviation in the context of batch size variation. As shown in Figure 6.12*(d)*, we observe weak positive correlations between *Entropy* and model accuracy for MNIST ($r = 0.545$), Fashion-MNIST ($r = 0.373$), and CIFAR-10 ($r = 0.248$). These suggest that higher *Entropy* may accompany improved model performance in some cases. However, K-MNIST shows a reversed trend with a negative correlation, indicating reduced *Entropy* as accuracy increases.

When considered alongside previously mixed trends between *Entropy* and batch size (see Figure 6.11*(d)*), these results suggest that *Entropy* is not a reliable indicator of diversity under batch size manipulations. Its inconsistent trends across datasets make it difficult to interpret in relation to generalisation performance.

Discussion: The findings show that batch size significantly influences model diversity, though the direction and strength of this effect may vary by metric and dataset. Measures like *Disagreement* and *DF* consistently show that smaller batch sizes promote greater diversity. As batch size increases, *Disagreement* decreases and *DF* increases, indicating that subpredictors tend to become more similar and co-fail more often. These diversity changes are strongly and positively related with model performance, suggesting that smaller batch sizes not only help improve generalisation, but also increase diversity within a neural network.

The *Q-statistic*, however, continues to show inconsistent behaviour. While MNIST generally follows the expected trend of decreasing agreement with smaller batch sizes, other datasets—especially K-MNIST—often show reversed or weak correlations. This instability makes *Q-statistic* a poor choice for assessing diversity in this setting—at least until the exact cause for its deviations are investigated.

Entropy also shows mixed results. While some datasets show a mild increase in uncertainty with larger batch sizes, the patterns are weak and inconsistent. The varying correlations between *Entropy* and model performance may be suggesting that varying the batch sizes influences *Entropy* in different ways compared to Dropout or DropConnect.

Altogether, these results show that smaller batch sizes help improve generalisation, and along with it increases the diversity of its subpredictors—when using *Disagreement* and *DF*.

6.5.4 Summary

This section explored how varying the batch size when training affects the model performance and internal diversity in single neural networks. Smaller batch sizes consistently led to higher test accuracy across all datasets—for layer accuracy, it especially influenced the deeper layers. This performance gain is likely due to the regularising effect of noisier gradient updates during training.

Diversity analyses revealed that *Disagreement* and *DF* metrics reliably increased as batch size decreased, indicating greater subpredictor independence. These trends closely followed improvements in model accuracy, illustrating the link between diversity and generalisation.

In contrast, the *Q-statistic* showed inconsistent patterns—particularly on K-MNIST—making us doubt its reliability across datasets. *Entropy* results were similarly mixed and weakly correlated with both batch size and model performance, limiting its interpretability in this setting.

Overall, we find that using smaller batch sizes act as an effective form of regularisation that helps improve generalisation and diversity within a single neural network, as captured by *Disagreement* and *DF*.

6.6 Conclusion

This chapter explored whether three techniques—Dropout, DropConnect, and variation in batch size—can be used to induce diversity within single neural networks, and whether this induced diversity correlates with improvements in model performance. We evaluated

these methods across multiple datasets and model architectures using a consistent set of diversity metrics: *Disagreement*, *Double-fault*, *Q-statistic*, and *Entropy*.

Across all three experimental setups, we find that both Dropout and DropConnect act as effective mechanisms for inducing diversity in neural networks. When measured using *Disagreement* and *DF*, both regularisers consistently increased diversity and improved model performance. These two metrics also showed strong correlations with model accuracy, confirming their reliability and interpretability in this context.

The *Entropy* and *Q-statistic* diversity metrics showed less consistent behaviour, however. While *Entropy* often decreased with increased regularisation—suggesting reduced uncertainty—its correlation with performance was often weak and data-set dependent. The *Q-statistic* was the least reliable, frequently displaying reversed or erratic trends, especially on the K-MNIST dataset. This raises questions about its general applicability as a diversity metric in single-network analyses.

Varying batch sizes also proved effective as a regularisation mechanism. Smaller batch sizes consistently led to higher accuracy and greater diversity according to *Disagreement* and *DF*. This aligns with the interpretation that smaller batches introduce beneficial gradient noise during training, encouraging subpredictor independence and better generalisation.

In conclusion, Dropout, DropConnect, and small batch sizes can all be used to induce diversity in single neural networks. When this diversity is measured through reliable metrics like *Disagreement* and *DF*, it shows a clear and positive relationship with improved model generalisation. These findings emphasize the value of viewing single neural networks as implicit ensembles, where internal diversity plays a role in model performance.

Table 6.3: CNN specification (conv bias=True, activation=ReLU, BN=False) with DropConnect being applied. "$\times n$" indicates repeated blocks.

Stage	Layers	Kernel / Stride / Pad	Output size (for 32×32 input)
Input			32×32×3
S1	Conv 64 ×2	$(3, 1, 1)$	32×32×64
	DropConnect		
	MaxPool	$(2, 2, 0)$	16×16×64
S2	Conv 128 ×2	$(3, 1, 1)$	16×16×128
	DropConnect		
	MaxPool	$(2, 2, 0)$	8×8×128
S3	Conv 256 ×3	$(3, 1, 1)$	8×8×256
	DropConnect		
	MaxPool	$(2, 2, 0)$	4×4×256
S4	Conv 512 ×3	$(3, 1, 1)$	4×4×512
	DropConnect		
	MaxPool	$(2, 2, 0)$	2×2×512
S5	Conv 512 ×3	$(3, 1, 1)$	2×2×512
	DropConnect		
	MaxPool	$(2, 2, 0)$	1×1×512
Head	Flatten		512
	Dense 512		512
	DropConnect		
	Dense 512		512
	DropConnect		
	Dense 512		512
	DropConnect		
	Dense(C)		C

Table 6.5: Summary of hyperparameters for MLP and CNN architectures. This configuration uses a varying batch sizes as the only regularisation mechanism applied.

Hyperparameter	**Value**
Optimizer	Adam
Learning rate	0.0003
Batch size	{16, 32, 64, 128, 256 (baseline), 512}
Epochs (max)	1 000
Dropout/DropConnect	None
BatchNorm	None
Learning rate scheduler	{Step size = 1, gamma = 0.99}

Chapter 7

Conclusion

In this chapter we summarize key findings and discuss the implications of these findings. We also reflect on limitations and possible future work.

7.1 Introduction

As initially discussed in Chapter 1, the goal of this study was to investigate and better understand diversity within the context of *single* neural networks that are viewed as implicit deep ensembles. In this chapter, we discuss our key findings, followed by the contributions and implications of the results. We conclude the study by mentioning the limitations and proposing avenues for future work.

7.2 Key findings

In Section 1.2, we asked the question: How can diversity among the ensemble members of a single neural network be quantified, interpreted, and actively induced, and how does

this diversity influence the model's expected generalisation performance? In this section, we answer this question and expand on the investigations we performed.

After conducting a series of experiments across multiple datasets, neural network architectures, and configurations, we identified several key findings. For each experiment, we analysed the results and discussed their implications. The most important findings and conclusions drawn from these investigations include:

- **Subpredictor viability and validation (Section 3.4):** Through a targeted empirical experiment, we verified that individual node-based subpredictors consistently outperform random guessing accuracy—supporting our subpredictor definition. Successfully replicating Davel et al.'s [1] results verified our implementation.
- **Applicability of classical diversity measures (Section 5.2.4):** Explicit diversity measures can be meaningfully applied to a *single* neural network at the hidden layer level. Through this application we found that, especially in deeper layers of a network, diversity and layer-level accuracy were positively correlated.
- **Diversity measure groupings and behaviour (Section 5.2.3):** The diversity measures can be categorised into interpretable groups based on their formulation and empirical behaviour:
  - Disagreement-based: *Disagreement*, *Binary Entropy*, *Kohavi–Wolpert variance*
  - Joint-error-based: *Double-fault*, *Generalised diversity*, *Coincident failure diversity*
  - Correlation-based: *Q-statistic*, *Kappa-statistic*, *Correlation coefficient*
  - Multi-class specific: *Entropy*

  Measures within the same group tended to exhibit similar behaviour across experiments. Combined with our earlier result demonstrating the successful application of classical diversity measures to hidden layers in single networks, this addresses our questions posed regarding how diversity can be quantified and interpreted within implicit ensembles.

- **Dataset-specific inconsistencies (Section 5.2.4):** Correlation-based measures along with the *Entropy* measure showed inconsistent or reversed relationships between layer accuracy and diversity on the K-MNIST dataset, suggesting some sensitivity to data characteristics. This calls attention to the need for caution when interpreting these measures across tasks.

- **Impact of network capacity (Section 5.3.4):** Increasing network width for a single layer network improved model-level generalisation performance, but results in decreased diversity and layer-level accuracy. This indicates that while overall capacity improves model performance, the effectiveness of individual subpredictors decreases. The *Double-fault* measure remained the most reliable in tracking these trends, correlating positively with model accuracy—indicating decreased diversity—and negatively with layer performance. This finding also suggests that the relationship between layer accuracy and overall model accuracy is still unclear.

- **Diversity evolution during training (Section 5.4.4):** It was shown that diversity is not a static component of the network, it evolves with training. We observed that diversity across subpredictors increased consistently as the model learned, tracking both the model accuracy and the layer accuracy. This trend suggests that diversity emerges naturally during optimisation.

- **Diversity induction through regularisation (Sections 6.3.4 and 6.4.4):** Regularisation techniques such as Dropout and DropConnect effectively increased diversity among subpredictors while also improving model and layer performance. *Disagreement* and *Double-fault* were the most reliable and consistent indicators of diversity. In contrast, the *Q-statistic* remained inconsistent across datasets making it a less reliable indicator of diversity. *Entropy* showed opposing trends, suggesting a different viewpoint on diversity compared to the disagreement and joint-error-based metrics. This finding addresses our questions regarding how diversity can be induced, as well as how it relates to the model's generalisation performance.

- **Diversity induction through varying batch size (Section 6.5.4):** Smaller batch sizes improved model generalisation performance and induced higher diversity.

Consistent with earlier findings, disagreement-based and joint-error-based measures reliably showed the positive relationship between generalisation and diversity. Both the *Q-statistic* and *Entropy* metrics show inconsistent results across datasets (especially on the K-MNIST dataset), limiting their interpretability and reliability for practical use. This finding shows another method for influencing diversity, and also answers our questions regarding how to induce diversity among subpredictors, and what relationship this induction of diversity has with model performance.

- **Summary on diversity induction results (Section 6.6):** Across all three diversity induction mechanisms—Dropout, DropConnect, and reduced batch size—we confirmed that diversity among subpredictors could be induced effectively. Among all evaluated diversity metrics, disagreement- and joint-error-based metrics emerged as the most consistent and interpretable indicators of diversity within implicit deep ensembles. These metrics normally align strongly with improved generalisation as diversity is induced.

Detailed discussions and contextualised conclusions relating to each finding are presented within the respective sections where the results were introduced. See Figure 7.1 for a high-level summary of our key findings, together with pointers to the relevant results supporting each one.

Conclusion: To address the most important question of this study—can we actively induce diversity, and does this lead to improved model generalisation performance?—the answer is nuanced. Across the majority of our experiments, we observed strong positive relationships between diversity and generalisation performance, especially when measured using disagreement- and joint-error-based metrics. We found that inducing diversity through methods like Dropout, DropConnect, or smaller batch sizes consistently resulted in improved model performance.

However, this relationship was not consistent across all experiments—in our experiments on increasing model capacity by increasing the width of the network, we observed that

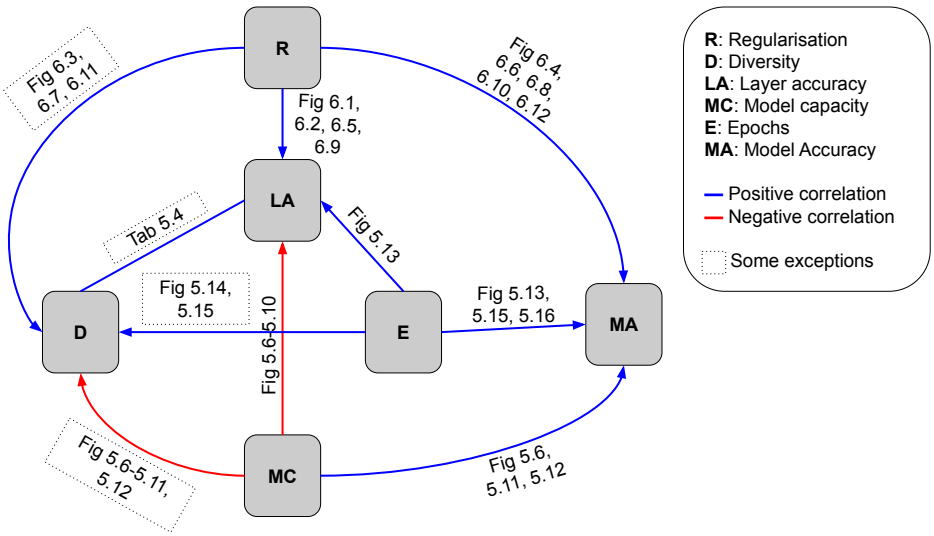


Figure 7.1: A visual illustration of our findings regarding diversity, model performance, capacity, and regularisation. Blue and red vertices indicate a positive and negative relationship, respectively. Arrows points from the actual independent variable that was varied in the relevant experiments. Vertices are tagged with the relevant results in the dissertation. Tags that have dotted outlines indicate the presence of some exceptions.

generalisation performance continued to improve even as diversity declined. This deviation suggests that while diversity contributes to improved generalisation, it is not strictly a causal factor.

In conclusion, diversity and generalisation are not always correlated, but in every case where diversity was explicitly induced through different regularisation mechanisms, increased diversity did correspond with improved model performance.

7.3 Contributions

This dissertation makes the following original contributions:

- Building on, and extending prior work by Davel et al. [1] by performing a deeper analysis of the nodes as classifiers.

- We demonstrate that well-established explicit ensemble diversity metrics can be applied within a single neural network, opening up new avenues for internal model analysis.
- We empirically evaluate the relationship between subpredictor diversity and generalisation performance.
- Finally, we provide—as far as we know of—one of the first comparative analyses on diversity induction techniques on implicit deep ensembles, and their relationship with model performance.

Another contribution includes a conference paper which has been accepted for publication [35].

This study addresses the questions and objectives posed in Section 1.2 by showing how ensemble theory can be meaningfully extended to the internal components of deep neural networks.

7.4 Limitations and future work

Although this study presents strong evidence for the role of diversity within neural networks, we acknowledge several limitations, and mention possible avenues for future work based on new questions that emerged.

Our findings presented evidence that the *Disagreement* and *Double-fault* metrics consistently correlate with generalisation. However, we found that there are instances where this correlation does not hold. In our experiment on exploring the effect of varying network capacity on the relationship between diversity and model performance, we observed a decline in diversity despite an increase in generalisation performance. This presents a notable exception to the overall trend and suggests that the relationship between diversity and generalisation may be context-dependent. Alternatively, the diversity metrics we investigate might be scale-dependent in a way that would artificially reduce the resulting

diversity value when more classifiers are considered.

From the literature, we know that the regularisation techniques we apply have an effect on the effective capacity of the model [87]. However, we see contradictory trends when modifying the representational capacity directly (i.e. by varying model width). Although all three regularisation techniques led to increased diversity and generalisation, they do so by *constraining* model capacity, unlike when we increase the width of model which in turn increases the model capacity. We find this divergence in results to be an interesting finding—while diversity often seems to be important for generalisation, based on this divergence, we hypothesize that there is a deeper mechanism that affects both diversity and generalisation, potentially linked to the way subpredictors are distributed and learn through training.

Another limitation lies in the sensitivity of the *Q-statistic* and *Entropy* measures to dataset-specific characteristics. The *Q-statistic*, for example, produced inconsistent or reversed trends—especially on the K-MNIST dataset—suggesting it may be less robust across different types of data.

Our experiments were limited to relatively shallow MLPs and small CNN architectures. While we believe this to be sufficient to investigate and explore subpredictor behaviour and diversity within neural networks, these models may not fully capture the complexity of modern deep learning architectures such as ResNets [12] or Transformers [3]. Future work would investigate the diversity dynamics for more modern architectures to confirm whether the same diversity dynamics hold.

Finally, our definition of subpredictors was contrained to fully-connected nodes. However, convolutional layers were excluded and out of scope for this study. Extending our framework to handle convolutional feature maps as subpredictors would offer a more complete view of ensemble behaviour in CNNs.

7.5 Final remarks

This study contributes to fields that attempt to better understand the internal workings of the "black box" known as deep neural networks. By exploring diversity—usually related to classical ensemble theory—from the perspective of implicit ensembles, we demonstrated that we can gleam valuable insights into the internal workings of these networks, and their relationship to generalisation. Our results show that diversity among subpredictors is not only measurable, but can also be induced and that this induced diversity correlates positively with generalisation performance in most cases.

While some limitations and inconsistencies emerged, and open questions remain, this work lays a foundation for further investigations into diversity-driven mechanisms in deep neural networks.

This work is based on the research supported in part by the National Research Foundation of South Africa (Grant Reference Numbers: PSTD23042898868, RCDL240215206999).

Bibliography

[1] M. H. Davel, M. W. Theunissen, A. M. Pretorius, and E. Barnard, "DNNs as layers of cooperating classifiers," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020. DOI: 10.1609/aaai.v34i04.5782.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2012.

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[4] G. Hinton, L. Deng, D. Yu, *et al.*, "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org.

[7] S. J. Prince, *Understanding Deep Learning*. The MIT Press, 2023. [Online]. Available: http://udlbook.com.

[8] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *International Conference on Learning Representations (ICLR)*, 2017.

[9] F. Doshi-Velez and B. Kim, "Towards a rigorous science of interpretable machine learning," *arXiv preprint arXiv:1702.08608*, 2017.

[10] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998. DOI: 10.1109/5.726791.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] C. Szegedy, W. Liu, Y. Jia, *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–9.

[14] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 1877–1901, 2020.

[15] B. Neyshabur, S. Bhojanapalli, D. McAllester, and N. Srebro, "Exploring generalization in deep learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[16] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang, *On large-batch training for deep learning: Generalization gap and sharp minima*, 2017. arXiv: 1609.04836 [cs.LG].

[17] A. Krogh and J. A. Hertz, "A simple weight decay can improve generalization," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 4, 1992, pp. 950–957.

[18] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research (JMLR)*, vol. 15, no. 1, pp. 1929–1958, 2014. DOI: 10.5555/2627435.2670313.

[19] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, PMLR, 2015, pp. 448–456.

[20] D. G. T. Barrett, B. Dherin, and S. L. Smith, "On the origin of implicit regularization in stochastic gradient descent," in *International Conference on Learning Representations (ICLR)*, 2021.

[21] D. G. T. Barrett and B. Dherin, "Implicit gradient regularization," in *International Conference on Learning Representations (ICLR)*, 2021.

[22] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," *arXiv preprint arXiv:1803.05407*, 2018.

[23] C. Olah, N. Cammarata, L. Schubert, G. Goh, M. Petrov, and S. Carter, *Zoom in: An introduction to circuits*, Distill, 2020. [Online]. Available: `https://distill.pub/2020/circuits/zoom-in/`.

[24] R. Csordás, S. van Steenkiste, and J. Schmidhuber, *Are Neural Nets Modular? Inspecting Functional Modularity Through Differentiable Weight Masks*, Mar. 2021. arXiv: `2010.02066 [cs]`.

[25] D. Filan, S. Casper, S. Hod, C. Wild, A. Critch, and S. Russell, "Clusterability in neural networks," *arXiv preprint arXiv:2103.03386*, 2021.

[26] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson, "Loss surfaces, mode connectivity, and fast ensembling of dnns," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, 2018.

[27] A. Veit, M. J. Wilber, and S. Belongie, "Residual networks behave like ensembles of relatively shallow networks," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016. DOI: `10.48550/arXiv.1605.06431`.

[28] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine Learning*, vol. 51, pp. 181–207, 2003. DOI: `10.1023/A:1022859003006`.

[29] G. Giacinto and F. Roli, "An approach to the automatic design of multiple classifier systems," *Pattern Recognition Letters*, vol. 22, no. 1, pp. 25–33, 2001.

[30] M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, and P. N. Suganthan, "Ensemble deep learning: A review," *Engineering Applications of Artificial Intelligence*, vol. 115, pp. 105–151, 2022. DOI: 10.1016/j.engappai.2022.105151.

[31] Y. Bian and H. Chen, "When Does Diversity Help Generalization in Classification Ensembles?" *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 9059–9075, Sep. 2022. DOI: 10.1109/TCYB.2021.3053165.

[32] T. Clanuwat, M. Bober-Irizar, A. Kitamoto, A. Lamb, K. Yamamoto, and D. Ha, "Deep learning for classical japanese literature," *arXiv preprint arXiv:1812.01718*, 2018. DOI: 10.48550/arXiv.1812.01718.

[33] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[34] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

[35] R. P. van der Spoel and R. Rabe, "Investigating the relationship between diversity and generalization in deep neural networks," in *Northern Lights Deep Learning Conference 2026*, 2025. [Online]. Available: https://openreview.net/forum?id=Khtjo6nLG8.

[36] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, JMLR Workshop and Conference Proceedings, vol. 15, PMLR, Nov. 2011, pp. 315–323.

[37] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International Conference on Artificial Neural Networks (ICANN)*, Springer, 2010, pp. 92–101.

[38] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *Journal of Artificial Intelligence Research (JAIR)*, vol. 11, pp. 169–198, 1999.

[39] T. G. Dietterich, "Ensemble methods in machine learning," in *International Workshop on Multiple Classifier Systems*, Springer, 2000, pp. 1–15. DOI: 10.1007/3-540-45014-9_1.

[40] M. d. Condorcet, "Essay on the application of analysis to the probability of majority decisions," *Paris: Imprimerie Royale*, p. 1785, 1785.

[41] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[42] L. I. Kuncheva, *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[43] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, pp. 123–140, 1996. DOI: 10.1007/BF00058655.

[44] Y. Freund and R. E. Schapire, "A desicion-theoretic generalization of on-line learning and an application to boosting," in *Computational Learning Theory*, Springer Berlin Heidelberg, 1995, pp. 23–37. DOI: 10.1006/jcss.1997.1504.

[45] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Annals of Statistics*, pp. 1189–1232, 2001.

[46] D. H. Wolpert, "Stacked generalization," *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[47] A. Krogh and J. Vedelsby, "Neural network ensembles, cross validation, and active learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 7, 1994.

[48] L. Hansen and P. Salamon, "Neural Network Ensembles," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 993–1001, 1990. DOI: 10.1109/34.58871.

[49] S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," *Neural Computation*, vol. 4, no. 1, pp. 1–58, 1992.

[50] P. Domingos, "A unified bias-variance decomposition," *Proceedings of the 17th International Conference on Machine Learning (ICML)*, pp. 231–238, 2000.

[51] D. Wood, T. Mu, A. M. Webb, H. W. Reeve, M. Luján, and G. Brown, "A unified theory of diversity in ensemble learning," *Journal of Machine Learning Research (JMLR)*, vol. 24, no. 359, pp. 1–49, 2023. DOI: 10.48550/arXiv.2301.03962.

[52] G. Brown, J. L. Wyatt, P. Tino, and Y. Bengio, "Managing diversity in regression ensembles," *Journal of Machine Learning Research (JMLR)*, vol. 6, no. 9, 2005.

[53] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. CRC press, 2012.

[54] G. U. Yule, "On the association of attributes in statistics: With illustrations from the material of the childhood society," *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, vol. 194, no. 252-261, pp. 257–319, 1900.

[55] P. H. A. Sneath and R. R. Sokal, *Numerical Taxonomy: The Principles and Practice of Numerical Classification*. San Francisco: W. H. Freeman, 1973.

[56] D. B. Skalak, "The sources of increased accuracy for two proposed boosting algorithms," in *Proceedings of the AAAI-96 Integrating Multiple Learned Models Workshop*, vol. 1129, 1996, pp. 1133–1134.

[57] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[58] J. Cohen, "A coefficient of agreement for nominal scales," *Educational and Psychological Measurement*, vol. 20, no. 1, pp. 37–46, 1960.

[59] R. Kohavi, D. H. Wolpert, *et al.*, "Bias plus variance decomposition for zero-one loss functions," in *International Conference on Machine Learning (ICML)*, vol. 96, 1996, pp. 275–283.

[60] D. Partridge and W. Krzanowski, "Software diversity: Practical statistics for its measurement and exploitation," *Information and Software Technology*, vol. 39, no. 10, pp. 707–717, 1997.

[61] P. Cunningham and J. Carney, "Diversity versus quality in classification ensembles based on feature selection," in *European Conference on Machine Learning (ECML)*, Springer, 2000, pp. 109–116. DOI: 10.1007/3-540-45164-1_12.

[62] C. Watanabe, "Interpreting layered neural networks via hierarchical modular representation," in *International Conference on Neural Information Processing (ICONIP)*, Springer International Publishing, 2019, pp. 376–388. DOI: 10.1007/978-3-030-36808-1_41.

[63] C. Watanabe, K. Hiramatsu, and K. Kashino, "Modular representation of autoencoder networks," in *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2017, pp. 1–8.

[64] C. Watanabe, K. Hiramatsu, and K. Kashino, "Recursive extraction of modular structure from layered neural networks using variational Bayes method," in *International Conference on Discovery Science*, Springer, 2017, pp. 207–222.

[65] C. Watanabe, K. Hiramatsu, and K. Kashino, "Modular representation of layered neural networks," *Neural Networks*, vol. 97, pp. 62–73, 2018.

[66] C. Watanabe, K. Hiramatsu, and K. Kashino, "Understanding community structure in layered neural networks," *arXiv preprint arXiv:1804.04778*, 2018.

[67] D. Filan, S. Hod, C. Wild, A. Critch, and S. Russell, "Pruned neural networks are surprisingly modular," *arXiv preprint arXiv:2003.04881*, 2020.

[68] R. D. Lange, D. Rolnick, and K. Kording, "Clustering units in neural networks: Upstream vs downstream information," *Transactions on Machine Learning Research (TMLR)*, 2022.

[69] D. Zhang, K. Ahuja, Y. Xu, Y. Wang, and A. Courville, "Can subnetwork structure be the key to out-of-distribution generalization?" In *International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 12 356–12 367.

[70] M. Patil, L. Michael, and C. Dovrolis, "Neural sculpting: Uncovering hierarchically modular task structure in neural networks through pruning and network analysis," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, 2023, pp. 18 495–18 531.

[71] M. Sun and P. Chatain, *Do Neural Networks Generalize from Self-Averaging Subclassifiers in the Same Way As Adaptive Boosting?* Feb. 2023. DOI: 10.48550/arXiv.2302.06923.

[72] T. Theunissen, M. Davel, and E. Barnard, "Insights regarding overfitting on noise in deep learning," in *FAIR 2020*, 2019. [Online]. Available: `http://ceur-ws.org/Vol-2540/FAIR2019_paper_46.pdf`.

[73] D. Doimo, A. Glielmo, S. Goldt, and A. Laio, "Redundant representations help generalization in wide neural networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, 2022, pp. 19 659–19 672.

[74] A. M. Proca, F. E. Rosas, A. I. Luppi, D. Bor, M. Crosby, and P. A. Mediano, "Synergistic information supports modality integration and flexible learning in neural networks solving multiple tasks," *PLoS Computational Biology*, vol. 20, no. 6, 2024.

[75] F. Locatello, S. Bauer, M. Lucic, *et al.*, "Challenging common assumptions in the unsupervised learning of disentangled representations," in *International Conference on Machine Learning (ICML)*, ser. Proceedings of Machine Learning Research, vol. 97, 2019, pp. 4114–4124.

[76] L. Ruis, J. Andreas, M. Baroni, *et al.*, "A benchmark for systematic generalization in grounded language understanding," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, 2020.

[77] R. Hu, J. Andreas, T. Darrell, and K. Saenko, "Explainable neural computation via stack neural module networks," in *European Conference on Computer Vision (ECCV)*, 2018.

[78] A. Goyal, A. Lamb, J. Hoffmann, *et al.*, "Recurrent independent mechanisms," *arXiv preprint arXiv:1909.10893*, 2019.

[79] W. Dorrell, K. Hsu, L. Hollingsworth, *et al.*, "Range, not independence, drives modularity in biologically inspired representations," *arXiv preprint arXiv:2410.06232*, 2024.

[80] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[81] A. Paszke, S. Gross, S. Chintala, *et al.*, "Automatic differentiation in PyTorch," in *Advances in Neural Information Processing Systems (NeurIPS) Autodiff Workshop*, 2017.

[82] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, vol. 116, no. 32, pp. 15 849–15 854, 2019. DOI: 10.1073/pnas.1903070116.

[83] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever, "Deep double descent: Where bigger models and more data hurt," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2021, no. 12, 2021. DOI: 10.1088/1742-5468/ac3a74.

[84] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *International Conference on Machine Learning (ICML)*, 2013, pp. 1058–1066. DOI: 10.5555/3042817.3043055.

[85] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, Dec. 2014.

[86] S. Smith, E. Elsen, and S. De, "On the generalization benefit of noise in stochastic gradient descent," in *Proceedings of the 37th International Conference on Machine Learning (ICML)*, H. Daumé and A. Singh, Eds., vol. 119, PMLR, Jul. 2020, pp. 9058–9067.

[87] R. Arora, P. Bartlett, P. Mianjy, and N. Srebro, "Dropout: Explicit forms and capacity control," in *International Conference on Machine Learning (ICML)*, PMLR, 2021, pp. 351–361.

Appendix A

Supplemental content

A.1 Replication of prior work

A.1.1 Overview

In this section, we replicate key results from Davel et al. [1]'s framework in order to ensure that our implementation of nodes-as-subpredictors align with prior work. We begin by examining the activation distributions of individual nodes across classes, which provide intuition about how hidden units differentiate between input categories. We then evaluate the layer-wise classification performance under the continuous estimation strategy proposed by Davel et al. [1]. This section directly relates to the verification steps discussed in Section 3.4.

A.1.2 Experimental setup

Training protocol: Our replication experiments are done following a similar setup that was employed by Davel et al. [1].

During these experiments, models were trained to a maximum of 50 epochs using early stopping. The model with the best validation accuracy across epochs was saved. The

Adam [85] optimiser was used with a learning rate of 0.001 along with a batch size of 256, and no batch normalisation was applied. The models were trained on the MNIST [10] dataset with a 50 000-sample training split, a 10 000-sample validation split, and the pre-defined 10 000-sample test split was used. Although Davel et al. [1] conducted their experiments on the Fashion-MNIST [33] dataset, we use MNIST here solely as a means of verification.

The model architecture was chosen to be similar to what was used in their paper:

- Fully-connected MLP
- Six hidden-layers
- 100 nodes per hidden layer

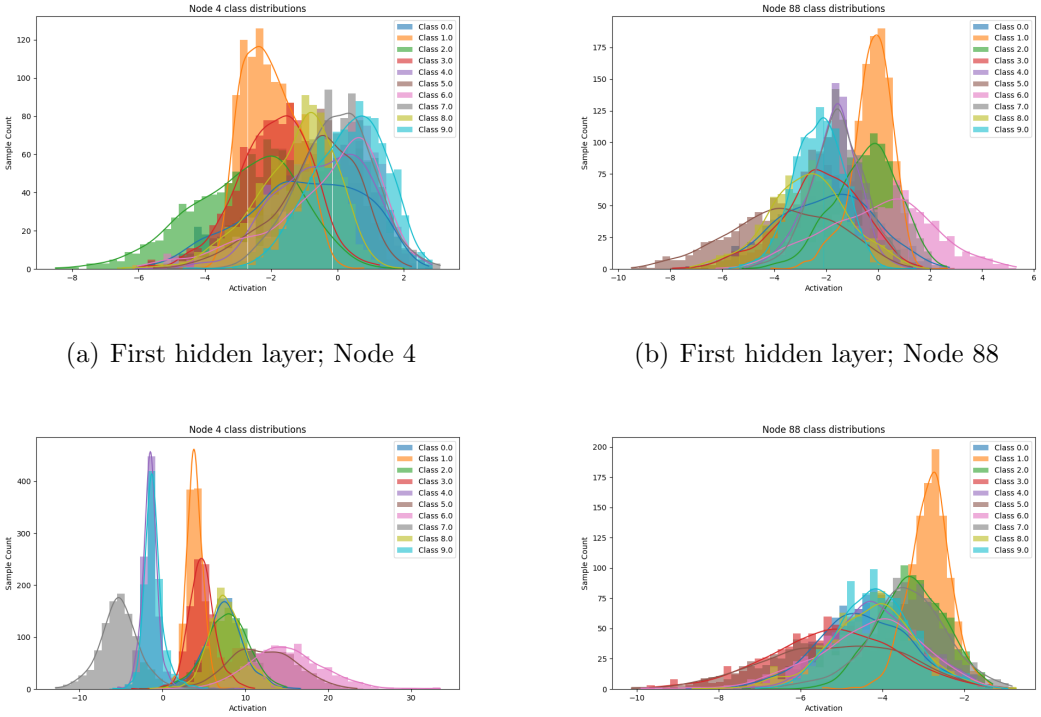
A.1.3 Node-level class activation distributions

To gain insight into the class selectivity of hidden nodes, we visualise $P(z^n_{(\ell)}(\mathbf{x}^{(i)}) \mid c)$ as defined in Section 3.3.1, but for all samples in the test set instead of the training set. Using these plots, we illustrate how strongly individual nodes respond to different classes.

The results shown in Figure A.1 illustrate $P(z^n_{(\ell)}(\mathbf{x}) \mid c)$ for different classes at four randomly chosen nodes. These pre-activation distributions illustrate that individual nodes vary significantly in how they respond to samples of different classes.

In the first hidden layer (Figure A.1*(a-b)*), we observe substantial overlap between classes. We also observe that the variance differs by class: some classes are tightly clustered around a mean activation value (see for example, Figure A.1*(b)* Class 1), while others are spread more widely (Figure A.1*(b)* Class 5).

In contrast, the last hidden layer (Figure A.1*(c-d)*) exhibits sharper and more distinct peaks for several classes, with clearer separation between class means, especially for Figure A.1*(c)*. This supports the notion that deeper layers develop nodes that are increasingly class-specific, with stronger and more consistent responses for certain categories [1].



(a) First hidden layer; Node 4

(b) First hidden layer; Node 88

(c) Last hidden layer; Node 4

(d) Last hidden layer; Node 88

Figure A.1: Class activation distribution at four randomly selected nodes of a trained six-hidden-layer MLP on the MNIST test set. Each colour represents a different class. Each curve represents the distribution of pre-activation values (before the activation function) for samples belonging to a specific class.

Overall, these findings support the intuition that certain nodes activate more strongly for certain classes, and that class activation distributions can be used as a basis for generating class predictions from node pre-activation values, as proposed by Davel et al. [1].

A.1.4 Layer-wise accuracy under different estimates

In Figure A.2, we present the layer-specific accuracies for the continuous classification method mentioned by Davel et al. [1]. Their approach uses the node-as-classifier construction introduced in Section 3.3.1, but instead of using "hard votes" and a majority vote rule to obtain a layer prediction, they combine the class-conditional probabilities

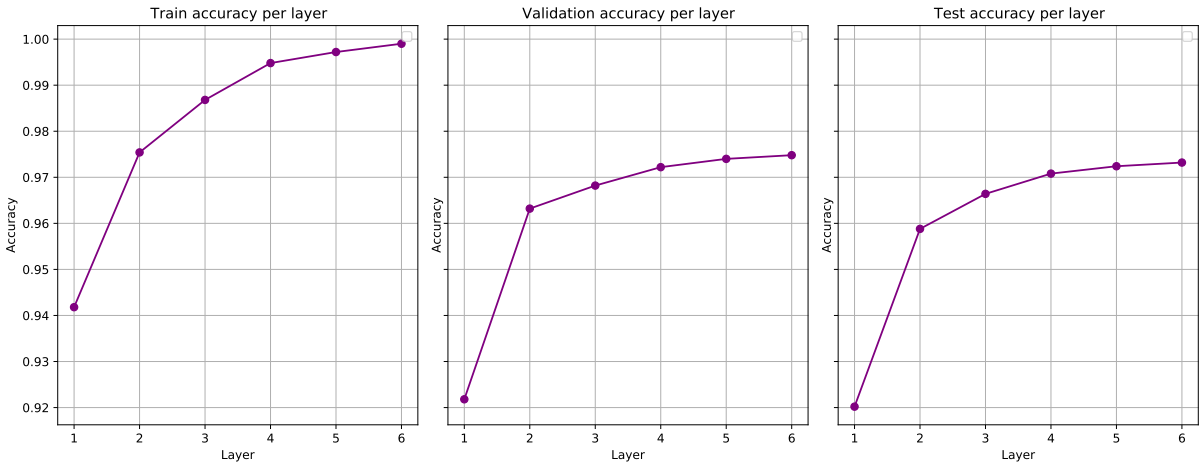


Figure A.2: Layer-wise accuracy on the MNIST training, validation, and test set. The three sub-plots represent training *(left)*, validation *(middle)*, and test *(right)* accuracy per layer. The results in the figure were obtained using Davel et al.'s [1] method of combining node probabilities within a layer.

$P\big(c \mid z^n_{(\ell)}\big)$ across all nodes in a layer by taking the log-sum of these values. The resulting aggregated score is then used to produce a layer prediction by selecting the most likely class:

$$\hat{y}_{(\ell)}(\mathbf{x}^{(i)}) = \arg\max_c \sum_n^{\mathcal{N}_{(\ell)}} P\big(c \mid z^n_{(\ell)}(\mathbf{x}^{(i)})\big).$$

We observe a consistent increase in layer accuracy with network depth. This is shown by the layer accuracy of 92.02% in Layer 1 that increases to 97.32% in Layer 6 on the test set. The observed trend remain consistent across the training, validation and test sets.

When comparing our results to those reported in Figure 4 by Davel et al. [1], we find that our re-implementation produces comparable results, verifying that our nodes-as-subpredictors framework is consistent with and faithful to the original methodology.

A.1.5 Discussion

The experiments presented in this section serve to verify our nodes-as-subpredictors implementation by reproducing related results from Davel et al. [1]. The visualisation of

class activation distributions illustrated that individual nodes have varying levels of class selectivity. This variation in class selectivity supports our construction of node-level subpredictors as it illustrates that different nodes become more sensitive to specific classes.

The layer-wise classification performance obtained using the continuous estimation strategy followed the trends reported in Davel et al. [1]. The high accuracy observed over all layers suggests that the ensembling of nodes in a layer may yield more reliable predictions than the individual nodes (similar to what we would expect from the outputs of members of an ensemble being combined).

Although our method employs a different method of combining the node-level predictions to obtain layer-level accuracies, these results support the underlying principle of using nodes as subpredictors within a single ANN that we view as an implicit ensemble.

A.2 Sensitivity analysis

In this sensitivity analysis, we investigate how the number of samples used to compute the diversity metrics impact the stability and reliability of these metrics across layers in a ANN. Specifically, we evaluate four of the diversity measures: *Disagreement*, *Double-fault*, *Q-statistic*, and *Entropy* (the representative measures as mentioned in Section 5.2.3).

A.2.1 Experimental setup

Training protocol and hyperparameter setup: The experiments were completed using the same same training protocol as described in Section 4.3.2, along with the same hyperparameter setup as in Table 5.1.

Model architectures: Instead of using a simple three layer network, we make use of a deeper network with six hidden layers. This network is purely an MLP that has been constructed in a similar manner as described in Section 5.2.

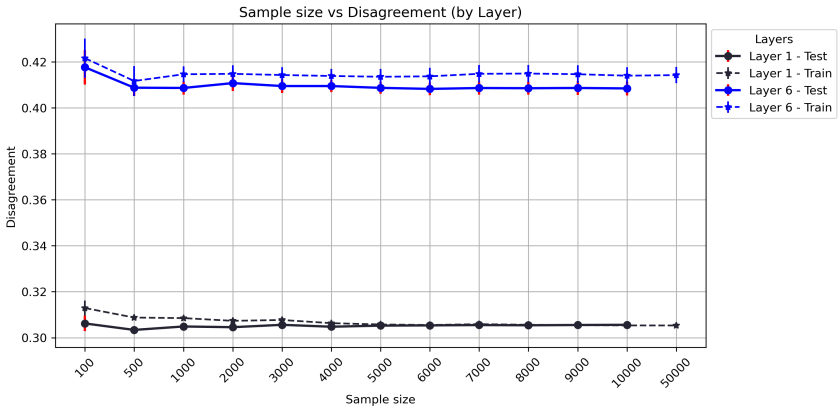
Sampling and evaluation setup: Each metric is assessed on both training and test datasets. To reduce randomness in the results, five different seeded initialisations were employed during training. The average and standard deviation over these five initialisations are then used for the analysis.

When sub-sampling from the full training or test data sets, there is a risk that the resulting subsets may not accurately reflect the original dataset's distributions—in other words, by naively sampling from the full dataset, we introduce the possibility of causing class imbalances. To mitigate this effect, we employ stratified sampling. This technique ensures that the relative proportions of each class in the subset closely follows those in the original dataset.

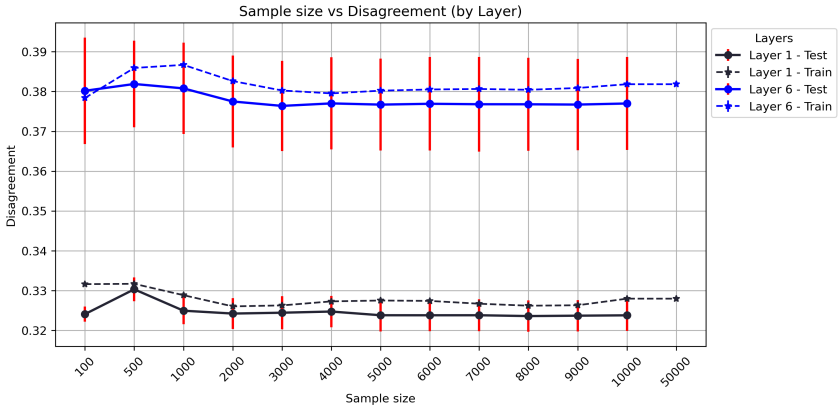
A.2.2 Results

In this section, we present the results showing how the representative diversity measures—*Disagreement*, *Double-fault*, *Q-statistic*, and *Entropy*—behave when calculated on progressively smaller stratified subsets of the training and test set. The ideal sample size for testing is when the diversity curves obtained from the training and test subsets behave similarly and plateaus as sample size increases, indicating that both subsets provide comparably representative estimates of the underlying data distribution. That is, we identify the point at which the diversity estimates for both the training and test sets stabilise, i.e., when further increases in sample size yield negligible changes in the measured values.

***Disagreement*:** In Figure A.3, we see how the *Disagreement* measure is influenced by using subsets of the full dataset. For both MNIST and Fashion-MNIST, we see that the train and test sets plateau after around 4 000 samples. This suggests that using a stratified subset of at least 4 000 samples should result in stable and relatively accurate results, at least based on the *Disagreement* results.



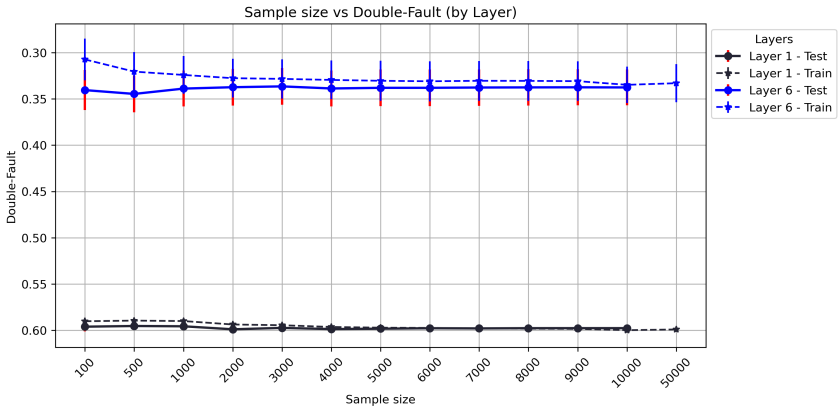
(a) MNIST



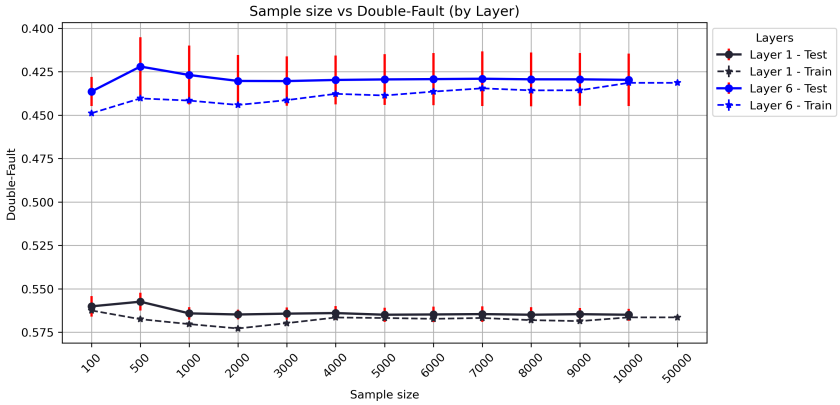
(b) Fashion-MNIST

Figure A.3: *Disagreement* as measured across varying sample sizes for two representative layers (Layer 1 and 6), comparing results for both the training and test sets. Lines in red indicate the standard deviation across five random initialisations.

***Double-fault*:** Figure A.4 illustrates how the *Double-fault* measure changes with the number of samples used for analysis. Similar to the *Disagreement* measure, we see that the train and test sets reach a plateau near 4 000 samples for MNIST, whereas for Fashion-MNIST, only the first layer plateaus near 4 000 samples. For Layer 6, the difference between the train and test set is small relative to the observed variability (as indicated by the overlapping standard deviation bars), suggesting that the apparent gap is unlikely to meaningfully affect subsequent analyses. After around 4 000 samples, the curves flatten and variability stabilises, suggesting that at least 4 000 samples should be used for analyses on both MNIST and Fashion-MNIST datasets.



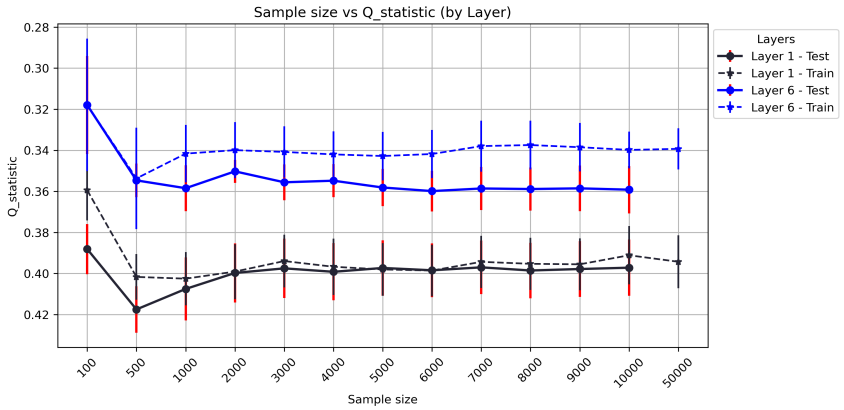
(a) MNIST



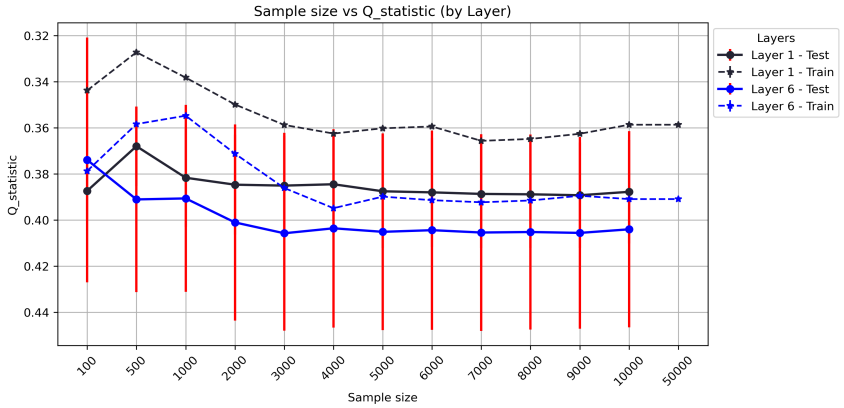
(b) Fashion-MNIST

Figure A.4: *Double-fault* as measured across varying sample sizes for two representative layers (Layer 1 and 6), comparing results for both the training and test sets. Lines in red indicate the standard deviation across initialisations.

***Q-statistic*:** The *Q-statistic*, as shown in Figure A.5, has a greater difference between train and test sets, especially for the Fashion-MNIST dataset (Figure A.5*(b)*). Due to the large standard deviation observed for the MNIST, and especially for the Fashion-MNIST dataset, we find the *Q-statistic* to be sensitive to changes in sample size. We do, however, observe that the difference between train and test set results still stabilises to some degree after around 4 000 samples. This suggests that, although it would be ideal to use the full train and test sets for analysis, we could still make use of a minimum of 4 000 stratified samples for analysis.



(a) MNIST

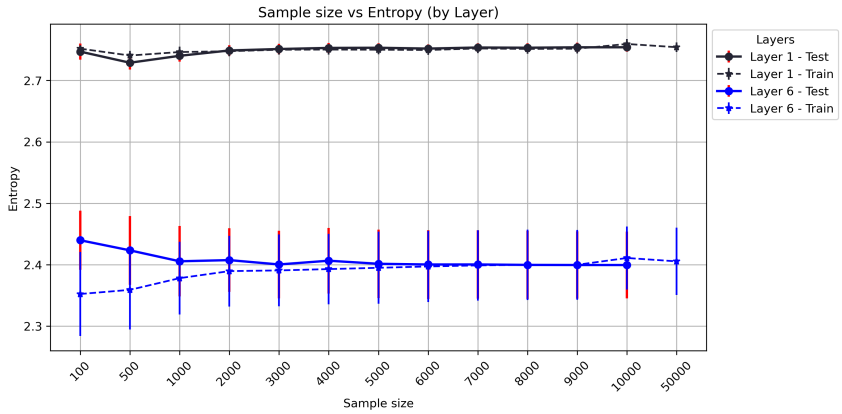


(b) Fashion-MNIST

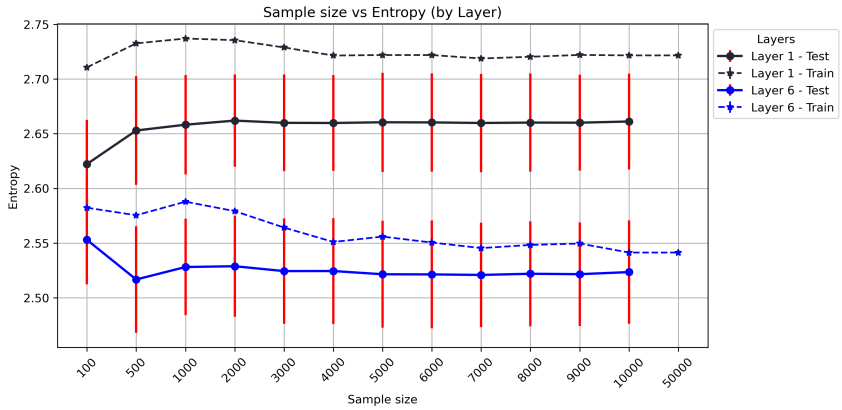
Figure A.5: *Q-statistic* as measured across varying sample sizes for two representative layers (Layer 1 and 6), comparing results for both the training and test sets. Lines in red indicate the standard deviation across initialisations.

Entropy: Figure A.6 illustrates how the *Entropy* measure varies as a function of sample size.

For MNIST (Figure A.6.a), the curves stabilise rapidly. For Layer 1, both the train and test sets exhibit minimal variation beyond $2\,000 - 4\,000$ samples, with only negligible changes thereafter. Similar behaviour is observed for Layer 6, where *Entropy* remains effectively constant across increasing sample size. Although minor fluctuations are visible at very small sample sizes (e.g., $100 - 500$ samples). Overall, the *Entropy* trends for MNIST suggest that relatively modest stratified subsets ($\approx 4\,000$ samples) are sufficient



(a) MNIST



(b) Fashion-MNIST

Figure A.6: Entropy as measured across varying sample sizes for two representative layers (Layer 1 and 6), comparing results for both the training and test sets. Lines in red indicate the standard deviation across initialisations.

to obtain stable and representative diversity estimates.

For Fashion-MNIST (Figure A.6.*b*), the behaviour is slightly less stable—especially at very small sample sizes. In Layer 1, *Entropy* exhibits noticeable variability below approximately $1\,000 - 2\,000$ samples. However, beyond approximately 4 000 samples, the mean *Entropy* values plateau. A similar pattern is observed for Layer 6, where *Entropy* initially fluctuates before stabilising as the sample size increases. Importantly, after approximately 4 000 samples, additional increases in sample size produce negligible changes in the observed *Entropy* trends.

A.2.3 Conclusion

The observed trends remain consistent across most diversity measures, at least after using a stratified subset of 4 000 samples. However, larger sample sizes more closely approximate the results obtained from the full training and test sets.

Based on the results, we make the following conclusions and recommendations:

- A minimum of 4 000 stratified samples is recommended for any meaningful analysis.
- The optimal sample size for analysis, when using stratified sampling, is 5 000 samples.
- Sample sizes greater than 5 000 provide results comparable to those of the full dataset but may lead to unnecessarily long analyses and excessive computational usage.

Accordingly, all analyses in this study will employ stratified sampling with 5 000 test samples.

A.3 Capacity experiments

A.3.1 Weight-change analysis

To test the hypothesis that the decrease in layer accuracy at larger network widths arises from the model weights moving less, we quantified how much the hidden-layer weights change from initialisation to the end of training (i.e., when interpolation was reached). Let $\mathbf{W}_{(0),(1)}$ denote the weight matrix between the input layer ($\ell = 0$) and the first hidden layer ($\ell = 1$). The connection from input node p to hidden node n is denoted by $\mathbf{W}^{p,n}_{(0),(1)}$. We write $\mathbf{W}_{(0),(1)}(0)$ for this matrix at initialisation and $\mathbf{W}_{(0),(1)}(t)$ for the same matrix at the end of training. The element $W^{p,n}_{(0),(1)}(t)$ denotes the weight from input node p to hidden node n at time t, and $\mathbf{W}_{(0),(1)}(0)$ denotes its initial value.

For each model width, we compared $\mathbf{W}_{(0),(1)}(t)$ to $\mathbf{W}_{(0),(1)}(0)$ and computed several complementary measures at both the weight and node level. The first set of measures—the *mean* and *maximum absolute weight change*—captures how much individual weight magnitudes vary due to training. For a given connection weight $W^{p,n}_{(0),(1)}$, the absolute change is defined as

$$|\Delta W^{p,n}_{(0),(1)}| = \left|W^{p,n}_{(0),(1)}(t) - W^{p,n}_{(0),(1)}(0)\right|.$$

The mean absolute weight change (mean$|\Delta \mathbf{W}_{(0),(1)}|$) is computed over all entries in the hidden-layer weight matrix $\mathbf{W}_{(0),(1)}$. This provides a global measure of how much a typical weight in the layer shifts during training. The maximum absolute weight change (max$|\Delta \mathbf{W}_{(0),(1)}|$) indicates the most extreme change among all connections in that layer. The values reported in Table A.1 show the mean and standard deviation of these quantities across random seeds.

To assess node-level movement in weight space, we considered the incoming weight vector for node n, defined as:

$$\mathbf{W}^{n}_{(0),(1)}(t) = \left(W^{1,n}_{(0),(1)}(t) \ldots, W^{d,n}_{(0),(1)}(t)\right),$$

where d is the input dimensionality. For each node, we computed its L_2 displacement between initialisation and the end of training,

$$\left\|\mathbf{W}^{n}_{(0),(1)}(t) - \mathbf{w}^{n}_{(0),(1)}(0)\right\|_2,$$

which corresponds to the Euclidean distance between the initial and final weight vector. Larger L_2 displacements indicate that the node's incoming weight vector has moved further from its initial configuration. From the per-node displacements in a given hidden layer, we summarised the distribution by its median and mean; Table A.1 reports these statistics (again averaged over seeds, with standard deviations).

Weight change results The results, summarised in Table A.1 (with similar trends observed for Fashion-MNIST in Table A.2), show a clear and consistent pattern: as network width increases, the magnitude of weight changes decreases across all metrics. The narrowest network (35 hidden nodes) exhibits the largest mean and maximum absolute changes

(mean$|\Delta \mathbf{W}_{(0),(1)}| \approx 0.074$, max$|\Delta \mathbf{W}_{(0),(1)}| \approx 0.53$), while the widest network (5 000 hidden nodes) displays the smallest changes (mean$|\Delta \mathbf{W}_{(0),(1)}| \approx 0.0085$, max$|\Delta \mathbf{W}_{(0),(1)}| \approx 0.064$).

A similar trend is observed in the node-level L_2 displacements: the median displacement decreases from approximately 2.9 in the 35-node model to approximately 0.32 in the 5 000-node model, with the mean displacements showing the same monotonic decline. These results are consistent with the model entering a progressively more "lazy" training regime as width increases, in the sense that the hidden-layer weights remain closer to their initial values and the network behaviour can be increasingly well-approximated by a linearisation around the initialisation.

Table A.1: Weight change statistics for MNIST models of varying width.

Model width	**Mean $\lvert\Delta \mathbf{W}_{(0),(1)}\rvert$**	**Max $\lvert\Delta \mathbf{W}_{(0),(1)}\rvert$**	**Median L2 disp.**	**Mean L2 disp.**
35	0.0742 ± 0.0189	0.526 ± 0.360	2.911 ± 0.850	2.828 ± 0.769
50	0.0618 ± 0.0015	0.382 ± 0.013	2.391 ± 0.053	2.359 ± 0.060
100	0.0447 ± 0.0010	0.317 ± 0.020	1.710 ± 0.028	1.700 ± 0.040
200	0.0329 ± 0.0007	0.252 ± 0.017	1.257 ± 0.033	1.254 ± 0.029
250	0.0295 ± 0.0010	0.227 ± 0.025	1.115 ± 0.039	1.125 ± 0.040
500	0.0212 ± 0.0005	0.162 ± 0.011	0.800 ± 0.016	0.803 ± 0.020
1000	0.0163 ± 0.0005	0.141 ± 0.014	0.620 ± 0.019	0.622 ± 0.019
5000	0.0085 ± 0.0001	0.064 ± 0.006	0.324 ± 0.006	0.320 ± 0.006

Discussion The weight-change analysis provides clear evidence that wider networks update each parameter substantially less during training. Across all metrics—mean and maximum absolute weight change, as well as node-level L_2 displacements—the magnitude of movement decreases monotonically with width. This indicates that as the number of hidden nodes grow, individual nodes require progressively smaller adjustments for the model to achieve interpolation. The observed decline in layer accuracy at large widths is therefore consistent with reduced changes in the node weights—the node weights shift too little to develop strong class-specific behaviours.

Table A.2: Weight change statistics for Fashion-MNIST models of varying width.

Model width	**Mean $\lvert\Delta \mathbf{W}_{(0),(1)}\rvert$**	**Max $\lvert\Delta \mathbf{W}_{(0),(1)}\rvert$**	**Median L2 disp.**
300	0.035	0.396	1.31
350	0.033	0.412	1.26
400	0.032	0.410	1.15
500	0.029	0.366	1.05
800	0.023	0.318	0.85
1000	0.021	0.309	0.75
2000	0.016	0.267	0.57
5000	0.011	0.180	0.41