Deep Learning Convolutional Neural Network Applications in Image Analysis to Identify Defects

BY

SRIJITH PARAKKAT UMAKANTH
B.Sc., University of Mumbai, 2015

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Master of Science in Mechanical Engineering
in the Graduate College of the
University of Illinois at Chicago, 2018

Chicago, Illinois

Defense Committee:

Dr. Sabri Cetin, Chair and Advisor
Dr. Arunkumar Subramanian
Dr. Jeremiah Abiade

I dedicate this thesis to my family, my friends and my mentor and advisor Prof. Sabri Cetin...

ACKNOWLEDGMENTS

The end of this thesis project marks the end of an amazing journey and at the same time the beginning of an even more exciting one. The period spent at UIC for my studies gave me the opportunity to know people from all over the world, who shared with me their own cultures, made me live their countries with their stories, and with whom I shared my culture and the love for my mother country India.

I'm indebted to many people for their continuing support leading to this dissertation. First, I would like to thank my adviser, Dr. Sabri Cetin, for his support and trust. His supportive guidance helped me navigate through problems and his trust gave me enough freedom to enjoy exploring this field. I would also like to thank him for the provision of necessary books, the drone to test my ideas and opening a new learning platform for me by making me his Teaching Assistant. Being his Teaching Assistant not only helped me financially but also helped me in refining my concepts even better by helping his fellow students with lab sessions. He is more than a mentor to me in every sense who has guided and motivated me in every step of my journey here at UIC as well as for this thesis project.

I would also like to thank my thesis committee, Dr. Arunkumar Subramanian and Dr. Jeremiah Abiade for reviewing the project and giving me their valuable insights. A special thanks to Syed Ammenuddin in guiding me throughout in writing this document and proof reading it.

I would also like to thank my good friend Charlie for encouraging me with food and coffee each time he saw that I was maxed out. Also my room mates Altamash and Ajinkya for helping me steer clear of all possible disturbances and better my concepts in certain areas of Statistical Learning which they are experts in. I also want to thank my near and dear Anju for bearing with my tantrums when

ACKNOWLEDGMENTS (Continued)

things do not work the way I planned and also calming me down with her sweet words and motivating me to achieve my goals.

Lastly, I would like to thank my parents Mrinalini (mom) and Umakanthan (dad) for their constant support and trust in me, helping me out financially when in need and praying for me to achieve my goals.

Srijith P Umakanth

May 2018

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CHAPTER **PAGE**

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF ABBREVIATIONS

USA	United States of America
UAV	Unmanned Aerial Vehicle
DC	Direct Current
GPS	Global Positioning System
USD	United States Dollar
B	Billion
K	Thousand
DIP	Digital Image Processing
CNN	Convolutional Neural Network
UIC	University of Illinois at Chicago
MLA	Machine Learning Algorithm
DNN	Deep Neural Network
IPT	Image Processing Technique
RGB	Red Green Blue
ANN	Artificial Neural Network
FC	Fully Connected Layer
ReLU	Rectifier Linear Unit

LIST OF ABBREVIATIONS (Continued)

SGD	Stochastic Gradient Descent
MSE	Mean Squared Error
IVSVRC	Image Net Large Scale Visual Recognition Challenge
VGG Net	Very Deep Convolutional Neural Network for Large Scale Image Classification
SGDM	Stochastic Gradient Descent with Momentum
GPU	Graphics Processing Unit
FPGA	Field-programmable Gate Array

SUMMARY

Safety inspection of concrete and steel structures should be carried out regularly because it is closely related with the structural health and reliability of that infrastructure. But then it is difficult to visually find cracks and other defects for extremely large structures because of time and cost constraints. Therefore, the development of smart inspection systems has been given utmost importance. A number of Image Processing Techniques (IPT's) have been implemented for detecting civil infrastructure defects to partially replace human-conducted onsite inspections. These techniques are primarily used to extract defect features like cracks on the concrete or steel structures. However, due to extensively varying real world conditions like change in lighting, shadows e.t.c, implementation of these primary techniques can be quite challenging. To over come such challenges, this thesis introduces the use of Convolutional Neural Networks (CNN's) for detecting such defects from images captures by drones. Thus making the entire inspection process smarter and more efficient as well as cost-friendly.

This thesis introduces a CNN (**Concrete_net**), designed using Transfer Learning technique, which is trained on 40000 images containing both concrete structures with crack and without cracks and tested against an unknown set of images which the model has not seen before, thus achieving an excellent overall accuracy.

CHAPTER 1

INTRODUCTION

Cities,states and metropolitan areas throughout any nation face a great challenge in economic, demographic, fiscal and environmental sectors that make it imperative for the public and private sectors to rethink the way they do business and contribute to the nations economy and livelihood of the people staying there. These forces are incredibly diverse, but they share an underlying need for modern, efficient and reliable infrastructure.

The essential building blocks of a nations economy in this 21st century are concrete, steel and fiber-optic cables. Infrastructure enables trade, creates job opportunities for communities and protects the nation from natural calamities like hurricanes, tornadoes, earthquakes etc. From investment in telecommunication systems, broadband networks, railroads, roadways, bridges, buildings and parks, infrastructure is the backbone of a healthy economy.

President Obama believes that America must build " 21^{st} century infrastructure - modern ports, stronger bridges, faster trains and the fastest internet". The American Society of Civil Engineers says that US needs massive investments in all essential infrastructure, from bridges, airports, railways and dams. According to the society's most recent infrastructure report card, the US earns a **D+** for its infrastructure. If it continues to the state of disrepair, the long-term hit to our economy could be catastrophic (Jordan,).

Despite the importance of infrastructure, the U.S. has not spent enough for decades to maintain and improve it. It accounts for about 2.5 % of the economy budget for infrastructure maintenance compared

to about 3.9 % spent in Canada, Australia and South Korea, 5 % for Europe and 9-12 % in China. In this thesis, the main focus is inspection of brigdes but the principles descibed could be extended to other infrastructures as well.

It is important to maintain a record of accurate knowledge of the condition of a structure and groups of structures on a timely basis so that maintenance, repairs and replacements if need be can be coordinated, planned for and implemented. Bridge management decides what repairs should be made to which structures of the bridge and when they should be repaired. Accurate and intimate knowledge of its conditions allows effective and timely maintenance, repair and replacement activities.

Monitoring existing deterioration of a structure to prepare a repair and prevent failure of a member is very important. Not all deterioration requires repairs. Often the recommendation for minor deterioration is to monitor and track the deterioration to see if it is worsening. While conditions that continue to deteriorate will need repair, conditions that have not shown additional deterioration might not need a repair, especially if they do not affect the structural integrity of the infrastructure.

My thesis aims to improve the capabilities of general purpose Unmanned Aerial Vehicles (UAV) or drones for specific applications, namely the bridge, road, airport runway inspection applications. Drone based inspection of these civil structures **increases safety** (eliminates the need for a human operator to inspect the bridge while hanging with ropes) **and reduce cost.**

The bridge inspection market in the US is estimated to be about $1.2B annually. The bridge collapse on highway I-35 in Minneapolis in 2007, which resulted in the death of 13 people, could have been avoided if there had been proper inspection of the bridge.

Current state of the art commercial drones are called ”flying cameras”. The main sensor they carry is a digital camera. They can be remotely controlled as well as have autonomous or semi-autonomous operation capabilities. There are two markets of Drones: commercial market and military market. Military market drones are about 1000 times more expensive than commercial drones. The cost of a commercial drone which can be bought over internet is in the order of a few thousand dollars. The cost of a military drone is in the order of a few million dollars. Commercial drones typically fly with four DC motor driven propellers, carry an articulated camera and perhaps a few other sensors, such as GPS. Commercial drone market has been growing in an exponential rate in the following markets: surveying of construction site, inspection of agricultural crop, inspection of roads, bridges and airport runways and airplanes, monitoring fire in forests and buildings, real-time imaging in security applications such as police and fireman support functions. The estimated commercial drone market in the world was valued at $2.1B in year 2016, and expected to grow to $11B USD by the year 2022.

1.1 Value proposition of Drones in Bridge Inspection

Drones as a new technology provide new inspection and monitoring capability that was not available before, and at a lower cost than doing the inspection/monitoring via manual means. For instance, it is expensive and dangerous to inspect bridges for defects by human inspectors using rope suspension. It is far safer (no human life is put in danger) and lower cost to do the inspection via a drone.

It is estimated that drone based inspection of bridges, homes after fire or hurricanes, costs about 1/3 of the manual human inspection. There are over 600,000 bridges in the USA, and 20% of them (120,000 bridges) need to be inspected annually. In other words, every bridge needs to be inspected at least once every five years.

Each inspection costs about $5K-$10K depending on the size and type of bridge. That represents bridge inspection annual market value of $600M - $1.2B. The estimated annual cost of bridge repairs is about x100 the cost of inspection; which is $120B to $150B range. Bridge inspection cost is about 1% of the repair and maintenance cost; inspection annual market size $1.2B, repair and maintenance annual market size $120B in the USA.

Thus drones are the most economic and safe way to carry out inspection of bridges. The multitude of different sensors especially a high resolution digital camera on the drone could provide a valuable asset for efficient and safe inspections of the infrastructure.

1.2 Convolutional Neural Network (CNN) for identifying defects from Images

Just as we human inspectors see a structural member of the bridge and determine where there are defects like cracks on the concrete structure, delamination of concrete that is a corrosion-induced deterioration of concrete bridge decks where deicing chemicals are used and even bending of a structural member, computers can also do the same thing when they are trained to detect such defects just with the raw images that we provide to it. These images could be automatically analyzed to detect defects for example a hairline crack on a concrete structure of the bridge and generate alerts when needed to the bridge inspection team during a routine inspection.

Several approaches have been suggested in Computer Vision to solve such a problem. Common of all methods involves extraction of features from images using various Image Processing techniques (Gonzalez and Woods, 2017) or Machine Learning. (Bishop, 2006).

“A feature is an individual measurable property that describes the raw image; it has to be informative, discriminative and independent from other features used to describe the same input.” (Gonzalez and

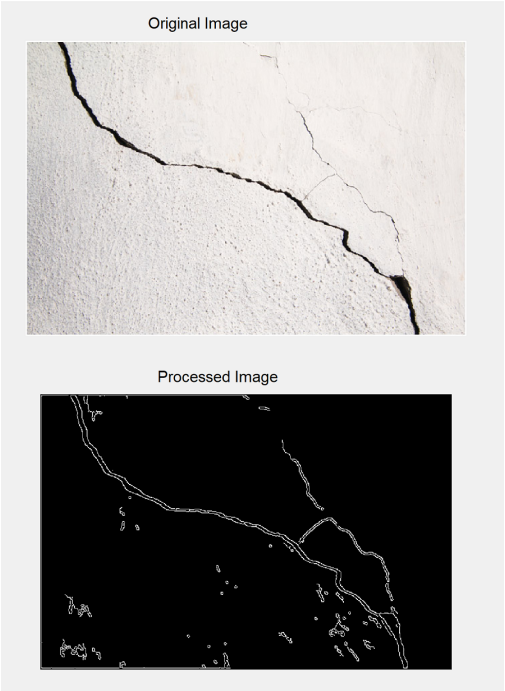
Woods, 2017). Examples of features used in defect detection of concrete, in images are based on finding strong change in brightness or intensities in image pixel values that may indicate a possible crack.

A number of vision based techniques for detecting damages primarily include Digital Image Processing (DIP) like edge detection using Sobel, Canny or Hessian-Matrix based edge detection just as shown in the figure 1. The only drawback of such techniques is that it does not work great for detecting possible cracks on steel structures because of their inherent rusty surface. "However such techniques is an ill-posed problem, as the results are substantially affected by the noises created, mainly from lighting and distortion and also with the number of parameters to take care of depends on such noisy environment image to image and no optimal solution exists." (Cha et al.,)

The manual feature extraction can be obviated by automated learning techniques. One possible solution for faster and most efficient solution to such a real-world situation is using, "Machine Learning Algorithms (MLA's) and Deep Neural Networks (DNNs). Convolutional Neural Networks (CNNs), in particular, allows automatic learning of features from raw images." (Lecun et al., 1998)

"DNNs are multi-layered feed-forward neural networks. They consists of many simple processing units called as artificial neurons, which are organized in layers. In each layer, neurons are connected to the neurons in the next layers through weighted connections and this cascade of multiple layers are used in automatic extraction of features from the raw images that becomes more and more complex with abstractions of multiple levels." (Alecci, 2017). The weights are the parameters used by the DNN model for the transformation of the input raw images into useful classification as the models output. The outputs from each of the intermediate layers can be considered as a recombination of the inputs

Figure 1: Example of using DIP to detect cracks on concrete surfaces



produced by the previous layer on the original input data. The parameters are learnt during the training period to make this recombination useful to finally predict the right output from the input images during the testing period. Hence, to extract useful information or features from complex images requires to have more number of layers, which in turn increases the corresponding number of parameters to be learnt during its training period.

In application that involves image analysis as the input to the DNNs, the most common type of DNNs used are CNNs, which are inspired by the visual system of animals. “CNNs are characterized by the presence of atleast one Convolution layer, where each neuron is connected only to a small region

called receptive field or a kernel also known as a filter, of the layer before. The filter is slided across the entire input of the layer and a convolution operation is called out between the input and the filter and its output is sent to the another neuron on the next layer." (Gonzalez and Woods, 2017). CNNs are different from the DNNs because they take into account the spatial nature of an image. CNNs also have a drawback, one of them being the number of parameters that needs to be adjusted during the training phase, optimally. A more detailed explanation along with the corresponding math is explained in Chapter 2 of this thesis.

For the capabilities just mentioned, CNNs are the best choice to solve the problem in this thesis. Because of the sheer number of parameters to be adjusted during the training phase, a clever yet more efficient technique called Transfer Learning is utilized to solve the problem. In Transfer Learning, pre-trained network is used and modifications are done to meet the objective.

1.3 Research Objective

As discussed before, there is very little ongoing research addressing automatic defect detection on infrastructure inspection employing CNNs. Our aim is to explore more possibilities along with a greater insight into different techniques like Transfer Learning, this research project address the following questions.

Is it possible to recognize defects like cracks on a concrete structure from an image using Convolutional Neural Network?

- Looking at the difficulty of the task and the lack of very large dataset, can Transfer Learning help in achieving great performance?

To answer this question, we define three objectives:

1. **Exploring the existing work on crack detection of concrete structures and also understanding Deep Learning and Transfer Learning techniques, we frame a solution for Image based defect detection on Concrete structures using CNN.**

   By understanding the nature of the problem and examining the work that has been already done, we use Deep Learning and Transfer Learning techniques to guides us through the tools that are available to achieve our research goals. They also help us understand the pros and cons of the tools involved. A literature study on the previous works, helps us understand the challenges of the task and also various solutions that were adopted. Chapter 2 describes the literature in fulfillment of this objective.

2. **Collecting a large dataset of labeled images representing a diverse vareity of cracks on the concrete surface as well as concrete surface without cracks.**

   A large dataset containing 20000 images of concrete structure with crack and 20000 images of concrete structure without crack is used for classification (Ennis,). Our aim is to use this dataset to train and evaluate the chosen CNN architecture. More details of the collection of dataset used in the thesis is presented in Chapter 4.

3. **Proposing architecture optimization of the CNN.**

   We aim to optimize our proposed CNN architecture for automatic crack classification, by tweaking the parameters during the training process and taking measures to avoid problems like overfitting. To fulfill this objective, we use different Transfer Learning techniques like re-training the

entire CNN architecture or just re-training a subset of the network layers. The proposed CNN architecture is described in Chapter 5.

1.4 Contributions

An optimized CNN architecture using Transfer Learning for automatic crack detection of concrete structures with raw images as input and an accurate and robust CNN with **99.71%** accuracy is achieved.

1.5 Thesis Outline

Rest of this thesis is organized as follows: In Chapter 2, the literature review is presented which includes exploration of Deep Learning field along with previous work done by other researches on this topic. In Chapter 3, research methodology used is described. The details on the creation of dataset are presented in Chapter 4. In Chapter 5, we present our CNN architecture optimization and the outcomes of the final product using all the Transfer Learning techniques. Lastly, in Chapter 6, conclusions are presented and discussed along with recommendations for future work.

CHAPTER 2

LITERATURE REVIEW AND RELATED WORK

This chapter presents the literature review on which this thesis is based on. In Section 2.1, we present an overview of various Image Processing Techniques (IPTs) used in Automatic crack detection. In Section 2.2, we discuss in depth about Convolutional Neural Networks. In Section 2.3, the specific CNN that will be used in this thesis is described and finally, at the end of this chapter, we conclude with a review of various transfer learning techniques.

2.1 Crack detection from Images using traditional Image Processing Techniques

Image Processing steps for identifying cracks on a concrete structure involves the following procedures:

1. Transformation of the acquired digital color image into a grayscale image.
2. Subtraction method for the images in uniform brightness.
3. Gaussian filtering for smoothening the images.
4. Binarization.

In the first step the true-colour RGB (Red Green and Blue) image is converted into gray level intensity image by getting rid of the hue and saturation information and retaining just the luminance (Hyeong-Gyeong and Jung-Hoon, 2010) This technique reduces the dimension of the image matrix from a 3-by-3 matrix to 2-by-2 matrix. For example, the RGB colour pixel having a depth of 24-bit $[(2^8)^3 = 16,777,216]$ color is converted to an 8-bit gray $[(2^8 = 256)]$ level.

In step 2, the gray level image obtained from step one is first smoothed using a median filter to remove any noisy concrete surface like dust or small perforations on the surface and then the smoothed image is subtracted from the original image to obtain the subtracted image.

The subtracted image is obtained by the following equation 2.1 (Hyeong-Gyeong and Jung-Hoon, 2010):

$$I_s(x_i) = max\begin{cases} median_{x_j \in R_j} I(x_j) - I(x_i) \\ 0 \end{cases} \tag{2.1}$$

where $I(x_i)$ means the intensity of the pixel x_i, and R_i means the neighborhood of the pixel x_i. When the result of the subtraction is a negative number, the result is represented as 0.

In step 3, the subtracted image is further smoothed using Gaussian low-pass filters. The aim of using this filter is to connect any small gaps in the now visible crack lines. It also helps in adjusting the distortion of the crack shape.

Finally in the step 4, various variable as well as global thresholding (Gonzalez and Woods, 2017) algorithms like Otsu's method are employed to binarize the images are highlight the cracks in the image.

These above mentioned steps requires proper selection of many parameters such as median filter size, threshold values, Gaussian filter size, its standard deviation and variance etc. When these parameters are optimally selected, the crack is effectively separated as a feature in the image and can be separated from the background of the image.

The major drawback of these techniques is that the parameters constantly change depending on the lighting conditions, orientation and noise in the images when it was clicked. Since, a large amount of images are generated as dataset to be inspected for defects like cracks, optimization of parameters for

that sheer number of images is not efficient and can also not guarantee to extract crack from the image. The best approach for a fast and efficient detection is using CNNs.

2.2 Convolutional Neural Networks

Classification of images is an important area of Computer Vision. It refers to the assignment of labels from a predefined collection of categories to the input image. This task is pretty challenging for computers because interpreting images from just their pixel intensity values is not a trivial task as the images are subjected to many variations. In the case of object recognition from images, the object can be subjected to different lighting conditions from different angles, so images from different angles of the same object may vary significantly in their pixel intensity values. Thus, a very sophisticated understanding of the input image data, just like we humans develop our knowledge from past experiences, will be necessary to solve such challenging tasks. Just by explicitly writing a computer program to achieve this non-trivial task is not possible, thus machine vision researches have taken techniques and ideas from machine learning, which enables computers to learn from experience just like we humans do, by the virtue of example and analogy automatically. (Gonzalez and Woods, 2017).

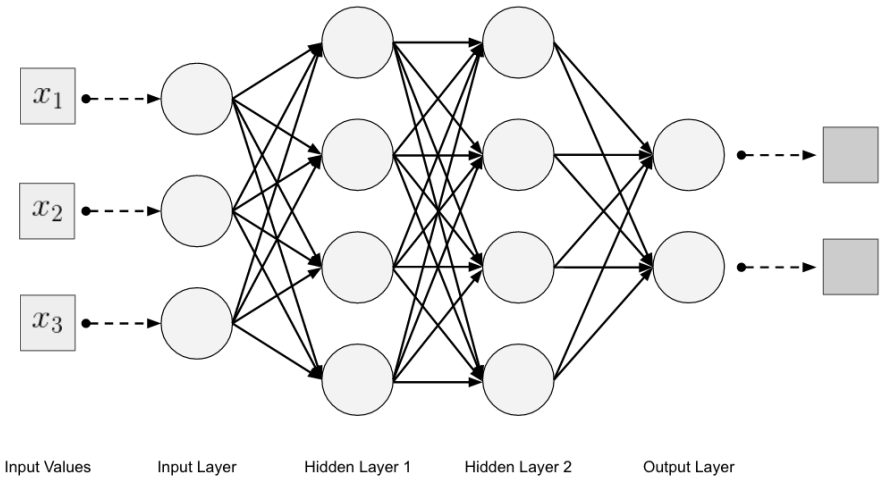
"In Supervised Learning, given an input image, such as x, we want to predict a label y, that aptly describes the input image (e.g. object depicted in the image), using a function $y = f(x, \theta)$. The model is not known at priori, hence our aim is to use a generic model described through a set of parameters given by θ, that will then be specialized on the target task" (Alecci, 2017). To achieve this, we apply a supervised learning method in which the model is presented with a set of examples with correct pairs of the input x and its corresponding label y and then update the parameters θ of the model so that the obtained output is as close as to the original associated labels with their corresponding inputs. To

evaluate the difference between the label $\hat{y}$ predicted by the model and the already known correct label y, a loss function similar to the one defined as follows $L(y - \hat{y})$ can be used. The goal is to choose parameters θ for the model that minimizes this loss function. There are many ways to achieve this criteria and they are known as optimization techniques. One example of many such techniques is the family of gradient descent algorithms. They are discussed in great detail in the following section of this thesis.

"The most popular approaches used in machine learning are Artificial Neural Networks (ANNs), which are primarily inspired by biological neural systems. An ANN consists of a number of simple and interconnected computing elements , called *artificial neurons*, that are connected with each other via links associated with a numerical weight. The weighted link between two neurons A and B of two adjacent layers expresses the strength of the influence of A on B. An artificial neuron is a mathematical function that operates a weighted sum on the inputs it receives to produce an output. This sum is then passed to a function called *activation function*, whose role is to introduce non-linearity in the model. Non-linearity is necessary to deal with real world problems that are usually non-linear, hence linear functions are often insufficient to model the real world problems." (Gonzalez and Woods, 2017)

Figure 2 shows a generic ANN architecture. It has one input layer, via which the input data x (typically the pixel intensity values of the input raw image in the form of an array) enters the model, one or more hidden layers (fig. 2 shows 2 hidden layers) and finally at the end, one output layer which outputs a prediction $\hat{y}$ of the original output y. The input propagates forward (left to right in the case of fig. 2), layer by layer through all of the hidden layers containing artificial neurons until it reaches the

Figure 2: General ANN architecture



output layer. Here in the fig. 2, two neurons are showed in the output layer, that means there are two labels to be predicted, like for example, whether the given input image is an apple or an orange.

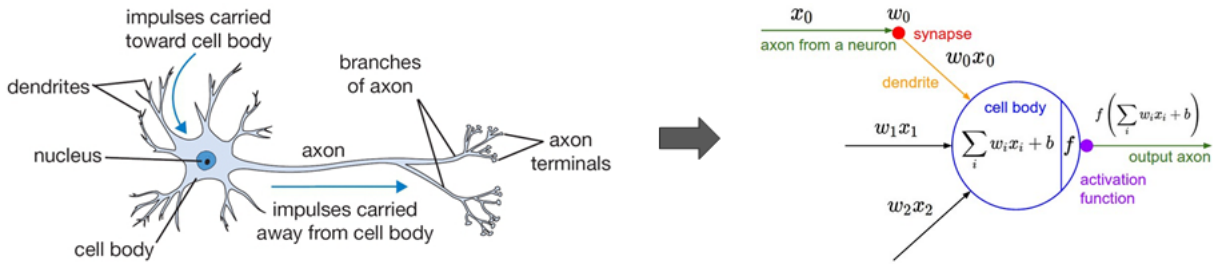
One of the most standard methods of training a neural network is called the back propagation algorithm. This algorithm along with an optimization technique like Stochastic Gradient Descent as described earlier, is used to compute the gradient of the loss function with respect to the network parameters θ. The algorithm works in the following way, starting from the output layer, as the name suggests the data propagates backwards to the first hidden layer and it simultaneously computes the gradients of each layer. These gradients are used as a parameter by the optimization scheme to update the network parameters θ in order to minimize the loss function as defined earlier.

"When there is more than one hidden layer within the network, the architecture is then termed as Deep Neural Network (DNN). Among several Deep Learning architectures, Convolutional Neural Networks are one of the most popular and most promising tools with respect to image classification and analysis." (Gonzalez and Woods, 2017)

"CNNs are a type of Deep Feed-forward Neural Networks specialized in visual recognition tasks and also widely used for natural language processing, video analysis, speech recognition and so on." (Lecun et al., 1998)

"Their architecture is inspired by the structure of the animal visual cortex as shown in the figure 3. In the visual cortex small regions of neurons are sensitive to specific regions of the visual field and respond to specific features such as edges." (Gonzalez and Woods, 2017)

Figure 3: Analogy of a neuron of animal cortex to an artificial neuron



Convolutional Neural Network as the name suggests, performs a convolution operation in the convolutional layer. This operation acts like a filter to detect features or patterns like edges and blobs present in the input data. Instead of assigning parameters manually to filter out the features from the input image as done in traditional digital image processing, in the case of CNN these parameters are learnt based on the training data we provide to the CNN model during the training phase.

"It has been shown that the lower layers in a CNN are able to detect features that are usually general for each image recognition task such as edges and curves. As the number of layers increases in the network, it gets specialized in detecting features that are more abstract and related to the specific target task." (Yosinski et al., 2014)

CNNs have one key characteristic called sparse interaction. This property talks about the interactions, which are indirect, a CNN could have with its deeper layers of neurons and the large portion of the input image, which could describe a complex relationship between many variables even though they are not directly connected with each other. CNNs have another great characteristic which makes it more efficient than other types of neural networks called parameter sharing. This ability of reducing the number of parameters to be learnt, is useful when a feature used to compute the network parameters in some spatial position of an image can also be useful in some other spatial location if the same feature exists instead of learning them again.

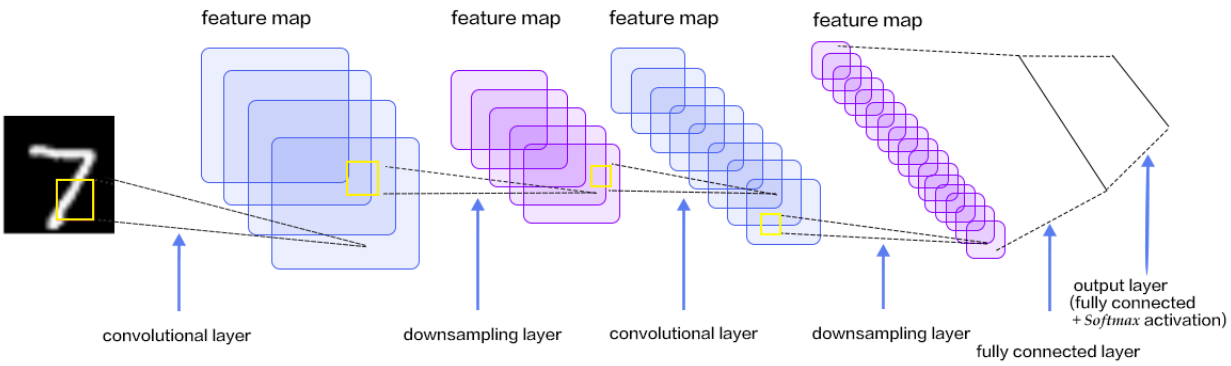
2.2.1 CNN Architecture

A CNN consists of layers which contains artificial neurons which are arranged in three dimensions along the width, height and the depth. In classification of images, the CNN can directly take raw images as its input. For example in a RGB image, the input image resembles a three dimensional matrix of pixel

intensity values which contains the height, width of the image, and depth of the matrix are the three Red, Green and Blue channels. Passing this as the input through the CNN network layers, they undergo transformations and then the output from the CNN is a 1-D vector with its dimension corresponding to the number of classes or categories to be classified.

Mainly, the three main layers arranged in sequence are the building blocks of a CNN architecture. They are, the convolutional layer, a pooling layer or also known as the down sampling layer and the fully connected layer. Figure 4 shows an example of the same. The next few sections of this chapter discusses more about the functions of each building block.

Figure 4: Example of a CNN architecture



Convolutional Layer

This layer is the main building block in CNN. The parameters of this layer are learned during the training process so they are also known as learnable filters. The dimensions of this filter are spatially smaller than the dimension of the raw input image. The filter size, i.e the width and the height is a design choice but the depth is fixed as it is corresponding to the number of the input channels, which is basically 3 for a RGB image or 0 for a grayscale image.

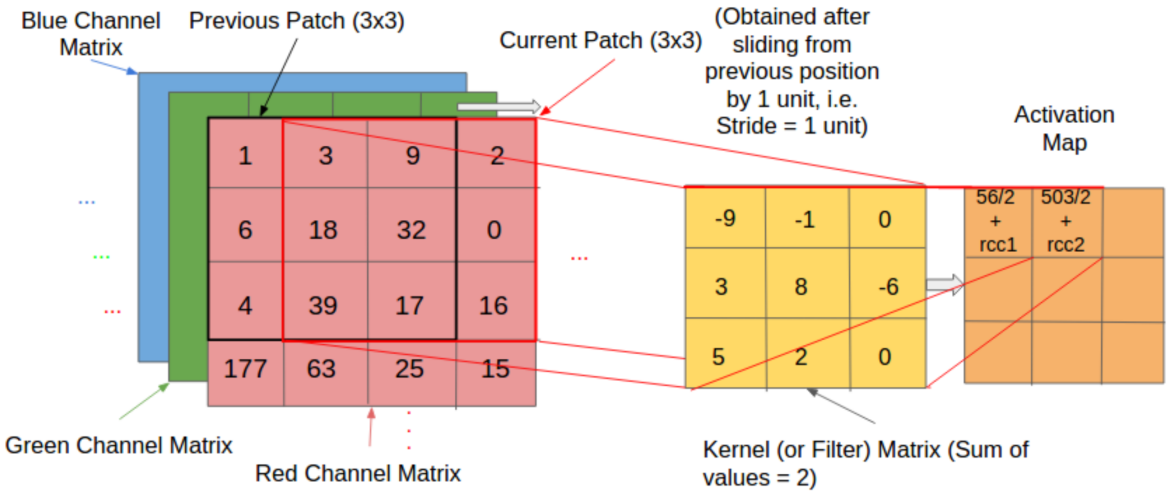
As shown in the figure 5, during the forward pass, the filters is slided across width and height of the input image matrix and a convolution operation is performed. Stride is a parameter which defines the value with which the filter slides across the input image. The operation of sliding the filters is mathematically translated as a dot product of the filter and the input image which is basically known as convolution operation. The result is a 2D output called as a feature map, which are stacked together along with other such feature maps in the depth dimension to form the output from that particular layer. The output spatial size can also be controlled by using a method called zero-padding. It is basically a technique of adding zeros around the border of the input image as shown in the figure 7,

"The output of one of the layers employing convolution operation in forward pass, along with activation function is written down mathematically as shown in the equation 2.2,

$$y_i^l = s\left(\sum_{j=1}^{C^{l-1}} f_{i,j}^l * y_i^{l-1} + b^l\right) \tag{2.2}$$

where, y_i^l is the output of the i^{th} filter for the l^{th} convolutional layer, with a total of C number of filters, and $f_{i,j}^l$, is the i^{th} filter of the convolutional layer l, that is connected to the j^{th} feature map of the layer

Figure 5: Example of a Convolution operation on a RGB input image



$l-1$, b^l is the bias vector of the layer l and finally s is the activation function used."(Gonzalez and Woods, 2017)

During the backward propogation algorithm or backward pass (training phase), a similar convolution operation is performed but this time the filters are flipped spatially along both the axes (width and the height). The parameters $f^l_{i,j}$, are learned by the model and updated using a backpropogation algorithm and thus the model is specialized in solving similar tasks later.

Pooling or Down sampling Layer

A Convolutional layer is generally followed by a Pooling layer or also known as Down sampling layer. The function of this layer is to reduce the spatial size of the input. It then operates independently on every depth slice. This layer does not contain any parameters or weights to be updated. It only consists of filters that slide, with a preset stride value, across its input layer, which is the output produced by the

preceding convolutional layer (Goodfellow et al., 2016). There are different ways to achieve this down sampling process, the most commonly used are:

- **Max Pooling:** It takes the maximum value of the input image within the filter size as shown in the figure 6 (refer back 1). In sub-figures (i), (ii) and (iii) 3 colour channels as input are max pooled. The operation uses a stride value of [2, 2]. The dark and red boundary regions describe the window movement. Sub-figure (iv) shows the operation applied for a stride value of [1,1], resulting in a 3x3 matrix and here we observe overlap between regions.
- **Average Pooling:** It takes the average value of the input image within the filter size.

Figure 6: Example of Max Pooling

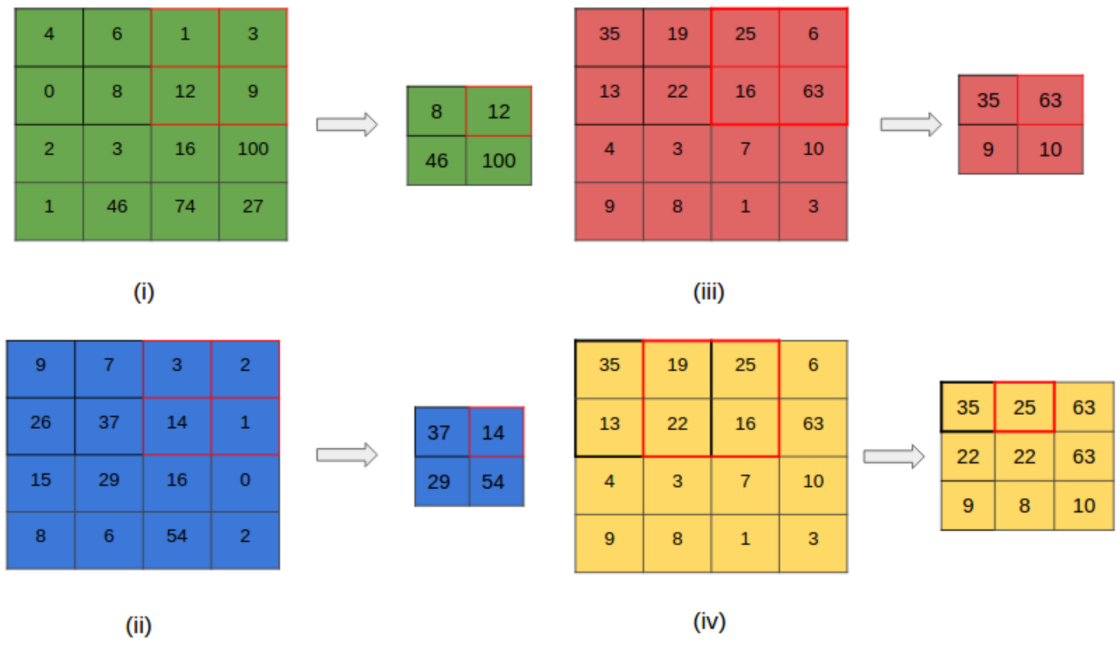
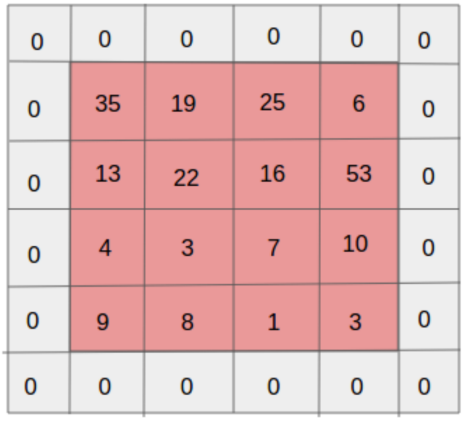


Figure 7: Example of a zero padding an input image



Fully Connected Layer

A minimum of one Fully Connected (FC) layer is present in CNN. It is the layer right before the output layer of the network. It converts the computed features by the previous layers into the respective class scores. Each neuron in this layer is connected to all the neurons in the previous layer. Its main function is to learn parameters i.e. weights and biases to map the input image into the correspondent output class.

"The output y^l for a fully connected layer l is mathematically represented as shown in the equation 2.3,

$$y^l = s\left(y^{l-1} * W^l + b^l\right) \tag{2.3}$$

where, W^l and b^l are the weights and the bias vectors of the layer l, and s is the activation function used." (Alecci, 2017)

Unlike the convolutional layers, the fully connected layer doesn't share the parameters, thus resulting in a substantial increase of the learnable parameters of the CNN.

Activation Function

Real world problems are highly non-linear in nature, thus in order to introduce non-linearity in the network we use activation functions. There many different activation functions like the Sigmoid, Hyberbolic tangent, out of which the most popular is the Rectifier Linear Unit (ReLU).

"ReLU, as shown in figure 8, is a function that thresholds the activation at zero. It is described by equation 2.4, where z is the input to an artificial neuron in the network.

$$s(z) = max(0, z) \tag{2.4}$$

ReLU being nearly linear allows to preserve the properties of a linear models that it easy to optimize through gradient-based algorithms. Furthermore, it is easy to implement and helps in greatly accelerating the convergence of the optimization problem." (Goodfellow et al., 2016)

Another popular activation function, is called the Softmax, as shown in the figure 9.

"This function is often used in the output layer of a neural network to model the probability distribution of the output over multiple classes. It is mathematically represented as in equation 2.5,

$$s(z)_j = \frac{e^{z_j}}{\sum_{K=1}^{K} e^{z_K}} \tag{2.5}$$

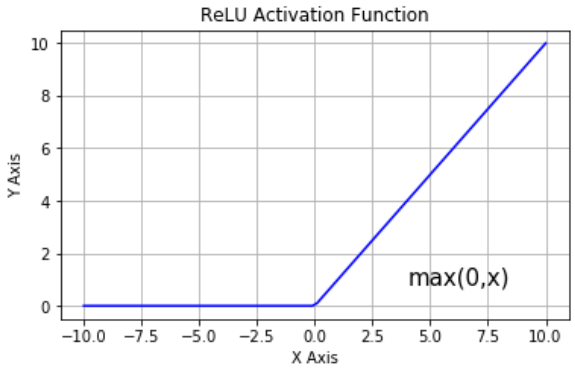


Figure 8: ReLU Activation Function

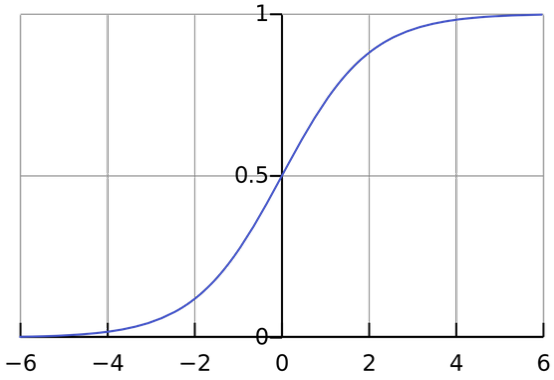


Figure 9: Softmax Activation Function

where, z is the input vector to the final output layer and $j = 1, 2, ..., K$, indexes the output artificial neurons, with K number of classes. The value associated to each output artificial neuron, corresponds to the probability that the input signal belongs to that class represented by it. The output values are bounded between 0 and 1 as shown in the figure 9. Thus, the sum of all the output values are constrained to 1."(Alecci, 2017)

The main advantage of bounding the sum of all the probabilities to 1 is that if the output value of one class increases, it makes the values of the other class decrease thus making it able to identify a unique output for a class and avoiding two or more neurons coming up with a same output value.

2.2.2 CNN Training

Training a CNN is not a trivial task since we are dealing with big network architectures and training datasets. Thus, we must introduce various important elements that will be relevant for the training

process of CNNs. Those important elements will help us greatly to understand the experiments better that we perform in this thesis.

Gradient Descent Optimization

The Gradient descent optimization method is one of the most commonly used algorithms during training phase to optimize neural networks. Its aim is to minimize the loss function that we have defined by finding the gradient and the in which the loss function decreases the fastest using the mathematical directional derivatives. It is used along with the back-propagation algorithm which helps to detemine the extent by which we need to update the network parameters i.e. weights at each iteration. There are many variants in the family of the gradient descent algorithms. Which one to choose is merely a design criteria based on the amount of computation power and the size of the training dataset used. Following are some of the variants of the gradient descent algorithms:

- **Batch Gradient Descent**: "In this method, during the training phase, the gradient of the loss function with respect to the network parameters θ for the entire training dataset are computed. The drawback of this method is that, since the parameters are updated just once after calculating the gradients for the whole dataset, if the training dataset is too large (which is usually the case), this method can be very slow and may require many iterations before converging. Equation 2.6 gives the mathematical expression according which parameters θ are update at each iteration.

$$\theta_t = \theta_{t-1} - \eta * \nabla_\theta L(\theta_{t-1}) \tag{2.6}$$

where, $\nabla_\theta L(\theta)$, is the gradient of the the loss function $L(\theta)$ with respect to the parameters θ (weights and parameters) and $\eta$is the learning rate, which determines the size of the steps to be taken to reach a (local) minimum of the loss function."(Alecci, 2017)

- **Stochastic Gradient Descent (SGD)**: "In this method, during the training phase, the gradient of the loss function with respect to the network parameters θ for each input-label pair, $x^{(i)} - y^{(i)}$ from the training dataset are computed, thus greatly improving the learning speed. However, it performs frequent updates of the parameters with a high variance that can cause the objective function as shown in the equation 2.7 to fluctuate heavily, which on one hand could help in escaping local minima but on the other hand could prevent the network from converging to the exact minimum since it will keep overshooting as shown in the figure 10,

$$\theta_t = \theta_{t-1} - \eta * \nabla_\theta L\left(\theta_{t-1}; x^{(i)}; y^{(i)}\right) \tag{2.7}$$

where, θ are the network parameters, $\nabla_\theta L(\theta)$ is the gradient of the loss function $L(\theta)$ with respect to the parameters, η is the learning rate." (Alecci, 2017)

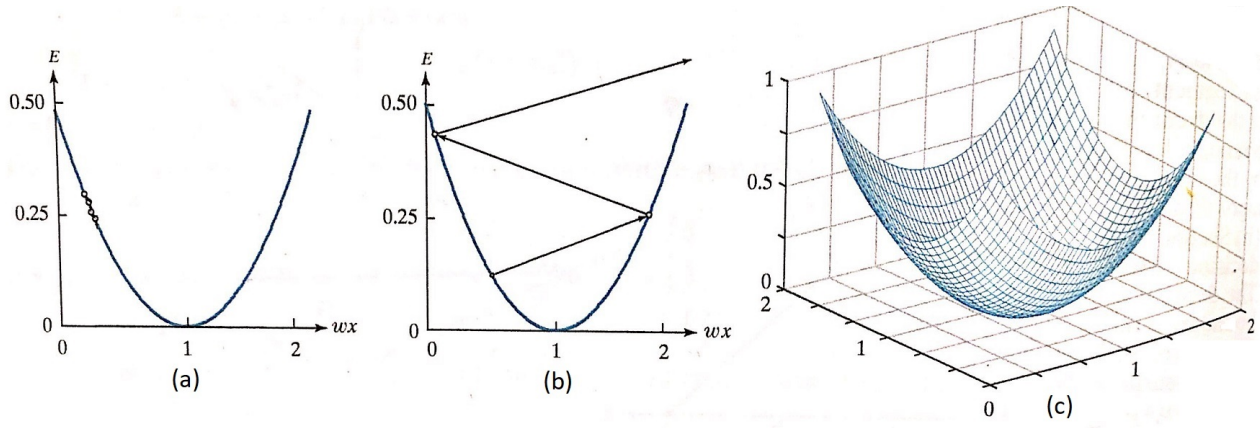
The loss function should ideally be differentiable and have a unique minimum. The function of choice for this purpose is a quadratic of the form as shown in the equation 2.8,

$$E(w) = \frac{1}{2}\left(r - w^T x\right)^2 \tag{2.8}$$

where, w is the augmented weight vectors, that minimizes the Mean Squared Error (MSE) between the desired (r) and the actual response ($w^T x$), where x is the input vector. The weight

vectors w and the desired response r are the part of the network parameters θ. In figure 10, sub-figure (a) shows the plot of E i.e the MSE as a function of wx, when $r = 1$ and the learning rate η is too small thus taking a longer time to converge to the minimum and sub-figure (b) shows the same plot when the learning rate η is too large thus large oscillations or divergence occurring. Sub-figure (c) shows the error function E in 3-D, for visualization.

Figure 10: Example of Stochastic Gradient Descent



- **Mini-batch Gradient Descent**: "In this method, during the training phase, the gradient of the loss function with respect to the network parameters θ for a subset of the training dataset including N patterns, $x^{i,i+N}; y^{i,i+N}$, called mini-batch are computed. This method compared to the SGD,

helps to reduce the variance of the parameter updates, leading to a more stable convergence. It is mathematically represented by the equation 2.9,

$$\theta_t = \theta_{t-1} - \eta * \nabla_\theta L\left(\theta_{t-1}; x^{(i,i+N)}; y^{(i,i+N)}\right) \tag{2.9}$$

where, $\nabla_\theta L(\theta)$, is the gradient of the the loss function $L(\theta)$ with respect to the parameters θ (weights and parameters) and η is the learning rate." (Alecci, 2017)

There are many such variations of the gradient descent algorithm. They differ the way network parameters θ are updated during the training phase, thus helping the entire training algorithm to be stable and converge faster.

- **Momentum**: " This is a method that helps accelerate the gradient descent algorithm to gain faster convergence and dampen oscillations. This can be necessary for the problems that these algorithms encounter when the surface of the loss function as shown in figure 10 (c), curves more steeply in one dimension than in another. The momentum term increases for dimensions whose gradients point in the same direction and then it reduces the updates for dimensions whose gradients change directions. Equations 2.10 and 2.11 give the mathematical expressions for Momentum update rule.

$$v_t = m * v_{t-1} - \eta * \nabla_\theta L(\theta_{t-1}) \tag{2.10}$$

$$\theta_t = \theta_{t-1} + v_t \tag{2.11}$$

where, v_t is the momentum variable at the time step t, initialized to zero at the beginning and $m \in [0,1]$ is the momentum coefficient, which is useful to dampen the velocity and oscillations in the system." (Haykin, 2009)

- **Nesterov Momentum**: "This is an improved version of the previously stated momentum method. It takes into account the next position of the parameters to calculate the gradient with respect to the approximate future position of the the parameters. Mathematically it is expressed as shown in the equations 2.12 and 2.13.

$$v_t = m * v_{t-1} - \eta * \nabla_\theta L(\theta_{t-1} + m * v_{t-1}) \tag{2.12}$$

$$\theta_t = \theta_{t-1} + v_t \tag{2.13}$$

where, v_t is the momentum variable at the time step t, initialized to zero at the beginning and $m \in [0,1]$ is the momentum coefficient." (Haykin, 2009)

Reduce Overfitting

"Overfitting is a serious issue in networks containing multiple hidden layers such as DNNs. It is defined as, the use of models or procedures that violate Occams razor, which means that any given complex function is a priori less probable than any given simple function, by using more adjustable parameters than are necessary or by using more complicated approaches than are necessary." (Hawkins, 2004)

"This happens mainly because of the complex relationship between the input and the output that the model would be able to learn by adjusting parameters. But that, with limited training dataset, compared to the number of network parameters, results to be an over specialized on the training data. Hence,

this relationship exists in the training data but not in the test ones, and the model loses the ability to generalize on previously unseen data." (James et al., 2013)

There are different ways to deal with such a problem. One example of it is called regularization. In this method, a penalty is introduced to the loss function. A penalty is usually a function containing the weights which penalizes the weights and helps in limit their growth to achieve a more stable model. There are many regularization techniques, some popular ones are:

- **L1 Regularization:** "This is a regularization technique that leads the weight vectors to become more sparse during optimization. In this way the network will prefer a sparse subset of its most important inputs becoming nearly invariant to noisy input. It is implemented as per equation 2.14

$$L = L + \lambda * |\theta| \tag{2.14}$$

where, $|\theta|$ is the sum of the absolute values of all the network weights present in the network and λ, is the regularization strength that decides the contribution of the term w in the loss/cost function as shown on the equation 2.8." (James et al., 2013)

- **L2 Regularization:** This regularization method is the most commonly applied. It penalizes the peaky weight vectors and giving preference only to the diffuse ones so to avoid the network to prefer some of its inputs over the others. Another advantage is that , by employing this method

during the gradient descent parameter updates, it leads every weight vectors to linearly decay to zero. It is mathematically represented as shown in the equation 2.15

$$L = L + \lambda * \theta^2 \tag{2.15}$$

where, θ^2, is the L2 norm of the parameter present in a network and λ, is the regularization strength that decides the contribution of w^2 in the loss/cost function as shown on the equation 2.8." (James et al., 2013)

- **Dropout:** "Although this technique is not a regularization method similar to the ones listed above but it acts as one. This method randomly drops artificial neurons and their connections, with a probability p, from the neural network during the training phase to prevent them from co-adapting too much, and thus encouraging each hidden artificial neuron to create good features without relying on the other ones and thus adjust its mistakes. Applying dropout is equivalent to training a different architecture at every training iteration and updating the network parameters based on the average computation of all these architectures. This results in learning a more robust model. During testing, all the network connections are restored and their weights are multiplied by the probability value p with which they were dropped off." (Srivastava et al., 2014)

- **Data Augmentation:** This method similar to Dropout method is not a regularization method per se, but is highly usesful when the training data is not sufficient to learn a generalizable model. The idea behind this technique is very simple, we can represent an image data in different orientation by rotating, distorting, mirroring, cropping or skewing it. This basically means we are making

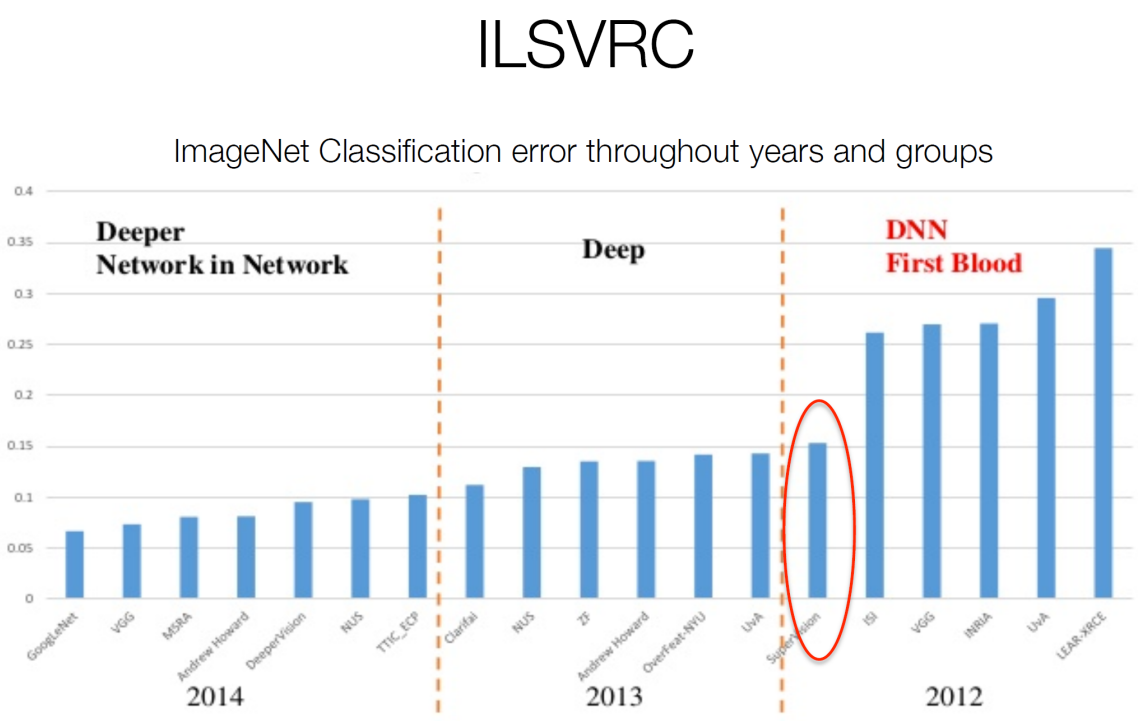
multiple copies of the same image but from different perspectives or visual angles. Figure 11 helps visualize the idea just explained.

Figure 11: Example of Data Augmentation

2.3 CNN Overview

"In the last few years CNNs have become more popular for solving complex Computer Vision problems, as it can be seen from the winner of the 2012 ImageNet Large Scale Visual Recognition Challenges (ILSVRCs). This challenge evaluates algorithms/architectures for object detection and image classification involving the recognition of 1000 classes." (Deng et al., 2009)

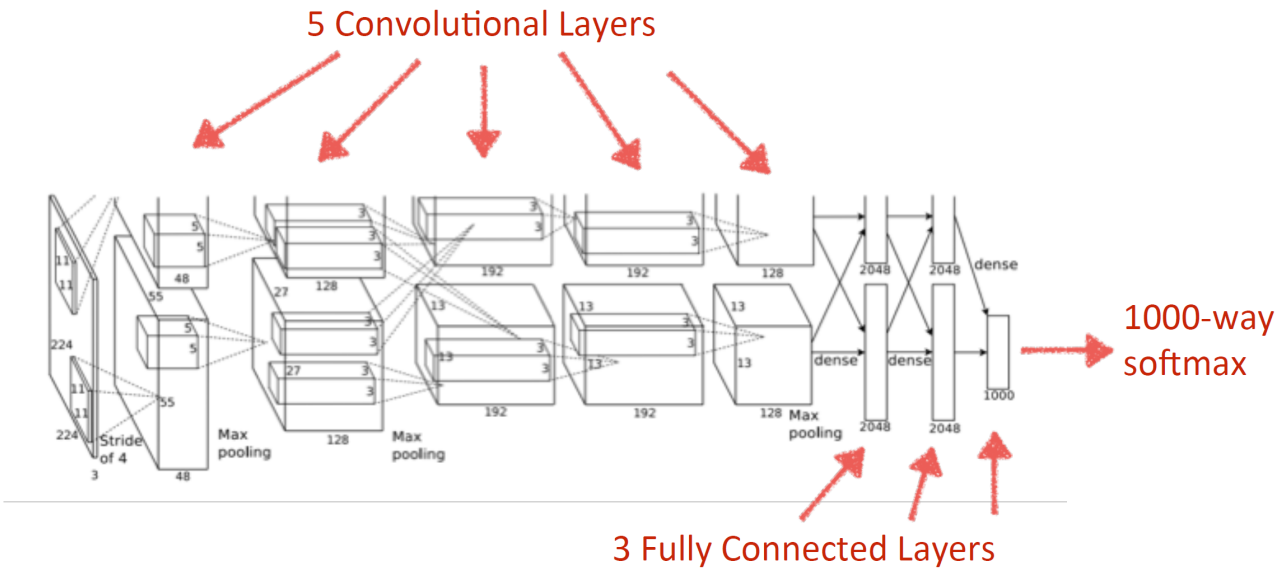
Figure 12: Imagenet classification errors throughout the years and groups.



”The first CNN winning the ILSVRCs, which also made CNNs very popular, was the **Alex-net architecture** in 2012.” (Krizhevsky et al., 2012)

As show in in the figure 12, Supervision was the group who designed Alex-net to participate in the competition and they were the one to achieve a substantial decrease in the error rate as compared to all of the other architectures or algorithms used before them. “Alex-net architecture is composed of 5 convolutional layers, max-pooling layers, dropout layers and 3 fully-connected layers as shown in figure 13 and employs ReLU as an activation function. It obtained a top-5 error rate of 15.6%, which is the error in classifying an image within the closest 5 classes.” (Alecci, 2017)

Figure 13: Alexnet Architecture



Alex-net was then improved by the authors of (Zeiler and Fergus, 2014) in the next year by tweaking its parameters and achieving a top -5 error rate of 11.25%.

"In 2014, VGGNet (Simonyan and Zisserman, 2014) even though it did not win the competition, showed that it was possible to reduce the number of parameters and at the same time to increase the depth of the network, achieving better performance than the architecture mentioned above with an error-rate of 7.3%. This architecture is composed by more convolutional layers than AlexNet, 13 exactly, which are smaller in terms of filters dimensions leading to a reduction of parameters but being able to learn more high-level features than previous CNN." (Alecci, 2017)

2.4 Transfer Learning

Transfer learning is based on the idea that the knowledge gained through an already trained neural network can be harnessed to improve the training process of a network to do a new, but similar task. We all know that training a CNN from scratch is computationally very heavy and also difficult because of the lack of large datasets. It is considered as an art to choose the right training parameters that allows the network to converge without causing it to overfit. To cope with this problem, transfer learning techniques comes to our rescue.

"It involves the concepts of a domain D and a task T. A domain $D = \{\chi, P(X)\}$ consists of a feature space χ and a marginal probability distribution $P(X)$, where $X = \{x_1, ..., x_n\} \in \chi$.

Given a domain, $D = \{\chi, P(X)\}$, a task $T = \{\gamma, f(\cdot)\}$, consists of a labels γ and an objective predictive function $f(\cdot)$ learned from the training data, consisting of pairs $\{x_i, y_i\}$, where $x_i \in X$ and $y_i \in \gamma$, which is used to predict the corresponding label $f(x)$ at a new instance of x, i.e. is the test data.

Now given a source domain/pre-trained CNN architecture like Alex-net D_s and a corresponding source task T_s, similarly a target domain D_t and a corresponding target task T_t, the transfer learning techniques aims to help improve the learning of the target predictive function $f_t(\cdot)$ in D_t using the knowledge in D_s and T_s, where $D_s \neq D_t$ and $T_s \neq T_t$." (Pan and Yang, 2010)

"A distinction can be made, based on the different situations concerning source and target domains and the corresponding tasks:

- **Inductive Transfer Learning:** Where source and target domain can either be the same or different & source and target task are different but related.
- **Transductive Transfer Learning:** Where source and target domain are different but related and source and target task are the same.
- **Unsupervised Transfer Learning:** Where source and target domain are different but related but source and target task are the same and no label data are available for the target task." (Alecci, 2017)

According to the above distinctions, our case can be associated with **Inductive Transfer Learning**, since the domain that we need and the source domain (Alex-net) are different but related as both are used for image classification and we have label for the target task which is our training dataset.

There are many ways in which the ideas of Transfer learning can be applied on a CNN (Yosinski et al., 2014). Since we have a CNN that was trained on a different task like the Image net classification previously, we can use one of the following two approaches:

- "**Fine-Tuning:** In this approach, all the network parameters are re-trained by back propagating the errors into the whole network.
- **Freezing Layers:** In this approach, most of the transferred feature layers are left unchanged during training on the new task. This is motivated by the fact that the first layers contain more generic feature extraction capabilities, common to many tasks, while the others become progressively more specific to the target dataset (Elhoseiny et al., 2015)." (Alecci, 2017)

Since we have a dataset that is different but of medium size as compared to the original one (ImageNet dataset used by Alex-net), it will be ideal to use the fine-tuning approach as it is expected that the lower-level features are going to be more relevant for the target dataset (Crack or no crack on the concrete surface).

2.5 Chapter Conclusion

In this chapter, we presented background information in the field of automatic crack detection on concrete structures and Convolutional Neural Network (CNN).

The literature review was useful for a better understanding of the challenges our target task involves, i.e. detecting defects like cracks, delamination etc on concrete structures from just images. The benefits from using CNNs out weighs the benefits using traditional image processing techniques to achieve the same target task.

Thus, based on the literature study on CNN, it was observed that they would be extremely successful in the detecting varieties of defects on any type of structures. Especially by using deeper architectures many complex problems can be solved and by having sufficiently large dataset for training the model,

the network can be made extremely robust and efficient in solving complex problems even on a real-time environment.

Thus, from all of the information and knowledge gathered, **Alex-net** is the preferred pre-trained model (domain source), detecting cracks on concrete structures is the preferred target task and inductive transfer learning with freezing layer approach is employed to re-train Alex-net to achieve the target task.

CHAPTER 3

METHODOLOGY

In the literature study presented in Chapter 2, we can observe that CNN approach is actually the state-of-art among other image recognition techniques. Among all the other types of CNN, Alex-net is the most desirable architecture for our thesis.

As discussed in Chapter 1, our goal is to come up with a system using the state-of-the-art CNN, that is able to classify images showing crack on concrete structures and others. We are interested in analyzing images captured by drones flying around the infrastructure., thus making the routine infrastructure inspection smarter.

To achieve such a task, we need to train our developed CNN network on the appropriate data collected. We will use Transfer Learning techniques on Alex-net architecture which was originally trained on the ImageNet dataset, and then fine-tune its parameters by training it on the dataset which is already labeled correct (training dataset). We then optimize this newly designed CNN model to improve its accuracy and avoid over-fitting.

Following are the steps followed to achieve the desired output:

1. **Step 1:** A large dataset containing 20000 images of concrete structure with crack and 20000 images of concrete structure without crack is used as a training dataset (Ennis,).

2. **Step 2:** The original Alex-net architecture was used to classify 1000 different things, in our case only two classification outputs are required (Crack or No Crack) , thus the original architecture is modified to predict only these two things.

3. **Step 3:** The modified architecture is then re-trained using the training dataset using the techniques of transfer learning as discussed in Chapter 2.

4. **Step 4:** The re-trained modified network is then tested against a known test set of images to predict the outputs and the accuracy is determined.

5. **Step 5:** The modified architecture is then optimized to increase accuracy and reduce overfitting using techniques as discussed in Chapter 2.

CHAPTER 4

DATASET

In Chapter 2, we have seen various applications dedicated to defect detection on infrastructures. In my thesis, drones are used as a medium to inspect the infrastructure and using a high resolution camera mounted on the drone, images are acquired of the infrastructure. These inspection images serves as a means for testing the network.

In order to train the CNN, we used a large dataset containing 20000 images of concrete structures with crack and 20000 images of concrete structures without crack (Ennis,).

4.1 Drones:

Drones are ”flying robots”. Current state of art in commercial drones are called ”flying cameras”. They can be remotely controlled as well as have autonomous or semi-autonomous operation capabilities.

There are two markets of Drones: Commercial market and Military market. Military market drones are about 1000 times more expensive than commercial drones. The cost of a commercial drones which can be bought over internet is in the order a few thousand dollars. The cost of a military drone is in the order of a few million dollars. The basic underlying technologies are same in both markets, the difference being in the performance. For example, commercial drones have a flight time of 30 minutes, where as a military drone can have a flight time of 30 hours. Similarly, payload capacities are different.

Figure 14: Example of Commercial drone (DJI Matrice 200)

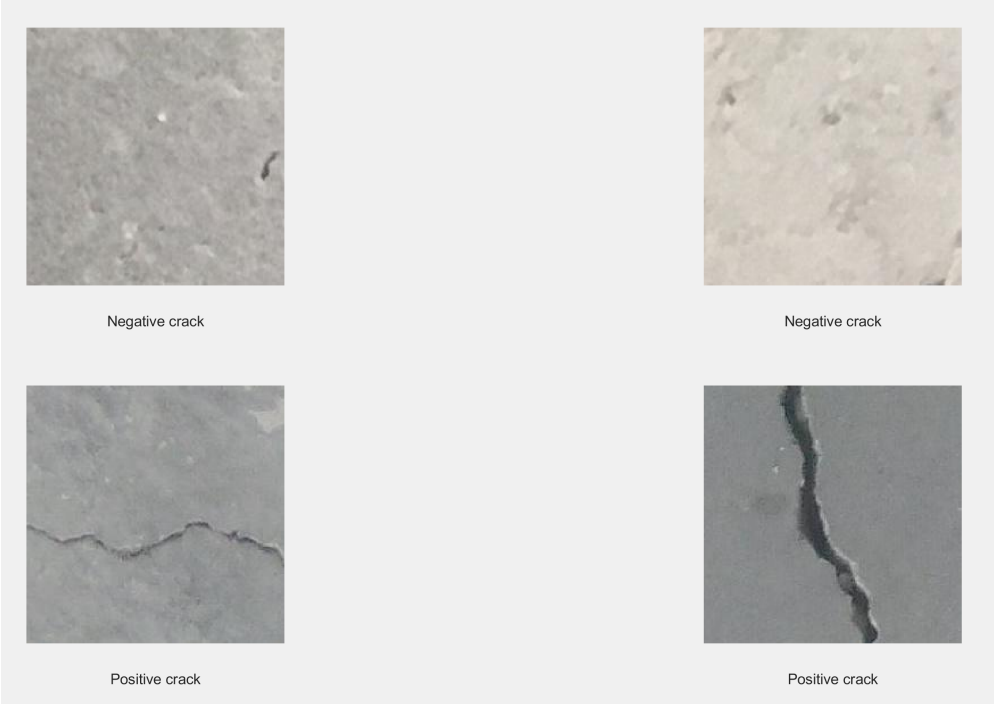
The focus of this work is commercial drones like as shown in the figure 14. As it has historically been in other fields of technology, the commercial drone market will undoubtedly benefit from military drone technology as that technology will increasing be made available for commercial markets.

4.2 Data:

The figure 15, shows the example of the training dataset used to train the CNN. The concrete surface with no cracks are marked as **Negative Crack** and concrete surface with cracks are marked as **Positive Crack**.

In order to test the network, a part of the 40000 images (40% of the training dataset called Validation Images) is used. 20% (8000 images with known true labels) of the total image dataset is used as test images and in addition to that, other test images which are directly acquired from the drones and smart

Figure 15: Example of Training images



phones as shown in fig 16 of real infrastructures are also used to further validate the network.

Next in this thesis in Chapter 5, we will see how the pre-trained network Alex-net was modified using transfer learning techniques to best suite our objective in this thesis.

Figure 16: Example of Test images

CHAPTER 5

TRANSFER LEARNING FOR CONCRETE CRACK DETECTION

In this chapter, we discuss the experiments done by applying Transfer Learning techniques on the CNN architecture, Alex-Net. Our goal is to design and optimized the CNN for detecting cracks and other defects in concrete structures using images captured from drones while doing infrastructure inspection as described in Chapter 4 applying Transfer Learning techniques.

As seen in Section 2.4, depending on the source and the target task and also taking into account the similarity of the source and the target datasets, there are different transfer learning techniques that are adopted. In the fine-tuning technique, re-training the CNN network layers that were previously trained on a source dataset, in our case ImageNet, can be attempted. We have a medium sized dataset available and the source and the target task could be considered different because our attempt is to perform classification of input images whose features to detect are different from the ones Alex-net was trained in. Moreover, re-training the lower layers of the original Alex-net would also be beneficial, since they are usually known for picking up general features like texture or shape when object recognition or other similar tasks are performed. Thus, in our specific case, fine-tuning approach of Transfer Learning is adopted.

In the next section, Section 5.1, more details about the experimental setup are discussed. Sections 5.1.2, the results obtained after applying the fine-tuning transfer learning technique on the Alex-Net architecture are reported.

5.1 Experimental Setup

For this thesis project we used a Alex-Net architecture implemented by Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton from University of Toronto as our pre-trained model.(Krizhevsky et al., 2012).

The parameters used for training the architecture are listed below:

- **Optimization Method:** Stochastic Gradient Descent algorithm with momentum.
- **Loss Function:** Cross Entropy function
- **Regularization Method:** L2 regularization.

The rate of learning is kept constant at 0.0001 along with the momentum variable 0.9 for the Stochastic Gradient Descent algorithm with momentum as discussed in Chapter 2. The mini-batch size is set to 256 and the model is trained for 10 epochs.

The network receives in input as an image sample with dimensions 227x227x3. These samples have been randomly shuffled previously and pre-processed to suit the input layer requirements as explained in the next subsection.

For the purpose of training the architecture, the entire dataset with proper labels as discussed in Chapter 4 was randomly shuffled and divided into Training Images containing 80% of the the total 40000 samples and Testing Images containing the remaining 20%. Out of the Training Images, 40% of those were used as Validation Images and the rest 60% of the training images were actually used for training.

While training, the network would update weights using the the mini-batch set using Stochastic Gradient Descent with Momentum (SGDM) algorithm and back propagation technique and then validate it twice every epoch to check the fit of the model. After every epoch, the validation set is randomly shuffled and then checked again twice to prevent any kind of over-fitting.

The training is continued and the validation accuracy is kept on check. A track of best validation accuracy so far during the training is kept and if the accuracy does not improve for at least 3 validations, then the training is stopped.

The trained model is then tested on the Test Images which was randomly separated at first (the samples that the model has not seen yet) and then the outputs are predicted. The accuracy of the model on the test data is calculated as the correct labels for the test data is already known to us, this helps in determining the correct accuracy of the trained model. A Confusion Matrix is plotted to visually described the results of the trained model on the test data.

After that the model is tested against some samples which was totally not available as part of the original dataset (images of real infrastructure cracks and without cracks are captured using drones and cell phones) and the correct labels are predicted using the model and then validated by us (humans).

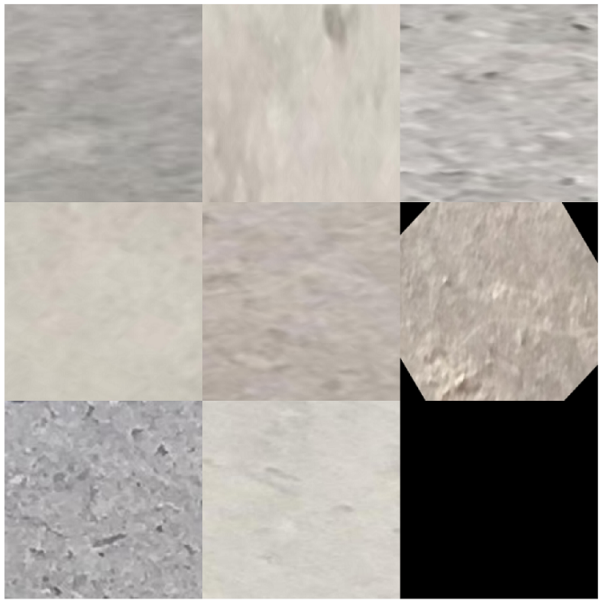
5.1.1 Image Preprocessing

In this subsection the preprocessing techniques applied on the input images are described.

In order to avoid over-fitting the model, different techniques can be followed and the most popular method during the training phase is called Data Augmentation as discussed in Chapter 2. My entire thesis is work in done using Mathworks's MATLAB and their Neural Network add-on toolbox package. Their package has extensive functions to achieve various tasks in developing, training, testing and vali-

dating the CNN.

Figure 17: Augmented Training data



- **Data Augmentation:**

  As you can see in the figure 17, using *'imageDataAugmenter'* function from Neural Network tool box of **MATLAB**, the training samples are randomly reflected along the X and the Y planes, randomly rotated complete 360° and randomly translated by plus and minus 5 pixel values along X and the Y axis. This is done so that the CNN does not learn the training data as it is causing it to over-fit and give really bad results for test data. By doing such data augmentation, we are training the model for all forms of cracks which can show up along any direction in the input image.

  The training as well the validation images are also resized to 227x227x3, size that is acceptable as input to our pre-trained CNN (**Concrete_net**) model.

5.1.2 Fine-tuning Approach

In this section, the actual experimental setup and the experiments performed following a fine-tuning approach and their corresponding results are presented.

5.1.2.1 Concrete_net Setup

Alex-net previously was used to classify 1000 different categories, but then Concrete_net which is modified version of Alex-net is used for classification of only two categories:

- **Positive** (crack is present)
- **Negative** (crack is not present)

Figure 18 lists down the number of layers of the original Alex-net CNN. It has five Convolutional Layers (Layer number: 2,6,10,12 and 14), two Fully Connected layers with 4096 artificial neurons in

Figure 18: Alexnet Layers

```
layers =

  25x1 Layer array with layers:

     1   'data'     Image Input                 227x227x3 images with 'zerocenter' normalization
     2   'conv1'    Convolution                 96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
     3   'relu1'    ReLU                        ReLU
     4   'norm1'    Cross Channel Normalization   cross channel normalization with 5 channels per element
     5   'pool1'    Max Pooling                 3x3 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv2'    Convolution                 256 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'    ReLU                        ReLU
     8   'norm2'    Cross Channel Normalization   cross channel normalization with 5 channels per element
     9   'pool2'    Max Pooling                 3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    10   'conv3'    Convolution                 384 3x3x256 convolutions with stride [1  1] and padding [1  1  1  1]
    11   'relu3'    ReLU                        ReLU
    12   'conv4'    Convolution                 384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    13   'relu4'    ReLU                        ReLU
    14   'conv5'    Convolution                 256 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    15   'relu5'    ReLU                        ReLU
    16   'pool5'    Max Pooling                 3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    17   'fc6'      Fully Connected             4096 fully connected layer
    18   'relu6'    ReLU                        ReLU
    19   'drop6'    Dropout                     50% dropout
    20   'fc7'      Fully Connected             4096 fully connected layer
    21   'relu7'    ReLU                        ReLU
    22   'drop7'    Dropout                     50% dropout
    23   'fc8'      Fully Connected             1000 fully connected layer
    24   'prob'     Softmax                     softmax
    25   'output'   Classification Output       crossentropyex with 'tench' and 999 other classes
```

them (Layer number: 17 and 20) and finally the last Fully Connected Layer with 1000 artificial neurons (Layer number: 23) followed by the Softmax layer for classification of 1000 different categories.

Concrete_net is created by modifying Layer number 23 and 25 to output only two categories. By using MATLAB Neural Network toolbox this can be easily achieved with a few lines of code. Figure 19 shows the Layers of Concrete_net CNN with the Layer 23 replaces with a Fully Connected layer containing just 2 artificial neuron followed by the Softmax layer for classification of two categories (Positive crack or Negative crack).

The Fully connected layer is configured with two different parameters namely *'WeightLearnRateFactor'* and *'BiasLearnRateFactor'*.

- **WeightLearnRateFactor:** Learning rate factor for the weights, specified as a nonnegative scalar. The software multiplies this factor by the global learning rate to determine the learning rate for the weights in the layer. For example, if *'WeightLearnRateFactor'* is 2, then the learning rate for the weights in the layer is twice the current global learning rate. The software determines the global learning rate based on the settings specified with the *'trainingOptions'* function.
- **BiasLearnRateFactor:** Learning rate factor for the biases, specified as a nonnegative scalar. The software multiplies this factor by the global learning rate to determine the learning rate for the biases in the layer. For example, if *'BiasLearnRateFactor'* is 2, then the learning rate for the biases in the layer is twice the current global learning rate. The software determines the global learning rate based on the settings specified with the *'trainingOptions'* function.

By introducting these two functions in the Fully connected layer, we are essentially increasing the learning rate of the modified layer of Alex-net so that the newly introduced layer can learn things faster than the previously learned layer of Alex-net making Concrete_net much more robust to the newly learned features which are introduced during the training phase as well as all the general features it has already learned with the Image Net dataset. This makes Concrete_net robust enough to read cracks on top of images of bridge pillars taken from a drone which may contain other surrounding features which it has learned from Image Net.

The training options are described in the next section of this chapter.

Figure 19: Concrete_net Layers

```
layers =

  25x1 Layer array with layers:

     1   'data'    Image Input                  227x227x3 images with 'zerocenter' normalization
     2   'conv1'   Convolution                  96 11x11x3 convolutions with stride [4  4] and padding [0  0  0  0]
     3   'relu1'   ReLU                         ReLU
     4   'norm1'   Cross Channel Normalization  cross channel normalization with 5 channels per element
     5   'pool1'   Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
     6   'conv2'   Convolution                  256 5x5x48 convolutions with stride [1  1] and padding [2  2  2  2]
     7   'relu2'   ReLU                         ReLU
     8   'norm2'   Cross Channel Normalization  cross channel normalization with 5 channels per element
     9   'pool2'   Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    10   'conv3'   Convolution                  384 3x3x256 convolutions with stride [1  1] and padding [1  1  1  1]
    11   'relu3'   ReLU                         ReLU
    12   'conv4'   Convolution                  384 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    13   'relu4'   ReLU                         ReLU
    14   'conv5'   Convolution                  256 3x3x192 convolutions with stride [1  1] and padding [1  1  1  1]
    15   'relu5'   ReLU                         ReLU
    16   'pool5'   Max Pooling                  3x3 max pooling with stride [2  2] and padding [0  0  0  0]
    17   'fc6'     Fully Connected              4096 fully connected layer
    18   'relu6'   ReLU                         ReLU
    19   'drop6'   Dropout                      50% dropout
    20   'fc7'     Fully Connected              4096 fully connected layer
    21   'relu7'   ReLU                         ReLU
    22   'drop7'   Dropout                      50% dropout
    23   ''        Fully Connected              2 fully connected layer
    24   'prob'    Softmax                      softmax
    25   ''        Classification Output        crossentropyex
```

5.1.2.2 Concrete_net Training Setup

As per the discussion in Chapter 2, Concrete_net is setup with different training parameters in the *'trainingOptions'* function on MATLAB. The following are the parameters used in training:

- **Solver Name:** *'sgdm'*, Stochastic Gradient Descent with Momentum and the default momentum value is 0.9
- **Mini Batch Size:** *'MiniBatchSize'*, is set to 256

- **Maximum Epochs:** *'MaxEpochs'*, is set to 10
- **Initial / Global learning Rate:** *'InitialLearnRate'*, this corresponds to the global learning rate mentioned in the previous section and is set to 0.0001
- **L2 Regularization Rate or Weight decay:** *'L2Regularization'*, is set to 0.0001 (default).
- **Verbose:** *'Verbose'*, Indicator to display training progress information in the command window of MATLAB. The displayed information includes the epoch number, iteration number, time elapsed, mini-batch loss, mini-batch accuracy, and base/global/initial learning rate. It also includes the validation loss and validation accuracy of the validation test samples which is also given as input while training to avoid over-fitting. In our case Verbose is set to *'true'*.
- **Validation Data:** *'ValidationData'*, is set to *'augValidationImgs'*, which is basically the same validation images which was set aside when the training samples were divided.
- **Validation Frequency:** *'ValidationFrequency'*, is calculated using the the total number of epochs and mini-batch size and set to twice every epoch. The entire validation data is then shuffled randomly every epoch re-validated twice every epoch while training.
- **Plotting:** *'Plots'*, is set to *'training-progress'*. When we specify *'training-progress'* as the *'Plots'* value in *'trainingOptions'* and start network training, *'trainNetwork'* creates a figure and displays training metrics at every iteration. Each iteration is an estimation of the gradient and an update of the network parameters.
- **Output Function:** *'OutputFcn'*, is set to *'@(info)stopIfAccuracyNotImproving(info,3))'*. This custom output function keep a track of the best 3 validation accuracies and if the validation

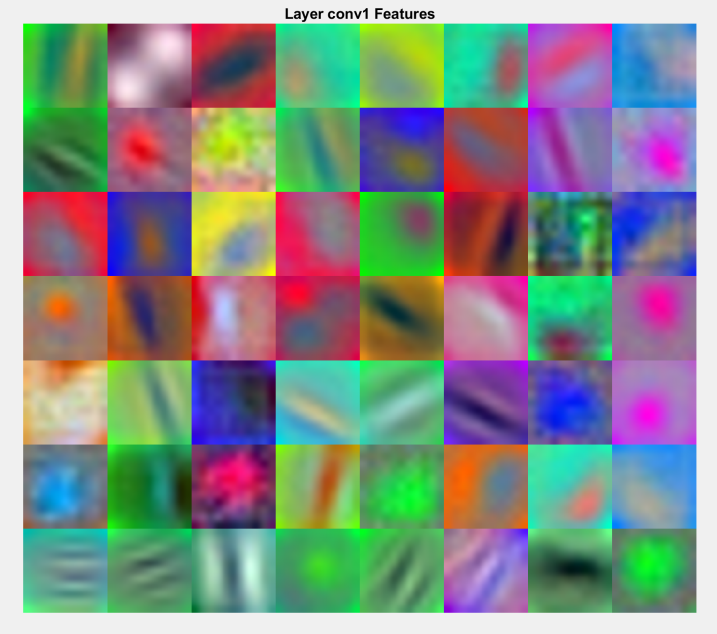
accuracy dos not improve for at least 3 validations, then the training automatically stops thus preventing any over-fitting.

5.1.2.3 Concrete_net Results

In this section we will discuss the results obtained after training Concrete_net.

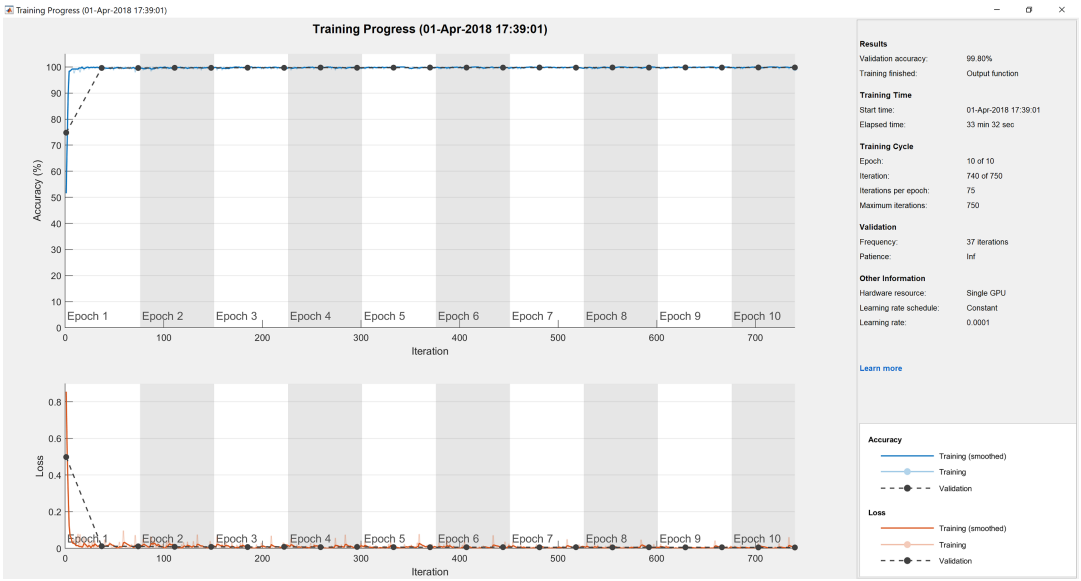
Figure 20 shows the features extracted by the first convolutional layer of Concrete_net. As we can see in the figure, it picked up all the important information like strong edges, sudden change in gradient of pixel intensities in the form of blobs and strips, all these features are strong indicators of a possible crack on the concrete surface.

Figure 20: Features extracted by the first convolutional layer of Concrete_net



The training was performed on a DELL Inspiron 7000 Gaming edition Laptop with NVIDIA GEFORCE GTX 1050Ti Graphics Processing Unit (GPU). Figure 21, shows the complete training progress. As dis-

Figure 21: Plot of the training progress using MATLAB



cussed in the previous section, the training stopped automatically following the output function when the validation accuracy did not improve.

Figure 21 also shows the plots of training accuracy in percentage versus the number of iterations performed (top plot) as well as the plot of a loss function determined by updating the weights using SGDM versus the number of iterations performed (bottom plot). The black plot shows the validation accuracy

and the corresponding validation loss versus the number of iterations performed. As you can see the validation was performed twice every epoch.

The training automatically stopped when the validation accuracy did not improve and the validation accuracy achieved was **99.80%**

Figure 22: Concrete_net tested against unknown Validation Images

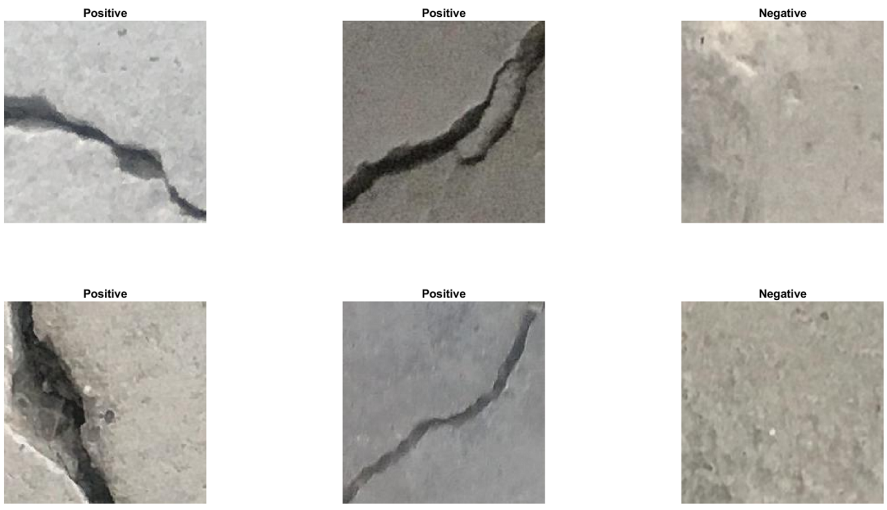
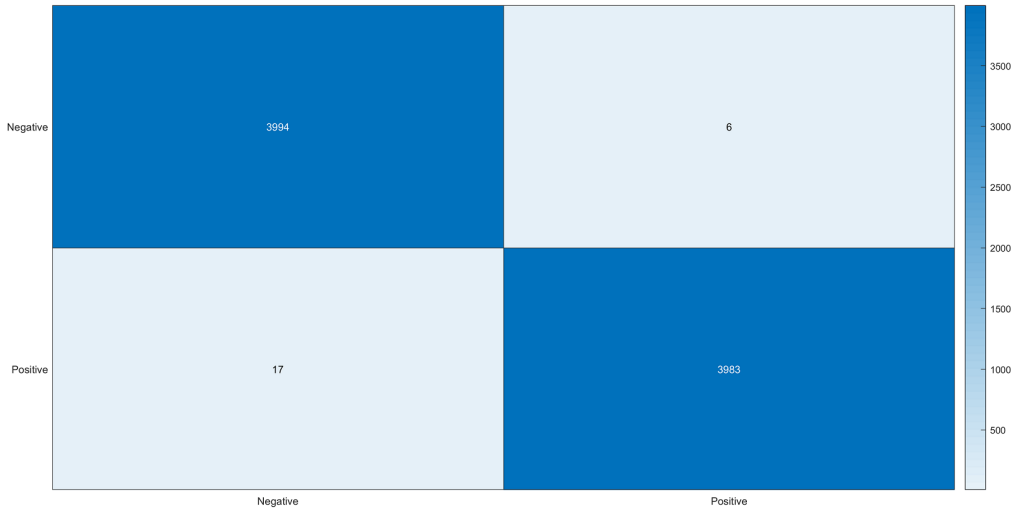


Figure 22, shows the results of Concrete_net tested again the Validation images which was set aside and were not used during the training process. These images were unknown to Concrete_net and it accurately identified the correct labels for these images indicating if there is a crack present shown as **Positive** or not shown as **Negative**.

After that, Concrete_net was tested against a set of test images which was previously set aside while partitioning the image dataset of which true labels were known as described in the section 4.2.
A confusion matrix as shown in the figure 23 was then plotted to visualize the results.

Figure 23: Confusion Matrix plotted for the results obtained on Test Images



As seen in the figure 23, out of 8000 test images, 17 images which had true labels as Negative were wrongly classified as Positive and 6 images which had Positive true labels were wrongly classified as Negative. Concrete_net thus achieved **99.71%** accuracy on those test images which it had never seen before.

The real test of Concrete_net lies in accurately identifying the cracks on real bridges and other scenarios with a great level of accuracy. Concrete_net was thus tested against a completely different set of images which it had never seen and was actually captured by us using drones and smart phones. Figure 16 shows some examples of such images. As shown in the figure 24, Concrete_net did a really good job in accurately predicting correct labels on those images which were visually verified by us humans.

Figure 24: Concrete_net tested against Real Test Images

CHAPTER 6

CONCLUSION AND FUTURE WORK

In this chapter we will summarize the work done, present some of the limitations and provide suggestions for future work.

6.0.1 Contributions

In this thesis project, our main goal was to explore the design of CNN, Alex-net specifically, employing the Fine-tuning approach of Transfer Learning technique in order to be able to detect cracks on the concrete surface just using images as input.

After performing an intensive architecture optimization, we proposed a modified architecture of Alex-net called Concrete_net.

Concrete_net has in total 25 layers out of which five are convolutional layers which does an excellent job of extracting information from images to determine if there is a crack or not, and two fully connected layers which has in total 8192 artificial neurons and also one fully connected layer with only 2 neurons right before the final softmax layer to classify if there is a crack or not.

After training Concrete_net on 40K images containing both crack and no-crack concrete surfaces, and doing validation on an unknown set of images which was shuffled randomly every epoch for 10 epochs, we achieved an accuracy of **99.71%** on test set of images which the model had never seen before.

Furthermore, Concrete_net was further tested against a set of images of real infrastructure with actual defects whose results are then verified by we humans. And it did a real good job on that.

6.0.2 Limitations

There are still many limitations to this work.

For example, the accuracy and robustness of Concrete_net could be further increased with more data of images taken during actual bridge or any other inspection and whose true labels are known. By training Concrete_net on such a data set, we can increase the generality of the information given to the model like surrounding vegetation, water mass or some other kind of terrain thus help in substantially increasing the robustness and accuracy of Concrete_net.

Another example would be the input size Concrete_net accepts. So every image sent to Concrete_net has to be resized to 227x227x3, which reduces the quality of the image thus reducing some of the details on the image. Measures can be taken to avoid this resizing but it increases the computation while training the model quite substantially.

Some of the ideas to improve on these limitations and further ideas to make Concrete_net an efficient and smart tool for automatic image processing in recognizing defects just from an image are discussed in the next section of this chapter.

6.0.3 Future Work

Based on the outcome of this project. we provide following recommendations for future work.

1. We recommend improving the content of the dataset used for training and validation. As discussed in the previous section of this chapter, if we have more information from the training images, like vegetation, water mass etc around the infrastructure which is being inspected and also the true

labels i.e whether the infrastructure on that image has a defect or no defect, then the robustness and the accuracy of Concrete_net could be substantially be increased.

2. We recommend changing the input size of the images being used for training, validation and testing from 227x227x3 to a higher resolution so that even more details are captured by Concrete_net during the training phase thus substantially improving the predictability of the model.

3. We recommend uploading the final trained Concrete_net CNN on to the drone itself. This can be achieved by doing a video feed capture from the drone's camera and processed in real-time using a Field-programmable Gate Array (FPGA) or a GPU directly mounted on the drone. The drone will then be instructed to click images only when it sees a defect on the infrastructure, thus making it a smart tool.

4. We recommend having dataset containing images of various other types of defects found on the infrastructure like leakage, delamination of concrete surface, corrosion of steel structures, leaching cracks e.t.c. Concrete_net can be easily modified to include more such categories of defects can be trained and fine-tuned to detect them thus adding more value to the entire inspection process of the infrastructure and saving a ton of money in that business.

CITED LITERATURE

[Alecci, 2017] Alecci, J.: Transfer learning for rain detection in images. Department of Computer Science, TUDelft University, 2017.

[Bishop, 2006] Bishop, C. M.: Pattern Recognition and Machine Learning. Springer, 2006.

[Cha et al. ,] Cha, Y., Choi, W., and Bykztrk, O.: Deep learning-based crack damage detection using convolutional neural networks. Computer-Aided Civil and Infrastructure Engineering, 32(5):361–378.

[Deng et al. , 2009] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255, June 2009.

[Elhoseiny et al. , 2015] Elhoseiny, M., Huang, S., and Elgammal, A.: Weather classification with deep convolutional neural networks. In 2015 IEEE International Conference on Image Processing (ICIP), pages 3349–3353, Sept 2015.

[Ennis,] Ennis, B.: Workhardeningtadp_raw data.

[Gonzalez and Woods, 2017] Gonzalez, R. C. and Woods, R. E.: Digital Image Processing. Pearson, 2017.

[Goodfellow et al. , 2016] Goodfellow, I., Bengio, Y., and Courville, A.: Deep Learning. Cambridge, Massachusetts : MIT Press, 2016.

[Hawkins, 2004] Hawkins, D. M.: The problem of overfitting. Journal of Chemical Information and Computer Sciences, 44(1):1–12, 2004. PMID: 14741005.

[Haykin, 2009] Haykin, S.: Neural Networks and Learning Machines. New York : Prentice Hall/Pearson, 2009.

[Hyeong-Gyeong and Jung-Hoon, 2010] Hyeong-Gyeong, M. and Jung-Hoon, K.: Intelligent crack-detecting algorithm on concrete crack image using neural network. Department of Civil and Environment Engineering, Yonsei University, pages 1461–1467, 2010.

[James et al. , 2013] James, G., Witten, D., and Hastie, T.: An Introduction to Statistical Learning with Applications in R. Springer, New York, 2013.

[Jordan,] Jordan, G.: Wired.

[Krizhevsky et al. , 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E.: Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems 25, eds. F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, pages 1097–1105. Curran Associates, Inc., 2012.

[Lecun et al. , 1998] Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, Nov 1998.

[Pan and Yang, 2010] Pan, S. J. and Yang, Q.: A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering, 22(10):1345–1359, Oct 2010.

[Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A.: Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014.

[Srivastava et al. , 2014] Srivastava, N., Hilton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15, 15:1929, 2014.

[Yosinski et al. , 2014] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H.: How transferable are features in deep neural networks? In Advances in Neural Information Processing Systems 27, eds. Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, pages 3320–3328. Curran Associates, Inc., 2014.

[Zeiler and Fergus, 2014] Zeiler, M. D. and Fergus, R.: Visualizing and understanding convolutional networks. In Computer Vision – ECCV 2014, eds. D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, pages 818–833, Cham, 2014. Springer International Publishing.