REPUBLIC OF TURKEY
İSTANBUL KÜLTÜR UNIVERSITY
INSTITUTE OF GRADUATE STUDIES

A COMPARATIVE STUDY OF DEEP LEARNING TECHNIQUES IN CONCRETE CRACK DETECTION: CONVOLUTIONAL NEURAL NETWORKS AND LOGISTIC REGRESSION

MASTER OF SCIENCE THESIS

Azhi Yassin RASUL

Department: Industrial Engineering

Program: Engineering Management

Supervisor: Prof. Dr. Fadime ÜNEY YÜKSEKTEPE

JUNE 2021

REPUBLIC OF TURKEY

İSTANBUL KÜLTÜR UNIVERSITY

INSTITUTE OF GRADUATE STUDIES

A COMPARATIVE STUDY OF DEEP LEARNING TECHNIQUES IN CONCRETE CRACK DETECTION: CONVOLUTIONAL NEURAL NETWORKS AND LOGISTIC REGRESSION

M.Sc. Thesis

by Azhi Yassin RASUL

1900001511

Date of submission: **30 May 2021**

Date of defence examination: **14 June 2021**

Supervisor and Chairperson: **Prof. Fadime ÜNEY YÜKSEKTEPE**

Members of Examining Committee: **Assist. Prof. Dr. Zeynep GERGİN**

Prof. Dr. Semra AĞRALI

June 2021

ACKNOWLEDGEMENT

Exalted be he who created existence from nothingness. He who taught humanity what they knew not. Many blessings and praises be to God Almighty for blessing me with the curse of knowledge and the curiosity of learning.

I would love to express my absolute gratitude and utmost appreciation to my parents for their relentless support and backing in this journey.

Also, my supervisor Prof. Dr. Fadime Yüksektepe for her invaluable supervision and support throughout the process of researching and bringing this thesis into light.

And to anyone who was supportive and pushed me to reach beyond my limits and comfort zone, leave home and achieve my dream of academic degrees.

30/05/2021 Azhi Yassin RASUL

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Üniversite : **İstanbul Kültür Üniversitesi**

Enstitü : **Lisansüstü Eğitim Enstitüsü**

Anabilim Dalı : **Endüstri Mühendisliği**

Programı : **Mühendislik Yönetimi**

Tez Danışmanı : **Prof. Dr. Fadime ÜNEY YÜKSEKTEPE**

Tez Türü ve Tarihi : **Yüksek Lisans – Haziran 2021**

ÖZET

BETON ÇATLAĞI TAHMİNDE DERİN ÖĞRENME YÖNTEMLERİNİN KIYASLAMALI BİR ÇALIŞMASI: EVRİŞİMSEL SİNİR AĞLARI VE LOJİSTİK REGRESYON

Azhi Yassin RASUL

İnşaat alanlarında zorlu durumlarla günlük olarak karşılaşılmaktadır. Bu zorlukları yönetmek için yeni teknik ve yöntemler ortaya çıkmakta ve geliştirilmektedir. Klasik Makine Öğrenmesi (MÖ) ve Derin Öğrenme (DÖ) yöntemlerinin inşaat yönetimi alanında kullanılması da artmaya başlamıştır. Makine öğrenmesinin günlük problemleri çözme ve pratiğe dökülmesi için kullanılması, mühendislerin öne çıkarması ve başarması gereken bir görevdir. İnşaat alanlarında karşılaşılan problemlerden biri beton çatlağıdır. Çatlaklar yapılarda ortaya çıkan ve fark edilmesi zor olan hatalı oluşumlardır. Yapılardaki bozulmaları arttıracakları için erken zamanda tahmin edilmeleri çok önemlidir. Bu çalışma basit kameralarla toplanmış olan görüntüden oluşan bir veri seti için çatlak tahmininde DÖ yöntemlerinin kullanılmasını araştırmaktadır. Çatlak ve çatlak olmayan 40000 farklı görüntüden oluşan veri kümesi eğitim, doğrulama ve test olmak üzere üç gruba bölünmüştür. Bu veri kümesi, derin öğrenme yöntemlerinden biri olan ve yapay sinir ağları formundaki Evrişimsel Sinir Ağları (ESA) ve ikili sınıflandırma problem

yöntemlerinden Lojistik Regresyon (LR) kullanılarak analiz edilmiştir. Son olarak, sonuçlar hem iki yöntem arasında hem literatürdeki mevcut çalışmalarla hem de gerçek hayat verileri ile kıyaslanmıştır. Aynı veri setinde hem ESA hem de LR modelleri iyi sonuçlar vermiştir ama ESA yöntemi doğruluk oranı ve kullanım açısından daha iyi olarak değerlendirilmiştir. Elde edilen sonuçlar umut vericidir ve ESA'nın gerçek hayat inşaat yönetim uygulamalarında yakın gelecekte kullanılması beklenmektedir.

Anahtar Kelimeler: Çatlak Tahmini, Derin Öğrenme, Evrişimsel Sinir Ağları, Lojistik Regresyon, Veri Sınıflandırma

Bilim Dalı Sayısal Kodu: ….

University : **İstanbul Kültür University**

Institttute : **Institute of Graduate Studies**

Department : **Industrial Engineering**

Program : **Engineering Management**

Supervisor : **Prof. Dr. Fadime ÜNEY-YÜKSEKTEPE**

Degree Awarded and Date : **MS – June 2021**

ABSTRACT

A COMPARATIVE STUDY OF DEEP LEARNING TECHNIQUES IN CONCRETE CRACK DETECTION: CONVOLUTIONAL NEURAL NETWORKS AND LOGISTIC REGRESSION

Azhi Yassin RASUL

A construction site faces challenges on a daily basis. In order to manage these challenges, new techniques and methods emerge into existence and constantly developed. The utilization of traditional Machine Learning (ML) techniques and Deep Learning (DL) is starting to grow in the construction management area. To put machine learning in practice to solve the daily obstacles, is a task engineers need to address and achieve. One of the problems that faces construction sites is concrete cracks. Cracks are subtle forms of failure that appear in structures. As they will increase the deterioration process in structures, detecting them early in the process is vital. This study investigates the process of crack detection using DL algorithms for a dataset consists of images collected by simple cameras. A dataset of 40,000 images of cracked and non-cracked concrete surfaces is split into three separate training,

validation, and test datasets. The data set is analyzed using Convolutional Neural Networks (CNN), which is a deep learning method and a form of artificial neural networks, and Logistic Regression (LR), which is a method of classification of binary and dichotomous problems. Finally, the results of analysis are compared to each other, other studies in the literature, as well as real-life eye inspection. Both CNN and LR models give satisfactory results on the same dataset, but CNN model evaluated to be better in terms of accuracy and ease of use. The outcome of the analysis is promising and using CNN in real life construction management practice is expected to be utilized in the near future.

Keywords: Crack Detection, Deep Learning, Convolutional Neural Networks, Logistic Regression, Data Classification

Science Code: …

1. INTRODUCTION

The brain of a human has always been the core of fascination of mankind. With its advanced structure, wiring and meticulous functions along with its significance in the life of an individual and the history of human, the human brain proved to be the single organ that grabbed all the attention of scientists and researchers. With the progress in mathematics and neural sciences, the question of incorporating human brain into machines arose. The idea of making a man-made machine to calculate, analyze and perform functions like a human brain became a question that researchers wanted to develop an answer for. The first attempt towards figuring out the answer was by Turing in 1950, when he questioned whether machines were able to think. This was the first step towards many more studies in this field of science. Scientist came to a collective agreement to give the name of Artificial Intelligence (AI) to the branch that gathered the related sciences. The introduction of the term, AI, helped the cooperation of sciences that seemed unrelated before, such as mathematics and neuroscience. Consequently, studies on human intelligence, machine intelligence and their interactions intensified.

As time passed and the computational science reached a level never seen before, the problems started focusing on problem solving and optimizing the ideal situations by defining the problem, detecting the constraints and objective of the problem aiming towards a novel solution. This step suggested that in order to solve these problems, we needed more data. Since modern computational sciences started finding its way in algorithms, it's normal to anticipate the requirement of more data. And more importantly, what kind of data? Should the acquired data consist of numbers or other forms can also be used? One of the branches of AI that is strongly in use in our day in Machine Learning (ML). The data used in machine learning is from previous occurrences of the problem under study and it reflects the implicit field studies in which the system structure is developed. ML has feeds on data forwarding and back warding mechanism, which is called training of data, this

technique is mainly used in optimization processes. The data used in ML is used depending on the situation, it can be either numerical or categorical. In other words, categorical data is qualitative whereas numerical data is quantitative. Machine learning techniques used in processing such large amount of data is include classification, regression, association analysis and clustering. As mentioned before, these techniques are only valuable when there is enough prepossessed data in hand. Now, we can imagine the data used in terms of numbers, but what if we want to analyze images? Is there a way to convert the image into numbers and we extract the features of the images in question and classify them? This is where a ML branch called deep learning comes into the picture.

Deep learning is considered to be a subset of ML algorithms. Deep learning gained popularity as a response to the demand of big data processing (Alsheikh *et al.*, 2016; Wilamowski, Wu and Korniak, 2016; Zhang *et al.*, 2018). The difference between deep learning and traditional ML is that in the later, researcher is required to explicitly define the features of the data in hand and then process it, however, in deep learning the algorithm is defined in such a way that it performs the feature extraction withing itself, defining the relevant features and its learning parameters. With deep learning, the associative way of problem solving becomes antiquated and the innovative approach detects features in which the researcher may see as irrelevant. Nevertheless, these features can lead to further inspection and introspection thus, a new out of the box way of problem solving. Overall, we can say that Deep Learning describes a family of learning algorithms rather than a single method used to solve all the complex problems (LeCun et al., 2015).

The most prominent uses of deep learning are performing various key tasks in image understanding. Deep learning is widely adopted to perform tasks such as object detection, image classification, retrieval, and semantic segmentation (Özgenel, 2018). Deep learning enables us to inspect images in more details by assigning as set of labels and probability of detecting a particular object. Our main focus in this study will be image detection. More specifically, detecting cracks on concrete surfaces.

Cracks are one of the major challenges that an on-site civil engineer has to deal with. Cracking appears in various structures including concretes, pavements, and masonries. The growing of cracks increases the risk of deterioration and ultimately

endangers the habitants and users of the structure. There are various forms of cracks depending on their shape, form, line of projected propagation, etc. and detecting these can be very subtle at some points. Therefore, the civil engineer and their crew must be alert and try to solve the issue as soon as possible. Traditionally, the inspection and detection of cracks are performed manually. i.e., someone must regularly check and detect. However, these procedures need excessive time and there is no reliability that the inspections are performed on required time. In addition, due to human reasons, repeatability and reproducibility will not be at required level and it also needs a lot of labor. Hence, the benefits of deep learning come into play.

The current study is focused on using deep learning to detect the cracks by simply taking their pictures and importing them to the model and letting the deep learning algorithm detect the cracks. Second, the suitability of such methods in real life applications and its requirements will be discussed. Moreover, the effects of this prediction from the perspective of labor requirements and the changes that can be initiated will be investigated.

This study is comprised of 5 chapters, chapter 1 is the introduction to the thesis. Chapter 2 is a review of modern literature in academia and the practices that used deep learning and machine learning in Engineering in general as well as other disciplines. Chapter 3 is a thorough description of the methodology undertaken in this study to analyze the dataset. Chapter 4 examines the implementation of the theoretical methods and showcases the results obtained and whether they work or not. Finally, chapter 5 is a conclusion of the study with comments regarding the results and suggestions for future studies.

2. LITERATURE REVIEW

In the wake of the 21st century and information technology, one word is topping the premises of innovation. In fact, it is the building blocks of modern technology and groundbreaking entrepreneurial inventions, that is "Data". In recent times, data boomed in an unprecedented manner and novel terminology opened their way inside the scientific literature such as "Big Data". With the presence of new literature and the plethora amount of data standing and piling up in front of scientists and engineers, other novel terminology was needed to deal with the new form of data in existence. Hence, the scientific community came up with new terminology for people that extract, separate, refine, decide, and act on the relevancy of data. Such experts go by the names of data scientists, data architects, feature, or data engineers, etc. Henceforth, their new key role in the field is to decide on how relevant a dataset is for a certain problem, and what enhancements are necessary in that regard. Then, machine learning emerged and branched out into numerous techniques and a leap in computer technology meant a leap in machine learning, and therefore, new ways and the birth of neural networks from there deep learning and many other branches.

This chapter presents the current state of science on machine learning and the difference between traditional Machine Learning (ML) and Deep Learning (DL) in the first section. The second section is separated for the literature regarding Convolutional Neural Networks and finally, the third chapter is to discuss logistic regression.

2.1 Deep Learning vs Traditional Machine Learning

To define ML, going back to the original definition by Arthur Samuel is an eye opener, where machine learning is defined as "the field of study that gives computers the ability to learn without being explicitly programmed" (Samuel, 1959). Despite the importance of this definition as it clarifies the principal idea behind machine learning, this definition is not so technical. Tom Mitchell (1997) gave a more technical definition as "systems which are optimized with respect to the patterns in

raining data in order to make a prediction for new query" which is more technical and give an idea about the steps and how ML functions.

Traditional ML techniques deal with an input of features. The word "input" suggests that they are manually determined and needs adjustment. Henceforth, traditional ML requires a feature engineering phase. In this phase, the user excerpts the features that are believed to be significant and benefit the task in hand. Data preprocessing occurs by means of probability distribution calculations and number of features appearing in the solution. This process is different in DL, for DL is a form of utilization of neural networks. Inspired by the work mechanism of human brain, Artificial Neural Networks (ANN) works by receiving an input and analyzing in the inner layers of the network. DL algorithms omit the preprocessing phase which the relevant features are determined. As a result, the user selects the data which are believed to represent the task (Özgenel, 2018). This suggests that the data preprocessing part doesn't need a manual intervention as done in traditional ML. Wang et al. (2020) described traditional ML as methods dealing with shallow structures for, they possess only one or two non-linear feature transformation layers which can be regarded as having one or no hidden layers at all. Wang et al. (2020) suggested that the reason DL is described as "deep" is relative to the shallowness of the data structure in traditional ML. According to them, DL have five, six or more hidden layers. Hence, for the purpose of data extraction, the choice of data has primary significance in DL.

"The concept of deep learning was put forward in 2006 at first. After that, DL is still continually developing at broad." (Du et al. 2016). Ever since then many researchers and big tech companies such as Google and Facebook have made loads of research and achievements in deep learning and implemented them on numerous fields. Also, since 2010, yearly international challenges for deep learning algorithm development are held, such as The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in which programmers and researchers present their most recent findings the area of deep learning for image classification and object detection. ILSVRC gave birth to the algorithm of pretrained networks such as AlexNet, GoogLeNet, etc.

In recent times, researchers worked to study the distinguishing characteristics and necessary development Traditional ML and DL bring to their respective branch of science. For instance, Wang et al. (2020) analyzed image classification using traditional ML and DL. The methods used in the study were support vector machines (SVM) and Convolutional Neural Networks (CNN). Both methods were applied on large scale datasets of MNIST, and it was found that CNN performed better than SVM on large datasets. However, on small sample datasets, SVM performed slightly better. Paterakis et al. (2017) compared traditional ML with DL for aggregated energy demand prediction. In the study, eight traditional ML methods such as SVM, Gaussian Process, regression trees, etc. were used for the analysis of the day ahead energy prediction, on the counter part, multi layers perceptron method of DL was used. As a result, it was concluded that multilayer perceptron outperformed all the other eight methods used as a benchmark in that study. DL and ML are not necessarily solely used on image recognition and object detection problems, Stanik et al. (2019) used traditional ML and DL to analyze multilingual use feedback for applications and classify them into user reports, inquiries and irrelevant. The results obtained from the study is that DL produced results comparable to DL.

2.2 Convolutional Neural Networks

The recent advance in ML has been influential to fields that have been seeking independent running of duties. Increasing capabilities of ML framework, hold promise for high speed, high accuracy, and high precision and autonomous predictions (Özgenel, 2018). Henceforth, it's safe to say that computer algorithms could have the ability to perform decision making tasks without being subjected to biases. In the heart of cracks detection problems, different ML techniques consist of a few steps that are essential to output a result, those steps are image segmentation, feature extraction and image classification and then an output. On the other hand, CNN omits these manual steps and automatically performs them in the hidden layers within a single framework without the need of image segmentation. So, it can be said that due to its autonomous nature, CNN avoids many biases that are caused in manual feature extraction processes.

Several studies in recent literature are found that implemented CNN on crack detection problems such as (Özgenel, 2018) where pretrained CNN networks were applied to a several datasets of more than 40 thousand images. In the end of the study, Özgenel compared the accuracy obtained from each network and produced a new parameter confidence weighted accuracy to measure and compare each network. Zhang et al. (2016), trained a CNN from scratch to evaluate 500 images of pavement cracks taken by a low-cost smart phone. At the end of the study, it was concluded that DL network methods provided far superior results in feature extraction and crack detection performance when compared to other manual feature extraction methods.

The studies on crack detection and DL still continue with slow yet progressive results, Alipour et al. (2019), used deep fully CNN for crack detection at a pixel level. After analysis, the results showed that the proposed model in the study successfully detected over 92% of crack pixels and a staggering 99.9% of non-crack pixels and later compared the results to other studies and traditional CNN methods of patch wise model and traditional edge detecting methods. Kim et al. (2018) also analyzed crack detection on concrete surface images using CNN. Their study came with an accuracy of 98% of detecting crack and non-crack images. Cha et al. (2017) conducted a study using a framework of 4 convolutional layers for crack detection in concrete buildings. In the study, the relation between training dataset size and performance of network is investigated by training the network on various datasets of different sizes ranging from 2000 to 40,000 images. At the end of the study, it was advised to utilize more than 10,000 images for training based on validation scores obtained from analysis. By taking these studies into consideration, in this study a 40,000 images dataset is used with 28,000 split for training in which the details will be discussed in the next chapters.

It is important to point that, along with comparison, the aim of this thesis is to obtain near optimum results from the CNN networks. Hence, benefitting from previous studies, the best conditions and requirements needed were set for the network and then analysis are conducted, as well as the nuance improvement in the model to prevent the issues other studies have faced.

2.3 Logistic Regression

Logistic regression is useful technique to understand complex phenomena (Connelly, 2020). It's a form of traditional ML that is used to analyze and solve dichotomous and binary problems. In LR, it's required to test a predictor of a dichotomous dependent variables combined with independent numerical variables such as whether a patient lived or died, or in the case of this study, whether the concrete is cracked or not. This information involved is extended to predict similar instances using a set of predictors.

Even though LR is an old classification technique, in recent times many research studies and comparative studies have been conducted using logistic regression in classification problems. Musa (2012) used SVR in comparison to LR used on various balanced and unbalanced datasets. The study concluded that generally, SVM and LR have an overall equal performance for balanced and unbalanced data but, SVM works better with data.

The use of LR is not bound only to a single field of study. It's been used in variety of fields. Dreiseitl and Ohno-Machado (2003) studied the difference between LR and ANN in classification tasks of biomedical data. The study concluded that LR is working well with such data and by tweaking and enhancing some parameters, LR can be utilized. Chen (2011) used an integration of decision tree and LR for predicting corporate financial distress. For the purpose of improvement in the rules of operation of the Taiwan stock exchange corporation, the author collected a list of 100 companies as initial samples. Decision Tree and LR were used to apply the financial distress prediction model. The author concludes that in short run, decision tree obtains better results, however in the long run, LR is significantly better prediction accuracy. For that, it is advised to approach the problems with AI for it will be more suitable than traditional statistical methods.

In recent times, not much research can be found in literature using LR for crack detection. Moreover, to further complicate the specification, almost no studies of crack detection on building concrete surfaces by LR could be found. This in turn makes this study unique in its own. Recent studies about LR and crack detection

include about developing a crack detection algorithm for non-routed pavement images, using ANN and binary LR (Yoo & Kim, 2015). In the study, actual high resolution pavement images have been used to compare and verify the accuracy of the algorithm that is based on ANN and LR. In the study, 7 images of actual pavements were selected to extract 8 characteristics from binary images and more than 300,000 object data. The study derived a LR equation to determine crack and noise objects in the images. The results showed that, despite scoring a recall and precision more than 95%, LR didn't work as well as ANN. ANN showed more precision but less sensitivity. Thus, accuracy of ANN was found to be way superior to that of LR. After conducting a comparative analysis, it was found that ANN model was 82% more productive than LR mode from an actual operator's perspective.

The main idea of this study is to develop a model that can be used on a daily basis on construction sites to facilitate detecting cracks in various members of the building, thus, CNN and LR are there to help achieve this. From the literature study, there is a common consensus that ANN works better, but there is no abundance in these comparative studies to give a solid information in this regard. Additionally, this study is first of its type that compares CNN to LR, using a huge dataset. All these factors combined, will help to have a say in this argument supported by empirical results.

3. METHODOLOGY

The methodology used in the process of analysis of the case can be split into two different sections: convolutional neural networks (CNN) and Logistic Regression (LR). Both methods aim at classifying the available data into classes. As shown in the Figure 3.1 the data is prepared and split into training and test sets, then CNN and LR are implemented. And finally, both methods are tested and compared.

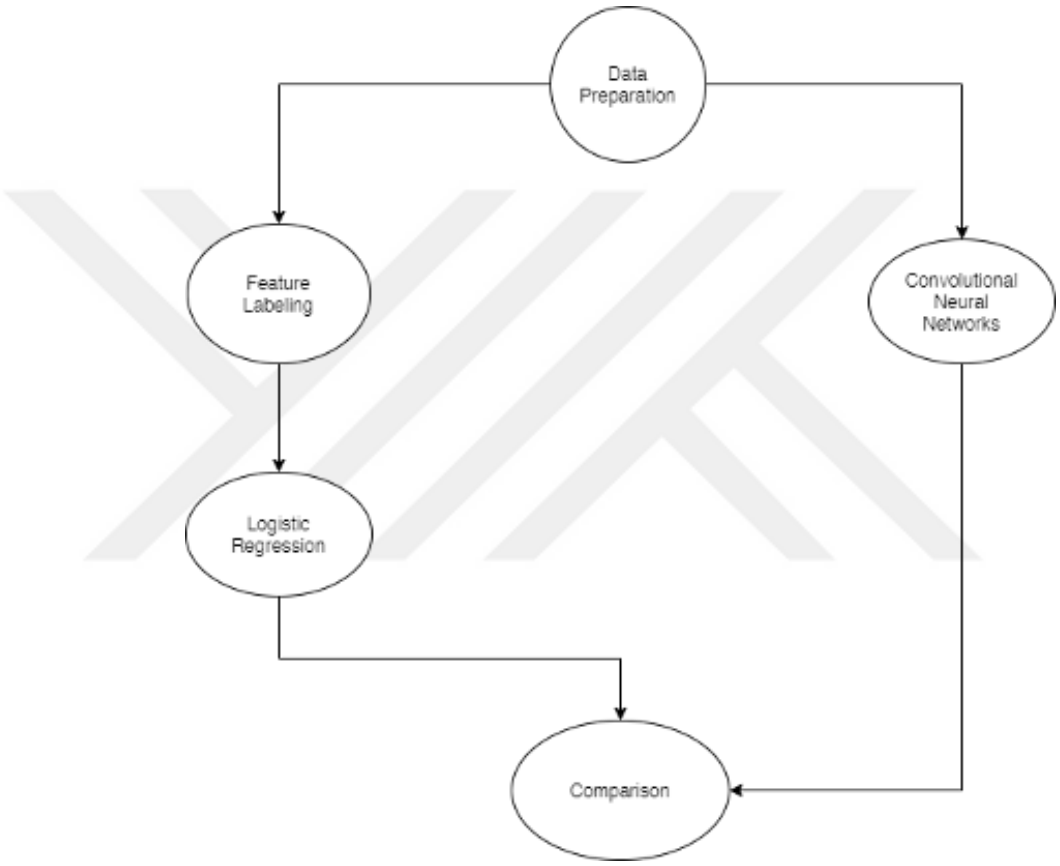


Figure 3.1: Roadmap of the study.

3.1 Dataset

3.1.1 Data Acquisition

The size of the dataset in training is one of the key factors that aids in obtaining a more reliable result for analysis. Since the more data, the network sees, the more accustomed it'll be to the challenges and problems coming on its way. This is not the case for pretrained networks which are already trained on a vast amount of data, and

thus it requires a lot less data in order to become a well-trained network, however, more data is always better when it comes to training neural networks.

The dataset used is obtained from Özgenel and Sorguç (2018) which was publicly available on Mendely. The dataset includes cracked concrete. The data is collected from several METU Campus Buildings and is divided into two. In other words, negative and positive crack images for image classification. Each class has 20,000 images and a complete of 40,000 images with 227x227 pixels with RGB channels. The dataset is made from 458 high-resolution images (4032x3024 pixel) with Zhang et al. (2016) proposed method. High-resolution images vary in numerous conditions, specifically, in terms of surface finish and illumination. No form of data augmentation is applied to the images.

3.1.2 Data Split

The dataset of 40,000 images is randomly split into 3 categories: training, validation, and test sets. Each set contain equal values of crack negative and positive images. The partitioning is conducted on a ratio of 70, 15, 15, respectively. As a result, there will be 28,000 training images, 6000 validation and 6000 test images in hand. It's important to remember that each set has 50% negative and 50% positive images. Thus, the procedure is made sure to be conducted in an unbiased manner.

3.1.3 Training dataset

As its name suggests, this dataset is used to train the network, or to instigate the process of learning by the constructed model. This dataset is the base pictures in which the network bases its judgement on. In other words, the model will be deciding on the test sets and other images by comparing it to what it already has seen and observed. At first, error on the training set is huge and as the process of learning continues, the value of error lessens. This process is displayed in learning curves.

3.1.4 Validation Dataset

Validation dataset is also known as the monitoring dataset. The aim of this dataset is to monitor the training process through the learning curves. The number of images is chosen by the ratio that is originally designated, which was 15% of the whole dataset. This ratio is similar to the ratio in the Özgenel and Sorguç (2018).

3.1.5 Test Dataset

As the name suggest, this dataset is used to examine the model on data that are not seen by the model during training and validation. In other words, brand new unseen images will be classified by both models. This dataset constitutes the remaining 15% of the original dataset of 40,000 images. In order to challenge the model even harder, the author took images of cracked surfaces in his own house and tested them using the model, in which its corresponding results will be discussed in Chapter 4.

3.2 Convolutional Neural Networks (CNN)

Artificial Neural Networks (ANNs) are computational processing systems that are intensely inspired by the way of operation of biological organ systems. They are formed by highly interconnected computational nodes. Since the whole systems is inspired by the human brain, these connecting nodes are called neurons. Neurons are distributed in multi-level fashion, and they receive a series of inputs and in turn, convert them into outputs of meaning. An ANN can work with multiple inputs as well as having various layers of neurons. A simple structure of a neural network is demonstrated in Figure 3.2.

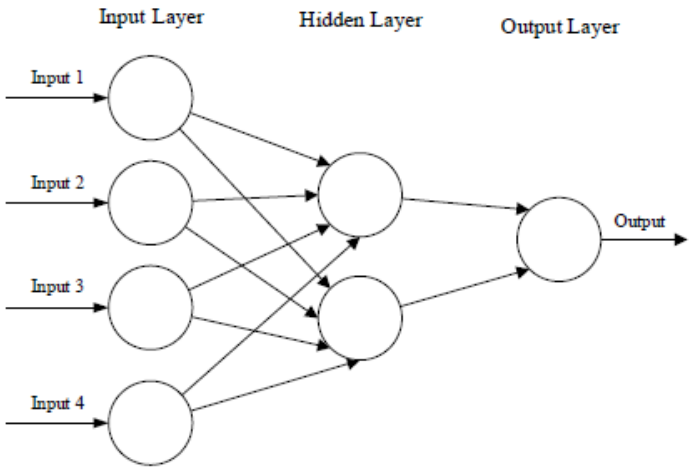


Figure 3.2: A simple 3 layered artificial neural network (Keiron O'Shea et al., 2015).

In a basic structure of ANN, multi vector data are entering through the input layer, for the most part as a multidimensional vector to the input layer of which will transfer and distribute them to the hidden layers. The hidden layers will at that point

settle on choices from the past layer and weigh up how a stochastic change inside itself hinders or enhances the last output, and this comprises the process of learning.

Having different hidden layers stacked upon one another is often referred to as deep learning. This process is performed by the network itself without any outside interference by conducting researcher. Thus, the exciting part is that the output results are generated by the neurons. This means there are huge possibilities of encountering novel findings in the data, since the network is training itself and reaching outcomes by itself too.

As mentioned before, Convolutional Neural Networks (CNN) deviate from the path of traditional machines learning due to their computational structure. As a form of deep learning, it can be understood that CNN is a form of ANN that optimizes the visual input data through self-trained learning. Convolutional Neural Networks (CNNs) are analogous to standard ANNs in that they're constructed from neurons that self-optimize via learning. Each neuron will nevertheless obtain an input and carry out an operation. This is the principal assumption of countless ANNs. From the enter raw image vectors to the very last output of the class score, the whole network will still convey a single perceptive rating function (the weight). The final layer will include the loss functions related to the classes.

The most notable difference between CNN and traditional ANN is that CNNs are primarily used in the field of pattern recognition with images (Keiron O'Shea et al., 2015). This means that it's possible to insert image features into an architecture and then undergo the learning process throughout multiple hidden layers, since CNN is a part of deep learning, and finally obtain outputs from the trained network. This makes CNN extremely suitable for image centered tasks.

One of the problems encountered even before starting the training process is the complexity of calculation for computing image data. The dimensions of each image as mentioned before is 227×227 with RGB channels. This means the weight of a single neuron is 154,587. To compare this, standard benchmark datasets in machine learning such as the Modified National Institute of Standards and Technology (MNSIT) database, which is a dataset of images of handwritten numbers, has a shape of (28×28×1), a total of 784 is the weight of a single neuron. The number 1 means

that the image in question has 1 color channel, i.e., its grayscale. Meanwhile 3 means it's colored. It can be observed that colored images weighted more, and therefore, more analysis parameters.

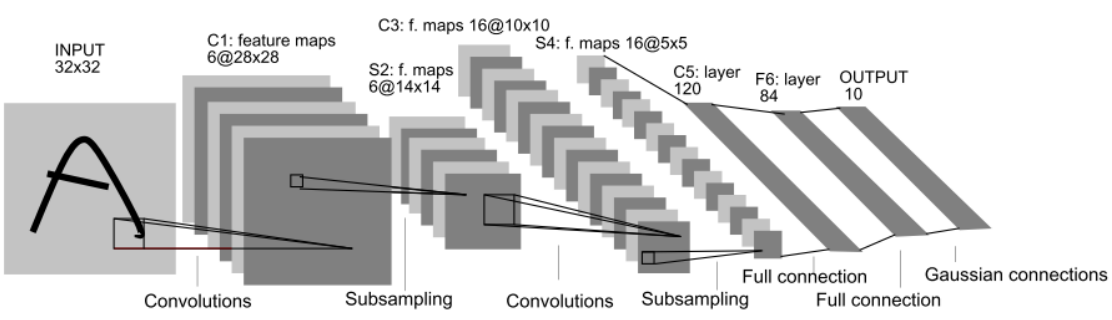
The model constructed in this study deals with colored images and thus, 3 color channels. In the network architecture section, a detailed view of the network and the total number of parameters will be given.

As understood, the more neurons in a single network, the more accurate is the output. However, there exists a major drawback. Huge networks of numerous neurons will consume too much computational power and time to train the ANNs, as a result, new ways should be taken to tackle the issues of inaccuracy. One of such issues is the curse of overfitting. Tan et al. (2014) describe overfitting as the small number of misclassifications of unseen data. To elaborate more, in classification examples there are two types of error, training and validation errors. Training errors are the number of misclassifications during training phase, whereas validation error is the error that is expected from the network on a previously unseen record. A good classification data must not only fit training data well, but also unseen data too. To ease this even more, a good model must have low number of both types of errors. When a model fits the data in training exceptionally well but doesn't show the same in validation, it means overfitting happened there. An overfitted data will show exceptionally remarkable results that's when you know something is not right.

3.2.1 CNN Architecture

CNNs function similarly to ANNs in terms of hidden layers, weights, and bias factors. These factors are all optimized throughout the training process. In contrast to ANNs, CNN layers do not require to be fully connected. In addition, CNNs inputs are represented as 3D arrays or tensors having height, weight, and feature channels. While ANNs are commonly utilized with limited number of inputs in feature space (Özgenel and Sorguç, 2018).

A typical CNN consist of 4 major types of layers to convolve: convolutional



layer, activation layer, pooling layer, and fully connected layer. These are commonly known as the building blocks of convolutional neural networks as shown in Figure 3.3. The CNN architecture used in this analysis is a simple convolutional neural network consisting of eight layers. Those are two convolutional layers, two maxpooling layers, one flatten, one dropout layer and a fully connected layer.

Figure 3.3: A simple depiction of CNN plan (Lecun et al., 1998).

3.2.2 Convolutional layer

As the name of the network suggests, this is the most important operation in the network. In this stage, images are presented in numerical values for each pixel in the image. Each number corresponds to the row and column cells' value which denotes the brightness of the aforementioned cells. The matrices can be multi-dimensional, reliant on the color mode. To put that into more context, the more the color channels the more depth is given to the image. Consequentially, more dimension. Our dataset is colored using RGB channels. Thus, 3-dimension matrices are used in the layer.

The operation of convolution is the dot product of the image pixel data to a convolution operator, or a kernel. Kernels are matrices of smaller dimensions that are multiplied and iterated from right to left. Figure 3.4 demonstrates the convolution operation where a 7x7 image input is multiplied by a 3x3 kernel. The result of the multiplications is put together side by side from left to right. The output of this operation is called a feature map or an activation map.

CNNs have two other user-defined hyperparameters which are strides and zero-padding. Strides can be defined as the number of steps taken after a convolution multiplication. Zero padding is the number of pixels to pad the image so that the border is incorporated and obtain outputs of preferred dimensions for the next convolution.

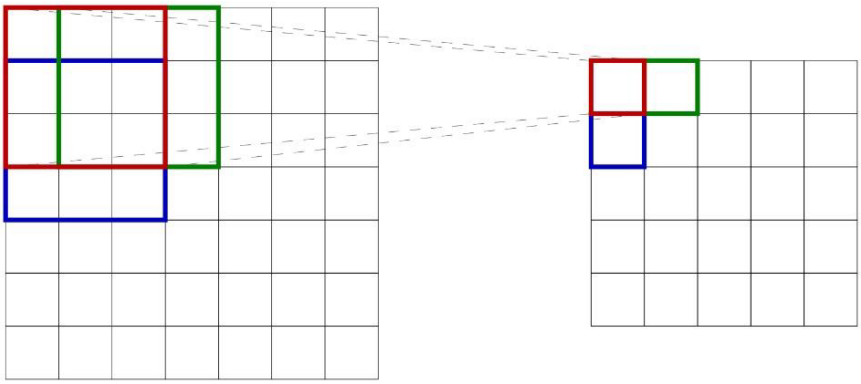


Figure 3.4: Convolution operation of a 7x7 input with 3x3 kernel.

Rectified Linear Unit (ReLU) is the activation function used in the operations. ReLU function states that it will keep the input directly as output if positive and will change the negative inputs into zeroes. Due to its ease of use and it simplifies the training operation, ReLU has become the default activation function in many networks. Other activation functions include sigmoid and tanh. But as of the moment ReLU is the handiest activation function.

3.2.3 Pooling

Pooling essentially reduces the dimensionality of an image in use which helps to speed up the algorithm. To put it in another way, pooling performs the subsampling operation on an image along the spatial dimensionality of the input and produces a reduced number of parameters within the activation in hand. In this study, a 2x2 maxpooling layer with two strides have been utilized to reduce the size of the image for the activation process. A 2x2 maxpooling layer means taking the feature map and partitioning it into multiple 2x2 regions. Maximum value of each region is taken. To shed more light on that, this means if there is a NxN feature map, the maxpooling layer turns it into a N/2 x N/2 image with maximum value taken from each region. This helps to reduce the number of parameters and conserving the computational powers to train the model meanwhile the most valuable information is preserved.

3.2.4 Flatten Layer

Flattening is the conversion of data into a 1D array to input it to the next layer. The output of the convolutional layer is flattened to create a single long feature vector. And later connect it to the final classification model, which is called a fully

connected layer. In other words, all the feature data are gathered in one row to determine a connection with the final layer. one more time.

3.2.5 Fully Connected Layer

A fully connected layer is a transformation wherein every output is the result of a linear combination of the input. This means if there are 10 inputs ($X_1 - X_{10}$) a single output is representing the sum of the 10 inputs. It's important to note that there can also exist an arbitrary number of outputs each representing a different linear combination. Fully connected layers are also called dense layers.

3.2.6 Drop Out Layer

As the name suggests, it's a technique that allows us to intentionally drop some of the neurons during the training phase. By randomly setting the output of neurons to zero, the neurons are dropped from training. For example, if there is a fully connected layer with 100 outputs, randomly selected 10 of them is set to 0 and continue to the next layer. At each epoch, a separate set of neurons are removed, and this allows all the neurons to learn. This step is only used during training to allow all the neurons to learn. This helps the neurons to be independent and forces them to learn patterns about the data and not solely some specific features. As a result, they are trained more efficiently, and risk of overfitting is immensely reduced.

3.2.7 Other Parameters

In the later stage of analysis, during the model compilation phase, there are some other parameters to be encountered such as Epoch, optimizers, and loss function. The following is a brief explanation of these parameters that are essential in the learning process.

Epoch, linguistically, a synonym of time, era, period, etc. As the name suggest, it's the specified time for the model to perform the calculations. The process occurs for multiple epochs. To put it in a more mathematical term, a single epoch means a single iteration of the algorithm. 10 epochs are used in the CNN model of this study.

The loss function is used to optimize the parameter values in the neural network model. The loss function maps a set of parameter values of the network into

scalar values, which indicate the degree to which these parameters complete the tasks that the network intends to complete. Several loss functions are available in keras module of python. A binary cross entropy loss function is used in this study.

The optimizer's job is to reduce the loss function. It's a metric that decides how well or how bad the model is doing at predicting. Evidently, the lower the loss, the better the algorithm. The simplest of optimizers is called Gradient Decent. The gradient decent always changes the weights in the direction in which the loss is optimized. This step is taken by calculating the gradient. However, Gradient Decent has downsides which results in not obtaining the best optimized loss function and misses out important points. A better optimizer is "Adam" which is the state-of-the-art method of estimation. Hence, it's the optimizer used in this study.

The summary of the CNN model used in this study is shown Figure 3.5. The model has a total of 2,579,841 parameters of which all of them are trainable.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 256)	7168
activation (Activation)	(None, 48, 48, 256)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 256)	0
conv2d_1 (Conv2D)	(None, 22, 22, 256)	590080
activation_1 (Activation)	(None, 22, 22, 256)	0
max_pooling2d_1 (MaxPooling2	(None, 11, 11, 256)	0
flatten (Flatten)	(None, 30976)	0
dense (Dense)	(None, 64)	1982528
activation_2 (Activation)	(None, 64)	0
dropout (Dropout)	(None, 64)	0
dense_1 (Dense)	(None, 1)	65
activation_3 (Activation)	(None, 1)	0

Total params: 2,579,841
Trainable params: 2,579,841
Non-trainable params: 0

Figure 3.5: All CNN parameters used in this study.

It's important to mention that this network is a simple one compared to the ones used in other studies. For instance, the pretrained networks of ISLVRC consist of more convolutional layers and more or less parameters which in turn affects the

results. However, for the purpose of crack detection, a simple network like Figure 3.5 can perform sufficiently and produce a satisfactory result.

3.3 Logistic Regression

Logistic regression (LR) is another statistical technique used in the field of machine learning. It's a useful method for binary classification problems. It's the name of the function used at the mathematical core of the method, which is also known as the logistic function. It's a method of predicting or setting predictors for dichotomous and dependent variables, such as is the player injured or fit. In LR, dependent variables are always categorical. Independent variables can be nominal, ordinal (ranked), interval, or ratio level (or continuous) data. The predicted class in this study is categorical: cracked and non-cracked concrete surface.

The logistic function, is also called a sigmoid function, was initially developed by statisticians to investigate the growth of population in ecology. The function given in Equation 1 is an S shaped curve that will take any real valued number and map them into 0-1 values.

$$P(X) = \frac{1}{1+e^{-(\alpha+\sum\beta iXi)}} \qquad (1)$$

The graphical representation of the logistic regression is called an S curve, and that's due to the shape of the curve resembling the letter S. The representation demonstrates the binary distribution between the categorical values of the problem in hand. To shed light on an example for a thorough understanding of the curve, assume that there is a list of players along with their respective heights. The goal is to determine the probability of whether they play basketball or not, solely based on their heights, the curve is shown in the Figure 3.6.

The graph showcases that probability of each individual is playing basketball. For the sake of simplicity, 2 categories are assigned, however more categories could be added.

The advantage of a curved line against of a traditional linear regression fit line is that a curve is more robust and is less affected by the occurrence of outliers compared to a fit line. In the example above, the single outlier at 100 cm would

decrease the threshold of comparison between 0 and 1, and instead of a threshold of 190 cm, as threshold of 180 cm will be observed, this results in misclassification of certain points in the dataset.

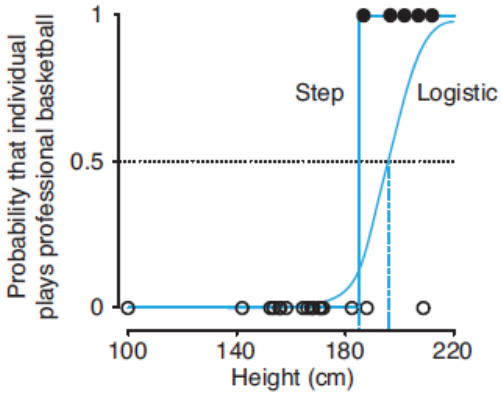


Figure 3.6: Graphical demonstration of S curve. (Lever et al., 2016).

In this study, the image files are imported and with a converted into arrays with a programming script, to set them up for the next step of classification. Later, by utilizing Scikitlearn's logistic regression module from Python, the image dataset is classified into crack and non-crack. The data split is similar to that of CNN, this indicates that there will be 28,000 training, 6000 validation and 6000 test images of both cracked and non-cracked surfaces. Due to the humongous amount of data, the curve of isn't drawn, however, confusion matrix of each dataset is to be obtained. In a similar manner to the CNN, images foreign to the LR datasets are also tested for the exact purpose of ensuring that the method works.

3.4 Analysis Method and Comparison

As mentioned in Chapter 1, the aim of this study is to dive into both methods and look into the facts and scrutinize the reasons supporting each method. Thus, comparison come into the picture. For the purpose of comparing, several data mining methods are utilized. From each method the values of True Negative, True Positive, False Negative, False Positive, confusion matrix, Recall, F1-score, accuracy, precision, and loss function are extracted.

In case of CNN, these values are extracted at the end of each epoch of analysis. The comparison occurs between the maximum values of obtained by CNN at of all epochs for each dataset and the values of obtained for each dataset at the end of

logistic regression analysis. The next section describes the parameters mentioned above in detail.

3.4.1 Confusion Matrix

The confusion matrix summarizes the result of classification problems. It shows how well the model is predicting. The number of correct and incorrect predictions are classified by counting each result and breaking them down to each class. To put it down more concisely, the confusion matrix shows how confused is the model whilst performing the predictions. From the values appearing, the matrix gives an insight into the errors made by the model as well as their respective types. Table 3.1 shows the general format of a confusion matrix for a two-class problem.

Table 3.1: Confusion matrix template for a two-class problem.

		PREDICTED CLASS	
		Yes	**No**
ACTUAL CLASS	**Yes**	TP	FN
	No	FP	TN

As can be seen in Table 3.1, the confusion matrix is a 2x2 matrix where the rows are for the actual class and columns for predicted class. The notations TP, FN, FP, TN are abbreviations for True Positive, false Negative, False Positive and True Negative, respectively. Entries of the confusion matrix is explained as follows:

- True Negative (TN): The number variables that the model predicted to be negative and are actually negative.
- True Positive (TP): The number variables that the model predicted to be Positive and are actually Positive.
- False Negative (FN): The number variables that the model predicted to be negative, but they are actually positive.
- False Positive (FP): The number variables that the model predicted to be positive, but they are actually negative.

3.4.2 Accuracy

Accuracy is an instinctive performance measure. It shows how close the results are to the true value. Accuracy is annotated as the number of true predictions per total predictions as given in Equation 2.

$$\text{Accuracy} = \frac{\text{TN+TP}}{\text{TN+TP+FN+FP}} \qquad (2)$$

The problem with accuracy is that it doesn't tell everything about the data. To elaborate more, take COVID-19 cases as an example, the accuracy of individuals in the population that is tested negative for the virus can be 99%, but this doesn't guarantee safety and that the remaining 1% is not necessary to be dealt with. The high results of accuracy do not present everything needed to evaluate data. Therefore, more evaluation and assessment metrics are needed.

3.4.3 Precision

Precision is the reproducibility degree to which a process can repeat itself. The precision of the model is how many times the model correctly recognizes a cracked image and a non-cracked image. Mathematically it's the ratio of true positives per the total predicted positives from the confusion matrix as given in Equation 3.

$$\text{Precision} = \frac{\text{TP}}{\text{TP+FP}} \qquad (3)$$

3.4.4 Recall

Recall is defined as the rate of sensitivity. Recall calculates the number of positive instances the model captures by truly labeling them as positive. Such it's denoted as the rate of truly detected positive instances per actual results as given in Equation 4.

$$\text{Recall} = \frac{\text{TP}}{TP+FN} \qquad (4)$$

3.4.5 F1-Score

F1-Score calculates the rate of accuracy by taking Recall and Precision into account and thus giving more weight to false positive and false negatives and not letting large numbers of true negatives go out of hand and influence the score. Equation 5 gives the details of F1-Score calculation.

$$\mathrm{F1} = 2 \times \frac{\mathrm{precision} \times \mathrm{Recall}}{\mathrm{Precison} + \mathrm{Recall}} \qquad (5)$$

3.4.6 Loss Function

Loss Function is a way to evaluate the effectiveness of an algorithm in modeling a data set. If the prediction is completely invalid, the loss function will output a higher number. In contrast, if the model is going well, a lower number output by loss function is noticed. By tweaking the algorithm and monitoring the loss function, a better result for the model is obtained.

In the CNN evaluation, loss of both training and validation datasets is compared. A stark difference in loss means the model is not working as desired. Therefore, it is important for both validation and training loss to be relatively close.

3.4.7 P-value Test

The P-value is the smallest level of significance that would lead to rejection of the null hypothesis with the given data (Montgomery and Runger, 2014). It is used to see whether the tests implemented on both datasets have any importance or was it a one-time thing. In other words, to see whether it is possible for the same results to repeat or it was a random instance. Equation 6 shows how P-value testing is calculated.

$$P = \frac{|E1 - E2|}{\sqrt{q\,(1-q)\left(\frac{1}{n1} + \frac{1}{n2}\right)}} \qquad (6)$$

E is the error rate of each dataset, *q* being the average of error rates and *n* is the number of instances. If P value is greater than 2, means that the results between the two instances is statistically significant and is not due to chance.

4. IMPLEMENTATION AND RESULTS

The programming language used is Python 3.8, and anaconda3 is used to write and run the codes on an HP Pavilion g6 computer with Intel core i7 @ 2.20GHz CPU and RAM of 5.00 GB. As can be seen, the analysis device is pretty primitive for 2021, but it's enough to perform both analysis in a reasonable amount of time. Compared to other studies, the limitations of the hardware can be easily noticed. Özgenel used MatConvNet library and Matlab 2017a (Mathworks, 2018) on a desktop workstation with 2 Intel Xeon E5-2697 v2 @2,7 GHz CPU cores, 64GB RAM and NVIDIA Quadro K6000 GPU. The stark difference between the hardware resources and software in use could be easily noticed. Moreover, this affects implementation time and degree of accuracy, which will be discussed later in this chapter. It's important to note that the programming language of Python is chosen due to the knowledge of author. Both CNNs and LR can be conducted using other programming languages that's suitable to the knowledge of respective author.

For CNN, a Python script is written using Keras library. At first, the script assigns labels to each image group since the images are taken separately with no labels. The researcher can categorize them with naked eyes. However, since it's a matter of computer vision, human vision and categorization becomes obsolete.

The script uses 5 libraries of Python and their respective modules come together to undertake the task of classification using CNN models, these include Keras, Numpy, Tensorboard, Pyplot, and Tensorflow. Similarly, there are other libraries involved to log the data and results.

The logistic regression script is significantly less complicated than that of CNN. The logistic regression module from Scikitlearn library of Python performs the analysis in a considerably less time. Nevertheless, the results are also different, and such is the aim of this study that is to be discussed in later sections.

Apart from the size of code and script, the main difference in between LR and CNN algorithm is that LR includes a feature one more extra phase of assigning each label and manually separating the images at first. Then letting the algorithm classify them. In contrast, CNN performs this act of separation on its own inside the hidden layers of the neural networks, this has tremendous impact on the result, for the CNN

learns new features of the image during the convolution phase inside the hidden layers and learns more parameters that will help recognize an image. The same can't be said for LR.

The image dataset used in this study is a massive 40,000 images of cracked and non-cracked concrete surfaces. The crack to non-crack represents %50 of the dataset each. Meaning that there are 20,000 cracked concrete images and similarly, 20,000 of its counterpart. The dataset is split into Training, Validation and Test sets by a ratio of 70, 15, 15. This suggests that there are 28,000 images to training, 6000 for validating the training results, and 6000 to test. Training set contains 14,000 images of cracked concrete and 14,000 non cracked. Validation set contains 3000 cracked and another 3000 non cracked, and finally, test set contains 3000 cracked and 3000 non-cracked images.

Another crucial step in both algorithms is randomization of input images during training phase. If during the training phase the algorithm reads all the images as cracked, then reads non-cracked images, the algorithm would tell that 50% of the times the images are cracked. To avoid this detrimental error in classification, a simple line of random module is written, to randomly shuffle the input images, and hence every time the algorithm runs for training, a different stack of images is trained. Similar to a getting new set of cards every new round in a game of poker.

4.1 Results of Convolutional Neural Networks

An important evaluation metric in CNN is the loss per epoch. Figure 4.1 shows the relation between loss and epochs visualized by Tensorboard.

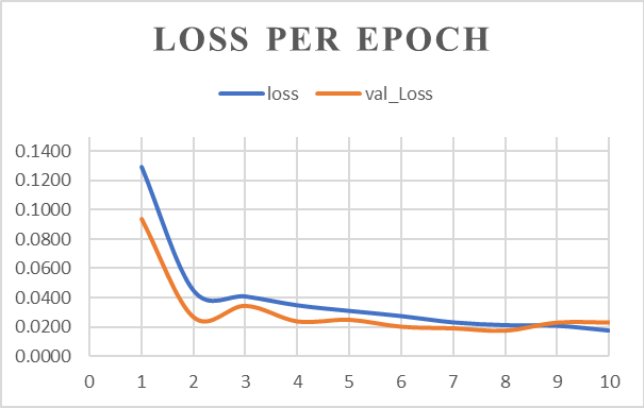


Figure 4.1: Training and validation loss per epoch.

The graph shows that training and validation loss, assigned by orange and blue respectively, decrease as epochs progress. A training loss of 0.0175 in the final epoch is an indication that the model works well. But in order to authenticate the validity of the classification, comparison with validation loss is essential. In the final epoch, the validation loss is 0.0230. It's obvious that there is no stark difference in loss between the two sets of images the model worked on, as a result it can be concluded that the model is good enough to perform a classification problem.

Another parameter to measure the performance of the CNN model is accuracy per epoch. Accuracy is discussed in the previous chapter.

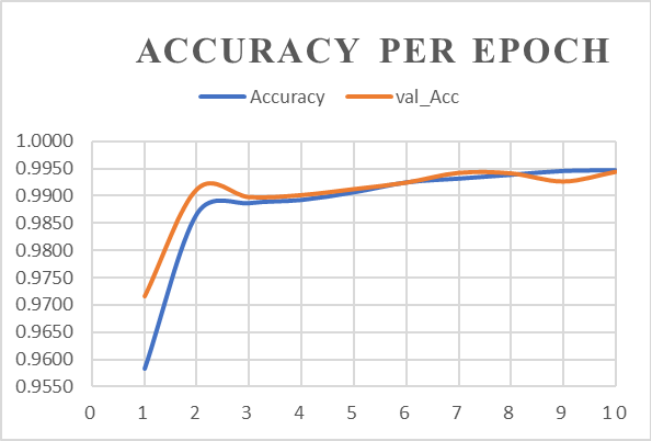


Figure 4.2: Model accuracy per epoch.

As can be seen from Figure 4.2, the validation and training accuracy of the model increases as the epochs go by. It can be noticed that after the 5^{th} epoch, the accuracy becomes stagnated in between 0.99 and above, this suggests that the model reach a conclusion at around that epoch. The remaining epochs carry the risk of falling in the pit of overfitting. The graphs for recall and precision reflect the similar observations regarding the model. As epochs pass, recall and precision increase.

Figure 4.3 shows the relation between the precision of the model and each epoch of analysis. An unlikely graph is noticed for validation precision. That can be explained as the model did not find difficulty in repeating the same procedure and get comparable results on the validation dataset.

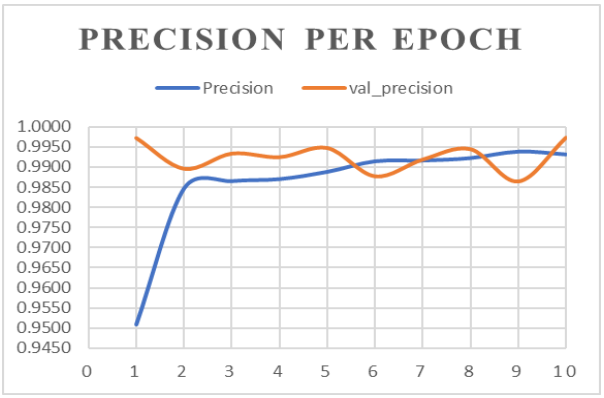


Figure 4.3: Precision per epoch.

Thus, the validation precision at the first epoch is 0.9972 meanwhile training precision is 0.9510. and at the 10th epoch validation and training precision values are 0.9973 and 0.9931, respectively.

At the end of the analysis, recall values is also recorded for each epoch. The values demonstrate an ascending trajectory. Figure 4.4 demonstrates the values of epochs and recall values.

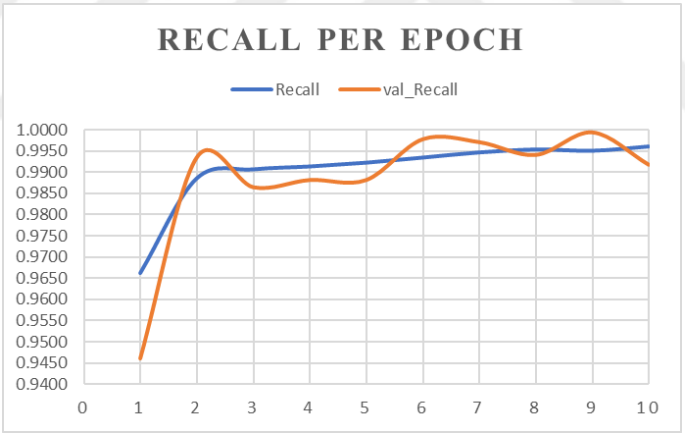


Figure 4.4: Recall per epoch.

It can be seen from the graph that the model generated a training and validation recall of 0.9663 and 0.9461 at the first epoch, and 0.9961 and 0.9917 at the 10th epoch, respectively. This means that with time, the sensitivity of the model to truly capture the true positive instances has increased on both datasets. This means that the model is more likely to capture true positive instances than false instances. Thus, the model is more reliable.

The algorithm is written to give the confusion matrix of the analysis at the end of each epoch, as a mean of comparison to other matrices and also to count the

positive and negative instances along with those wrongly classified. The confusion matrix is drawn for both training and validation datasets at the end of each epoch (Table 4.1 and Table 4.2).

Table 4.1: Training confusion matrix at Epoch 1.

		Predicted class	
		Crack	**Non-crack**
Actual class	**Crack**	13504	471
	Non-crack	696	13329

Table 4.2: Validation confusion matrix at Epoch 1.

		Predicted class	
		Crack	**Non-crack**
Actual class	**Crack**	2862	163
	Non-crack	8	2967

As can be noticed from the confusion matrix, the model successfully classified the images with a truly allotting each to their category. These numbers increase by each epoch, and consequently, the values of Accuracy, Precision and Recall increase as well as, a decrease in loss.

Table 4.3 and 4.4 shows the confusion matrices at the end of the 5^{th} epoch halfway through the analysis.

Table 4.3: Training confusion matrix at Epoch 5.

		Predicted class	
		Crack	**Non-crack**
Actual class	**Crack**	13863	107
	Non-crack	157	13868

Table 4.4: Validation confusion matrix at Epoch 5.

		PREDICTED CLASS	
		Crack	**Non-crack**
ACTUAL CLASS	**Crack**	2989	36
	Non-crack	16	2959

The difference between the confusion matrices of at the end of both epochs can be seen. As the model progresses in the analysis, the number of truly classified images increase and inversely, the number of false classified images decrease. Despite the ascending trajectory in the truly classified instance at these epochs, the increase in the numbers slowed down. Tables 4.5 and 4.6 shows the confusion matrices at the end of the final epoch.

Table 4.5: Training confusion matrix at Epoch 10.

		PREDICTED CLASS	
		Crack	**Non-crack**
ACTUAL CLASS	**Crack**	13920	55
	Non-crack	97	13928

Table 4.6: Validation confusion matrix at Epoch 10.

		PREDICTED CLASS	
		Crack	**Non-crack**
ACTUAL CLASS	**Crack**	3000	25
	Non-crack	8	2967

The difference between the increase of Epoch 1-5 and Epoch 5-10 can be observed as that the increase in the later is rationally less the former. This is due to the fact that the model learned everything the datasets could offer, and only slight improvement is possible from that point forward.

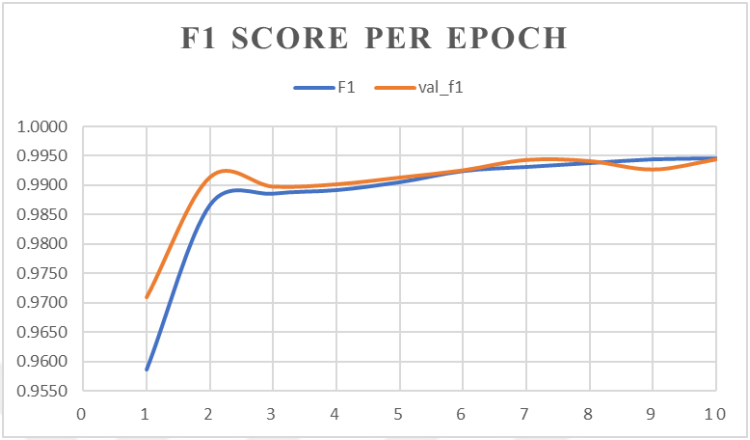


Figure 4.5: F1 score per epoch.

The final parameter is the F1 score at the end of each epoch. Figure 4.5 shows the relation between F1 scores per each epoch of analysis. as the analysis goes, the F1 score increases. Since the F1 scores, consider the details of analysis such as recall and precision, their value owns significance. It can be seen from the graphs that the model is doing a pretty decent job at recognizing and classifying cracked images accurately, with maximum training and validation F1 scores of 0.9946 and 0.9945, respectively.

With the specified properties of the computer used in this analysis, the algorithm took nearly 6 hours to classify 34,000 images. 28,000 training and 6000 validation sets. At full performance capacity, each epoch took around 1600-2200 seconds to classify. The only exception is the 5^{th} epoch when for a brief moment the computer worked on battery, as a result it took 3792 seconds. The minimum time is at the 10^{th} epoch which took 1665 seconds. In the later sections these values will be classified with other studies.

Finally, the model runs the 6000 images in the test set. Unlike the training and validation step, it took only 61 second to run the test set and analyze it. The test dataset produced accuracy, recall, precision, and f score values of 0.9965, 0.9937,

0.9993 and 0.9965, respectively. Table 4.7 shows the confusion matrix of the test dataset and the precise number of truly detected instances.

Table 4.7: Confusion matrix of test dataset.

		Predicted class	
		Crack	**Non-crack**
Actual class	**Crack**	2981	19
	Non-crack	2	2998

It's important to highlight that the test dataset is analyzed all at once and not in epochs, since epochs are, by definition, iterations of training process. In order to test the model, several images from the test set are taken, and the model is predicting respective classes.

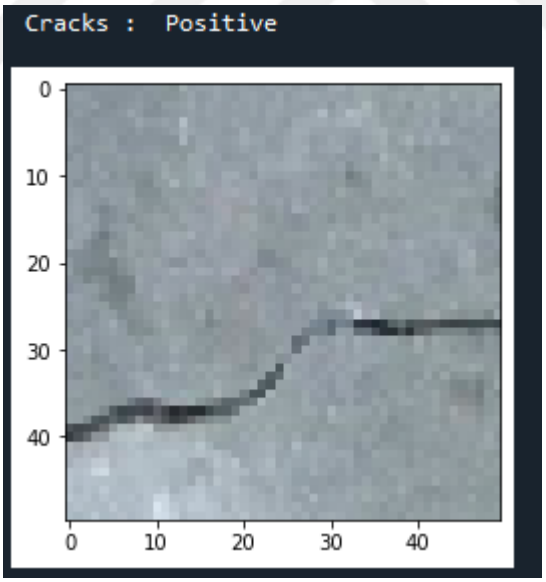


Figure 4.6: Cracked images truly predicted by model.

Figure 4.6 is the image of a cracked concrete surface. The model rightly predicted it to be cracked.

The model also predicted the non-cracked images successfully. Figure 4.7 is an image which the model truly predicted as non-cracks.

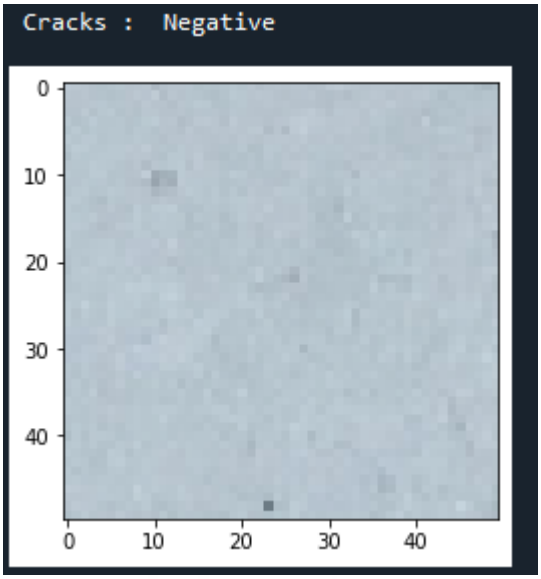


Figure 4.7: Non-cracked image truly predicted by model.

To further try the model, further images are taken by the author in his own home, these images are unique and can't be found neither in the training nor test datasets. The images are of concrete surfaces, taken under similar circumstances in which the images of the main dataset are taken. Figures 4.8 and 4.9 show the image and what the model predicted.

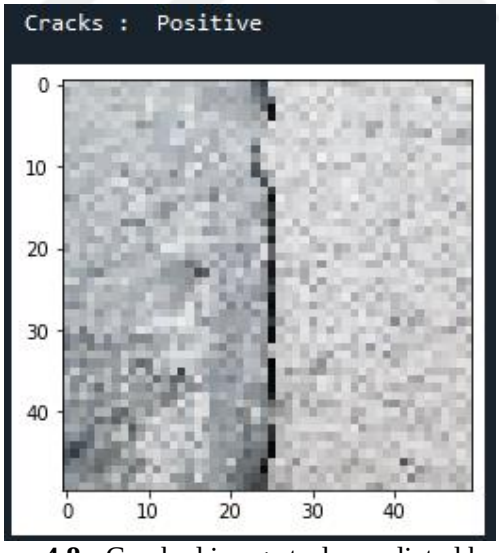


Figure 4.8: Cracked image truly predicted by model.

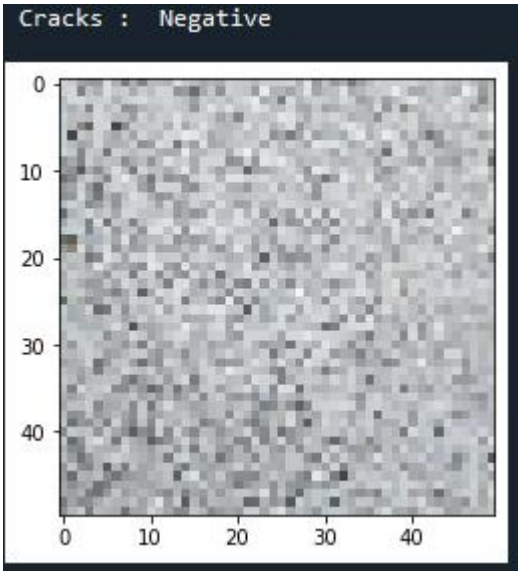
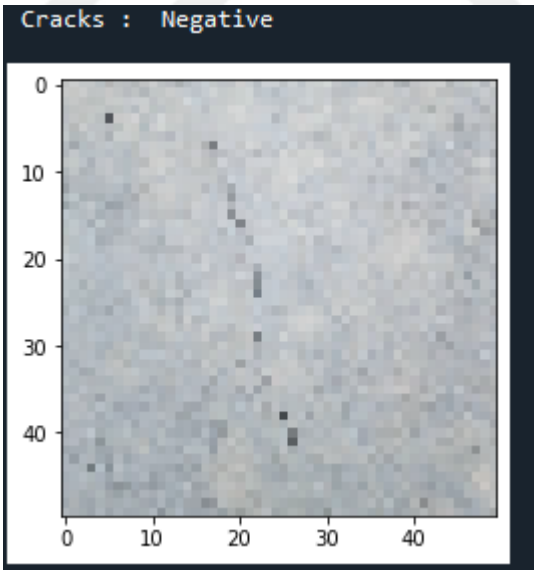


Figure 4.9: Non-cracked image truly predicted by model.

These results show the efficiency of the model, even though there are examples that can be singled out in which the model predicts it falsely as in Figure 4.10 in which the image is of a cracked surface. The model predicts the image as non-cracked. These rare cases are expected, for the reason that the confusion matrix already figured out the possibility of 19 false negative instances at the end of test



dataset analysis.

Figure 4.10: Cracked image falsely predicted as non-crack by model.

To sum up, Table 4.8 and Table 4.9 summarize the CNN analysis and every respective metric extracted from confusion matrices for evaluation at training and validation. At the end of each epoch, the model calculated and evaluated the results.

Table 4.8: Training evaluation metrics for CNN.

Epoch	Loss	Accuracy	Recall	Precision	F1
1	0.1290	0.9583	0.9663	0.9510	0.9586
2	0.0447	0.9867	0.9886	0.9847	0.9866
3	0.0408	0.9886	0.9907	0.9865	0.9886
4	0.0347	0.9892	0.9914	0.9870	0.9892
5	0.0309	0.9906	0.9923	0.9888	0.9905
6	0.0274	0.9924	0.9935	0.9914	0.9924
7	0.0231	0.9931	0.9947	0.9916	0.9931
8	0.0212	0.9938	0.9954	0.9922	0.9938
9	0.0208	0.9945	0.9951	0.9938	0.9944
10	0.0175	0.9946	0.9961	0.9931	0.9946

Table 4.9: Validation evaluation metrics at the end of each epoch.

Epoch	val_Loss	val_Acc	val_Recall	val_precision	val_f1
1	0.0936	0.9715	0.9461	0.9972	0.9710
2	0.0266	0.9913	0.9934	0.9895	0.9914
3	0.0344	0.9898	0.9864	0.9933	0.9898
4	0.0237	0.9902	0.9881	0.9924	0.9902
5	0.0248	0.9913	0.9881	0.9947	0.9914
6	0.0201	0.9925	0.9977	0.9876	0.9926
7	0.0190	0.9943	0.9970	0.9918	0.9944
8	0.0174	0.9942	0.9940	0.9944	0.9942
9	0.0229	0.9927	0.9993	0.9863	0.9928
10	0.0230	0.9945	0.9917	0.9973	0.9945

Next, Table 4.10 shows the time taken by each epoch, in seconds to perform the analysis. The computer is set to maximum performance to undergo the task of analysis and classification as quickly as possible. At the 5^{th} epoch, a problem with

cables led the PC to perform is less capacity and hence the odd amount of time that was recorded for the specified epoch.

Table 4.10: Time taken by each epoch for analysis.

Epoch	Time (s)
1	2226
2	2048
3	1817
4	1868
5	3792
6	1781
7	1737
8	1820
9	1678
10	1665
Total	20432

4.2 Results of Logistic Regression

Logistic regression is performed on the same dataset of images. A total of 40,000 images are divided into 3 sub-datasets of training, validation, and testing. The analytic model uses Python's Scikitlearn libraries to classify the images and calculate the respective accuracy of prediction at each dataset. Finally, at the end of the dataset, similar evaluation metrics used for CNN are used to analyze the outcomes of classification. Unlike CNN, LR doesn't work in epochs, but performs the analysis all at once. Hence, at the end of analysis, only one confusion matrix, accuracy, precision, recall and F1 scores are obtained per dataset. The dataset is split into 3 at the 70, 15, 15, ratio for training, validation, and test, respectively.

At the end of the training dataset, the output is a confusion matrix, and from then, accuracy and other metrics are evaluated. Table 4.11 is the confusion matrix at the end of training phase.

Table 4.11: Confusion matrix for training dataset.

		PREDICTED CLASS	
		Crack	**Non-crack**
ACTUAL CLASS	**Crack**	12463	1504
	Non-crack	702	13331

Also, unlike the CNN model, the LR model doesn't automatically calculate the other metrics, as a result, accuracy, precision, recall and F1 score metrics are calculated using the information extracted from the confusion matrix by the means of manual calculation. Thus, from the equations 2, 3, 4, and 5, it can be learned that the values of accuracy, precision, recall and F1 score of training dataset are 0.9212, 0.8923, 0.9467 and 0.9167, respectively.

Thereafter, test dataset is analyzed by the LR model, as a result the confusion matrix in Table 4.12 is built. And respective evaluation metrics are calculated similar to the training dataset.

Table 4.12: Confusion matrix of test dataset.

		PREDICTED CLASS	
		Crack	**Non-crack**
ACTUAL CLASS	**Crack**	2822	211
	Non-crack	91	2876

From the confusion matrix at Table 4.12 it's learned that the test dataset has an accuracy of 0.9497, a recall of 0.9688, a precision of 0.9304 and an F1 score of 0.9492. it can be noticed that the difference between the values at training and validation are a bit larger than what is calculated by the CNN matrix. as could be

noticed, LR model in Scikitlearn library doesn't have a validation dataset. Rather tests the model immediately.

Prediction using LR is relatively easier done compared to CNN. As for LR, the model can predict individual images and a list of images. To further elaborate this, the model changes the images into array of numbers. Outputs are designated to be in form of integers of 0, 1. 0 for cracks and 1 for non-cracks. The model deals with indexes of outputs designated and named. At first when predicting, the model output the indexes mentioned, by writing a line of code assigning a name to each index, the output is demonstrated as negative and positive.

The predicted output shows "0" or "Positive" for cracked images and "1" or "Negative" for non-cracked images. Individual images could be extracted and predicted separately by entering the index of desired image in the prediction model formula or by directly importing from the source. Also, a group of images, could be predicted at once by writing the range of their indexes. The output will be a list of predictions for image at each index.

To further clarify this, Figure 4.11. represents a visual demonstration of the prediction procedure using the LR model.

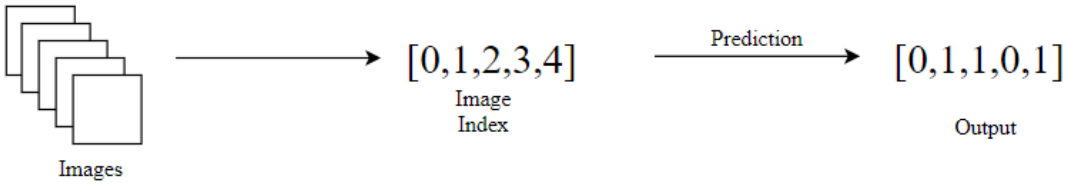


Figure 4.11: Procedure of outputting predictions with LR.

At the first step in the Figure, 5 images are selected for prediction. Each image has is turned into an array with corresponding index, indexes start from 0, hence, an index of zero corresponds to image number 1 and index of 1 for image number 2 and so on. After the prediction is done, an array of results is displayed as seen in the output part of Figure 4.11. The array is informing that the surface of the concrete in image number 1 and 4 are cracked, in contrast, images number 2,3 and 5 are not cracked. Results of LR model, are displayed in an equivalent manner. Figures 4.12, 4.13. and 4.14. show the Python output for LR predictions. Figure 4.12 shows the

prediction of a single number using 0,1 indexes. Figure 4.13 is the categorical prediction of the same image, it's a crack positive image. Figure 4.14 is the index prediction of 100 images selected. For multiple images, the categorical prediction is not possible. So only the index could be obtained. Finally, it's important to note that, it's not possible to upload a single image from device and predict its outcome like CNN.

```
Existence of Crack in this image in term of numbers is : [0]
```

Figure 4.12: Prediction of a single image in index form.

```
Existence of crack in this image is : Positive
```

Figure 4.13: Prediction of a single image in categorical form.

```
 Existence of Crack in these images in term of numbers is:  [0 0 1 1 1 1 1 1 1 0 0 0 1
0 1 0 0 0 1 1 0 1 1 0 1 1 0 0 0 0 0 0 0 1 1 0 1
 0 0 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 0 0 1 1 1 1 0 1 0 1 0 1 1 1 0 0 1
 1 1 0 1 1 1 0 0 1 1 0 1 0 1 0 1 0 0 1 1 1 0 1 1 1 1]
```

Figure 4.14: Prediction of 100 images in index form. (0: Positive, 1:Negative).

4.3 Comparison of the Results

In this section the results of the implemented methods are compared with each other and other studies in the same field. This study is the first study in recent literature that compares Convolutional Neural Networks with Logistic Regression for image classification. Hence the comparison is going to be between the extracted results of both methods. However, since CNN has been implemented before in crack detection problems, such as (Özgenel 2018) and (Alipour et al. 2019), the comparison is going to be with other studies and models followed for the purpose of classification.

The comparison is going to be between the evaluation metrics explained in the previous chapter and found in the other studies too, as well as the computational time it took for the models to perform the analysis.

The study of (Özgenel, 2018), uses the pretrained convolutional neural networks to classify images on the same dataset used in this study. In the study, multiple datasets were formed with different sets for training image. In this study, 28,000 training images are used, and compared with the test with similar parameters in their study.

4.3.1 Comparison of CNN and LR Results

The fundamental difference between CNN and LR is the feature engineering phase which is discussed in the previous chapter. In this study, CNN proved to achieve better results in terms of accuracy and precision. Table 4.13 summarizes the comparative results of both models. For CNN, values of the epoch with maximum values are considered. In this case, it is Epoch 10, which presented the lowest loss and maximum accuracy. The value of the same epoch is used to compare with other studies too.

Table 4.13: Comparative results of CNN and LR.

	Train CNN	**Train LR**	**Validation CNN**	**Test CNN**	**Test LR**
Accuracy	0.9946	0.9212	0.9945	0.9965	0.9497
Recall	0.9961	0.9467	0.9917	0.9937	0.9688
Precision	0.9931	0.8923	0.9973	0.9993	0.9304
F1 score	0.9946	0.9167	0.9945	0.9965	0.9492

From Table 4.13 it can be learned that CNN provides much better results than LR. This is due to the learning process that takes place in the hidden layers of the neural networks. From the confusion matrix of CNN at 10^{th} epoch and confusion matrix of LR at training and testing (Table 4.11 and Table 4.12). It can be said that at

the time LR model truly detects 12,463 instances of cracked images, the CNN model detects 13,920 instances. This is a 7.34 percent difference in the accuracy of both models at training. Comparing the test results, CNN truly detected 2981 positive instances whilst LR detected only 2822. However, it took CNN, 20432 seconds that is more than 340 minutes and nearly 6 hours to come up with these results, meanwhile it took LR less than 5 minutes. Hence, better results needed more time. One more parameter is needed for comparison is to see whether if the results obtained are statistically significant. To find this out, P-value testing for statistical significance is used through equation 6 for both test set of CNN and LR models. From Equation 6, the P-value between CNN and LR was found to be 15.80 which in turn is greater than 2. This achieves that the results obtained by CNN is statistically significant than LR.

4.3.2 Comparison with Other Studies

Next stage of the study is to compare the obtained results with previous studies. One particular study in recent times is by Özgenel (2018) that studied and compared the performances of pretrained CNNs in crack detection. Özgenel used the same dataset used in this study and analyzed them using 7 different pretrained CNN models. Those models were namely, AlexNET, VGG16, VGG19, GoogLeNet, ResNet50, ResNet101 and ResNet152.

The model used in this study contained significantly less parameters than those used by Özgenel. As explained in Chapter 3, the model used consisted of 5 convolutional layers and 2,579,841 trainable parameters, this number is significantly less than those of the pretrained networks, which the least of them is around 7M trainable parameters (GoogLeNet) and 8 layers of convolution (AlexNet). Table 4.14 shows summarized the number of convolution layers and number of parameters of this model and the pretrained models used in Özgenel's study.

Table 4.14: Number of convolution layers and trainable parameters.

Model name	# of Convolution Layers	# of Trainable Parameters
This study	5	2.5M

AlexNet	8	60M
VGG16	16	138M
VGG19	19	144M
GoogLeNet	22	7M
ResNet50	50	25.6M
ResNet101	101	44.5M
ResNet152	152	60.2M

As it can be observed from Table 4.14, it is easy to understand how the naming of most of the networks were decided. The reason the model used in this study doesn't have a distinguishing name is due to its simplicity and widespread use of it in primitive models especially those used to analyze MNIST datasets.

The computation time of each network is heavily depending on the number of convolutional layers. In case of this study, it took a total of 20,432 seconds, that it nearly 5 hours and 40 minutes to train 28,000 images dataset. An average of 2043.2 seconds per epoch. Table 4.15 shows the computation time per epoch for training 28,000 image dataset for each of the pretrained networks.

Table 4.15: Computational time for training 28k images per epoch (Özgenel, 2018).

28k dataset per epoch	Training time (s)
ALEXNET	133
VGG16	2827
VGG19	2943
GOOGLENET	1227
RESNET50	1666
RESNET101	2447
RESNET152	3789

The stark difference between the result obtained from this study and the pretrained network is due to the computers used in respective studies. Due to the fact that the desktop computer used to train the pretrained networks was far more developed than the personal computer used in this study, the difference in computation time is this huge. However, by comparing the time taken for AlexNet which has the nearest layers of convolution to the model in this study, it can be deducted that in case of similar hardware prowess, the model in this study would take less than 100 seconds per epoch instead of 2043.2 seconds. But if the roles are reversed, the computer used in this study would take days to train 28,000 images using the pretrained network VGG16 for instance.

As for statistical significance, the CNN model of this study is compared to each pretrained networks. The AlexNet, VGG16, VGG19, GoogLeNet, ResNet50, ResNet101, ResNet152 models used by their study scored test set accuracies of 0.9988, 0.9997, 0.9997, 0.9986, 0.9991, 0.9990 and 0.9826, respectively. The accuracy of the test set in this study is 0.9965. By positioning each of these values in Equation 6, P-values of each instance are obtained. Table 4.16 summarizes the P-values between the model of this study and the pretrained networks.

Table 4.16: P-values of the compared models.

Models	Accuracy	P-test value
AlexNET	0.9988	2.60
VGG16	0.9997	4.02
VGG19	0.9997	4.02
GoogLeNet	0.9986	2.33
ResNet50	0.9991	3.04
ResNet101	0.9990	2.89
ResNet152	0.9826	7.48

From Table 4.16, it can be seen that all the P-values are greater than 2. For the models except ResNet152, their accuracies are statistically significant with respect to the model used in this study. On the other hand, proposed CNN model's performance is statistically significant than ResNet152's result. When the network complexity and computational time is considered, the proposed CNN model is better with respect to the models in Table 4.16. 99.65% test set accuracy is high and good enough for real life usage. Moreover, computer performance existing in the construction sites will not be so high. Hence, the proposed CNN model will be easily implemented by a construction engineer.

5. CONCLUSION

With the development in science and technology, it's important to adapt to the change occurring in the surrounding environment and applied science in general. The growth in utilization of machine learning in recent literature, has driven many researchers to use this new form of learning in their respective fields of study. Similarly, this study is an attempt to use machine learning techniques, specifically deep learning into the field of Civil Engineering in general and building management to be specific. In this study, the use of machine learning techniques for detecting cracks in concrete surfaces and buildings is investigated. The first part of the study evolves around Convolutional Neural networks (CNN) which is a form of machine learning that uses Neural Networks to analyze images and related problems. A dataset of 40,000 images of cracked and non-cracked concrete surfaces is split into three separate training, validation, and test datasets. At the end of analysis, the evaluation metrics, consisting of loss, accuracy, recall, precisions, F1 score and confusion matrix are obtained from the CNN algorithm. Finally, results are compared to other studies and the second part of this study.

The second part is Logistic regression (LR), which is another machine learning technique similar to traditional Linear Regression. However, LR is used for classification of binary and dichotomous problems. The same dataset is used with the same split as CNN. At the end of this phase, the identical evaluation metrics are extracted to analyze and compare the results.

In this thesis, CNN model went on analyzing the dataset for 10 epochs, which took nearly 6 hours given the conditions and hardware used in the study. At the end of each epoch, accuracy of the model recorded to be a minimum of 0.9583 at Epoch 1 and 0.9946 at the 10^{th} and final epoch. This showed the rate of error to be reduced to minimum. After that, the model analyzed the test dataset consisting of 6000 images. The model scored a 99.64% accuracy. This accuracy is also compared to the other models by means of P-value testing.

At the stage after analysis, images from the test set and other images taken by the author that were outside of the dataset were tested for prediction, and the model predicted the right results. As a result, the CNN model successfully operated on

images within the dataset and outside the dataset. Compared to other studies, these results proved to be competitive enough to be used in crack detection using images. There is a slight chance of overfitting, thus, other models with more convolutional layers can bear a more accurate result. Nevertheless, this does not diminish the results provided by the outputs of the CNN model utilized in this study.

The analysis of the dataset using LR is constituted the second part, in which the LR model showed an efficient accuracy but not as much as the CNN model. From the confusion matrix at the end of analysis, an accuracy of 92.12% is obtained from the same dataset. This is relatively less than the CNN model which is also proven by p-value analysis. Thus, the result of CNN is significantly high and acceptable.

Both CNN and LR models worked on the same dataset, but when it comes to comparison, CNN model worked better in terms of accuracy and ease of use. From the CNN model, individual images could be tested to be predicted, however a group of images can't be tested at once. In contrast, the LR model, can work on group of images, and singular images. Though, the outputs can be in terms of (0,1) integers assigned to each property. In this study 0 was for positive and 1 was for negative. The LR model, could not predict the images outside the dataset that was given to analyze, unlike the CNN that predicted images outside of dataset.

LR is a useful technique for prediction in images however, in field of building and construction management, it does not come handy since it cannot provide the required utility for image analyses, which was the main aim of this research in the first place. The CNN model can be utilized on construction and work sites to predict images for whether they are cracked or not. This reduces time, and amount of work. All the model needs are an image taken in real time and given to the model to analyze.

In conclusion, CNN models can be the next big revolution in construction management field, for its efficiency and ease. The model can predict images and faults within them. Also, by using this model on construction sites, managers can create a database of images and archive them with their corresponding label, later to be used as documentation for the risks that can face the building and be used years later to check flaws during construction or not. This model can be used as a legal

alibi for most of the issues civil engineers face during their career. Additionally, it's keeps the process of decision making easier in many in situ scenarios. The documentation of such cases can save many lives by averting disasters before they come into place.

Finally, the CNN model can be studied further and developed more to finalize the work started in the field of crack detection. For instance, the use of CNN to recognize the types of cracks on concrete surfaces and the probable reason behind it. CNN can also be used in crack segregation and object detection, as well as to recognize the impurities in images and classify an image as non-crack and giving the reason why. Furthermore, it's also a step into digitalizing the job of a civil engineer by importing computer vision and machine learning into the field, this means, the margins of errors and can be reduced to minimum. Hence designing the safest building which is the ultimate goal of a civil engineer.

6. BIBLIOGRAPHY

Alipour, M., Harris, D. K., & Miller, G. R. (2019). Robust pixel-level crack detection using deep fully convolutional neural networks. *Journal of Computing in Civil Engineering*, *33*(6), 04019040.

Alsheikh, M. A., Niyato, D., Lin, S., Tan, H. P., & Han, Z. (2016). Mobile big data analytics using deep learning and apache spark. *IEEE network*, *30*(3), 22-29. Applied to Document Recognition', *Proceedings of the IEEE*, 86(11), pp. 2278 2323. doi: 10.1109/5.726791.

Bengio, Y., & LeCun, Y. (2007). Scaling learning algorithms towards AI. *Large-scale kernel machines*, *34*(5), 1-41.

Cha, Y. J., Choi, W., & Büyüköztürk, O. (2017). Deep learning- based crack damage detection using convolutional neural networks. *Computer‐Aided Civil and Infrastructure Engineering*, *32*(5), 361-378.

Chen, M. Y. (2011). Predicting corporate financial distress based on integration of decision tree classification and logistic regression. *Expert systems with applications*, *38*(9), 11261-11272.

Connelly, L. (2020). Logistic regression. *Medsurg Nursing*, *29*(5), 353-354.

Dreiseitl, S., & Ohno-Machado, L. (2002). Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, *35*(5-6), 352-359.

Du, X., Cai, Y., Wang, S., & Zhang, L. (2016, November). Overview of deep learning. In *2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC)* (pp. 159-164). IEEE.

Gopalakrishnan, K., Khaitan, S. K., Choudhary, A., & Agrawal, A. (2017). Deep convolutional neural networks with transfer learning for computer vision-based data-driven pavement distress detection. *Construction and Building Materials*, *157*, 322-330. *International Conference on Intelligent Engineering Systems (INES)*. Budapest: IEEE, pp. 11–16. doi: 10.1109/INES.2016.7555103

Retrieved from https://www.image-net.org/challenges/LSVRC/index.php

Retrieved from https://www.udemy.com/course/data-science-and-machine-learning-with-python-hands-on/

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Kim, H., Ahn, E., Shin, M., & Sim, S. H. (2019). Crack and noncrack classification from concrete surface images using machine learning. *Structural Health Monitoring*, *18*(3), 725-738.

Kleinbaum, D. G., Dietz, K., Gail, M., Klein, M., & Klein, M. (2002). *Logistic regression*. New York: Springer-Verlag.

Lecun, Y., Bengio, Y. and Hinton, G. (2015) 'Deep learning', *Nature*, 521(7553), pp. 436–444. doi: 10.1038/nature14539.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, *521*(7553), 436-444.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016, September). Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3708-3712). IEEE.

Lever, J., Krzywinski, M., & Altman, N. (2016). Logistic regression.

Mitchell, T. (1997) *Machine learning*. McGraw-Hill. doi: 10.1007/978-3-540-75488-6_2.

Montgomery, D. C., & Runger, G. C. (2014). *Applied statistics and probability for engineers*. John Wiley & Sons.

Musa, A. B. (2013). Comparative study on classification performance between support vector machine and logistic regression. *International Journal of Machine Learning and Cybernetics*, *4*(1), 13-24.

O'Shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Özgenel, Ç. F. (2018). *Crack detection with deep learning: an exemplary study of data design in architecture* (Doctoral dissertation, MIDDLE EAST TECHNICAL UNIVERSITY).

Özgenel, Ç.F., Gönenç Sorguç, A. (2018) "Performance Comparison of Pretrained Convolutional Neural Networks on Crack Detection in Buildings", ISARC 2018, Berlin.

Özgenel, Çağlar Fırat (2019), "Concrete Crack Images for Classification", Mendeley Data, V2, doi: 10.17632/5y9wdsg2zt.2

Paterakis, N. G., Mocanu, E., Gibescu, M., Stappers, B., & van Alst, W. (2017, September). Deep learning versus traditional machine learning methods for aggregated energy demand prediction. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)* (pp. 1-6). IEEE.

Samuel, A. L. (1959) 'Some Studies in Machine Learning Using the Game of Checkers', *IBM Journal of Research and Development*, 3(3), pp. 210–229.

Simard, P. Y., Steinkraus, D., & Platt, J. C. (2003, August). Best practices for convolutional neural networks applied to visual document analysis. In *Icdar* (Vol. 3, No. 2003).

Stanik, C., Haering, M., & Maalej, W. (2019, September). Classifying multilingual user feedback using traditional machine learning and deep learning. In *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)* (pp. 220-226). IEEE.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., ... & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9).

Tan, P. N., Steinbach, M., & Kumar, V. (2014). *Introduction to data mining*. Pearson Education India.

Turing, A. M. (1950) 'Computing machinery and intelligence', *Mind*, 49, pp. 433 460. doi: 10.1007/978-1-4020-6710-5_3.

Wang, K. C. P., Zhang, A., Li, J. Q., Fei, Y., Chen, C. and Li, B. (2017) 'Deep Learning for Asphalt Pavement Cracking Recognition Using Convolutional

Neural Network', in *Conference: International Conference on Highway Pavements and Airfield Technology 2017*, pp. 166–177.

Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, *141*, 61-67.

Wang, P., Fan, E., & Wang, P. (2021). Comparative analysis of image classification algorithms based on traditional machine learning and deep learning. *Pattern Recognition Letters*, *141*, 61-67.

Wilamowski, B. M., Wu, B. and Korniak, J. (2016) 'Big Data an Deep Learning', in

Yoo, H. S., & Kim, Y. S. (2016). Development of a crack recognition algorithm from non-routed pavement images using artificial neural network and binary logistic regression. *KSCE Journal of Civil Engineering*, *20*(4), 1151-1162.

Zhang, L., Yang, F., Zhang, Y. D., & Zhu, Y. J. (2016, September). Road crack detection using deep convolutional neural network. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3708-3712). IEEE.

Zhang, Q., Yang, L. T., Chen, Z. and Li, P. (2018) 'A survey on deep learning for big data', *Information Fusion*. Elsevier, 42(October 2017), pp. 146–157. doi: 10.1016/j.inffus.2017.10.006