Investigating the NIN Structure

Undersökning av NIN-strukturen

LUKAS LUNDMARK AND VIKTOR HOLMÉR
Investigating the Convolutional NIN Structure
Varying number of Stages & Layer Depth

Lukas Lundmark
Viktor Holmér

Spring Semester, 2016

DD143X, Degree project in Computer Science
Supervisor: Mårten Björkman
Examinator: Örjan Ekeberg
Abstract

In this thesis the NIN artificial neural network structure created by Min Lin et al in 2014 is investigated. This is done by varying stage numbers and layer depth. By doing this ten different networks including the original NIN were created. Testing is carried out on a preprocessed version of the CIFAR10 dataset for these ten networks for a maximum of 150'000 iterations. The results show that the number of stages generally affect NIN performance more than layer depth does.

The network with three stages and a layer depth of two performs best at a top accuracy of 87.44%. This is below Min Lin et al’s results. However, this is likely due to overfitting and lack of specifics on their training methods. The thesis concludes that studies of different types of micro-networks in the NIN are required. Studies are also required on deeper NINs with larger datasets to prevent the overfitting observed in the results. Larger datasets could be obtained by data augmentation. Furthermore, the results suggests that less complicated (by less complicated it is meant that the network stages have less depth) NIN implementations are more accurate than deeper ones.

Abstract in Swedish


Nätverket med tre moduler och ett lagerdjup på två har högst prestanda med 87.44% rätt. Detta är under Min Lin et als resultat. Detta beror dock troligen på overfitting och en brist på närmare detaljer rörande träningsmetoderna de använt. Slutsatsen av arbetet är att studier på olika typer av mikronätverk i NIN behövs. Studier på djupare NIN-nätverk med större dataset för att förhindre den overfitting som syns i resultaten behövs även. Större dataset skulle kunna erhållas genom data-augmentering. Resultaten verkar även antyda att mindre komplicerade (med mindre komplicerade menas att modulerna har mindre djup) NIN-implementationer har högre prestanda än djupa sådana.
**Terminology**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Network</td>
</tr>
<tr>
<td>CIFAR</td>
<td>Canadian Institute For Advanced Research</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional Neural Network</td>
</tr>
<tr>
<td>cuDNN</td>
<td>CUDA Deep Neural Network library</td>
</tr>
<tr>
<td>DBM</td>
<td>Deep Boltzmann Machine</td>
</tr>
<tr>
<td>DCNN</td>
<td>Deep Convolutional Neural Network</td>
</tr>
<tr>
<td>DNN</td>
<td>Deep Neural Network</td>
</tr>
<tr>
<td>FNN</td>
<td>Feedforward Neural Network</td>
</tr>
<tr>
<td>GBM</td>
<td>General Boltzmann Machine</td>
</tr>
<tr>
<td>ILSVRC</td>
<td>ImageNet Large Scale Visual Recognition Challenge</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron</td>
</tr>
<tr>
<td>MNIST</td>
<td>Mixed National Institute of Standards and Technology</td>
</tr>
<tr>
<td>NIN</td>
<td>Network In Network</td>
</tr>
<tr>
<td>RBM</td>
<td>Restricted Boltzmann Machine</td>
</tr>
<tr>
<td>ReLU</td>
<td>Rectified Linear Unit</td>
</tr>
<tr>
<td>RNN</td>
<td>Recurrent Neural Network</td>
</tr>
<tr>
<td>SVHN</td>
<td>Street View House Number</td>
</tr>
</tbody>
</table>
Table of Contents

1 Introduction ................................................................. 1
  1.1 Problem Statement ..................................................... 2
  1.2 Scope ........................................................................ 2
  1.3 Overview .................................................................... 2

2 Background ................................................................. 2
  2.1 Artificial Neural Networks .............................................. 2
    2.1.1 Fundamental Principles .......................................... 3
    2.1.2 Learning .............................................................. 4
    2.1.3 Overfitting ......................................................... 4
  2.2 Convolutional Networks ............................................... 5
    2.2.1 Related Work ..................................................... 6
  2.3 Datasets .................................................................... 7

3 Methods ................................................................. 8
  3.1 Dataset Utilization .................................................... 8
  3.2 Programming Libraries .............................................. 9
  3.3 Parameters ............................................................... 9
  3.4 Training .................................................................. 9
  3.5 Measurements .......................................................... 9

4 Results ................................................................. 10

5 Discussion ............................................................... 10
  5.1 Constraints .............................................................. 11
  5.2 Further Research ..................................................... 12
  5.3 Conclusion .............................................................. 12

6 References ............................................................. 13
1 Introduction

Object recognition and image classification is the problem of detecting and classifying images based on objects in the pictures. The problem of accurately categorizing images based on the objects within them has been studied since the sixties. More accurate object recognition systems have been developed since the advent of efficient Artificial Neural Networks (ANNs).

ANNs are computational models based on biological nerve cells (neurons) found in the brains of different kinds of animals[1][2]. They are comprised of layers of artificial neurons that are each computationally simple in design and function. ANNs are today commonplace in the field of medicine and economics and their use in other fields is growing[3][4].

Convolutional Neural Networks (CNNs) are types of ANNs based on the human eye. CNNs are used for image, handwriting and and object recognition[1]. Deep Neural Networks (DNNs) and Deep CNNs (DCNNs) are ANNs constructed with many more layers than what was historically possible. DNNs and DCNNs have only recently started being studied due to earlier constraints in hardware performance.

Recognizing and categorizing objects in images is an example of a classification problem where the goal is to label input data thus classifying the data. The CIFAR10 and CIFAR100 datasets presents the problem of classifying images to 10 or 100 different classes. These include the airplane, automobile, bird and cat classes[5]. Today it is possible to detect and classify objects with an error rate of 15.3% by using DCNNs[6]. This is however with the assumption that the images are of the ImageNet dataset which is regarded as one of the more difficult datasets due to its size of more than 14 million images[7][6]. The CIFAR datasets are some orders of magnitude smaller and classification accuracies in these have reached above 90% in recent years[8].

CIFAR10 is a prime example of an object recognition problem where the goal is to identify one of ten types of objects that is in an image. CIFAR10 is an extension of its counterpart. It has ten different coarse classes that in turn has ten fine classes each. This makes for 100 classes in total increasing the challenge of designing a network for this problem[5]. Businesses are increasingly using DCNNs to tackle such problems. For example there are industrial object recognition networks in supermarkets for detecting shoppers, in facial recognition software and for scanning house numbers and license plates when extending Google Streetview[9].

Constructing a DCNN for accurate object recognition in real world photographs is no trivial task. There are no easily obtainable statistics on accuracies and implementations are thus left to a sort of trial and error. The trial and error can however be mitigated by empirically studying variations of already developed network structures[10]. It follows that it can be a difficult task to create a CNN that performs within a certain error margin. This is of importance to the previously mentioned industrial applications and access to such statistics are already helping to spur further successful implementations of DCNNs.

Investigating newly developed neural network structures in order to see if their performances can be increased has previously been done many times[10][11]. By doing this one can optimize such structures and aid in finding new and more efficient models for problems such as object recognition. Varying certain parameters in networks that perform well could also help further the understanding of how DCNNs are affected by said parameters[10].

Network in Network (NIN) is a DNN structure created by Min Lin et al in 2014 at the
National University of Singapore specifically for the CIFAR datasets[12]. With NIN Min Lin et al were able to achieve state-of-the-art performance comparable to the most accurate networks applied to CIFAR10[5][12]. This thesis studies NIN’s performance when varying its depth in accordance with Min Lin et al’s recommendation of stacking the major structure[12]. A different number of stages which are defined in the method section are also investigated.

1.1 Problem Statement

The impact of varying depths and number of stages (defined in the method section) on DCNN accuracies is not well documented. In recently developed structures like NIN even less so. Thus this thesis will investigate the impact of varying the depth and stage numbers on the accuracy of the NIN structure. The CIFAR10 (Canadian Institute For Advanced Research 10) dataset will be used for testing.

1.2 Scope

The goal of this project is to empirically study the performance of the NIN structure when varying its depth and changing its number of stages. Since NIN was tailor made for CIFAR it was decided to test variations of it on CIFAR10. The intention is to show some interesting correlations between depth, stage number and accuracies using statistics gathered from the experiments.

Note that there is no intention to achieve state-of-the-art performance nor even to improve on the NIN network. The aim of the thesis is to study to what extent depth and stage number affect NIN performance on CIFAR10.

1.3 Overview

Section 2 explains the fundamental principles of ANNs, CNNs and briefly introduces the datasets considered for this thesis. Section 3 firstly explains how the datasets were utilized and augmented. Furthermore, it discusses the programming libraries that were used in the thesis’ experiments. Lastly it details how the constructed networks were trained and how the thesis’ data were measured and collected.

Section 4 lists the results of the thesis’ experiments in tables. Finally, section 5 analyzes the results found in section 4 as well as the constraints on the project and propositions for further research. It concludes with the thesis’ conclusions drawn from the previously mentioned analysis.

2 Background

This section will introduce concepts and earlier research discussed later on throughout the entire thesis. It firstly discusses the principles of Artificial Neural Networks, the workings of such networks and moves on to discuss Convolutional Neural Networks which are the main focus of this thesis. Lastly the datasets considered in this study are briefly discussed.

2.1 Artificial Neural Networks

Artificial Neural Networks (ANNs) are as their name implies simulated structures designed to loosely model the functions of neurons (nerve cells) that exists in the brains of different animals[1][2]. The reasons for recent rising interests in the field of ANNs are many. Neural networks can for example solve traditionally difficult problems. Using conventional techniques to solve problems such as facial recognition, speech recognition or image classification can be complicated and time consuming. Furthermore, ANNs have been shown to perform better at solving such ill-defined, stochastic and non-linear problems[1]. It was for example proven in 1993 that ANNs can approximate any function, regardless of its dimensionality, under certain conditions[13].

An ANN is comprised of several layers of
2.1 Artificial Neural Networks

**BACKGROUND**

Artificial neural networks (ANNs) consist of artificial neurons; an input layer, one or more hidden layers and an output layer. By performing computationally trivial calculations combined with self-adjusting biases and weights (explained in section 2.1.2) an ANN is capable of learning[1][2]. Learning could for example be done by feeding the network with pictures of cats and pictures without cats. If a sufficient amount of balanced training data is supplied the network should then, to some extent, be able to classify any image as “contains a cat” or “does not contain a cat”.

Today the applications of ANNs are becoming increasingly apparent. In 2001 their usefulness in the field of medicine were an obvious application[2]. In 2007 ANNs were already becoming commonplace in medicine and especially useful in cardiology[3]. However, currently their applications are legion. They are for example used in: speech recognition, facial recognition, handwriting recognition, medical analysis and economics[3][14][4][15]. The reader probably recognizes many of these features from his/her own smartphone.

As has already been discussed there are several advantages to ANNs compared to conventional approaches to solving non-linear, stochastic and ill-defined problems. In theory ANNs can be more robust than ordinary sequential circuitry due to the paralell and independent nature of neurons[1]. However, custom engineered hardware built for such independent and paralell execution of neuronal computations is as of yet in its youth. For example in 2006 an Australian team created a neural network engine specifically for a certain CPU to yield a high speed chip for connectionist (ANN) simulations[16].

### 2.1.1 Fundamental Principles

ANNs are applicable to ill-defined, stochastic and non-linear problems due to their ability to learn from processed data. More specifically, they are capable of self-correcting and adapting to input data during what is called training. There is no need for reprogramming of the ANN neurons, since the network can by itself control the threshold of neurons firing[1]. Threshold refers to the weights and biases controlling the strength of the activation function of a neuron e.g. the value of its output data.

The artificial neurons of an ANN are what makes up its layers. They are numerous in function, the most common will be mentioned here. The Input Layer is comprised of input neurons that receive input data and pass it to the following layers. The Hidden Layer is simply called hidden due to its invisibility to users and programmers; what occurs in this layer is defined simply from the activation functions of individual neurons. The final layer is the Output Layer which simply outputs the results from the Hidden Layer. Together these layers of neurons form an ANN[1][17]. In figure 1 a simple ANN with one hidden layer can be seen. It is an FNN, a term that will be explained later in this section.

![Figure 1: A very simple ANN.](image)

Deep Neural Networks (DNNs) are ANNs with a multitude of hidden layers; deep hidden layers. The more ordinary counterpart is, unsurprisingly, referred to as shallow networks[17].

A Restricted Boltzmann Machine (RBM) is a common type of network and has applications in DNNs, in which they are called Deep Boltzmann Machines (DBMs). An ordinary General Boltzmann Machine (GBM)
is a network were the neurons of the hidden and visible layers form a complete graph. An RBM is thus a GBM were the hidden and visible layers form a bipartite graph. A DBM has the same requirement, however the hidden layers can be manifold and must form bipartite graphs between neighboring layers[18].

There are many types of neurons that ANNs can be comprised of. The perceptron is one of the most common and oldest artificial neurons that commonly uses a step-function as its so called activation function. Perceptrons can however use other activation functions and is sometimes referred to based on the function’s name. For example a sigmoid’s activation function is not a step-function, but is still a perceptron neuron even though it is usually referred to as a sigmoid[1]. Sigmoid were shown to outperform the heaviside step function previously used in perceptrons[19]. A rectified linear unit (ReLU) is an artificial neuron that has the activation function \( f(x) = \max(0,x) \). They were shown to outperform sigmoids in some DNN applications due to its closer similarity to biological neurons[17][20]. ReLUs can reduce training time for DCNNs by a factor of six and thus allows for experimentation with much deeper networks than would otherwise be possible[6].

Feedforward Neural Networks (FNNs) are ANNs where connections between the network’s artificial neurons do not form a cycle. A frequently presented example is the multilayer perceptron (MLP), one of the first FNNs developed[1][2]. The MLP is still frequently in use to this day, for example in the Network in Network structure[12].

Recurrent Neural Networks (RNNs) are ANNs where connections between the network’s artificial neurons form a directed cycle. This allows neurons to interact with each other creating a sort of memory. An RNNs internal state thus depends on all of its previous internal states. It is this internal memory that has made RNNs successful within speech recognition and handwriting recognition[15].

### 2.1.2 Learning

Weights and biases are values that control when neuron activation functions are triggered and is the reason ANNs can learn and self-configure[1]. It is by this process that ANNs are able to extrapolate on input data and solve the previously mentioned ill-defined problems of for example object recognition, handwriting recognition and facial detection.

Backpropagation was invented along with multilayered perceptrons (detailed in section 2.2.1) and made DNNs practical. It is thus primarily used by DNNs to learn by passing errors seen in output data backwards through the network and adjusting weights accordingly. This is often done by utilizing a mathematical optimization approach called gradient descent[1][2].

### 2.1.3 Overfitting

Overfitting occurs when a network is unnecessarily complex and thus learns to recognize random noise instead of desired patterns. This can for example occur to a network that has been trained on an insufficient amount of data[21]. Large, deeper networks are even more prone to overfitting than shallow ones[22].

One method often used for the prevention of overfitting is the stochastic shutdown of neurons in training iterations. This is called dropout[22]. A fully connected layer, a layer whose every node is connected to the succeeding layer’s nodes, is for instance very prone to overfitting. Dropout was shown by Min Lin et al to greatly reduce accuracy loss due to overfitting[12]. Srivastava et al invented dropout and simultaneously showed that it can reduce overfitting in DNNs rendering them more useful than previously[22].
Data augmentation is a common approach to easily increase the amount of available training data in datasets by applying subtle changes to the original input data. These changes can include cropping, rotating and applying colour-filters to the input images. This preprocessing makes the augmented input distinct from the original but similar enough to keep the data labels valid. This approach is easier and more time effective than manually collecting and adding new data to the dataset\cite{22}\cite{6}\cite{23}.

Dropout is a technique that can be used during training and can reduce overfitting in larger networks. During training a set of randomly chosen nodes are disabled which renders their connections temporarily removed from the network. This can last for one or a couple of iterations depending on randomness. The remaining parts of the thinned network consists of the nodes which did not get disabled and is then trained like any normal network. Dropout can therefore be viewed as training a set of smaller and randomly chosen distinct subnetworks. After training the subnetworks the full network can be considered used as an average of all the small trained networks\cite{22}.

![Figure 2: A typical CNN structure.](image)

### 2.2 Convolutional Networks

CNNs are trainable multi-stage neural network structures. The inputs and outputs of each stage consists of sets of arrays called feature maps. Consider a network that is trained to classify the content of 248x248 RGB images. The input Layer would then consist of a 248x248x3 array containing colour values for each pixel. Feature maps represent a unique feature extracted from input. A feature is defined as a measurable property of an observed phenomenon\cite{24}.

A CNN is primarily comprised of convolutional layers and pooling layers. The convolutional layer consists of a set of learnable filters which outputs feature maps. This is done by applying linear functions on a sub-region (kernel or receptive field) of all input matrices and systematically moving the kernel across the image data\cite{9}\cite{12}. Figure 3 shows the basic principle of the kernel.

![Figure 3: The basic principle of the kernel.](image)
2.2 Convolutional Networks

The convolutional layer can preserve information regarding positions of different features in the input data. Because of this the CNN structure is more suited for image classification than more traditional network structures[12][17]. Figure 2 shows a typical CNN architecture.

The pooling layer typically applies a pooling operator on non-overlapping subregions of individual feature maps received as input. The pooling operator can be sum, average, max or another form of function[9]. This reduces the resolution of the feature map for succeeding networks or nodes.

This downsampling is used in CNNs since distinct feature positions become less relevant as information moves through the network. Only their relative position in the original input data remain relevant for the succeeding network layers[25][9]. An example of max pooling can be seen in figure 4.

![An example of max pooling](image)

**Figure 4:** An example of max pooling.

Following the convolutional stages the feature map data is vectorized and processed by a set of fully connected layers. This layer is responsible for the final classification of the feature data extracted by previous stages. Fully connected layers have a more primitive structure inherited from other forms of ANNs.

Every node in such a layer is fully connected to the previous layer[2][12]. The last layer is responsible for the final classification and has one output node corresponding to each object class in the dataset. A network applied to CIFAR10 should therefore have an output consisting of 10 nodes[26].

2.2.1 Related Work

In 2012 Krizhevsky et al created the largest CNN thus far made. It contained five convolutional layers with some of them followed by sub-sampling layers. Finally the network had three fully connected layers. The number of neurons within the entire network was 650’000 and the number of free variables was 60 million. The network was tasked to classify the 1.2 million images of the ILSVRC-2010 contest in 1000 different classes. When training the network they utilized dropout and data augmentation to minimize overfitting. The end result was that the network was able to classify images with a top-1 error rate of 37.6% making it a benchmarking network for all image classification to date[6].

Simonyan et al created a CNN containing 16 layers in 2015[27]. By reducing the size of the receptive fields of the convolutional layers as well as increasing the total amount of layers they were able to outperform the records previously set by Krizhevsky et al. This further showed that increased network depth can have a significant impact on accuracy for object classification tasks[6].

The Network in Network (NIN) structure was proposed as an alternative to the conventional CNN structure in 2013 by Min Lin et al. As described in section 2.2 conventional CNNs usually consists of alternating convolutional filters and pooling layers ending with a fully connected classifier. The convolutional filters acts as a generalized linear model. Min Lin et al argued that this produced poor feature abstraction and suggested introducing a more potent non-linear function approximator. This was suggested because of the fact that features in image data are commonly not linearly separable[12].
2.3 Datasets

The NIN structure replaces the convolutional filters with micro-networks called multilayer perceptrons (MLPs). They are sequentially stacked together similar to how convolutional layers are. The structure replaces the fully connected classification layers with a global average pooling layer. In figure 5 the micro-networks and the global average pooling layer can be seen. This enforces a stronger relation between the feature maps and the classification categories. It has the advantage of reducing the amount of free parameters in the network due to pooling layers not having adjustable weights. This makes the NIN less prone to overfitting\cite{12}.

Min Lin et al tested a NIN consisting of three stacked MLPs (see figure 5) and one global average pooling layer. Each MLP consisted of a small CNN with three convolutional layers and one pooling layer. They were able to achieve state-of-the-art performance on the CIFAR10 dataset with an error rate of 8.81%. They utilized data preprocessing, data augmentation and dropout\cite{12}.

2.3 Datasets

During the training and benchmarking of any ANN it is important to have sufficient amounts of data to provide the network with. The negative effects of having small or poorly balanced sets of training data have been documented by several studies in the past\cite{28}. Following this paragraph is a list of five datasets which were deemed good candidates for testing the accuracy of the DCNNs written for this thesis, based on previous studies.

ImageNet is a large scale image dataset currently in development. Upon its completion the dataset is predicted to contain over 50 million labeled images. But in its 2015 release it has 10 million labeled images. The data is structured in a hierarchy based on the WordNet lexical database\cite{29}. Despite being in continuous development since 2009 the dataset has become a benchmarking dataset in machine learning. It is most commonly used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)\cite{7}.

The SVHN dataset is dataset containing over 600'000 digital images picturing different house numbers. The dataset contains two parts, one which contains 32x32 cropped images which can be used to train character recognition networks and have been used in several known studies in the past\cite{26}. The other part is a set of images in varying resolutions with a house number somewhere within the image. This dataset can be used to train networks which aim to solve the more complex problem of recognizing digits and number in natural scene images\cite{30}.

The National Institute of Standards and Technology have created a dataset of labeled handwritten digits used for character recognition called MNIST. The dataset contains a training dataset of 60'000 handwritten digits from over 250 different people and a test set of 10'000 digits from the same groups.

All of them are grouped into ten different classes, one for each digit. The individual character test data have all been cropped to 28x28 pictures, each containing one digit,
and have undergone anti-aliasing to improve the grayscaling of the images[31]. A subset of the dataset can be seen in figure 6.

![Image](airplane automobile bird cat deer dog frog horse ship truck)

Figure 7: A subset of the CIFAR10 dataset.

The two CIFAR datasets are labeled collections of small RGB-images, collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The content of both are taken from a larger, unlabeled dataset called 80 million tiny images. The CIFAR10 dataset contains 60,000 32x32 images, divided into 10 distinct classes corresponding to the content of the image, which varies from e.g. birds, cats, dogs, cars, making it very suitable for training network for image content classification. The CIFAR100 dataset is similar in content to CIFAR10 but has been divided into 100 distinct subclasses, with 600 images in each, creating a more challenging classification problem[5]. A subset of CIFAR10 can be seen in figure 7 (original is in color).

### 3 Methods

Nine networks were written based on the NIN architecture with differences both suggested by the NIN authors and by varying the number of stages (stage structures defined in section 3.3)[12]. The original NIN network definition was included in testing which brings the total to ten networks. The results achieved by Min Lin et al were not possible to entirely reproduce, although an accuracy of 87.44% was reached when training their implementation. The networks were trained using the Caffe library with python bindings on the CIFAR10 dataset[32]. The training process was accelerated by using cuDNN along with Caffe. CudNN is a deep neural network library specifically made for NVIDIA GPUs by NVIDIA which can accelerate training speed by a factor of six[33].

The NIN was chosen as a target of investigation in part due to its novel uninvestigated architecture and also because of Min Lin et al’s extensive documentation[12]. Another contributing factor in the choice of a network structure was the NIN’s position on the CIFAR10 top results[5]. Its documented state-of-the-art accuracy showed promise for further investigation.

Before training, the dataset was preprocessed by applying the same ZCA whitening and global contrast normalization as Min Lin et al when they proposed the NIN architecture[12]. The 10,000 testing images in CIFAR10 was used for validation data for each of the ten NIN versions.

#### 3.1 Dataset Utilization

The MNIST, SVHN, CIFAR and ImageNet datasets were considered for training and testing. MNIST was used only for comparing our results on CIFAR10 to see if the results were credible (see section 4). Both the SVHN and ImageNet datasets were discarded due to time constraints and lack of state-of-the-art hardware. The CIFAR10 dataset was thus used for training and testing. The 10,000 testing images were used for validation and the 50,000 training images for training.

The preprocessing techniques outlined in the Maxout Networks article by Ian J. Goodfellow et al (ZCA whitening and global contrast normalization) were used to preprocess CIFAR10 by Min Lin et al[34][12]. Preprocessing training data have previously been shown to speed up the training process and increase the performance of networks under
3.2 Programming Libraries

The programming libraries used for the thesis are cuDNN and Caffe with python bindings. The ten created versions of NIN were written in Caffe’s own prototxt format. This allowed for rapid development and fine tuning of networks during the training process. The training itself was greatly sped up by utilizing cuDNN functions, which required very little manipulation after setup.

Caffe was chosen as our DNN library due to its ease of integration with cuDNN and its prototxt network format. Since a GTX980 graphics card with 4GB of video memory was available for use cuDNN could speed up the thesis’ experiments greatly. Furthermore, Caffe’s prototxt network format could reduce the need to write our own code greatly. This could thus reduce human error and, again, speed up the thesis’ experiments.

3.3 Parameters

The ten NIN versions were tested by varying two main parameters; numbers of stages and stage depth. We define a stage as a series of convolutional layers with ReLU layers between them followed by a pooling layer. This stage structure can be observed as being stacked in the NIN structure. It was suggested by Min Lin et al to further extend the NIN by stacking more stages[12]. In this thesis this is extended upon by also investigating the effect of the depth of the stages on NIN performance. Varying the stage depth refers to changing the number of convolutional layers (with ReLUs in between them) in each stage. Thus if a stage depth of four is used the number of convolutional layers will be four in each stage. The kernel size of each network had to be adapted to its depth.

Varying the depth of all stages simultaneously was done in order to generalize the investigation in how the parameters would affect NIN performance. Due to time constraints varying the depth of each stage individually was not possible. Instead of ten nets there would be hundreds and such a thorough investigation was not within the scope of this thesis.

These two parameters were chosen in part due to Min Lin’s mentioned suggestion but also due to Krizhevsky et al’s success with deeper CNNs for ImageNet[6][12]. Krizhevsky et al showed that deeper networks can for certain problems and structures outperform their shallow counterparts[6]. Furthermore, because of NIN’s use of average pooling the risk of overfitting that is usually associated with DNNs is slightly mitigated.

3.4 Training

The ten networks were trained using a learning rate of 0.1 that was divided by 10 up to two times whenever learning stagnated. This is almost identical to how Min Lin et al trained the original NIN[12]. However, in the NIN article no maximum number of iterations are mentioned. 150’000 iterations was chosen as our maximum number of training iterations. This maximum number of iterations was chosen in part due to time constraints. It was also chosen due to networks usually not needing more than 100’000 training iterations before stagnating. The preprocessed data was fed to the network in batches of 128 images per training iteration. Testing was carried out every 500 iterations using the 10’000 testing images in the preprocessed dataset.

3.5 Measurements

Every 10’000 training iterations Caffe was set up to create a snapshot of the current weights and biases. This allowed us to, after the completed training, return to these snapshots and look at their scores on CI-
5 DISCUSSION

The data in section 4 was gathered after training the networks. This means that tests were run on each snapshot on the iterations listed in the results section. The tests were carried out with the 10'000 testing images from CIFAR. The tests ran for 50 batches with 100 images randomly selected for each batch. The accuracies in section 4 are the mean value of all testing batches for that particular network.

4 Results

Note that all of the network names below follow a certain format. A network labelled XsYd means that the network has X number of stages with a depth of Y each. Thus the 3s3d network refers to the original NIN structure. Our scores on CIFAR are shown in table 1. MNIST scores can be seen in table 2. All accuracies listed are percentages of successful classifications. The data gathered showed that a greater number of stages generally implied greater accuracy on CIFAR10.

Note that the networks with two stages all performed worse than the three staged networks (except for 3s6d). It is thought that this performance loss has to do with how the downsampling had to be adapted to the network sizes. In the original NIN the downsampling linearly got smaller for each layer the deeper the information went into the structure. This is also the case in all of the three staged networks investigated in this thesis. However, the two staged networks had to have their downsampling increased in order to reach sufficiently downsampled data for the outputs. This causes information loss which reduces the accuracy for these networks.

The 2s3d performed best of all two staged networks and the 3s2d performed best of the three staged networks. This seem to suggest that less complicated NIN implementations perform better than its deeper counterparts.

5 Discussion

Some conclusions can be drawn from the limited testing carried out. It is however difficult to draw conclusions from the accuracy data gathered. Overall the networks with less depth performed better than their deeper counterparts. This is probably due to information loss in the micro networks as the depth incrementally rises.

Although not described in section 3 increasing the dropout for later layers had little effect on the 3s6d performance. In fact the end result was the same as in table 1. Initializing the 3s6d training with weights from other better performing networks did not increase performance. Thus overfitting is likely the issue and the 3s6d structure either needs longer training time or other structural tweaks to achieve state-of-the-art accuracies.

As can be seen in both tables there are differences in accuracy between the 2s3d and 2s6d. The 2s3d has a top accuracy of 77.08% for CIFAR10 and 98.82% for MNIST. The 2s6d on the other hand has a top accuracy of 76.62% for CIFAR10 and 98.64% for MNIST. On the MNIST dataset there is thus an accuracy drop of 0.18%. For CIFAR10 there is a drop of 0.46%. Thus the depth of the stages does not seem to significantly impact NIN accuracy.

The 3s6d network performed poorly. In fact it performed the worst out of all ten networks. This suggests that some data was lost due to extensive feature extraction of the network stages. As stated in the NIN report the micro networks depend on dropout and are extremely prone to overfitting[12]. This is likely the cause of the 3s6d’s poor performance. During training the loss function showed high values for the 3s6d and
5.1 Constraints

Several factors limited the extent to which the NIN could be investigated. The prime constraint was time. The networks took several hours each to train which limited the scope of the thesis. The limitations imposed on computing power further increased the time needed.

Table 1: Accuracies from CIFAR10 tests on all ten networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>10k</th>
<th>30k</th>
<th>50k</th>
<th>70k</th>
<th>90k</th>
<th>110k</th>
<th>130k</th>
<th>150k</th>
</tr>
</thead>
<tbody>
<tr>
<td>2s2d</td>
<td>67.66</td>
<td>70.74</td>
<td>72.04</td>
<td>71.90</td>
<td>72.12</td>
<td>75.58</td>
<td>76.32</td>
<td>76.28</td>
</tr>
<tr>
<td>2s3d</td>
<td>68.56</td>
<td>69.82</td>
<td>72.46</td>
<td>71.30</td>
<td>73.56</td>
<td>77.08</td>
<td>75.94</td>
<td>75.94</td>
</tr>
<tr>
<td>2s4d</td>
<td>65.30</td>
<td>66.80</td>
<td>67.76</td>
<td>69.62</td>
<td>70.24</td>
<td>74.58</td>
<td>74.92</td>
<td>74.92</td>
</tr>
<tr>
<td>2s5d</td>
<td>65.84</td>
<td>67.46</td>
<td>67.80</td>
<td>73.40</td>
<td>72.62</td>
<td>74.16</td>
<td>74.42</td>
<td>74.16</td>
</tr>
<tr>
<td>2s6d</td>
<td>66.08</td>
<td>70.66</td>
<td>68.64</td>
<td>76.04</td>
<td>75.66</td>
<td>76.62</td>
<td>76.62</td>
<td>76.60</td>
</tr>
<tr>
<td>3s2d</td>
<td>70.12</td>
<td>82.68</td>
<td>83.26</td>
<td>83.92</td>
<td>83.26</td>
<td>86.94</td>
<td>87.32</td>
<td>87.44</td>
</tr>
<tr>
<td>3s3d</td>
<td>77.46</td>
<td>81.90</td>
<td>81.54</td>
<td>82.32</td>
<td>83.42</td>
<td>86.74</td>
<td>87.16</td>
<td>87.02</td>
</tr>
<tr>
<td>3s4d</td>
<td>70.59</td>
<td>80.50</td>
<td>81.24</td>
<td>82.54</td>
<td>86.16</td>
<td>86.46</td>
<td>86.24</td>
<td></td>
</tr>
<tr>
<td>3s5d</td>
<td>15.38</td>
<td>54.24</td>
<td>58.26</td>
<td>59.56</td>
<td>60.78</td>
<td>63.38</td>
<td>62.28</td>
<td>62.52</td>
</tr>
<tr>
<td>3s6d</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Accuracies from MNIST tests on all ten networks.

<table>
<thead>
<tr>
<th>Network</th>
<th>10k</th>
<th>30k</th>
<th>50k</th>
<th>70k</th>
<th>90k</th>
<th>110k</th>
<th>130k</th>
<th>150k</th>
</tr>
</thead>
<tbody>
<tr>
<td>2s2d</td>
<td>66.26</td>
<td>88.64</td>
<td>88.92</td>
<td>98.6</td>
<td>98.62</td>
<td>98.86</td>
<td>98.46</td>
<td>98.46</td>
</tr>
<tr>
<td>2s3d</td>
<td>98.08</td>
<td>98.66</td>
<td>98.6</td>
<td>98.46</td>
<td>98.66</td>
<td>98.82</td>
<td>98.82</td>
<td>98.78</td>
</tr>
<tr>
<td>2s4d</td>
<td>97.98</td>
<td>98.34</td>
<td>98.46</td>
<td>98.52</td>
<td>98.42</td>
<td>98.7</td>
<td>98.66</td>
<td>98.74</td>
</tr>
<tr>
<td>2s5d</td>
<td>97.74</td>
<td>98.34</td>
<td>98.44</td>
<td>98.54</td>
<td>98.52</td>
<td>98.6</td>
<td>98.6</td>
<td>98.6</td>
</tr>
<tr>
<td>2s6d</td>
<td>97.9</td>
<td>98.12</td>
<td>98.13</td>
<td>98.48</td>
<td>98.58</td>
<td>98.62</td>
<td>98.64</td>
<td>98.64</td>
</tr>
<tr>
<td>3s4d</td>
<td>98.84</td>
<td>98.9</td>
<td>99.06</td>
<td>98.96</td>
<td>98.98</td>
<td>99.4</td>
<td>99.28</td>
<td>99.32</td>
</tr>
<tr>
<td>3s5d</td>
<td>98.78</td>
<td>99.1</td>
<td>99.28</td>
<td>99.28</td>
<td>99.08</td>
<td>99.32</td>
<td>99.3</td>
<td>99.24</td>
</tr>
<tr>
<td>3s6d</td>
<td>98.3</td>
<td>98.58</td>
<td>98.76</td>
<td>98.9</td>
<td>98.72</td>
<td>99.08</td>
<td>99.18</td>
<td>99.18</td>
</tr>
</tbody>
</table>

3s5d networks which further suggests overfitting.

The number of stages did have an impact on NIN accuracy, unlike the layer depth. In the tables above there is a 10% accuracy loss when using two stages instead of three for CIFAR. There is also a 1% accuracy loss when using two stages instead of three for MNIST. The highest scoring network was the 3s2d version of the NIN, which is simply the original NIN with one less depth. This suggests that stage numbers affect NIN performance more than the depth of each stage, more extensive testing and research could improve on the conclusions of this thesis.

As suggested in the NIN article stages were stacked to create deep NIN networks[12]. However, the performance of these deep NINs were subpar in the conducted tests. This could possibly be due to time and training constraints. A fixed number of iterations were used and these deeper networks could have needed more time for training. Overfitting could also have caused their low accuracies. Deeper networks have previously been shown to be more prone to overfitting, as discussed in the NIN article[12].

5.1 Constraints

Several factors limited the extent to which the NIN could be investigated. The prime constraint was time. The networks took several hours each to train which limited the scope of the thesis. The limitations imposed on computing power further increased the time needed.
5.2 Further Research

As suggested above further research should study NIN stage number effect on performance. This could help achieve high-end results for CIFAR and potentially yield more conclusive results for implementations of DCNNs. Furthermore, future research could investigate deeper NIN versions more extensively by using data augmentation to a greater extent. With greater amounts of data deeper versions of the NIN could be trained and tested due to less overfitting.

It is the recommendation of this thesis to further study the core concepts of the NIN. Maintaining the idea of global average pooling but investigating micro-network structures could potentially increase deeper NIN accuracy.

The NIN structure has not been thoroughly studied. Because of this further investigation into its structure and its micro-networks is something this thesis recommends. Despite the constraints outlined in section 5.1 good accuracies were achieved. This along with NIN’s original success points to its further potential as an object recognition network.

5.3 Conclusion

The problem statement was carried out and investigated. However, several constraints on the project limited the extensiveness of the experiments. Thus only limited conclusions can be drawn from the data gathered. Generally networks with less depth but more stages performed better although this must be more thoroughly investigated.

An interesting result was that the 3s2d network performed best of all ten. This must however also be further investigated since the original NIN results were not reproducible. It is the conclusion of this thesis that less complicated NINs seem to generally suggest better results. By less complicated it is meant that the network stages have less depth. Further studies of the NIN could therefore potentially yield results better than those of Min Lin et al which could increase CIFAR10 accuracies.
6 References


