CONVOLUTIONAL NEURAL NETWORK FOR IMAGE CLASSIFICATION BASED ON TRANSFER LEARNING TECHNIQUE

by

Ghassan Mohammed Halawani

Bachelor of Science

King Abdulaziz University

2013

A master's research project

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Engineering

in

Electrical and Computer Engineering

Toronto, Ontario, Canada, 2018

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this project. This is a true copy of the project, including any required final revisions.

I authorize Ryerson University to lend this project to other institutions or individuals for the purpose of scholarly research.

I further authorize Ryerson University to reproduce this project by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

I understand that my thesis may be made electronically available to the public.

Abstract

Convolutional Neural Network for Image Classification Based on Transfer Learning Technique

2018

Ghassan Mohammed Halawani

Master of Engineering

Electrical and Computer Engineering

Ryerson University

The main purpose of this project is to modify a convolutional neural network for image classification, based on a deep-learning framework. A transfer learning technique is used by the MATLAB interface to Alex-Net to train and modify the parameters in the last two fully connected layers of Alex-Net with a new dataset to perform classifications of thousands of images.

First, the general common architecture of most neural networks and their benefits are presented. The mathematical models and the role of each part in the neural network are explained in detail. Second, different neural networks are studied in terms of architecture, application, and the working method to highlight the strengths and weaknesses of each of neural network. The final part conducts a detailed study on one of the most powerful deep-learning networks in image classification – i.e. the convolutional neural network – and how it can be modified to suit different classification tasks by using transfer learning technique in MATLAB.

Acknowledgements

A thesis becomes a reality with kind support and help from many individuals. I would like to extend my sincere thanks to all of them.

Foremost, I would like to thank **God** for the wisdom he bestowed upon me, the strength, peace of mind and good health in order to finish this research.

I would like to express my special gratitude and thanks to SACB **Saudi Arabian Cultural Bureau** in Canada for its financial help and support. Without it, I would never have been able to pursue my higher education in the field of Electrical and Computer Engineering.

I would like to acknowledge my professor **Dr. Kaamran Raahemifar**, who always believed in my success and abilities, and gave me the chance to prove myself.

I also would like to acknowledge **Dr. Kandasamy Illanko** for his valuable advice in improving this project in particular.

My thanks and appreciations also go to my dear friends **Fares Shaikh Omar** and **Mazen Sarhan** who sincerely helped me to overcome many issues during my study

Dedication

I deeply dedicate this work to my parents who've worked hard for me to get here and inspired me to achieve, and to my dear wife **Bayan Wali** who patiently sacrificed time with me so that I can dedicate myself to this work. I also dedicate this work to the beloved one, my daughter **Layan Halawani**, who drew a smile on my face when I was tired and depressed

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 The Success of NNs and the Type of Problems NNs Address

Artificial neural networks (NN) are computational models derived from the simulation of typical human brain activities such as image classification, pattern recognition, and language understanding. Computer programs are very useful tools for performing tasks with high speed, accuracy, and reliability. However, these information processing systems are not intelligent, which requires human intervention for updating. The use of prior experience (i.e. data) for learning is what makes NNs intelligent and powerful.

Many types of NNs have been used in the last three decades [1]. Each type of NN is suitable for certain applications. Deep neural networks (DNN) are used for applications such as autonomous cars for path recognition, learning systems [2], and skin cancer classifications [3]. Rough neural networks are used for forecasting [4] and classification [5]. Moreover, critical-based neural networks utilize a supervisor to evaluate the performance of the network, based on as specific cost function. This type of network is used for robust optimal tracking control [6] and data-driven control systems [7].

A convolutional neural network (CNN) is a major type of DNN, which can be applied for classification, regression, and image recognition. Image classification is one of the most common tasks that CNNs can perform. The way CNNs can classify images mainly depends on the architecture of the neural network (e.g. the type, size, and order of the layers). Two networks can have the same architecture, but will behave differently if they use different datasets. In addition, the layer itself comes with many parameters such as weights and bias that play a significant role in the network performance.

1.2 The Architecture of Neural Network

The simplest neural network architecture can be described as black box with inputs and outputs. The inputs of the NN is a vector of any dimension as shown in Fig.1 and is defined as $x \in R^n$, where R^n indicates the n-dimensional Euclidean Space. x_i and y_i represent the input and output of the network respectively. These input vectors could be real numbers, integers, or binary numbers as components. In general, all neural networks, whether shallow or deep, can perform classification. The input vectors in NNs are classified into classes or categories. Each class is defined by a particular output. Inside the black box is the architecture of the neural network, which will be covered in detail in chapter 2. The perceptron is the fundamental building block of the neural network.

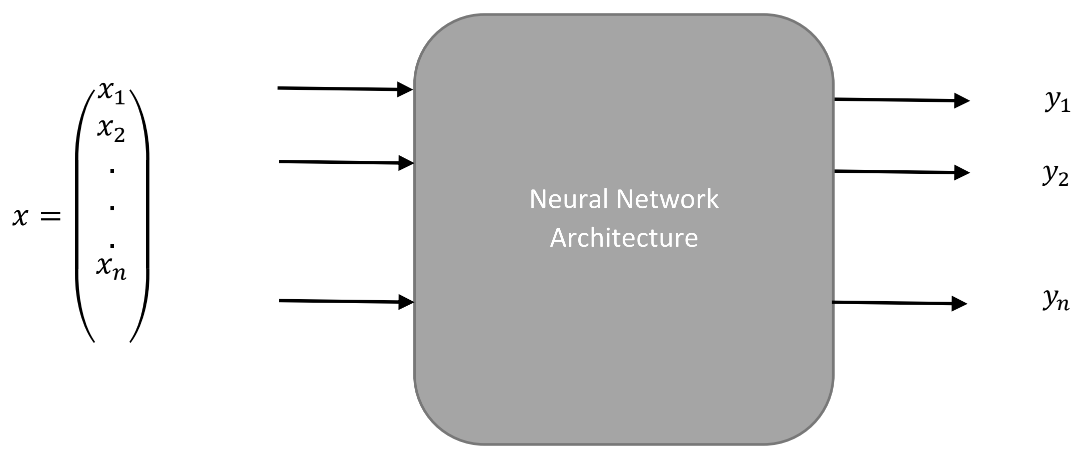


Figure 1: A Neural Network as a Black Box

1.3 Fundamental Building Block and Parameters

The basic building block of all NNs is a single neuron, which only has one input and a single binary output. In general, this basic neuron does nothing but classify vectors into two groups, provided these vectors can be separated by a linear boundary in space and is defined in Equation (1). However, the basic neuron simulates multiple basic functions of the biological neuron, such as evaluating the intensity of each input, summing up the different inputs, and comparing the result with an appropriate threshold. Ultimately, it determines the output value [8]. The basic neuron is presented in Fig 2.

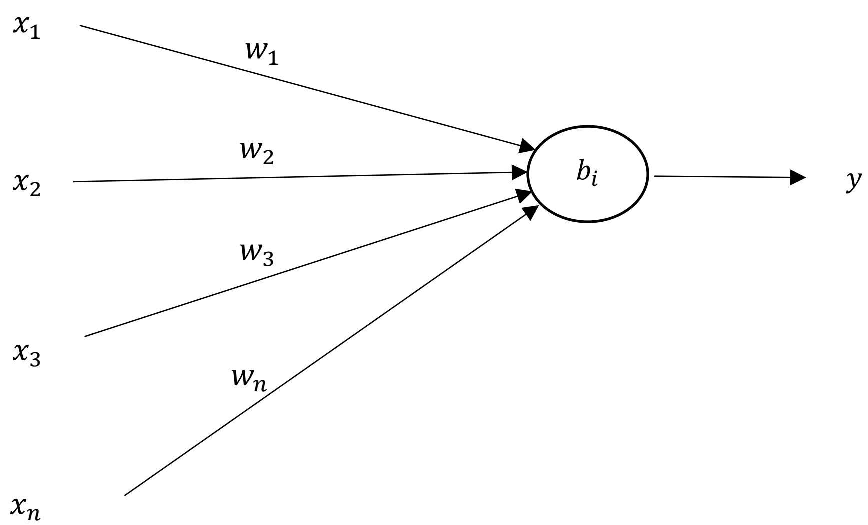


Figure 2: The Basic Neuron

w_n and b_i are the weights and the corresponding bias. The weights are assigned to each input of the neuron for modulating the influence of this input on the total sum to determine the potential of the neuron. Both weights and bias are adjustable parameters of the NN.

$$f(z) = \begin{cases} 1 \quad if\ b + \sum_{i=1}^{n} w_i \,.\, x_i > 0 \\ 0 \quad if\ b + \sum_{i=1}^{n} w_i \,.\, x_i \leq 0 \end{cases} \qquad (1)$$

1.4 Multilayer NNs and Activating function

The output of each neuron in the NN is a modified base on a particular function. This function is called an activating function, which is applied to the output of the neuron to determine its actual potential. In other words, it improves the performance of the NN. However, no one has been able to explain why this is so. There are several activating functions used in NNs. Yet in this thesis, the rectilinear function is used and is defined in Equation (2).

$$f(z) = \begin{cases} z & if\ z > 0 \\ 0 & if\ z \leq 0 \end{cases} \tag{2}$$

A typical NN can have multiple basic neurons, arranged and connected in layers as presented in Figure 3. Each layer added to the network increases its computational capacity. The layers between the input and output layers are called the hidden layers.

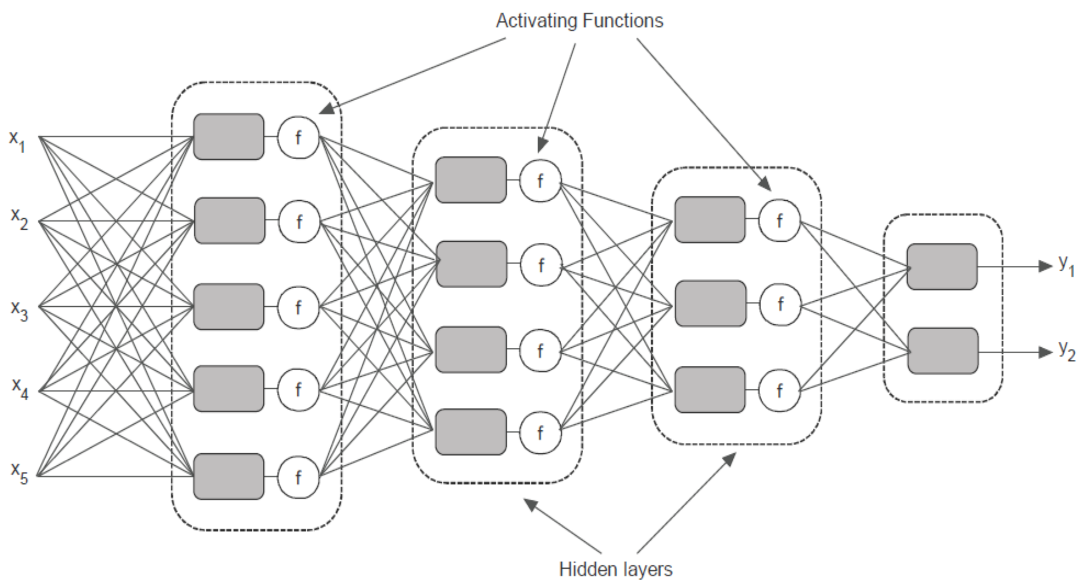


Figure 3: Hidden Layers and Activating Functions

1.5 Learning Strategies of the Neural Network

Learning strategies refer to how the network is trained in different approaches. The learning strategies are divided into supervised, unsupervised, and reinforcement learning. In supervised learning, the network is learned by providing labeled or known inputs and outputs (i.e. the training set). Then the learning rule is applied to adjust the parameters of the network to match the network output to the target. One of the most fundamental algorithms that is used for supervised learning is called back propagation.

In unsupervised learning, the network is provided unlabeled input data, and there are no target outputs available. In this type of learning, the neural network is clustering or classifying the input data based on their statistical features, such as the mean, the variance, and the standard deviation. Cluster analysis and self-organizing maps are the most common types of unsupervised learning.

Reinforcement learning is similar to unsupervised learning. Yet in reinforcement learning, the network algorithm assigns a score. This score is a measure of the network performance over some sequence of inputs.

1.6 Training and Testing the NNs

A typical deep NN may have thousands or millions of weights and a corresponding number of biases. A fraction of the dataset is mostly used to determine the value of these parameters. In other words, the network is regulated by changing the input weights so that actual outputs (i.e. what we calculated specifically) match the target (i.e. what we want to obtain). This fraction of data is called training data. Figure 4 shows the training phase of the NNs. For one of two dimensions of input vectors, deriving the parameter's values can be performed analytically; but in large-input vectors, this analytical approach does not work, which is why the training phase is used. The remaining input data (called testing data) is used later to test the network before it is put to use in the real application.

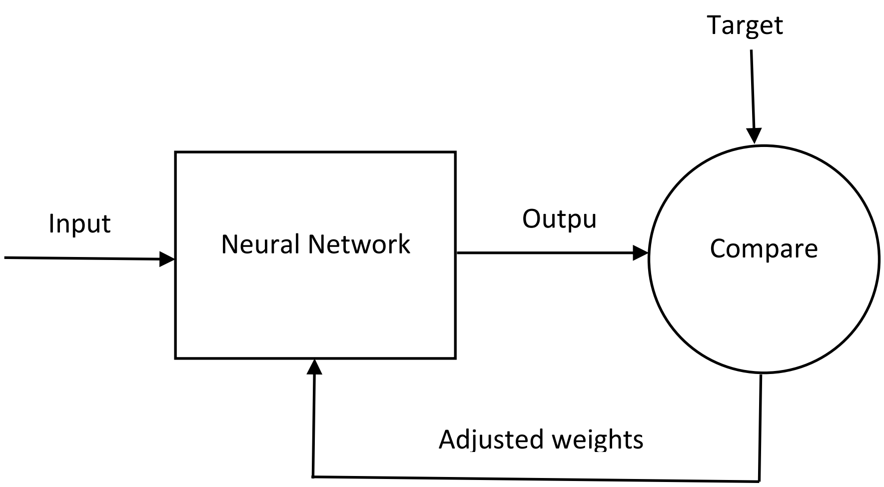


Figure 4: Flow Chart of Training Phase

1.7 Deep Neural Networks

In neural networks, the term deep refers to the number of layers in that network. In particular, the number of hidden layers is what define the deep networks – the number of hidden layers characterize the size of the NN.

In deep networks, there are multiple hidden layers, while in shallow networks only one hidden layer is presented. Choosing the optimal size of the NN is implemented by trial and error because no analytical approach has been found so far. In addition, no analytical solution has been found for determining the number of neurons in each layer. The greater the number of hidden layers and neurons in each layer, the more complex tasks the NNs can perform. But this runs a risk of overfitting; it may generalize poorly to future data. Moreover, NNs with many neurons can be costly and slow to train.

1.8 Objective of the Thesis

In this thesis, a deep NN, such as convolutional neural network (CNN), is used to classify thousands of images into seven classes using the transfer learning technique. An academically proven pretrained network, such as Alex-Net, is used and modified for classifying images that have never been seen before by Alex-Net. This implementation was built using MATLAB library.

1.9 Thesis Outlines

This thesis is organized as follows: an ordinary neural network is explained in detail in Chapter 2. In Chapter 3, different architecture of the NN is described, including the recurrent neural network and what is used for. Chapter 4 presents the convolutional neural network, which is used in the architecture of this thesis. Ultimately, the experimental results of the CNN are presented in chapter 5.

Chapter 2

Ordinary Neural Networks

The ordinary neural networks are those that have only one hidden layer, and are sometimes referred to as shallow network. The other way around, the recurrent neural network (RNN) and the convolutional neural network (CNN) refer to deep networks that have multiple hidden layers.

2.1 Basic Perceptron

As explained briefly in Chapter 1, the basic perceptron can have a single input vector x of any dimensions at a time and provides a single output y. Each input vector is multiplied by a weight w_i and the results are added [9]. The obtained results of the basic neuron are subject to a particular function (i.e. Activating Function) before it applies to the output of the neuron. This activating function determines the actual potential of the neuron. The defining equation of a neuron is shown in (2.1).

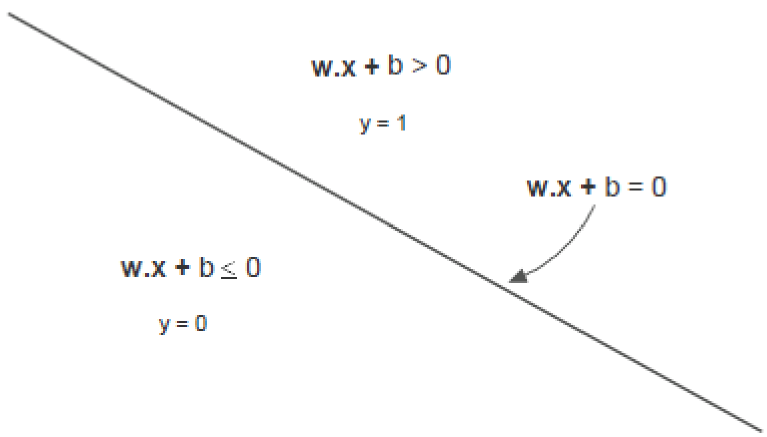


Figure 5: Separating Line of a Basic Perceptron [10]

$$f(z) = \begin{cases} 1 \quad if\ b + \sum_{i=1}^{n} w_i \,.\, x_i > 0 \\ 0 \quad if\ b + \sum_{i=1}^{n} w_i \,.\, x_i \leq\ 0 \end{cases} \tag{2.1}$$

Based on the dimension of the input data x, the separating boundary of the basic neuron could be straight line, a plane, or hyper plane. Figure 5 shows a boundary line of a single perceptron.

2.1.1 Perceptron Learning Rule

The process of modifying the parameter values (weights and bias) of a neuron is called Learning Rule. It is an update procedure that uses a particular algorithm (i.e. training algorithm) such as a steepest decent algorithm to find the optimal values of the parameters [10]. This optimal value means the error e between the actual output y and the target input t of the neuron is zero (2.1), and the neuron separates two classes perfectly. However, it is possible to have multiple optimal values that separate two classes perfectly.

$$e = y - t \tag{2.2}$$

For the weights, the product of the error and the corresponding input x is added with the weight. While for the bias, the error is just added. The Equations from (2.3) to (2.5) show how parameters are updated.

$$w_1 \coloneqq w_1 + ex_1 \tag{2.3}$$

$$w_1 \coloneqq w_1 + ex_1 \tag{2.4}$$

$$b \coloneqq b + e \tag{2.5}$$

When the last input vector is fed to the neuron, the parameter values of the neuron will be fixed at the end of training.

2.2 Single Layer Neural Networks

In a single layer network, multiple neurons are present rather than one neuron. Each neuron is connected to the all components of the input vector. Figure 6 shows that if the dimension of the input vector is R^n, then each neuron will have n+1 parameters, where n is the number of weights and 1 is the corresponding bias. In general, the parameter value of each neuron will have a different value for the other neuron. Yet in other types of deep neural network such as CNN, the case is different.

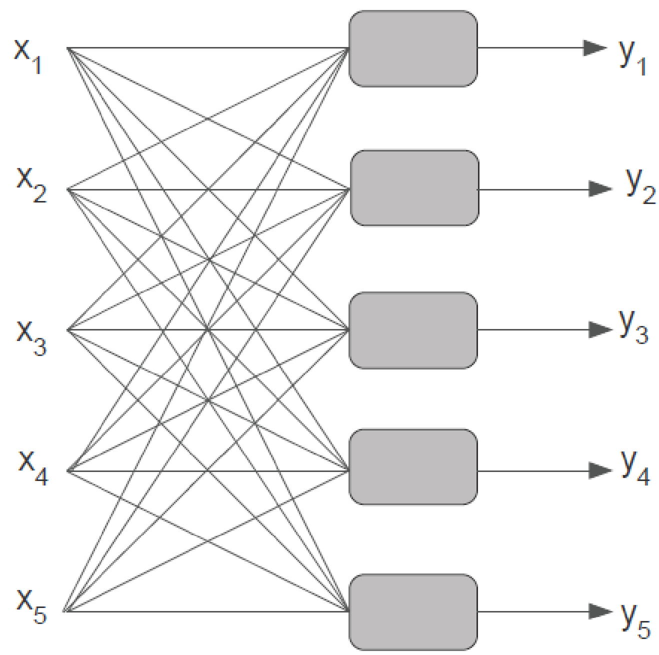


Figure 6: A single Layer with 5 Perceptron

In single-layer, multi-neural network, classification of multiple outputs can be achieved. However, these outputs must be linearly separable; otherwise, the error rate will be high. This drawback of a single layer neural network can be solved using a multi-layer neural network.

2.3 Multi-Layer Neural Network (MLN)

In a deep neural network, a typical number of layers may range between two to 15 hidden layers (Figure 3). Each neuron in each layer is connected to all outputs of the neurons in the previous layer. The network that has two or more layers is more powerful than a single hidden layer neural network. The main reason is that MLN can classify convex polygons by further processing the linear boundary of the neuron. This process is called function approximation [11].

2.3.1 Convex Optimization

Convex optimization is a subset of optimization researchers used to find the minimum and maximum points of in a convex function over a convex set. Convex optimization is utilized in NN to find the minimum point that minimizes the error function [12]. This error function, and the algorithm used to deal with it, is explained in detail in the following sections. The optimality condition of a convex function says if a function f is convex and differentiable in domain D and $\nabla f(a) = 0$, then $f(a)$ is the global minimum of f in D. Before using this theorem, the function must be proven to be convex. There are many mathematical ways to prove a function is convex, but this is not the interest of this research.

2.3.2 Steepest Decent Algorithm

Steepest decent is a powerful algorithm that is utilized to find the parameter values that minimize the error of the function [13]. The maximum rate of change of the scalar function f is given by $\| \nabla f \|$ and it takes the place of ∇f. In other words, if $\nabla f(x)$ moves in the direction of fastest rate of increase of f, at point x, this means $-\nabla f(x)$ moves in the opposite direction with a maximum rate of decrease of f, at the point x. Therefore, if an arbitrary point x_0 is chosen in the domain of a convex function, and it travels in the direction $-\nabla(f)$, it is most likely the point where $\nabla f(x) = 0$ will be reached.

The steepest decent algorithm is sensitive to what is called step size. This term refers to how far $\nabla(f)$ will travel along the convex function to reach the minimal point. A large step size may overshoot the optimal point, while a very small step size will result in too slow an algorithm. Obtaining acceptable step size depends on the function to be optimized and is mostly determined by trial and error.

The optimization problem of a non-convex function becomes a very difficult. In NN, if the steepest decent algorithm is used on a non-convex function, it may reach the local minimum rather than the global minimum. In such a scenario, the steepest decent can be run many times in different initial points until the global minimum is reached.

2.3.3 Error Function

Most likely, an error in the output of a neural network will be shown at least once because the starting point of parameter values in the steepest decent algorithm are initialized random values. There are many types of error functions used in NN, but the most common one is the square error function [10], and it is defined as in (2.6).

$$f_e = \sum_{i=1}^{m} \big(t(i) - f_o(i)\big)^2 \tag{2.6}$$

$t(i)$ and $f_o(i)$ denote the target output and actual output of the network, respectively, and i represents the index of the input data.

2.3.4 The Back-Propagation algorithm

Back propagation algorithm is based on the steepest decent algorithm to find the parameter values that minimize the prediction error of a neural network. In a multi-layer network, back propagation considers the error output of the network as a multivariable function of the parameters of the network. For this reason, back propagation assumes the error function as convex function.

Chapter 3

Recurrent Neural Network

3.1 Time Delayed Versions of the inputs

A recurrent neural network (RNN, is a type of deep learning network that has different architecture than other networks. The main difference in architecture is that RNN may take time-delayed versions of the same input series, due to the different conduction time and the length of the input series itself. Moreover, it may have feedback connections from the outputs of the later layers to the inputs of the previous layers, which may also lead to time delay at the output [14] [15]. A salient feature of RNN is that the output does not only rely on the current input, but also a series of earlier inputs. RNNs are suitable for time-series data applications, such as language and speech processing, musical analysis and processing, and stock market prediction and financial engineering. In other words, RNNs take a series vectors as input and provide a single vector as output at any given time.

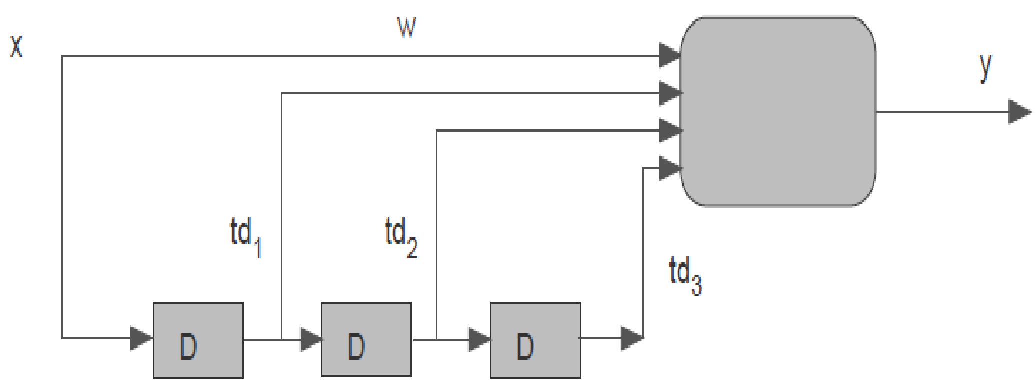
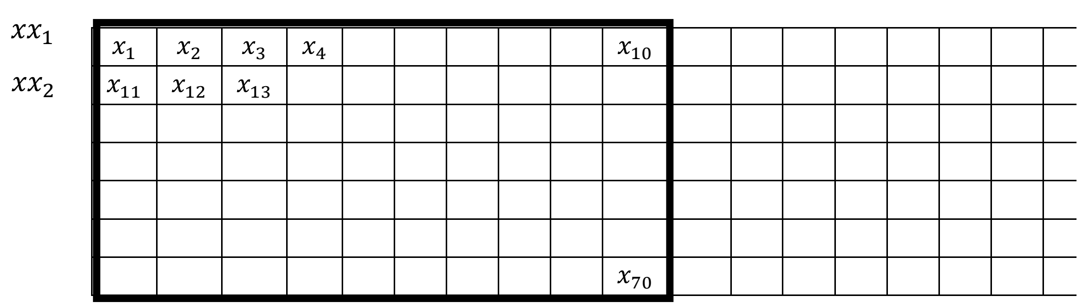


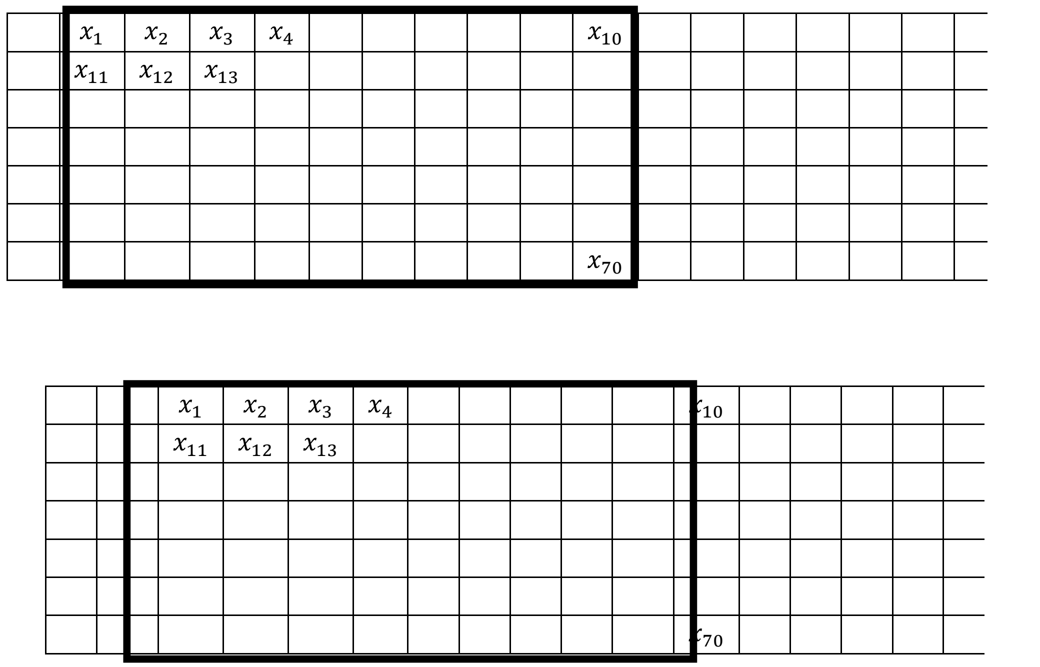
Figure 7: Pattern Recognizer [10]

Figure 7 shows the simplest type of RNN, a single recurrent neural network, that receives a stream of binary numbers and provides a binary stream at output, but has no feedback connections. The key function of this network is pattern recognition of an input sequence [16] [17]. For instance, if a RNN is designed to recognize 00110 pattern at the input, the resultant output of the network show 1; otherwise, the output is zero.

3.2 Sliding Window Technique

The sliding window technique is used to capture the time delay between the cause of the event and the actual event [18]. For instance, if a neural network is designed to predict a stock performance that relies on three stock values over last 10 days among 2,000 days' data, then the window size will be 8 times 10, where 8 and 10 represent the number of inputs and time delays respectively. Hence there would have been 80 inputs, if a td_ind w_i are collected. This window will slide over the data of 2000 days to predict the stock value of the next day. However, the size of the window and segment may increase until less error approximation is reached [19]. The process of the sliding window is shown in Figure 8. In neural networks, part of the 2000 days' data is used for training while the rest will be used for testing. The back propagation of this network is simple because there is no feedback connection.





$x()$

Figure 8: The Sliding Window

3.3 Neural Network with Feedback Connections

A recurrent neural network uses feedback connections from one or more of its output neurons as an input in selecting the next output [20]. This means the network output depends on the current input at time t and the past inputs at time $t - i$. Hence the name, recurrent neural networks. Just like in sequential digital circuits, the outputs are fed back after time delay. However, applying a back-propagation algorithm with presence of a feedback connections becomes difficult because it there is a complex web of dependencies between the output of the neurons and the parameters [10]. There are two versions of back-propagation algorithm for recurrent neural networks, but the focus of this chapter is on one called back propagation through time (BPTT).

3.4 Back Propagation Through Time

Back propagation through time (BPTT) is perhaps the most commonly used algorithm in recurrent neural networks. BPTT can be used for feature classification in recurrent neural networks. As per theory, in BPTT algorithm, the threshold of the network and weight values are initialized randomly. Then these values are adjusted accordingly, using the training samples, to minimize the error function [21]. Figure 9 shows a two-layer neural network that has two random time-series inputs and three feedback connections. Before applying the BPTT on the network, writing the defining Equation (3.1) will make the calculation easier.

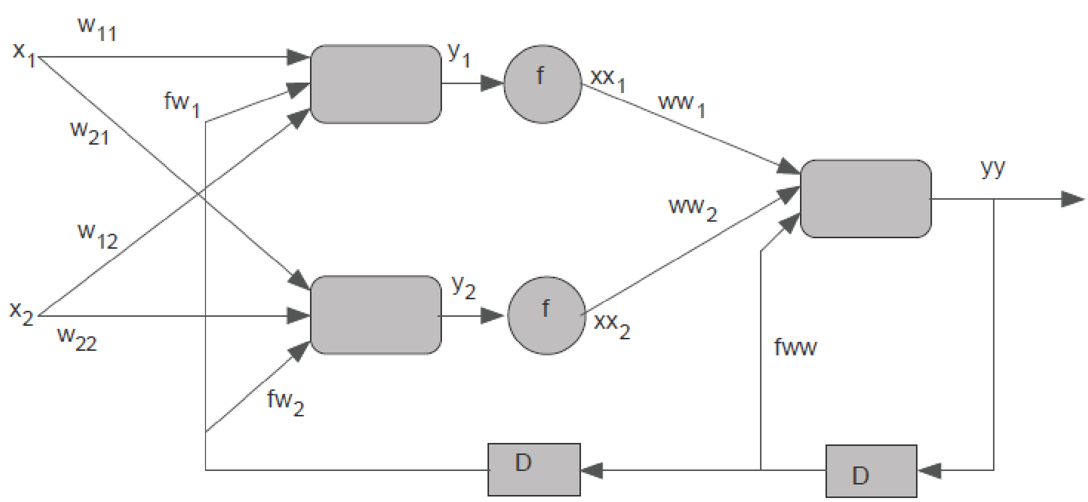


Figure 9:A Neural Network with Feed Back Connection [10]

$$yy(i) = bb + ww_1 xx_1(i) + ww_2 xx_2(i) + f_{ww} yy(i-1) \quad (3.1)$$

By differentiating both sides with respect to bb, Equation (3.2) is obtained.

$$\frac{yy(i)}{\partial bb} = 1 + f_{ww}\frac{yy(i-1)}{\partial bb} \tag{3.2}$$

This shows that the derivative at time i depends mainly on the derivative of the earlier time $i-1$.

By realization, the partial derivative can be calculated by two methods. First, at each time slot, the partial derivative of that time slot can be saved, then used to calculate the partial derivative of the nest time slot. Secondly, the partial derivative can be calculated in a specific time by applying the recursive relation all the way back to the first time slot.

Chapter 4

Convolutional Neural Network

4.1 Why ONNs are Not enough

Although a typical ordinary neural network (ONN) can be used for image classification, it becomes insufficient when utilized to classify complex images. For instance, images that contain one salient feature are easily classified by ONNs. However, when a salient feature of an image occupies only a small part of the overall image, or when the image has complex objects, or has many objects or features, the ONN cannot perform classification [10]. For this purpose, a convolutional neural network (CNN) is used to focus on multiple features in an image, rather than looking at an image as whole.

4.2 Architecture and Operations of CNN

A convolutional neural network (CNN) is one of the most common algorithms for deep learning with images and video. Like other typical neural networks, a CNN consists of an input layer, an output layer, and several hidden layers in between [22] [23]. In CNN, an input image is subject to three different operations when it passes through the hidden layers: convolution, pooling, and a rectified linear unit (ReLU). In addition, a process called normalization is applied in between of these layers. These three operations are repeated many times over the hidden layers, where each layer is learning to detect a different feature. In Figure 10, a CNN for classifying images into two categories is presented.

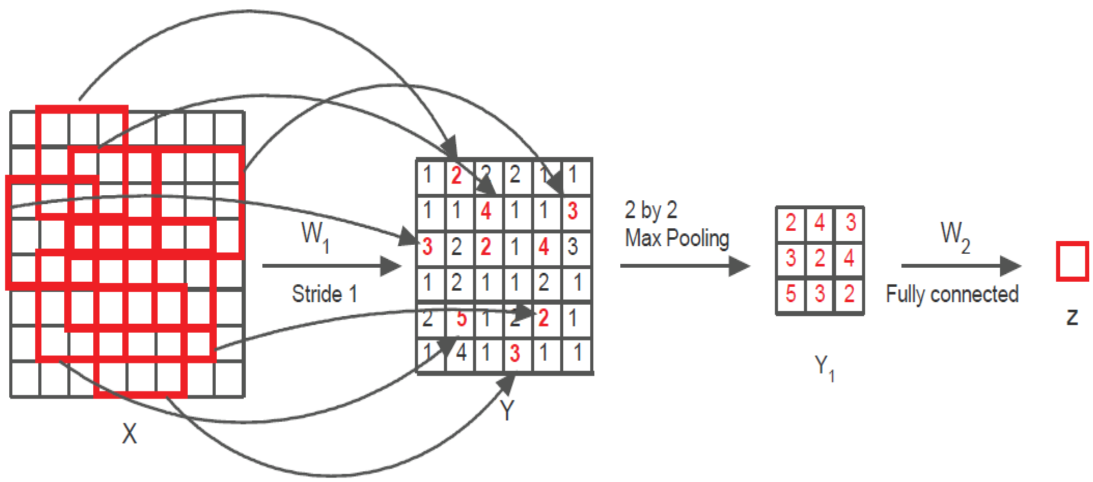


Figure 10: A CNN for Classifying Images into Two Categories

4.2.1 Convolutional Filters

In engineering, the term convolution refers to moving weighted average. For instance, if there are two vectors, as shown in Fig. 11, the multiplication of these two vectors is just dot product, and the result will be one value. However, when the second vector is moving over the first one and dot product is applied at each move, the result will be a third vector in different values. The moving vector (i.e. the second one) is the weight vector that functions as the filter of the first vector. For this purpose, it is called a convolutional filter. The main objective of a convolutional filter is to extract image features [24].

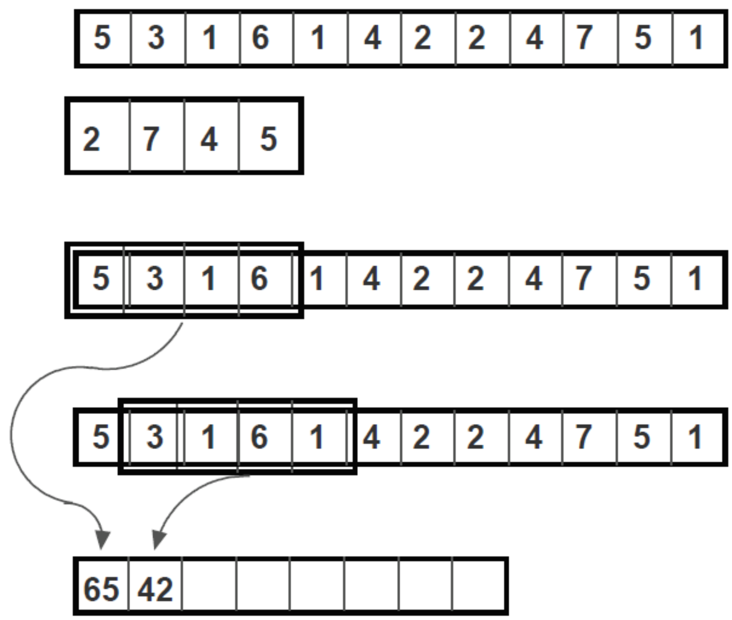
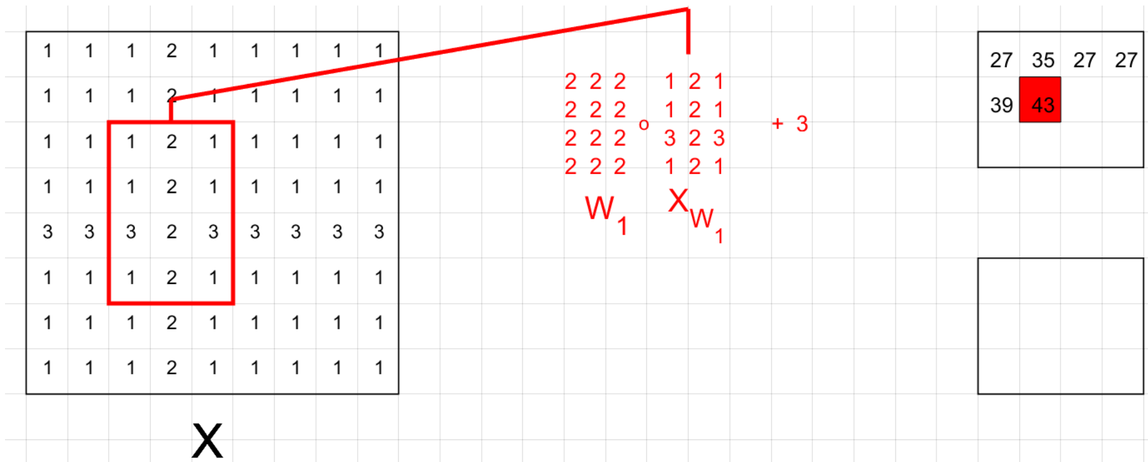
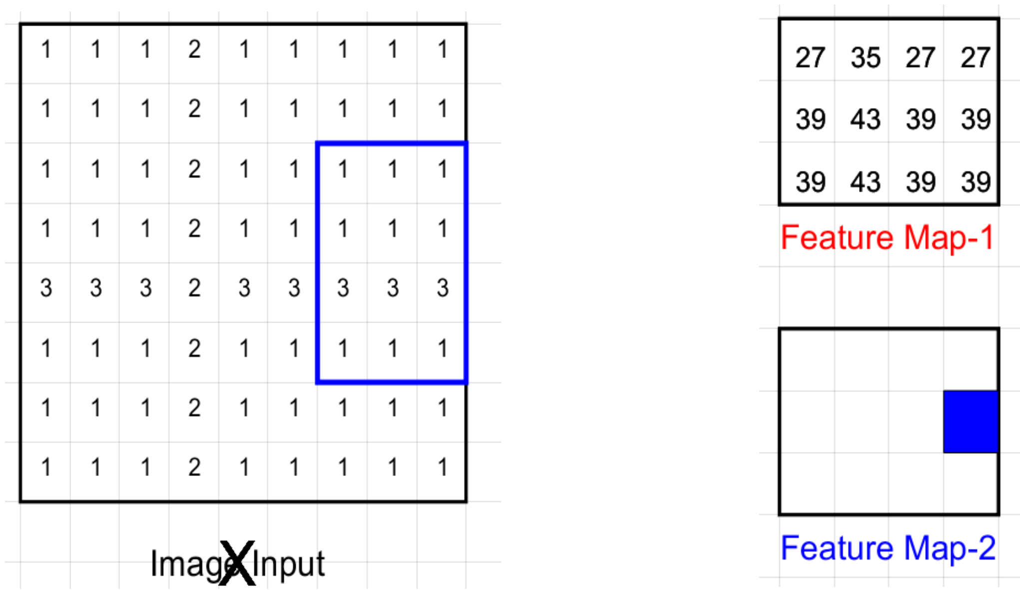


Figure 11: Convolutional is Moving Weighted Average

In CNN, the input image has many pixels, and each pixel has a specific value [10]. When an image is fed to a CNN, this image is considered as a matrix with many values, where the pixel values represent the values of this matrix. A convolutional filter is used as second matrix in smaller dimensions than the image itself (i.e. the first matrix) for applying dot product. By doing so, the entries of the two matrices are multiplied and summed up, then the bias is added with the result of the dot product. The value of the dot product will be the pixel value in the next layer. The filter now is shifting by a specific amount to the right; this shift is called stride. The smaller the stride, the more accurate, but the greater the network becomes. Stride is normally set in such a way that the output volume is an integer and not a fraction. With each shift, the dot product is applied until the filter slides across the whole picture. Finally, another matrix is created in the second layer; this matrix is called the feature map, which has multiple pixel values. Each value represents a specific feature on the input image itself [25] [26]. This process is shown in Figure 12.



(a) First feature Map extraction



(b) Second feature map extraction

Figure 12: Two-Dimensional Convolution and Feature

The dimension of the feature map in the second layer is smaller than the input image but bigger than the convolutional filter. The dimension of the feature map becomes smaller as the process continues until it becomes too small. Then the pixel values of the last layer is connected to one output neuron by a weight matrix. In this case, the output neuron is fully connected to the last feature map. Yet in the early layers of the network, preserving as much information about the original input volume is important, to ensure low level features can be extracted. To do so, a process called padding is used. For instance, when a convolutional filter is applied over an input image, the output size will be smaller than the input volume itself. However, if an input image needs to be the same size in the next layer and is multiplied by a convolutional filter, then zero padding of specific size is applied to the first layer [27]. This would result in an input volume as shown in Figure 13. Zero padding pads the input size with zeros around the boarder and is defined by Equation (4.1).

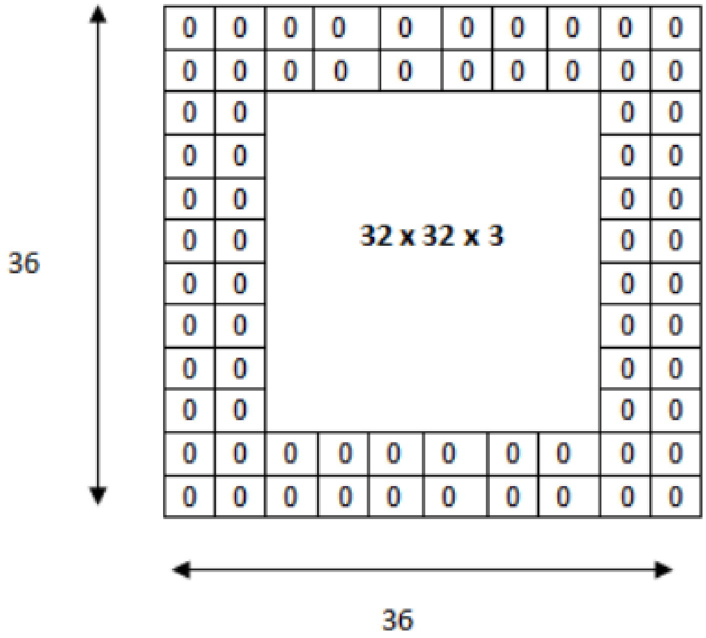


Figure 13: Zero Padding of 2 on Input Image

The input image can be a two-dimensional grayscale image or a colored image, where the three fundamental colors (RGB) represent the third dimension. The dimension of the feature map is given by Equation (4.2) to (4.3), where H and W represent the height and width of the feature map respectively. F is the filter size, S is the stride, and P is the padding.

$$\text{Zero padding} = \frac{(F-1)}{2} \tag{4.1}$$

$$W = \frac{W-F+2P}{S} + 1 \tag{4.2}$$

$$H = \frac{H-F+2P}{S} + 1 \tag{4.3}$$

Like ONNs, the weights in CNNs are initialized, and then a back-propagation algorithm is used to train the network to find the weights that minimize the error function at the output. However, in ONNs, each entry input has a different weight value, while in CNNs multiple entries (pixels) could share the same weighted values as shown in Figure 14, [28].

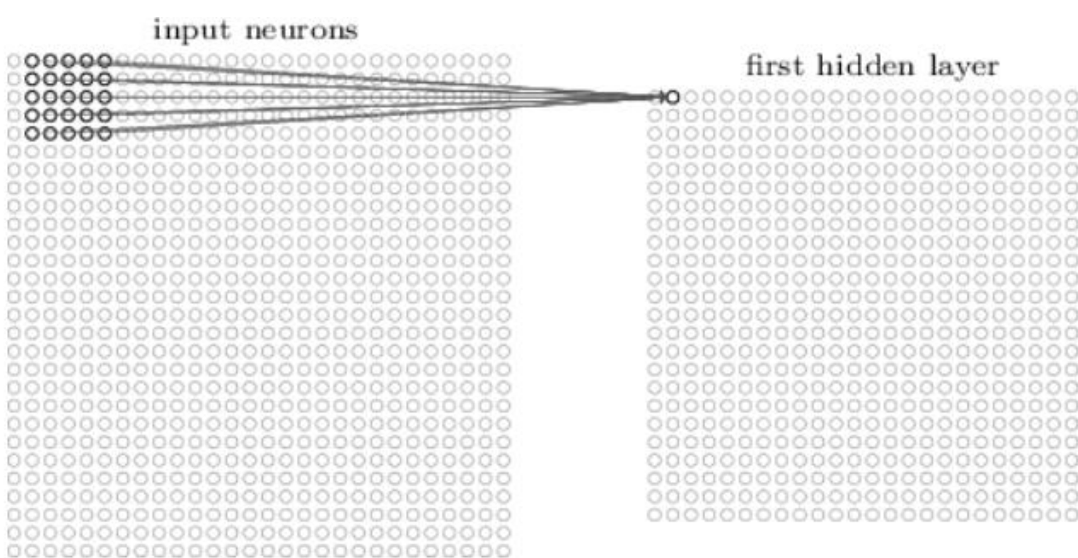


Figure 14: Weights Shared by Many Neurons and Represented as One Neuron in the Next Feature Map

4.2.2 Rectified Linear Unit (ReLU)

Moving from one feature map to another is implemented through a rectified linear unit, which is the activating function. Rectilinear functions are mostly more effective than other activating functions, based on the results that appear when they are utilized in neural networks. By using ReLU, the training process becomes faster and more effective by mapping negative values in the previous feature map to zero and maintaining positive values, as shown in Fig 15.

18	5	13	20
-6	-25	-7	11
1	7	10	20
-3	-16	-8	-8

Feature map 1

ReLU

18	5	13	20
0	0	0	0
1	7	10	20
0	0	0	0

Feature map 2

Figure 15: Activating Function (ReLU)

4.2.3 Subsampling (Pooling)

Sub-sampling or pooling refers to a filter that moves across the feature map and picks or calculates one value as representative for all numbers under the filter. The resultant feature map from sub-sampling will have a smaller dimension than the previous one. There are different methods of pooling, and the most common one is max-pooling, where it picks the maximum value among the numbers under the filter. The convolutional layers are separated by the pooling layers. Figure 16 demonstrates how max-pooling is applied. The key objective of subsampling or pooling operations is to reduce the number of parameters the network needs to learn about, thus reducing the computation cost. Second, it controls the overfitting problem. The overfitting term refers to when the weights of a neural network is so tuned to the training examples that they cannot be generalized properly during the testing phase. A symptom of overfitting is when a small error rate is found in the training phase, but in test phase the error rate is much larger.

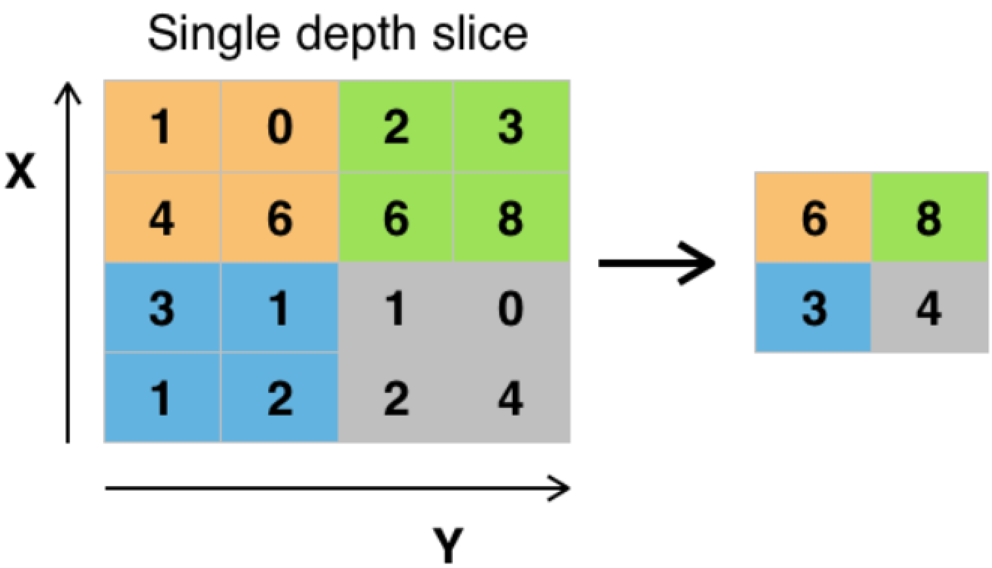


Figure 16: Sub-Sampling (Max-Pooling)

4.2.4 Dropout Layers

The main function of the "dropout" layer is to reduce the overfitting problem. Simply, a random set of the hidden neurons is set to zero. By doing so, these neurons do not participate in forward and backward operations, shown in Figure 17. Therefore, every time an input image is presented, the neural network takes a different architecture. This technique reduces complex co-adaptations of neurons, since a neuron cannot depend on the other neighbor neurons. Therefore, it is obliged to learn more robust features that are useful in connection with multiple subsets of the other neurons. This layer is only used during the training phase; but in the testing phase, all neurons are used.

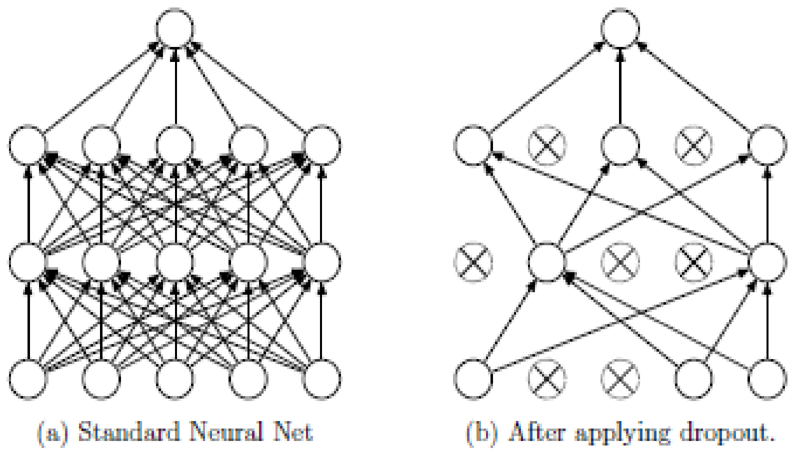


Figure 17: Dropout model. Left: standard neural network. Right: thinned network with dropped out unit

4.3 Training CNNs

The training phase of a CNN is much harder than the typical ONN, as the input color image that has 200 by 300 pixels is fed to a CNN as 200 by 300 by 3 pixels. The third dimension represents the primary colors: red, green, and blue. Therefore, three different matrices are created for each color matrix, and then all convolutional and subsampling filters are applied for the three colors independently. Based on the number of layers, the number of input data, and the number of parameters, the training phase in CNN may take between hours to several days. For instance, some pretrained CNN such as Alex-Net – which consists of 25 layers, has 12 million input images, and uses around 60 million parameters

– needed a whole week to classify 1,000 possible categories during the training and testing phase. The modified CNN in this thesis needed almost five hours to train the network.

The learning rate and momentum are the two key parameters that affect how the training algorithm updates the weights. As mentioned earlier, the main goal of training is to minimize the error or loss function, which is a measure of how badly the network performs on the training data. A common way to visualize the training process is to imagine a function that has summits and valleys as shown in Figure 18.

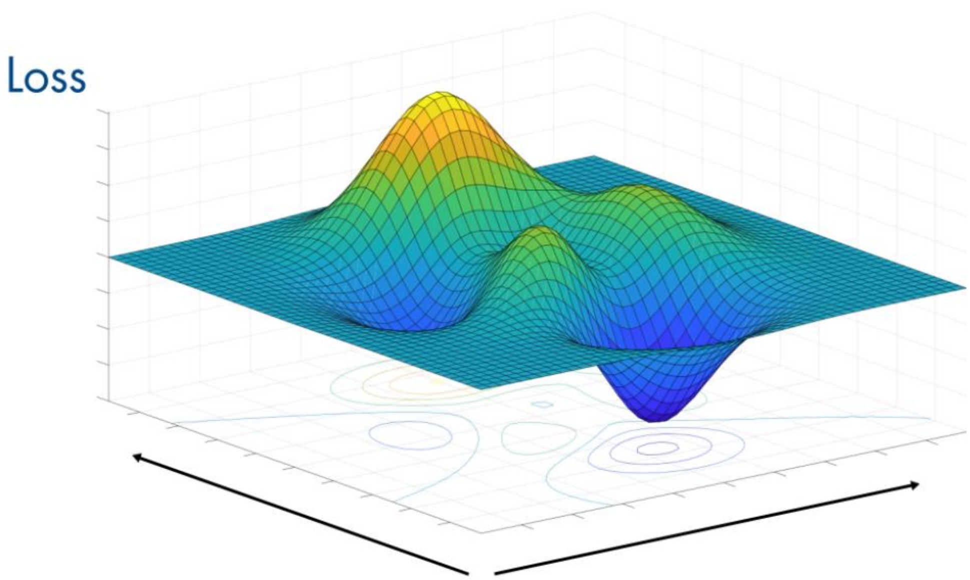


Figure 18: A general Function

The height of the ground is the value of the error of loss function. The gradient decent algorithm tries to figure out the direction that leads to the steepest point by taking the value of the step size as shown in Figure 19. However as explained earlier, if the step size is too small, the training process becomes slow and creates a large step size, resulting in overshoot.

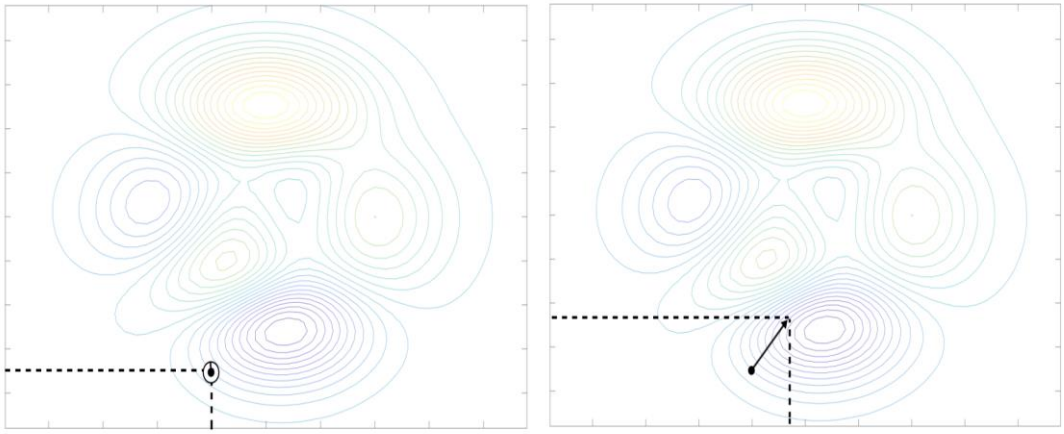


Figure 19: The Step Size of The Training Algorithm

To overcome the issue of step size, a momentum term is used. Gradient decent with momentum tries to stop the jumping around, without having to take very small steps. For instance, after one step down toward the direction of the gradient, the next direction does not change completely to the new one, but instead the algorithm turns toward that direction. The next step is, therefore, in a direction that is a weighted average of the previous direction and the new direction. This heightening is set by the momentum option. Figure 20 depicts the movement of the momentum term.

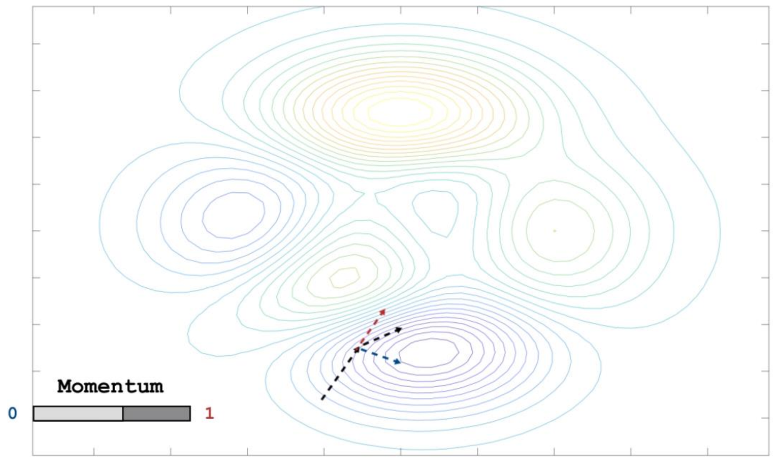


Figure 20: Momentum Term

4.4 CNNs and Graphic Processing Unit (GPUs)

A graphic processing unit (GPU) can be used to significantly speed up the many computations required in deep learning. These computations include processing input images, training the deep learning model, storing the trained deep learning model, and deployment of the model. Yet why is GPU required in deep learning networks such as CNN? During the training phase of a neural network, two operations are performed: forward pass and backward pass. In forward pass operation, input is passed through the network, then output is obtained, while in backward pass, the weights are updated many times based on the error received in the output. In both operations, the dot product is applied. Therefore, when a deep learning model is used, the GPU is highly recommended.

4.5 Alex-Net

The pre-trained network, Alex-Net, is an example of a convolutional neural network that was designed for image classification tasks, as shown in Figure 21, [29]. The network was made up of five convolutional layers, max-pooling layers, dropout layers, and three fully connected layers. The pre-trained Alex-net, which consists of 25 layers, has 12 million input images and uses around 60 million parameters, needed a whole week to classify 1,000 possible categories during the training and testing phase. The modified Alex-Net in this thesis needed almost five hours to train the network using MATLAB library.

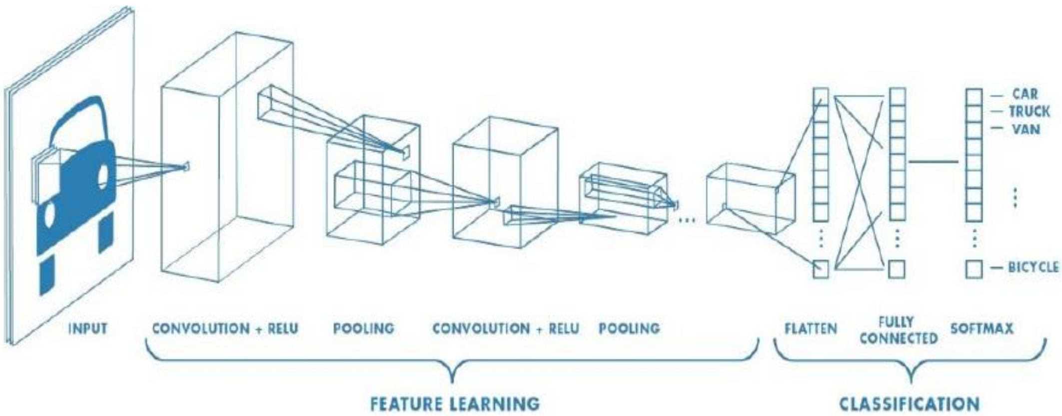


Figure 21: Example of CNN of Alex-Net

4.6 Transfer Learning

Rather than start over from scratch and build a new architecture for a neural network, Alex-Net can be modified to suit any problem and train it on different categories. The process of modifying Alex-Net and retraining it on a new data is called transfer learning, and it is a very effective way of addressing many deep learning problems [30] [33]. Three steps are required to perform transfer learning: network to train (i.e. network layers), data to train, and setting a set of training options.

First, Alex-Net neural network is modified and used to train. Second, the datasets that are used for training in the proposed work are taken from the IMAGENET data store, where these datasets are divided into three groups in order to perform classification differently. The learning strategy in Alex-Net is supervised learning, which means input images are already labeled and the outputs are known. Training options involve applying an algorithm that iteratively improves the network's performance to correctly identify the training images. This algorithm can be fine-tuned with many parameters, such as how many training images to use at each step (i.e. batch size), the maximum number of iterations to take, the learning rate, the momentum, and others.

The first 22 layers are in Alex-Net are designed for features extractions – hence they are fixed – but the modification is implemented in the last three layers. The 23rd is a fully connected layer with 1,000 neurons. This layer takes the extracted features from the previous layers and maps them on the 1,000 output categories, yet the proposed network is modified to classify different numbers of categories [32]. The next layer, the soft max layer, turns the raw values for the 1,000 classes into normalized scores so that each value can be interpreted as the network's prediction of the probability that the image belongs to that class [31]. The last layer then takes these probabilities and returns the most likely category as the network's output.

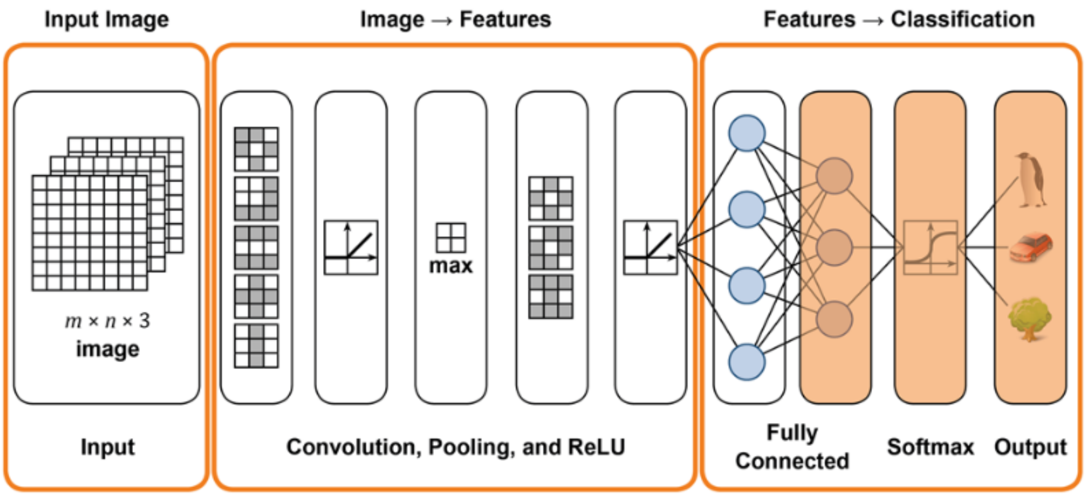


Figure 22: CNN of the Proposed Design

In the transfer learning technique, only the last three layers are modified, as appears in Figure 22. In this approach, the network has feature extraction behavior like the pre-trained network, but has not been trained to classify new categories.

Chapter 5

Numerical Result

Alex-Net is able to classify 1,000 categories; e.g. car, bird, tree. However, this CNN of Alex-Net is not able to classify subcategories of cars, birds, trees and so forth. In addition, there are many categories that Alex-Net has not been trained with, hence it cannot classify them. The first experimental results show how the transfer learning technique is able to classify 3,000 images into six subcategories of birds. The second experimental result presents classification of three subcategories of mammals, but with fewer images, only 200. The last experimental result shows the transfer learning technique to classify four different categories that have never been seen by Alex-Net.

5.1 Classifying Multiple Sub-Categories of Birds

The dataset had seven sub-categories of birds for classification. However, because some images of one sub-category of birds had different image artifacts that Alex-Net could not deal with, this sub-category was removed from the dataset. The numerical results of classifying 3,000 images into six sub-categories were implemented by applying transfer learning technique. Therefore, the proposed network architecture had six outputs.

It is unknown which ratio of training dataset would result in best performance. Hence, the number of training dataset changes each time. The ratio of training dataset ranges between 40 and 70 percent, and the remaining dataset is used for testing. Figures (23.a) to (23.d) show the performance evaluation of the neural network at classifying different sub-categories correctly during training phase. The performance improves from 94.56 to 95.56 percent as a larger ratio of data is used for training. However, a small dataset used for testing may not reflect the performance of the network accurately when it is used in real-time applications, even if the accuracy of the network is high in the training phase. Therefore, the more data used for training the and testing, the better results will be shown by the network in real applications.

There are two factors that must be considered during the network training: accuracy and loss. These two factors give an indication of the network's performance. The accuracy is the percentage of the network images that the network classifies correctly. This percentage should increase as the training proceeds. However, accuracy does not measure how confident the network is about each prediction. Differently, the loss is the measure of how far from the perfect prediction the network totaled over the set of training images. This measure should decrease toward zero as the training proceeds. These two factors in Figures (23.a) to (23.d) reflect how the predictions of the CNN is improving.

```
Initializing image normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:01 |       17.97% |       2.6374 |          0.0010 |
|       6 |          50 |       00:00:59 |      100.00% |       0.0244 |          0.0010 |
|      12 |         100 |       00:01:58 |       99.22% |       0.0204 |          0.0010 |
|      17 |         150 |       00:02:58 |      100.00% |       0.0139 |          0.0010 |
|      23 |         200 |       00:03:57 |       99.22% |       0.0216 |          0.0010 |
|      28 |         250 |       00:04:56 |      100.00% |       0.0029 |          0.0010 |
|      30 |         270 |       00:05:20 |      100.00% |       0.0010 |          0.0010 |
|========================================================================================|

nnz(testpreds == testImgs.Labels)/numel(testpreds) = 0.9456
```

(a) The ouput record of the network at 40 percent training data.

```
Initializing image normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:01 |       10.94% |       2.6382 |          0.0010 |
|       5 |          50 |       00:01:00 |       97.66% |       0.0593 |          0.0010 |
|      10 |         100 |       00:02:00 |       98.44% |       0.0280 |          0.0010 |
|      14 |         150 |       00:03:01 |      100.00% |       0.0151 |          0.0010 |
|      19 |         200 |       00:04:01 |      100.00% |       0.0030 |          0.0010 |
|      23 |         250 |       00:05:01 |      100.00% |       0.0031 |          0.0010 |
|      28 |         300 |       00:06:01 |      100.00% |       0.0076 |          0.0010 |
|      30 |         330 |       00:06:38 |      100.00% |       0.0073 |          0.0010 |
|========================================================================================|

nnz(testpreds == testImgs.Labels)/numel(testpreds) = 0.9500
```

(b) The ouput record of the network at 50 percent training data.

Initializing image normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Mini-batch Loss	Base Learning Rate
1	1	00:00:01	18.75%	2.5129	0.0010
4	50	00:00:59	96.09%	0.0926	0.0010
8	100	00:01:58	98.44%	0.0382	0.0010
11	150	00:02:58	99.22%	0.0236	0.0010
15	200	00:03:57	99.22%	0.0133	0.0010
18	250	00:04:56	100.00%	0.0039	0.0010
22	300	00:05:57	100.00%	0.0016	0.0010
25	350	00:06:57	100.00%	0.0076	0.0010
29	400	00:07:57	100.00%	0.0016	0.0010
30	420	00:08:21	100.00%	0.0011	0.0010

nnz(testpreds == testImgs.Labels)/numel(testpreds) = 0.9542

(c) The ouput record of the network at 60 percent training data.

Initializing image normalization.

Epoch	Iteration	Time Elapsed (hh:mm:ss)	Mini-batch Accuracy	Mini-batch Loss	Base Learning Rate
1	1	00:00:01	17.19%	2.3908	0.0010
4	50	00:00:59	93.75%	0.0998	0.0010
7	100	00:01:58	98.44%	0.0400	0.0010
10	150	00:02:58	99.22%	0.0190	0.0010
13	200	00:03:57	100.00%	0.0021	0.0010
16	250	00:04:56	100.00%	0.0051	0.0010
19	300	00:05:56	100.00%	0.0021	0.0010
22	350	00:06:55	100.00%	0.0024	0.0010
25	400	00:07:54	100.00%	0.0010	0.0010
29	450	00:08:54	100.00%	0.0060	0.0010
30	480	00:09:29	100.00%	0.0050	0.0010

nnz(testpreds == testImgs.Labels)/numel(testpreds) = 0.9556

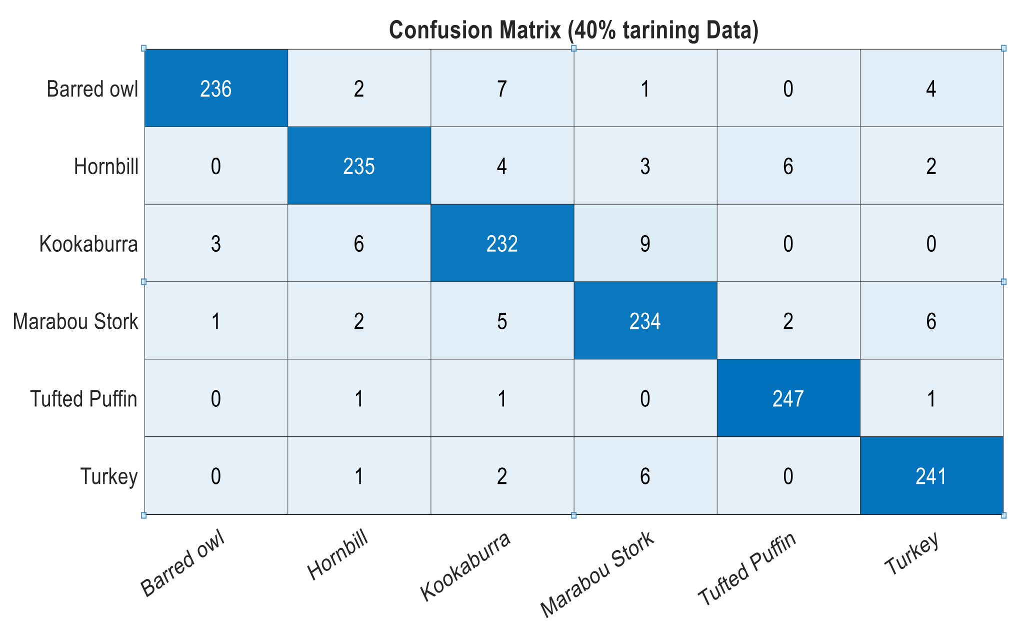
(d) The ouput record of the network at 70 percent training data.

Figure 23: The output records of the CNN at different fractions of training set

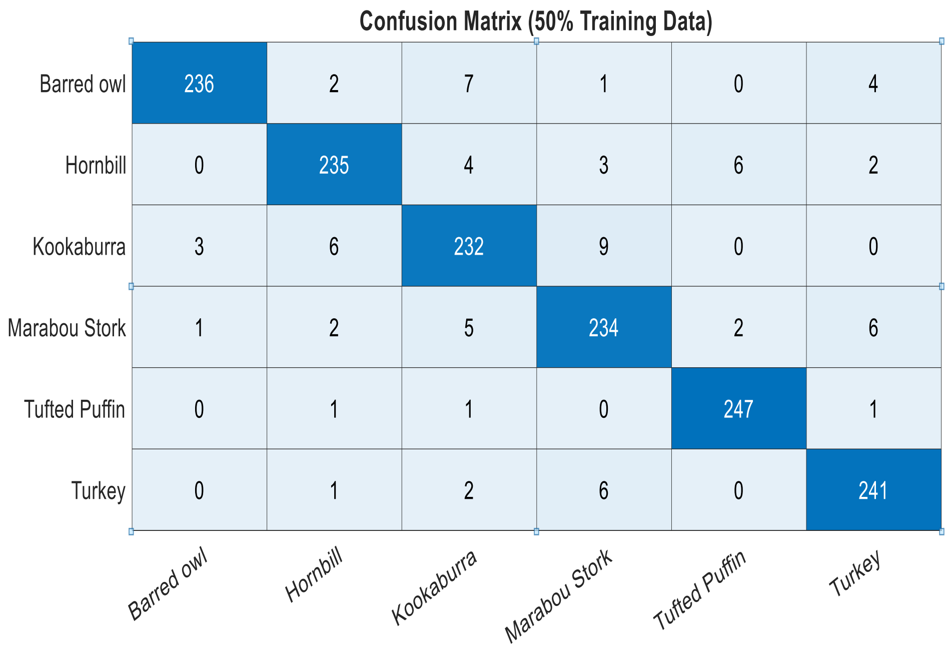
Figure 23 shows how, at each iteration, a subset of the training images, known as a mini-batch, is used to update the weights. Each iteration utilizes a different mini-batch. As the whole training set has been used, that is known as an epoch. The maximum number of epochs and the size of the mini-batches are parameters that can be set in the training algorithm options.

Confusion matrix is a measure of the network's confidence in predicting the outputs. All off-diagonal elements on the confusion matrix represent the misclassified data, while the diagonal elements represent outputs that the network correctly predicted. A good classifier will yield a confusion matrix that will look dominantly diagonal.

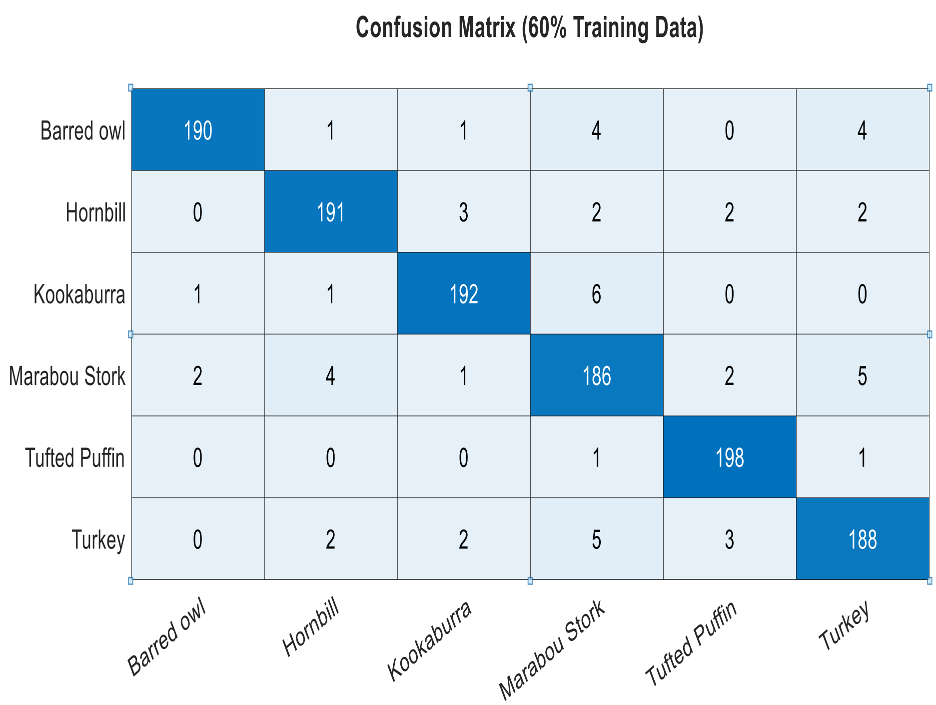
The following figures (24.a) to (24.d) show the network's prediction of six subcategories of birds in the testing phase. In Figure (24.a) and (2.b), the ratio of data that used for testing are 60 percent and 50 percent respectively; both experiments show the total accuracy of 93.5 percent. However, Figure (2.c) shows the result of 40 percent of testing data. The accuracy decreases to 62 percent. In Figure (24.d), the total accuracy rises to 95.5 percent with a ratio of 30 percent of testing data.



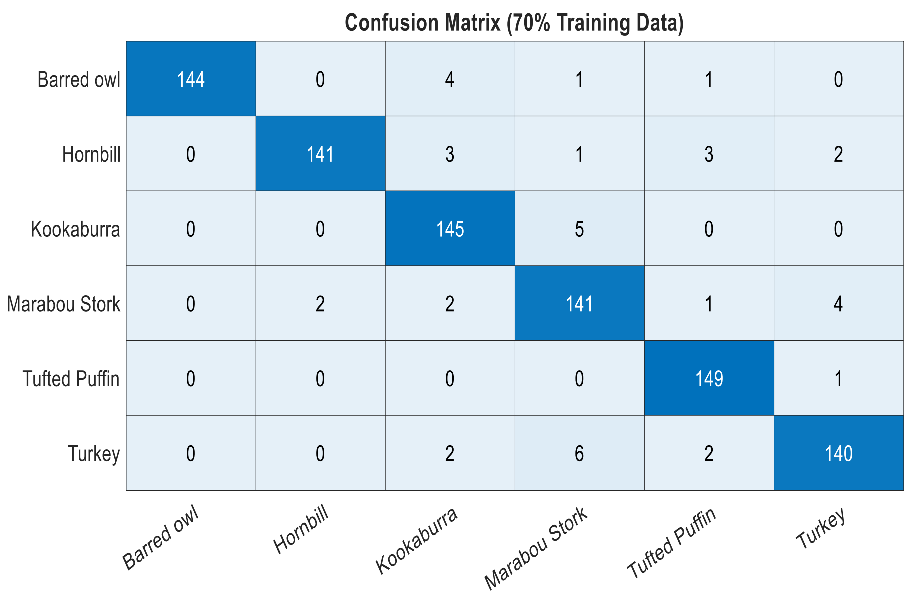
(a) Confusion Matrix of 40 percent training data



(b) Confusion Matrix of 50 percent training data



(c) Confusion Matrix of 60 percent training data



(d) Confusion Matrix of 70 percent training data

Figure 24: The classification results of the CNN for six sub-categories of birds

5.2 Classifying Multiple Sub-Categories of Mammals

The confusion matrix below shows a classification of three sub-categories of mammals in Figure 25. With a total of 600 images, the new network shows a very good performance, with total accuracy of 96.3 percent during the testing phase. Even in the training phase, the network shows almost the same accuracy.

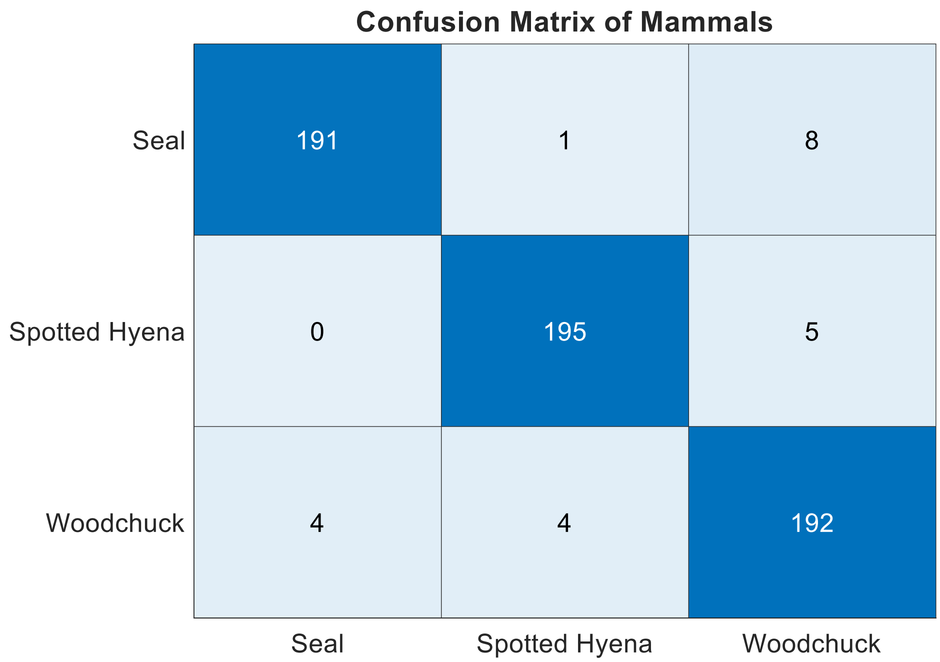


Figure 25: The classification results of the CNN for three sub-categories of mammals

```
Initializing image normalization.
|========================================================================================|
|  Epoch  |  Iteration  |  Time Elapsed  |  Mini-batch  |  Mini-batch  |  Base Learning  |
|         |             |   (hh:mm:ss)   |   Accuracy   |     Loss     |      Rate       |
|========================================================================================|
|       1 |           1 |       00:00:01 |       37.50% |       1.3915 |          0.0010 |
|       8 |          50 |       00:01:00 |      100.00% |       0.0111 |          0.0010 |
|      15 |         100 |       00:02:01 |      100.00% |       0.0012 |          0.0010 |
|      22 |         150 |       00:03:02 |      100.00% |       0.0008 |          0.0010 |
|      29 |         200 |       00:04:02 |      100.00% |       0.0013 |          0.0010 |
|      30 |         210 |       00:04:14 |      100.00% |       0.0007 |          0.0010 |
|========================================================================================|

nnz(testpreds == testImgs.Labels)/numel(testpreds) = 0.9633
```

5.3 Classifying Multiple New Categories

In the last experimental result, Figure 26, new categories are used for training the network using the transfer learning technique. These new categories have never been recognized by Alex-Net, but the network still shows an impressive result, which proves the ability of this technique to address such classification issues. Even though the dataset is small, the result shows high performance.

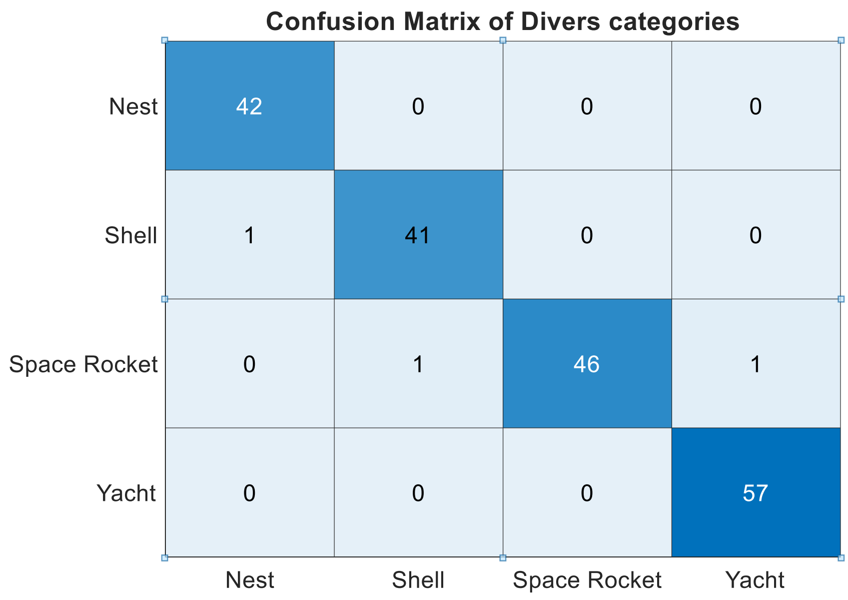


Figure 26: The classification results of the CNN for new sub-categories

Chapter 6

Conclusion

This report presented three main neural network architectures and the differences between each of them in terms of architecture, application, and method of network training and testing. However, the key objective of this project was to design a neural network that can perform image classification based on a convolutional neural network (CNN). To perform this, the pre-trained CNN, Alex-Net, was used by the MATLAB tool kit for saving time;

yet Alex-Net was subjected to some modifications to suit the problem presented in this project.

The modification over the pre-trained CNN was implemented by the transfer learning technique. This modified CNN managed to perform three different experiments of image classifications. First, 3,000 images were used for classifying six sub-categories of birds. Secondly, 600 images of mammals were classified into three sub-categories. Ultimately, four new categories of images were classified correctly. The used dataset was obtained from the ImageNet website, and had never been seen before by Alex-Net. The network's performance in the three experiments was satisfying and showed the benefits of the transfer learning technique for image classification.

References

1- A. J. Al-Mahasneh, S. G. Anavatti and M. A. Garratt, "The development of neural networks applications from perceptron to deep learning," *International Conference on Advanced Mechatronics, Intelligent Manufacture, and Industrial Automation (ICAMIMIA)*, Surabaya. Oct 2017.

2- D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., “Mastering the game of go with deep neural networks and tree search,” Nature, 2016.

3- A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” Nature, 2017.

4- T. Zhang, D. Liu, and D. Yue, “Rough neuron based rbf neural networks for short-term load forecasting,” in Energy Internet (ICEI), IEEE International Conference on. IEEE, 2017.

5- W. K. Mleczko, T. Kapu´sci´nski, and R. K. Nowicki, “Rough deep belief network-application to incomplete handwritten digits pattern classification,” in International Conference on Information and Software Technologies. Springer, 2015.

6- L. Liu, Z. Wang, and H. Zhang, “Neural-network-based robust optimal tracking control for mimo discrete-time systems with unknown uncertainty using adaptive critic design,” IEEE Transactions on Neural Networks and Learning Systems, 2017.

7- H. Jiang, H. Zhang, Y. Liu, and J. Han, “Neural-network-based control scheme for a class of nonlinear systems with actuator faults via data driven reinforcement learning method,” Neurocomputing, vol. 239, pp.1–8, 2017.

8- G. Ciaburro, " MATLAB for Machine Learning". Packet. Book. 2017

9- F. Piewak, T. Rehfeld, M. Weber and J. M. Zöllner, "Fully convolutional neural networks for dynamic object detection in grid maps," *2017 IEEE Intelligent Vehicles Symposium (IV)*, Los Angeles, CA, June 2017, pp. 392-398

10- K. Illanko," Neural Network and Deep Learning Fundamentals". Book. 2018.

11- David Kriesel, "*A Brief Introduction to Neural Networks"*, available at **http://www.dkriesel.com**. 2007. Online book.

12- M. N. Ahmed, "Deep Vision Pipeline for Self-Driving Cars Based on Machine Learning Methods". Ryerson University. Thesis of MASc. 2017.

13- S. Qin and X. Xue, "A Two-Layer Recurrent Neural Network for Nonsmooth Convex Optimization Problems," in *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 6, pp. 1149-1160, June 2015.

14- S.-S. Kim, “Time-delay recurrent neural network for temporal correlations and prediction,” Neurocomputing, vol. 20, nos. 1–3, pp. 253–263, 1998.

15- K. J. Lang, A. H. Waibel, and G. E. Hinton, “A time-delay neural network architecture for isolated word recognition,” Neural Netw., vol. 3, no. 1, pp. 23–43, 1990.

16- H. Ge, W. Du, F. Qian, and Y. Liang, “A novel time-delay recurrent neural network and application for identifying and controlling nonlinear systems,” in Proc. Nat. Comput., vol. 1. Haikou, China, 2007, pp. 44–48.

17- C. L. Giles, S. Lawrence, and A. C. Tsoi, “Noisy time series prediction using recurrent neural networks and grammatical inference,” Mach. Learn., vol. 44, nos. 1–2, pp. 161–183, 2001.

18- K. R. Ku-Mahamud, N. Zakaria, N. Katuk and M. Shbier, "Flood Pattern Detection Using Sliding Window Technique," *2009 Third Asia International Conference on Modelling & Simulation*, Bali, May 2009, pp. 45-50.

19- H. Izzeldin, V. S. Asirvadam and N. Saad, "Overview of data store management for sliding-window learning using MLP networks," *2012 4th International Conference on Intelligent and Advanced Systems (ICIAS2012)*, Kuala Lumpur, June 2012, pp. 56-59.

20-Q. Cao, B. Ewing, M. Thompson, "Forecasting wind speed with recurrent neural networks", Eur. J. Oper. Res. Aug. 2012.

21- A. Mazumder, A. Rakshit and D. N. Tibarewala, "A back-propagation through time based recurrent neural network approach for classification of cognitive EEG states," *2015 IEEE International Conference on Engineering and Technology (ICETECH)*, Coimbatore, Mar. 2015.

22- Y. Yoo and S. Y. Oh, "Fast training of convolutional neural network classifiers through extreme learning machines," *2016 International Joint Conference on Neural Networks (IJCNN)*, Vancouver, BC, July 2016.

23- A. T. Vo, H. S. Tran and T. H. Le, "Advertisement image classification using convolutional neural network," *2017 9th International Conference on Knowledge and Systems Engineering (KSE)*, Hue, Oct. 2017, pp. 197-202.

24- C. Tensmeyer and T. Martinez, "Analysis of Convolutional Neural Networks for Document Image Classification," *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, Kyoto, Nov. 2017, pp. 388-393.

25- Y. Seo and K. s. Shin, "Image classification of fine-grained fashion image based on style using pre-trained convolutional neural network," *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, Shanghai, March 2018, pp. 387-390.

26- K. Liu, G. Kang, N. Zhang and B. Hou, "Breast Cancer Classification Based on Fully-Connected Layer First Convolutional Neural Networks," in *IEEE Access*, vol. 6, pp. 23722-23732, Mar. 2018.

27- A. Saleh, M. Abdel-Nasser, Md. Kamal Sarker, V. Singh, S. Abdulwahab, N. Saffari, M. Garcia and D. Puig, "Deep Visual Embedding for Image Classification" in IEEE international conference on innovation and trends in computer engineering. Mar. 2018.

28- Y. Shima, Y. Nakashima and M. Yasuda, "Pattern augmentation for handwritten digit classification based on combination of pre-trained CNN and SVM," *2017 6th International Conference on Informatics, Electronics and Vision & 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT)*, Himeji, Sep. 2017.

29- Krizhevsky, A., Sutskever, I. & Hinton, G. E. ImageNet Classifcation with Deep Convolutional Neural Networks. Commun Acm 60, 84–90 (2017).

30- X. Li, T. Pang, B. Xiong, W. Liu, P. Liang and T. Wang, "Convolutional neural networks-based transfer learning for diabetic retinopathy fundus image classification," *2017 10th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, Shanghai, 2017, pp. 1-11.

31-P. Pimkote and T. Kangkachit, "Classification of alcohol brand logos using convolutional neural networks," *2018 International Conference on Digital Arts, Media and Technology (ICDAMT)*, Phayao, Feb. 2018, pp. 135-138.

32- H. Shin *et al*., "Deep Convolutional Neural Networks for Computer-Aided Detection: CNN Architectures, Dataset Characteristics and Transfer Learning," in *IEEE Transactions on Medical Imaging*, vol. 35, no. 5, pp. 1285-1298, May 2016.

33- Y. Zhu, Z. Fu and J. Fei, "An image augmentation method using convolutional network for thyroid nodule classification by transfer learning," *2017 3rd IEEE International Conference on Computer and Communications (ICCC)*, Chengdu, 2017, pp. 1819-1823.