Pothole Detection Using Lightweight Network Models

by

S. M. Abdullah
19201050
Shakib Al Hasan
19201049
Antara Firoz Parsa
20101437
MD. Asif Shahidullah Kabbya
20301017
Anika Hasan Talukder
20301331

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2024

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Shakib Al Hasan
19201049

S. M. Abdullah
19201050

Md Asif Shahidullah Kabbya
20301017

Anika Hasan Talukder
20301331

Antara Firoz Parsa
20101437

Approval

The thesis/project titled “Pothole Detection Using Lightweight Network Models” submitted by

1. S. M. Abdullah (19201050)
2. Shakib Al Hasan (19201049)
3. Antara Firoz Parsa (20101437)
4. MD. Asif Shahidullah Kabbya (20301017)
5. Anika Hasan Talukder (20301331)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on January 09, 2024.

Examining Committee:

Supervisor:
(Member)

Dr. Jannatun Noor
Assistant Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Program Coordinator:
(Member)

Dr. Golam Rabiul Alam

Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Head of Department:

Dr. Sadia Hamid Kazi

Associate Professor
Department of Computer Science and Engineering
School of Data and Sciences
Brac University

Abstract

Potholes are defective cavities found on road surfaces. Potholes can lead to serious accidents and vehicle damage if not properly detected. Thus, we are proposing the use of neural network models for pothole classification. The study involves a comprehensive performance analysis of existing lightweight neural network models in pothole classification, compared against the traditional heavyweight models. Lightweight models are emphasized in the thesis due to their low computational requirements, faster prediction times and better compatibility with real-time detection. We have tested six lightweight models (CCT, CNN, INN, Swin Transformer, EANet and ConvMixer) and four heavyweight models (VGG16. ResNet50, DenseNet201 and Xception). A custom dataset of 900 images containing image samples from roads of Dhaka and Bogura was created by the authors to run the models. The dataset was further augmented into 10,000 images by applying various augmentation methods. Separate tests for each model were conducted in the augmented dataset to compare performance against the original dataset. Augmentation enhanced the performance of 9 out of the 10 models. CNN achieved the highest accuracy of 96.55% and the highest F1 score of 0.96 in our testing. Furthermore, CCT exhibited accuracy of 94.6% and F1 score of 0.9. The lightweight models overall performed better than the heavyweight models in both datasets.

Keywords: Pothole, Machine Learning, Neural Networks, Deep learning, Lightweight Models.

Acknowledgement

We are grateful to our thesis supervisor, Dr. Jannatun Noor, Assistant Professor, Department of Computer Science and Engineering for her continued guidance throughout the whole thesis process. We also thank Md. Farhadul Islam, Graduate Research Assistant, Department of Computer Science and Engineering for his valuable assistance and guidance in topic selection and execution of the research. Furthermore, we acknowledge Ibne Hassan, Student, Department of Computer Science and Engineering, BRAC University for aiding us with essential resources of deep learning. Lastly, we want to extend our gratitude to all our friends who cooperated meaningfully throughout the thesis journey.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Background and Problem Statement:

Roads play a pivotal role in our communication infrastructure. Developed and well-connected roads are crucial for our transportation and communication. Roads are the pathways that facilitate the transportation of goods and services, the movement of people, and the development of trade and commerce in one country. Nowadays, potholes in roads have become a major concern in our region. Potholes are basically small to large defective holes or cavities on roads that are caused by a number of factors including massive traffic, weather conditions, and substandard road maintenance [57]. Because of potholes, it is difficult for drivers to maintain control over their vehicles.

In Bangladesh, the condition of the roads is below standard, and streets with potholes cause heavy traffic jams in cities [35]. Potholes have become much more conspicuous at the onset of the monsoon and the condition of road transport gets worse. During monsoon, the roads of Dhaka show notably more potholes resulting in heavy traffic jams [35]. Pothole-ridden roads are risky for commuters and they also induce intensified traffic accidents.

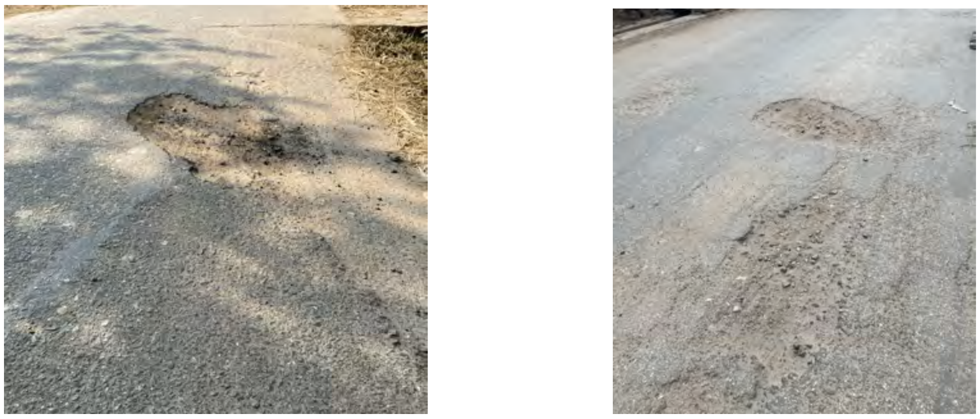


Figure 1.1: Pothole samples from our dataset

In Bangladesh, one of the significant reasons for potholes is the malfunctioning construction of roads. Moreover, the below-standard maintenance of roads and highways contributes greatly to potholes [48]. As per a survey conducted between November 2019 and March 2020, the Zilla roads condition in Bangladesh were overserved to be deteriorating [48]. According to a study conducted in Rajshahi city, 23% of roads were in failure condition [8]. These challenges associated with road infrastructure safety are becoming a significant concern. Based on another study, potholes were found as a major contributing factor in road accidents of Bangladesh. About 80% of the traffic casualty victims in Bangladesh are foot travellers and bicyclists [1]. Another study recognises road potholes as a major factor behind road accidents in Bangladesh [12]. According to an article in June 2021, four vehicles fell in a pothole on the Dhaka-Aricha highway [42]. Some people were injured and a truck driver was killed. Furthermore, according to a report published in August 2023, large potholes were found on a concrete bridge in Ajmiriganj upazila of Habiganj in a crumbling state which poses accident risks in Habiganj [41]. According to a another survey released on RHD website, 12002.75 km (64.97%) roads were in good, 3465.2 km (18.76%) roads were in fair, 1524.29 km (8.25%) roads were in poor, 789.6km (4.27%) roads were in bad and 691.83 km (3.74%) roads were found to be in very bad condition which were considered as one of the major causes behind road accidents [54]. Also, based on an article, the Lebukhali-Dhamki-Chargarbadi roads of Patuakhali are full of potholes and accidents happen frequently there. At Kanthaltala, Satani Bypass Junction of Lebukhali, 20 passengers were injured and bus helpers and three motorcyclists were killed in several accidents [38].

Figure 1.2: A truck fell in a pothole on Dhaka-Aricha highway [42]

For the road safety of Bangladesh, pothole detection plays a key role in this issue. The accurate detection of potholes can prevent major road accidents. With an important consideration of the topic, we are opting to compare performance between existing lightweight neural network models for the detection of potholes. Lightweight models are emphasised in the thesis due to their low computational requirements, faster prediction times and better compatibility with real-time detection. Along with that, we were unable to find any comprehensive performance analysis of existing lightweight models. That's why in our current study, we have comprehensively compared the classification results of existing lightweight with traditional heavyweight models. In addition, there were no Bangladeshi dataset for potholes which we have created for the thesis in order to keep the study in Bangladeshi context.

1.2 Motivation:

The motivation of the research is to analyse the performance of pothole classification among the existing lightweight and heavyweight models in order to determine the optimal models for the task. We aimed to conduct our study in Bangladeshi context in order to address the research gaps in the field. For that, we created our own dataset of 900 images containing image samples from roads of Dhaka and Bogura. Among the 900 images, 462 were pothole samples and 438 were normal road samples. Moreover, we also aimed to find out the effects of augmentation on machine learning models. For that, we augmented our dataset and separately conducted tests for the augmented dataset in all the tested models. In a broader sense, we want to contribute through our study in the mitigation of accidents and vehicle damages resulting from potholes. That is why we attempted to provide a comprehensive and meaningful study of the topic. Ultimately, we want to communicate our findings to city corporations of Bangladesh and propose them to integrate the preferred models into road CCTV cameras in order to ensure live monitoring of potholes. We hope we are able to leave a positive impact with our study.

1.3 Limitation of Existing Research and Our Contribution:

Existing literatures on pothole detection mostly evaluated performance of traditional models. Some literatures utilised lightweight models, however, there was an absence of in-depth comparison of existing lightweight models. We also could not find any comprehensive performance comparison between lightweight models and traditional heavyweight models. Our study mitigates this gap by analysing six lightweight models along with four heavyweight models for pothole classification. Thus, the study would be a meaningful read for any organisation or individual developer who are building pothole detection software. They can look into our analysis to decide which models to employ in their software and which models to avoid. As per our testing of all ten models, CNN achieved the highest accuracy of 96.65% and F1 score of 0.96, which is quite a decent result to consider for any researcher or developer working with pothole detection.

Our study more specifically may benefit researchers or developers who are working with real-time detection and model employment on embedded devices. Lightweight models are more suitable for those tasks. Traditional models have higher parameters which require higher computational powers. That is why lightweight models are the go to options for real-time detection tasks within systems involving limited computational power.

Moreover, there was an absence of any Bangladeshi pothole dataset in the internet archive. To address that, we have created our own dataset of 900 images containing image samples from roads of Dhaka and Bogura. There was also a lack of pothole detection studies in Bangladeshi context for which the whole thesis was conducted in Bangladeshi context, utilising Bangladeshi dataset.

Furthermore, we could not find studies involving a comprehensive comparison of the effects of augmentation on a large sample of models. We have implemented this part in our research. Our study includes in depth comparison of the effects of augmentation on all 10 machine learning models we have tested. Variations of accuracy, f1 score, precision, recall, loss curve and accuracy curves of the models are compared in our study in reference to the augmented dataset.

Additionally, our research also provides an in-depth comparison of training times among all the 10 models we have tested. Training times are vital for researchers and our study may aid researchers to plan the testing of models in a reliable way. Even though training time does not directly correspond to pothole detection accuracy or detection times, still it's an important metric to consider for researchers, students and software developers. We were unable to find any detailed documentation of model training times within the existing literature. To address that, we have documented and compared training times per epoch of all the models for both the original dataset and the augmented dataset. In addition, our study also compares the efficiency and optimisation of the model training times. To illustrate, some heavyweight models are able to complete training even faster than models with fewer parameters. For example, ResNet50, even after being a heavyweight model it was able to complete training faster than ConvMixer and EANet, which had significantly lower parameters. Efficiency of training times is an interesting discussion we could not find in the existing literature. We have compared this factor for all the 10 models we have tested.

With all that, we believe our research may have a meaningful impact on the pothole detection in research sphere.

1.4 Description of Tested Models:

Upon thorough research, we selected the following deep learning neural network models for our study- Compact Convolutional Transformers (CCT), Convolutional

Neural Network (CNN), Involutional Neural Network (INN), Swin Transformer, EANet (External Attention Transformer), ConvMixer, VGG16. ResNet50, DenseNet201 and Xception. The first six of the mentioned models are lightweight models and the last four are heavyweight models. The models were implemented into our custom dataset on which detailed analysis is provided in the 'Dataset' chapter. Below is the architectural description of the tested models. The architecture diagrams used in this segment were generated in draw.io.

1.4.1 Convolutional Neural Network (CNN)

CNNs are a specific type of neural network that is designed to interpret gridlike data, such as pictures [39]. They are perfect for tasks like picture classification and object detection because they are composed of convolutional layers that scan and store local patterns in the input data [39]. Two functions may be multiplied to produce a third function, representing how the shape of one function can change by another in this type of linear operation [40]. It has demonstrated amazing progress in solving the picture recognition challenge by introducing a new degree of scalability and precision [51]. The architecture consists of six layers in total, namely the Input, Convolutional, Pooling, Flattening, and Output layers [51]. A collection of "filters" or "kernels" is pushed over the input layer by convolutional layers whereas in order to simplify processing and minimise memory requirements, pooling layers are placed after convolutional layers [51]. The high-dimensional output from the convolutional and pooling layers is taken by the flattening layer, which uses a fully connected neural network so that it can be flattened into a one-dimensional vector and lastly, the actual output is generated by the output layer [51]. This architecture can process both RGB and Grayscale images also it has the ability to drastically decrease the number of network parameters [58].

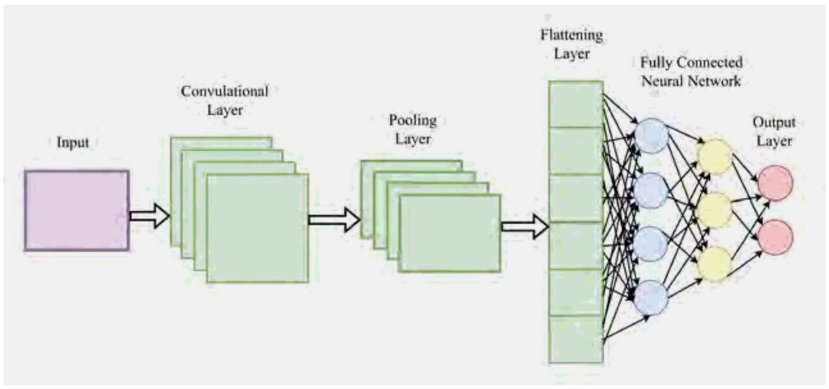


Figure 1.3: Architecture of CNN

1.4.2 Involutional Neural Network (INN)

The INN family of machine learning methods combines neural networks and prototype-based classifiers [23]. Compared to traditional methodologies, they are using less data to provide state-of-the-art performance [23]. INNs are recognised for their ability to produce explainable models, to include both unsupervised and supervised classification techniques, and to learn from small datasets [23]. It is able to identify distinct pairings of latent variables and measurements of the magnetic field by means of a bijective mapping that ultimately results in the invertibility of INN [27].

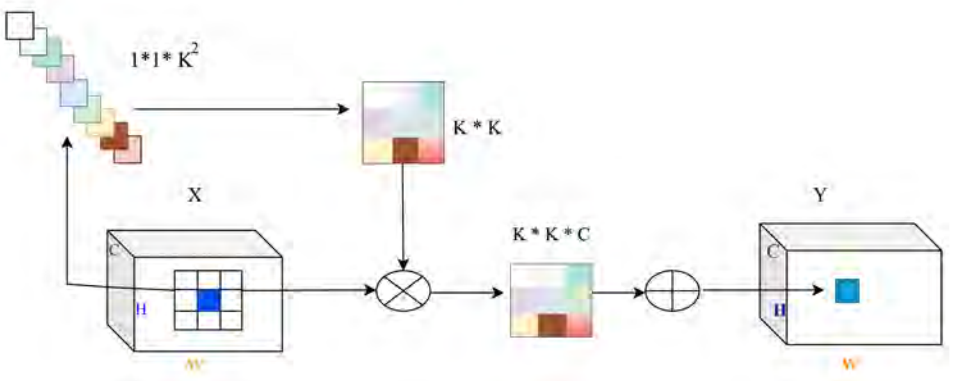


Figure 1.4: Architecture of INN

1.4.3 EANet (External Attention Transformer)

The full form of EANet is External Attention Transformer which is a lightweight CNN model [18]. It is more accurate than other algorithms for predicting trajectories, and its prediction error is also better it can also help the model achieve a lower prediction error that is less than the initial outcome and rapidly decrease the model's prediction error while maintaining the capacity for future learning [34]. This model demonstrates that concentrating on the edge of image segmentation features may enhance the quality by effectively separating buildings from aerial photos [47]. To analyse the incoming visuals and external knowledge concurrently, EANet integrates multi-head attention [26]. EANet is based on two external, teachable, small, shared memories and uses the external attention method [26]. Patches containing redundant and unneeded information are eliminated using EANet to improve the speed and efficiency of computation [26]. Two normalisation layers and two cascaded linear layers are used to implement external attention [26]. To process these incoming pictures and external knowledge concurrently, it integrates multi-head attention. Semantic fusion is then used to merge the information from the two sources [26].

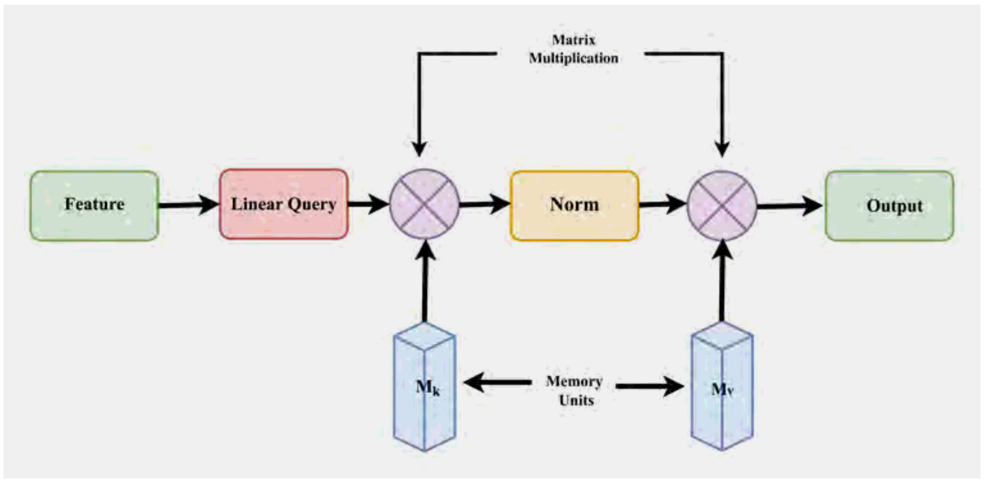


Figure 1.5: Architecture of EANet

1.4.4 Swin Transformer

Swin Transformer is a transformer architecture variant aimed at image processing efficiency. It captures both local and global context in images using a hierarchical approach with shifting windows, making it flexible for computer vision tasks. Swin Transformer is adept at dealing with both local and global contexts, making it an excellent candidate for recognising potholes in broader road sceneries. In theory, it should be able to locate potholes within large photos efficiently. The architecture consists of a total of 4 stages by connecting the patch merging regions and multiple blocks of Swin Transformer. These blocks are the backbone of the architecture [56].

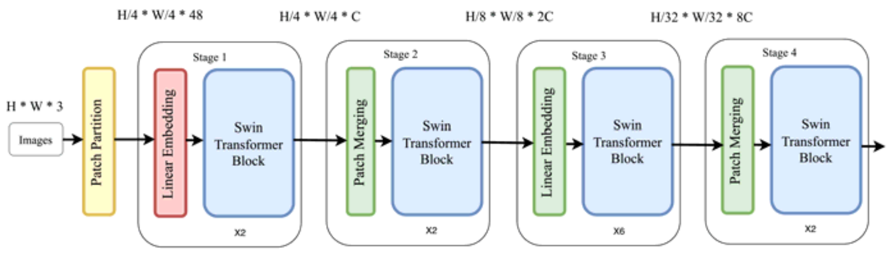


Figure 1.6: Architecture of Swin Transformer

1.4.5 ConvMixer

ConvMixer is a neural network architecture that combines channel mixing and depth-wise convolutions. It is helpful for a variety of image processing applications, including classification, since it makes the construction of convolutional networks simpler while keeping excellent performance. Applications where speed is essential, including the rapid and effective detection of potholes, are suitable for its use. It has a simple architecture with a patch embedding stage followed by some repeated convolutional blocks. The patch embedding process summarises the p x p into the embedded vector which has e dimensions and finally, the image shape was changed into h×n/p× n/p shape. There are repeated ConvMixer blocks that replaced the pooling and convolutional of CNNs and transformers of ViTs [55].

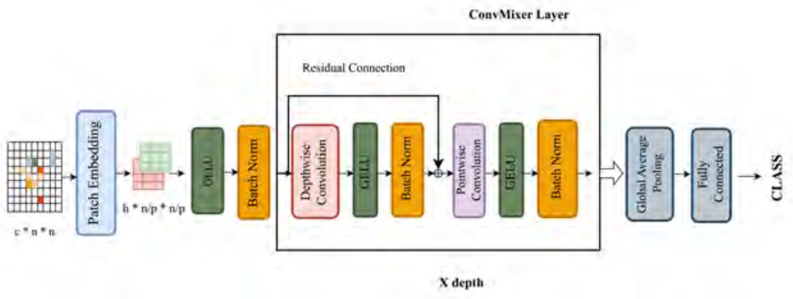


Figure 1.7: Architecture of Convmixer

1.4.6 Compact Convolutional Transformers (CCT)

CCTs are minimal versions of the Transformer architecture. They seek to deliver competitive performance yet require less computational power. They perform computer vision tasks smoothly even under resource-constrained settings. The main architecture is used on large scales when there is a lot of data for training. There are 6 stages of the architecture. First, the images were reshaped coming from the convolutional and patching blocks. After that, it goes through the transformer encoder, sequence pooling, MLP Head, and finally the driven class. In this architecture, embedding is the easiest part. Patches of equal sizes are used to build the architecture. In order to create an image that resembles a chain of words, each patch must be coded and fed into the ordinary transformer architecture [59].

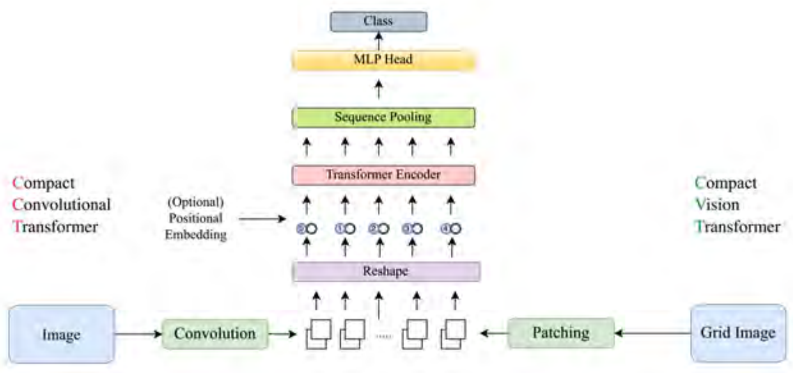

Class
MLP Head
Sequence Pooling
Transformer Encoder
Compact Convolutional Transformer
(Optional) Positional Embedding
Compact Vision Transformer
Reshape
Image
Convolution
Patching
Grid Image

Figure 1.8: Architecture of CCT

1.4.7 DenseNet201

DenseNet-201 is a 201-layered deep convolutional neural network architecture [46]. In this architecture, every layer has a feed-forward connection to every other layer

[44]. Each layer uses its own feature maps as inputs into all following layers while the feature maps of all previous levels are utilised as inputs for each layer [44]. DenseNet-201 is renowned for its complex architecture, which uses dense connection patterns to improve gradient flow and feature propagation across the network [45]. DenseNet-201 can capture intricate connections between features, thanks to its densely linked layers, which makes it appropriate for jobs demanding fine-grained classification [45]. Using a variety of datasets, including ImageNet and CIFAR-10, the DenseNet 201 has demonstrated impressive performance [6]. Due to the architecture of DenseNet201, which considers feature maps to be the global state of the network, performs brilliantly even at slower growth rates [6]. When compared to certain well-known deep transfer learning models, comparative evaluations show that the CNN based on DenseNet 201 performs significantly better [6].

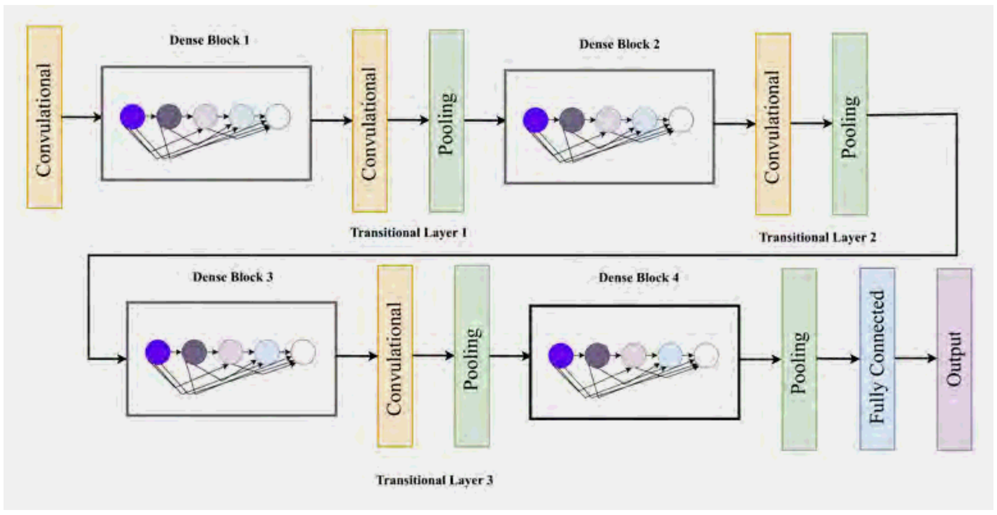


Figure 1.9: Architecture of DenseNet201

1.4.8 ResNet50

There are many variants of ResNet or Residual Network and ResNet50 is one of them. It has a total of 50 layers. It consists of two layers: an average pool layer and a max pool layer and the rest of them are convolutional layers. Three convolution layers make up each convolution block also there are three convolution layers in each identification block [36]. The previous 34-layer ResNet variant is customised in 50 layers by replacing 34-layer net's every 2-layer block with the bottleneck block of 3 layers [43]. It changed the single linear structure to create a better transmission channel. ResNet50 is broadly used for larger vision datasets. Moreover, using this model, the accuracy of the training dataset is not compromised [52]. This model has more than 23 million parameters and all of that can be trained. ResNet-50 is an extremely improved model among all the ResNet variants.

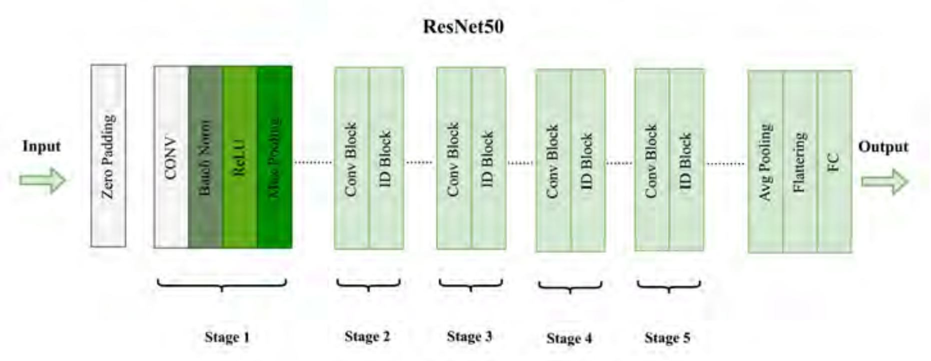


Figure 1.10: Architecture of ResNet50

1.4.9 VGG-16

The convolutional neural networks are extensively used for image recognition. One of them is VGG-16. It has a total of 16 layers. Three of the layers are fully connected and the other 13 layers are convolutional. It has a modular structure. The fully connected layers include max pooling. It lessens the total volume size. VGG-16 requires a fixed input size that is 224 x 224 RGB image. If this network is pre-trained properly, this can classify up to 1000 object categories. Thus, With a huge number of photos the network has learnt large representations of features to improve the detection accuracy, VGG-16 increases the network depth by stacking the convolutional layers [52]. The depth of the architecture increases with inner convolution layers. Nonetheless, the main attraction of this architecture is its simplicity [60]. There are repeated convolutional layers in the architecture of VGG-16. It has very small 3 x 3 filters throughout the whole structure and these looped to the input layer at every pixel. Additionally, several of the Convolutional layers are followed by five max-pooling layers, and finally, convolutional layers are followed by three fully connected layers [37].

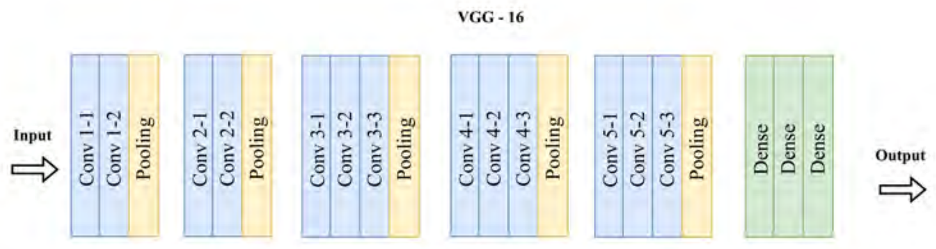


Figure 1.11: Architecture of VGG-16

1.4.10 Xception

A deep convolutional neural network architecture, Xception involves Depthwise Separable Convolutions. This architecture has three stages entry, middle, and exit flow [61]. The data first enters the entry flow then goes to the middle flow and repeats it eight times after that the data goes to the exit flow [61]. Xception provides an architecture consisting of Maxpooling + Depthwise Separable Convolution blocks connected via shortcuts similar to ResNet implementations [61]. Inspired by Google's

Inception model, a deep CNN with 71 layers also an extreme interpretation of the Inception model serves as the foundation for the Xception model [17]. With less processing power needed, Xception seeks to be a better version of Inception [53]. InceptionV3 and Xception both have around the same amount of parameters [36].

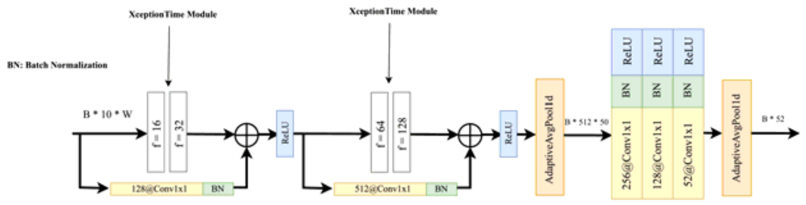


Figure 1.12: Architecture of Xception

1.5 Thesis Organization

The thesis is further organised into the following segments- Literature Review, Research Methodology, Dataset, Performance Evaluation, Discussion, Research Limitation, Conclusion and Future Work. In the Literature Review chapter, we have discussed the existing studies on pothole detection using machine learning models and papers on dataset collection methodology and augmentation techniques. Furthermore, the Dataset chapter is divided into three categories where we have talked about our dataset collection procedure, dataset augmentation method and the dataset model input pseudocode. In the performance evaluation chapter, we have mentioned our detailed test bed setup and included all the experimental results. Under experimental results, we have included performance analysis of all tested models for both the original and augmented datasets. Along with that, we have also included a performance analysis of each individual model under the experimental results. In the Discussion chapter, we have included our overall research findings and a comparative analysis table that compares our research with the existing literature. Furthermore, in the Research Limitation chapter, we have talked about the limiting factors of the present thesis. Lastly, in the Conclusion and Future Work chapter, we have expressed an overall conclusion of our study and discussed regarding possible future improvements of our research.

Chapter 2

Literature Review

Various authors published different study articles on the same that topic we are doing and some of them helped our study. So, reviews of them are provided below.

Asad et al., 2022 have proposed a real-time pothole detection system in their paper. The proposed system uses OAK-D, Tiny-YOLOv4, and Raspberry Pi for real-time pothole detection. SSD and YOLO are two of the best models of object detection, to get a better result the performance of the Tiny-YOLOv4 is compared to YOLOv5, YOLOv4, YOLOv3, YOLOv2, YOLOv1, and SSD-Mobilenetv2. Comparing the results these models are unable to detect potholes at long distances and also cannot detect the small ones where the potholes are detected with almost 96% accuracy with Tiny-YOLOv4. Since the Raspberry Pi serves as the OpenCV AI Kit's host computer, it will be positioned in the middle of the car to capture the most amount of space. When it comes to real-time pothole detection, Tiny-YOLOv4 outperforms YOLOv2, YOLOv3, YOLOv4, and SSD-Mobilenetv2 in terms of both maximum FPS and detection rate, making it the top-performing model. The suggested method enables road maintenance authorities to swiftly and efficiently plan actions for repairing road infrastructure [4].

Ahmed et al., 2021 have shown the crucial problem of pothole detection in transportation infrastructure in their paper. This automated pothole-detecting method would assist in locating potholes and alert the drivers to them. This study has created an effective deep learning convolutional neural network (CNN) and a modified VGG16 (MVGG16) network so that the potholes can be identified in real-time, lower the computing cost, and enhance training outcomes. The Faster R-CNN in this research uses the MVGG16 as its backbone network. The object detectors using deep learning make use of two methods. Furthermore employed are one- and two-stage detectors. The two-stage detector Faster R-CNN is a good illustration of this. The region proposal network (RPN) provides region proposals that suggest bounding boxes with the likelihood of including an item in the first stage of the two-stage detector. During the second stage's RoI pooling process, the limits of space produced by the bounding-box regression task and the RPN for classification are extracted. The detectors with one stage include SSD and YOLO. Without a region proposal phase, these regression models concurrently estimate bounding boxes and classification probabilities. For the dataset, the author assembled images of potholes from several sources as well as mounting smartphone cameras on the windscreens of moving automobiles to obtain additional images from the roads in Carbondale. A total of 2139 potholes were present in the 665 photos used for training. In addition, this study compared the performance of YOLOv5 with that of the ResNet101 backbone and Faster R-CNN. Windows 10 was operating on the system used for training; 0.00036 for YOLOv5 was used, but none for Faster R-CNN. It demonstrates that YOLOv5 required 1200 epochs to converge, but Faster R-CNN only needed 100 epochs. The author of this research builds 10 CNN models, including YOLOv5 variants, two YOLOR variations, and Faster R-CNN with a different backbone, with the goal of accurately and quickly detecting objects. The investigation reveals that faster R-CNN ResNet50 has 91.9% greater accuracy than MobileNetV2, which was last, followed by Yl, Ym, and the planned MVGG16 [10].

Al-Shaghouri et al., 2021 have shown the crucial problem of potholes on roads and their impact on safety and functionality in their paper. This study has implemented and analysed multiple deep-learning architectures, to identify potholes. In this study, the authors use two datasets. The first database consists of 431 various images, available online. The second dataset consists of a blend of internet sources and roadside photographs. First, a cell phone attached to the windshield of the car captures several images of the potholes. Also downloaded pothole images extended the variability and the amount of the database. Several object detection methods are utilised for identifying potholes in road images. The authors examine the pothole detection outcomes of multiple real-time deep learning algorithms, including SSDTensorFlow, YOLOv3-Darknet53, and YOLOv4-CSPDarknet53. Among all the algorithms YOLOv4 demonstrated to be an effective object detection architecture. YOLOv4 stands out with an impressive 81% recall, 85% accuracy, and 85.39% mean Average accuracy (mAP). The SSD-TensorFlow pothole detector processing speed was too slow for real-time applications as a 73% precision was achieved, a 37.5% recall, and a mean Average Precision (mAP) of 32.5% [16].

Wo et al., 2020 proposed an automated road-monitoring system that will detect potholes using smartphones in their paper. This detection system uses the vibration sensor and the GPS receivers in smartphones. This proposed system has 4 steps, data acquisition is the first step. For data acquisition, the built-in accelerometer in smartphones is used to capture the shock caused by potholes, while the GPS chip is utilised to tag the location and store the information on a digital map. In the second step data is processed in 5 steps which are resampling, reorientation, filtering, labelling, and segmentation. After processing data features are extracted from the dataset. Traditional classifiers like Logistic regression (LR), SVM, and Random forest (RF) are used in the classification step. The performance will be evaluated using these three classifiers which will be applied to the extracted features from various domains of the vibration signal. Logistic regression (LR) is a linear model that maps continuous input variables to binary output values. On the other hand, SVM identifies a hyperplane with a maximum margin, and can also transform data points into a higher-dimensional space, enabling the possibility of nonlinear separation. It makes nonlinear separation possible by mapping data points to a higher dimensional space. Random forest (RF) is extensively employed in machine learning and leverages swarm intelligence to combine individual predictions from multiple decision trees to determine the ultimate classification. So, this study utilises the RF classifier. Also, the suggested approach utilises datasets derived from various road categories to examine their overall applicability and robustness [9].

Kim et al., 2022 in their paper offered an in-depth analysis of a variety of automated approaches for spotting potholes on roads. For comparison, three different methods are discussed in this paper which are the vision-based, the vibration-based and the 3D reconstruction-based methods. The first method employs images or videos as its input and verifies whether there are potholes or not. Additionally, this method utilises both image processing and advanced deep learning technology. Feature extraction and the subsequent training and testing phases are crucial components in vision-based techniques. This method employs image-processing technologies like edge detection and SIFT for feature extraction. During the training and testing phases of these approaches, deep learning technologies including CNN, YOLO, and SVM are utilised. By examining data from the vehicle's acceleration sensor, the vibration-based technique determines the presence of potholes and calculates their depth. This approach involves three processes, data preprocessing, feature extraction, and classification. Signal processing methods like filtering, Fourier transformation, and correlation are employed during data preprocessing and feature extraction. Classification, on the other hand, makes use of machine learning methods like K-nearest neighbour, linear regression, and random forest. The stereo-vision technology used in the 3D reconstruction-based technique predicts the geometry of potholes and determines their volume. Using deep learning techniques including filtering, PCA, and a customised version of the U-Net, this strategy comprises data processing, training, and testing. The effectiveness of these models proposed using these three techniques involves evaluating their performance in terms of accuracy, precision, and recall. Within these 3 models, the 3D reconstruction-based approach has the most accuracy [20].

Gayatri K et al., 2021 discussed the significance of vehicle detection and pothole identification in the context of automatic driving and traffic surveillance in their paper. It highlights how poor road conditions, particularly potholes, can lead to traffic accidents and vehicle damage, emphasising the need for effective detection methods. The suggested method uses picture data to identify automobiles and potholes using deep learning models like the Inception Network V2 and Faster Region-Based Convolutional Neural Network. The Faster R-CNN model's performance is contrasted with that of two other well-known algorithms, SSD and YOLO, in this work. Accuracy was the main assessment criteria employed throughout the research. The results indicate that the proposed Faster R-CNN model outperforms SSD and YOLO, showing a notable 5% improvement in accuracy compared to these existing methods. The paper acknowledges the computational demands of SSD and YOLO, particularly during training, making them less practical for some applications. The article also provides a comprehensive review of related work in the field of traffic management and vehicle detection. Various studies, such as those focused on traffic density estimation and traffic light control through image processing, are discussed. Additionally, research using accelerometers, support vector machines, and neural networks for pothole detection is mentioned, demonstrating the diverse approaches taken in addressing similar challenges. The proposed methodology for vehicle and pothole detection involves transfer learning, which is explained as the process of training a neural network model on a related problem and fine-tuning it for the specific task. This approach is highlighted as an efficient way to save time in model development. The Faster R-CNN model is detailed in terms of its architecture and its combination with region proposal networks to create Faster R-CNN. It is explained that the key difference between Faster R-CNN and Fast R-CNN lies in the region proposal method. Finally, the article emphasises the importance of performance evaluation metrics such as precision, recall, IoU, AP, and mAP in assessing the effectiveness of object detection models that ensure the proposed model's success is measured comprehensively [7].

Park et al., 2021 addressed the crucial issue of pothole detection in road maintenance in their paper. Pothole repair is a paramount task in ensuring road safety and reducing accidents, making effective road surface monitoring essential. Traditional pothole detection methods rely on manual image processing, which takes a lot of time and effort. Computer vision presents an opportunity to automate this process, allowing for the efficient identification of potholes from a collection of images. The authors of this paper investigate pothole identification using many YOLO models, including YOLOv5s, YOLOv4-tiny, and YOLOv4. The authors develop a dataset of 665 720 by 720-pixel photographs that capture various kinds of potholes on various sorts of road surfaces. To assess the effectiveness of the models, they separate the dataset into testing, training, and validation subsets. Mean average accuracy at a 50% Intersection-over-Union criterion is the main assessment statistic employed. The results reveal that YOLOv4-tiny outperforms the other models with a mAP_0.5 of 78.7%, followed by YOLOv4 at 77.7% and YOLOv5s at 74.8%. The study demonstrates that YOLOv4-tiny is the most suitable model for real-time pothole detection, balancing accuracy and computational efficiency. The paper highlights the importance of object detection in computer vision applications and offers a thorough overview of the state of object detection and classification today. It highlights the evolution of YOLO architectures, from YOLOv1 to YOLOv5, and how they have revolutionised real-time object detection with their speed and accuracy. This also discusses the challenges in road maintenance and the importance of accurate pothole detection, particularly in reducing road accidents caused by poor road conditions. They emphasise the advantages of using deep learning and CNNs for object detection, highlighting YOLO's ability to respond in real-time. The dataset creation process is explained, including the labelling of potholes within the images by expert groups, ensuring accurate ground truth annotations. The authors detail the training process for YOLOv4, YOLOv4-tiny, and YOLOv5s, with YOLOv4-tiny showcasing faster convergence and stability during training. It concludes by noting the significance of identifying the most fitted pothole detection model, which can significantly contribute to road maintenance and safety. Future research directions are suggested, such as further extending the network architecture for improved accuracy and automating labelling strategies to enhance efficiency. Ultimately, this work contributes to more effective and precise road repair practises by addressing an important real-world issue and offering insightful information about the use of YOLO models for pothole detection [13].

Tahir et al., 2023 conducted a study in their paper that presents the critical issue of pothole detection in the context of self-driving vehicles and intelligent transportation systems. The introduction sets the stage by highlighting the significance of self-driving cars in reducing accidents and traffic while emphasising the role of AI in these vehicles' functioning. It also underscores the importance of pothole detection, as these road imperfections can disrupt the vehicle's perception systems. Following that, the article describes the prerequisites of AI systems for pothole detection, highlighting the necessity for compact distributed neural networks, top-notch training data, and dependable communication. Following the introduction, a description of distributed deep learning as a dynamic research field follows. It highlights recent advancements in data and model parallelism, communication strategies, and federated learning, all of which contribute to more efficient and scalable distributed deep learning systems. This prepares the reader for the paper's major comparison of the distributed TensorFlow and PyTorch APIs for pothole identification in autonomous vehicles. The paper introduces the two main distributed learning strategies: data parallelism and model parallelism, explaining their relevance and use cases. It emphasises that TensorFlow and PyTorch are the primary libraries supporting distributed learning, with PyTorch gaining popularity for its flexibility and robustness in model and data distribution. The authors then acknowledge the challenge posed by the ever-expanding datasets and models, which outpace computational capabilities. They express the need for efficient distributed training approaches, leading to the study's technical contributions. The technical contributions of the study are outlined, including the development of a pipeline that is hybrid which combines both model parallelism and data, the use of an edge cluster testbed for performance evaluation, and a comparison of the hybrid approach that is proposed with PyTorch and TensorFlow APIs. These contributions aim to address the challenges of distributed pothole detection in resource-constrained edge computing environments. The study is divided into various sections, which include a thorough comparison of the distributed APIs of PyTorch and TensorFlow, the suggested pipeline that is hybrid distributed, experimental data sources, and in-depth findings and debates. Each section provides in-depth insights into the methodology and findings of the study. The authors present a dataset of pothole images, emphasising its diversity and relevance for real-world scenarios faced by autonomous vehicles. They explain how the dataset was prepared and annotated, setting the stage for the experimental evaluation of the proposed hybrid approach. In the experimental section, the authors conduct a rigorous evaluation of their approach, considering various aspects such as training time, model accuracy, and detection quality. They also extend their evaluation to include other road distress conditions, showcasing the versatility of their proposed methodology. In summarising the main conclusions, the report highlights the hybrid distributed pipeline's potential to overcome the difficulties associated with pothole detection in self-driving automobiles. It also suggests the possibility of extending the approach to videographic analysis in smart cities [30].

Xin et al., 2023 have proposed a system in their paper that can detect potholes or cracks on the road using a smartphone accelerometer, GPS, and camera. Accelerometer and GPS data are collected from a mobile app built by the team and video of road conditions are taken with a phone camera. While acceleration-based identification is used in low light, rainy, or cloudy settings, video data, and acceleration sensors are fused to implement detections in lighted, clear surroundings. The results are then refined using geographical density-based clustering to get more accurate data. A confusion matrix is used to assess the model's performance. This method has been found to enhance accuracy by 6% in comparison to traditional methods that utilise image recognition or acceleration sequence classification methods. The traditional method's results are greatly affected by ambient lighting, unlike the proposed model [33].

Tamagusko et al., 2023 examined the state-of-the-art computer vision (CV) algorithms for identifying pavement potholes in their paper in order to compare the effectiveness of different deep learning (DL) models. This study uses 665 road photos with labelled potholes, on which the YOLO (You Only Look Once) v3, v4, and v5 deep learning models were deployed. While YOLOv4 offers the greatest MAP (Mean Average Precision of 83.2%) and YOLOv4-tiny obtains the best interference time, making it suited for mobile or low computational power devices, YOLOv5s proves to be the easiest to develop and scale. YOLOv5 results are found to be better after doing some tuning. Its py-torch-based implementation also enhances usability. The outcomes can be used to boost cost-effectiveness, road safety, and management of traffic [31].

Hoe et al., 2023 suggested a model to use 2D pothole photos for real-time pothole detection and risk classification in thair paper which is called SPFPN-YOLOv4 tiny. The study provides more accurate results compared to YOLOv2, v3, and v4 tiny. The feature pyramid network, CSPDarknet53-tiny, and spatial pyramid pooling are combined to build the model. Using horizontal flip, gamma regulation, and scaling, 2665 datasets were gathered and enlarged. 10% was utilised for testing, 20% for validation, and 70% for training. In comparison to YOLOv2 and v3, the SPFPN-YOLOv4 small has faster implementation times and the greatest MAP (Mean Average Precision) values. In the study, a distance equation and pinhole cameras were used to determine the 3D dimensions of potholes. The proposed algorithm turns out to be suitable for extracting rich features from pictures in a low computing power device, unlike traditional models [25].

Lee et al., 2022 proposed a model in their paper to improve the accuracy and reliability of pothole detection models by adding variables into consideration such as temperature, humidity, traffic volume, and precipitation. 12,000 images of pavements in various environmental conditions were utilised to create the dataset. Training involved 8000 images, validation involved 2000 images and the test results were implemented on the other 2000 images. Multilinear regression was selected as the optimal pothole detection algorithm in the present test. Linear regression showed the worst results. The accuracy of the model came out to be 95% upon 2000 iterations [28].

Paullada et al., 2021 surveyed in their paper about the recent issues of data handling in machine learning and focus their work on Natural Language Processing(NLP) and computer vision. The survey is structured into three themes inclusive of Dataset design and development, Dataset in(tro)spection and Dataset culture. In the first theme, Dataset design and development, they deal with the critical reviews of the design of the dataset that is used as a benchmark inclusive of studies such as auditing existing datasets for biases, identifying spurious correlations, analyse the framing of tasks, work promoting and documentation practises. In the second theme, Dataset in(tro)spection, they review approaches that are aimed at improving and exploring numerous aspects of datasets including modelling techniques to mitigate the impact of bias in datasets and augmenting and filtering data and it reveals that these approaches do not address broader issues that are linked to data use. Lastly in the third theme, Dataset culture, their focus is to work on dataset practises inclusive of critiques of the datasets that are used as performance targets, their viewpoint on data management and reuse and look into the papers of legal issues pertaining to data collection and distribution [14].

Whang et al., 2023 emphasised the challenges in their paper which were faced during model training and testing while data quality is not as up to the mark. Moreover, there is a huge importance of copious data for deep learning procedures in different approaches. Not only this, the study also emphasises the unwillingness of different industries to approach deep learning because of the imperfectness of different models. The deep inception of different data collection methods, labelling, and improving models, stretch both the contribution of ML and data management communities. Moreover, the study captures the flaws of existing techniques of data cleaning and also acknowledges the significant literature on data cleaning. Also, the reasons for data poisoning and how to overcome the situation are discussed vastly in the paper. One of the new research branches called data sanitisation is established to defend against the various attacks on data which can affect the model accuracy vastly. But still, there is a risk in model training when dealing with imperfect data even after the data cleaning and data validation. In this part of the study, it recognises that the data can be wholly purified. Furthermore, the study underlines the fairness concerns not previously noticed in the age of responsible AI. It explains that while several groups have historically studied robust and equitable model training, data cleaning, validation, and data collection, Combining these efforts within a holistic framework is essential for advancing data-centric AI [32].

UCAR et al., 2022 focus on different methods of data augmentation on convolutional neural networks in their paper. There are challenges in certain classes of limited data availability which are addressed in the study. To increase the size of the dataset for six classes in the cases of limited images, three well-defined methods have been applied. These are shifting, rotating, and flipping. Using the methods, a fresh dataset is created by reshaping the old one. Moreover, the study addressed different misleading images occurring during the data augmentation process and the way of avoiding such obstacles with a new dataset. Regarding that, a total of fifteen CNNs were trained and tested and the results are magnificent as it improved by 5% overall. Again, It is done for both the original datasets and augmented datasets. A better classification rate is achieved by the proper process if done correctly. Also, it represents the advantage of DA in increasing the datasets. Thus, the whole study signifies the absolute success of data augmentation even with erroneous images and giving better output than before. The entire study makes a strong argument for DA's beneficial impact on categorisation performance [22].

Shijie et al., 2017 explore in their paper, data augmentation methods in the context of image classification, emphasising both unsupervised and supervised approaches in particular. Category-free transformations like flipping, rotation, cropping, shifting, colour jittering, noise addition, and PCA jittering are examples of unsupervised data augmentation. These techniques seek to produce a variety of training examples without taking data labels into account. Conversely, label-related methods are used in supervised data augmentation; notable examples of these methods are Generative Adversarial Networks (GANs) and their variations. The study explores the GAN model, which consists of a discriminative model (D) and a generative model (G), describing the objective function and training procedure. It shows how training instability can be addressed with Wasserstein GAN. Next, the focus of the study shifts to the fundamentals of convolutional neural networks, or CNNs, and how they developed from the Neocognitron to more recent models like AlexNet. The study examines the effects of data augmentation on picture classification tasks using the traditional CNN model, AlexNet. The impact of data augmentation on picture classification tasks is examined in this research using the traditional CNN model, AlexNet. We introduce the AlexNet architecture, which includes fully connected, pooling, and convolutional layers. TensorFlow 1.0 is the deep learning platform, Linux Ubuntu 14.04 is the operating system, Python 3.5 is the development language, and an Intel Core i7-6700 CPU with 16GB of RAM is all used in the paper's setup. Two datasets with different training set sizes—CIFAR10 and a part of ImageNet—are used for the experiments. Different kinds of supervised and unsupervised data augmentation techniques are used, such as WGAN, flipping, rotation, cropping, and combinations of these. The results show that models trained on augmented datasets outperform models trained on non-augmented datasets in general, and the improvement becomes more obvious with larger augmentation sample sizes. Additionally, the study shows that several combinations of augmentation techniques—like Flipping+Cropping and Flipping+WGAN—perform better than the individual techniques. We investigate how the size of training sets affects the effectiveness of augmentation, finding that augmentation is more effective for smaller training sets. The study claims that more investigation is required to examine the effects of data augmentation in more general settings, such as with bigger datasets, complex network models, and class imbalance adjustment. In conclusion, the experimental results offer insightful information on how well WGAN, flipping, rotation, and cropping augmentation techniques can improve the performance of deep convolutional neural networks in image classification tasks [3].

Wong et al., 2016 in their paper explores the impact of data augmentation in both data-space and feature-space on the performance of machine learning classifiers, utilising the MNIST database. This study employs two primary data augmentation techniques: data warping and synthetic oversampling. In the data-space augmentation, the authors implement elastic deformations, including affine transformations and introduce novel elastic deformations with adjustable parameters like displacement strength (σ) and smoothness (σ). The experiments say that large deformations ($\alpha \geq 8$ pixels) occasionally compromise label integrity, leading to characters difficult for human recognition. The authors empirically set $\sigma = 1.2$ pixels with $\sigma = 20$ for optimal performance using the CELM algorithm. For feature-space augmentation, the Synthetic Minority Over-Sampling Technique (SMOTE) and Density Based SMOTE (DBSMOTE) are employed to address class imbalance. However, DBSMOTE is found to potentially increase overfitting. The study establishes baseline performance figures by reducing real training data, comparing them with the performance of an equivalent amount of augmented data. The experimental system employs a two-stage architecture to assess the impact of data augmentation in both data-space and feature-space on machine learning classifiers, focusing on convolutional neural networks (CNN), support vector machines (SVM), and extreme learning machines (ELM). The experiments, conducted on handwritten digit recognition using convolutional neural networks (CNN), support vector machines (CSVM), and extreme learning machines (CELM), reveal that augmentation in data-space, specifically using elastic distortions, outperforms feature-space augmentation. The study begins with a baseline experiment, manipulating the number of real and synthetic data samples for training each classifier. Results demonstrate that increasing the number of real samples reduces overfitting, with CNN benefiting the most. Data-space augmentation using elastic deformations outshines feature-space augmentation for CNN. The paper provides detailed error percentage data and insightful observations regarding the effectiveness of different augmentation techniques for each classifier on the MNIST dataset. In conclusion, this paper investigates the efficacy of data augmentation in data-space and feature-space, employing a two-stage classification architecture [2].

Pandey et al., 2022 in their paper presented a Convolutional Neural Network application on accelerometer data for pothole detection. The findings of this experiment signify the importance and advantages of the CNN approach with great accuracy and computational complexity while detecting potholes. In order to identify the road defects this study proposes a huge data collection approach that will be utilised by Convolutional Neural Networks and standard smartphone sensors. Data collection and data processing are the first two steps of the proposed method. Sensors Pro application of the iOS store has been used in this research for collecting 3-dimensional accelerometer data including GPS information and the timestamp. Data preprocessing is the prerequisite of any research for attaining meaningful and satisfactory results. This study has broken down data processing into three stages which are augmentation, resampling, and labelling. The goal of this study is to process raw data using CNN since the data has been obtained from a smartphone without applying any data processing technique. Several combinations of hyperparameters were tested in simulation using the processed dataset. Concluding the experimental results CNN approach shows significant accuracy [21].

Bhutad et al., 2022 in their paper described how data has been captured and processed for machine learning models. This dataset includes pothole images of different roads in different weather and the article includes a description of how the data has been captured, which devices have been used, the ratio of the images, and other data-related information. Since machine learning models are trained on datasets collected in certain environments and weather they can not always provide accurate results, the goal of this dataset is to help provide accurate results in summer and rainy seasons. The dataset contains road images of both paved and unpaved roads in different folders with subfolders containing raw, rotated, and final images. It contains 10 videos and 8484 images which are formatted as MP4, and .jpg respectively also these videos and images are captured from the side view as well as the top view. Since the dataset is of road surface potholes, all roadside barriers and speed breakers are included in the dataset. Samsung Galaxy A22 RGB Quad camera was used to capture the images and all the images were resized into 512 $\times$ 512 dimensions. The study will help identify different road surfaces not only in two different weather but also in paved and unpaved roads which will improve the machine learning models much more [19].

Egaji et al., 2021 in their paper compares five different models for classification for detecting potholes. Logistic Regression, Random Forest Tree, Naive Bayes, K-Nearest Neighbour (KNN), and Support Vector Machine (SVM) models are compared in this paper for a relative study on machine learning models for pothole detection. Data collection and labelling is the primary stage of this project. As a result, two custom apps are created: the first app records the GPS data, accelerometer, and gyroscope on the other hand the second app is used for data labelling. As for labelling and data collection three Android phones have been used which are Motorola G7, Nokia 3.1, and Nokia 5.1. The first app was attached at the centre of the car's dashboard using one phone while on the other hand, the second app was used by a passenger to press and hold the blue button whenever the car approached a pothole. Multiple routes, cars, and Android devices were the source of data collection for data pre-processing, a 2-second non-overlapping moving window was used for training. On the other hand from the validation and training dataset, the test dataset was isolated. Since high-frequency data was collected it resulted in some noisy data points. Therefore, the data was grouped into two parts, one part was aggregated statistical features for each interval chunk and the other part was

2-second interval chunks. The dataset processing was one of the main tasks of this project since a lot of things depend on the dataset. After comparing all five models KNN and The Random Forest showed the best performance [11].

Shorten et al., 2019 in their paper conducted a study where the importance of data augmentation on limited data for building better deep-learning models. Data augmentation improves the training dataset of deep-learning models by size and quality. This study briefly explains ten different augmentation techniques, how they work, experimental results, and disadvantages. The ten augmentation techniques are Kernal filters, Adversarial training, Colour Space augmentation, Mixing images, Meta-learning schemes, Geometric transformation, Random erasing, Neural style transfer, GAN-based augmentation, and Feature space augmentation. Each of the augmentation algorithms is detailed demonstrating its operational mechanism, experimental outcomes, and associated limitations. Geometric transformations one of the categories of augmentation in particular are based on fundamental image manipulation which includes translation, rotation, cropping, and flipping. Each of these carries distinct implications for label preservation. Geometric transformation appeared as a powerful solution that addresses positional biases in training data and offers ease of implementation yet introduces potential label alternations and additional computational costs. On the other hand, Colour space transformations offer creative solutions in image recognition for overcoming lighting challenges that also involve modification in RGB channels. Kernel filters are mainly explored for their potential in data augmentation and their similarity to the Convolutional Neural Network mechanism but typically it is explored for blurring or sharpening images. Non-traditional methods like combining photos by averaging pixel values expose illogical yet useful enhancement techniques. One method that shows significant error rate reductions, particularly in limited data applications, is Sample Pairing, which combines randomly cropped and flipped pictures. Dropout regularisation algorithms serve as the inspiration for random erasing, which forces the model to focus on full pictures rather than individual visual components to address occlusion issues in image identification. Researchers have found that augmenting data at test time is much more effective than augmenting training data. The speed of the model can be limited due to computational cost depending on how the augmentation has performed. In real-time prediction, it can be a very costly obstacle. However, in medical image diagnosis, it is a very promising practise. The test time augmentation has a great impact on classification accuracy. It is another method of measuring the toughness of a classifier. Some classification models increase the speed of their necessity by depending on test-time augmentation. For One-Shot learning systems like Facenet, Curriculum learning decisions are very important. It is not constrained by limited data but applies to all Deep Learning models. Resolution has a great impact on data augmentation the more resolution the more requirement of memory and processing time. But with Super-Resolution Generative Adversarial Networks and Super Resolution Convolutional Neural Networks images with higher resolution would result in better models. GANs resolution is very important but due to mode collapse and training stability issues higher resolution output is very difficult. If the output resolution gets higher it will be beneficial for data augmentation. The final dataset size is significant for data augmentation. So, image serving can speed up by augmenting images before training. Class imbalance is one of the common problems and one of the solutions is data augmentation. A simple random oversampling using a small geometric transformation would be a great solution to oversampling using data augmentation. Deep learning-based oversampling methods like GANs, meta-learning schemes, Neural Style Transfer, and adversarial training can be also used as a great oversampling method. Data warping and oversampling are two methods to enhance image data, with some enhancing image classifiers while others do not. Understanding the effectiveness of these augmentations and the representations learnt by neural networks remains challenging. Traditional hand-crafted augmentation techniques like cropping and colour space alteration are being enhanced with GANs, Neural Style Transfer, and meta-learning search algorithms. Image-to-image translation has numerous potential uses in Data Augmentation, with Neural Style Transfer being a more powerful technique than traditional colour space augmentations. Both methods can be combined for enhanced performance. These augmentation methods are unique in their ability to be combined. The initial, limited dataset has an intrinsic bias, and existing augmentation techniques cannot correct the poor diversity of the testing data. This survey focuses on image data applications but can be applied to other data domains. Data augmentation cannot eliminate all biases in small datasets. To sum up, data augmentation makes small datasets more like big data, which helps to avoid overfitting [5].

Deepika et al., 2023 conducted a deep-learning approach for identifying road hazards in their paper. For recognising the road hazards this paper has approached a model with two steps. The goal of this paper is to offer a strong solution so that road safety can be improved. Also, it offers a practise of efficient road maintenance. For this reason, two of the deep learning models have been used so that they can detect road hazards properly in real-time and give accurate classification and those are YOLOv8 and ResNet50. Among these deep learning models ResNet50 has shown better performance in accurate classification with 95% accuracy on the other hand YOLOv8 has shown better performance in detection [49].

Parasnis et al., 2023 proposes a unique pothole detection model that uses image processing and deep learning techniques in their paper. The proposed model of this study extracts features from the VGG16 which is transferred to the unique custom Siamese network architecture which is much more efficient. The suggested model is inexpensive because it uses fewer parameters and training data, and it is computationally efficient since the results it produces are comparable and better than the state-of-the-art. Two of the well-known metrics used to measure the effectiveness of the pothole detection system are the EER (Equal Error Rate) and the AUPR (Area Under Precision-Recall) curve. In terms of performance metrics, the proposed model has outperformed the previous state-of-the-art works with a testing accuracy of 96.12%, EER of 3.89%, AUROC of 0.988, and AUPR of 0.985 values. It is clear from the study that the network presented in the research is optimum. It yields as few as 372,448 parameters, but it also produces excellent and successful outcomes that reinforce its position as a reliable state-of-the-art technology. Our network's capacity to develop a personalised, lightweight, and cost-effective model through our trials. To sum up, this approach has outperformed unprocessed pictures and basic CNNs in terms of detection performance, having proved effective in classifying both potholes and regular roads [29].

Arjapure et al., 2020 present a study based on the classification and detection of potholes using convolutional neural networks in their paper for detecting potholes on the road. The process is done on a total of 838 images among them 86 percent of them are used for training and 14 percent are for testing. The images were raw images instead of requiring the derivation of different features from the images. The models used in this study are DenseNet201, ResNet152, ResNet50, ResNet50V2, ResNet152V2, InceptionV3, and InceptionResNetV2. Moreover, five convolution layers with a ReLU activation function make up the model. The average accuracy from all the pre-trained models was 87.5% and CNN accuracy was 80.17%. The accuracy from the DenseNet201 and InceptionResNetV2 were very good and it was 89.66%. Three metrics- precision, recall, and accuracy were used to measure the performance of the proposed method. From the comparison of various algorithms, it was really clear which model is efficient for the detection of potholes in the road [50].

Chatterjee et al., 2023 in their research focus on image classification performance with the help of a multi-head channel attention mechanism and transfer learning. For that, the Xception model which was pre-trained with the dataset of ImageNet. The accuracy of the experiment was favourable and it was 96.99%. The future pandemic situations can be dealt with with the help of this study by using artificial intelligence in urgent image analysing tasks. By using Xception Net, the study approaches to capture vital features and improve the overall model's performance. Additionally, to upgrade the accuracy of classification, an attention module was integrated to the model which would automatically recognise the patterns. In the research, transfer learning is used in Xception. Moreover, A multi-head channel attention mechanism was implemented for image classification tasks to increase feature extraction and elevate classification performance. The Xception architecture was used because of its powerful design in resource-constrained schemes and has contributed to advancements in the field of image analysis. The total dataset was split into two halves and 80% of that are used for training and the rest of them are used for testing. The back-propagation and gradient descent were used to update the weight during training and to minimise the loss function. So, the study's findings suggested that the channel attention mechanism together with transfer learning is an efficient method to deal with image classifying problems [24].

Pramanik et al., 2021 used the methodology of Transfer Learning and VGG16 and ResNet50, two conventional neural network models in their paper to identify the potholes on roads. For the experiment, they collected 1490 images which were then divided into 80:20 ratios for training and testing. The resolution of the images was reduced during the preprocessing phase to 224 x 224 pixels for VGG16 and ResNet50, it was reduced to 256 x 256 pixels. The processing of the model begins with data collection. After that, image preprocessing started which was followed by augmentation and finally, ResNet50 and VGG16 were used to pre-trained weights along with feature extraction. The data augmentation process increased the number of images and solved the problem of overshifting. By evaluating the confusion matrix, precision value, accuracy score, specificity, sensitivity, false positive rate, false negative rate and F-1 score, they selected the best model. The accuracies obtained from VGG-16 and Resnet-50 were 96.31% and 98.66% respectively [15].

Chapter 3

Research Methodology

Our study contains a comprehensive performance analysis of pothole classification among existing lightweight and traditional heavyweight models. Firstly, we have created a dataset of 900 image samples which has been collected from the roads of Dhaka and Bogura. The image samples were captured from the following devices- iPhone 11, iPhone XR, iPhone 6s Plus and Xiaomi Redmi Note 12. Additional details regarding the Dataset can be found in the Dataset Chapter of the thesis book. After that, we augmented our dataset by employing methods such as flipping, random brightness, random rotation, skew tilt and shear etc. Furthermore, the augmented dataset having 10,000 image samples and the original dataset having 900 image samples were run into six lightweight and four heavyweight traditional neural network models.

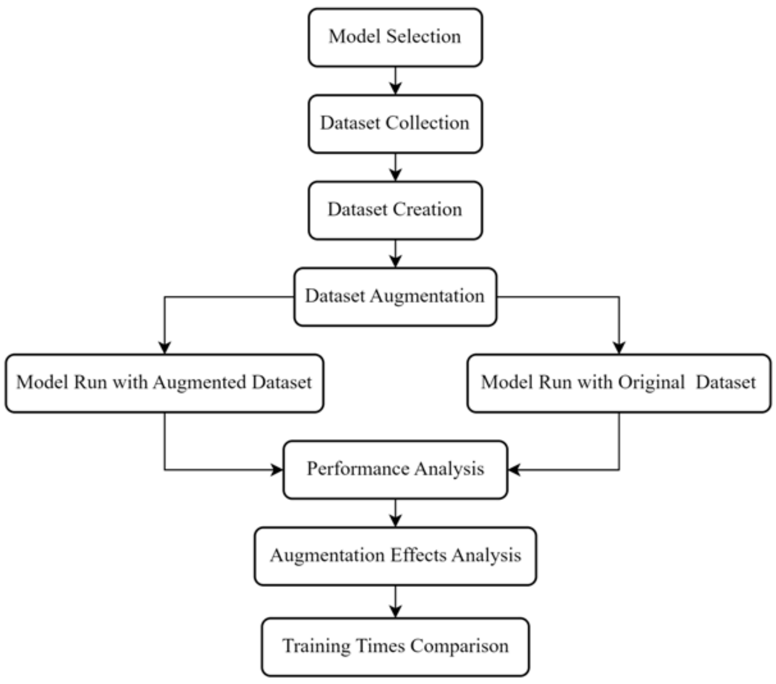

Model Selection
Dataset Collection
Dataset Creation
Dataset Augmentation
Model Run with Augmented Dataset
Model Run with Original Dataset
Performance Analysis
Augmentation Effects Analysis
Training Times Comparison

Figure 3.1: Research Methodology

Separate performance evaluation has been performed for both datasets. The tested lightweight models included- Compact Convolutional Transformer (CCT), Swin Transformer, Convolutional Neural Network (CNN), Involutional Neural Network (INN), ConvMixer and EANet; and tested heavyweight traditional models include- VGG16, ResNet50, DenseNet201 and Xception. The models were executed into Google Colab. We have utilised Colab's Nvidia Tesla T4 GPU acceleration as the preferred runtime type in order to achieve faster training times. We have run 200 epochs for all the models in our original dataset of 900 images and 100 epochs for the augmented dataset of 10,000 images. Running 200 epochs for the augmented dataset was not possible due to Google Colab's GPU Runtime Limitation for non-colab-pro users. Because of the same limitation, we were only able to run 50 iterations for the EANet and ConvMixer models due to their slower training times. Performance metrics including Test accuracy, Precision, Recall and F1 scores for all the test runs were comprehensively analysed in our study to reach meaningful experimental conclusions. Moreover, the loss curve and accuracy curve of all the models were evaluated to determine the stability, overfitting and underfitting of the models during training times. In addition, the training times of the models were also monitored and analysed as an extendable performance metric. Lastly, as we tested all the models in the augmented dataset, we were also able to comprehensively determine the effects of augmentation on different lightweight and heavyweight models.

Chapter 4

Dataset

We have compiled our own dataset of 900 images from roads of Dhaka and Bogura. Our dataset has two classes- normal roads and potholes. Among the 900 images, 438 were normal road samples and 462 were pothole samples. We have collected and processed the dataset complying with the dataset collection methodology enshrined in Whang et al. [The VLDB Journal, 2023]. Let us further explore our Dataset collection and pre-processing methodology.

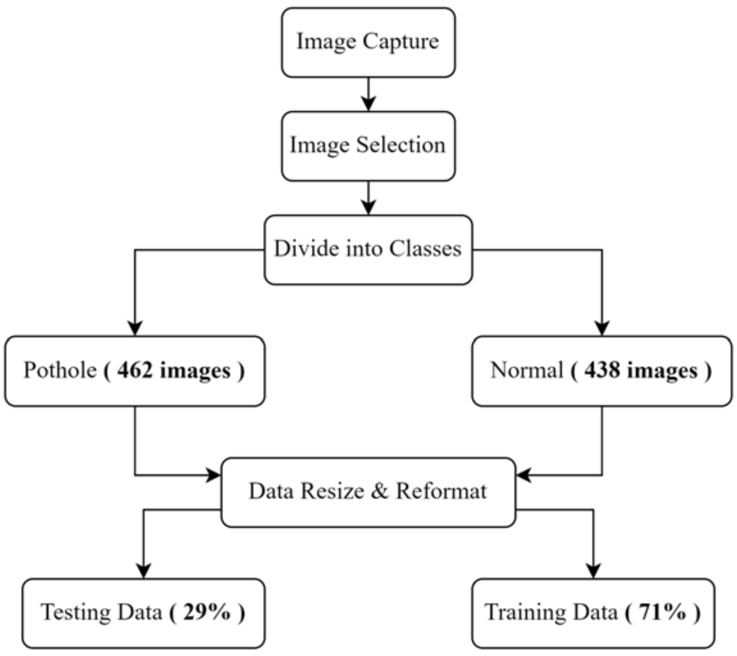


Figure 4.1: Data Collection Methodology

4.1 Collection Methodology:

Image Collection: We initially collected 1367 images combining normal road and pothole samples. Among 1367 images, we have selected 900 final images for our dataset. We have cleaned our dataset by excluding the unclear and duplicate images. We have also excluded or blurred images containing human faces or human-sensitive information such as car number plates, address plates etc., referring to data sanitisation. Moreover, we have kept an almost similar ratio of the two classes of samples (462:438) in order to avoid any kind of class imbalance or class biasness while executing into models. We have included a diverse set of pothole types in order to replicate real-life scenarios. The pictures also included the extended environmental components such as sky, trees and off-road elements along with road surfaces in order to avoid road-only biasness and present a real-world environment. The pictures were taken on iPhone 11, iPhone XR, iPhone 6s Plus and Xiaomi Redmi Note 12, ensuring acceptable picture clarity.

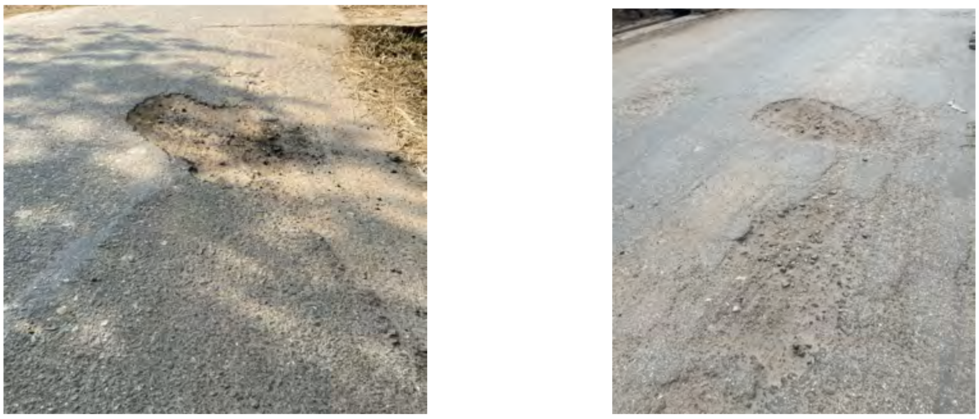


Figure 4.2: Pothole samples from our dataset

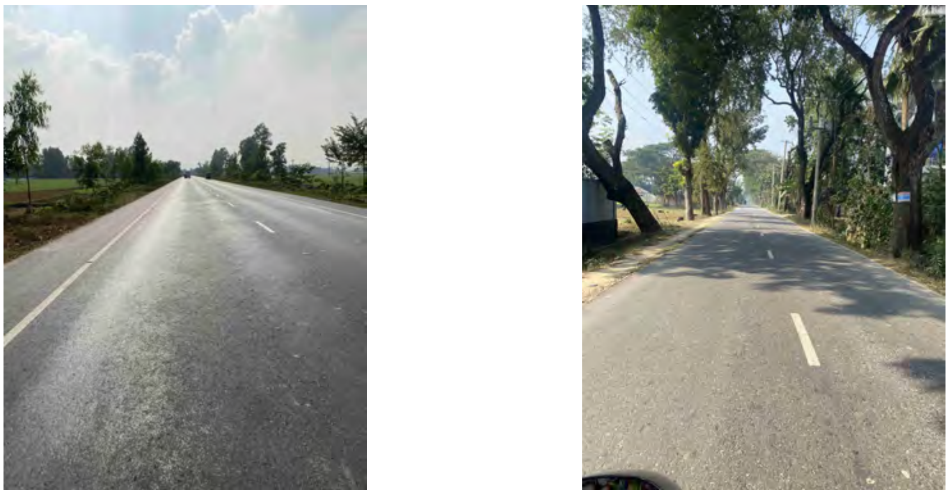


Figure 4.3: Normal road samples from our dataset

Batch Image Resize and Reformat: Upon methodically selecting 900 images from 1367 samples, we have resized and reformatted all the selected images into 768x1024 resolution JPG. With that, all the high resolution HEIC format images from iPhones and JPG format images from Xiaomi Redmi Note 12 got converted into 768x1024 JPGs. We have reduced the resolution in order to reduce size of the dataset. However, we ensured that all features of the images were still clearly visible. We also tested lower resolutions, but rejected them due to unsatisfactory image clarity. The resolution 768x1024 came out to be a satisfactory resolution upon several tests. The images were batch resized and reformatted by using a free Windows software named "FastStone Photo Resizer". The software offers robust functionalities for batch resize, rename and reformat.

We have selected the below mentioned settings in FastStone Photo Resizer app for batch reformatting our dataset. Before that, we selected all the images where the conversion would be conducted. This allowed all images to be converted into same resolution and same format in a single execution.

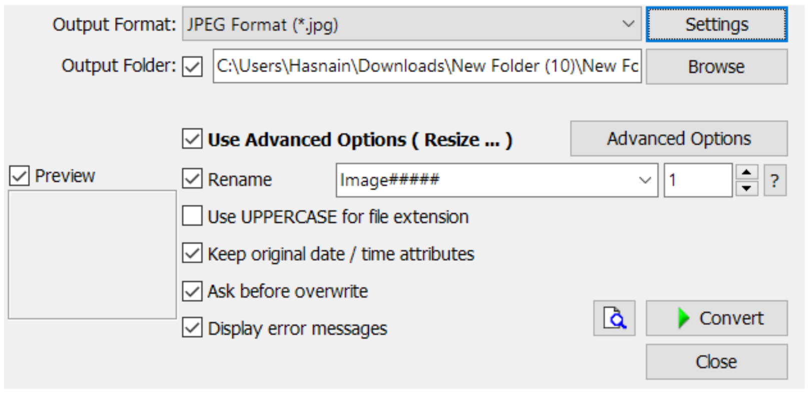

Output Format: JPEG Format (*.jpg)
Output Folder: ☑ C:\Users\Hasnain\Downloads\New Folder (10)\New Fc
☑ Use Advanced Options (Resize ...)
☑ Preview
☑ Rename Image##### 1
☐ Use UPPERCASE for file extension
☑ Keep original date / time attributes
☑ Ask before overwrite
☑ Display error messages

Figure 4.4: Selection of Output Format and Batch Rename (FastStone Photo Resizer)

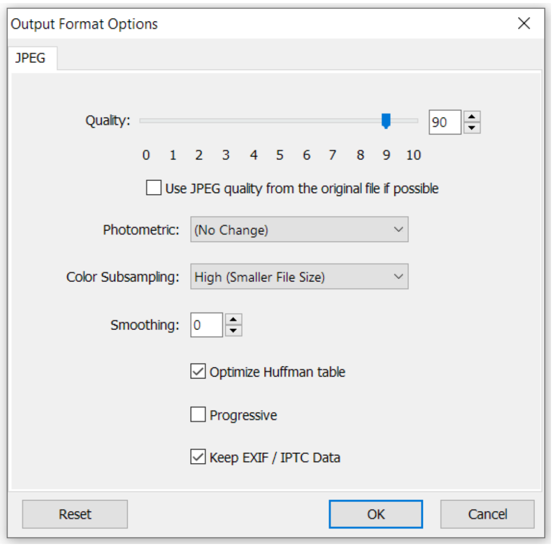
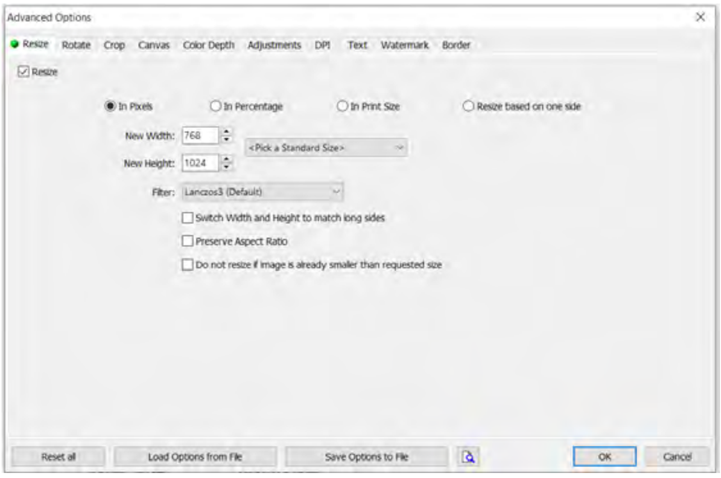

Output Format Options
JPEG
Quality: 90
0 1 2 3 4 5 6 7 8 9 10
☐ Use JPEG quality from the original file if possible
Photometric: (No Change)
Color Subsampling: High (Smaller File Size)
Smoothing: 0
☑ Optimize Huffman table
☐ Progressive
☑ Keep EXIF / IPTC Data

Figure 4.5: Output Format Options (Settings)



[✓] Resize

● In Pixels ○ In Percentage ○ In Print Size ○ Resize based on one side

New Width: 768

New Height: 1024

<Pick a Standard Size>

Filter: Lanczos3 (Default)

☐ Switch Width and Height to match long sides

☐ Preserve Aspect Ratio

☐ Do not resize if image is already smaller than requested size

Figure 4.6: Advanced Options

It took 9 minutes 44 seconds to batch resize our 900 high-resolution images in the Asus X542UN laptop (i5 8250U, 12GB Ram, 256GB SSD). The CPU utilisation was between 83% to 95% during execution. With that, 3.13GB images got compressed into 277MB only and we had images for our final dataset.

Image Labelling: As we were only testing classification, not segmentation, so we opted for manual labelling of classes by dividing the pictures into specified folders. The dataset in the first view has two folders- 'train' and 'test'. In both the folders, there are two subfolders- 'normal' and 'pothole'. We have assigned 71% data of both classes (328 out of 462 for pothole, 311 out of 438 for normal) in the train folder and 29% data (134 out of 462 for pothole, 127 out of 438 for normal) in the test folder. Upon assigning the 71% of data for training and 29% of data for testing, we now have our final dataset ready.

Archiving and Uploading to Google Drive: After finalising our dataset, we zipped the train and test folders using "Winrar" and then we uploaded the zip file into Google Drive. As we ran the neural network models from Google Colab, we mounted Google Drive inside Colab notebooks to easily access the dataset.

4.2 Data Augmentation:

Data augmentation refers to artificially increasing dataset samples by applying techniques such as rotation, cropping, padding, flipping, brightness and contrast adjustments etc. This way, the diversity of training data is increased while maintaining the originality of the dataset samples. We also have implemented data augmentation in our dataset and have later compared the performance difference between the original dataset and the augmented one. Augmentation significantly enhanced test accuracy and reduced overfitting in the models as per our tests. Test accuracy and F1 scores of 9 out of 10 tested models were incremented after using the augmented dataset. Only ConvMixer among all the models has performed negatively in the augmented dataset.

We have used Python Augmentor library to augment our dataset. 10,000 images were generated from our dataset of 900 images by applying left-right top-down flipping, random brightness, random contrast, random rotate, skew tilt and shear. 5000 images were generated for each class. We have avoided random cropping because cropping may cut out pothole instances from pictures, leading to inaccurate training and validation data. We have also capped the rotation of images to up to 22 degrees in order to avoid pothole cutting instances. However, we have enabled both left-right and top-down flipping as they may increase diversity of data without cutting pothole instances. We have also set minimum and maximum limits of brightness and contrast in order to avoid unclear images. Lastly, we have implemented skewing and shearing in small scales in the augmented dataset by reducing their corresponding probability.

The augmentation operation was performed locally in the X542UN laptop (i5 8250U, 12GB Ram, 4GB MX150 GPU, 256GB SSD). It took 8 minutes 32 seconds to generate 10,000 images, 5000 of each class. The code was run into Visual Studio Code while importing the Augmentor library. The augmented pictures were then assigned into train and test folders into two classes similar to the original dataset. However, this time 80% of the data were assigned for training and 20% for testing. The final 10,000 image dataset came out to be 1.43GB. This dataset was then uploaded into Google Drive and was similarly executed into models in Google Colab like the original dataset. In most of the models, the training time of the augmented dataset was about 10 to 11 times higher compared to the original dataset as there were 11 times more data samples now. Augmented dataset significantly increased the test accuracy of the models while also reducing overfitting. We will explore detailed performance analysis of the augmented dataset in the ‘Experimental Results” segment of this thesis book. In addition, a dataset containing 4000 images was created with the exact same methodology for specifically testing the ConvMixer model as the model presented unstable results with the 10,000 sample dataset.

4.3 Model Input Psuedo-Code

While running the models, we have inserted our dataset into x_train, x_test, y_train and y_test variables by implementing the following pseudocode-

```
1. Mount Google Drive
2. Extract the dataset zip folder in Google Colab
3. Iterate through the train and test folder and store all corresponding images being resized to 128x128 pixels into the x_train and x_test list variables.
4. Convert x_train and x_test lists into NumPy arrays.
5. Extract labels from loaded images and store them into y_train and y_test variable.
```

Chapter 5

Performance Evaluation

5.1 Test Setup:

Mobile Phones:(For capturing dataset pictures)

1. iPhone 11
2. iPhone XR
3. iPhone 6s Plus
4. Xiaomi Redmi Note 12

Computers:(For dataset batch resize, dataset augmentation and running models on Colab)

1. Asus Vivobook X542UN: CPU: Intel Core i5 8250U, Ram: 12GB, GPU: Nvidia MX150 4GB, Storage: Transcend 820s 256GB M.2 SSD.
2. Lenovo Thinkpad T420: CPU: Intel Core i5 2520M, Ram: 4GB, GPU: Intel HD Graphics 3000, Storage: Samsung SM841N 256GB mSATA SSD.
3. Desktop PC: CPU: Intel Xeon E3-1220, Ram: 16GB, GPU: Nvidia GT 710, Storage: Intel DC S3710 Series 200GB Sata SSD.

Operating System:

Windows 10 Home Build 19045 (On 3 computers)

Google Colab GPU Runtime System Specs:

Ram: 12GB

GPU: Nvidia Tesla T4 15GB VRAM

Disk Allocation: 80GB

Tools:

1. FastStone Photo Resizer (For batch resizing dataset images to 768x1024 jpg)
2. Winrar (For dataset Archiving)
3. Python Augmentor Library (For dataset augmentation)
4. Visual Studio Code (For dataset augmentation)
5. Microsoft Excel (To generate graphs)
6. Microsoft Word and Google Docs (Word Processor)
7. Google Drive (To store dataset)
8. Google Colab (To run Models)
9. Draw.io (To draw model architecture diagrams)

5.2 Experimental Results

Our original dataset of 900 images and the augmented dataset of 10,000 images were run into six lightweight and four heavyweight models. The lightweight models overall performed better in both of the datasets compared to the traditional models while having fewer parameters and faster training times. Let us explore the experimental results in detail. This segment is divided into three categories. At first, we will look into the performance analysis of all the tested models on our original dataset of 900 images. Secondly, we will explore the performance comparison on the augmented dataset of 10,000 images. Lastly, we will explore the performance analysis of each individual model along with their loss and accuracy curves.

5.2.1 Performance Analysis of All Tested Models (For Original Dataset)

We have evaluated the performance of all the models while being run onto our original dataset of 900 images in this segment. Test Accuracy, F1 Score, Training Time /Epoch and parameters of the models have been compared here. As the F1 score involves consideration of both precision and recall, so we have included a comparison of the F1 score only in this segment. Each model was run for 200 epochs. Additionally, a performance record table is provided below which has the record of all the tests conducted over the original dataset.

Performance Record Table (All Models):

Model	Test Accuracy (%)	Precision	Recall	F1 Score	Time /epoch(s)	Parameters	Size-(MB)
CCT	77.01	0.73	0.67	0.7	4	407107	1.55
Swin Transformer	62.45	0.75	0.71	0.73	5	412530	1.64
CNN	78.56	0.73	0.81	0.77	1	641600	2.45
INN	60.92	0.61	0.61	0.61	<1	196880	0.77
ConvMixer	83.91	0.63	0.75	0.68	20	600834	2.99
EANet	60.54	0.71	0.69	0.7	15	355017	1.35
VGG16	62.45	0.63	0.65	0.64	8	14731074	56.19
ResNet50	65.13	0.71	0.66	0.68	8	23653250	90.23
DenseNet201	63.60	0.65	0.64	0.64	9	18377154	70.1
Xception	51.34	0.50	0.50	0.5	9	20926442	79.83

Table 5.1: Performance Record (900 images, 200 Epoch)

Test Accuracy Comparison (All Models)

ConvMixer has achieved the highest accuracy of 83.91% among all the tested models. The top three accuracy scores came from the lightweight models with CNN having 78.56% accuracy and CCT having 77.01. None of the heavyweight models were able to cross the accuracy of 70%. ResNet50, among the heavyweight models, has achieved the highest accuracy of 65.13%. Xception was the worst performing model while being run onto the 900 image dataset, having a test accuracy of only 51.34%.

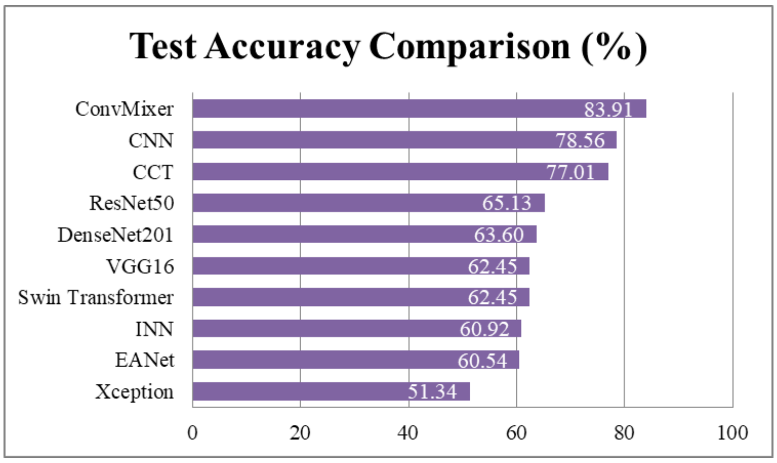


Figure 5.1: Test Accuracy Comparison (Original Dataset, 200 Epoch)

F1 Score Comparison (All Models)

F1 Score is considered as a better metric for performance evaluation of machine learning models compared to test accuracy as it involves consideration of both precision and recall. We can see in the graph that even though ConvMixer achieved the highest test accuracy, it lagged behind in F1 scores. CNN has provided the highest F1 score of 0.77, followed next by Swin Transformer, EANET and CCT. All three of these models achieved an F1 score of 0.70 or more. Even though Swin Transformer and EANet had lower accuracy scores, they exhibited higher F1 scores. The heavyweight models were followed after them. Xception was once again the worst-performing model, exhibiting an F1 score of only 0.5.

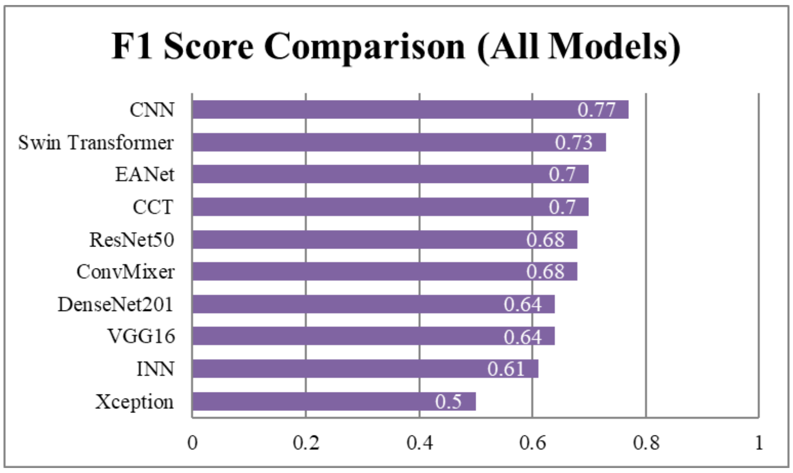


Figure 5.2: F1 Score Comparison (Original Dataset, 200 Epoch)

Training Time/Epoch Comparison (All Models)

Even though training times directly do not affect the accuracy or prediction times, it is an important metric to consider for developers and researchers. They can organise their research better if the training time of models are known. Along with that, training times also indicate the optimisation of a model. For example, all four heavyweight models we have tested finished training of 200 epochs before EANET and ConvMixer despite having a significantly high number of parameters. In our testing, INN was the fastest model to finish training with per epoch time of less than 1 second. CNN was the second fastest with a per epoch time of 1 second, followed by CCT and Swin Transformer, having per epoch times of 4 and 5 seconds respectively. The heavyweight models, in comparison took 8-9 seconds per epoch in the training phase. Lastly, EANet and ConvMixer exhibited the slowest training times even after having parameters of less than 1 million. This is reminiscent of being unoptimised models for training. EANet took 50 minutes and ConvMixer took more than one hour to finish 200 epochs. However, even though two of the lightweight models have shown slower training times, it must not be mistaken for prediction or inference time. Prediction or detection time is more dependent upon model parameters as models are deployed in a system along with their parameters and size. That's why lightweight models will always have an edge over heavyweight models in case of prediction times, making them always preferable for real-time detection and to be used on embedded devices.

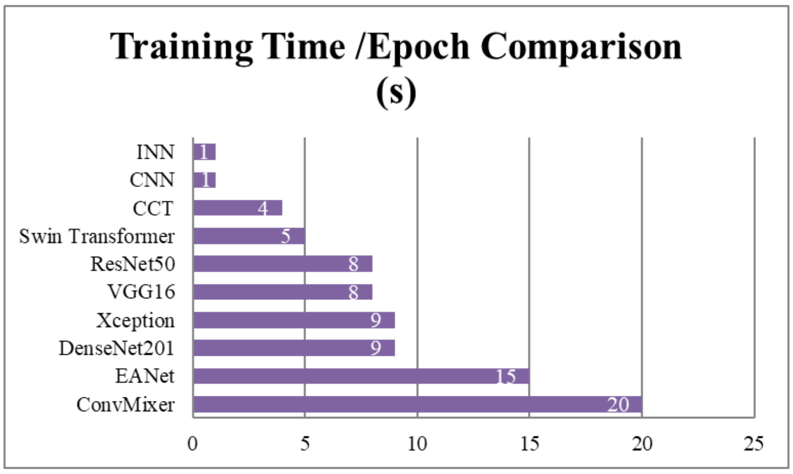


Figure 5.3: Training Time /Epoch Comparison (Original Dataset)

Parameter Comparison (All Models)

Below is a graph containing a parameter comparison of all the tested models. We can see that all the lightweight models had parameters of less than one million. Heavyweight models on the other hand contained 20 to 40 times more parameters compared to the lightweight models. INN had the least number of parameters and also finished the training phase the fastest. CNN, even after having the highest parameters among all lightweight models in our testing, had a per epoch time of only 1 second which signifies its decent training optimisation. Among the heavyweight models, ResNet50 has shown the fastest training times even after having the highest parameters among the heavyweight models which signify its decent optimisation also.

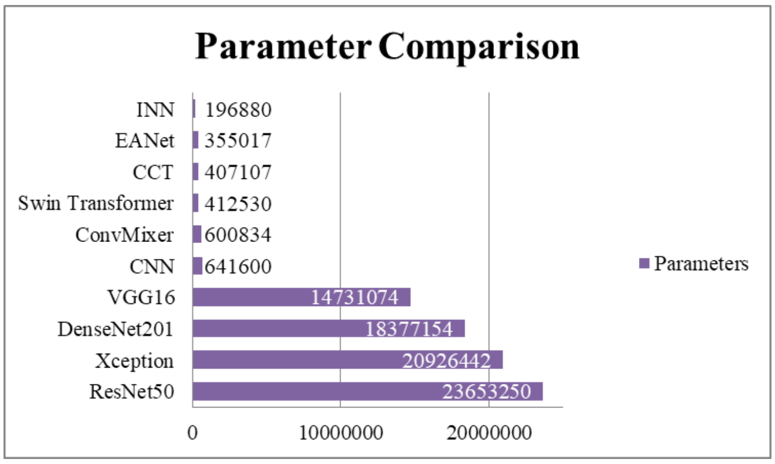


Figure 5.4: Parameter Comparison (All Tested Models)

Overall Results:

CNN and CCT were the overall best models while being run onto our 900 image dataset. Both of them had balanced test accuracy and f1 scores. Other models had instabilities while considering both metrics. The heavyweight models exhibited overfitting and gave out disappointing test accuracy. We will further look into the loss and accuracy curves of each individual model in the 5.2.3 segment of the book in order to assess the overfitting and underfitting of the models.

5.2.2 Performance Analysis of All Tested Models (For Augmented Dataset)

For the augmented dataset of 10,000 images, 100 epochs were run to assess the performance of the models due to Google Colab GPU Runtime limitation. Due to the same limitation, we were only able to run 50 epochs for EANet and ConvMixer due to their slower training times. Below are the performance record table and graphs of Test Accuracy, F1 Score and Training Time /Epoch comparison for the augmented dataset.

Performance Record Table (All Models)

Model	Test Accuracy (%)	Precision	Recall	F1 Score	Time /epoch(s)	Parameters	Size-(MB)
CCT	94.6	0.87	0.94	0.9	50	407107	1.55
Swin Transformer	92.55	0.90	0.91	0.9	54	412530	1.64
CNN	96.65	0.96	0.97	0.96	11	641600	2.45
INN	92.10	0.89	0.92	0.9	5	196880	0.77
ConvMixer	63.80	0.49	0.57	0.53	229	600834	2.99
EANet	83.5	0.81	0.74	0.77	180	355017	1.35
VGG16	78.15	0.78	0.78	0.78	54	14731074	56.19
ResNet50	89.60	0.90	0.88	0.89	52	23653250	90.23
DenseNet201	88.65	0.89	0.88	0.88	55	18377154	70.1
Xception	70	0.70	0.70	0.7	55	20926442	79.83

Table 5.2: Performance Record (10000 images, 100 Epoch)

Test Accuracy Comparison (All Models)

Augmentation significantly improved the performance of the tests. This time, we had the highest accuracy of 96.65% achieved by CNN, followed by CCT's 94.6%, Swin

Transformer's 92.55% and INN's 92.10%. All 4 of these models achieved accuracy scores of over 90 percent. Lightweight models excelled in the accuracy scores once again. ResNet50 has achieved the highest accuracy among the heavyweight models once again, having an accuracy of 89.60%. It was followed by DenseNet201 having 88.65% accuracy. Surprisingly, ConvMixer performed the worst in the augmented dataset whereas it performed the best in terms of accuracy in the original dataset. We can now assess the importance of evaluating F1 scores along with test accuracy as test accuracy can sometimes be misleading. All the models with balanced accuracy and F1 scores performed best here also.

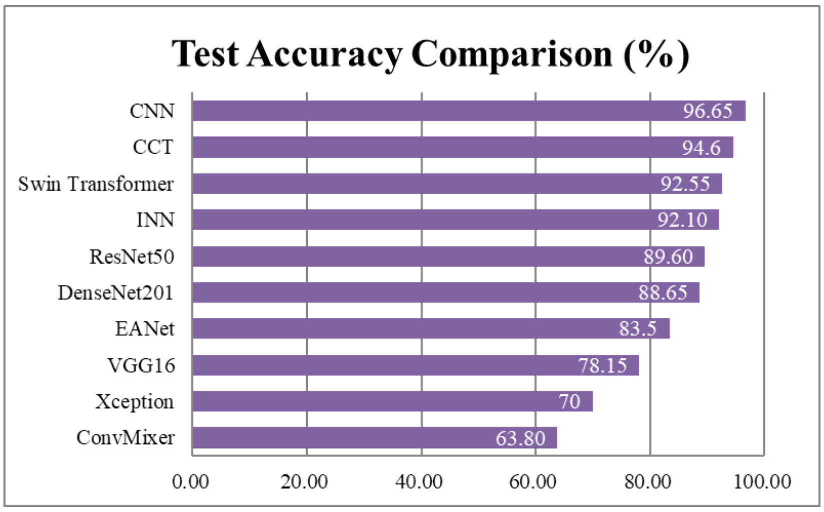


Figure 5.5: Test Accuracy Comparison (Augmented Dataset, 100 Epoch)

F1 Score Comparison (All Models)

CNN has provided the highest F1 scores alongside the highest test accuracy, establishing it as the best overall model among all the models tested. INN has provided the second-highest F1 score of 0.9. INN has significantly improved in terms of accuracy and F1 score in the augmented dataset. Swin transformer and CCT also provided balanced F1 scores of 0.9 alongside decent accuracy. ResNet50 and DenseNet201 have also provided stable F1 scores alongside decent accuracy. ConvMixer, unsurprisingly this time, provided the lowest F1 score, establishing it as the worst performing and the most unstable model among all the tested models.

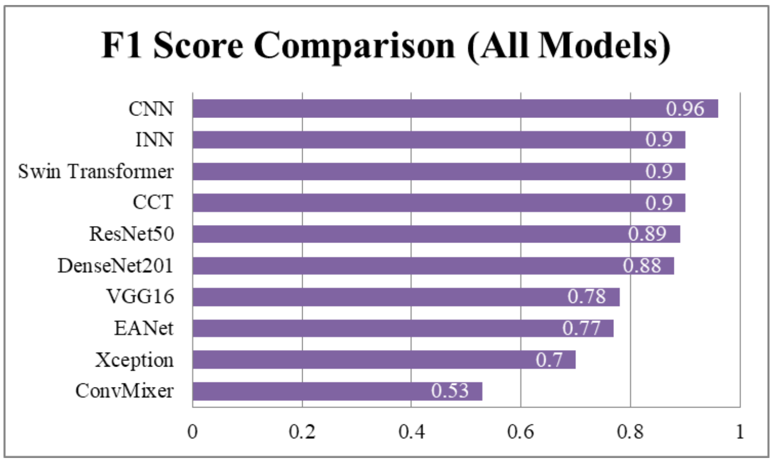


Figure 5.6: F1 Score Comparison (Augmented Dataset, 100 Epoch)

Training Time/epoch Comparison (All Models)

While executing in the augmented dataset, the training times of the models increased 7 to 11 times compared to the original dataset while having 11 times more image samples. INN again came out to be the fastest in terms of training time finishing per epoch in only 5 seconds. The speed ranks stayed more or less the same except for ResNet50 and VGG16 finishing faster than swin transformer this time. The heavyweight models provided lower training time increments while running in the augmented dataset, compared to the lightweight models. EANet took 180 seconds or 3 minutes and ConvMixer took 229 seconds or nearly 4 minutes to finish 1 epoch. That's why we had to test these two models for 50 epochs. Training them for 100 epochs could have changed their accuracy and F1 score outcomes.

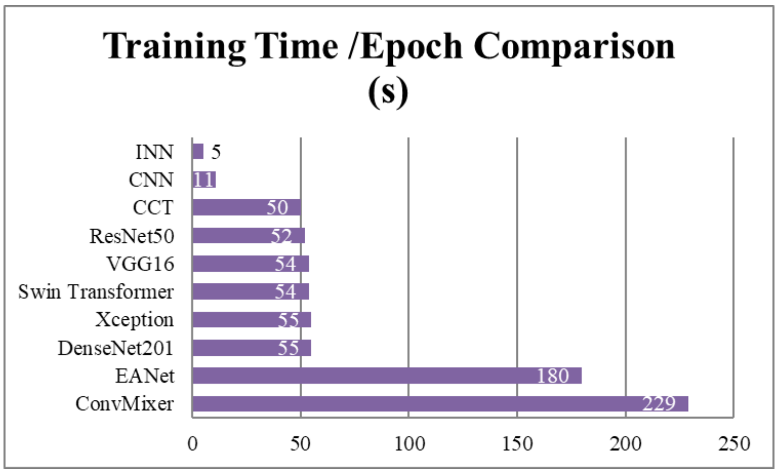


Figure 5.7: Training Time /Epoch Comparison (Augmented Dataset)

Overall Results:

CNN and CCT were the best models once again while also executing in the augmented dataset. INN and Swin transformers also performed really well having decent accuracy and F1 scores. ResNet50 was the best model among the heavyweight models having the highest accuracy, highest f1 score and the lowest training time among all the heavyweight models. DenseNet50 was 2nd best among heavyweight models in terms of accuracy and F1 score. Lightweight models overall performed better compared to heavyweight models once again. The accuracy of the heavyweight models could be further increased if 200 epochs were executed.

5.2.3 Performance Analysis of Individual Tested Models:

Let us now look at the detailed performance analysis of the individual models. This segment includes comparison of accuracy rates, precision, recall and f1 scores of each model while being run onto both the original and augmented dataset. Loss curves and accuracy curve characteristics are also compared for both the datasets in this segment.

5.2.3.1 CCT

Test Accuracy, Precision, Recall, F1 Score:

Augmentation increased all four performance metrics for the CCT model. In the augmented dataset, accuracy increased from 77.01% to 94.6%, F1 score increased from 0.7 to 0.9, precision increased from 0.73 to 0.87 and recall increased from 0.67 to 0.94 compared to the original dataset. CCT achieved the second-highest accuracy among all the models tested in the augmented dataset.

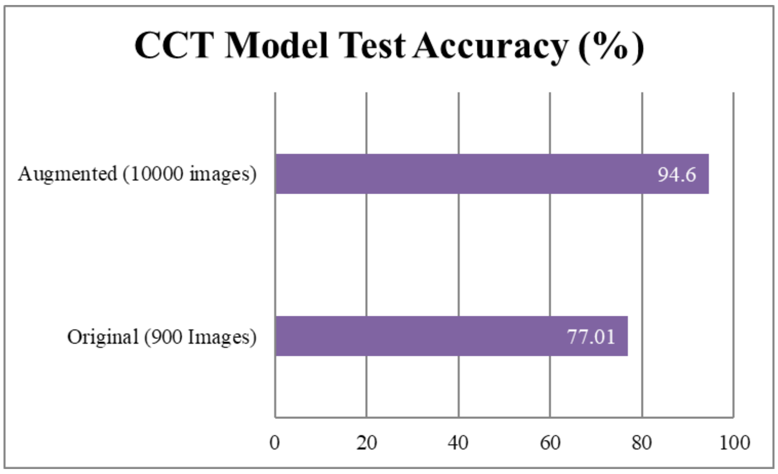


Figure 5.8: CCT Test Accuracy (Original vs Augmented Dataset)

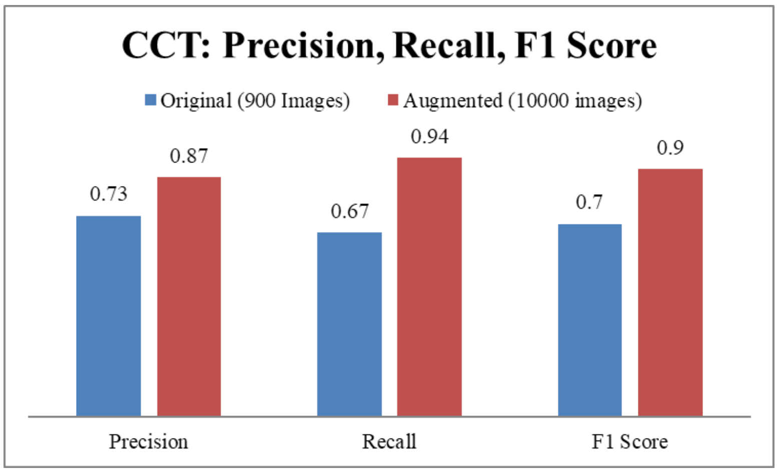


Figure 5.9: CCT F1, Precision, Recall (Original vs Augmented Dataset)

Loss Curve:

We can see that in the below curves, the augmented dataset has reduced train and validation losses by looking at the y-axis values. The augmented curve is also more stable compared to the original dataset curve. The train and validation curves are close to each other, with a minor occurrence of overfitting which is further reduced in the augmented curve.

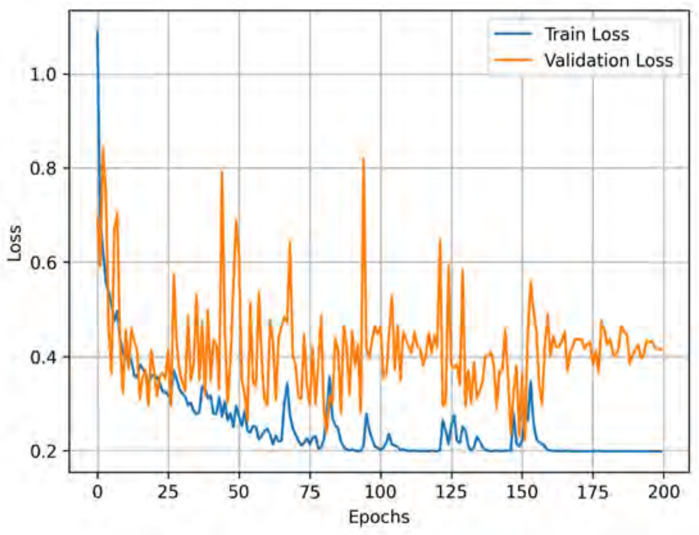


Figure 5.10: CCT Loss Curve (Original Dataset)

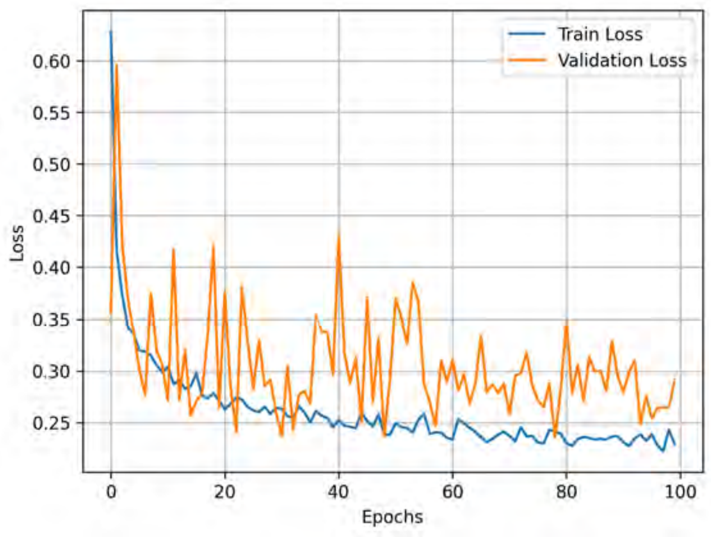


Figure 5.11: CCT Loss Curve (Augmented Dataset)

Accuracy Curve:

We can see from the figures that augmentation has stabilised the accuracy curve. The initial accuracies are also higher in the augmented curve if we look at the y-axis values.

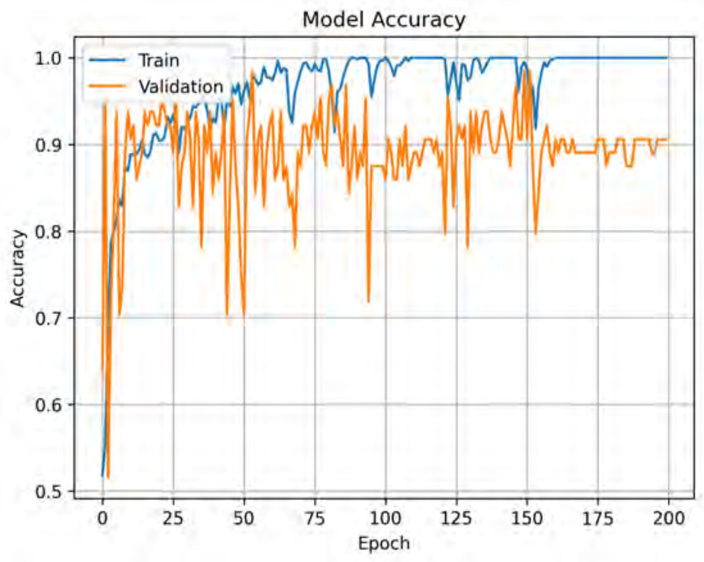


Figure 5.12: CCT Accuracy Curve (Original Dataset)

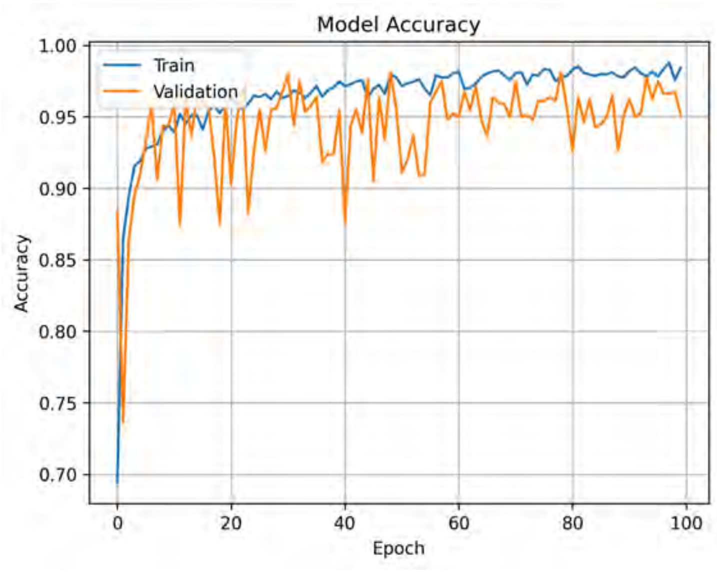


Figure 5.13: CCT Accuracy Curve (Augmented Dataset)

5.2.3.2 Swin Transformer:

Test Accuracy, Precision, Recall, F1 Score:

All four performance metrics in the Swin Transformer model were increased upon augmentation. Here, the accuracy rate was increased from 62.45% to 92.55%, the F1 score increased from 0.73 to 0.9, precision increased from 0.75 to 0.9 and recall increased from 0.71 to 0.94 in contrast to the original dataset of 900 images. It was overall the 3rd best performing model in our testing.

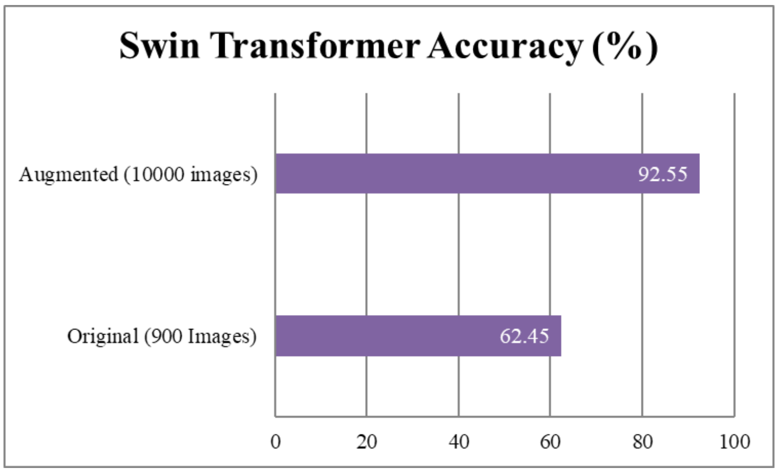


Figure 5.14: Swin Transformer Test Accuracy (Original vs Augmented Dataset)

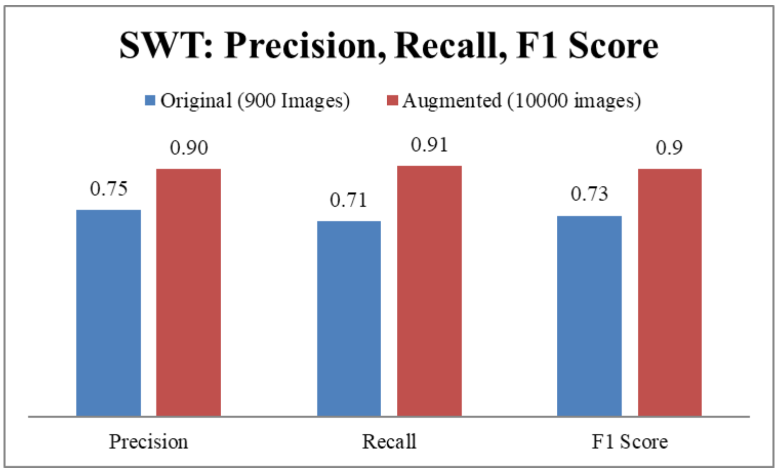


Figure 5.15: Swin Transformer F1, Precision, Recall (Original vs Augmented Dataset)

Loss Curve:

The loss curves look really flat because initial losses were significantly higher. Yet, if we look at the y-axis values, then we can see that loss is decreased in the augmented dataset. The overall curves look pretty stable with some minor curves.

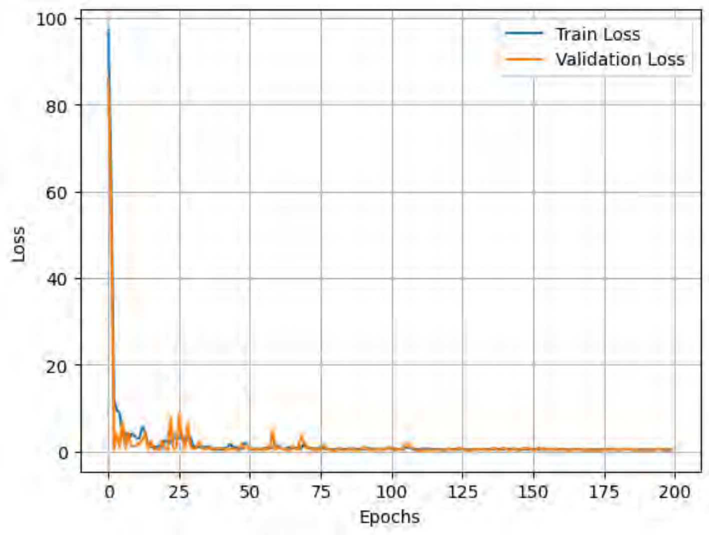


Figure 5.16: Swin Transformer Loss Curve (Original Dataset)

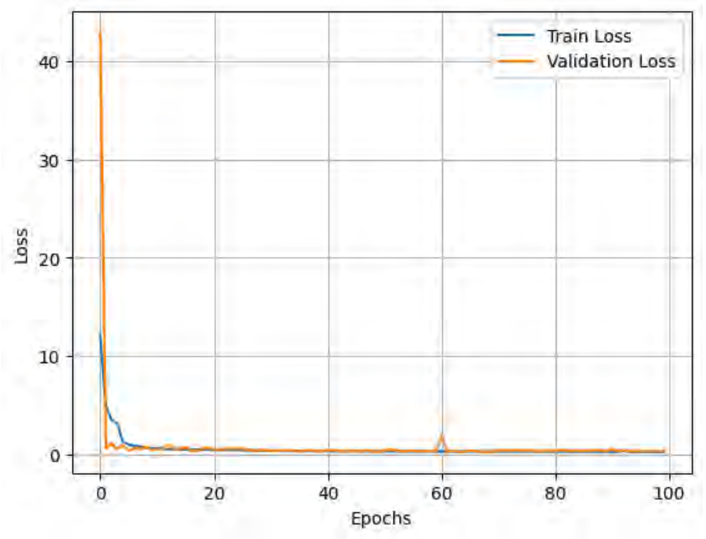


Figure 5.17: Swin Transformer Loss Curve (Augmented Dataset)

Accuracy Curve:

The loss curves look really flat because initial losses were significantly higher. Yet, if we look at the y-axis values, then we can see that loss has decreased in the augmented dataset. The overall curves look pretty stable with some minor curves.

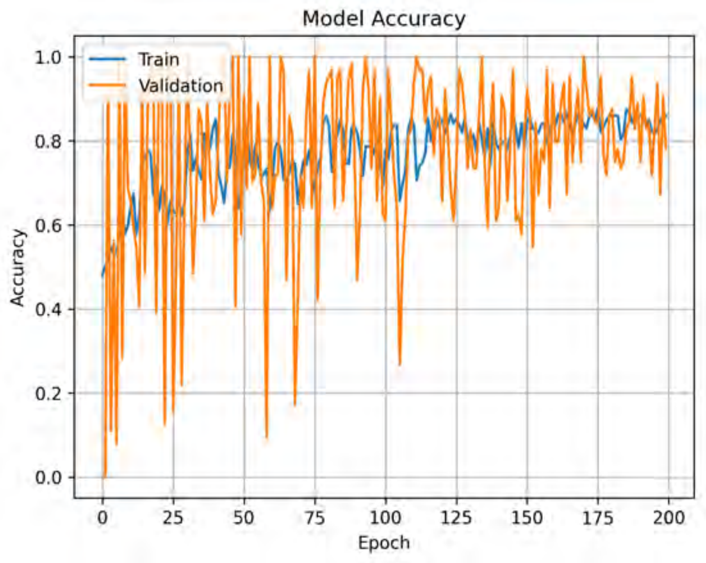


Figure 5.18: Swin Transformer Accuracy Curve (Original Dataset)

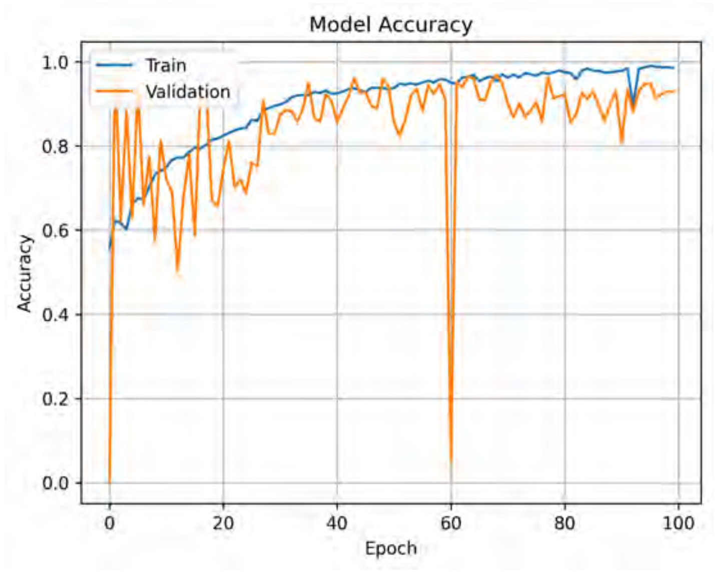


Figure 5.19: Swin Transformer Accuracy Curve (Augmented Dataset)

5.2.3.3 CNN

Test Accuracy, Precision, Recall, F1 Score:

All four performance metrics in the CNN model increased upon augmentation. Here, the accuracy rate was increased from 78.56% to 96.65%, F1 score increased from 0.77 to 0.96, precision increased from 0.73 to 0.96 and recall increased from 0.81 to 0.97 in contrast to the original dataset. CNN exhibited the highest accuracy and f1 score among all the tested models in the augmented dataset. It also achieved the highest f1 score for the original dataset also. It was the overall best performing model as per our testing.

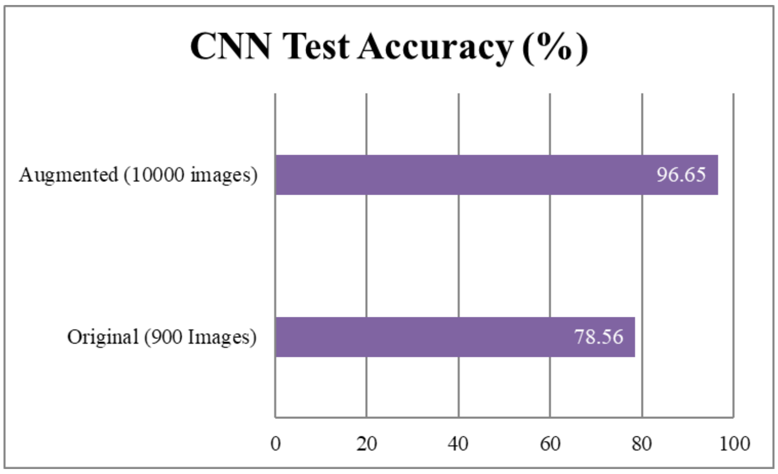


Figure 5.20: CNN Test Accuracy (Original vs Augmented Dataset)

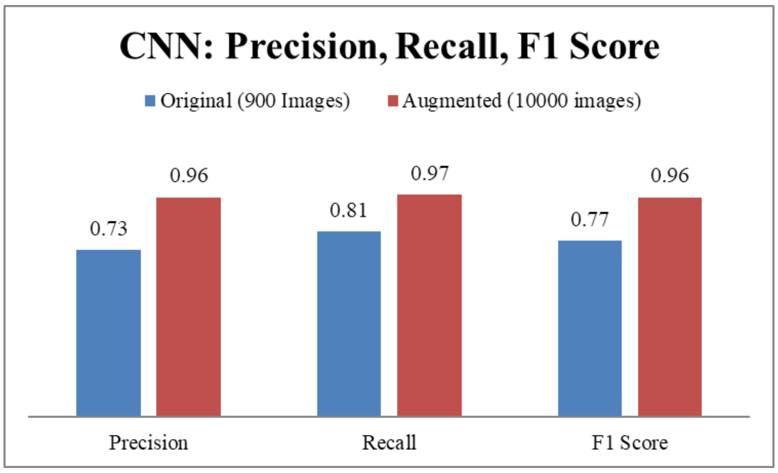


Figure 5.21: CNN Precision, Recall, F1 (Original vs Augmented Dataset)

Accuracy and Loss Curve (Original Dataset):

Here, we can see that the model loss curve has stayed quite stable because of really high initial loss. Also, validation accuracies in the accuracy curve are quite lower compared to training accuracies.

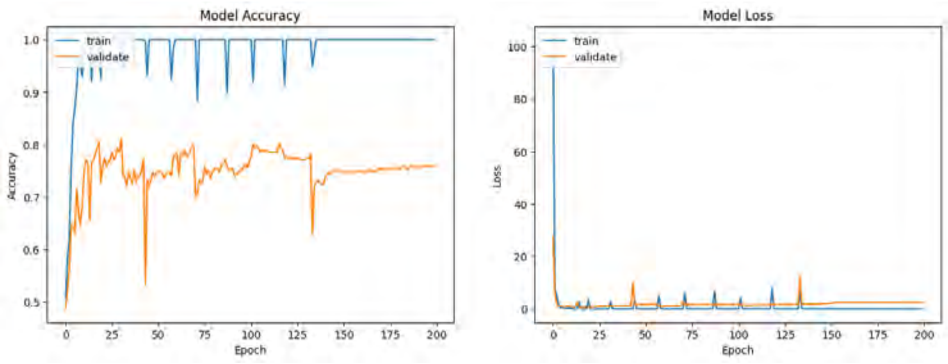


Figure 5.22: CNN Accuracy and Loss Curve (Original Dataset)

Accuracy and Loss Curve (Augmented Dataset):

Augmentation has significantly reduced model loss which can be seen by looking at the y-axis values. Augmentation has also incremented the validation accuracies and has overall stabilised the accuracy curve.

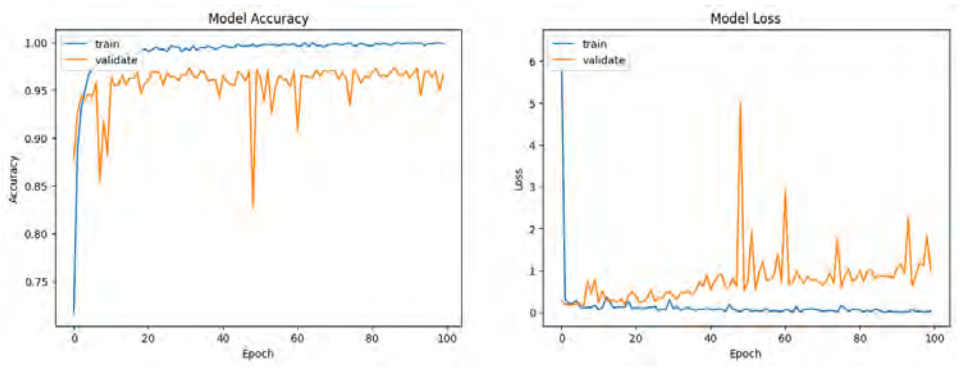


Figure 5.23: Fig: CNN Accuracy and Loss Curve (Augmented Dataset)

5.2.3.4 INN

Test Accuracy, Precision, Recall, F1 Score:

INN also showed increment on all four performance metrics upon augmentation. Here, the accuracy rate was increased from 60.92% to 92.10%, F1 score increased from 0.61 to 0.9, precision increased from 0.61 to 0.89 and recall increased from 0.61 to 0.92 in contrast to the original dataset. INN, among all the models, had the highest performance bump after augmentation. The performance improvement was over 150% which shows that, it was able to greatly utilise the diversified data of the augmented dataset.

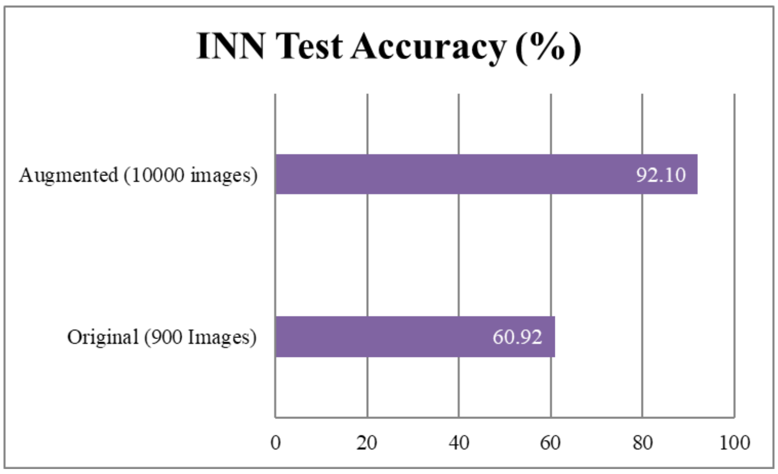


Figure 5.24: INN Test Accuracy (Original vs Augmented Dataset)

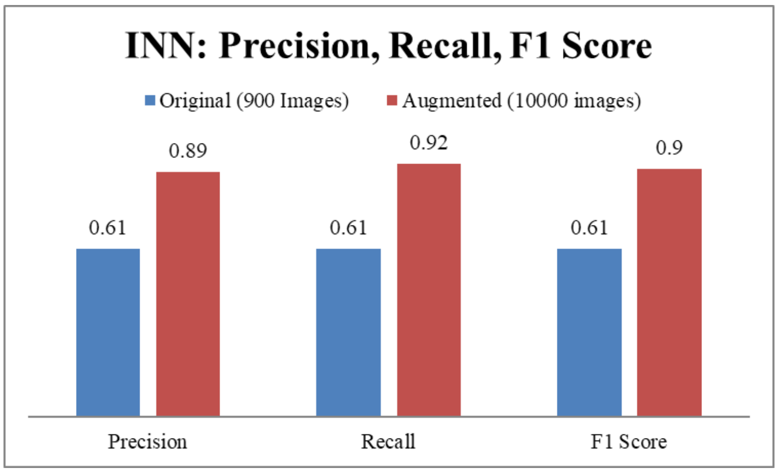


Figure 5.25: INN Precision, Recall, F1 (Original vs Augmented Dataset)

Accuracy and Loss Curve (Original Dataset):

INN showed insanely high initial validation loss in the original dataset for which validation loss has kind of disappeared. The accuracy gap in the accuracy curve is also very high for the original dataset. Overall, these are pretty unstable loss and accuracy curves.

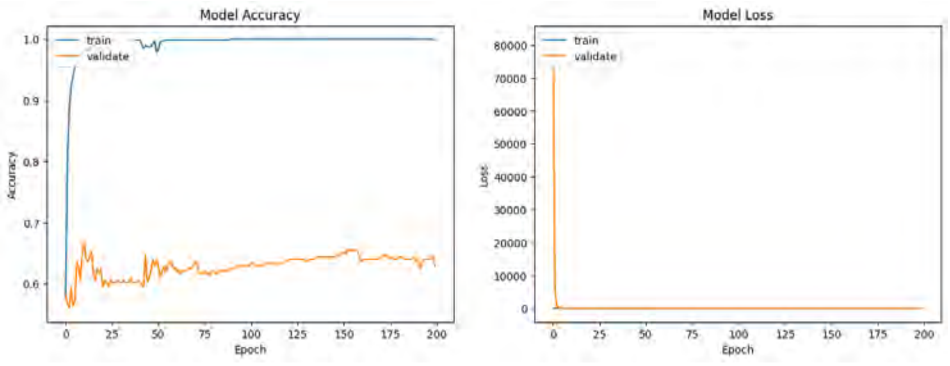


Figure 5.26: INN Accuracy and Loss Curve (Original Dataset)

Accuracy and Loss Curve (Augmented Dataset):

Augmentation has significantly reduced overall model loss which can seen by looking at y-axis values. However, there is still some overfitting present in the loss curve which signifies that, there is still scope of performance improvement in this model. The accuracy curve has also become significantly more stable upon augmentation.

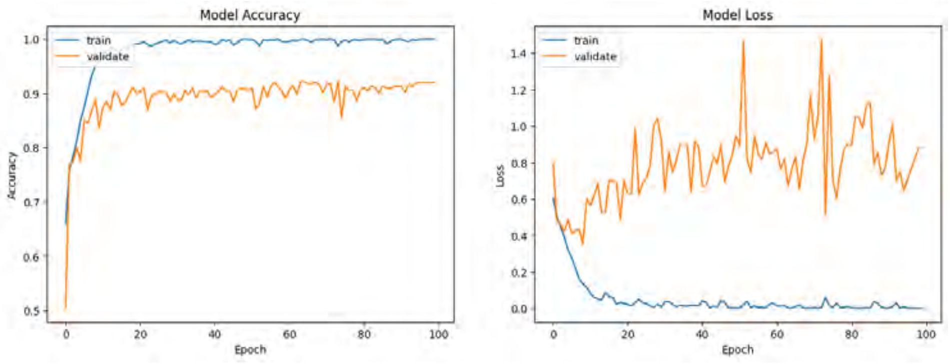


Figure 5.27: INN Accuracy and Loss Curve (Augmented Dataset)

5.2.3.5 ConvMixer

Test Accuracy, Precision, Recall, F1 Score:

ConvMixer was the only model in our testing that had a negative impact on performance upon augmentation. All four performance metrics decreased for ConvMixer upon augmentation. The augmented data seems to create biasness in prediction for which we manually created another dataset of 4000 images and tested all the performance metrics. This time also, all four metrics dropped compared to running the model in the original dataset, however, the drop was lower compared to the dataset of 10,000 images. ConvMixer came out to be a non-augmentation-friendly model as per our testing. All the performance metric scores are visible in the below graphs.

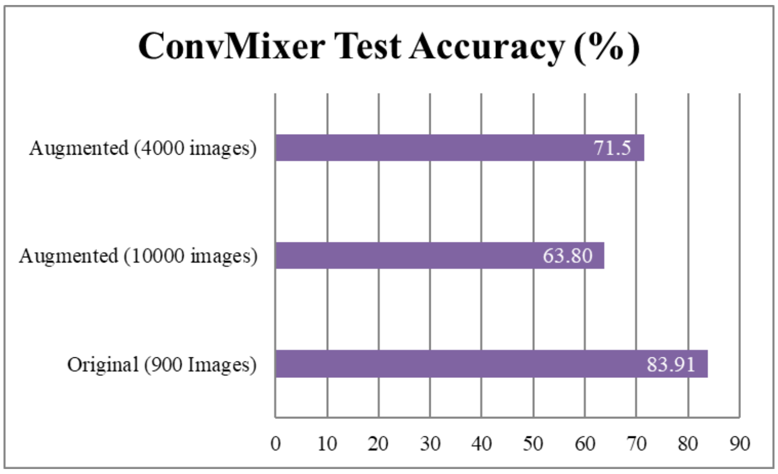


Figure 5.28: ConvMixer Test Accuracy (Original vs Augmented Datasets)

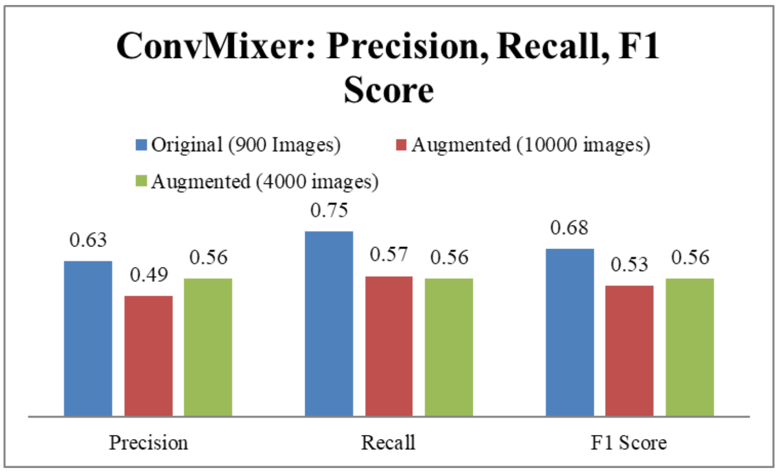


Figure 5.29: ConvMixer Precision, Recall, F1 (Original vs Augmented Datasets)

Loss Curve:

The loss curve for the original dataset has many spikes. The loss curve of 10,000 image dataset has less curves, but there have been instance of overfitting. Overfitting was reduced in the 4000 image dataset and the curve seems more stable. Yet, we encountered pretty low-performance values.

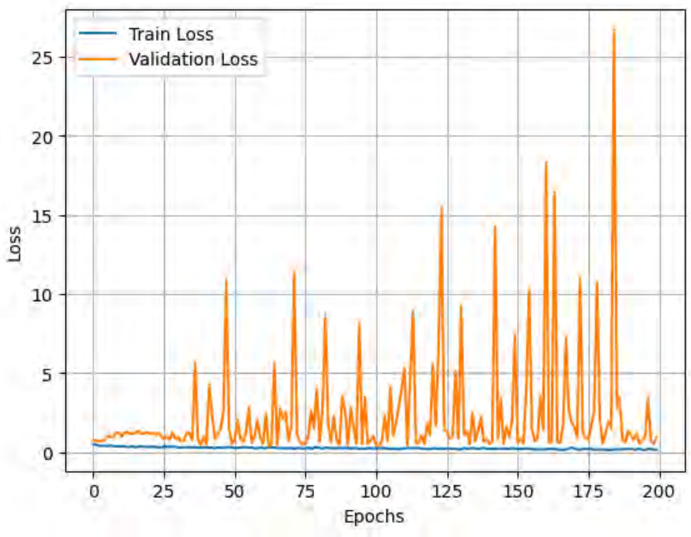


Figure 5.30: ConvMixer Loss Curve (Original Dataset)

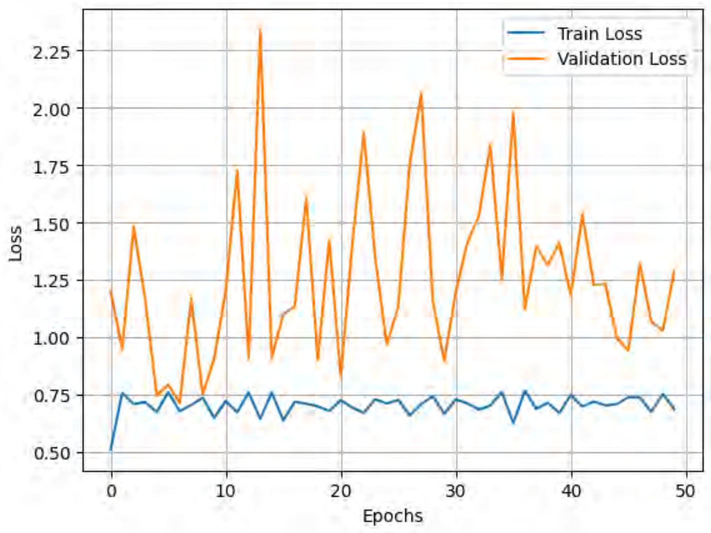


Figure 5.31: ConvMixer Loss Curve (Augmented Dataset 10000 Images)

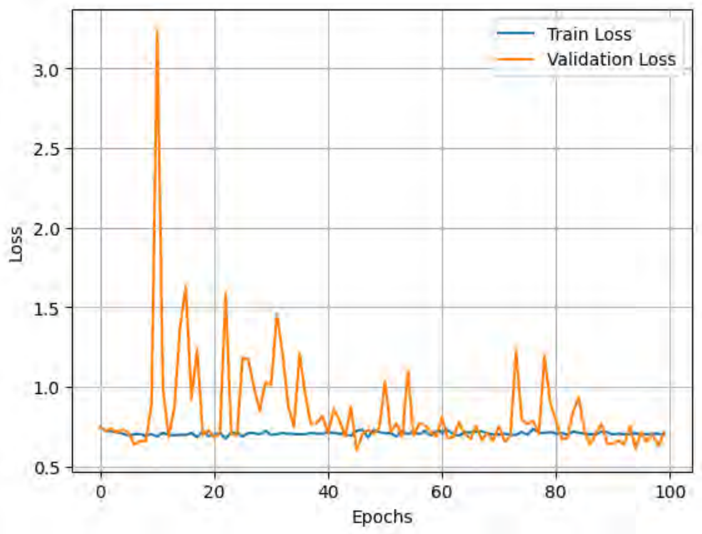


Figure 5.32: ConvMixer Loss Curve (Augmented Dataset 4000 Images)

Accuracy Curve:

The accuracy curve scenario is also similar to loss curves. It was more unstable for the original dataset, less unstable for the 10000 image dataset and least unstable for the 4000 image dataset.

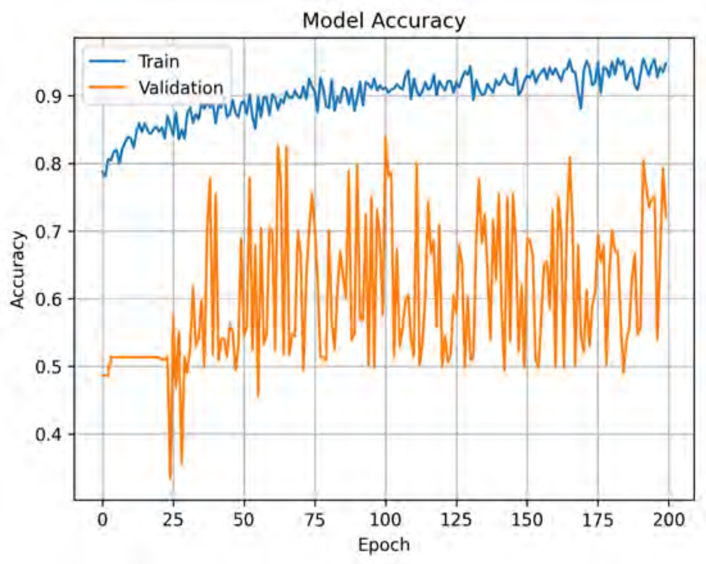


Figure 5.33: ConvMixer Accuracy Curve (Original Dataset)

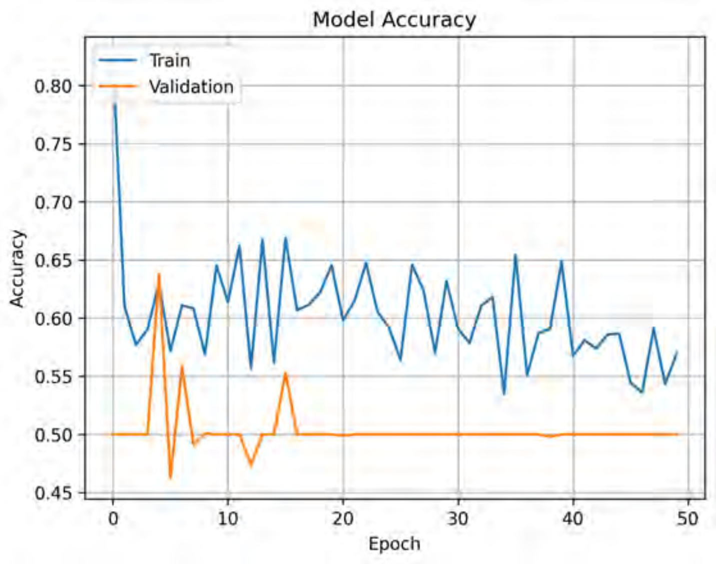


Figure 5.34: ConvMixer Accuracy Curve (Augmented Dataset 10000 Images)

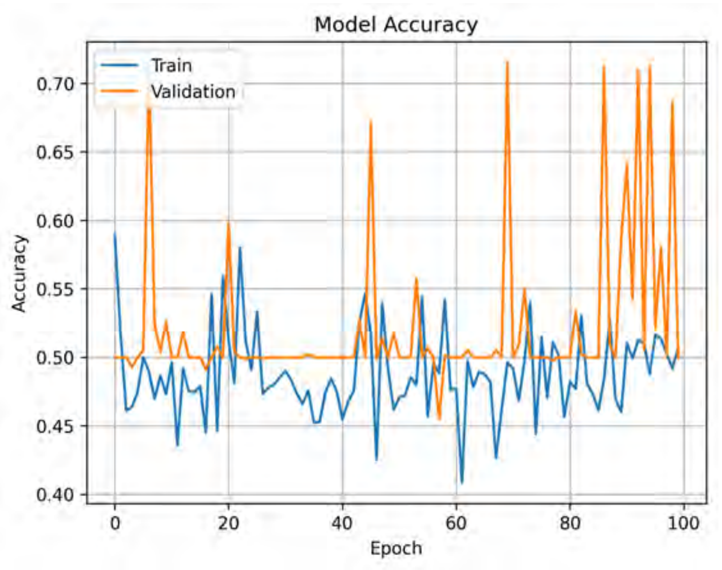


Figure 5.35: ConvMixer Accuracy Curve (Augmented Dataset 4000 Images)

5.2.3.6 EANet

Test Accuracy, Precision, Recall, F1 Score:

EANet showed increment on all four performance metrics upon augmentation. We were only able to run 50 epochs for the augmented dataset due to Google Colab GPU runtime limitation. Further epochs could have increased the metrics more. Here, accuracy rate was increased from 60.54% to 83.5%, F1 score increased from 0.7 to 0.77, precision increased from 0.71 to 0.81 and recall increased from 0.69 to 0.74 in contrast to the original dataset.

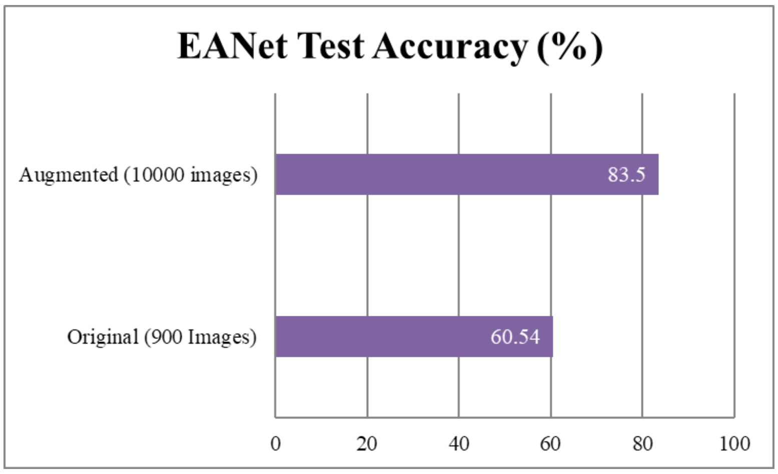


Figure 5.36: EANet Test Accuracy (Original vs Augmented Dataset)

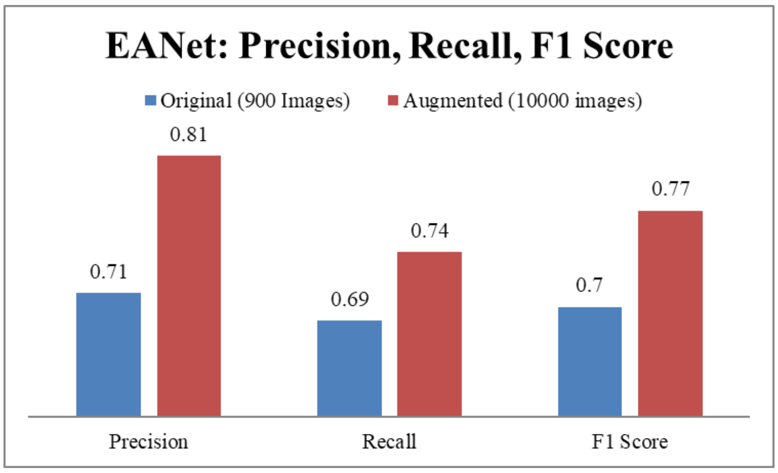


Figure 5.37: EANet Precision, Recall,F1 (Original vs Augmented Dataset)

Loss Curve:

Loss curves for both datasets are free of overfitting and look quite stable. The higher number of spikes in the original dataset picture is due to 4 times higher number of epochs. Augmentation has further stabilised the loss curve.

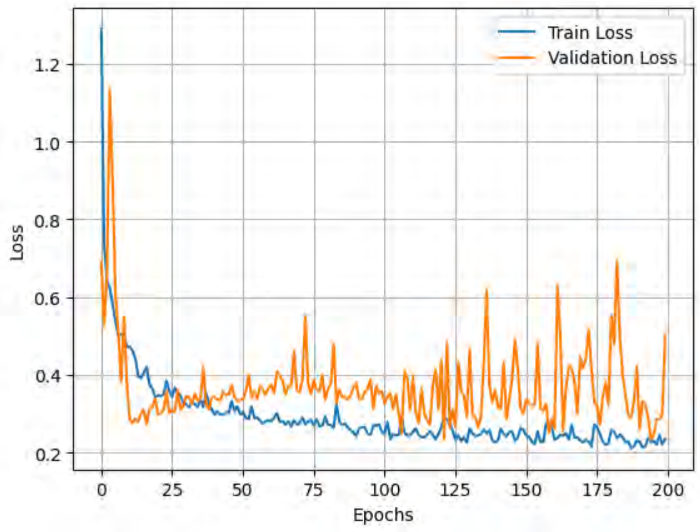


Figure 5.38: EANet Loss Curve (Original Dataset)

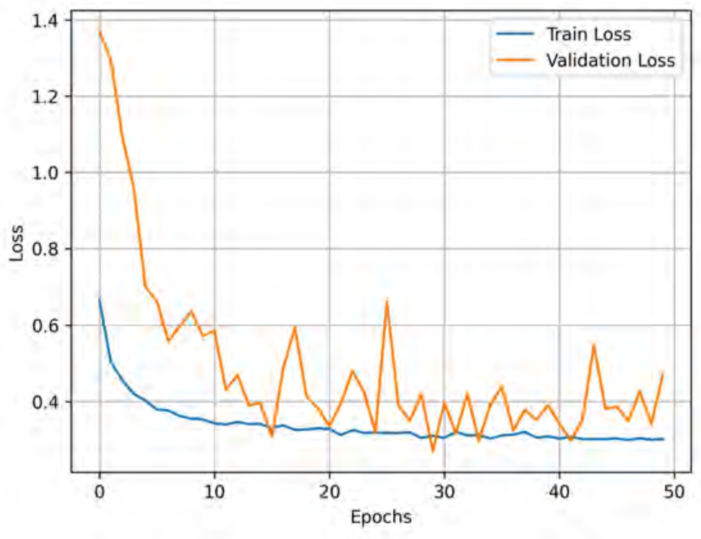


Figure 5.39: EANet Loss Curve (Augmented Dataset)

Accuracy Curve:

Both the accuracy curves look appealingly stable. EANet has produced overall very stable loss and accuracy curves compared to other models. Training further epochs of EANet seems to have great potential in increasing accuracy and other performance metrics.

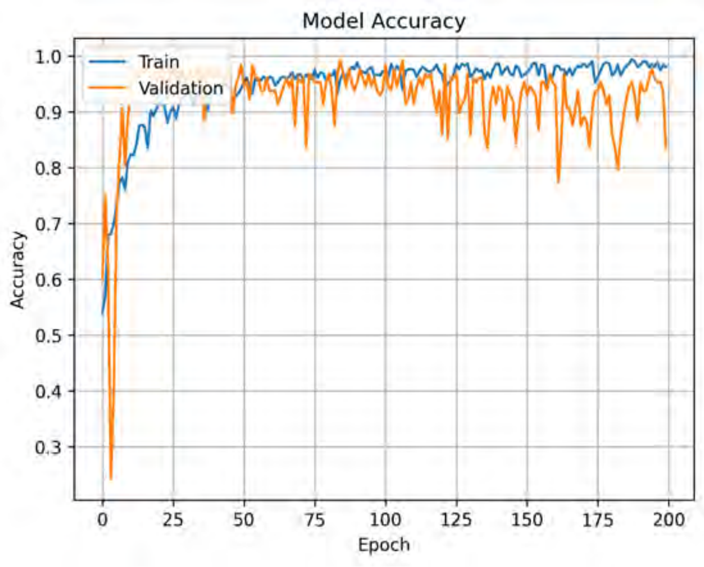


Figure 5.40: EANet Accuracy Curve (Original Dataset)

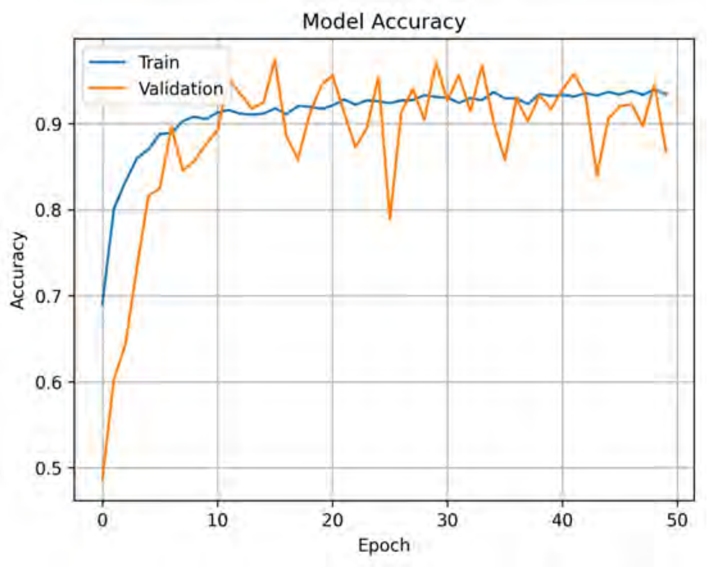


Figure 5.41: EANet Accuracy Curve (Augmented Dataset)

5.2.3.7 VGG16

Test Accuracy, Precision, Recall, F1 Score:

VGG16 has an overall mediocre performance compared to other models in our testing. It performed only better than Xception among the heavyweight models. VGG16 also had an increase in all four performance metrics upon augmentation. Here, the accuracy rate was increased from 62.42% to 78.15%, F1 score increased from 0.64 to 0.78, precision increased from 0.63 to 0.78 and recall increased from 0.65 to 0.78 in contrast to the original dataset.

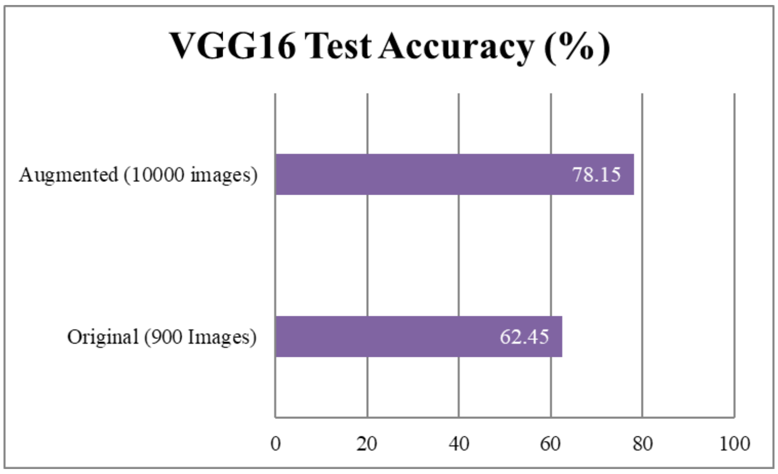


Figure 5.42: VGG16 Test Accuracy (Original vs Augmented Dataset)

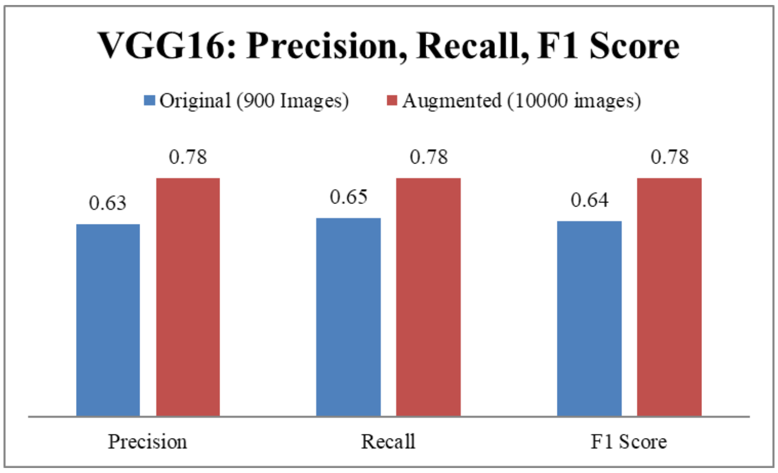


Figure 5.43: Fig: VGG16 Precision, Recall,F1 (Original vs Augmented Dataset)

Loss Curve:

Even though the loss curve has incurred overfitting, it is not as unstable as it looks. If we look carefully we will see the validation loss dips occurred within 0.05 units of the y-axis area. Augmentation has produced a nearly perfect loss curve with very little inconsistency.

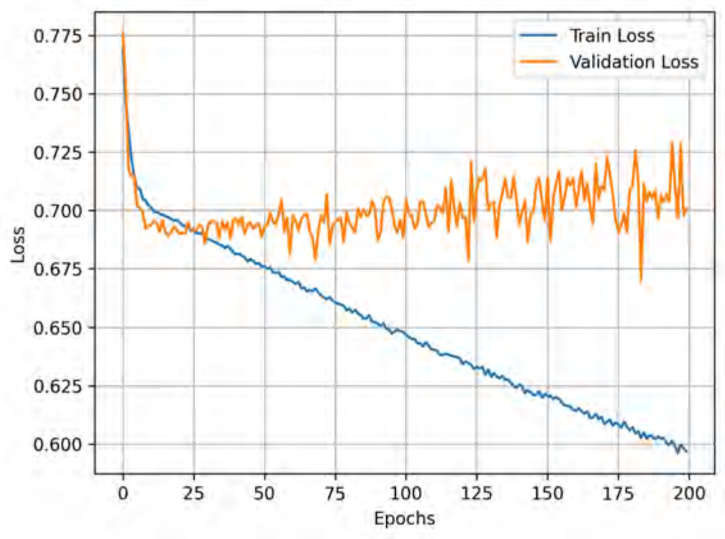


Figure 5.44: VGG16 Loss Curve (Original Dataset)

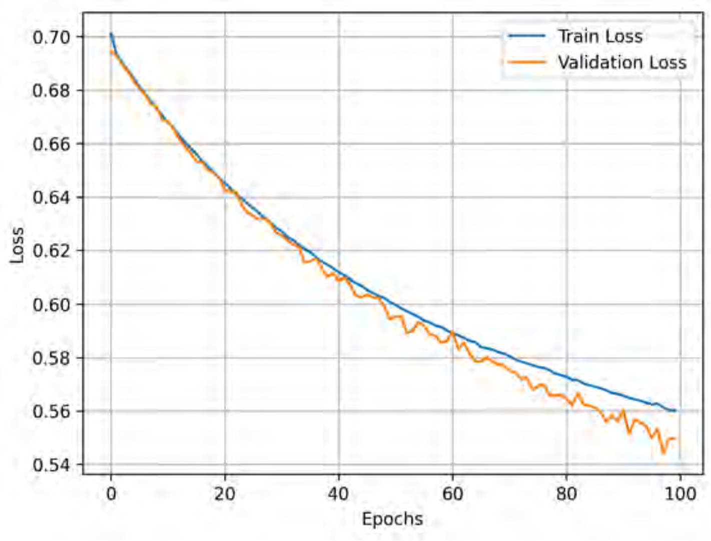


Figure 5.45: VGG16 Loss Curve (Augmented Dataset)

Accuracy Curve:

The accuracy curve has also been stabilised in the augmented dataset, leading to the scope of higher performance values.

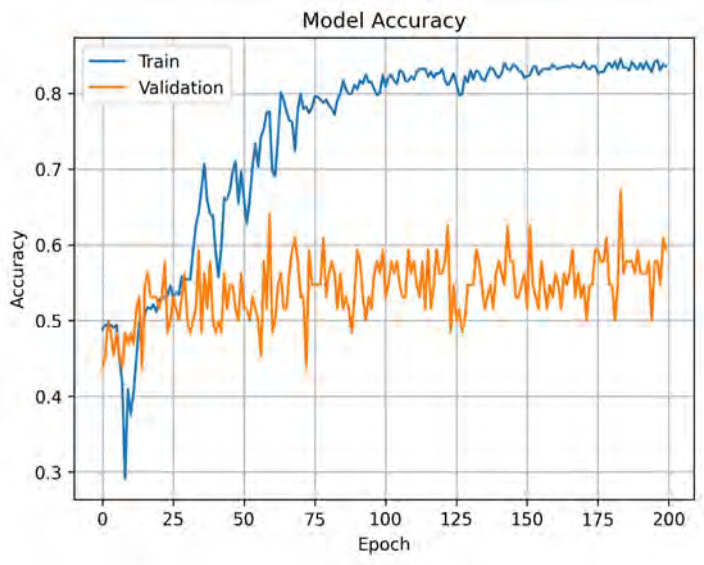


Figure 5.46: VGG16 Accuracy Curve (Original Dataset)

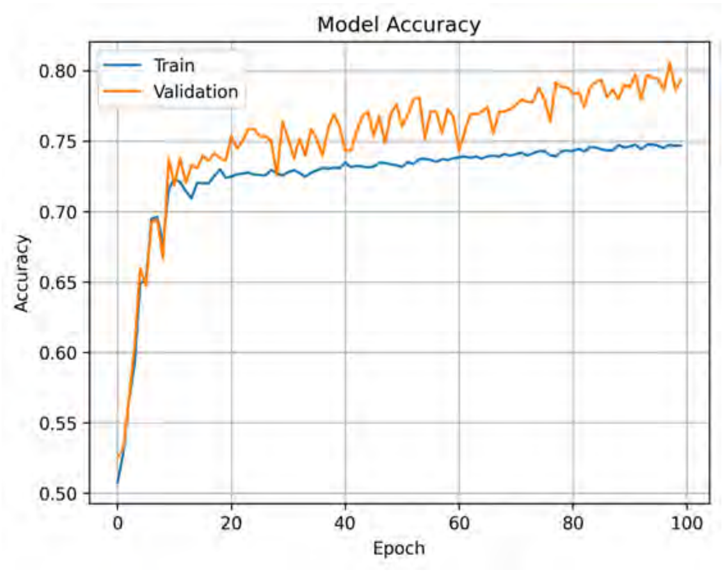


Figure 5.47: VGG16 Accuracy Curve (Augmented Dataset)

5.2.3.8 ResNet50

Test Accuracy, Precision, Recall, F1 Score:

ResNet50 produced the best results among all the heavyweight models tested for both datasets. All four performance metrics increased upon augmentation for ResNet50 also. Here, the accuracy rate increased from 65.13% to 89.6%, F1 score increased from 0.68 to 0.89, precision increased from 0.71 to 0.90 and recall increased from 0.66 to 0.88 in contrast to the original dataset.

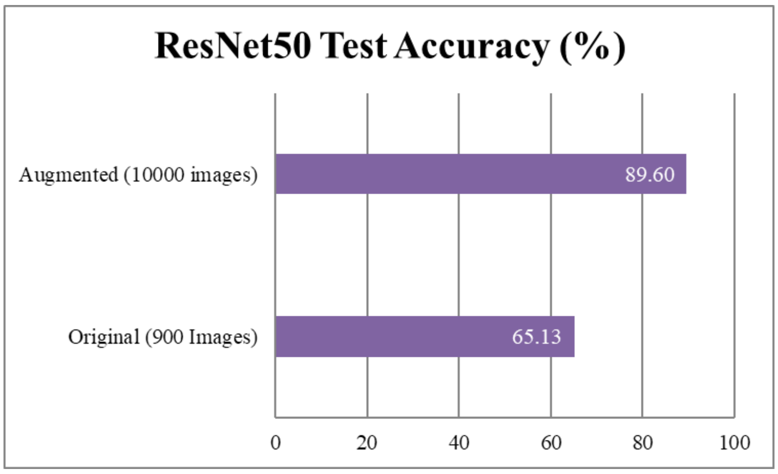


Figure 5.48: ResNet50 Test Accuracy (Original vs Augmented Dataset)

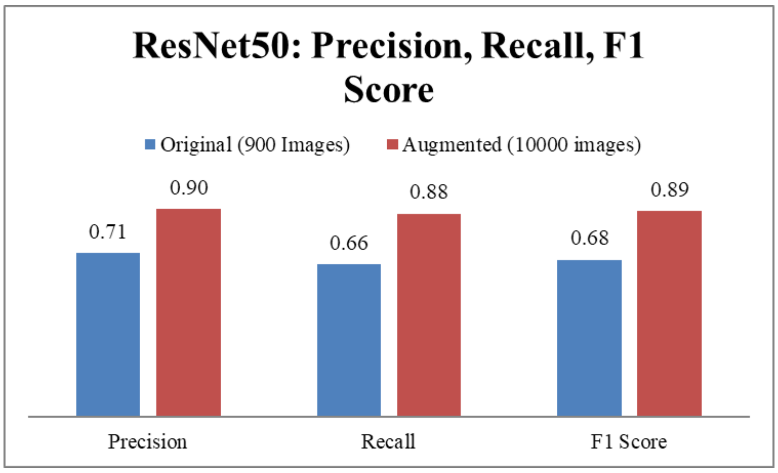


Figure 5.49: ResNet50 Precision, Recall, F1 (Original vs Augmented Dataset)

Loss Curve:

Augmentation turned the loss curve into a nearly perfect curve with slight underfitting. Overfitting was visible while running the model into the original dataset. Augmentation resolved the overfitting.

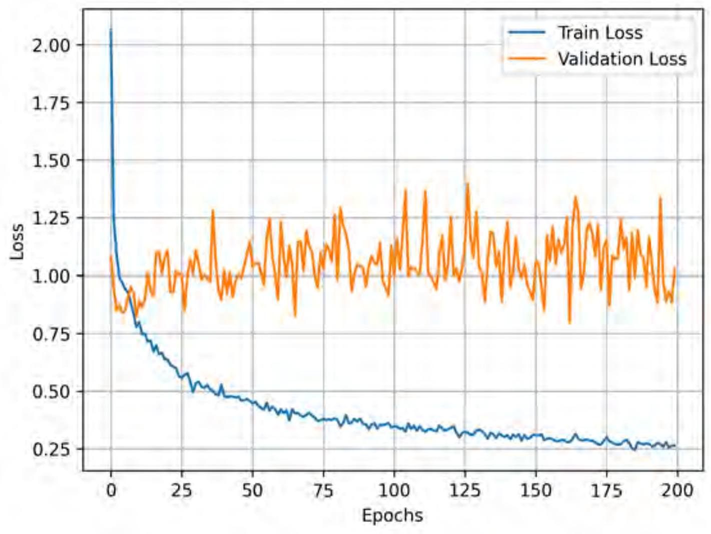


Figure 5.50: ResNet50 Loss Curve (Original Dataset)

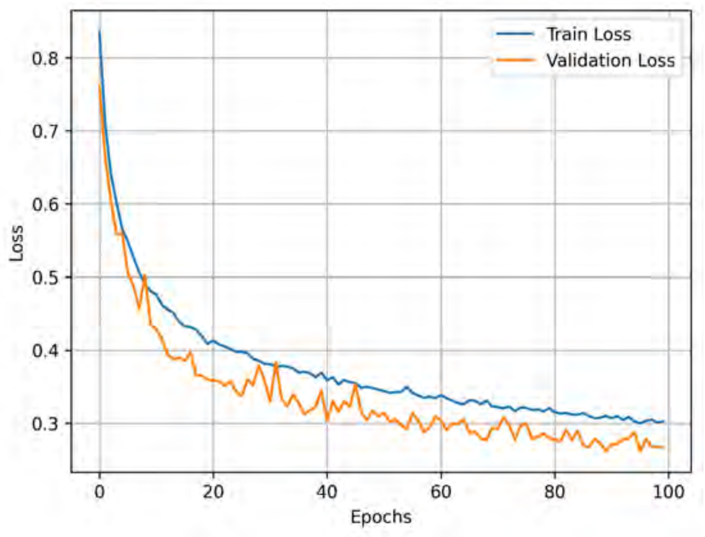


Figure 5.51: ResNet50 Loss Curve (Augmented Dataset)

Accuracy Curve:

The accuracy curve for the original dataset had very low validation accuracy which was resolved in the augmented dataset. The curve for the augmented dataset is almost a perfect accuracy curve with slight spikes.

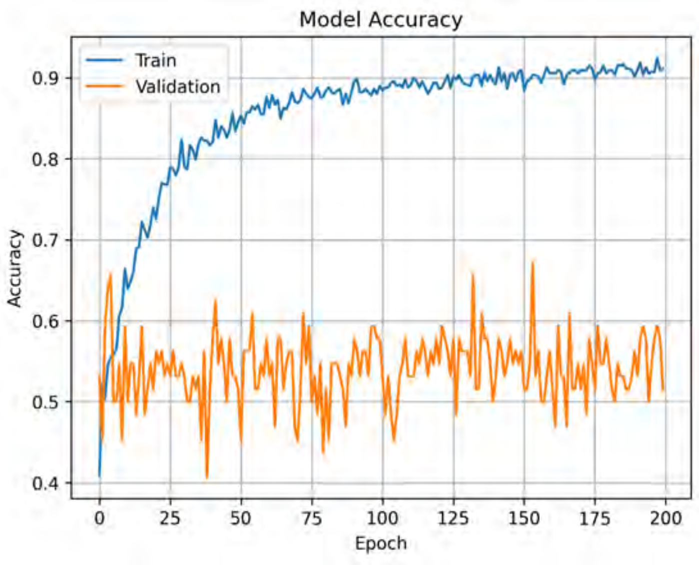


Figure 5.52: ResNet50 Accuracy Curve (Original Dataset)

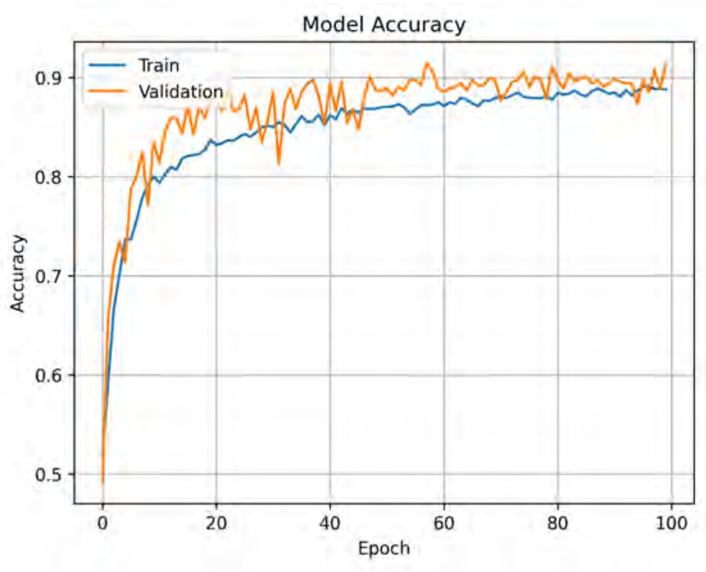


Figure 5.53: ResNet50 Accuracy Curve (Augmented Dataset)

5.2.3.9 DenseNet201:

Test Accuracy, Precision, Recall, F1 Score:

ResNet50 produced the 2nd best results among all the heavyweight models tested for both datasets. All four performance metrics increased upon augmentation for DenseNet201 also. Here, the accuracy rate increased from 63.60% to 88.65%, the F1 score increased from 0.64 to 0.88, precision increased from 0.65 to 0.89 and recall increased from 0.64 to 0.88 in contrast to the original dataset.

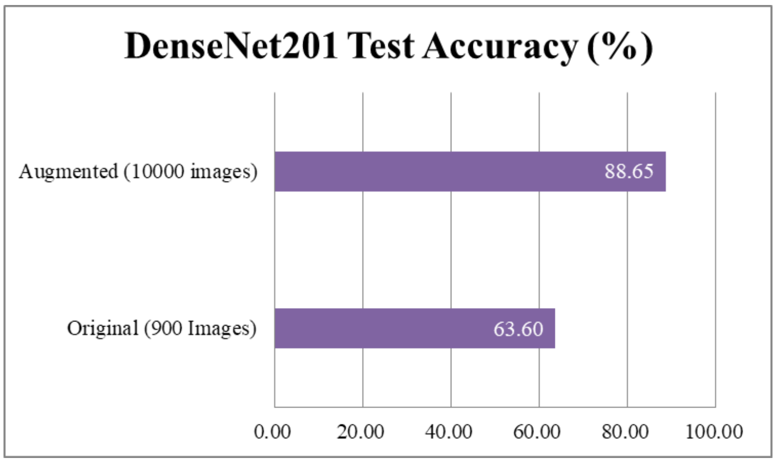


Figure 5.54: DenseNet201 Test Accuracy (Original vs Augmented Dataset)

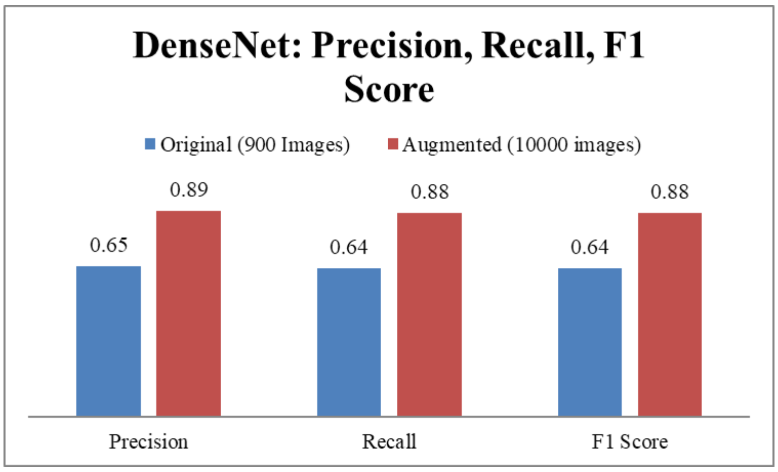


Figure 5.55: DenseNet201 Precision, Recall, F1 (Original vs Augmented Dataset)

Loss Curve:

Like other heavyweight models, the overfitting in the original dataset was resolved and we got a nearly perfect loss curve with slight underfitting upon augmentation.

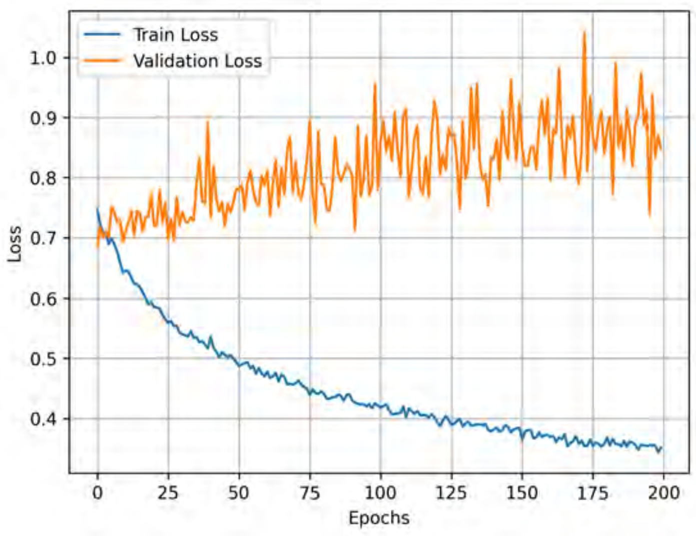


Figure 5.56: DenseNet201 Loss Curve (Original Dataset)

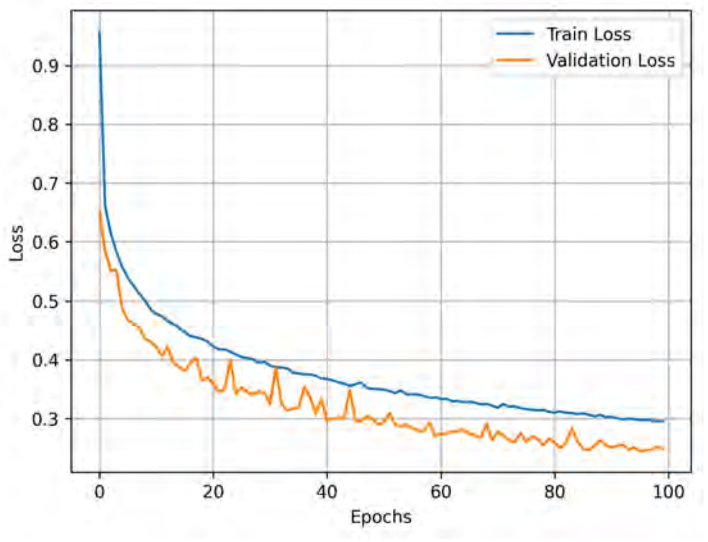


Figure 5.57: DenseNet201 Loss Curve (Augmented Dataset)

Accuracy Curve:

Validation accuracies had been really low for the original dataset which was resolved in the augmented dataset. The augmented accuracy curve exhibited almost similar values for train and validation accuracies in all the epochs.

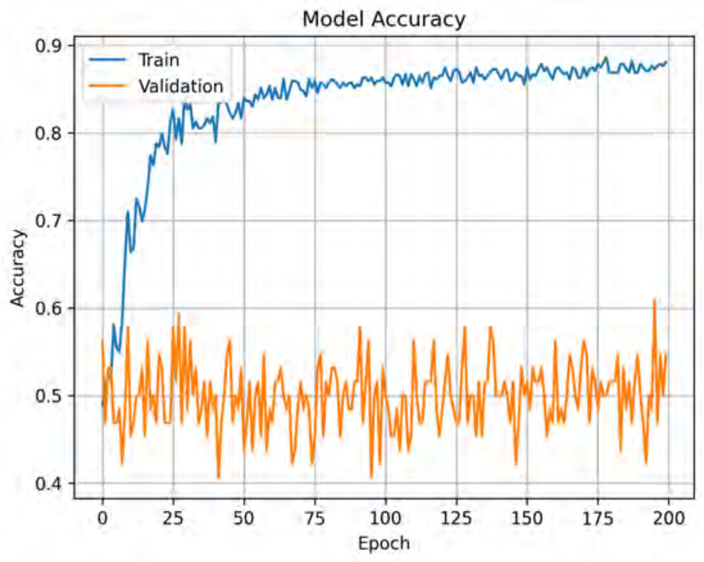


Figure 5.58: DenseNet201 Accuracy Curve (Original Dataset)

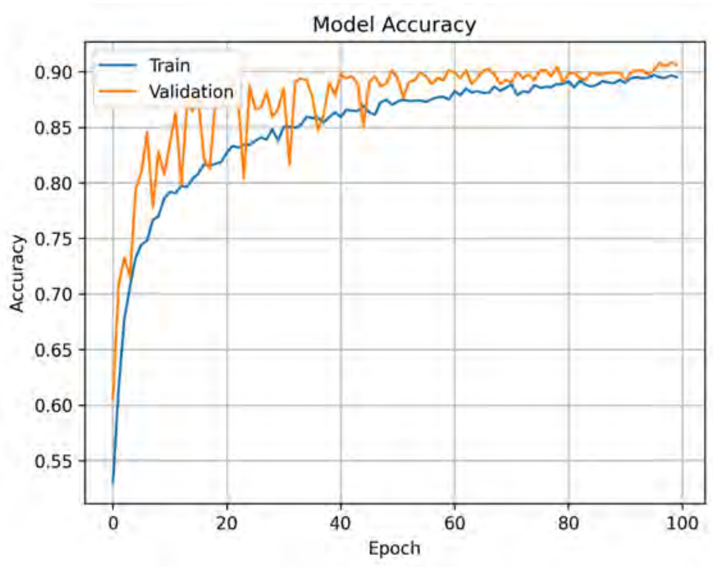


Figure 5.59: DenseNet201 Accuracy Curve (Augmented Dataset)

5.2.3.10 Xception:

Test Accuracy, Precision, Recall, F1 Score:

Xception produced the worst results among the heavyweight models tested. However, all four performance metrics increased upon augmentation for Xception also. Here, the accuracy rate increased from 51.34% to 70.00%, the F1 score increased from 0.5 to 0.7, precision increased from 0.5 to 0.7 and recall increased from 0.5 to 0.7 in contrast to the original dataset.

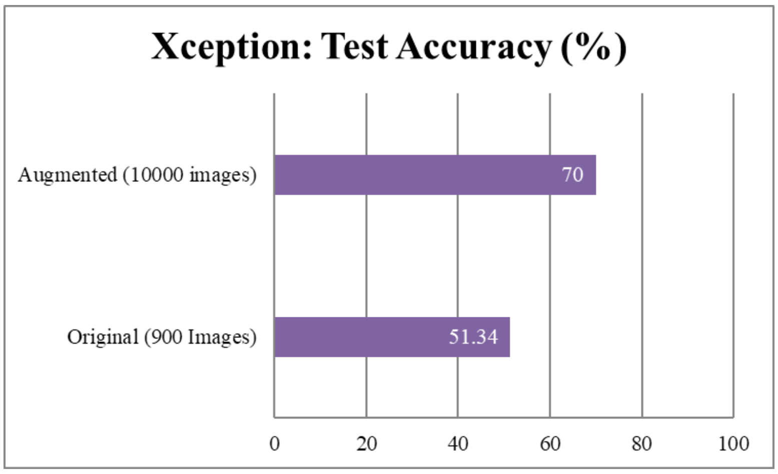


Figure 5.60: Xception Test Accuracy (Original vs Augmented Dataset)

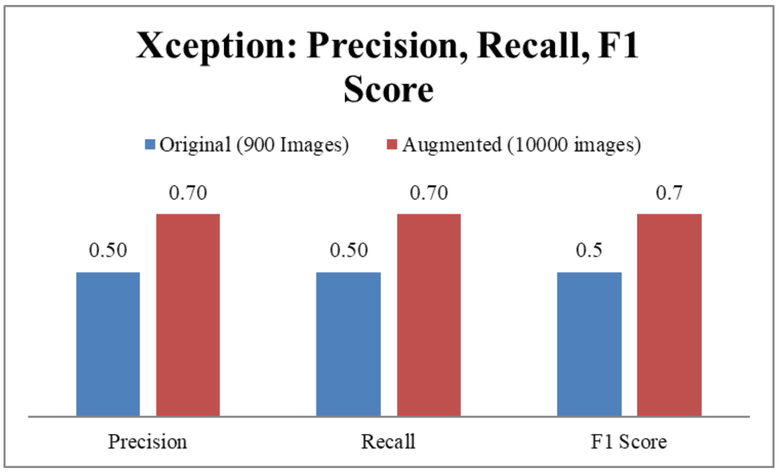


Figure 5.61: Xception Precision, Recall, F1 (Original vs Augmented Dataset)

Loss Curve:

The loss curve for the original dataset had spikes which were converted into a very stable loss curve upon augmentation. The augmented curve exhibits slight underfitting like the other heavyweight models.

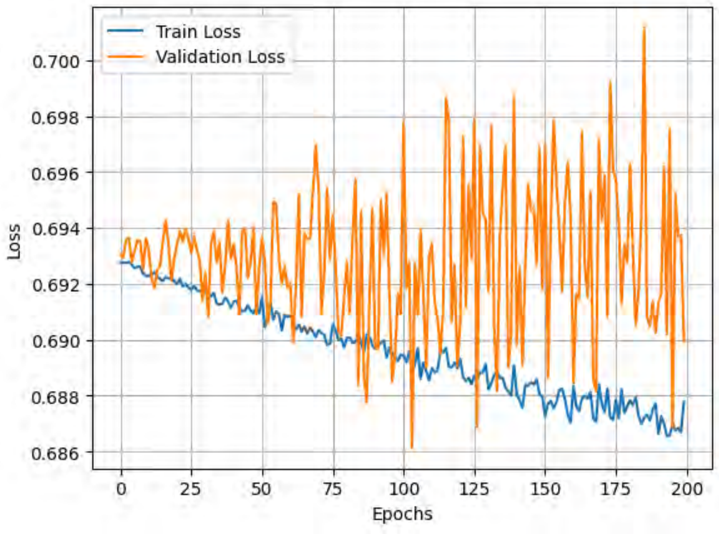


Figure 5.62: Xception Loss Curve (Original Dataset)

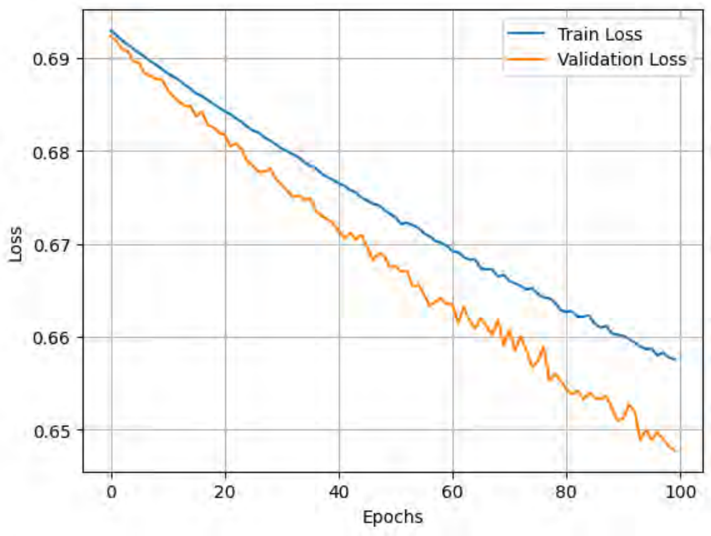


Figure 5.63: Xception Loss Curve (Augmented Dataset)

Accuracy Curve:

Both training and validation accuracy had stayed low throughout the training period in Xception. Augmentation enhanced both the accuracies and presented a stable accuracy curve for the model.

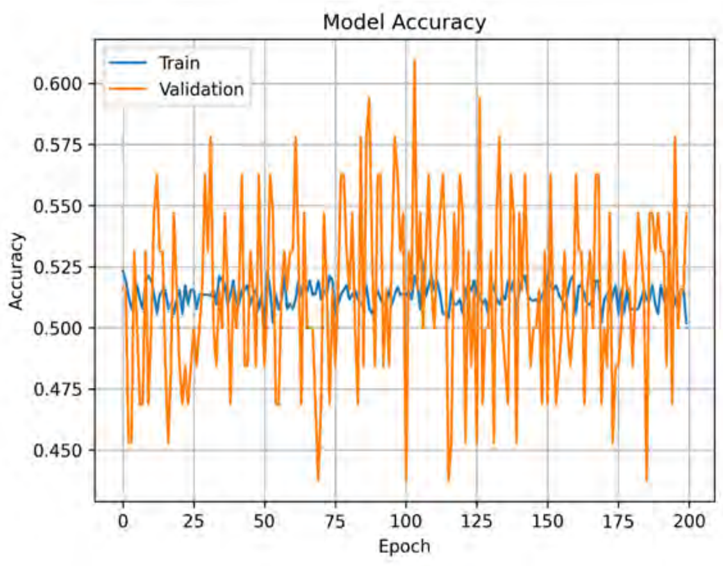


Figure 5.64: Xception Accuracy Curve (Original Dataset)

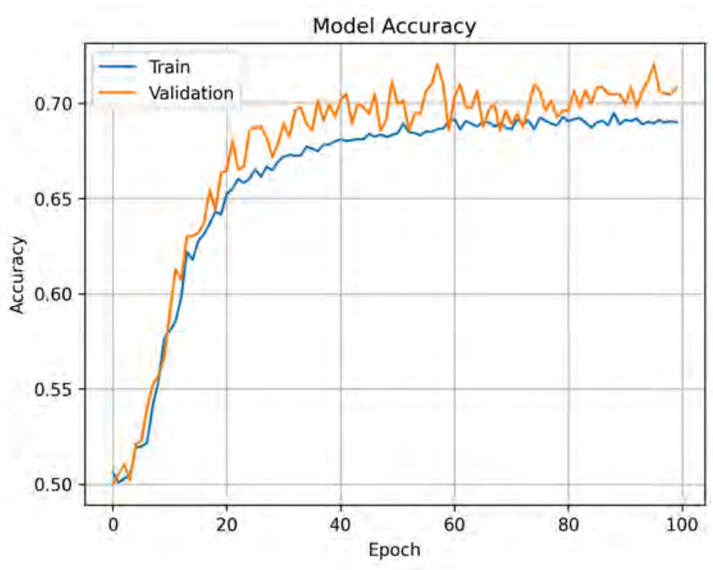


Figure 5.65: Xception Accuracy Curve (Augmented Dataset)

Chapter 6

Discussion

6.1 Research Findings

Summary of our overall research findings are encoded below.

Model Performance

Considering all testing, CNN was the best overall model exhibiting the highest accuracy of 96.65% and F1 score of 0.96. CCT was second second-best model with an accuracy of 94.6% and F1 score of 0.9. Among the heavyweight models, ResNet achieved the highest accuracy of 89.6% and F1 score of 0.89. ConvMixer was the overall worst performing model in our testing having the lowest accuracy of 63.8% and F1 score of 0.53. Among the heavyweight models, Xception gave the lowest impressions with an accuracy of 70% and F1 score of 0.7. Considering model efficiency and optimisation, CNN and INN were the best models with the lowest training times and high enough accuracy.

Contrast between Heavyweight and Lightweight Models

Lightweight models overall performed better than the heavyweight models on both datasets. CNN, CCT and Swin Transformer were the top three among all tested models considering the performance of both datasets. Heavyweight models were more prone to overfitting in the original dataset as per our testing. However, the performance of heavyweight models could further increase if more iterations were executed.

Effects of Augmentation

Augmentation enhanced the performance of 9 out of 10 tested models. All four performance metrics (Test Accuracy, Precision, Recall and F1 Score) were positively incremented on all models upon augmentation. ConvMixer was the only model that was negatively affected by Augmentation. Augmentation also stabilised the loss and accuracy curves of the models. Even ConvMixer curves were improved upon augmentation.

Loss and Accuracy Curves

Augmentation resolved overfitting problems and stabilised the loss and accuracy curves of all tested models. Heavyweight models overall produced the most stable curves upon augmentation.

Importance of Other Performance Metrics Alongside Accuracy

Our study establishes that only considering test accuracy for model evaluation can oftentimes be misleading. For example, ConvMixer gave the highest accuracy over the original dataset while having a low F1 score. It later came out to be the worst performing model in the augmented dataset. Thus, we can infer that F1 score had been the superior performance metric considering this case. Similar patterns were observed in Swin Transformer and EANet where the initial test accuracies were low but F1 scores were high. Both of them performed good in the augmented dataset with Swin Transformer having accuracy of 92.55%. In this case also, F1 score can be inferred as a better performance metric to evaluate a model's potential. That's why, additional performance metrics like F1 score, precision and recall should be considered alongside test accuracy while evaluating models. We have preferred F1 score as the best performance metric as it provides the harmonic mean of both precision and recall. F1 score= 2*(P*R/P+R).

Training Times

Our tests denote that, lightweight models may not always provide the fastest training times. The heavyweight models we have tested trained faster than EANet and ConvMixer models. Both of the models had less than 1 million parameters. Thus, it can be inferred that lower parameters do not always guarantee faster training times. Rather, training time depends more on model optimisation and number of trainable parameters. All four heavyweight models exhibited great optimisation in our testing as they were on par with lightweight models in terms of training time. INN came out be the fastest model with having per epoch time of only 5 seconds for training 10,000 128x128 images. CNN moreover spend per epoch time of only 11 seconds for the same task. All other models are executed per epoch in 50 or more seconds. However, trainings times should not be confused with prediction times. For the prediction or detection of potholes, lightweight models may always perform faster than heavyweight models due to their low parameters.

6.2 Comparative Analysis with Existing Research:

Paper	Tested Models	Preferred Model	Preferred Model Accuracy
Asad et al., 2022	YOLOv1, YOLOv2, YOLOv3, YOLOv4, Tiny-YOLOv4, YOLOv5, and SSD-mobilenetv2	Tiny-YOLOv4	90%
Ahmed et al., 2021	YOLOv5, ResNet50, VGG16, MVGG16, Inception V3, MobileNet V2	ResNet50	91.9%
Shaghouri et al., 2021	SSD-TensorFlow, YOLOv3, and YOLOv4	YOLOv4	Precision: 85%
Wu et al., 2020	Logistic regression, SVM, and Random Forest	Random Forest	Precision: 88.5%
Gayathri et al., 2021	Faster R-CNN, SSD, and YOLO	Faster R-CNN	mAP: 86.41%
Park et al., 2021	YOLOv4, YOLOv4-tiny, and YOLOv5s	YOLOv4-tiny	mAP (0.5): 78.7%
Xin et al., 2023	Threshold-based method, Long short-term memory(LSTM), Random Forest, Optics, and Joint optimisation model	Joint optimisation model	F1 Score: 0.856%
Tamagusko et al., 2023	YOLOv3-tiny, YOLOv3, YOLOv4-tiny, YOLOv4, YOLOv5s, and YOLOv5x	YOLOv4	mAP (0.5): 83.2%
Heo et al., 2023	SPFPN-YOLOv4 tiny, YOLOv4 tiny, YOLO 2, and YOLOv3	SPFPN-YOLOv4 tiny	Precision: 0.89%
Egaji et al., 2021	Naive Bayes, Logistic regression, SVM, KNN, and Random Forest Tree	Random Forest Tree	94.44%
Deepika et al., 2023	YOLOv8, and ResNet50		95%
Arjapure et al., 2020	ResNet50, DenseNet201, ResNet152, InceptionResNEtV2, InceptionV3	DenseNet201	89.66%
Chatterjee et al., 2023	Xception	Xception	96.99%
Parasnis et al., 2023	VGG16	VGG16	95.5%
Pramanik et al., 2021	VGG16, ResNet50	ResNet50	98.66%
Our Thesis	CCT, CNN, INN, Swin Transformer, ConvMixer, EANet, ResNet50, VGG16, DenseNet201, and Xception	CNN and CCT	CNN: 96.65%, F1 Score: 0.96; CCT: 94.6%, F1 Score: 0.9

Table 6.1: Comparative Analysis With Existing Research

Chapter 7

Research Limitation

We ran the models for 200 epochs in the original dataset of 900 images. However, we were only able to run 100 epochs for the augmented dataset of 10,000 images due to Google Colab's GPU Runtime limitation. The training time for the augmented dataset increased 7 to 11 times due to the increase of image samples. Within Google Colab, if the runtime limitation is crossed, then runtime gets disconnected and after that, a model has to be trained from the very beginning with slower CPU runtime. For the same limitation, we were only able to run 50 epochs for EANet and ConvMixer due to their slower training times. We incurred the runtime exhaustion error while running 100 epochs for both models. So, the experimental results might vary for EANet and ConvMixer if they were able to run 100 epochs. Moreover, in our testing, the lightweight models achieved greater accuracy than heavyweight models. However, if training epochs were increased for the heavyweight models, then the performance of the models could also increase. That was out of our scope in the present research. Furthermore, we wanted to include the confusion matrix of all the models, however, we initially tested the models without implementation of the confusion matrix. As the models took hours to train and also we were working with ten models and two datasets, thus, we decided to exclude the confusion matrix from our testing. Lastly, a few of the loss curves in our thesis book exhibit comparably lower resolution compared to other figures due to some errors in Google Colab. Some images were generated in lower quality in Colab and it was not feasible for us to re-run the models to re-generate the curves due to being involved with testing multiple models. That's why we included the images in their original quality.

Chapter 8

Conclusion and Future work

The research has turned out meaningful as we were able to achieve the highest accuracy of 96.65% in pothole classification using lightweight models. Our research also establishes that; lightweight models overall perform better than traditional heavyweight models when compared within similar testing conditions. The findings present a new outlook and consideration towards lightweight models like CNN, CCT and Swin Transformers in detection of potholes. The overall study enhances the acceptability of lightweight models in pothole detection. The experimental results also show that augmentation can enhance classification performance by 15 to 30% depending upon models. Potholes are dangerous road hazards. Utilising lightweight solutions for pothole detection enables compatibility with all kinds of hardware ranging from embedded devices to powerful computers. Lightweight models should become the go to options for real-time pothole detection where availability of the resource are limited.

For future work, we are planning to test all the models for higher number of epochs by Utilising local GPU acceleration instead of using virtual resources of Google Colab. This will enable us to test the models on same number of epochs without worrying about runtime limitations. Moreover, we want to evaluate performance of the models for pothole segmentation. With that, we can compare the results of classification and segmentation to suggest better models for real-world pothole detection. Furthermore, we want to test additional popular models like YOLO, KNN, Random Forest and Logistic Regression to present an all-encompassing research concerning pothole detection. Lastly, we want to share our findings with concerned authorities that can practically apply our research outcomes for the betterment of humanity.

Bibliography

[1] M. Hoque, G. Smith, and S. M. Mahmud, "Safer roads in bangladesh: Addressing the challenges of road infrastructure safety and linear settlements," Sep. 2011.

[2] S. C. Wong, A. Gatt, V. Stamatescu, and M. D. McDonnell, "Understanding data augmentation for classification: When to warp?" In *2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, 2016, pp. 1–6. DOI: 10.1109/DICTA.2016.7797091.

[3] J. Shijie, W. Ping, J. Peiyi, and H. Siping, "Research on data augmentation for image classification based on convolution neural networks," in *2017 Chinese Automation Congress (CAC)*, 2017, pp. 4165–4170. DOI: 10.1109/CAC.2017.8243510.

[4] C. Ng, T. Law, F. Jakarni, and S. Kulanthayan, "Road infrastructure development and economic growth," *IOP Conference Series: Materials Science and Engineering*, vol. 512, no. 1, p. 012 045, 2019. DOI: 10.1088/1757-899X/512/1/012045.

[5] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, p. 60, Jul. 2019, ISSN: 2196-1115. DOI: 10.1186/s40537-019-0197-0. [Online]. Available: https://doi.org/10.1186/s40537-019-0197-0.

[6] V. Chahar, A. Jaiswal, N. Gianchandani, D. Singh, and M. Kaur, "Classification of the covid-19 infected patients using densenet201 based deep transfer learning," *Journal of Biomolecular Structure & Dynamics*, vol. 39, 2020. DOI: 10.1080/07391102.2020.1788642.

[7] J. Dharneeshkar, V. S. Dhakshana, S. A. Aniruthan, R. Karthika, and L. Parameswaran, "Deep learning based detection of potholes in indian roads using yolo," in *International Conference on Inventive Computation Technologies (ICICT)*, Coimbatore, India, 2020, pp. 381–385.

[8] M. Hasan and M. Sobhan, "Highway failure and their maintenance: A study of rajshahi metropolitan city, rajshahi, bangladesh," *International Journal of Sustainable Transportation Technology*, vol. 3, no. 2, pp. 45–50, Oct. 2020. DOI: 10.31427/IJSTT.2020.3.2.2.

[9] C. Wu, Z. Wang, S. Hu, *et al.*, "An automated machine-learning approach for road pothole detection using smartphone sensor data," *Sensors*, vol. 20, no. 19, 2020, ISSN: 1424-8220. DOI: 10.3390/s20195564. [Online]. Available: https://www.mdpi.com/1424-8220/20/19/5564.

[10] K. R. Ahmed, "Smart pothole detection using deep learning based on dilated convolution," *Sensors*, vol. 21, no. 24, 2021, ISSN: 1424-8220. DOI: 10.3390/s21248406. [Online]. Available: https://www.mdpi.com/1424-8220/21/24/8406.

[11] O. A. Egaji, G. Evans, M. G. Griffiths, and G. Islas, "Real-time machine learning-based approach for pothole detection," *Expert Systems with Applications*, vol. 184, p. 115 562, 2021, ISSN: 0957-4174. DOI: https://doi.org/10.1016/j.eswa.2021.115562. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421009684.

[12] A. Islam and Y. Dinar, "Evaluation and spatial analysis of road accidents in bangladesh: An emerging and alarming issue," *Transportation in Developing Economies*, vol. 7, Apr. 2021. DOI: 10.1007/s40890-021-00118-3.

[13] S.-S. Park, V.-T. Tran, and D.-E. Lee, "Application of various yolo models for computer vision-based real-time pothole detection," *Applied Sciences*, vol. 11, no. 23, 2021, ISSN: 2076-3417. DOI: 10.3390/app112311229. [Online]. Available: https://www.mdpi.com/2076-3417/11/23/11229.

[14] A. Paullada, I. D. Raji, E. M. Bender, E. Denton, and A. Hanna, "Data and its (dis)contents: A survey of dataset development and use in machine learning research," *Patterns*, vol. 2, no. 11, p. 100 336, 2021, ISSN: 2666-3899. DOI: https://doi.org/10.1016/j.patter.2021.100336. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666389921001847.

[15] A. Pramanik, M. H. I. Bijoy, and M. S. Rahman, "Detection of potholes using convolutional neural network models: A transfer learning approach," in *2021 IEEE International Conference on Robotics, Automation, Artificial-Intelligence and Internet-of-Things (RAAICON)*, 2021, pp. 73–78. DOI: 10.1109/RAAICON54709.2021.9929623.

[16] A. A. Shaghouri, R. Alkhatib, and S. Berjaoui, *Real-time pothole detection using deep learning*, 2021. arXiv: 2107.06356 `[cs.CV]`.

[17] K. Srinivasan, L. Garg, D. Datta, *et al.*, "Performance comparison of deep cnn models for detecting driver's distraction," *CMC - Tech Science Press*, vol. 68, pp. 4109–4124, 2021. DOI: 10.32604/cmc.2021.016736.

[18] S. Albahli and M. Masood, "Efficient attention-based cnn network (eanet) for multi-class maize crop disease classification," *Frontiers in Plant Science*, vol. 13, p. 1 003 152, Oct. 2022. DOI: 10.3389/fpls.2022.1003152.

[19] S. Bhutad and K. Patil, "Dataset of road surface images with seasons for machine learning applications," *Data in Brief*, vol. 42, p. 108 023, 2022, ISSN: 2352-3409. DOI: https://doi.org/10.1016/j.dib.2022.108023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352340922002347.

[20] Y.-M. Kim, Y.-G. Kim, S.-Y. Son, S.-Y. Lim, B.-Y. Choi, and D.-H. Choi, "Review of recent automated pothole-detection methods," *Applied Sciences*, vol. 12, no. 11, 2022, ISSN: 2076-3417. DOI: 10.3390/app12115320. [Online]. Available: https://www.mdpi.com/2076-3417/12/11/5320.

[21] A. K. Pandey, R. Iqbal, T. Maniak, C. Karyotis, S. Akuma, and V. Palade, "Convolution neural networks for pothole detection of critical road infrastructure," *Computers and Electrical Engineering*, vol. 99, p. 107 725, 2022, ISSN: 0045-7906. DOI: https://doi.org/10.1016/j.compeleceng.2022.107725. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622000398.

[22] K. UÇAR and H. E. KOCER, "Measuring the effect of data augmentation methods for improving the success of convolutional neural network," *GJES (Global Journal of Engineering Science)*, vol. 8, no. 3, pp. 430–438, 2022.

[23] S. Cholet and E. Biabiany, "A framework for frugal supervised learning with incremental neural networks," *Applied Sciences*, vol. 13, p. 5489, 2023. DOI: 10.3390/app13095489.

[24] S. Ghosh and A. Chatterjee, *Automated covid-19 ct image classification using multi-head channel attention in deep cnn*, 2023. arXiv: 2308.00715 [eess.IV].

[25] D.-H. Heo, J.-Y. Choi, S.-B. Kim, T.-O. Tak, and S.-P. Zhang, "Image-based pothole detection using multi-scale feature network and risk assessment," *Electronics*, vol. 12, no. 4, 2023, ISSN: 2079-9292. DOI: 10.3390/electronics12040826. [Online]. Available: https://www.mdpi.com/2079-9292/12/4/826.

[26] S. Hossain, M. T. Reza, A. Chakrabarty, and Y. Jung, "Aggregating different scales of attention on feature variants for tomato leaf disease diagnosis from image data: A transformer driven study," *Sensors*, vol. 23, p. 3751, Apr. 2023. DOI: 10.3390/s23073751.

[27] N. Kumar, L. Krause, T. Wondrak, S. Eckert, K. Eckert, and S. Gumhold, *Learning to Reconstruct the Bubble Distribution with Conductivity Maps using Invertible Neural Networks and Error Diffusion.* Jul. 2023.

[28] S.-Y. Lee, T. H. M. Le, and Y.-M. Kim, "Prediction and detection of potholes in urban roads: Machine learning and deep learning based image segmentation approaches," *Developments in the Built Environment*, vol. 13, no. 100109, p. 100 109, 2023. DOI: 10.1016/j.dibe.2022.100109.

[29] G. Parasnis, A. Chokshi, V. Jain, and K. Devadkar, *Roadscan: A novel and robust transfer learning framework for autonomous pothole detection in roads*, 2023. arXiv: 2308.03467 [cs.CV].

[30] H. Tahir and E.-S. Jung, "Comparative study on distributed lightweight deep learning models for road pothole detection," *Sensors*, vol. 23, no. 9, 2023, ISSN: 1424-8220. DOI: 10.3390/s23094347. [Online]. Available: https://www.mdpi.com/1424-8220/23/9/4347.

[31] T. Tamagusko and A. Ferreira, "Optimizing pothole detection in pavements: A comparative analysis of deep learning models," *Engineering Proceedings*, vol. 36, no. 1, 2023, ISSN: 2673-4591. DOI: 10.3390/engproc2023036011. [Online]. Available: https://www.mdpi.com/2673-4591/36/1/11.

[32] S. Whang, Y. Roh, H. Song, and J.-G. Lee, "Data collection and quality challenges in deep learning: A data-centric ai perspective," *The VLDB Journal*, vol. 32, Jan. 2023. DOI: 10.1007/s00778-022-00775-9.

[33] H. Xin, Y. Ye, X. Na, *et al.*, "Sustainable road pothole detection: A crowd-sourcing based multi-sensors fusion approach," *Sustainability*, vol. 15, no. 8, 2023, ISSN: 2071-1050. DOI: 10.3390/su15086610. [Online]. Available: https://www.mdpi.com/2071-1050/15/8/6610.

[34] P. Yao, T. Mao, M. Shi, J. Sun, and Z. Wang, *Eanet: Expert attention network for online trajectory prediction*, 2023. arXiv: 2309.05683 `[cs.LG]`.

[35] https://www.dhakatribune.com/bangladesh/dhaka/327544/pothole-ridden-roads-bring-traffic-to-a-standstill, [Accessed 07-01-2024].

[36] *A comparative study of multiple neural network for detection of COVID-19 on chest X-ray - EURASIP Journal on Advances in Signal Processing — asp-eurasipjournals.springeropen.com*, https://asp-eurasipjournals.springeropen.com/articles/10.1186/s13634-021-00755-1, [Accessed 08-01-2024].

[37] admin, *Keras VGG16 Model Example - PyTorch & Keras — androidkt.com*, https://androidkt.com/keras-vgg16-model-example/, [Accessed 09-01-2024].

[38] Author, *Roads in Dumki are full of potholes and accidents happen frequently — bangladesh.postsen.com*, https://shorturl.at/gjlRS, [Accessed 07-01-2024].

[39] *Basic CNN Architecture: Explaining 5 Layers of Convolutional Neural Network — upGrad blog — upgrad.com*, https://www.upgrad.com/blog/basic-cnn-architecture/, [Accessed 09-01-2024].

[40] *CNN Architecture - Detailed Explanation — interviewbit.com*, https://www.interviewbit.com/blog/cnn-architecture/, [Accessed 09-01-2024].

[41] *Crumbling roads pose accident risks in Habiganj — thedailystar.net*, https://www.thedailystar.net/news/bangladesh/news/crumbling-roads-pose-accident-risks-habiganj-3404821, [Accessed 07-01-2024].

[42] *Deadly pothole on Dhaka-Aricha highway finally filled — bangladeshpost.net*, https://bangladeshpost.net/posts/deadly-pothole-on-dhaka-aricha-highway-finally-filled-62623, [Accessed 07-01-2024].

[43] *Deep Residual Learning for Image Recognition — arxiv.org*, https://arxiv.org/abs/1512.03385v1, [Accessed 08-01-2024].

[44] *Densely Connected Convolutional Networks — arxiv.org*, https://arxiv.org/abs/1608.06993, [Accessed 08-01-2024].

[45] *DenseNet-201 and Xception Pre-Trained Deep Learning Models for Fruit Recognition — mdpi.com*, https://www.mdpi.com/2079-9292/12/14/3132, [Accessed 08-01-2024].

[46] *DenseNet-201 convolutional neural network - MATLAB densenet201 — mathworks.com*, https://www.mathworks.com/help/deeplearning/ref/densenet201.html, [Accessed 08-01-2024].

[47] *EANet: Edge-Aware Network for the Extraction of Buildings from Aerial Images — mdpi.com*, https://www.mdpi.com/2072-4292/12/13/2161, [Accessed 09-01-2024].

[48] *Government of the people's republic of bangladesh, available link:GOVERNMENT OF THE PEOPLE'S REPUBLIC*, Accessed 07-01-2024
.

[49] *Ijrpr.com*, https://ijrpr.com/uploads/V4ISSUE10/IJRPR18446.pdf, [Accessed 08-01-2024].

[50] *Ijrte.org*, https://www.ijrte.org/wp-content/uploads/papers/v8i6/F7349038620.pdf, [Accessed 08-01-2024].

[51] A. Kumar, *Different Types of CNN Architectures Explained: Examples — vitalflux.com*, https://vitalflux.com/different-types-of-cnn-architectures-explained-examples/, [Accessed 09-01-2024].

[52] *Large scale performance analysis of distributed deep learning frameworks for convolutional neural networks - Journal of Big Data — journalofbigdata..com*, https://journalofbigdata.springeropen.com/articles/10.1186/s40537-023-00765-w, [Accessed 08-01-2024].

[53] C. Martins, *Understanding Xception: Putting Inception and Residual Networks Together — cdanielaam.medium.com*, https://cdanielaam.medium.com/understanding-xception-putting-inception-and-residual-networks-together-8376e009f803, [Accessed 08-01-2024].

[54] *Over 3,000kms of roads poor, bad or very bad — thedailystar.net*, https://www.thedailystar.net/bangladesh/news/over-3000kms-roads-poor-bad-or-very-bad-2119633, [Accessed 07-01-2024].

[55] S. Park, *ConvMixer: Patches Are All You Need? Overview and thoughts — medium.com*, https://medium.com/codex/an-overview-on-convmixer-patches-are-all-you-need-8502a8d87011, [Accessed 09-01-2024].

[56] S. Park, *Swin Transformers: The most powerful tool in Computer Vision — sieunpark77.medium.com*, https://sieunpark77.medium.com/swin-transformers-the-most-powerful-tool-in-computer-vision-659f78744871, [Accessed 09-01-2024].

[57] *Pothole*, https://www.collinsdictionary.com/dictionary/english/pothole, [Accessed 07-01-2024].

[58] R. Pramoditha, *Convolutional Neural Network (CNN) Architecture Explained in Plain English Using Simple Diagrams — towardsdatascience.com*, https://towardsdatascience.com/convolutional-neural-network-cnn-architecture-explained-in-plain-english-using-simple-diagrams-e5de17eacc8f, [Accessed 09-01-2024].

[59] E. Randellini, *Image classification: ResNet vs EfficientNet vs EfficientNet_v2 vs Compact Convolutional... — enrico.randellini*, https://medium.com/@enrico.randellini/image-classification-resnet-vs-efficientnet-vs-efficientnet-v2-vs-compact-convolutional-c205838bbf49, [Accessed 09-01-2024].

[60] *Understanding VGG16: Concepts, Architecture, and Performance — datagen.tech*, https://datagen.tech/guides/computer-vision/vgg16/, [Accessed 09-01-2024].

[61] *XCeption Model and Depthwise Separable Convolutions — maelfabien.github.io*, https://maelfabien.github.io/deeplearning/xception/, [Accessed 08-01-2024].