Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Sondre Foslien

On optimal ensemble learning using the concept of diversity and negative correlation

Master's thesis in Cybernetics and Robotics
Supervisor: Francesco Scibilia, Co-Supervisor: Massimiliano Ruocco
June 2019

NTNU
Norwegian University of Science and Technology

equinor

Sondre Foslien

On optimal ensemble learning using the concept of diversity and negative correlation

Master's thesis in Cybernetics and Robotics
Supervisor: Francesco Scibilia, Co-Supervisor: Massimiliano Ruocco
June 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Summary

A problem is found with a set of Kaggle competition submissions, regarding the incredible size of the solutions which are used to win. With this motivation, a study of how to create well-performing ensembles is conducted. A technique that could be relevant to the original problem is found, called Negative Correlation Learning, which induces diversity in ensembles of classifiers. A study into the effect of diversity in ensembles of varying sizes is performed. In this study, an effect is shown, where ensembles with few members show no effect of diversity, while ensembles consisting of slightly more members have a significant effect. This is shown on multiple datasets, for different sizes of datasets and varying problem difficulties. It is made an attempt at explaining the observed effect based on a new definition of a local specialist classifier, and the analysis of single classifier activations using the tool GradCAM. Both the explanation and the local specialist definition appears to act in line with the given expectations, which backs up both the definition and the explanation.

Sammendrag

Under et studie av et sett vinnende Kaggle-løsninger, ble det fastslått at det er et problem at de vinnende løsningene består av veldig store ansamlinger av klassifikatorer. Med dette som motivasjon ble det utført et studie av forskjellige eksisterende teknikker som kan brukes for å lage velfungerende ansamlinger. I dette studiet ble det funnet en teknikk som var relevant for problemet, nemlig negativ korrelasjonslæring. Negativ korrelasjonslæring indusererer mangfold blant klassifikatorene i en ansamling. Negativ korrelasjonslæring ble brukt til å studere effekten av mangfold blant ansamlinger med forskjellig antall klassifikatorer. I denne studien blir det vist at ansamlinger av få klassifikatorer har ingen effekt av mangfold i klassifikatorene. Ansamlinger av flere klassifikatorer ble vist å ha stor effekt av mangfold blant klassifikatorene. Denne effekten blir vist på flere dataset, med forskjellige dataset-størrelser og for klassifikasjonsproblemer med varierende vanskelighetsgrad. Videre blir det gjort et forsøk på å forklare denne effekten ved bruk av en ny definisjon av en lokal, spesialistklassifikator og analyse av klassifikatorene sine aktiveringer ved bruk av verktøyet GradCAM. Både forklaringen og definisjonen av en lokal, spesialistklassifikator oppfører seg slik forventninger skulle tilsi, hvilket støtter oppunder både definisjonen og forklaringen.

Preface

This thesis is the concluding part of my Master of Technology in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The project was carried out during the spring of 2019 at the Department of Engineering Cybernetics at NTNU, Trondheim in collaboration with Equinor, which provided the dataset used.

This thesis was finished with much help from my supervisor Francesco Scibilia and co-supervisor Massimiliano Ruocco. I am grateful for the interest you have taken in my work, and for all the help, feedback, and support you have given. Thank you for prioritizing my work in your busy schedules.

TABLE OF CONTENTS

Summary **i**

Sammendrag **ii**

Preface **iii**

Table of Contents **vi**

List of Figures **x**

Abbreviations **xi**

1 Introduction **1**
1.1 The problem 1
1.2 Kaggle 2
1.3 The dataset 2
1.4 Motivation 3

2 Basic Theory **5**
2.1 Machine learning 5
2.1.1 Neural networks 6
2.2 Bias and Variance 8
2.3 Cross-validation 8
2.3.1 Hold out validation 9
2.3.2 k-fold cross validation 9
2.3.3 Leave-one-out cross validation 9
2.4 Ensemble learning 10
2.4.1 Boosting 10
2.4.2 Bagging 10
2.4.3 Combination methods 11

3 Literature Review **13**
3.1 Diversity . . . 13
3.1.1 Diversity definitions . . . 13
3.1.2 Diversity measurements . . . 16
3.1.3 Achieving diversity . . . 21
3.2 Auto-ML . . . 26

4 Method **29**
4.1 Defining the problem . . . 29
4.2 Tools and design choices . . . 30
4.3 NCL implementation . . . 31
4.4 Experiments . . . 33
4.5 Analsis tools . . . 36

5 Results **39**
5.1 NCL . . . 39
5.2 GradCAM . . . 41

6 Discussion **53**
6.1 NCL . . . 53
6.2 GradCAM . . . 57
6.3 Answering the research question . . . 60

7 Further work **63**

8 Conclusion **65**

Bibliography **67**

LIST OF FIGURES

1.1 An example image from the dataset, including both HH and HV polarization. The image illustrates a ship. 3

2.1 A figure illustrating a typical layout for a neural network. Figure courtesy of Glosser.ca (2013) 6
2.2 A sketch illustrating the relationship between bias, variance and generalization error. Figure courtesy of Brown (2004). 8
2.3 An illustration of the training-process when using 5-fold cross validation. 9

4.1 An illustration of how the algorithm is implemented in Keras for n different neural networks. 32
4.2 An illustration of what GradCAM can show. Here a convolutional net is trained to distinguish cats from dogs, and the tool illustrates which features were important for the classifier to determine that the image consisted of a dog or a cat. From the activations, one can see that the face of the boxer was significant, and probably a distinguishing feature between the cats and the dogs the classifier noticed in the training set. The cat has very high activation on the belly, probably due to the stripes being a distinguishing feature. Figure courtesy of Selvaraju et al. (2016). 36

5.1 A graph showing the measured diversity of each ensemble for a given diversity emphasis. ρ, Q and Double Fault are diversity measures where a lower value means a more diverse ensemble. 40
5.2 A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on the C-CORE/Equinor dataset. 40
5.3 A graph showing the ensemble loss of a eight-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on the C-CORE/Equinor dataset. 41
5.4 A graph showing the ensemble loss of a four-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset. 42

5.5 A graph showing the ensemble loss of a 12-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset. 42

5.6 A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset. 43

5.7 A graph showing the ensemble loss of a eight-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset. 43

5.8 A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of ships and not-ships. 44

5.9 A graph showing the ensemble loss of a eight member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of ships and not-ships. 44

5.10 A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of dogs and not-dogs. 45

5.11 A graph showing the ensemble loss of a eight member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of dogs and not-dogs. 45

5.12 A figure showing a RGB interpretation of a SAR image, and two activation heatmaps. The first is from a classifier trained with no emphasis on diversity, while the second is from the eight-member ensemble with an emphasis of $1 \cdot 10^{-7}$ 46

5.13 A figure showing a RGB picture of a boat from the CIFAR10 dataset, and two activation heatmaps. The first is from a classifier trained with no emphasis on diversity, while the second is from the eight-member ensemble with an emphasis of $1 \cdot 10^{-7}$ 46

5.14 A figure showing a RGB picture of a dog from the CIFAR10 dataset, and two activation heatmaps. The first is from a classifier trained with no emphasis on diversity, while the second is from the eight-member ensemble with an emphasis of $1 \cdot 10^{-9}$ 47

5.15 A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset 48

5.16 A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of $1 \cdot 10^{-7}$. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset. 48

5.17 A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture B, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset 49

5.18 A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of $1 \cdot 10^{-7}$. The model is referred to as architecture B, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset. 49

5.19 A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture C, which means the model defined as convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2, dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset 50

5.20 A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of $1 \cdot 10^{-7}$. The model is referred to as architecture C, which means the model defined as convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset. . . 50

5.21 A histogram showing the activations of each pixel made by a model from the two-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset 51

5.22 A histogram showing the activations of each pixel made by a model from the two-member ensemble trained using NCL and a diversity emphasis of 0.02. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset 51

5.23 A histogram showing the activations of each pixel made by a model from the two-member ensemble trained using NCL and a diversity emphasis of 0.1. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset 52

Abbreviations

NCL	=	Negative Correlation Learning
DF	=	Double Fault
SAR	=	Synthetic Aperture Radar
RGB	=	Red, Green and Blue
CNN	=	Convolutional Neural Network
ILSVRC	=	ImageNet Large-Scale Visual Recognition Challenge
Radar	=	Radio Detection And Ranging
ESA	=	European Space Agency
CIFAR	=	Canadian Institute for Advanced Research
CV	=	Cross Validation
TPOT	=	Tree-based Pipline Optimization Tool

CHAPTER

ONE

INTRODUCTION

This chapter will introduce the problem which this thesis will try to address. That includes section 1.1 which will present the problem and the steps that will be taken in order to answer it. Section 1.2 will give the reader an insight into the Kaggle platform because the motivation for the thesis builds on this platform. Section 1.3 will present the main dataset which will be used for this project, and section 1.4 will give an insight into the motivation of the author for pursuing this question.

1.1 The problem

In January of 2018, Equinor, together with partner C-CORE, hosted a competition on Kaggle.com. In this competition, the international machine learning community was challenged to make the best classifier for discriminating between icebergs and ships in satellite Synthetic Aperture Radar (SAR) imagery. Many different techniques were applied, by the 3343 international teams who entered the competition. Many of these entries used the concept of ensemble learning, which is presented in chapter 2.4. The problem that will be tackled in this master thesis is related to the ever-growing complexity of the ensemble solutions.

A robust process for getting high solution accuracy on a submission on Kaggle is to combine more and more models in the solution. Combining large amounts of models is even referenced as a winning strategy by the Kaggle platform itself (Gorman (2016)). This increases the solution complexity. This is not a concern in a Kaggle competition since anything that gives increased performance is worth it. However, if it is desired to bring a Kaggle solution to an industrial and operational context, the solution size to performance ratio becomes a concern. Based on this problem, the objective of this thesis is to explore the concept of ensemble learning and how to make the best performing ensemble possible with a limited number of classifiers.

To be more specific, a set of base models will be ensembled. Given N of these base models, the goal is to optimize the ensemble for some number $m \ll N$ of base models which would result in the “best" combination of performance in the categories accuracy,

inference time, variance and bias.

To try to address this problem, this thesis will contain:

1. Some machine learning theory, including general theory, ensemble methods, and how to evaluate models. This will be necessary background theory to understand the rest of the thesis.

2. A literature review of what has been done in this field related to creating optimal ensembles. This will include a look into the concept of diversity and the world of Auto-ML.

3. Based on this review, some approach will be chosen to explore, which is related to making well-performing ensembles. The exploration will be related to ensemble size and how the chosen approach affects this.

1.2 Kaggle

Usmani (2017) gives an introduction to Kaggle, which will be presented here. Kaggle is a crowd-sourced platform made to attract, train, and challenge data scientists to solve data science and machine learning problems. A lot of data scientists are only theorists who rarely get to practice their art before being employed. The Kaggle team wants to challenge that, by providing a platform for hosting real-life data science and machine learning competitions. The competitions can be anything from projects with only an educational purpose to genuine problems from real-life companies. Participation is incentivized by rewards, which can be anything from job offers to monetary rewards. The most remarkable were Heritage Heath, who offered $\$3\ 000\ 000$ in prize money for solving their problem. Many of the prize pools hover around the $\$10\ 000$ to $\$50\ 000$ range.

The Kaggle community is significant, with over one million members, having submitted over four million models to different competitions (Usmani (2017)). Of these four million submissions, $47\ 799$ was submitted to the Equinor/C-CORE competition, with an average of 514 submissions each day. This makes it the most popular image-based competition of all time, and the seventh most popular of all Kaggle competitions (C-CORE (2018)).

1.3 The dataset

The dataset selection process is described in C-CORE (2018). The dataset is a state-of-the-art dataset consisting of Sentinel-1 imagery, mostly acquired along the east coast of Newfoundland and Labrador in 2017. With some additional supplements from other parts of the word, the final dataset contains 2553 icebergs and 2488 ships. The targets were classified using expert analysts.

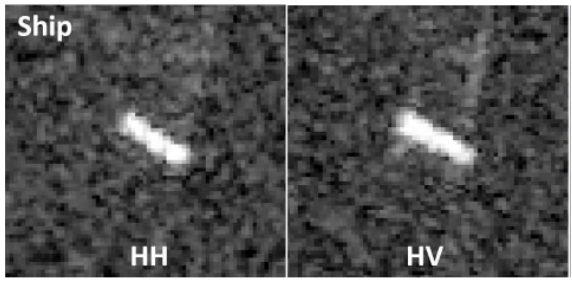


Figure 1.1: An example image from the dataset, including both HH and HV polarization. The image illustrates a ship.

Each sample is a 75×75 two-channel image containing the HH and HV channel from the Sentinel-1 SAR imagery. This is different from RGB images, as each pixel represents the intensity of the reflected signal the earth surface produces when illuminated by an energy beam. The images were collected using the Sentinel-1 Interferometric Wide Swath mode. There is no georeferencing information, but it contains metadata about the incidence angle of the target chip. An example image is shown in figure 1.1. The images were subject to several constraints: Each image should contain a target in open ocean, with only one target per image and no image borders or ambiguities. This resulted in over 5000 images, like what is shown in figure 1.1. This image is an RGB interpretation of the reflected intensities.

1.4 Motivation

The motivation for this project comes from the authors work relating to the Kaggle competition in an earlier project (Foslien (2018)). A review of the solutions given by the top 3 contestants showed three very complex solutions. One contestant reported inference times of about half an hour. On Kaggle, the only criteria the contestants are judged by are the model accuracy. There is no judgment of model complexity, inference time, bias, variance, or any other evaluation criteria. In the real world, this matters to the companies that host the competitions. If the winning solution should be employed in some industry setting, inference times of half an hour are often too long. In addition to this, the models use a lot of memory space in the computer and make it difficult to integrate it into pre-existing data processing pipelines. This sparked the motivation to try to explore the world of how to make optimal ensembles, in the hope that parts of the work could address some of the problems raised in this paragraph.

CHAPTER

TWO

BASIC THEORY

This section will present the basic theory needed to understand the rest of the thesis. The first section, section 2.1, will present the concept of machine learning and go deeper into the concept of neural networks, and convolutional neural networks. Then, in section 2.2, the concept of bias and variance will be presented. Section 2.3 will give an introduction of how to evaluate the performance of a machine learning classifier. The final section, section 2.4, will contain an introduction to some basic ensemble learning algorithms and ways of combining classifiers.

2.1 Machine learning

Alpaydin (2010) gives an introduction to machine learning. According to him, it is when a machine learns by itself to solve a problem. This is typically used when the algorithm needed to solve a problem is unknown. Then one can employ a machine learning algorithm to make the machine design the algorithm by itself. A typical example of this is the problem of classifying an e-mail as spam or not spam. The problem is known, the input is known, and in most cases, the desired output is known. However, an algorithm that can be used to map the input to the correct output is unknown. When facing this problem, a computer can be used to infer the algorithm. In many machine learning cases, the algorithm is extracted from large amounts of data. Machine learning is usually divided into three categories: supervised learning, unsupervised learning, and reinforcement learning. Unsupervised learning uses lots of data, but no ground truth to extract patterns that can be useful. Reinforcement learning interacts with some environment to learn the optimal way of controlling it based on some reward function. However, supervised learning will be the main focus of this thesis.

Mathematically, supervised learning can be described in the following way: It is an attempt at making an estimator $f(\boldsymbol{x}; W)$ that approximates an unknown function $g(\boldsymbol{x})$. This is done by using a training set of n pairs $\Theta = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_n}, y_n)\}$ and

modifying W such that

$$W(\Theta) = \arg\min_{W} \sum_{i=1}^{n} L(f(\boldsymbol{x_i}; W), y_i) \tag{2.1}$$

where $L(f(\boldsymbol{x_i}; W), y_i)$ is some loss function. The goal is that this f, parametrized by W, will generalize well to new, never before seen, pairs. This estimator can then be used to answer questions like $\boldsymbol{x_{test}} \in \mathbb{R}$ by generating a corresponding guess $y_{test} \in \mathbb{R}$. The estimator can be made with any machine learning algorithm, with varying levels of complexity and accuracy.

For further reading into the machine learning world, the reader is directed to Alpaydin (2010), Raschka (2015) or any other book on the subject. By far, the most heavily used machine learning algorithm in the Kaggle competition was neural networks. Therefore, neural networks will be a focus of this thesis and will be presented in the next section.

2.1.1 Neural networks

Neural networks were the most frequently used algorithm in the Kaggle competition. The reason for this is probably due to its historically high performance in image classification tasks. This can, at least partly, be attributed to the invention of convolutional neural networks (CNN), a type of neural network. CNN will be presented later.

According to Wu (1992), neural networks are heavily inspired by the human brain. The neural network is based on a large number of processing units, often called neurons, which are connected in such a way that they can learn from data they experience. In a dense network, the network is organized in layers, where a neuron has a connection to each of its predecessors in the earlier layer, as can be seen in figure 2.1. The first layer is an input layer, which is where the network is given its input. The last layer is the output layer, which produces the output. Finally, there is a hidden layer, which is used to add complexity so that the network can find complex patterns. One can use as many hidden layers as is necessary. Each neuron produces an output signal based on the sum of the weighted inputs from the last layer:

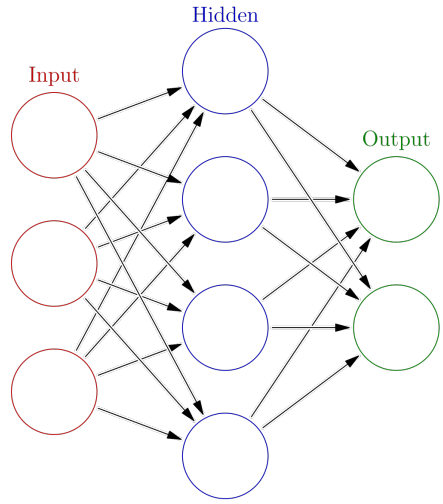


Figure 2.1: A figure illustrating a typical layout for a neural network. Figure courtesy of Glosser.ca (2013)

$$\hat{y_i} = f(\sum_{j=1}^{n} w_{ji} x_j) \tag{2.2}$$

where $\hat{y_i}$ is the output, w_{ji} is the weight of the connection from neuron j in the last layer to neuron i in the current layer, x_j is the output of neuron j in the previous layer and f is a nonlinear activation function. The weights w_{ij} are adjusted based on an algorithm called backpropagation. A loss function (i.e., mean squared error, binary cross entropy, or any other loss function) uses the network output $\hat{y_i}$ and the ground truth y_i to calculate some loss value. The gradient of this loss function is calculated and used to determine how to adjust each weight in order to minimize the loss.

$$\Delta w_{ij} = -\eta \frac{\partial L(\hat{y_i}, y_i)}{\partial w_{ij}} \tag{2.3}$$

where Δw_{ij} is the adjustment of w_{ij}, η is a constant between 0 and 1 determining the network learning rate and $L(\hat{y_i}, y_i)$ is the loss function given a ground truth y_i and a network output $\hat{y_i}$. This is done iteratively until the network hopefully converges towards a satisfying mapping f. This algorithm described here is called backpropagation. One can add optimizers to this algorithm, which adds effects like momentum or weight decay. Throughout this thesis, the optimizer RMSProp (Tieleman and Hinton (2012)) will be used.

For more information about neural networks and anything related, the reader is suggested reading Goodfellow et al. (2016), Wu (1992), LeCun et al. (2015) or any other introduction literature to neural networks.

Convolutional Neural Networks

CNN's was a breakthrough for neural network applications in image processing. A problem with densely connected neural networks is that the amount of learnable parameters quickly rises to enormous amounts with increasing input size and hidden layer sizes. A 28×28 pixel image results in $28 \cdot 28 = 784$ input neurons. A small 30 neuron hidden layer would then result in $784 \cdot 30 = 23520$ learnable parameters.

To alleviate this problem, the idea of CNN's was proposed. It is based on two basic ideas: local receptive fields and shared weights. Instead of connecting every neuron in the last layer to every neuron in the next, a small, local receptive field of neurons (for example 5×5 neurons) are connected to a single neuron in the next layer. This local receptive field moves with a certain stride between each step. These 5×5 weights are also shared, which further cuts down on the number of parameters. This is especially well suited for images since the shared weights will act as a filter, looking for the same patterns over the entire image. It exploits the spatial invariance of an image. To learn more about CNN's Goodfellow et al. (2016) is a good bet, but also Nielsen (2015).

CNN's sprung into popularity for image classification purposes in 2012 when AlexNet, a CNN architecture, (Krizhevsky et al. (2012)) and the SuperVision-team from the University of Toronto won the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) with an error of 15.4% (Russakovsky et al. (2015)). Second place had a 26.2% error rate. The SuperVision team was the first to use CNN's successfully in this contest, and it was the first significant proof that CNN's worked. In the 2013 ILSVRC, a majority of the contestants used CNN's for classification. This 2012 competition is seen as a breakthrough for CNN's, which has been immensely popular ever since.

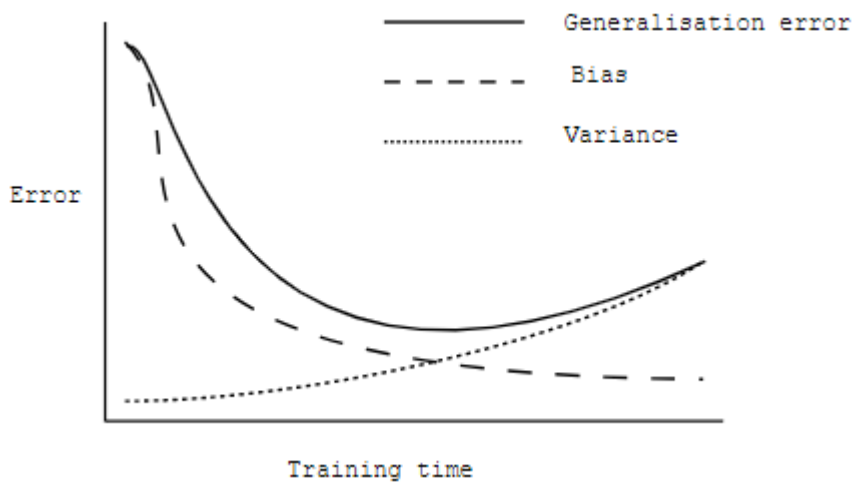


Figure 2.2: A sketch illustrating the relationship between bias, variance and generalization error. Figure courtesy of Brown (2004).

2.2 Bias and Variance

To understand how to minimize the generalization error of a machine learning algorithm, it is beneficial to look at how this generalization error can be decomposed. A decomposition of the generalization error was first presented by Geman et al. (1992), using a quadratic error function. He states that the error of a classifier C can be broken down into two parts: bias and variance.

Formally, this decomposition are given by the following:

$$\begin{aligned} E\{(f-d)^2\} &= E\{(f - E\{f\} + E\{f\} - d)^2\} \\ &= E\{((f - E\{f\}) + (E\{f\} - d))^2\} \\ &= E\{((f - E\{f\})^2 + (E\{f\} - d)^2 + 2(f - E\{f\})(E\{f\} - d\} \\ &= E\{(f - E\{f\}\} + (E\{f\} - d)^2 \\ &= var(f) + bias(f)^2 \end{aligned} \tag{2.4}$$

for some arbitrary testing point d. The intuitive explanation of bias is how close to the target the estimator f is, on average. Variance is a measure of how stable f is. A large variance and the estimator will tend to vary a lot more, jumping further from the average classifier output. One can also see figure 2.2. It presents a sketch of the relationship between error, bias, and variance given some training time.

2.3 Cross-validation

Cross-validation (CV) is one popular way of evaluating the performance of a machine learning classifier and will be used in this thesis. It is therefore presented here. Re-

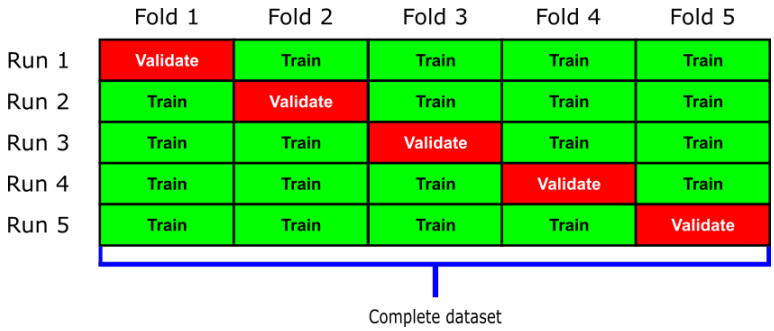


Figure 2.3: An illustration of the training-process when using 5-fold cross validation.

faeilzadeh et al. (2009) gives an introduction to the concept of CV. CV is a technique used for evaluating the performance of a machine learning algorithm. Evaluation is done by dividing the training-set into two parts, a training set, and a validation set. The main idea behind cross-validation is that each data point has to cross over between training and validation so one can validate against every point. The most basic form of CV is called hold out cross-validation.

2.3.1 Hold out validation

Hold out validation is when one uses one part of the dataset Θ for validation, holding it out of the training process. This has the disadvantage that it might give a skewed result as the data held out might be too hard or too easy. This is something Kaggle contestants are deeply afraid of, as it might have them submit their not best model. To mitigate these problems k-fold CV was introduced.

2.3.2 k-fold cross validation

In k-fold CV the training set Θ consists of n examples $\theta_i = (x_i, y_i)$ for $i \in \{1, 2, \ldots, n\}$, where x_i is a feature vector and y_i is its classification. This Θ is partitioned into k equal sized, disjoint sets. At all times, $k - 1$ of these sets are used for training (which will be called $\Theta_{j,1}$), and one is used for validation (which will be called $\Theta_{j,2}$). This is done k times, each time with a new set used as validation. This approach guarantees that every data point is validated against, and gives the best possible impression of the accuracy of the algorithm. The process is illustrated in figure 2.3, with $k = 5$.

2.3.3 Leave-one-out cross validation

A more specific version of k-fold CV is called leave-one-out CV. This is a particular case of k-fold, where the number of folds is equal to the number of samples. Only one sample is used for validation for each iteration. This is very computationally expensive but is often used when the data availability is low.

2.4 Ensemble learning

This section will introduce the most common ensemble learning methods used in the field of machine learning. *Boosting* and *bagging* will be given a quick introduction in sections 2.4.1 and 2.4.2. Following that, section 2.4.3 will present ways of combining the output of multiple models.

2.4.1 Boosting

Boosting is a concept first presented in Schapire (1990). In this paper Schapire tried to answer a question posed by Kearns and Valiant (1988, 1989): "Is strong model of learnability equivalent to weak model of learnability?" Strong learnability requires the output of a model to be of arbitrarily high accuracy. Weak learnability only requires "better-than-guessing" performance. This would require some method for boosting the low accuracy of the weak methods, and was therefore called the *hypothesis boosting problem*. This has later become known as *boosting*.

Zhou (2012) gives an intuitive explanation of the general boosting algorithm. Given a binary classification problem with a training set Θ, consisting of three parts θ_1, θ_2 and θ_3, a strong model of learnability should be made. Each of the three parts takes $1/3$ of the training set, and a model making random guesses has a 50% classification error on the problem. A classifier C_1 is trained, and it performs with 100% accuracy on θ_1 and θ_2, but not θ_3. This means that C_1 is a weak classifier. To correct the mistakes made by C_1 one can use the concept of boosting.

A new training set Θ' is made, with a higher concentration of samples from θ_3, which was where the model was wrong. Then a new model is trained. This might result in a classifier C_2 that performs well on θ_1 and θ_3, but not on θ_2. Combining C_1 and C_2 will result in 100% accuracy on θ_1, but varying performance on θ_2 and θ_3. The samples which the combined classifier misclassified is then used to create a third dataset Θ'', and a new model is trained using this dataset to create classifier C_3. Suppose this performs well on θ_2 and θ_3. By combining C_1, C_2 and C_3 a perfect classifier is made.

In this explanation of the boosting algorithm, there are still unspecified steps, like how to choose the new dataset and how to combine the classifiers. This is given by more specific instances of the boosting algorithm. An example of this is the AdaBoost algorithm, where the dataset update and the combination method of the classifiers are determined such that they minimize the exponential loss function:

$$l_{exp}(C|D) = \mathbb{E}_{\boldsymbol{x}\sim D}[e^{-f(\boldsymbol{x})C(\boldsymbol{x})}] \tag{2.5}$$

where the samples in Θ are drawn from the distribution D. For more information about AdaBoost, please see the original AdaBoost paper by Freund and Schapire (1997). Other iterations of the boosting algorithm includes LogitBoost (Friedman et al. (2000)), LPBoost (Demiriz et al. (2002)), XGBoost (Chen and Guestrin (2016)) and many more.

2.4.2 Bagging

Bagging, or Bootstrap Aggregating, as it is shorthand for, were first presented in Breiman (1996). Where boosting is what one would call a sequential ensemble method, bagging is

a parallel ensemble method. This means that many classifiers are trained at the same time. As explained in Zhou (2012), bagging exploits the fact that a combination of independent classifiers will result in a dramatic decrease in errors. Therefore, these classifiers should be as independent as possible. One could train each classifier on a separate part of the dataset, but this results in a small, unrepresentative dataset. Therefore one applies bootstrap sampling to make the data subsets. Given a training set Θ with m samples, T data subsets are created. Each subset consists of m samples, which are generated by sampling with replacement the original dataset.

2.4.3 Combination methods

The different ways of combining the outputs of multiple classifiers will now be presented. To do this, some of the explanations by Wolpert (1992) will be used. This paper built on the concept of generalizers, which earlier in this thesis has been referred to as machine learning algorithms (see section 2.1). This thesis will continue to use the name machine learning algorithm. It starts by defining the $\mathbb{R}^{n+1}$ space given by the training set Θ as the "level 0 space". This training set will be divided into k partitions, as described by k-fold cross-validation. These partitions make the sets $\{\Theta_{1,1}, \Theta_{2,1}, \ldots, \Theta_{k,1}\}$ and $\{\Theta_{1,2}, \Theta_{2,2}, \ldots, \Theta_{k,2}\}$, where $\Theta_{j,1}$ is the training set, and $\Theta_{j,2}$ is the test set of a certain partition. On this level 0 space a set of classifiers $\{C_1, C_2, \ldots, C_N\}$ will be trained. They are given by $C_l^j = L_l(\Theta_{j,1})$, where $L_l(\Theta_{j,1})$ is a machine learning algorithm trained on $\Theta_{j,1}$. This is called a "level 0 algorithm". For each of the k partitions, each classifier is used to generate predictions $\hat{y}_j^l = C_l^j(\Theta_{j,2})$. This results in the meta-level dataset on the form of $((\hat{y}_1^1, \ldots, \hat{y}_1^N, \hat{y}_2^1, \ldots \hat{y}_k^N), (y_1, \ldots, y_k))$. This is called the "level 1 space" and we now wish to generalize from Θ by looking at this new learning set. This results in a "level 1 algorithm". The ways to generalize from this set will noe be explained.

Voting

Voting is the simplest way of generalizing from the level 1 space. This can be done using majority wins, where every prediction $\hat{y}_i^l$, for all classifiers $l \in \{1, 2, \ldots, N\}$, is a vote for a label. The label with the most votes becomes the ensembles end classification $\hat{y}_i$. This can also be used on class probabilities, as first proposed by Ting and Witten (1999). Given a base classifier C_l which is to classify the sample x_i given m number of classes will result in the probability distribution:

$$\boldsymbol{p}^{C_l}(x_i) = (p^{C_l}(c_1|x_i),\ p^{C_l}(c_2|x_i),\ \ldots,\ p^{C_l}(c_m|x_i)) \tag{2.6}$$

As explained by Džeroski and Ženko (2004), voting is not the best way to do ensembling. No learning is taking place in the level 1 algorithm. The voting scheme remains the same no matter what classifiers are combined. There is the opportunity to use machine learning at the level 1 space, for the ensemble weighting to change depending on the input.

Machine learning approaches

As the level 1 algorithm, almost any machine learning algorithm can be used. Ting and Witten (1999) recommended to use multi-response linear regression. This is an adaptation

of linear regression for a classification problem with m classes, $\{c_1, c_2, \ldots, c_m\}$. For each class c_i, a regression problem is formed where a linear equation LR_i predicts a binary value which is 1 if the class label is c_i and zero if not. Given a new example x, $LR_i(x)$ is calculated for all i and the class label c_k is chosen such that $LR_k(x)$ is maximum. Another approach that has gained popularity is to use a neural network as the level 1 algorithm, especially when the level 0 algorithms are different variations of neural networks. This gives the benefit that it is possible to connect the level 0 and level 1 algorithms for a single pipeline. A neural network also has the possibility of finding highly complex relationships between the level 0 algorithms, which give it the opportunity of out-performing simpler algorithms like linear regression.

CHAPTER

THREE

LITERATURE REVIEW

This chapter will contain a literature review of research related to the problem presented in 1.1. Firstly, section 3.1 will present research done into making optimal ensembles by using the concept of diversity. Then, section 3.2 will present some Auto-ML approaches to the same problem.

3.1 Diversity

This section will present earlier attempts at using diversity to produce optimal ensembles. When combining classifiers in an ensemble, many will state that it intuitively makes sense to use as diverse base classifiers as possible. One will gain nothing from combining 1 million classifiers if they all give the same prediction. Combine a set of diverse classifiers, and they can, ideally, fill out each other's shortcomings. To explore the legitimacy of this assumption, a definition of diversity is needed.

3.1.1 Diversity definitions

Ambiguity

There have been many attempts at defining diversity in a mathematical framework. One of them comes from Krogh and Vedelsby (1994). Given the same supervised learning task as presented in section 2.1, an ensemble consisting of $i = \{1, 2, \ldots, N\}$ estimators f_i is created. The weighted ensemble average is given by

$$\bar{f}(\boldsymbol{x}) = \sum_i w_i f_i(\boldsymbol{x}). \tag{3.1}$$

Here w_i is the weighting of ensemble estimator f_i. Krogh and Vedelsby defines the *ambiguity* on input $\boldsymbol{x}$ as

$$a_i(\boldsymbol{x}) = (f_i(\boldsymbol{x}) - \bar{f}(\boldsymbol{x}))^2 \tag{3.2}$$

which gives the following definition of the *ensemble ambiguity*:

$$\bar{a}(\boldsymbol{x}) = \sum_i w_i a_i(\boldsymbol{x}) = \sum_i w_i (f_i(\boldsymbol{x}) - \bar{f}(\boldsymbol{x}))^2 \tag{3.3}$$

This equation gives a measurement of the disagreement among the N estimators. The ensemble ambiguity could also be called the diversity among the N estimators. Further, the single estimator error and the ensemble error is then defined:

$$e_i(\boldsymbol{x}) = (f_i(\boldsymbol{x}) - y_i)^2 \tag{3.4}$$

$$e(\boldsymbol{x}) = (\bar{f}(\boldsymbol{x}) - y_i)^2 \tag{3.5}$$

Using (3.3), (3.4) and (3.5) along with the fact that the weights $\sum_i w_i = 1$, the following equation can be derived:

$$\bar{a}(\boldsymbol{x}) = \sum_i w_i e_i(\boldsymbol{x}) - e(\boldsymbol{x}) \tag{3.6}$$

Defining $\sum_i w_i e_i(\boldsymbol{x}) = \bar{e}(\boldsymbol{x})$ gives

$$e(\boldsymbol{x}) = \bar{e}(\boldsymbol{x}) - \bar{a}(\boldsymbol{x}) \tag{3.7}$$

This equation can be interpreted as: The ensemble error equals the weighted average of all errors in an ensemble minus the weighted average of all the ambiguity among the estimators. Notably, this means that the ensemble error is always smaller than the weighted average error of all the single models. It also means the larger the ambiguity, the smaller the ensemble error is compared to the ensemble average.

Based on this conclusion, Krogh and Vedelsby goes on to propose different strategies to increase ambiguity among the ensembled estimators. It is proposed two different ways: using different training sets or mixtures of entirely different approximators.

Covariance

Another way of looking at diversity in ensembles is through the bias-variance decomposition first presented in section 2.2. According to Ueda and Nakano (1996) a further break down of the variance is possible, given a simple uniformly weighted combination of the ensembled estimators.

First, three concepts are defined: the average bias, the average variance, and the average covariance:

$$\overline{\text{Bias}}(X_0) = \frac{1}{M} \sum_{m=1}^{M} E_{\Theta_m^N}\{f_m(X_0; \Theta_m^N)\} - Y_0 \tag{3.8}$$

$$\overline{\text{Var}}(X_0) = \frac{1}{M} \sum_{m=1}^{M} E_{\Theta_m^N}\{(f_m(X_0; \Theta_m^N) - E_{\Theta_m^N}\{f_m(X_0; \Theta_m^N)\})^2\} \tag{3.9}$$

$$\begin{aligned}\overline{\text{Cov}}(X_0) = &\frac{1}{M(M-1)} \sum_m \sum_{m' \neq m} E_{\Theta_m^N, \Theta_{m'}^N}\{(f_m(X_0; \Theta_m^N) - \\ &E_{\Theta_m^N}\{f_m(X_0; \Theta_m^N)\})(f_{m'}(X_0; \Theta_{m'}^N) - E_{\Theta_{m'}^N}\{f_{m'}(X_0; \Theta_{m'}^N)\})\}\end{aligned} \tag{3.10}$$

Here the notation is the same as earlier, but X_0 is a random vector sample given to the estimator and Y_0 is the corresponding ground truth. $E_{\Theta_m^N}$ is the expectation over the particular training set Θ_m^N. For ease of notation, the expressions are shortened, using:

$$E_{\Theta_m^N}\{\cdot\} = E_m\{\cdot\}$$
$$f_m(X_0;\Theta_m^N) = f_m$$

giving

$$\overline{\text{Bias}} = \frac{1}{M}\sum_{m=1}^{M} E_m\{f_m\} - d \tag{3.11}$$

$$\overline{\text{Var}} = \frac{1}{M}\sum_{m=1}^{M} E_m\{(f_m - E_m\{f_m\})^2\} \tag{3.12}$$

$$\overline{\text{Cov}} = \frac{1}{M(M-1)}\sum_{m}\sum_{m'\neq m} E_{m,m'}\{(f_m - E_m\{f_m\})(f_{m'} - E_{m'}\{f_{m'}\})\} \tag{3.13}$$

where d is some testing point. Using these definitions alongside the fact that the ensemble f is a uniformly weighted combination and equation (2.4), Krogh and Vedelsby shows that it is possible to derive the following equation:

$$\text{Var}(\frac{1}{M}\sum_m f_m) = E\{(\frac{1}{M}\sum_m f_m - E\{\frac{1}{M}\sum_m f_m\})^2\} = (1-\frac{1}{M})\overline{\text{Cov}} + \frac{1}{M}\overline{\text{Var}} \tag{3.14}$$

Meaning that the ensemble variance can be divided into a variance portion and a covariance portion. One can also modify the bias:

$$\text{Bias}(\frac{1}{M}\sum_m f_m)^2 = (E\{\frac{1}{M}\sum_m f_m\} - d)^2 = (\frac{1}{M}\sum_m (E\{f_m\} - d))^2 \tag{3.15}$$

This gives the final expression for the ensemble mean squared error:

$$E\{(\bar{f} - d)^2\} = \overline{\text{Bias}}^2 + \frac{1}{M}\overline{Var} + (1 - \frac{1}{M})\overline{\text{Cov}} \tag{3.16}$$

This means that the error depends heavily on the covariance between the ensembled estimators.

Brown (2004) shows how to connect the covariance to ambiguity. Using the ambiguity term from (3.7), and the expected value of this term, Brown shows that

$$E\{\frac{1}{M}\sum_m (f_m - \bar{f})^2\} = \overline{\text{Var}} + \text{“deviations”} - \frac{1}{M}\overline{\text{Var}} - (1 - \frac{1}{M})\overline{\text{Cov}} \tag{3.17}$$

where “deviations” $= \frac{1}{M}\sum_m (E_m\{f_i\} - E\{\bar{f}\})^2$. This means that decreasing the covariance gives higher ambiguity which gives lower squared error. This legitimizes the assumption that higher diversity among ensembled estimators gives lower error.

3.1.2 Diversity measurements

Even though many will agree on the fact that diversity is an essential characteristic for improved ensemble accuracy, there exists no agreed preferred way of measuring diversity. Many measurements have been tried, and this section will sum up what has been presented in earlier literature.

Kuncheva and Whitaker (2003) gives a presentation of ten different diversity statistics that have been used to measure diversity. Kuncheva and Whitaker divide the measurements into two categories: pairwise and non-pairwise measurements. The pairwise measurements measure diversity between two models — the non-pairwise measurements measure the diversity among as many models as wanted models. Finding the diversity measurement for an ensemble consisting of more than two models when using a pairwise measurement, will require calculating the pairwise measurement for all possible combinations of two classifiers and then calculating the average.

The Q-statistic

The first measurement presented is the Q-statistic, which was first defined by Yule (1900). The Q-statistic is a pairwise measure. Given the set of classifiers $\{C_1, C_2, \ldots, C_N\}$, the training set $\Theta = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \ldots, (\boldsymbol{x_n}, y_n)\}$ and $l = \{l_{1,1}, l_{2,i}, \ldots, l_{N,1}, l_{1,2} \ldots, l_{N,n}\}$ such that $l_{j,i} = 1$ if C_i correctly recognizes θ_j and 0 otherwise, table 3.1 is defined.

	C_k correct	C_k wrong
C_i correct	N^{11}	N^{10}
C_i wrong	N^{01}	N^{00}

Table 3.1: A table illustrating the relationship of the variables used when measuring diversity among classifiers.

Here N^{ab} is the number of elements θ_j in the training set Θ, where $l_{j,i} = a$ and $l_{j,k} = b$. This gives the following definition of the Q statistic:

$$Q_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{N^{11}N^{00} + N^{01}N^{10}} \tag{3.18}$$

This equation implies that Q can vary between -1 and 1. If the classifiers tend to classify the same samples correctly, the Q value will approach 1. If they tend to misclassify different objects, the Q value will approach -1.

The correlation coefficient ρ

Sneath and Sokal (1973) first present the correlation coefficient. It is also a pairwise measurement for two classifiers C_i, C_k. It is defined as follows:

$$\rho_{i,k} = \frac{N^{11}N^{00} - N^{01}N^{10}}{\sqrt{(N^{11} + N^{10})(N^{01} + N^{00})(N^{11} + N^{01})(N^{10} + N^{00})}} \tag{3.19}$$

This equation uses the same definitions that are given in table 3.1.

The disagreement measure

The disagreement measure is the ratio between the number of training samples the classifiers disagree on and all the samples. This measurement has been used by Ho (1998) and Skalak (1996) for measuring diversity in machine learning ensembles. It is a pairwise measurement between two classifiers C_i, C_k, and is defined as follows:

$$Dis_{i,k} = \frac{N^{01} + N^{10}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{3.20}$$

It uses the same variables as is defined in table 3.1.

The double-fault measure

The double-fault measure is the ratio between the number of samples misclassified by both models and all the samples. This measurement was first used by Giacinto and Roli (2001) to measure and select the least related classifiers. It is also a pairwise measurement between two classifiers C_i, C_k, and is defined as follows:

$$DF_{i,k} = \frac{N^{00}}{N^{11} + N^{10} + N^{01} + N^{00}} \tag{3.21}$$

It uses the same variables as is defined in table 3.1.

The entropy measure E

The concept of entropy was first used to measure diversity by Cunningham and Carney (2000). It calculates the diversity of a set of classifiers $\mathcal{D}$ consisting of any number of classifiers. It is defined as follows:

$$E = \frac{1}{n} \sum_{j=1}^{n} \frac{1}{(L - \lceil L/2 \rceil)} \min\{l(\theta_j), L - l(\theta_j)\} \tag{3.22}$$

where L is the number of classifiers, and $l(\theta_j)$ is the number of classifiers that correctly classify the sample θ_j.

Kohavi-Wolpert variance

Kohavi and Wolpert (1996) derived an expression for the variability of the predicted class label y for $\boldsymbol{x}$.

$$variance_x = \frac{1}{2} \left(1 - \sum_{i=1}^{c} P(y = w_i|x)^2 \right) \tag{3.23}$$

where c is the number of classes and w_i is a class. Kuncheva and Whitaker (2003) shows that this results in the following expression for a binary classification problem

$$KW = \frac{1}{nL^2} \sum_{j=1}^{n} l(\theta_j)(L - l(\theta_j)) \tag{3.24}$$

This is proposed as another way of measuring diversity and is called the Kohavi-Wolpert variance.

The interrater agreement κ

The interrater agreement κ is a measurement of interrater reliability, first derived by Fleiss (1981). If one were to translate this to the terminology used in this thesis, a rater would be a classifier. κ is a measurement of the level of agreement between classifiers while correcting for chance. It is defined as follows:

$$\kappa = 1 - \frac{\frac{1}{L}\sum_{j=1}^{n} l(\theta_j)(L - l(\theta_j)}{n(L-1)\bar{p}(1-\bar{p})} \tag{3.25}$$

where $\bar{p}$ is the average individual classification accuracy.

The difficulty ϕ

Kuncheva and Whitaker (2003) defines the difficulty θ, which from now on will be referred to as ϕ since θ already is defined as the training set in this thesis. The difficulty measurement uses a discrete random variable X taking values in $\{0/L, 1/L, \ldots, L/L\}$. It denotes the number of classifiers in $\mathcal{D}$ that correctly recognizes $\boldsymbol{x}$. The difficulty is defined as the variance of X. Lower variance indicates greater diversity.

$$\phi = \mathrm{Var}(X) \tag{3.26}$$

Generalized diversity

This measurement was first proposed by Krzanowski and Partridge (1997). They define p_i as the probability that i of the L classifiers fail on a randomly chosen $\boldsymbol{x}$. Then $p(i)$ is defined as the probability that i randomly chosen classifiers will fail on a randomly chosen $\boldsymbol{x}$. On the case of two classifiers, maximum diversity is gained when the failure of one classifier is always accompanied by the success of the other. This results in the following definition:

$$GD = 1 - \frac{\sum_{i=1}^{L} \frac{i}{L} \frac{(i-1)}{(L-1)} p_i}{\sum_{i=1}^{L} \frac{i}{L} p_i} \tag{3.27}$$

Coincident failure diversity

The coincident failure diversity was also proposed by Krzanowski and Partridge (1997). It is designed to have a maximum value when every misclassification is unique.

$$CFD = \begin{cases} 0, & p_0 = 1.0 \\ \frac{1}{1-p_0} \sum_{i=1}^{L} \frac{L-i}{L-1} p_i, & p_0 < 1 \end{cases} \tag{3.28}$$

Kuncheva and Whitaker (2003) goes on to test these measurements, seeing if it is possible to find a correlation between the diversity measurements and ensemble accuracy. Four different experiments are conducted.

First a simulation is performed, using some Matlab script, to produce L binary classifier outputs for n objects so that the individual accuracy approximates p with a symmetric matrix of dependencies $Q = [Q_{i,k}]$, where $Q_{i,k}$ is the Q-statistic for classifier C_i and C_k. A large set of experiments were run, for different numbers of classifiers L and different approximate accuracy p. p_{max}, the maximum accuracy of a single classifier was found, along with the p_{maj}, the majority vote accuracy. This was calculated for a set of different Q_{av} in a uniform range from -1 to 1.

Q_{av} were plotted against $P_{maj} - P_{max}$, which showed a clear correlation between diversity and improvement in accuracy. It is worth noting that this is a result from both a highly synthetic dataset and synthetic classifier, and might not translate as well into the real world. Alongside this, the experiment did not use any other diversity measurement than the Q-statistic and therefore did not tell anything about how well the Q-statistic performs compared to other measurements. A point made by Kuncheva and Whitaker was that the worst ensemble was not made by combining identical classifiers. The worst ensemble was made by ensembling positively correlated but not identical classifiers.

A second experiment was conducted, where all possible combinations of $L = 3$ classifiers and $n = 30$ samples were enumerated for $p = 0.6$. This was done using the same Matlab script as the last experiment. Then diversity was measured using three different measurements for all iterations. This showed a clear increase in the majority vote accuracy as diversity was increased. However, it also showed that the worst ensemble was not the ensemble with minimum diversity. This agrees with the first experiment. Worst case were found for slightly positive Q-statistic, slightly positive ρ and $CFD = 0.5$. It is also interesting to note that there is a certain diversity threshold value, where for any diversity value greater than this threshold, a loss improvement is guaranteed.

A third experiment was conducted on a real (not-synthetic) dataset, the Wisconsin Diagnostic Breast Cancer database(Olvi L. Mangasarian (1995)). All possible partitions of the ten element feature set were made when splitting the features into partitions of size $4, 4, 2$ or $4, 3, 3$. Three classifiers were used and trained at a subset of the features. The classifiers strained were both linear and quadratic classifiers. They were ensembled using eight different combination methods. Kuncheva and Whitaker describes the results as showing there is no useful predictive value in diversity measures. Correlation between each diversity measure and the improvement in accuracy reaches a maximum of 47%, but is overall quite low. Maximum is reached for the difficulty measure ϕ.

Kuncheva and Whitaker (2003) explains the lacking results with the fact that very few classifiers were combined (only 3) and the overall improvement was small. It might have been too small for a diversity measure to be effective. Compared to the earlier experiments, this experiment was not as controlled, with no restrictions on individual accuracy or mutual dependency. This could be a reason for the experiment not quite lining up with earlier experiments.

The last experiment uses two datasets and three different ways of making the base classifiers for the ensemble. Bagging and using a simple linear classifier, bagging and using a small neural network as a classifier, and making weak classifiers using the linear

discriminant functions assigned with random coefficients. The number of base classifiers was set to $L = 9$, and the entire process was done 500 times with random splitting of the dataset into training and testing. The correlation between diversity and accuracy was again shown to be quite low, but it was highest for bagging using neural networks.

Another illustration was made, plotting the Q-statistic against accuracy, and marking the ensembles where p_{maj} were significantly higher than the average accuracy $\bar{p}$ of the ensemble. This illustration also showed a low correlation between diversity and an increase in p_{maj} over $\bar{p}$. This goes against what the theory showed in equation (3.7).

One criticism that can be made of the work done by Kuncheva and Whitaker is that a large number of base classifiers were never tested. Only the first experiment was tested for different amounts of base classifiers, and a larger L seemed to indicate a more significant effect of high diversity. This is something that definitely should have been followed up.

Lofstrom et al. (2007) were also critical of some of the work done in Kuncheva and Whitaker (2003). He says the experiments were conducted in a somewhat artificial setting, meaning that they did not use real datasets nor real classifiers.

Lofstrom et al. (2007) wanted to deal with these problems and therefore conducted an own set of experiments. Twenty neural networks were used to make 10 000 randomized ensembles. They also experimented with the use of a validation set, and the effects it could have, which was not used in Kuncheva and Whitaker (2003). Diversity among the 20 neural networks was targeted by using different network architectures, anything from 0 to 2 hidden layers with varying amounts of neurons. 10000 ensembles were made of varying sizes (2 - 20 members). This was tested on eight datasets from the UCI Repository (Dua and Graff (2017)).

A few different ways of illustrating the results were shown. First, the correlation between the diversity measures and test-set accuracy is compared to the correlation between training- and validation-set accuracy to test-set accuracy. This showed that no single diversity measure had a better correlation with test-set accuracy than the correlation with either validation- or test-set accuracy. The difficulty measure ϕ performed the closest to the validation- and test-set. Double fault, CFD and Kohavi-Wolpert were also good performers.

Lofstrom et al. argues that this measure might not be that interesting since one usually wants to choose one ensemble to apply to the unseen data. Would it be beneficial to choose an ensemble with higher diversity on training data? This is emulated by picking out the top 1% and the top 50% most diverse ensembles. The average top 1% ensemble performs worse than the average ensemble, but the average ensemble in top 50% performs better.

Another approach taken in Lofstrom et al. (2007) is to try to combine several measurements. A simple summation of the diversity measurements is done for every possible combination. The top 1% is extracted for each combination, and test-set accuracy is compared. For the classifiers trained without a validation set, training set accuracy resulted in the best top 1% ensembles, with a combination of accuracy and ϕ following close. For the classifiers trained with a validation set, the best performer was a combination of validation set accuracy and ϕ.

The thought of combining two or more measurements is an interesting proposal and makes sense when compared to the ambiguity decomposition in equation (3.7). Ensemble accuracy equals the average base classifier accuracy minus the ambiguity. This might also

explain why the top 1% most diverse ensembles perform worse than the average ensemble. Choosing the most diverse ensemble might result in a too big emphasis on diversity and results in a low average base classifier accuracy. Using a weighted combination of diversity and base classifier accuracy is more in line with the existing theory.

Another paper tackling the concept of measuring diversity is Tang et al. (2006). They base their article around 6 of the 10 diversity measurements from Kuncheva and Whitaker (2003): Disagreement (Equation (3.20), Double Fault (Equation (3.21)), KW (Equation (3.24)), interrater agreement (Equation (3.25)), generalized diversity (Equation (3.27)) and difficulty (Equation (3.26)).

The paper analyses these diversity measures based on the concept of margins, first introduced by Schapire et al. (1998) to explain the success of boosting algorithms. If $v_{i,l}$ is the total vote that the ensemble casts for label c on sample $\boldsymbol{x}_i$. The ensemble margin on sample $\boldsymbol{x}_i$ is defined as

$$m_i = v_{i,y_i} - \sum_{c \neq y_i} v_{i,c} \tag{3.29}$$

According to Tang et al., maximizing the minimum margin of an ensemble will result in the lowest generalization error. Then they go on to show that the minimum margin of an ensemble is not monotonically increasing with respect to any of the six diversity measurements they included. Higher diversity will often result in a lower minimum margin, but not necessarily. They argue that this is the reason for papers like Kuncheva and Whitaker (2003) not being able to find a clear correlation between diversity and accuracy.

A final way of measuring diversity which has been heavily cited by the community is brought up by Sharkey and Sharkey (1997b) and is described in terms of levels of diversity. They are defined in table 3.2. These levels are well suited for describing ensembles in more general categories but lack the accuracy and measurability of the earlier mentioned measures. Therefore, as an optimization criterion, the levels of diversity is probably not that well suited.

Even though nobody seems to have shown a direct correlation between ensemble diversity and ensemble generalization, the motivation of pursuing diversity among the base classifiers is still well accepted in the literature. The leading hypothesis is that the right formulation and measures for diversity are not found yet. Finding this definition and understanding how it affects ensemble performance remains a holy grail problem. (Zhou (2012))

3.1.3 Achieving diversity

While the literature seems not to have produced defining evidence that diversity results in better ensemble performance, there has been done much research related to how to achieve diversity among base classifiers in an ensemble. This section will consist of a presentation of the current state-of-the-art research done concerning achieving diversity in ensembles.

A survey was done by Brown et al. (2004) concerning methods for creating diversity. They define two categories to divide the diversity creating tactics: *implicit* and *explicit*. Which category the technique falls into depends on whether or not the technique takes into account information about diversity when constructing the ensemble. As an example,

Table 3.2: The levels of diversity as described by Sharkey and Sharkey (1997b)

Level 1	There are no coincident failures, meaning that no inputs resulted in more than one failing classifier
Level 2	There are some coincident failures, but the majority is always correct. More than one classifier can fail, but the function is always covered.
Level 3	When a simple majority vote will not result in the correct answer, but at least one classifier will always produce the right answer. It might be possible to weigh the outputs so that the right answer is always obtained.
Level 4	Failures are shared by all classifiers, resulting in an ensemble that never will be reliable. The classifier can however be used in conjunction with other ensembles to improve generalization.

the technique Bagging, as presented in section 2.4.2, be an implicit method, due to the random sampling of a training set. At no point is a measurement made to make sure that diversity is achieved.

On the other hand, there is boosting, presented in section 2.4.1, which is an explicit method. The training data is directly manipulated based on diversity in the base classifiers. Brown et al. explains the difference between explicit and implicit the following way: during the training of a classifier, an approximator will traverse the hypothesis space. The goal is to have classifiers that occupy different points in the hypothesis space. Implicit methods rely on randomness to make the classifiers traverse differently, while the explicit methods choose different paths for the classifiers.

Brown et al. cites Sharkey (1999) as having identified four different ways one can influence diversity. These categories were very specific to making ensembles of neural networks and did not encapsulate all techniques to be presented by Brown et al.. Three new categories were therefore made, which was supposed to encapsulate all possible techniques.

1. Starting point in hypothesis space.

2. Set of accessible hypotheses.

3. Traversal of hypothesis space.

Different techniques for ensuring diversity will now be presented and placed into the different categories proposed by Brown et al. (2004). A large part of this will be based on the survey done by Brown et al. (2004), but since this survey is quite old, this review will also consist of newer literature found by the author if this thesis.

Starting point in hypothesis space

This section will look at the different proposed techniques which ensure diversity by manipulating the starting point in hypothesis space. This means to initialize the classifiers with different initial conditions, which will make it more likely for them to traverse the hypothesis space differently. According to Brown et al. (2004), this is one of the most common ways of creating diversity, but also probably one of the worst. It is sometimes used as a benchmark to compare new techniques against.

Noel Sharkey (1995) explored the relationship between initial output weight vectors and the final backpropagation solutions in a neural network. They systematically varied the initial vector values used by the networks and trained the networks on the same fuzzy XOR task with the same dataset. The solutions were not found to be statistically independent, so they had converged towards very similar local optima.

This is also supported by Yates and Partridge (1996) and Parmanto et al. (1996), which ranked varying the initial conditions for neural networks dead last when comparing diversity making techniques with regards to the best generalization performance.

There has been little research done with regards to explicit enforcing of diversity using the initial conditions. The closest is Maclin et al. (1995), where they use competitive learning (Rumelhart and Zipser (1985)) to intelligently create initial network weights that are located initially far from the weight space origin, thereby potentially increasing the set of reachable local minima. This was showed to increase performance somewhat, especially for ensembles consisting of a larger number of networks.

Set of accessible hypotheses

This section will look at the different proposed techniques which ensure diversity by manipulating the set of hypotheses that are accessible to the classifier. This is done in one of two ways: restricting the accessible data for training, or changing the architecture of the classifier.

Restricting accessible data is probably the most popular way of creating diversity due to techniques like boosting and bagging. It is also probably the most widely investigated method. It is also possible to use k-fold cross validation for this (Krogh and Vedelsby (1994)), even though it is probably more relevant for evaluating classifier performance, as talked about in section 2.3

Another approach is to use distortion methods, meaning to pre-process some of the features to get a different representation of the sample. This opens a plethora of different ways to augment the data, so only a few documented approaches will be presented here.

In Sharkey and Sharkey (1997a) (which is explained not entirely right in Brown et al. (2004)), three neural nets were used for classification. They were called ANN1, ANN2, and ANN3 and worked on pressure data from a diesel engine. In addition to these three networks, two networks were used for transformation. Transformation A was an undercomplete autoencoder, a neural net trained to copy its input with hidden layers of smaller dimensions than the input and output layers (for more information, check Goodfellow et al. (2016)). The smaller dimension hidden layer representation was extracted and used as input to ANN1. ANN2 got its input from Transformation B, which was a randomly initialized neural network that had never been trained. ANN3 got the untransformed pres-

sure data. Sharkey and Sharkey showed that this ensemble produces better results than an ensemble of classifiers using only non-transformed data. Another example of transforming the features is Raviv and Intrator (1996), where they used a variation of Bagging, but added small amounts of Gaussian noise to the input vector. This was repeated multiple times with varying levels of noise, and the best performer was picked. This technique showed significant improvements over Bagging without noise.

More testing of implicit ways to create diversity was shown in Johansson and Löfström (2012). Here many different techniques were tested in different experiments, but notably restricting the number of accessible features for each ANN. They use something they call sparse nets, where a certain proportion of the connections in each layer were randomly removed, resulting in a different subset of features being used for each network. Ensembles of 15 neural networks were created, and through much testing, using sparse nets consistently resulted in the best performing ensembles. Another interesting point to note became apparent when Johansson and Löfström ran the same test for larger ensembles of 51 neural networks. Targeting diversity implicitly gave a much larger performance increase for larger ensembles compared to small ensembles when they are compared to the baseline model.

There are also more explicit ways of creating diversity using the restricting of hypotheses. Zenobi and Cunningham (2001) used a diversity measure to select a subset of features for each classifier. A feature was evaluated if it should be added to the subset of a classifier by using both a diversity measurement and the classifier error. If improvement to one of them comes at the cost of the other, then the feature is not added to that classifier's subset. The results showed that taking into account the diversity during feature evaluation gave a better result, but only when the ensembles were big enough. This is interesting, as it the same result as shown in Johansson and Löfström (2012). Zenobi and Cunningham proposes that this is because the classifiers become local specialists when they are created while taking into account diversity. A certain number of specialists is required before they become better than the "generalists" that are made when not accounting for diversity. A similar approach was taken by Oza and Tumer (2001) where they calculated correlations between the input features and classes. A classifier was trained on the features that have the highest correlation with a single class, making every classifier a class specialist. This was shown to produce better results compared to an ensemble of classifiers trained on features picked out by PCA. It would have been interesting to see this also compared to more popular ensemble techniques, like bagging or boosting.

Another example of an explicit method is called DECORATE as presented by Melville (2003). DECORATE is an acronym that stands for "Diverse Ensemble Creation by Oppositional Relabeling of Artificial Training Examples". The algorithm constructs diverse hypotheses by using additional artificially constructed training examples. Each classifier is trained on a separate set of oppositely labeled artificial training data. This reduces the correlation between the base classifiers, ergo increasing diversity. DECORATE was shown to perform better than AdaBoost, Bagging, and Random Forests, especially for small datasets. An experiment was also conducted, which showed the performance of DECORATE for varying ensemble sizes. Unfortunately, this is not compared to the other algorithms or a baseline, so there is no way of telling if DECORATE performs even better as the ensembles grow larger, as the techniques from Johansson and Löfström (2012) and

Zenobi and Cunningham (2001).

Changing the set of accessible hypotheses can also be done by manipulating the network architectures. Partridge (1996) did a study of the effect the number of neurons had on generalized diversity, as defined in equation (3.27). The number of neurons was shown to have about the same effect as varying the initial weight seed, and very much lower than varying the training set. Unfortunately, they only varied the number of hidden neurons from 8 to 12, which, to be frank, is not enough to draw any conclusions.

CNNE is an algorithm that manipulates the network architectures to achieve diversity. It uses incremental learning, negative correlation learning, and a constructive approach to designing the network architecture. Networks are trained iteratively using the concept of negative correlation. After a user-defined amount of epochs, the network is tested, and if a specific criterion is fulfilled, more neurons are added. If that criterion is not fulfilled, the ensemble is tested for another criterion which determines if another NN should be added to the ensemble. This loops until some performance measure are reached. The algorithm was tested on seven real-life classification datasets and outperformed other ensemble techniques(Bagging, Arcboost, Adaboost, for example) consistently. It also performed well on the Mackey–Glass Chaotic Time Series prediction problem (for more info see Glass and Mackey (2010)), which is a regression problem as opposed to a classification problem.

There has also been proposed genetic algorithms for finding diverse network architectures. An example of this is the ADDEMUP (Accurate anD Diverse Ensemble Maker giving United Predictions) genetic algorithm proposed by Opitz and Shavlik (1996). The algorithm starts with an initial population of trained neural networks and uses genetic operators to create new networks continually. It keeps the networks that are highly accurate and disagrees with each other as much as possible. This technique was shown to trade blows with bagging on four different datasets.

Hypothesis space traversal

If the search space is defined by the network architecture and the training data provided, a strategy for achieving diversity is to modify the traversal of the learning algorithm trough this space. A way one could do this is to add a penalty term to the loss function. A penalty term that penalizes low diversity, for example. This is first proposed by Rosen (1996), and extended by Liu and Yao (1999a). Rosen first proposed adding the penalty term to the loss-function, using the loss function

$$e_i = \frac{1}{2}(f_i - d)^2 + \lambda \sum_{j}^{i-1} c(j,i)(f_i - d)(f_j - d) \tag{3.30}$$

where $c(j,i)$ defines which networks i and j should be decorrelated. This was taken one step further by Liu and Yao, which proposed to train the networks in parallel and changed the penalty term to

$$p_i = (f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) \tag{3.31}$$

This was first called Cooperative Ensemble Learning System (CELS) (Liu and Yao (1999b)) and later renamed to Negative Correlation Learning (Liu and Yao (1999a)). It

has been shown to outperform other ensemble techniques consistently. The reason for this is explained by Brown (2004). They show how negative correlation directly controls the covariance of the ensemble. Which, trough the bias-variance-covariance-decomposition ((3.16)), is an explanation of why negative correlation learning can affect the generalization error of an ensemble.

As Brown said himself: *"When training a simple ensemble, we only minimize the errors of the individual networks, and therefore only explicitly influence the bias component in the ensemble. However, when training with NC, we use the individual error plus the Ambiguity term as a regularisation factor. The expected value of the Ambiguity term provides the missing second half of the ensemble error that is not included in simple ensemble learning. It, therefore, can be seen that the reason for the success of NC is that it trains the individual networks with error functions which more closely approximate the individual network's contribution to ensemble error, than that used by simple ensemble learning."* (Brown, 2004, p. 59)

This only applies to regression problems, due to the ambiguity decomposition being formulated based on a regression problem. It can be used for classification, as long as they are reformulated as a regression problem. In later years, the ambiguity decomposition has been formulated for the 0-1 error function (Chen (2008)), which made it possible to introduce an ambiguity term for that loss function. This ambiguity term can be used as the penalty term in the loss function, which makes it possible to do negative correlation learning for classification problems with 0-1 error functions as well.

These results were used by Wang et al. (2010) to create what they called AdaBoost.NC. He extended the ambiguity term from Chen (2008) to all classification tasks, including multi-class domains. This is then used in the framework of AdaBoost, by adding a diversity measure to the weights of the training examples along with the misclassification information used in the original AdaBoost. This information is then used to build the next classifier for the ensemble. This means that AdaBoost is closer to negative correlation learning with sequential training of the classifiers. The algorithm is tested and compared to NCL on ten different datasets. It was shown to trade blows with NCL, outperforming NCL on 50% of the datasets, and being worse on the other half.

Negative correlation learning has shown to perform very well and increase generalization over standard training for ensembles. However, Chan and Kasabov (2005) argues that a problem facing the user of NCL is its very slow training time. It requires a communication line between the component networks to measure the network diversity compared to the others. This slows down training and makes it a hassle to use third-party code. Therefore Chan and Kasabov proposes a method they call Negative Correlation Learning via Correlation-Corrected Data (NCCD). Correlation-Corrected (C-C) data is transformed training data that induce negative correlation when the networks are trained on it. This data is transformed again periodically to reflect the changes done to the networks during the training process.

3.2 Auto-ML

Auto-ML is short for automatic machine learning. The University of Freiburg has a machine learning lab that has published a book about the concept (Hutter et al. (2018)). Ac-

cording to them, Auto-ML aims to make decisions about hyperparameters, regularization methods, training procedures, and any choice which has to be made when using a machine learning algorithm, something which is automatically chosen based on the data. The Auto-ML algorithm will determine the approach that works best for this particular application. There have been attempts at making Auto-ML algorithms that design neural network ensembles. They differ from the techniques from the earlier section, in that they do not necessarily search after the concept of diversity.

An example of this is the very recent Autostacker algorithm, as presented by Chen et al. (2018). Autostacker automatically searches for the optimal stacked ensemble for a certain dataset. The user has to specify the maximum number of layers for each network and the maximum number of neurons in each layer. They can also specify the machine learning algorithms that can be used in each classifier. AutoStacker uses a basic evolutionary search algorithm to find the best set of hyperparameters for the problem at hand. A set of N complete pipelines is created at random. Half of them are mutated by changing one hyperparameter at random, and the other half is crossed over with the mutated set. This creates a new set of N pipelines. These total 2N pipelines are evaluated, and the N best performing are kept. This loops until the algorithm are finished. This was tested against other algorithms, more specifically Random Forest (Breiman (2001)), Tree-based Pipeline Optimization Tool (TPOT) (Olson et al. (2016)) and AutoSklearn (Feurer et al. (2015)). AutoStacker was better than Random Forest 100 % of the time, 12 of 15 times it was better than TPOT and 9 of 15 times it was better than AutoSklearn. It is unfortunate that they did not test AutoStacker against any other ensemble technique. It is unfair game to compare a classifier using a single model against one that uses 200, which is the case of AutoStacker.

Other than this, the author has not been successful at finding any other AutoML approach that makes ensembles. As Chen et al. (2018) talked about, the Auto-ML community has done much research related to automatically making single classifiers. These single classifiers can, of course, be put into an ensemble if that is wanted, but each classifier is designed to be optimal on its own. Autostacker is an ensemble method by default and designs each classifier to be optimal in an ensemble. The fact that Autostacker is the only algorithm to pursue this seems to indicate there is more research that can be done concerning AutoML ensembles.

CHAPTER
FOUR

METHOD

In this section, the methodology which will be used to answer the different aspects of the original problem given in the introduction will be presented. The design choices made in the experiments will be argued and explained. This section consists of five parts. First, the problem will be further defined in section 4.1. Secondly, the resources and tools used will be explained in section 4.2. Then, in section 4.3, some of the considerations regarding the code used will be presented. In section 4.4, the set of experiments that will be done is explained and argued. Finally, section 4.5 will explain the ways the results will be analyzed. All the code referenced here and that is used in this project, will be made available at the author's GitHub at `https://github.com/sondrfos/TTK4900--NCL/`.

4.1 Defining the problem

The research question posed in section 1.1, is quite a big question: how to produce the optimal ensemble when stacking multiple classifiers together. As the literature review has shown, there is a lot of different ways of tackling this problem. One could take the Auto-ML route, maybe use diversity or take a completely different route. Therefore, this question needs to be narrowed down, to stake out a clearer path to follow.

It was decided that it would be interesting to explore the effects of diversity on a homogeneous ensemble of neural networks. Exploring this could give valuable insights into how to create ensembles, and more specifically creating more optimal ensembles of limited size. It would also be highly relevant to the Kaggle submissions as homogeneous ensembles of neural networks often are both a used and winning strategy. Based on what was read in the literature review, and the original research question, it was decided to pursue the following two questions:

1. Is there a clear indication that ensembles with more members have a larger benefit from enforcing diversity?
2. Can we pose some hypothesis as to why this is the case, and back it up?

4.2 Tools and design choices

To produce diversity in the ensemble a strategy presented in the literature review will be used. Since a controlled and scientific study of the effects of diversity was desired, it was decided that it would be more interesting to pursue an explicit way of enforcing diversity into an ensemble. This way, the effect of diversity could be controlled, and this would lend itself to a more systematic review of the diversity effect.

These considerations narrow down the possible techniques that could be used from the literature review. A possible approach would have been to use a genetic algorithm, maybe something like ADDEMUP or DECORATE. Another possibility would be to use a diversity measure to select a subset of features for each classifier given by Zenobi and Cunningham (2001) or using sparse nets, as shown in Johansson and Löfström (2012). Unfortunately, none of these techniques give direct control over the amount of diversity in the ensemble. Because of this, it was decided to use negative correlation learning to experiment with diversity in ensembles for this thesis. Adaboost.NC was not chosen due to the unnecessarily complicated pipeline it would create for our experiments. NCCD did not directly control diversity in the ensemble as much as was wanted and was therefore not used.

There exists no openly available implementation of negative correlation learning on the internet. NCL was designed in quite a different time, before the time of GitHub and other code repository sites, which might be an explanation for why no implementation exists. Therefore, negative correlation learning will have to be implemented from the ground up using modern frameworks and tools.

The deep learning framework of choice will be Keras (Chollet et al. (2015)), because it is the framework the author has the most experience using. The use of Keras is somewhat of a weak point of this thesis. A framework with lower level control, like for example, Torch (Paszke et al. (2017)), would have been beneficial, but this was not realized before too late in the work. The reason why more low-level control would have been beneficial, will be pointed out in section 4.3, which talks about the implementation of NCL.

The primary dataset which will be used for testing is the C-CORE/Equinor dataset, as presented in section 1.3. The reason for this is the fact that the idea for the project sparked from the C-CORE/Equinor Kaggle competition, where this dataset was used. It is also a state-of-the-art binary classification image dataset, and will result in relevant results. Still, any observation made during testing on the C-CORE/Equinor dataset will require confirmation on other datasets. To confirm it will mean to do the same experiments another image classification dataset, like for example CIFAR10 (Krizhevsky and Hinton (2009)).

It was decided to use 4-fold-cross-validation, to make the results as representable as possible. 30% of the dataset was kept as a test-set and were used to test the performance of the classifier in the final results, which will be presented in this thesis. Further, it was decided to keep the learning rate consistent between all experiments done in this thesis. This is because tuning learning-rate would lead to another factor to tune during experimentation. If raw performance were important to the experiments, one would have to tune the learning rate. However, since this experiment is most concerned about the relative performance of the best performing ensemble compared to the baseline with no diversity emphasis, it was decided to keep the experiments with as few factors to modify as possible.

4.3 NCL implementation

This section will present how the NCL algorithm was implemented. The design choices made will be argued for, and it will all be illustrated by a figure. It will also consist of an explanation of why Keras was not the optimal framework for implementing NCL.

First, the theory behind NCL will be repeated. As explained in the literature review, NCL is a way to change the traversal of the learning algorithm through hypothesis space, so it converges towards more diverse spaces. Changing the traversal is done by adding a penalty term to the cost function, which penalizes low diversity. In NCL this penalty term is given by the equation:

$$p_i = (f_i - \bar{f}) \sum_{j \neq i} (f_j - \bar{f}) \tag{4.1}$$

where f_i is the classifier output of the classifier currently training, $\bar{f}$ is the average classifier output and the sum loops over all classifiers except the one in training currently. This penalty term is added to the cost function (which in our case will be mean squared error), weighted by a parameter which can be changed to give differing emphasis on diversity. This results in the following cost function:

$$e_i = \frac{1}{2}(f_i - d)^2 + \lambda p_i \tag{4.2}$$

This cost function, in addition to the fact that all networks should be trained at the same time, results in quite the challenge. How can one train multiple networks at the same time, which are dependent on each other?

Solving this challenge is where another framework than Keras would have come in handy. Keras has somewhat limited functionality when it comes to defining a cost function. One can define a custom cost function, but there is no way for this function to be dependent on some dynamic variable which is not part of the model that's currently training. This results in a significant restriction on how to implement negative correlation learning, which is entirely dependent on using the other models' dynamic outputs as input to its cost function. Implementing NCL would have been more natural in some lower level frameworks like Torch or even TensorFlow, but due to the author's limited knowledge of these frameworks, and the limited time at hand, it was chosen to pursue some functional implementation in Keras.

The final implementation for n networks is illustrated in figure 4.1. To overcome the restrictions on the cost function, all the models were connected for the cost function to be able to access all predictions from every model. A single input-layer feeds forward the input to each neural network, which is to be trained to correlate negatively with each other. $n - 1$ networks are frozen at all times (illustrated by the blue boxes in figure 4.1). The non-frozen network is the network that is currently trained, and its output is used to calculate the mean squared error. Its output is also used, along with all the other network outputs, to calculate the penalty term. This cost is then used to adjust the weights of the non-frozen network according to the backpropagation algorithm. The steps described runs once, before which network that's un-frozen rotates, and a new input batch is fed into the networks. This loops until all the networks converge towards some optimum.

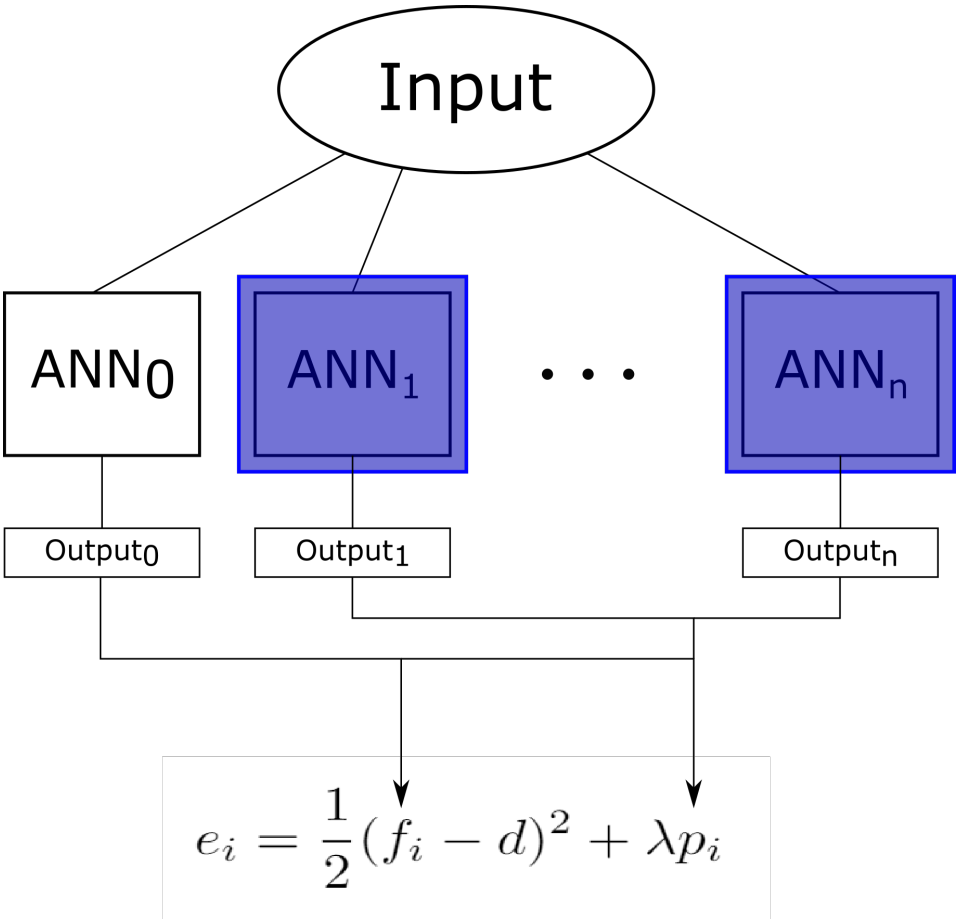


Figure 4.1: An illustration of how the algorithm is implemented in Keras for n different neural networks.

This implementation works and does what it is designed to do: negative correlation learning. It comes, however, at the cost of some performance. Freezing and unfreezing is quite a resource intensive task in Keras. It was also implemented a version of checkpointing, to make sure that the best version of each of the n networks always was kept. A patience-factor of 7 epochs were defined. If a network did not gain performance on the validation set when training, the checkpoint routine was started. In this routine, the network which did not improve was trained 7 epochs, consecutively. If it were not able to improve during these 7 epochs, the network was rolled back to the version from before the training started. This checkpoint routine, along with freezing and un-freezing, impacted the performance. This could be improved by using a framework with lower level control, which would not require such a hacky way to implement the training process.

4.4 Experiments

A set of experiments was conducted to gain some insight into the effect of diversity on ensembles. They will be described in this section, along with the reasoning behind them.

The literature review showed that some earlier experiments had indicated that larger ensembles benefited more from diverse ensembles. To investigate this was chosen as a goal for this thesis. Would the effect still be prevalent in this application? Further, this thesis will contain some analyzation to try to explain this effect. None of the proposed explanations for this effect has been backed up by any evidence in the literature as of yet.

All the experiments conducted are presented in table 4.1. This selection was made to be able to answer the research question. A strong foundation was a goal, so first, two experiments at full dataset size were conducted for ensembles of size 2 and of size 8. These two experiments will give a good representation of the effect diversity has on small ensembles compared to larger ones. Then the training dataset size was lowered. Lowering the dataset size was done with two goals in mind:

1. To study the effects of enforcing diversity in situations where the available data is limited.

2. To speed up the training process, to be able to get more interesting results quickly.

The same experiments were conducted again, along with tests for both ensembles with 4 members and 12 members. These experiments could help illustrate if the effect of diversity scaled for smaller and larger ensembles. Then the same experiments were repeated, but this time using the CIFAR10 dataset. The reason for this is to determine if the effects observed on the C-CORE/Equinor dataset were proprietary to that dataset, or if it is a more general result that is more likely to apply to many datasets. The CIFAR10 dataset will be modified to be a binary classification problem. This modification will be done by sampling the dataset, so it consists of 600 images of a predetermined class, and 600 random images of the other classes in the dataset. The task of the classifier will be to determine if the image is of the predetermined class or not (1/0). Experiments will be done using both ships and dogs as the predetermined class. This was an itneresting choice of classes since ships are considered the easiest to classify in the CIFAR10 dataset, while the dogs are considered the hardest (Antonio (2018)). These experiments will only be done with

two and eight members. The reason for not running for 12 and four members is mostly a time-saving measure. It is still considered enough experiments to show that the effect is not only prevalent on the C-CORE/Equinor dataset.

Table 4.1: The different experiments using NCL conducted in this thesis to answer the research question posed.

Experiment no.	Number of members	Number of runs	Dataset	Dataset size
1	2	10	C-CORE/Equinor	100 %
2	8	10	C-CORE/Equinor	100 %
3	2	10	C-CORE/Equinor	10 %
4	4	10	C-CORE/Equinor	10 %
5	8	10	C-CORE/Equinor	10 %
6	12	10	C-CORE/Equinor	10 %
7	2	10	CIFAR10 (ship)	10 %
8	8	10	CIFAR10 (ship)	10 %
9	2	10	CIFAR10 (dog)	10 %
10	8	10	CIFAR10 (dog)	10 %

The following neural network architectures were used during the experiments presented in table 4.1:

- 2 member ensemble:
  - convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2
- 4 member ensemble:
  - convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2
  - convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [256, 128], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2
- 8 member ensemble:
  - convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0
  - convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0
  - convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2
  - convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [256, 128], dropout: 0
  - convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0
  - convolutional layers: [32, 64, 128, 256], dense layers: [256, 128], dropout: 0.2

  - convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2

- 12 member ensemble:
  - convolutional layers: [64, 128, 256], dense layers: [128, 64], dropout: 0
  - convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0
  - convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0
  - convolutional layers: [64, 128, 256], dense layers: [128, 64], dropout: 0.2
  - convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2
  - convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [128, 64], dropout: 0
  - convolutional layers: [32, 64, 128, 256], dense layers: [256, 128], dropout: 0
  - convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0
  - convolutional layers: [32, 64, 128, 256], dense layers: [128, 64], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [256, 128], dropout: 0.2
  - convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2

This list specifies the unique part of each ensemble member. The general outline for each member is as follows: The input is passed through a batch normalization-layer, to improve performance and convergence time. This normalized input is passed on to a convolutional layer of size as specified in the list above. Kernel size on all convolutional layers are 3×3, and the activation function used were always rectified linear units. In between each convolutional layer is a max pooling layer with pool size 2×2 and strides of 2×2. The top layers are as specified by the dense layers in the list above. They also use the activation function rectified linear units. If dropout were specified, it was applied in between each layer.

The reasons for using the architectures as specified in the list above were due to different requirements and considerations. When working with neural networks, architectures like VGG16 are often used in the literature since it is well known and used by many. Ideally one would use VGG16 here as well, but due to the sheer size of that network, it was deemed infeasible to train 12 of them at the same time. Something relatively small and lightweight was wanted. Therefore the solutions proposed in the Kaggle competition were studied, and the small, lightweight architectures were extracted. Then a small initial test was performed, and the two architectures which performed the best in an ensemble using completely regular training were picked. That way, it was known that any increase in performance would be an actual improvement on the best solution. These two architectures were then used in the two-member ensemble. For all larger ensembles, slight modifications were done to the two original architectures before they were added to the ensemble.

It is also of note that the diversity emphasis will need to be shrunk as the ensembles glow large. As the number of ensemble members increases, it becomes harder to find a local minimum that is both highly accurate, while still being diverse compared to 7 or 11 other members. This local minimum will be easier to find in the case where backpropagation only has one other member to consider. Therefore the emphasis will be lowered as the ensembles grow in size. The goal will be to find the optimum diversity emphasis for each configuration. Optimum diversity will be found by trial and error.

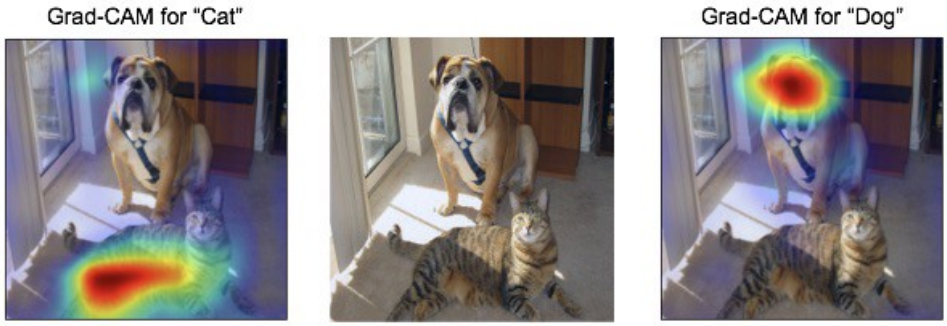


Figure 4.2: An illustration of what GradCAM can show. Here a convolutional net is trained to distinguish cats from dogs, and the tool illustrates which features were important for the classifier to determine that the image consisted of a dog or a cat. From the activations, one can see that the face of the boxer was significant, and probably a distinguishing feature between the cats and the dogs the classifier noticed in the training set. The cat has very high activation on the belly, probably due to the stripes being a distinguishing feature. Figure courtesy of Selvaraju et al. (2016).

4.5 Analsis tools

This section will present the tools an methods used for analyzing the results gained from running the experiments discussed in the last section. The goal of the analyzation is to answer the two questions posed in section 4.1 and the main research question.

The first question from section 4.1 has been shown in earlier literature. To analyze this, the ensemble loss after stacking will be plotted for multiple different diversity emphasizes. These plots will make it possible to see at what diversity emphasis the current ensemble performs the best, how the performance compares to the diversity emphasis and that the emphasis found is quite close to the optimum.

These ensemble loss plots from different experiments can be compared to see if there is some difference in the effect of diversity on larger ensembles compared to smaller ones.

With regards to question two, this has been speculated about in earlier literature but has never backed up by any experiment or theory. This thesis will build on this proposed explanation and try to back it up. This will be done by using a tool called Grad-CAM (Selvaraju et al. (2016)).

GradCAM is a tool used for visual explanations of why a CNN concluded as it did. It calculates the average activation of each pixel in the image and illustrates that as a heatmap on top of the original image. An illustration of this is showed in figure 4.2. GradCAM can give valuable information as to why a neural network acts as it does. This tool will be used to inspect the classifiers to back up the explanation to question 2.

The proposed hypothesis as to why large ensembles benefit more from diversity is that when an classifier is trained with an emphasis on diversity, they become local specialists. The author has not been successful at finding a definition of the term "local specialist", so in order to figure out if our classifiers are local specialists, a definition of the term will need to be established. This definition will be presented here, as it is considered crucial to have a clear definition of the term before an investigation is conducted.

When thinking about a specialist, one often thinks of something or someone that is devoted to a single and specific task. When transferring this to the field of machine learning, one could think of a specialist as a classifier that's devoted to a more specific task than the typical generalist classifier. When combining this with the term "local", one arrives at a classifier that good at a specific job and uses localized features. In this thesis, GradCAM will be employed to try to notice if classifiers trained with a more significant emphasis on diversity is more focused towards more specific attributes and localized features in the images they classify. This will be done by analyzing the activations of each classifier, and seeing if some classifiers have more concentrated, sharper spikes in their activations compared to other classifiers. If that is the case, especially as the average over the entire dataset, then this indicates that an emphasis on diversity in large ensembles creates local specialists.

One would expect a generalist to have larger blobs of medium to high activation, as it cares about bigger sections of the image. A local specialist would have a few areas with high activation, and many pixels with low activation. This might not always be the case on an individual image, as one could imagine an image having many versions of the same feature that the local specialist reacts to. Imagine an eye specialist classifier and a close up image of a spider. It might react to all the eyes on the spider, while the generalist does not activate on such a large area. One would, however, expect the general activation outline described earlier in this paragraph to be present when one averages over many images.

CHAPTER

FIVE

RESULTS

This chapter will present the results found when doing the experiments and analyzation, which was described in chapter 4. First, the plots gained from the experimentation with ensemble size and diversity will be presented in section 5.1, before a more in-depth look into the behavior of the networks in section 5.2 will be examined. The results will be discussed in section 6.

5.1 NCL

The results from the first experiment from table 4.1 will be presented first. It was considered important to see that the NCL implementation worked as intended, and therefore, a handful of diversity measurements were used as a sanity check of the effectiveness of the algorithm. This is seen in figure 5.1.

Continuing, the effect of diversity emphasis on the ensemble was plotted. This can be seen in figure 5.2. This is the result after much experimentation with both lower and higher diversity emphasis to find values that were high enough that they affected the traversal in hypothesis space, while still being small enough not to make the diversity emphasis so large that it dominated the actual classification loss. The result is the average of ten runs, in an attempt to counteract randomness in the results.

Following this experiment, the same experiment was done, with an increased amount of members in the ensemble. This time, eight members were used. The ensemble loss plotted against the diversity emphasis of this ensemble can be seen in figure 5.3. This result is again an average of 10 runs for the results to be as representative as possible. The emphasis is lower in this experiment, which is explained in the last chapter but also will be commented in the discussion later.

To see how the effect of diversity scales, experiments using both an in-between amount of members and a larger amount of members were conducted. The dataset size was now dropped to 10 % of the original, to speed up the training process and to see how enforcing diversity might act differently when the dataset-size is smaller. The ensemble loss plotted against the diversity emphasis for an in-between size of four members can be seen in figure

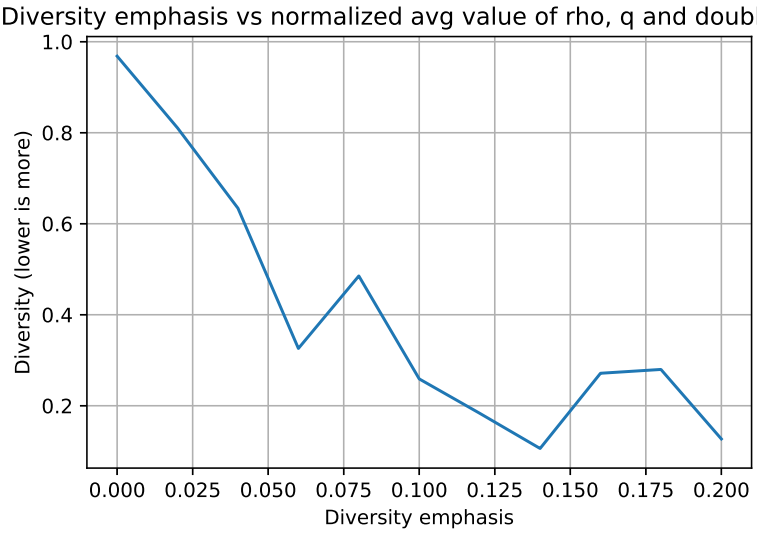


Figure 5.1: A graph showing the measured diversity of each ensemble for a given diversity emphasis. ρ, Q and Double Fault are diversity measures where a lower value means a more diverse ensemble.

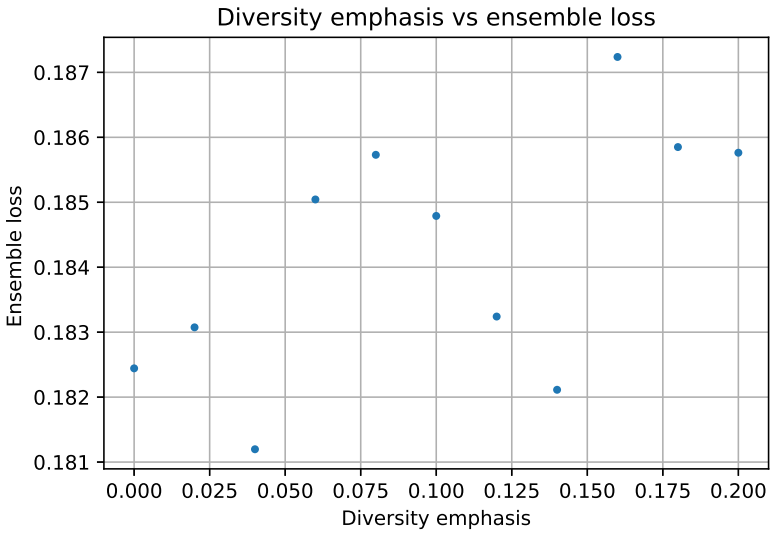


Figure 5.2: A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on the C-CORE/Equinor dataset.

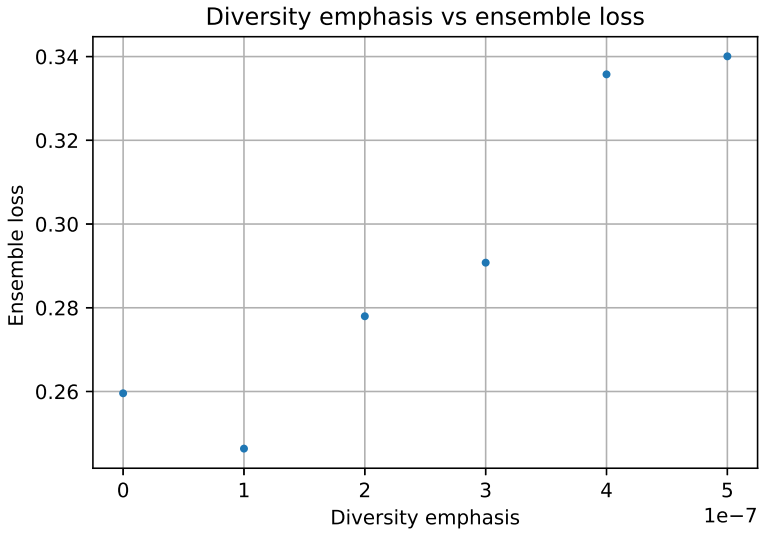


Figure 5.3: A graph showing the ensemble loss of a eight-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on the C-CORE/Equinor dataset.

5.4. The ensemble loss plotted against the diversity emphasis for an even larger size of 12 members can be seen in figure 5.5. Both of the plots are a result of averaging ten runs.

An experiment using the ensemble consisting of eight and two members were also conducted when using only 10% of the training data. The ensemble loss plotted against the diversity emphasis is shown in figures 5.6 and 5.7. Again, this is the result of 10 runs.

The same experiments were to be conducted on another dataset, in order to see if the effect was specific to the C-CORE/Equinor dataset or a more general observation. CIFAR10 was chosen as the second dataset and was adapted to be used in a binary classification setting, as described in section 4. First, the class to be classified was ship. 10% of the dataset was used, both a two-member ensemble and an eight member-ensemble were trained, and the results were compiled as the average of 10 runs. These results can be seen in figures 5.8 and 5.9.

The second experiment on another dataset was done with the same CIFAR10. The dataset was again adapted to a binary classification setting, but this time, the class to be classified were dog. 10% of the dataset was used, both a two-member ensemble and an eight member-ensemble were trained, and the results were compiled as the average of 10 runs. These results can be seen in figures 5.10 and 5.11

5.2 GradCAM

This section will show some of what could be considered typical activation heatmaps produced by GradCAM when analyzing the models trained with differing diversity emphasis. This section will also contain histograms of pixel activations, to look for trends or patterns

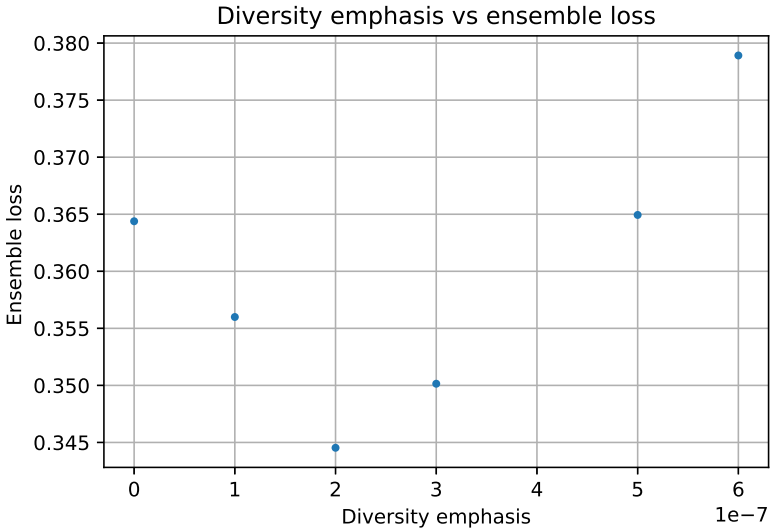


Figure 5.4: A graph showing the ensemble loss of a four-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset.

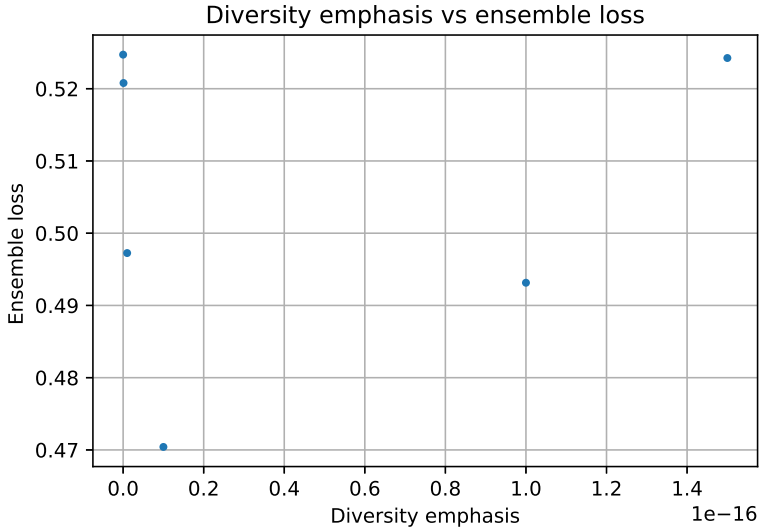


Figure 5.5: A graph showing the ensemble loss of a 12-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset.

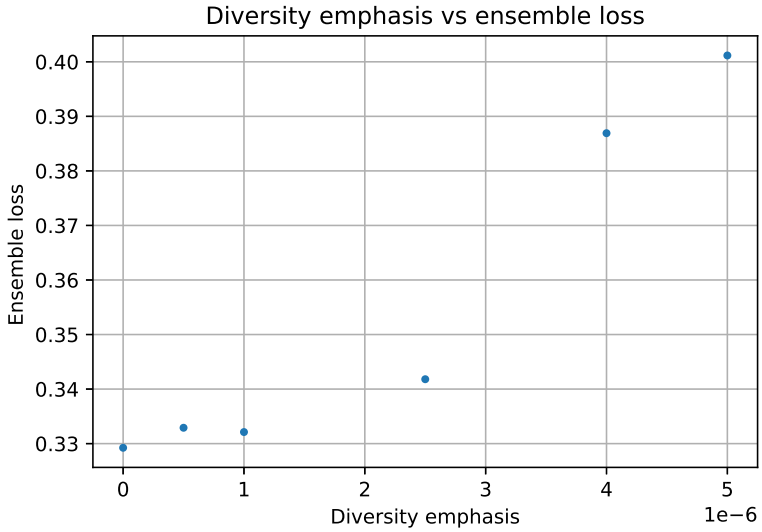


Figure 5.6: A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset.

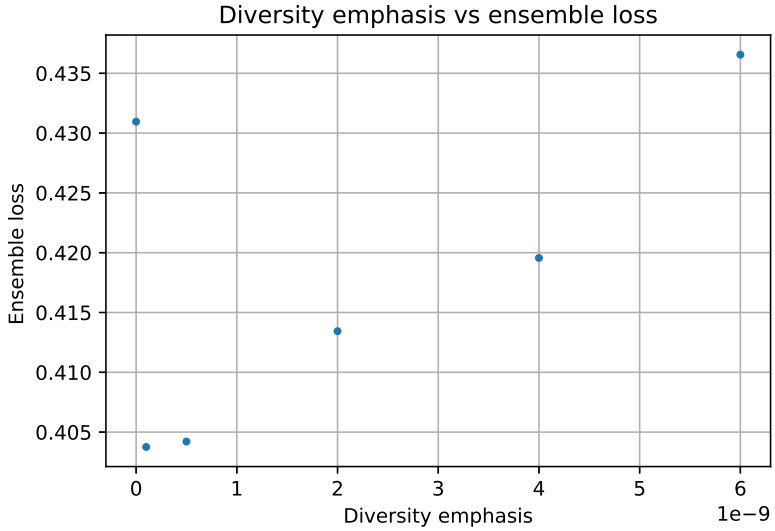


Figure 5.7: A graph showing the ensemble loss of a eight-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the C-CORE/Equinor dataset.

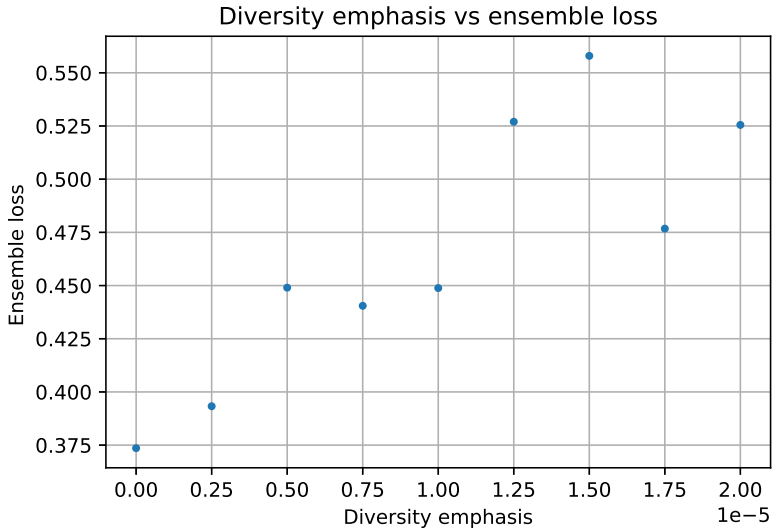


Figure 5.8: A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of ships and not-ships.

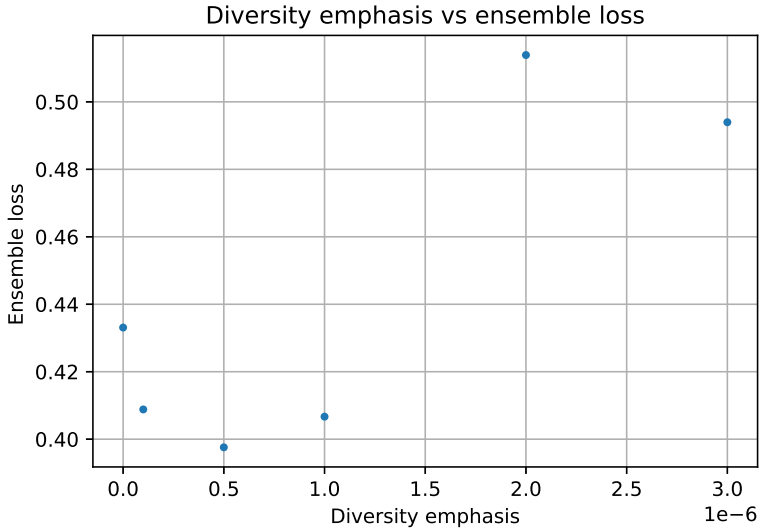


Figure 5.9: A graph showing the ensemble loss of a eight member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of ships and not-ships.

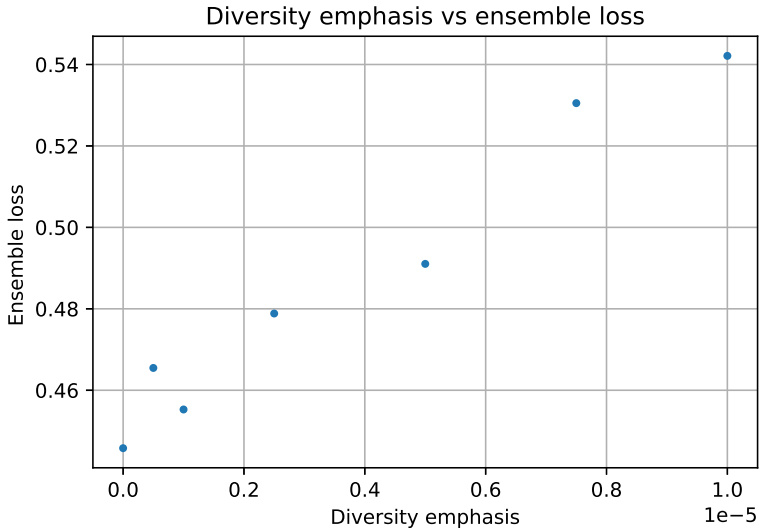


Figure 5.10: A graph showing the ensemble loss of a two-member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of dogs and not-dogs.

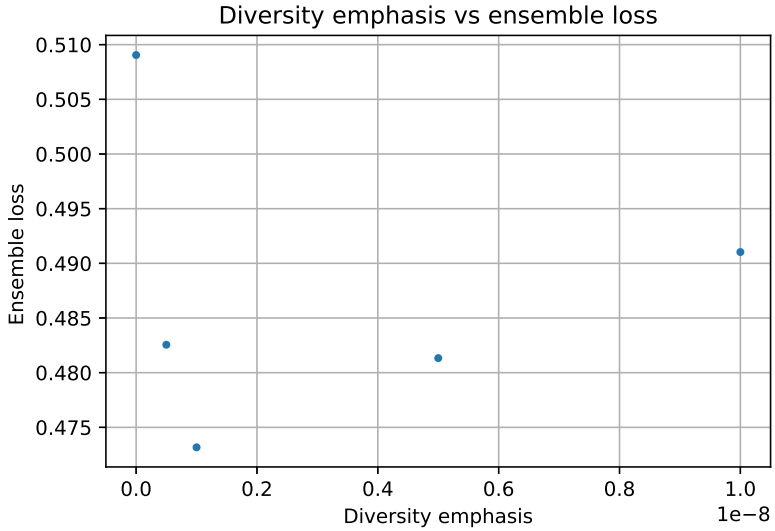


Figure 5.11: A graph showing the ensemble loss of a eight member ensemble plotted against the diversity emphasis used in the NCL algorithm trained on only 10% of the CIFAR10 dataset adapted to a binary classification problem of dogs and not-dogs.

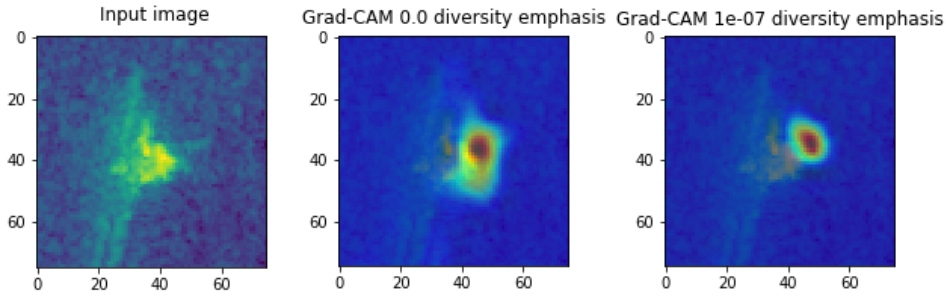


Figure 5.12: A figure showing a RGB interpretation of a SAR image, and two activation heatmaps. The first is from a classifier trained with no emphasis on diversity, while the second is from the eight-member ensemble with an emphasis of $1 \cdot 10^{-7}$

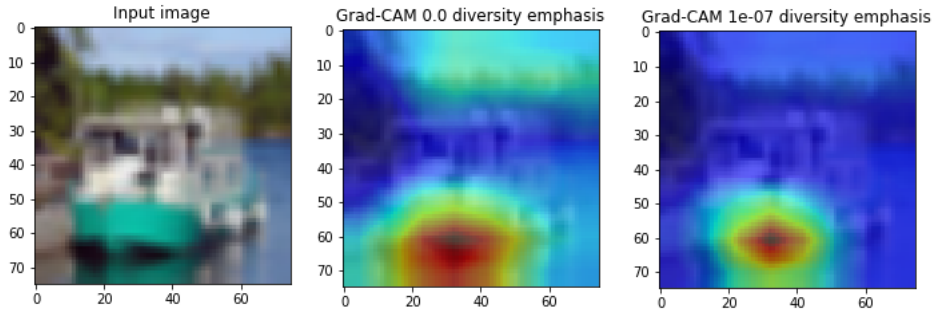


Figure 5.13: A figure showing a RGB picture of a boat from the CIFAR10 dataset, and two activation heatmaps. The first is from a classifier trained with no emphasis on diversity, while the second is from the eight-member ensemble with an emphasis of $1 \cdot 10^{-7}$

which can give information about the effect of diversity.

The typical activation patterns will be shown for three different networks. The first is trained on the C-CORE/Equinor dataset and will show activations on an RGB interpretation of a SAR-image. The model comes from the ensemble consisting of eight members and is defined as: convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. The activation heatmap can be seen in figure 5.12.

The second is a model trained on the CIFAR10 dataset to spot ships and not-ships. Again the model comes from the ensemble consisting of eight members and is defined as: convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. The activation heatmap can be seen in figure 5.13.

The third is a model trained on the CIFAR10 dataset to spot dogs and not-dogs. Again the model comes from the ensemble consisting of eight members and is defined as: convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. The activation heatmap can be seen in figure 5.14.

Finally, this result section contains histograms of pixel activations. A few different

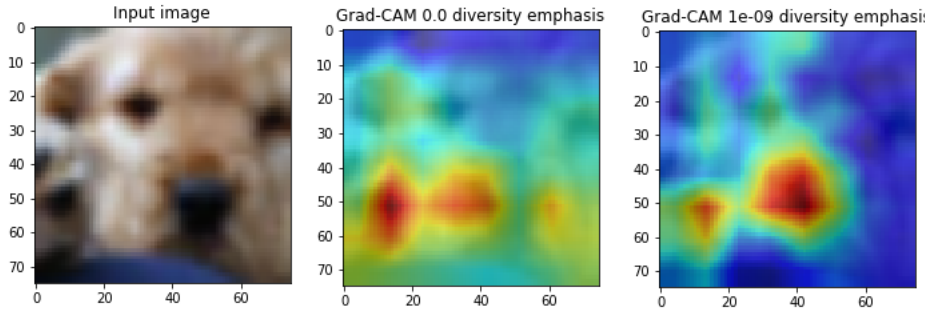


Figure 5.14: A figure showing a RGB picture of a dog from the CIFAR10 dataset, and two activation heatmaps. The first is from a classifier trained with no emphasis on diversity, while the second is from the eight-member ensemble with an emphasis of $1 \cdot 10^{-9}$

plots will be presented here, to look for consistencies across ensembles with differing amount of members, but somewhat equal diversity emphasis. The histograms presented here represent all pixel activations across all images in the dataset for all ten training runs for each model for all four cross-validation folds. This will give an as true as possible representation of the differing pixel activation patterns for each amount of diversity emphasis. The first histogram can be seen in figure 5.15, where the pixel activations for convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2 in the eight-member ensemble trained with 0.0 emphasis on diversity is shown. Continuing, figure 5.16 contains pixel activations for the same model architecture, but trained with a diversity emphasis of $1 \cdot 10^{-7}$.

The same activation histograms were also plotted for two other model architectures, namely convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2 and convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2. They are illustrated in figure 5.17 and 5.19 for 0.0 diversity emphasis and 5.18 and 5.20 for $1 \cdot 10^{-7}$ diversity emphasis.

The model activations were also studied in the two member ensemble. This can be seen in figure 5.21, 5.22 and 5.23 with diversity emphasis 0.0, 0.02 and 0.1, respectively. This is for architecture A, which refers to convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2.

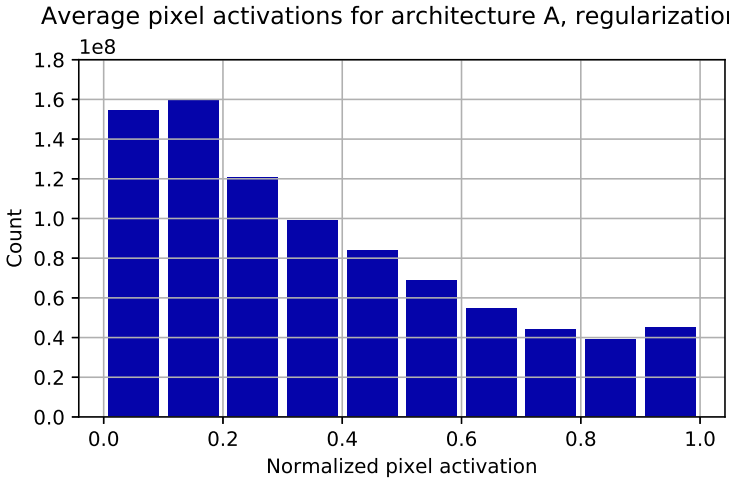


Figure 5.15: A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset

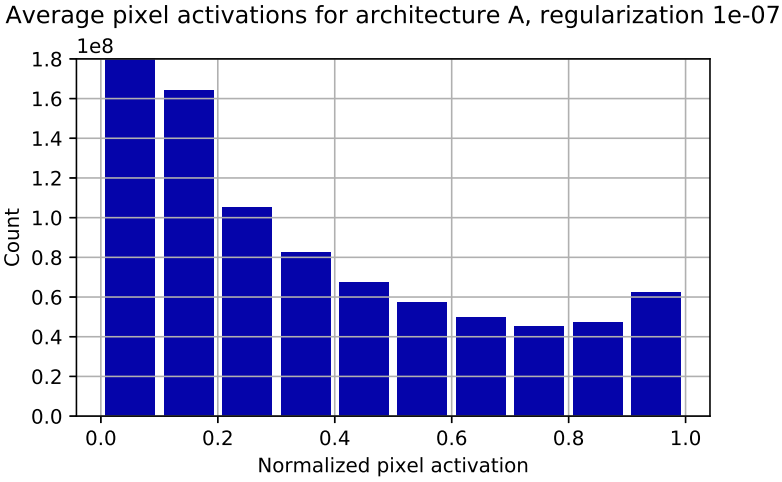


Figure 5.16: A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of $1 \cdot 10^{-7}$. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset.

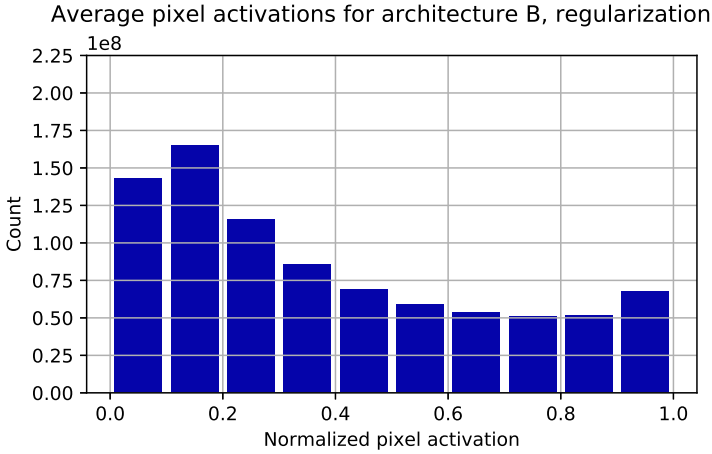


Figure 5.17: A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture B, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset

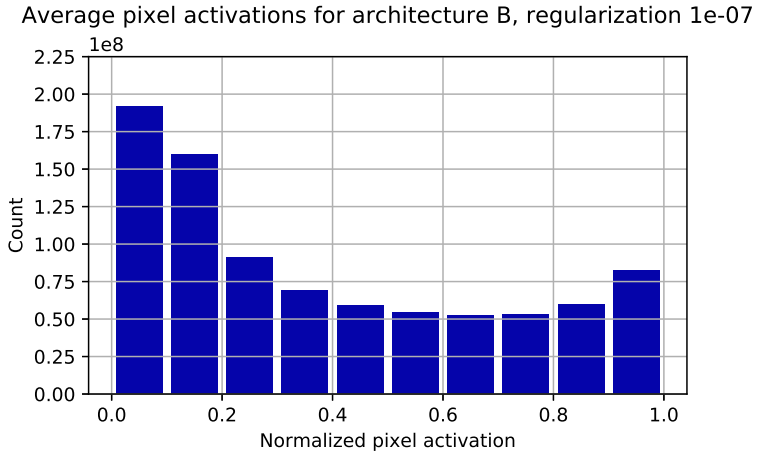


Figure 5.18: A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of $1 \cdot 10^{-7}$. The model is referred to as architecture B, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset.

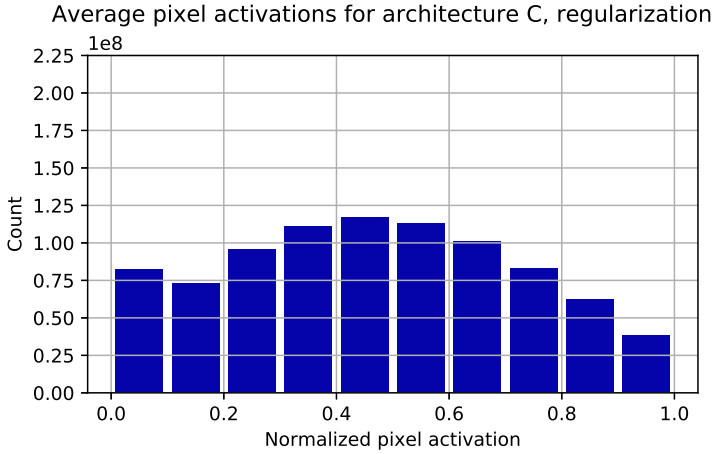


Figure 5.19: A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture C, which means the model defined as convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2, dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset

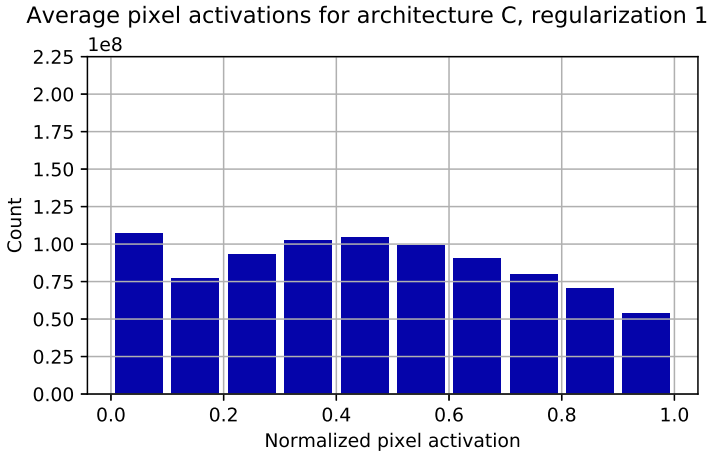


Figure 5.20: A histogram showing the activations of each pixel made by a model from the eight-member ensemble trained using NCL and a diversity emphasis of $1 \cdot 10^{-7}$. The model is referred to as architecture C, which means the model defined as convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset.

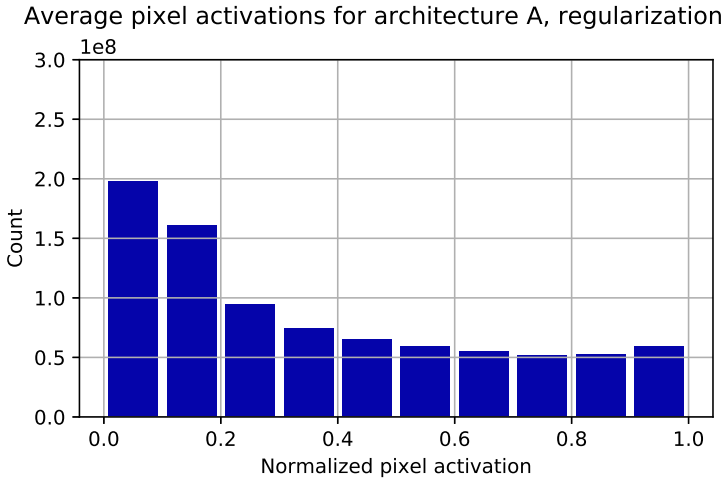


Figure 5.21: A histogram showing the activations of each pixel made by a model from the two-member ensemble trained using NCL and a diversity emphasis of 0.0. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset

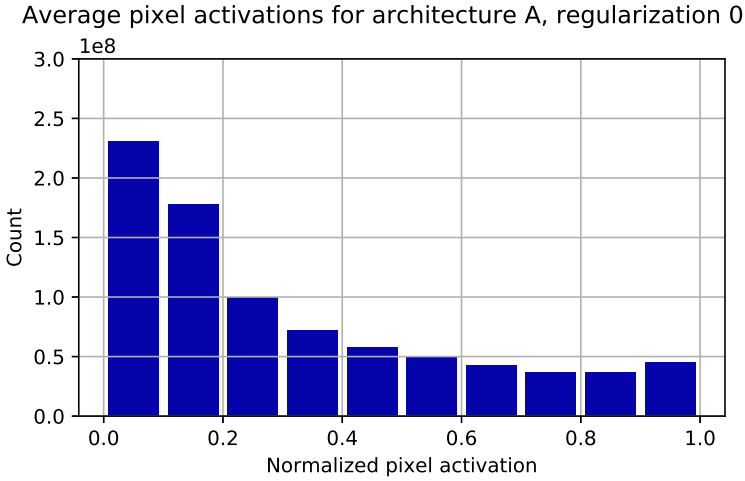


Figure 5.22: A histogram showing the activations of each pixel made by a model from the two-member ensemble trained using NCL and a diversity emphasis of 0.02. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset

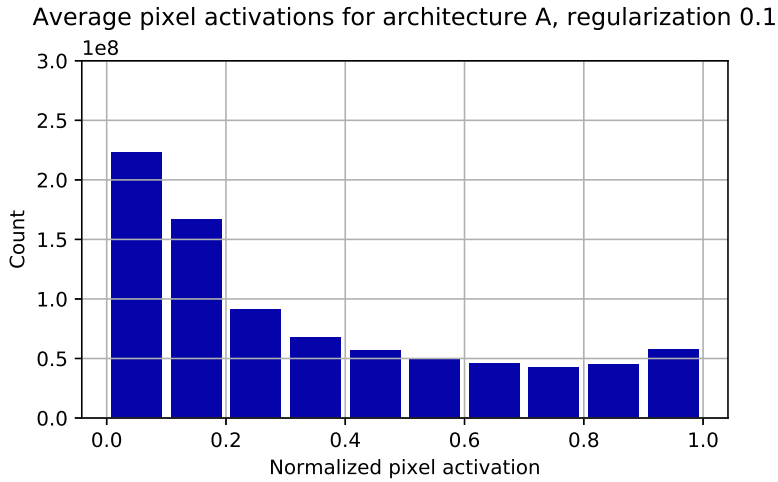


Figure 5.23: A histogram showing the activations of each pixel made by a model from the two-member ensemble trained using NCL and a diversity emphasis of 0.1. The model is referred to as architecture A, which means the model defined as convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. This is the sum of all pixel activations made by all folds and all runs on all images from the C-CORE/Equinor dataset

CHAPTER

SIX

DISCUSSION

This chapter will contain a discussion of the results presented in chapter 5. The main goal of this discussion is to comment on the findings, try to answer some of the questions posed in the chapter 4, and in the end give some answer to the research question posed in the introduction. To do this, this chapter will first contain comments and thoughts from each figure in chapter 5, before these comments will be summed up and used to answer the research questions. The first section, section 6.1 will contain comments regarding the results from negative correlation learning, and using this information to answer related research questions. The second section, section 6.2 will do the same, only focusing on the results from the GradCAM visualisations. Finally there will be done an effort to try to pull all the results together to answer the main research question of the thesis and draw some bigger lines in section 6.3.

6.1 NCL

First in this section, comments will be made on the results from the experiments conducted while using the NCL-algorithm. The first graph presented in figure 5.1, contains the average of three diversity measurements plotted against the diversity emphasis for the two-member ensemble. The point of this graph is to see that the implementation of NCL works, and increased diversity emphasis gives an increased amount of diversity among the classifiers. The measurements used here are the Q-statistic, ρ and double fault(DF) as presented in equations (3.18), (3.19) and (3.21), respectively. They have been scaled, so the lowest value for each measure is 0, and the highest is 1. Then all the measurements were averaged before this average value was presented in this graph. The reason this set of diversity measurements were chosen is that they are often used and they all work in the same way, where a lower value represents higher diversity. This makes a plot of them combined easily interpretable.

Generally, one can say that the diversity trends towards higher diversity(lower value) as the emphasis is increased. This trend is good news because this is what the algorithm was designed to do. One can also see that the diversity sometimes decreases when the diversity

emphasis is increased. This diversity decrease happens twice, but the diversity jumps quite quickly back up to even higher than before. A few theories are posed to explain why this happens. It could be an effect of randomness in the hypothesis space, where backpropagation overshoots the diverse space found by the lower emphasis solution, while not quite being able to reach the solution found when using higher diversity emphasis. This should not be a general problem, but might result in small spikes like the ones seen in figure 5.1. As long as the trend still is right, this will not be considered a problem. There is also the fact that there exists no definite measurement of diversity. There is no guarantee that the three measurements Q, ρ, and DF represents the diversity that is sought after. So, this plot is considered an indication that the algorithm works as intended, but it is no guarantee that good results will come.

The next graph is figure 5.2 containing a plot showing the ensemble loss for varying levels of diversity emphasis for the two-member ensemble. The base-case, with no emphasis on diversity, had a loss of 0.1824. The best model had an emphasis of 0.04 on diversity, which resulted in a loss of 0.1811. A slight improvement, but an improvement of 0.7% is not considered all that impressive. The general trend also seems to indicate that increasing diversity in an ensemble has no other effect than worsening the performance. This trend is in line with earlier work presented in the literature review, where the effect of enforcing diversity in small ensembles have been shown to have minimal effect on the classification loss. (Johansson and Löfström (2012), Zenobi and Cunningham (2001))

The third graph presented is the ensemble loss of the eight-member ensemble, plotted against the diversity emphasis. This can be seen in figure 5.3. Experimentation with varying diversity emphasis showed that using values in the range of 10^{-2}, as was done with the two-member ensemble, resulted in a cost function being completely dominated by the diversity term. In practical terms, this meant that the classifier jumped around in hypothesis space, never being able to converge towards any solution. This effect is a result of two factors. Since the penalty term, as defined in equation 4.1, is a simple summation of a pairwise diversity measure between one model and the others, this sum grows to roughly four times the size when using four times the amount of members. A way to remove this effect could have been to let the penalty term take into account the size of the ensemble, and divided by the number of members. The second factor, which, presumably, is the largest, is that the difficulty of finding eight spots in the hypothesis space that are both diverse from each other, while still being fairly accurate, is tremendous, compared to just finding two of those spots. These two factors make it necessary to decrease the diversity emphasis quite a lot.

Another thing worth commenting is the difference in baseline ensemble loss in the eight-member ensemble compared to the two-member ensemble. One would expect an ensemble consisting of eight members to be more accurate than an ensemble consisting of two members. This is not the case here but can be explained by the non-tuning of the learning rate. However, as explained in chapter 4, the concern is not raw performance, but the relative performance compared to the baseline.

A concern one might have with this is that what the experiments here are showing is that the worse an ensemble gets, the greater the performance gain of enforcing diversity. This point will be addressed in a set of experiments later in this section, where the dataset size will decrease, subsequently worsening the performance of the two-member ensemble.

Back to figure 5.3. The baseline loss with no diversity emphasis in this graph is 0.2595, while the best performing model with a diversity emphasis of $1 \cdot 10^{-7}$ had a loss of 0.2463. This difference is an improvement of 5%, which is 7 times higher than the performance increase of the two-member ensemble. This indicates that something happens when the amount of ensemble members increases. However, to say anything about how this effect manifests itself, it is needed to look at more results, for differing amounts of ensemble members and different datasets.

To continue, exactly that will be done. Figure 5.4 contains the ensemble loss of the four-member ensemble plotted against the diversity emphasis. First, it is important to note that this graph is the result of training using only 10% of the available data in the C-CORE/Equinor dataset. This explains the drastically lowered ensemble loss, as less training data means lower performance. It also explains why the four-member ensemble has optimum diversity emphasis around the same values as the ensemble with eight members trained on the full dataset. Changing the dataset size will make it harder to find good solutions that are both diverse and accurate. When inspecting the performance of the ensemble, something interesting can be noticed. It is quite clear that optimum diversity is somewhere in the region of $2 \cdot 10^{-7}$, where the ensemble loss is 0.3445. This is a performance increase of about 5.5%, which is even better than the eight-member ensemble. Does this mean the benefit of diversity does not necessarily grow as the ensemble adds more members? Alternatively, could it be that when training with a smaller dataset, the benefit of diversity is even larger? To investigate, one needs to inspect the performance of an eight-member ensemble trained with only 10% of the dataset.

The performance of the eight-member ensemble trained on only 10% of the C-CORE/ Equinor dataset can be seen in figure 5.7. The optimum diversity is quite a lot lower than the four-member ensemble, at $1 \cdot 10^{-10}$. This change is presumably due to the ensemble being twice as large, leading to higher difficulty in finding diverse and accurate hypotheses, and the diversity penalty in the loss function becoming twice the size. The best performance of the eight-member ensemble is 0.4037, while the baseline is 0.4309, an improvement of 7.5%. Better than the other eight-member ensemble which was trained on the full dataset, and better than the four-member ensemble. This is again a strong indication that larger ensembles benefit more from diversity in its members, but also that if the dataset one uses is smaller it is beneficial to enforce some diversity in the ensemble members at the expense of some raw, individual performance.

To continue to look at the scaling of the diversity effect, figure 5.5 will be commented next. It contains the ensemble loss of a 12-member ensemble plotted against the diversity emphasis. Again has the diversity emphasis been lowered quite a lot, due to the same reasons as before. Here an even more extreme increase in performance can be seen. The loss jumps from 0.5247 for 0.0 diversity emphasis, to 0.5207 for $1 \cdot 10^{-19}$ diversity emphasis, to 0.4972 for $1 \cdot 10^{-18}$ diversity emphasis, to 0.4704 for $1 \cdot 10^{-17}$ diversity emphasis. That's a performance increase of over 10% from the baseline to the best case. This is a significant indicator that the diversity effect scales with the size of the ensemble.

The same experiment, training using only 10% of the dataset has also been conducted using the two-member ensemble. This can be seen in figure 5.6. Again, there seems to be no discernible improvement for enforcing diversity. However, this plot can also be used to explore a concern raised earlier in the discussion. Since the ensembles gradually per-

form worse, it is hard to know if the relative improvement as the ensemble size becomes larger is because the ensemble can exploit the specialists to a larger degree or because the ensembles become worse performing. Maybe the effect comes from the fact that worse ensembles benefit more from diversity? In this experiment, the baseline with zero diversity emphasis had a loss of 0.3292, and it never improves for any diversity emphasis. The baseline of the eight-member ensemble trained on the full dataset had a baseline loss of 0.2595, and an improvement of 5% for some diversity emphasis. If the performance increase of the eight-member model were because it was worse performing, one would expect this two-member model to be able to benefit from some diversity emphasis. Its baseline is worse than the eight-member ensemble that showed clear improvement. These findings mean that one can put the concern to rest.

A related question, which was posed in section 4.4 is the effect of diversity when the dataset size is limited. When looking at figures 5.7 and 5.6 compared to figure 5.3 and 5.2, the results seems to indicate that there is two points one can make. The first is that there seems to be a larger improvement in performance when using a smaller dataset. The second is that if the ensemble is not able to reap the benefits of diversity when trained on a large dataset, it probably won't be able to if the dataset size is lowered either. This can be explained by the theory that enforcing diversity creates local specialists. You need a certain number of classifiers to take advantage of the specialists, no matter the training dataset size.

There was also conducted a quick experiment to look at the same effect of diversity on ensembles of varying sizes but using another dataset. With this goal in mind, the CIFAR10 dataset was adapted to a binary-classification problem, as described in chapter 4. Both ships and dogs were used as the class to classify. The results can be seen in figures 5.8 and 5.9 for the ship class. These graphs clearly show the same patterns that have been noticed earlier. While the small ensemble of two members seems to have no benefit from diversity, the eight-member ensemble benefits greatly from a diversity emphasis of $5 \cdot 10^{-7}$. The ensemble loss drops from 0.4331 to 0.4088, which is about a 5.5% performance increase. The dog-class results can be seen in figures 5.10 and 5.11. This again shows the same pattern, the two-member ensemble shows no improvement while the eight-member ensemble goes from 0.5090 to 0.4731, an improvement of 7%. Even better than the ship class, which might be attributed to that the dog class is a harder class to classify correctly.

The results will now be summed up to try to answer the research questions. In this section, the effect of enforcing diversity on an ensemble has been studied. A two-member ensemble showed no significant increase in performance by enforcing diversity while an eight-member ensemble showed some increase. This was also studied for a dataset of lower size, this time also with a four-member ensemble and a twelve-member ensemble. The effect seemed to scale, with a lower performance increase for the four-member ensemble, a medium increase for the eight-member ensemble and the highest for the twelve-member ensemble. It was also studied on another dataset, which also showed the same effect.

These experiments have been done in order to answer question 1, posed in chapter 4. It will be repeated here.

1. Is there a clear indication that ensembles with more members have a larger benefit from enforcing diversity?

It is the author's belief that this has been shown in the aforementioned figures and experiments. A very minimal effect on a two-member ensemble was shown for multiple datasets, and a large effect for an eight-member ensemble also shown on multiple datasets, and multiple dataset sizes. There was also shown an increase in the effect of diversity when the ensemble size was increased from four to eight to twelve. The next goal will be to try to explain this effect. To attempt this, the tool GradCAM will be used. The discussion about the GradCAM results will follow in the next section.

6.2 GradCAM

This section will contain a study into the illustrations and visualizations made by the GradCAM tool will be conducted. The first illustrations of the GradCAM activations can be found in figure 5.12. This figure consists of tree subfigures: an RGB interpretation of the SAR-image, the activations from a model trained with no emphasis on diversity printed on top of the RGB image, and the activations from a model from the eight-member ensemble trained with $1 \cdot 10^{-7}$ emphasis on diversity, which is also printed on top of the RGB interpretation. The first thing to notice is that both models seem to be reacting to the target in the middle of the image. This makes sense and gives an indication that both models are reacting to the right things in the image. One could become suspicious if there were many activations in the periphery of the image, as that is water. At the same time, one cannot say, as a human, that using the properties of the water to determine the image contents is the wrong approach. Maybe the water in some images consists of specific reflections that are only apparent when there is a boat in the image.

To continue, the activation differences between the models will be compared. These images are obliviously a cherry-picked example, to illustrate differences between a classifier that looks more like a generalist, and a classifier that, comparatively, looks more like a specialist. While it looks like the classifier trained with zero emphasis on diversity cares about the entire yellow “blob" in the RBG interpretation, the activation pattern of the classifier trained with some diversity emphasis is much more concentrated. This pattern is very much in line with the definition of local specialist provided in chapter 4.

Looking at the CIFAR10-image in figure 5.13, one can see the same general outline. The activation heatmap from the model trained with an emphasis on diversity has more concentrated activations, and only cares about the bow on the boat. The classifier trained with no diversity emphasis is more of a generalist. It has larger activations all over the bow on the boat, and it also cares somewhat about the sky to make its decision. This is also present in the dog activations in figure 5.14. The diversity enforced model only cares about the snout and the paw, while the model trained with no diversity emphasis has more medium-sized activations all over the dog face.

But, as said earlier, these are cherry-picked examples, mostly used to illustrate what to look for. To draw some conclusion about the activation patterns of each classifier, and the effect enforcing diversity has, the total average over all images need to be studied. Given these examples presented here, and the definition of a local specialist given in chapter 4, some expectations of the histograms can be formulated. One should expect the histogram of a specialist to have more pixels with low and high activations, since it cares greatly about some parts of the image, and not at all about others. The generalist should have more

medium sized activations, as it cares somewhat about broad parts of the image. It should, therefore, have less of the very high and very low activations compared to a specialist.

To study this, one can look at figures 5.15 and 5.16, which is the histogram of every single pixel activation for a classifier trained with no emphasis on diversity and $1 \cdot 10^{-7}$ emphasis on diversity. These are the average for all runs, all folds, and all images, on model architecture convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2. It is possible to see the patterns that were expected. The model with enforced diversity has more activations of size $0 - 0.1$ and $0.1 - 0.2$. All activation bins from 0.2 to 0.7 have more pixels on the model with no enforced diversity, and from activations of 0.7 to 1.0 there are more pixels on the model with enforced diversity. This is the exact pattern that was expected to be seen when comparing a generalist to a specialist. These histograms directly back up the theory that enforcing diversity when training models makes a set of local specialists.

To further build on this theory, the activations from other classifiers in the eight-member ensemble will be studied next. They are illustrated in figures 5.17 and 5.18 for 0.0 and $1 \cdot 10^{-7}$ diversity emphasis, respectively. These are, again, the average for all runs, all folds, and all images, on another model architecture, namely convolutional layers: [64, 128, 256], dense layers: [256, 128], dropout: 0.2. When the two patterns are compared, one can see the same pattern as was expected for a local specialist compared to a generalist. The generalist, with no emphasis on diversity, has a lot of medium sized activations. On the other hand, the specialist, with some diversity emphasis, has more very low activations and very high activations. This is, again, a strong indication that making diverse classifiers makes them local specialists, due to these patterns.

A third set of activations from a third model architecture were also plotted, as seen in figures 5.19 and 5.20 for 0.0 and $1 \cdot 10^{-7}$ diversity emphasis, respectively. This was quite a different architecture, convolutional layers: [32, 64, 128, 256], dense layers: [512, 256], dropout: 0.2. The reason this ends in such a different histogram of activations is that there is no padding in the convolutional layers in the models and there is max-pooling between each convolutional layer. Therefore, the 75×75 input shrinks to about 5×5, which when resized to 75×75 for the heatmap, results in a lot of medium sized activations compared to the two earlier models. Despite this large difference in general activation pattern, one can still clearly see the same differences between the activations of the model trained with 0.0 diversity emphasis and the model trained with $1 \cdot 10^{-7}$ diversity emphasis. $1 \cdot 10^{-7}$ diversity emphasis has a larger amount of low and high activations, while the 0.0 diversity emphasis has a lot more medium-sized activations.

To further back up this theory, the same histograms will be made using models from the two-member ensemble. They can be seen in figures 5.21 and 5.22. Here the pixel activations are from the model with architecture convolutional layers: [64, 128, 256], dense layers: [512, 256], dropout: 0.2, and a diversity emphasis of 0.0 and 0.02. A slightly different pattern can be seen here. They have very similar activations for medium values of activations, but at the extremes, there is a quite clear difference. The classifier with no emphasis on diversity has more high activations, while the classifier with some emphasis on diversity has more very low activations.

While this is a difference sort of in line with the theory of how the histograms should look, it is not everything that was expected. It is expected that the diverse classifier has

a larger amount of low activations, but it should also have more high activations. To understand if this can be explained by some factors, one needs to consider the differences between this experiment and the last. This classifier comes from an ensemble trained with only two members, and the last one came from an ensemble of eight members. It is oblivious that this affects the way the models are trained, but could it result in the effect that is observed here?

It is hypothesized that the reason for the two-member ensemble not benefiting from diversity, while the eight-member ensemble does, is that the diverse ensemble members created acts as local specialists. One is lead to believe that the two-member ensemble does not have enough specialists to benefit from them compared to the performance of a decent generalist. If that were the case, one would expect the histogram of activations from a classifier from the two-member ensemble to also exhibit specialist traits. The histogram presented in figure 5.22 does not fulfill all those expectations.

Why doesn't it fulfill the expectations set? It could be one of several reasons:

1. There is not enough diversity in the two-member ensemble

2. The local specialist definition given in this thesis is wrong

The first point will be addressed in this thesis, and the second will be saved for further work. A proposal for how to explore it, will be given in chapter 7, which is dedicated to further work.

To address point number one, it probably needs some explanation. Shouldn't there be enough diversity in the ensemble since the emphasis on diversity is so high compared to the low value of the eight-member ensemble? Not necessarily. When there are four times as many classifiers in the hypothesis space in the eight-member ensemble as the two-member ensemble, they could end up with pushing each other far more into the extremes than a two-member ensemble would, even though the emphasis is much lower. This is because it is more likely that there already is a classifier too close to the general solution normal backpropagation would find in an ensemble consisting of more members. The classifier will go searching into the more extremes to find something diverse from all the eight other solutions. When there is only one other classifier to care about, it is not necessary to go searching into such extremes.

Because of this, NCL with two members in the ensemble and even higher diversity emphasis will be studied next. It can be seen in figure 5.23. This is activations plotted for a diversity emphasis of 0.1. For a diversity emphasis of 0.1 the ensemble has very clearly started to become worse. In this histogram, one can now see all the expected traits of a local specialist — a larger amount of lower and high activations, and less of the medium activations.

This backs up the theory that when plotting the activations for a low diversity emphasis with approximately the same performance, the classifiers were not diverse enough yet to embody all the characteristics of a local specialist. When the emphasis was increased, the classifiers took on more of the characteristics of a local specialist. Unfortunately, at this point, the performance of the ensemble had become significantly worse. A possible explanation for this is that there were not enough members in the ensemble to properly exploit the properties of a specialist that makes it better than a generalist.

The GradCAM results will now be summarized, and used to answer the research questions. In this discussion a hypothesis was presented, which outlined how it was expected that the activations of a local specialist would look compared to a generalist. The activations of one of the classifiers from the eight-member ensemble were studied for zero and some diversity emphasis, and the patterns seemed to fall in line with the expected activations, suggesting that the classifier trained with an emphasis on diversity were a local specialist. This was also studied for two other architectures, which all showed the same pattern.

Next, it was wanted to back this up even further. The activations from the two-member ensemble were now studied, by making a histogram for 0.0 and 0.02 diversity emphasis. This deviated from some of the proposed expectations of a local specialist. It was suggested that it was because 0.02 was a too low emphasis on diversity, and a much higher emphasis of 0.1 was tested. At this point, the histogram showed a pattern more in line with what is expected of a local specialist.

Two questions were posed in section 4, when the problem was narrowed down. The one that relates to this section will be repeated here:

1. Can we pose some hypothesis as to why this is the case, and back it up?

This thesis based the hypothesis as to why this is the case, on a theory that had been presented in an earlier paper. Making very diverse classifiers will result in local specialists. One needs a certain amount of specialist before one can properly take advantage of them, compared to using some generalist. To back this up, the activation patterns of classifiers trained with differing emphasis on diversity where studied. The eight-member ensemble showed clear indications that enforcing diversity produced classifiers that exhibited more specialist-like traits compared to when no diversity was enforced. At the same time, the classifiers with an emphasis on diversity had a positive effect on the performance of the ensemble. The same local specialist traits were also shown in the two-member ensemble, but only for high diversity emphasis. It had to be tuned so high that the ensemble performance became significantly worse compared to the baseline. This is also in line with the original theory, which states that when the classifiers become local specialists, the two-member ensemble has too few specialists to get improved performance.

6.3 Answering the research question

The original, much more extensive and broad research question that was given in chapter 1, was how to make the best performing ensemble possible with a limited number of classifiers. The study that has been conducted in this thesis gives some answers to parts of this question. Pursuing diversity, to a certain extent, can have a good effect on an ensemble, given that the number of members so large that the ensemble can utilize it.

It is also very interesting to see that optimal ensembles are not necessarily made by making as well performing single models as possible and combining these. There are other considerations at play. This has implications in many fields. For example, regarding picking which models to use in an ensemble, given that the ensemble should be of a certain size. A typical way of doing this in, for example, a Kaggle competition is to evaluate every model using k-fold cross-validation, and picking the best performer. One might be able

to push the ensemble performance further by using a combination of diversity and cross-validation to evaluate each model.

This also applies to the world of Auto-ML. As talked about in the literature review, most of the research done in the field of Auto-ML has been related to designing single classifiers and making these single classifiers perform the best on their own. Autostacker was the first Auto-ML approach to try to make classifiers designed to perform their best in an ensemble along with other classifiers. The findings in this thesis do indicate that there is additional performance to be gained from making an Auto-ML algorithm which takes into account the fact that the classifiers should be made for an ensemble and not for best performance by itself.

CHAPTER

SEVEN

FURTHER WORK

This section will present more work to study the field of diversity in homogeneous ensembles further. This work will be other research directions that the author has thought about while making this thesis. These are research directions that could bring something new or further back up existing theories. Of course, with regards to the bigger question: how to make the best performing ensemble possible with a limited number of classifiers, there are too many ways to move forward to count. Therefore, this chapter will be restricted to further work that is directly related to the main problem tackled in this thesis, regarding diversity in ensembles.

First of all, for NCL to be useable for more than just to experiment with diversity, there should be made an effort into implementing NCL using some other framework than Keras. The goal should be a more effective implementation, which avoids the problem of freezing and un-freezing the models. A suggestion is to use Torch or Tensorflow, which grants more flexibility into defining a custom loss-functions.

A natural step to take this further is to experiment with much larger ensembles, for example, in the order of hundreds of members. This would require a more effective implementation of NCL as described above. The ensemble size would also be limited by the GPU memory of the computer training the models. Because of this, NCL will probably never be relevant for models consisting of thousands of models. At the same time, it is expected that at a certain point, the improvements gained by enforcing diversity might drop off. Given that the ensemble classifiers have slightly different architecture that makes them a little bit different, then a set of one thousand such classifiers will span a quite diverse hypothesis space naturally. Enforcing diversity in this set might not give any increased diversity due to the already high degree of diversity. So it would be interesting to try with larger ensembles, but it is expected that the positive effect of NCL will drop off at a certain point.

As talked about in the discussion in section 6.2, the definition of local specialist could use more research to be backed up. A straight forward way of doing this could be to apply the definition to other known classifier specialists and generalists, and see how well it holds up. The activation heatmaps could be compared, along with activation histograms and see if the same patterns are present. If that were the case, it would be an interesting

finding by itself, but it would also strengthen the work done in this thesis.

Another exciting route to research would be to train an ensemble of many members, for 0 diversity emphasis and some diversity emphasis that gives an improvement in ensemble performance. Then, one could extract, say, two of those members, trained both for 0 diversity emphasis and some diversity emphasis that gave a performance increase. The two extracted members with 0 diversity emphasis could be ensembled, and the two extracted members with some diversity emphasis ensembled. Then one could study the performance again. If two members in the ensemble are too few to take advantage of the specialists made by enforcing diversity, one would expect the two-member ensemble trained with 0 diversity emphasis to perform better than the other. Is that the case? This could further solidify the theory that the smaller ensembles do not gain any performance because the classifiers become local specialists.

In general, one could say that it would be beneficial to study further the properties of the members trained to have more diversity compared to no diversity. Other methods of making diversity could be used, and other ways of studying the classifiers.

One could also study the effect diversity has on other performance measures than ensemble loss. The literature review showed that diversity had a positive effect on the loss due to its effect on the variance and the covariance. Showing this effect on real classifiers would be very interesting. This would make enforcing diversity in the classifiers very relevant for use-cases where a low variance would be beneficial.

CHAPTER
EIGHT

CONCLUSION

This thesis has been a study into making and designing optimal ensembles, meaning as high performing as possible, of a specific size. A literature review was conducted to show ways that have been used to to make these optimal ensembles. This review mainly consisted of ways of exploiting diversity and an Auto-ML approach. Based on this review, it was decided to pursue the world of diversity, and looking into some earlier findings regarding diversity and ensemble size.

These earlier findings showed that smaller ensembles have minimal (if any at all) performance gain of diversity. Larger ensembles had a positive effect. This was recreated in this thesis, using an algorithm called negative correlation learning. An ensemble of two members showed no effect of enforcing diversity, while an eight-member ensemble had a significant performance increase. This effect was shown on multiple datasets for multiple network architectures.

This thesis wanted to contribute with something new. To do that, it was wanted to try to explain why this effect of diversity on ensembles was as shown. A hypothesis for this was proposed, which also had been proposed in earlier literature. This definition said that the ensemble members became local specialists when enforcing diversity when trained. A definition of a local specialist was given, which were to be used to study the classifiers.

The tool GradCAM was used to study the activations of each classifier on each image. The individual pixel activations were tallied up for every image and placed into a histogram to study if there were some discernible differences. It was proposed that a generalist classifier would have more medium sized activations, while a specialist classifier would have, comparatively, more very low and very high activations. This was shown for multiple classifiers with different architectures, in both a two-member ensemble and an eight-member ensemble. The classifier trained with an emphasis on diversity showed clear specialist traits, and also gave a clear performance increase in the eight-member ensemble compared to the baseline. At the same time, one had to increase the diversity emphasis to the point where the performance had been lowered significantly to see specialists-traits in the two-member ensemble.

These studies and experiments help answer a small part of the question: how does one make an optimal ensemble given a specific size? Enforcing diversity is beneficial, given

that the ensemble is of a large enough size. This research, along with other research into making optimal ensembles, could give the winning edge if the ensemble size is restricted, which it sometimes is due to hardware limitations.

BIBLIOGRAPHY

Alpaydin, E., 2010. Introduction to Machine Learning, 2nd Edition. The MIT Press.

Antonio, B., 2018. Image classification: Cifar-10.
URL https://medium.com/@bian0628/image-classification-cifar-10-dc1c23db46d5

Breiman, L., Aug. 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.
URL http://dx.doi.org/10.1023/A:1018054314350

Breiman, L., 2001. Random forests. Machine learning 45 (1), 5–32.

Brown, G., 2004. Diversity in neural network ensembles. Tech. rep., The University of Birmingham.

Brown, G., Wyatt, J. L., Harris, R., Yao, X., 2004. Diversity creation methods: a survey and categorisation. Information Fusion 6, 5–20.

C-CORE, 2018. Statoil kaggle competition services, report R-17-077-1386, Revision 2.0.

Chan, Z. S., Kasabov, N., 2005. A preliminary study on negative correlation learning via correlation-corrected data (nccd). Neural Processing Letters 21 (3), 207–214.

Chen, B., Wu, H., Mo, W., Chattopadhyay, I., Lipson, H., 2018. Autostacker: A compositional evolutionary learning system. In: Proceedings of the Genetic and Evolutionary Computation Conference. ACM, pp. 402–409.

Chen, H., 2008. Diversity and regularization in neural network ensembles. Ph.D. thesis, University of Birmingham.

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16. ACM, New York, NY, USA, pp. 785–794.
URL http://doi.acm.org/10.1145/2939672.2939785

Chollet, F., et al., 2015. Keras. https://keras.io.

Cunningham, P., Carney, J., 05 2000. Diversity versus quality in classification ensembles based on feature selection. In: Machine Learning: ECML 2000: 11th European Conference on Machine Learning. pp. 109–116.

Demiriz, A., Bennett, K. P., Shawe-Taylor, J., Mar. 2002. Linear programming boosting via column generation. Mach. Learn. 46 (1-3), 225–254.
URL http://dx.doi.org/10.1023/A:1012470815092

Dua, D., Graff, C., 2017. UCI machine learning repository.
URL http://archive.ics.uci.edu/ml

Džeroski, S., Ženko, B., Mar 2004. Is combining classifiers with stacking better than selecting the best one? Machine Learning 54 (3), 255–273.
URL https://doi.org/10.1023/B:MACH.0000015881.36452.6e

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., Hutter, F., 2015. Efficient and robust automated machine learning. In: Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., Garnett, R. (Eds.), Advances in Neural Information Processing Systems 28. Curran Associates, Inc., pp. 2962–2970.
URL http://papers.nips.cc/paper/5872-efficient-and-robust-automated-machine-learning.pdf

Fleiss, J., 1981. Statistical Methods for Rates and Proportions. Second Edition. Wiley, John and Sons, Incorporated, New York, N.Y.
URL https://books.google.no/books?id=I79_MgAACAAJ

Foslien, S., 2018. Machine learning for classifying iceberg/ship in satellite images. Tech. rep., Norwegian University of Science and Technology.

Freund, Y., Schapire, R. E., Aug. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. 55 (1), 119–139.
URL http://dx.doi.org/10.1006/jcss.1997.1504

Friedman, J., Hastie, T., Tibshirani, R., 04 2000. Additive logistic regression: a statistical view of boosting (with discussion and a rejoinder by the authors). Ann. Statist. 28 (2), 337–407.
URL https://doi.org/10.1214/aos/1016218223

Geman, S., Bienenstock, E., Doursat, R., Jan 1992. Neural networks and the bias/variance dilemma. Neural Computation 4 (1), 1–58.

Giacinto, G., Roli, F., 08 2001. Design of effective neural network ensembles for image classification purposes. Image and Vision Computing 19, 699–707.

Glass, L., Mackey, M., 2010. Mackey-Glass equation. Scholarpedia 5 (3), 6908, revision #186443.

Glosser.ca, 2013. Artificial neural network with layer coloring.
URL https://commons.wikimedia.org/wiki/File:Colored_neural_network.svg

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. The MIT Press.

Gorman, B., 2016. A kaggler's guide to model stacking in practice.
URL http://blog.kaggle.com/2016/12/27/a-kagglers-guide-to-model-stacking-in-practice/

Ho, T. K., Aug. 1998. The random subspace method for constructing decision forests. IEEE Trans. Pattern Anal. Mach. Intell. 20 (8), 832–844.
URL http://dx.doi.org/10.1109/34.709601

Hutter, F., Kotthoff, L., Vanschoren, J. (Eds.), 2018. Automated Machine Learning: Methods, Systems, Challenges. Springer, in press, available at http://automl.org/book.

Johansson, U., Löfström, T., 2012. Producing implicit diversity in ann ensembles. In: The 2012 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Kearns, M., Valiant, L., 1988. Learning Boolean Formulae Or Finite Automata is as Hard as Factoring. Technical report (Harvard University. Aiken Computation Laboratory). Harvard University, Center for Research in Computing Technology, Aiken Computation Laboratory.
URL https://books.google.no/books?id=r1VpPgAACAAJ

Kearns, M., Valiant, L. G., 1989. Crytographic limitations on learning boolean formulae and finite automata. In: Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing. STOC '89. ACM, New York, NY, USA, pp. 433–444.
URL http://doi.acm.org/10.1145/73007.73049

Kohavi, R., Wolpert, D., 1996. Bias plus variance decomposition for zero-one loss functions. In: Proceedings of the Thirteenth International Conference on International Conference on Machine Learning. ICML'96. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 275–283.
URL http://dl.acm.org/citation.cfm?id=3091696.3091730

Krizhevsky, A., Hinton, G., 2009. Learning multiple layers of features from tiny images. Tech. rep., Citeseer.

Krizhevsky, A., Sutskever, I., Hinton, G. E., 2012. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12. Curran Associates Inc., USA, pp. 1097–1105.
URL http://dl.acm.org/citation.cfm?id=2999134.2999257

Krogh, A., Vedelsby, J., 1994. Neural network ensembles, cross validation and active learning. In: Proceedings of the 7th International Conference on Neural Information Processing Systems. NIPS'94. MIT Press, Cambridge, MA, USA, pp. 231–238.
URL http://dl.acm.org/citation.cfm?id=2998687.2998716

Krzanowski, W., Partridge, D., 01 1997. Software diversity: Practical statistics for its measurement and exploitation. Information and Software Technology 39.

Kuncheva, L. I., Whitaker, C. J., May 2003. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. Mach. Learn. 51 (2), 181–207. URL https://doi.org/10.1023/A:1022859003006

LeCun, Y., Bengio, Y., Hinton, G., 05 2015. Deep learning. Nature 521, 436–44.

Liu, Y., Yao, X., 1999a. Ensemble learning via negative correlation. Neural networks 12 (10), 1399–1404.

Liu, Y., Yao, X., 1999b. Simultaneous training of negatively correlated neural networks in an ensemble. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 29 (6), 716–725.

Lofstrom, T., Johansson, U., Niklasson, L., July 2007. Empirically investigating the importance of diversity. In: 2007 10th International Conference on Information Fusion. pp. 1–8.

Maclin, R., Shavlik, J. W., et al., 1995. Combining the predictions of multiple classifiers: Using competitive learning to initialize neural networks. In: IJCAI. Citeseer, pp. 524–531.

Melville, P., 2003. Creating diverse ensemble classifiers. Tech. rep., THE UNIVERSITY OF TEXAS AT AUSTIN.

Nielsen, M. A., 2015. Neural Networks and Deep Learning. Determination Press.

Noel Sharkey, Amanda Sharkey, J. N., 1995. Searching weight space for backpropagaion solution types. In: Current Trends in Connectionism: Proceedings of the 1995 Swedish Conference on Connectionism. pp. 103–119.

Olson, R. S., Urbanowicz, R. J., Andrews, P. C., Lavender, N. A., Moore, J. H., et al., 2016. Automating biomedical data science through tree-based pipeline optimization. In: European Conference on the Applications of Evolutionary Computation. Springer, pp. 123–137.

Olvi L. Mangasarian, W. Nick Street, W. H. W., Aug. 1995. Breast cancer diagnosis and prognosis via linear programming. Oper. Res. 43 (4), 570–577. URL http://dx.doi.org/10.1287/opre.43.4.570

Opitz, D. W., Shavlik, J. W., 1996. Actively searching for an effective neural network ensemble. Connection Science 8 (3-4), 337–354.

Oza, N. C., Tumer, K., 2001. Input decimation ensembles: Decorrelation through dimensionality reduction. In: International Workshop on Multiple Classifier Systems. Springer, pp. 238–247.

Parmanto, B., Munro, P. W., Doyle, H. R., 1996. Improving committee diagnosis with resampling techniques. In: Advances in neural information processing systems. pp. 882–888.

Partridge, D., 1996. Network generalization differences quantified. Neural Networks 9 (2), 263–271.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic differentiation in pytorch. NIPS 2017.

Raschka, S., 2015. Python Machine Learning. Packt Publishing.

Raviv, Y., Intrator, N., 1996. Bootstrapping with noise: An effective regularization technique. Connection Science 8 (3-4), 355–372.

Refaeilzadeh, P., Tang, L., Liu, H., 01 2009. Cross-validation. Encyclopedia of Database Systems 532–538, 532–538.

Rosen, B. E., 1996. Ensemble learning using decorrelated neural networks. Connection science 8 (3-4), 373–384.

Rumelhart, D. E., Zipser, D., 1985. Feature discovery by competitive learning. Cognitive science 9 (1), 75–112.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L., 2015. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) 115 (3), 211–252.

Schapire, R. E., Jul. 1990. The strength of weak learnability. Mach. Learn. 5 (2), 197–227.
URL https://doi.org/10.1023/A:1022648800760

Schapire, R. E., Freund, Y., Bartlett, P., Lee, W. S., 1998. Boosting the margin: A new explanation for the effectiveness of voting methods. The Annals of Statistics 26 (5), 1651–1686.
URL https://eprints.qut.edu.au/43935/

Selvaraju, R. R., Das, A., Vedantam, R., Cogswell, M., Parikh, D., Batra, D., 2016. Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. CoRR abs/1610.02391.
URL http://arxiv.org/abs/1610.02391

Sharkey, A., Sharkey, N., 1997a. Diversity, selection, and ensembles of artificial neural nets. Neural Networks and their Applications (NEURAP'97), 205–212.

Sharkey, A. J., 1999. Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems. Springer-Verlag.

Sharkey, A. J. C., Sharkey, N. E., Sep. 1997b. Combining diverse neural nets. Knowl. Eng. Rev. 12 (3), 231–247.
URL http://dx.doi.org/10.1017/S0269888997003123

Skalak, D. B., 1996. The sources of increased accuracy for two proposed boosting algorithms. In: In Proc. American Association for Arti Intelligence, AAAI-96, Integrating Multiple Learned Models Workshop. pp. 120–125.

Sneath, P. H. A., Sokal, R. R., 1973. Numerical Taxonomy: The Principles and Practice of Numerical Classification. W. H. Freeman and Co.

Tang, E. K., Suganthan, P. N., Yao, X., Oct. 2006. An analysis of diversity measures. Mach. Learn. 65 (1), 247–271.
URL http://dx.doi.org/10.1007/s10994-006-9449-2

Tieleman, T., Hinton, G., 2012. Lecture 6a overview of mini-batch gradient descent.
URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf

Ting, K. M., Witten, I. H., May 1999. Issues in stacked generalization. J. Artif. Int. Res. 10 (1), 271–289.
URL http://dl.acm.org/citation.cfm?id=1622859.1622868

Ueda, N., Nakano, R., June 1996. Generalization error of ensemble estimators. In: Proceedings of International Conference on Neural Networks (ICNN'96). Vol. 1. pp. 90–95 vol.1.

Usmani, Z.-u.-h., 2017. Kaggle for Beginners: with Kernel Code. Gufhtugu Publications.

Wang, S., Chen, H., Yao, X., 2010. Negative correlation learning for classification ensembles. In: The 2010 International Joint Conference on Neural Networks (IJCNN). IEEE, pp. 1–8.

Wolpert, D. H., 1992. Stacked generalization. Neural Networks 5 (2), 241 – 259.
URL http://www.sciencedirect.com/science/article/pii/S0893608005800231

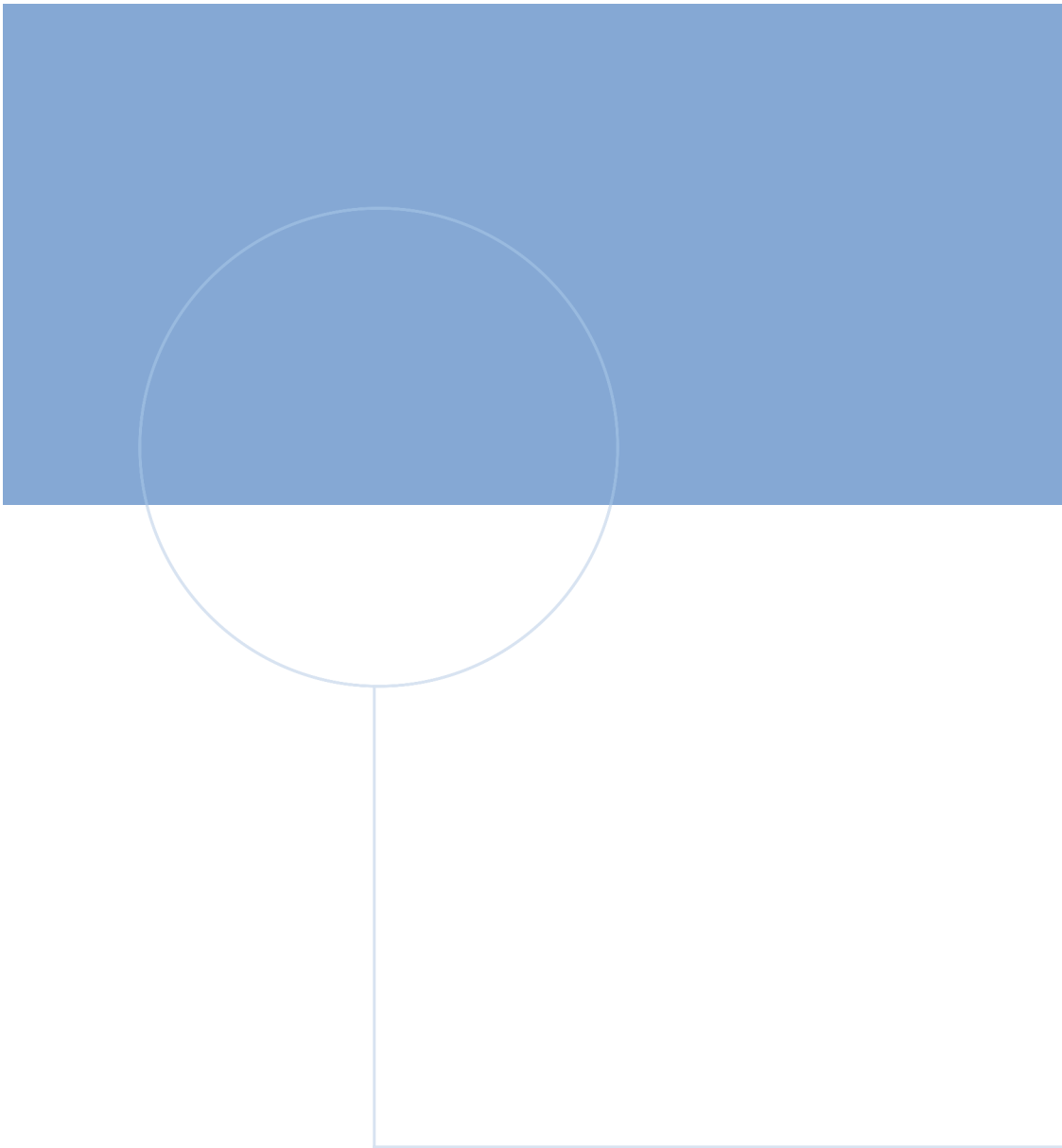
Wu, B., Dec 1992. An introduction to neural networks and their applications in manufacturing. Journal of Intelligent Manufacturing 3 (6), 391–403.
URL https://doi.org/10.1007/BF01473534

Yates, W. B., Partridge, D., 1996. Use of methodological diversity to improve neural network generalisation. Neural Computing & Applications 4 (2), 114–128.

Yule, G. U., 1900. 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 194, 257–319.
URL http://www.jstor.org/stable/90759

Zenobi, G., Cunningham, P., 2001. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In: European Conference on Machine Learning. Springer, pp. 576–587.

Zhou, Z.-H., 2012. Ensemble Methods: Foundations and Algorithms, 1st Edition. Chapman & Hall/CRC.



NTNU
Norwegian University of
Science and Technology

equinor