Team Brogrammers

CE784A Project Report: Semester 2020-21 (I)

Pavement Crack Detection and Segmentation Using Deep Learning

Aastha Ramteke, Akarsh Mittal, Lakshay Middha, Naman Jain, Neha Aggarwal

Abstract

Automated pavement crack detection has been under research for a long time due to the different texture and nature of crack in the pavement in real world scenarios. In this paper, we propose a deep learning based approach which has capability of dealing with different pavement conditions. Here, we will use Inception V3 architecture to classify the image into cracked one (i.e positive class) and non- cracked one (i.e negative class). Traditional methods usually fail to extract accurate crack information from pavement images. Therefore, we have proposed a convolution-deconvolution architecture called U-Net. We were able to achieve a maximum IOU of 0.5 and f1 score of 0.65

Keywords: Crack detection; Deep Learning; Convolutional Neural Network; Segmentation;Inception V3; Xception;U Net

1. Introduction

Cracks are fracture lines caused due to temperature change, moisture expansion, elastic deformation, settlement, or creep. Due to intensive use of roads, cracks arise in the pavements, which can cause serious damage in extreme conditions. According to a study, around 16% of road accidents occur due to poor pavement conditions. The maintenance of huge channels of roads is quite difficult and traditional crack detection methods depend on manual work and are laborious, hence new tools are required to automatically detect pavement cracks. Here we are proposing a solution to this problem - **Pavement Crack Detection and Segmentation,** using deep learning models. Firstly we will classify the image as cracked or uncracked and then we will run a pixel level segmentation model to locate the crack. To address these challenges,we proposed a deep learning based cnn model for pavement crack detection which is effective in achieving high validation accuracy via a high level feature extraction.Automated crack segmentation is a very challenging task with a goal of accurately marking crack layers. The CNN model acquires expressive features at different layers due to several trainable layers.
Periodic survey of pavement conditions and distress levels are important for long term pavement performance.
Old methods are time consuming labour intensive and less accurate.In present times, the expectations of highly accurate, efficient and comprehensive pavement distress assessment have increased. Due to advancement in computer technology, it is possible to develop models with satisfactory performance which are important for overall pavement performance assessment. Therefore, there is a need to develop automatic pavement crack detection models.
The method we propose helps to evaluate road distress quality and early detection of damages in the pavements,thus reducing the maintenance and restructuring costs. This idea can be extended to structural damage detection in buildings or detecting cracks in vehicle tires too.

2. Literature Review

Review of Traditional Methods

The development of this technology can be parted into three stages: 1. Traditional manual detection technology has totally relied on manual data acquisition, and its evaluation also depends on subjective mode. 2. Semi-automatic detection technology is the process through which an image acquisition system stores the collected data on hard disk equipment, and then the cracks are tracked and analysed manually. 3. Full automation technology refers to the analysis, identification, evaluation and judgment of cracks based on semi-automation technology and image processing algorithms.

Machine Learning uses feature extraction and pattern recognition for crack detection. Many scholars worked on crack detection including [1]Oliveira et al. use the mean and the standard deviation for unsupervised learning to distinguish blocks with crack from blocks without crack.In CrackForest,[2] Shi et al. proposed a new descriptor based on random structured forests to characterize cracks. These methods are not suitable for complicated pavement conditions, thus not effective for all pavements.To solve the above problem deep learning is a successful method for damage detection. [3]Cha et al. use a sliding window to divide the image into blocks and CNN is used to predict whether the block contains cracks or not. But the above method can only find patch level cracks without considering the pixel level. In [4-5], Zhang et al. use CNN to predict whether an individual pixel belongs to crack based on the local patch information. However, the method ignores the spatial relations between pixels and overestimates crack width. In Zhang et al. use CNN to predict the class for each pixel of the image. However, it still needs manually designed feature extractors for preprocessing, and CNN is only used as a classifier. Besides, its network architecture is strictly related to the input image size, which prevents generalization of the method. Our method also uses CNN, here we extract features from the raw images without their pre-processing. Images are converted to patches whose results are then used to predict whether cracks in pavement exist or not.

Review of New Methods

Classification only helps us to know whether there is a crack present or not, but segmentation helps us to localize the crack and obtain comprehensive crack characteristics. A pixel wise mask is created by image segmentation for each object in the image This technique provides us with far more granular understanding of the objects in the image. There are 3 broad levels of automatic pavement crack detection. 1. Image-level crack detection, which analyzes crack distributions and other characteristics based on analysis of the pavement surface texture conditions. 2. Block level pavement crack detection technique is another type of method for crack detection similar to object detection from given images.This crack analysis provides fast and accurate locating and counting of the surface cracks along the specific monitored road section. However,it has difficulty in providing accurate information about individual crack length, width, severity, and other parameters, which are important values for a comprehensive pavement condition evaluation. 3. Pixel-level or pixel wise pavement crack detection, which can provide precise crack parameters for pavement condition assessment.

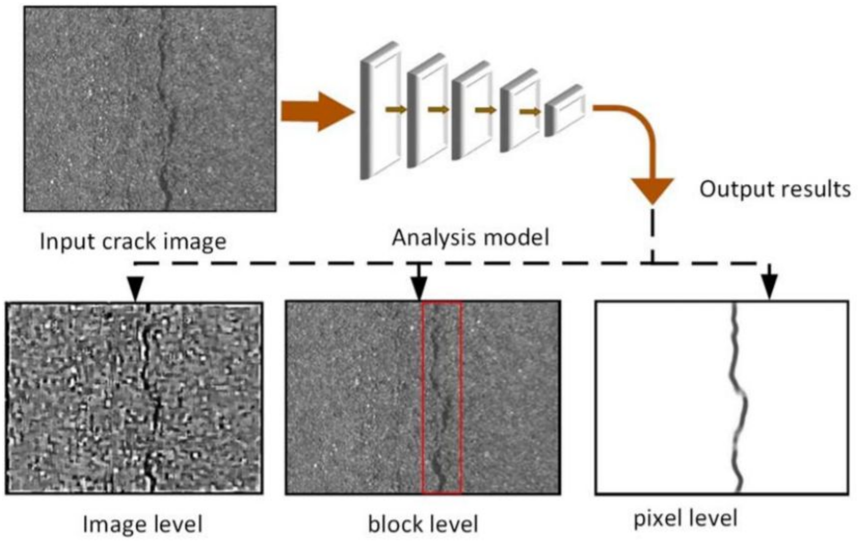


Figure 1 : Pixel-level or pixel wise pavement crack detection

Particularly, deep Convolutional Neural Networks (CNN) have demonstrated successes in large-scale object recognition problems (Krizhevsky et al., 2012; Zeiler and Fergus, 2013; Sermanet et al., 2014; Szegedy et al., 2014; Simonyan and Zisserman, 2015; He et al., 2015).Other convolutional networks were also proposed for pixelwise classifications (Pinheiro and Collobert, 2015; Shelhamer et al., 2016), which all include pooling layers, and thus inevitably lose original data.The pooling layers replaced with convolution layers with larger strides (Springenberg et al., 2015) resulting in similar spatial reductions and data losses in the All Convolutional Net.
The visible geometric features of cracks can be available regarding the shape, orientation, length, and width for which Pixel-perfect accuracy is critical for pavement crack detection.1)The pixel-perfect accuracy yields accurate identifications on the types of detected cracks like longitudinal cracks, transverse cracks, or alligator cracks etc. 2)Important indicators to identify its severity level are actual width, length and extent of a crack. 3)For timely monitoring on the developing behaviors of pavement cracks, the accurate measurements of cracks are useful.

Zhang et al. (2016b) proposed a CNN with 4 convolution layers, 4 max-pooling layers and 2 fully connected layers for crack detection.An overestimation of crack width was observed when the class label assigned to an individual pixel was still based on the local context around the pixel. It was shown in their study that deep CNN outperformed traditional machine learning techniques (i.e., SVM and Boosting method) for pavement crack detection. The classic architecture of AlexNet (Krizhevsky et al., 2012) was adopted to classify an input image as crack image or no-crack image (Some, 2016), and thus was incapable of detecting cracks at pixel level. Cha et al. (2017) also developed a deep CNN for piecewise classification on cracks. With the use of sliding window techniques, the class label was assigned to each small patch of size 256 × 256. It was demonstrated that their method achieved a high level of accuracy in classifying a small image patch as cracked or intact even under complex illuminations. However, the actual location of cracks in the small patch was not considered in their study, and the pixel-perfect accuracy was thus unattainable. Although the recent CNN based methodologies have limitations in terms of pixelperfect accuracy, they all reveal the great potential of deep CNN in detecting pavement cracks.
In summary, the need of the future is pixel perfect crack detection, hence our research is inclined towards implementing those high performance models and if possible to improve the results.

3. Data set Details

Our dataset contains 11298 images and their corresponding segmented masks.. Of which we did a 85:15 (train + validation) : test split. Train + validation dataset contains 9603 images of which 8404 are cracked and 1199 are non-cracked. Test dataset contains 1695 images of which 1483 are cracked and 212 are non-cracked. The splitting is stratified so that the proportion of each dataset in the train and test folder are similar with approximately 7:1 ratio for cracked to non-cracked images.

Type	Train+Validation	Test	Total
Cracked	8404	1483	9887
Non-Cracked	1199	212	1411
Total	9603	1695	**11298**

Table 1: Dataset Details

All the images are resized to the size of (448, 448). The name prefix of each image is assigned to the corresponding dataset name that the image belongs to. There're also images with no crack pixel, which could be filtered out by the file name pattern "non crack*". To save time, we manually filtered out the non-cracked images.

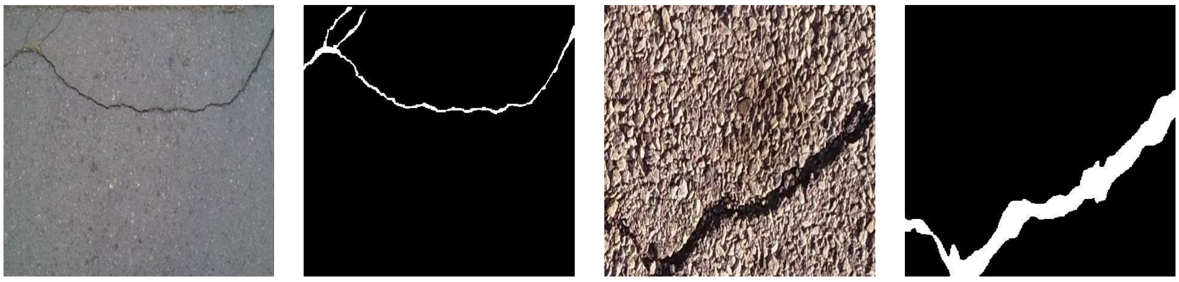

CFD Cracked   CFD Mask   CRACK500 Cracked   CRACK500 Mask

Figure 2 :Images and Masks from the dataset

4. Methodology

We have increased the diversity of our training dataset by applying random (but realistic) transformations such as image rotation via data augmentation. We introduced rescaling to resize the input in the [0, 255] range to be in the [0, 1] range. The conv2D is the traditional convolution that creates a convolution kernel that is convolved which takes layers of input to produce tensors of outputs. So, we have an image, with padding, and filter that slides through the image with a given stride. On the other hand, the SeparableConv2D is a variation of the traditional convolution that was used to compute it faster. It performs a depthwise spatial convolution followed by a pointwise convolution which mixes together the resulting output channels. Batch Normalisation normalizes the activations of the previous layer at each batch, i.e. applies a transformation that maintains the mean activation close to 0 and the activation standard deviation close to 1. Here we use activation function as “ReLU”(Rectified Linear Unit activation function),this activation layer is kept at the end(sometimes even between) of Neural Network to conclude whether a neuron should pass to other network as an input or not.Here we used “ReLU” which has the following inputs:(a)max_value:Default as none,(b)negative_slope:Default to 0,(c)threshold:Default to 0,it returns an element wise max(x,0). We have also used maxpooling 2D layer which takes the following as input (a)pool_size:a tuple of 2 integer which is the window size on which we take the maximum,(b)strides:tuple of 2 integers,it tells how far the pooling window moves for each pooling step,(c)padding:which can either be valid(means no padding) or same,(d)data_format:a string either of channels_last (default) or channels_first,it returns a tensor of rank 4 representing maximum pooled values.We have also used sigmoid function(= 1 / (1 + exp(-x))) as one of our activation layer,its arguments are x= input tensor and returns value according to the sigmoid function,i.e, output = 1 / (1 + exp(-x))

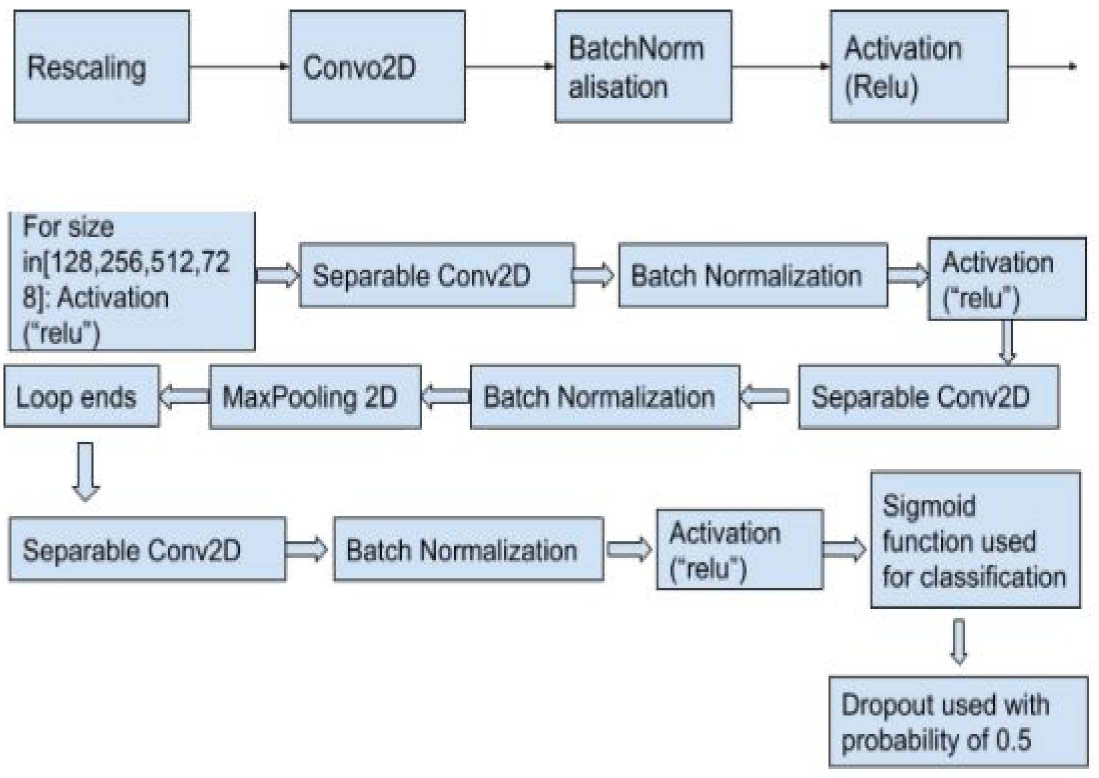


Figure 3: Architecture for model 1

Transfer Learning:

Our intuition behind transfer learning for image classification is that if a model is trained on a large and general enough dataset, this model will effectively serve as a generic model of the visual world. We can then take advantage of these learned feature maps without having to start from scratch by training a large model on a large dataset. We have used the representations learned by a previous network to extract meaningful features from new samples. We have added a new classifier, which will be trained from scratch, on top of the pretrained model.

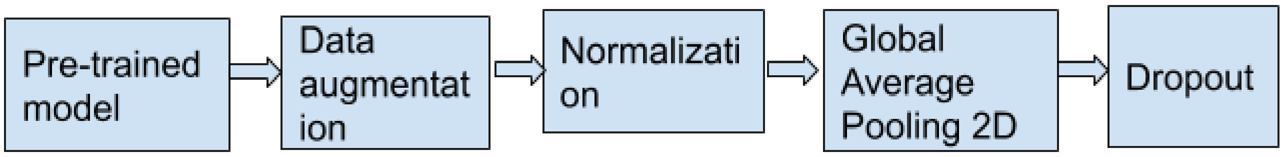


Figure 4: Transfer Learning Procedure

***Dropout input** = 0.2 (i.e. ignoring 20% of the neurons in the particular layer in the Neural Network where we have used dropout.)

Fine-Tuning :

Fine-Tuning of the model can be done through three strategies:

1) Train entire model: In this, the architecture of the pre-trained model is trained according to the dataset. A large dataset is required to learn the model from scratch.
2) Train some layers and leave others frozen: In case of small dataset and large parameters leave more layers frozen to avoid overfitting. If dataset is large and number of parameters is small,model can be improved by training more layers since overfitting will be an issue,
3) Freeze the convolutional base: In this convolutional base is kept to its original form and then its outputs are fed to classifiers. In this pre-trained model is used as a fixed feature extraction mechanism, useful when the dataset is small.

   Figure represents these three strategies in schematic way

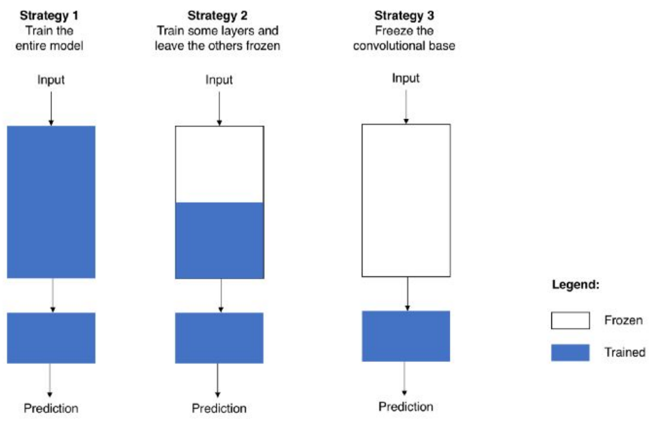


Figure 5 - 3 Strategies for Fine-Tuning

We choose strategy (3) for fine tuning our model, i.e. we freezed the convolutional base.

Till now, we have used two pre-trained models (Inception V3 and Xception) to test for better accuracy (results are shown in the later section).

1) Our first model uses pre-trained InceptionV3, adding to it an average pooling layer,dropout layer and a dense layer. InceptionV3 is made of symmetrical and asymmetrical building blocks, including convolutions, average pooling, dropouts, and fully connected layers. This neural network is 48 layers deep. We have used it with include_top = False (i.e. we are not including the fully-connected layer at the top, as the last layer) , weights ="imagenet" (i. e. pre-training on ImageNet), and image_shape =

256 * 256 * 3 . Optional pooling mode for feature extraction (taken only when include_top is False) was taken to be none. Classifier activation (i.e. activation function on top layer) was ignored because our include_top is False. Inception V3 takes an image size of (299,299,3) as input. So we have resized our input image to (299,299,3). Then a max pooling layer is used to form an input layer for the Inception V3 Model. Hence the input image size becomes (150,150,3).

After passing through Inception V3 it passes through average pooling layer which involves calculating of average for each patch of a feature map and then dropout layer is used which mainly drops out units in a neural network,the a dense layer is used which feeds all outputs from the previous layer to all its neurons, each neuron providing one output to the next layer.

Layer (Type)	Output Shape	Param #
input_4 (InputLayer)	[(None, 150, 150, 3)]	0
sequential (Sequential)	(None, 150, 150, 3)	0
normalization_1 (Normalization)	(None, 150, 150, 3)	7
inception_v3 (Functional)	(None, 6, 6, 2048)	21802784
global_average_pooling2d_1	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 1)	2049

Total params: 21,804,840
Trainable params: 21,770,401
Non-trainable params: 34,439

Table 2: Model Summary for Inception

2) Our second pre-trained model was Xception.

It is a cnn that is 71 layers deep. It involves Depth Wise Separable Convolutions. Depthwise Convolution is a first step in which instead of applying convolution of size d×d×C, we apply a convolution of size d×d×1. The rest network works similar to inception. Though it takes more time to train it gives more efficient results than inception.

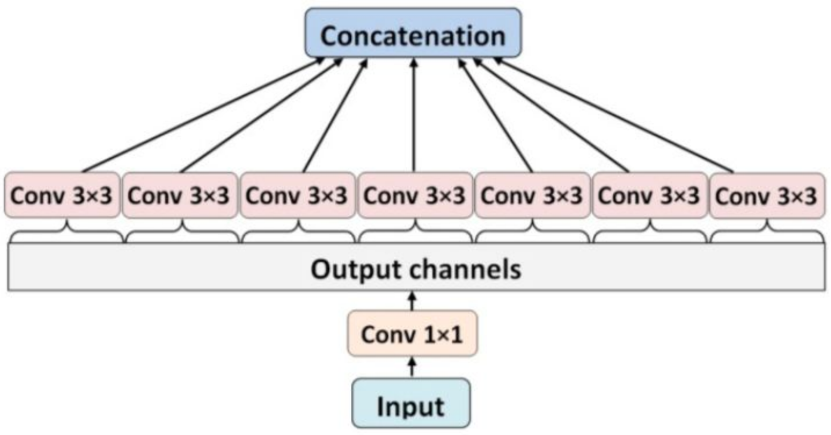
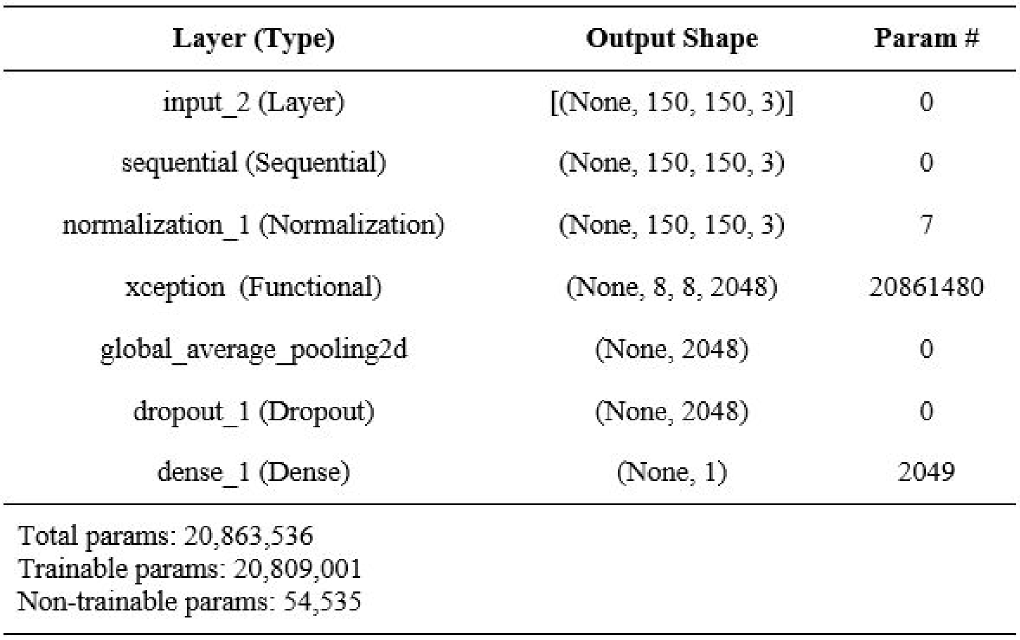


Figure 6: Xception Model Architecture



Layer (Type)	Output Shape	Param #
input_2 (Layer)	[(None, 150, 150, 3)]	0
sequential (Sequential)	(None, 150, 150, 3)	0
normalization_1 (Normalization)	(None, 150, 150, 3)	7
xception (Functional)	(None, 8, 8, 2048)	20861480
global_average_pooling2d	(None, 2048)	0
dropout_1 (Dropout)	(None, 2048)	0
dense_1 (Dense)	(None, 1)	2049

Total params: 20,863,536
Trainable params: 20,809,001
Non-trainable params: 54,535

Table 3 : Xception Model Summary

Segmentation:

Model Architecture:

In order to extract detailed features from the input image, multilevel semantic information is obtained through different layers of convolutional layers. U-Net is the basic network backbone of pavement surface crack image segmentation. U-net is composed of encoding trends (fully convolutional and pooling) and decoding trends (deconvolutional layers). The encoding layers extract multi level feature maps from the input image via fully convolutional and pooling operations. Pooling operations reduce the size of feature maps, hence the decoding layers try to recover it to the original size through transpose convolution. The size of the output image is maintained through concatenation. In each level of deconvolution, the feature maps with the same size in the convolution and pooling layers are copied to the output of the corresponding deconvolutional layers. Therefore, the feature information is best utilized and the image size is also conserved.

Dataset consists of Images of size (448,448) which was resized to (224,224) for feeding into the Unet model.During training, Images were passed through the a series of Fully Convolutional layer, which consisted of convolution, batch normalization and ReLU activation function to generate feature maps. The side output features are obtained by using convolutional blocks on the map generated by fully convolutional layers. It is then concatenated with the deconv(Conv 2D Transpose) layers which helps in predicting good output maps. The convolutional block is then passed through a max pooling layer with 2x2 filter size which is used to reduce plane size and also the number of parameters.

The four side-output layers are followed by deconvolutional layers, which are applied to upsample the plane size of the feature maps to be the same as the input image. Then, the upsampled feature maps are concatenated to form final features, which are followed by a convolutional layer and a sigmoid layer. As the output is a channel map of probabilities, therefore a threshold is used for the prediction of the label.

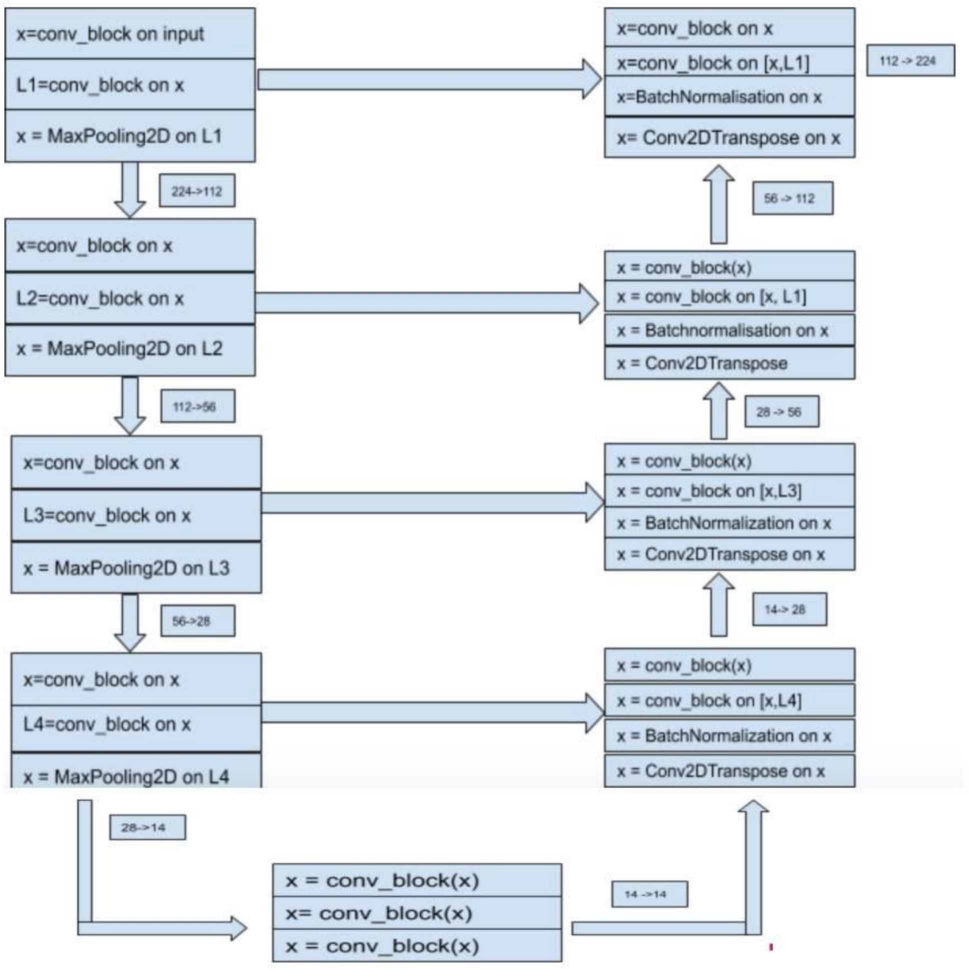


Figure 7. Architecture of Segmentation Model

Loss Function:

Crack pixels act as a minor part of the pavement image compared to the background. Thus the definition of the total loss might be influenced by the imbalance between the two classes. Therefore, we selected the pixel-wise cross-entropy loss function which is proved to be suitable for various object detection tasks.The pixelwise cross-entropy loss function of CrackU-net can be defined as

$$l(O_i;W) = \begin{cases} \log(1-P(O_i;W)), \text{if } y_i = 0 \\ \log(P(O_i;W)), \text{ otherwise} \end{cases},$$

where Oi denotes the convolutional outputs of the image pixel i. W denotes the standardized parameters of the network. P(O) denotes the sigmoid function that can project the output feature maps into crack probabilities. Based on this pixel wise definition, the objective of the model training can be expressed as

$$\operatorname{argmin}\left\{L(W)=\sum_{i=1}^{I}\sum_{j=1}^{J}l\left(o_i^{(j)};W\right)\right\},$$

where L(W) denotes the total loss. I denote the number of pixels. J denotes the number of convolution operations for each pixel.

5. Classification Model training:

Model 1:

The sigmoid units at the output layer predict the values between 0 and 1 for all individual pixels. Binary Cross entropy loss function is employed as the cost function to measure the similarity between the measured values and target values (Goodfellow et al., 2016; Nielsen, 2017) . Cross entropy is used because it improves learning speed and the network learns faster when the errors are large.Input image is resized to (256,256) and the model is defined accordingly.Images are trained using batches of size 32. Learning rate was kept 0.1 with Adam Optimizer. The model was trained on a Nvidia K80 GPU provided by Kaggle which decreases the training time significantly. The convolutional layers have the same padding in order to preserve size of the image and ReLU activation function to add non linearity. The model is trained for 30 epochs and each epoch takes a time of approximately 90 seconds. Initially the accuracy was less, but as we move forward in training the accuracy gradually increases and the losses decrease. Finally, we achieved an accuracy of 98.91% on the validation set

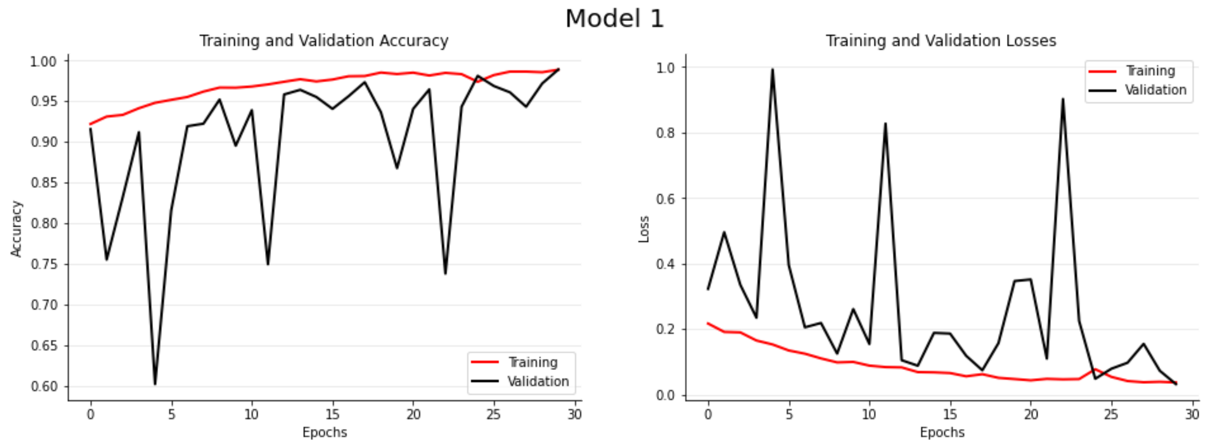


Figure 8: a) Graph of Training and Validation Accuracy of Model 1 with number of epochs b) Graph of Training and Validation Loss of Model 1 with number of epochs

Inception V3:

Similar to Model 1, the Inception V3 model was also trained on a NVidia K80 GPU available on Kaggle. GPU helps to achieve the purpose of parallel computing, which can significantly improve the efficiency of model training.The Inception V3 model took a running time of approximately 60 seconds per epoch. The model is trained for 7 epochs and resulted in validation accuracy of 99.90 %. Graphs represent the accuracy and losses of training and validation dataset with the number of epochs.

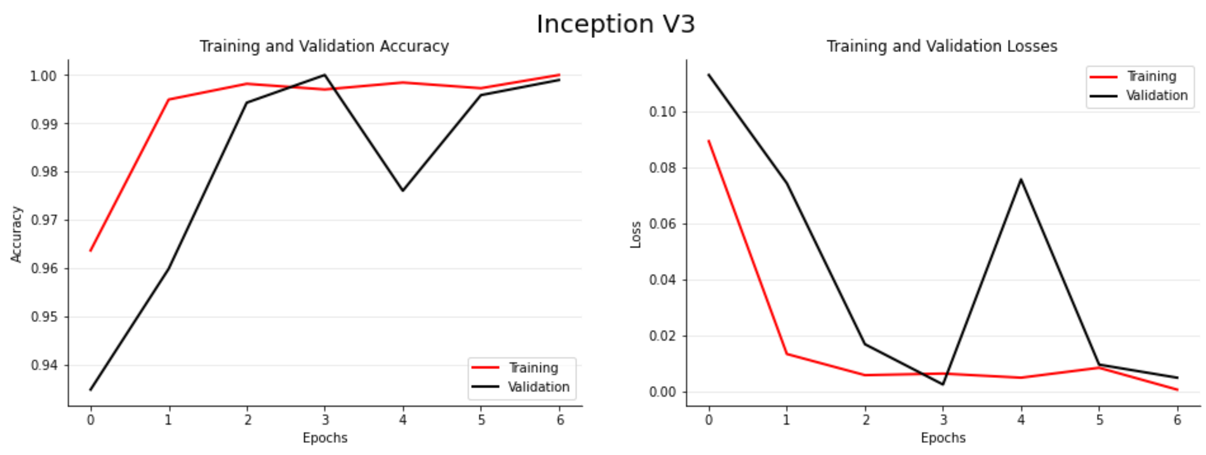


Figure 9: a) Graph of Training and Validation Accuracy of InceptionV3 with number of epoch
b) Graph of Training and Validation Loss of InceptionV3 with number of epochs

6. Segmentation Model Training

Around 8404 images and their annotations were used for training the UNet model. First the images were converted into tensor arrays, normalized and then fed for training. Training was performed for 20 epochs on GPU Nvidia K80/T4, provided by Google Colab. The first epoch took around 1.2 hours while the other 19 epochs were fast(around 3 minutes per epoch). Therefore the total training time was around 2 hours. Weights after each epoch were stored in a directory by calling ModelCheckpoint and the final weight was used for prediction on the test data. Learning rate was reduced by a factor of 0.2 using ReduceLROnPleateau whenever the validation loss was almost similar for 5 epochs. The model performed with an accuracy of 98.24% on the training data. Test images were used as a validation set and the model showed an accuracy of 98.14%.

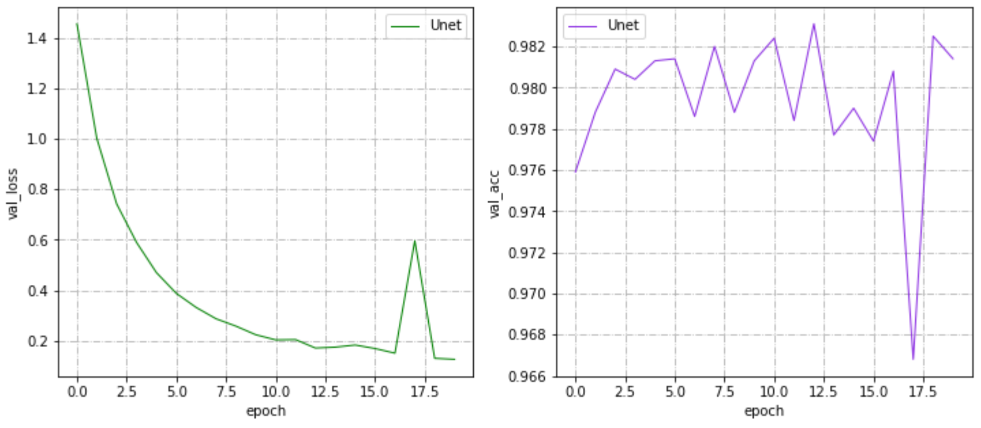


Figure 10: a) Graph of Validation Loss of Unet Model with number of epoch
b) Graph of Validation Accuracy of Unet Model with number of epochs

7. Testing and Evaluation for Classification

For evaluation (i.e testing the performance of our Model) we calculate accuracy for different models.
Accuracy = (TP + TN) / (TP +TN + FN+FP)
where;

TP = True Positive
TN = True Negative

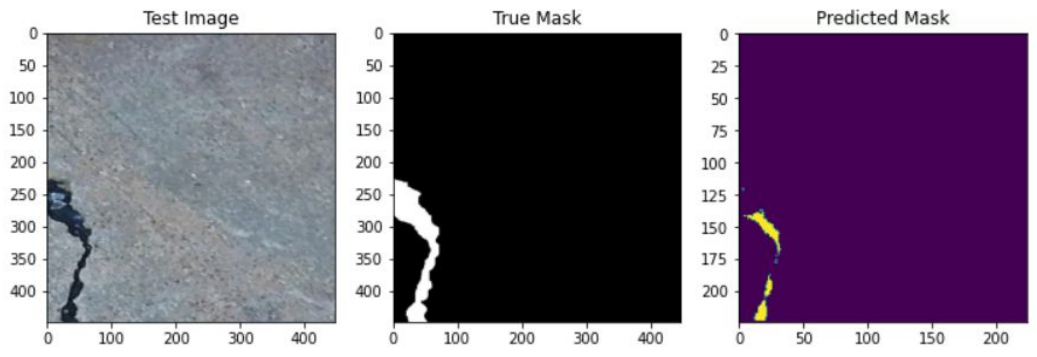
FP = False Positive
FN = False Negative

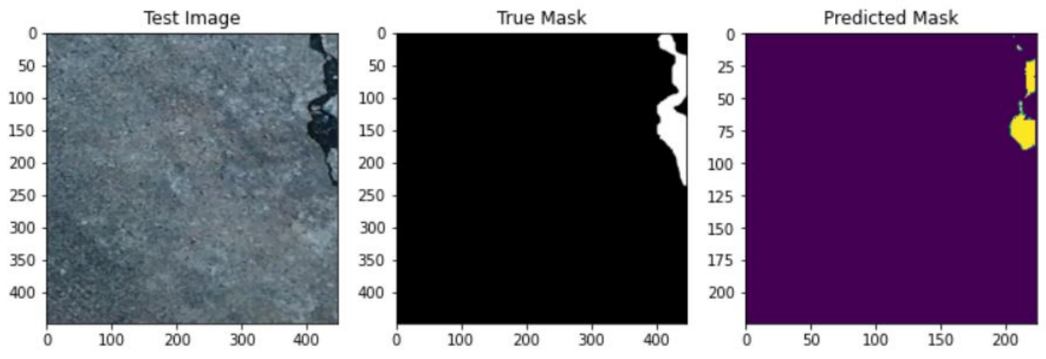
Table 4: Representing Accuracies for different Models

Model	**Cracked**				**Uncracked**				**Total**				**Accuracy**
	TP	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	**TP**	**TN**	**FP**	**FN**	
Model1	1465	0	18	0	0	196	0	16	1465	196	18	16	**0.979**
Xception	1481	0	2	0	0	204	0	8	1481	204	2	8	**0.994**
Inception V3	1481	0	2	0	0	211	0	1	1481	211	2	1	**0.998**

8. Testing and Results for Segmentation

The testing was done on nearly 1400 images, using the final weights returned by the model
The input image was resized, normalized and then fed into the model. Images were divided into batches of size 32, and an epoch of prediction took around 15 minutes.
Images examples:





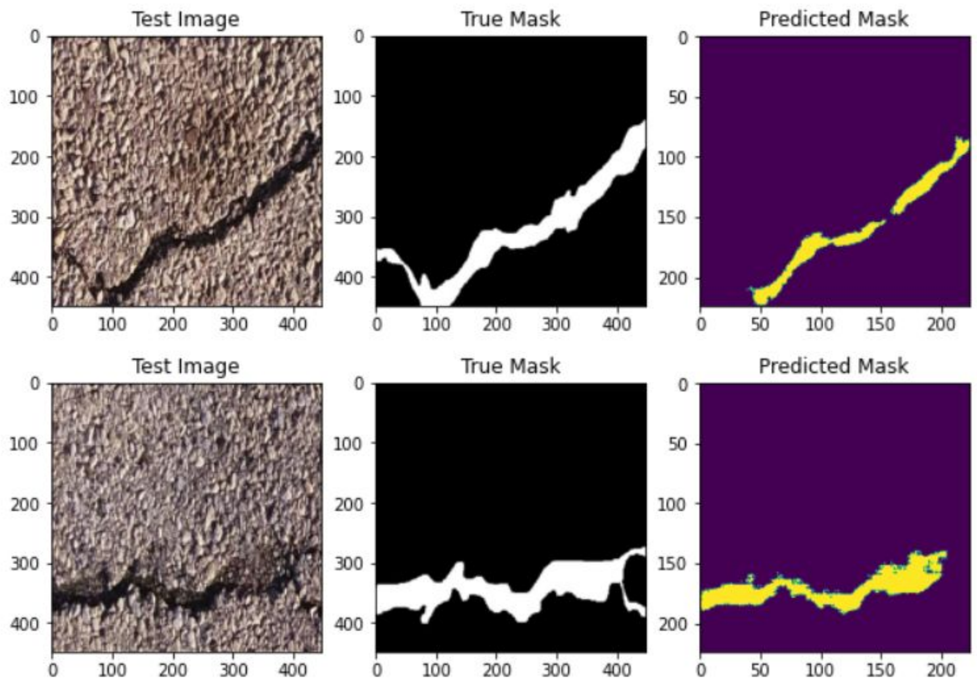


Figure 11 : Image Examples

But, looking more into predicted images, we came across such images

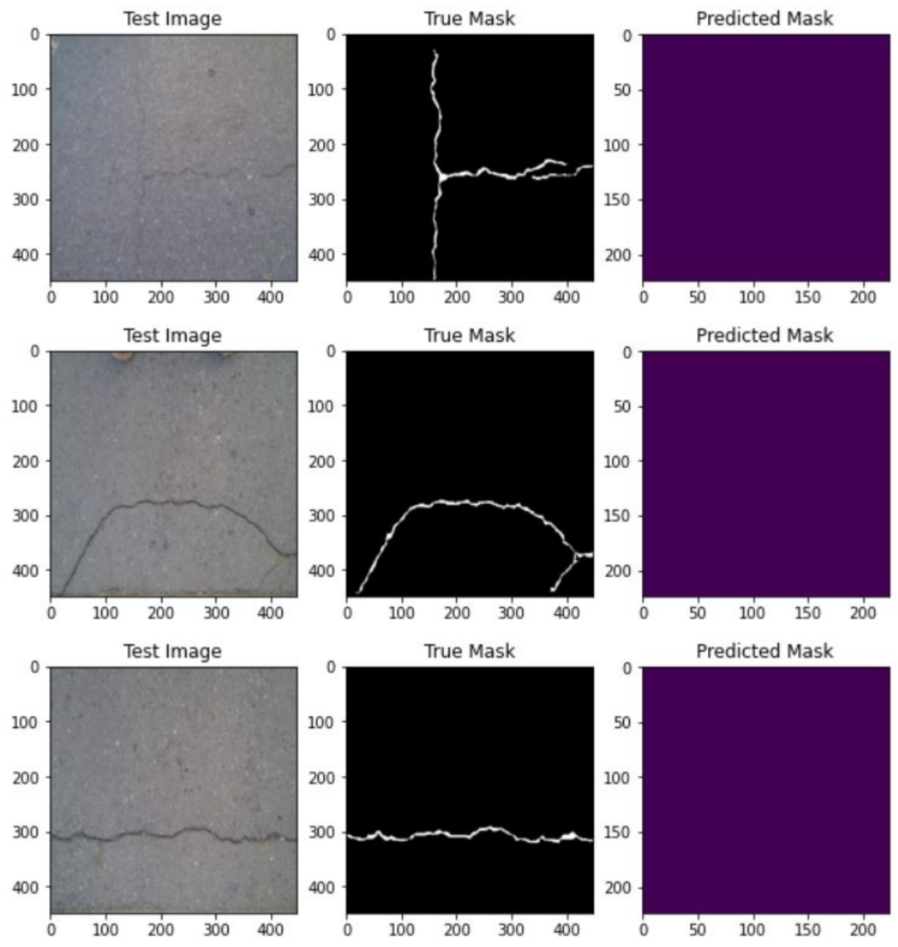


Figure 12 : Unusual Examples

Then we decided to find out the mean intersection over union and the f1 score for the training data and surprisingly it was very low. We tried to test some images of training data, to get some better insights of the problem. After this, we found out that for some specific dataset images(Crack 500) , the predicted mask had a good miou (near 50%) for some images and f1 score around 0.63 while for some other type of dataset(Crack Forest) iou was almost 0. Hence, we realised the limitation of our model, which was that it showed good results for a specific type of dataset and poor for others.
So the final predictions are made for the 504 images selected from crack 500 images.

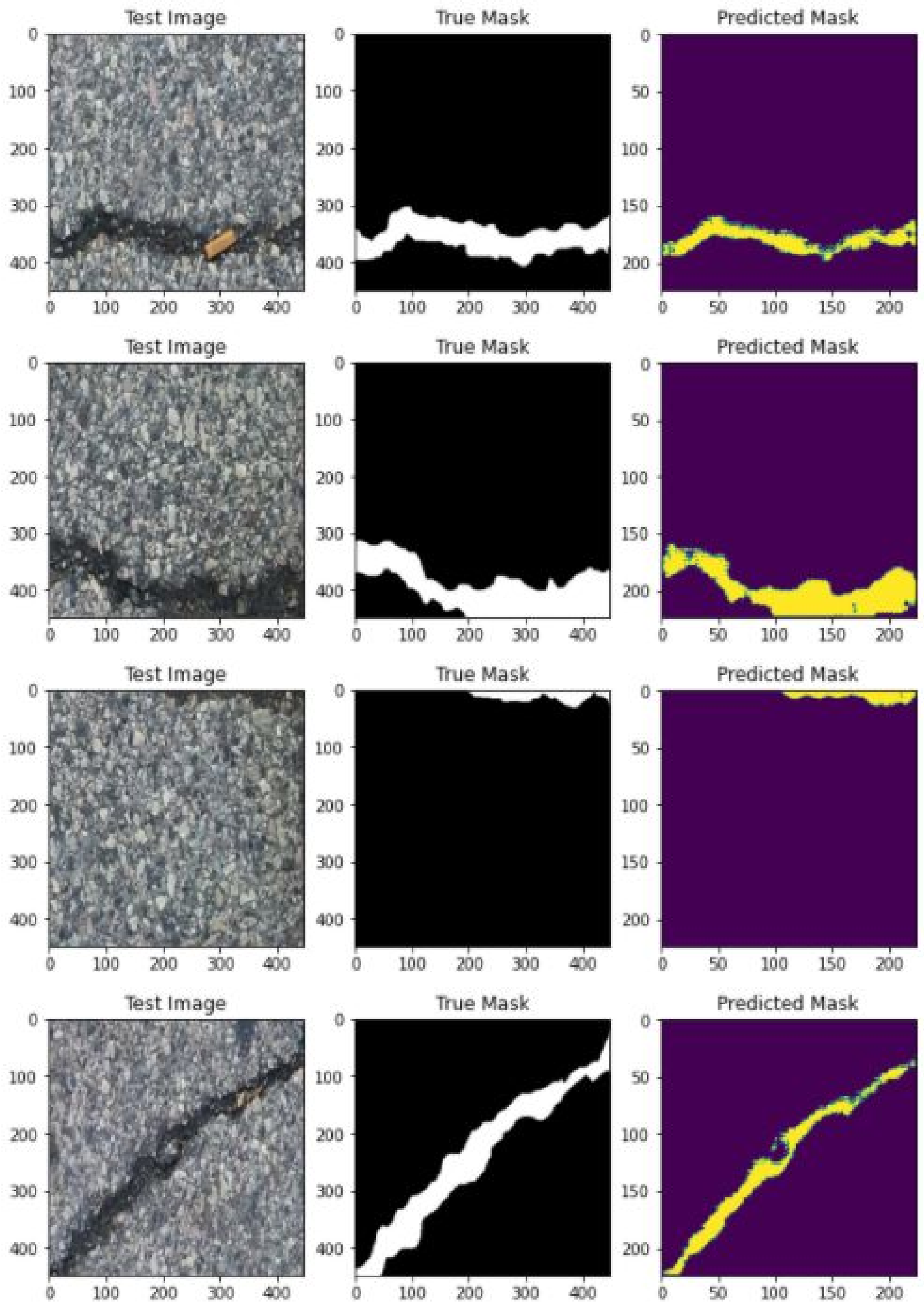


Figure 13 : Final Predictions

The model predicted with an IOU of greater than 0.5 and f1_score of greater than 0.6 for some images while for some of them, the iou was again almost zero thus reducing the mean IoU to 0.17 and F1_score of 0.255.
As the time was limited and most of our time was used in coding the model, we could not improve further.

9. Discussion:

1. From the results, it can be observed that the Inception V3 model has slightly better accuracy than Model 1 but has significantly decreased the training time for each epoch from 90 seconds to 60 seconds. Also, the Inception V3 model has achieved greater accuracy than the previous model in just 7 epochs. We also trained the Xception Mode on the same dataset, but the results were similar to the Inception V3 model with a validation accuracy of 99.4%. The Xception model also took relatively a longer time (130 seconds) for each epoch, so we concluded that the Inception V3 model works the best for our dataset.
2. The main problem in segmentation was to input training images along with their corresponding segmentation masks. The task of mapping the images with their masks and labels has been accomplished till now, and we are working on how to preprocess those images combined with their mask, to convert them into a format suitable for feeding into the image segmentation model.
3. Our model performed well for some of the images while it failed badly to predict on other images. After some research, we found that the model we chose performs well for a particular dataset while it fails to perform on other datasets.
4. The training time for each epoch was quite high, hence the training was done for 20 epochs only, although resulting in a good accuracy score. However, the results could have been improved further if we increased the number of training epochs.

10. Work Distribution

The main work which we have done is as follows-

1. Dataset preparation
2. Classification model 1
3. Classification using transfer learning (Inception and Xception as base models)
4. Segmentation model - U-Net

Although everyone contributed to every task to some extent, mainly the work of dataset preparation and segmentation planning was done by Akarsh and Lakshay. Classification model 1 was done by Neha and Aastha. Classification using transfer learning (Inception model) was done by Lakshay and Naman. Classification (Xception model) was done by Naman and Neha. Research for the segmentation model was done by everyone together and the model was run on a google colab notebook which was shared with everyone.

Everyone contributed equally to the final project report submission.

11. Conclusion

We used a dataset of over 11000 images to perform crack classification and segmentation. For crack classification, we achieved an accuracy of 99.5% using Transfer Learning technique. We used only cracked images to perform image segmentation and an encoder-decoder model was used to perform this task. The segmentation model did not perform as per expectations and the FI score was low. To improve the results, the research can be extended to training more advanced pavement crack segmentation models like Deep Crack, Crack U-Net.

12. References

[1] H. Oliveira and P. L. Correia, “Automatic road crack detection and characterization,” IEEE Transactions on Intelligent Transportation Systems,vol. 14, no. 1, pp. 155–168, 2013.
[2] Y. Shi, L. Cui, Z. Qi, F. Meng, and Z. Chen, “Automatic road crack detection using random structured forests,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 12, pp. 3434–3445, 2016.
[3] Y.-J. Cha, W. Choi, and O. Buy ¨ uk ¨ ozt ¨ urk, “Deep learning-based crack damage detection using convolutional neural networks,” Computer-Aided Civil and Infrastructure Engineering, vol. 32, no. 5, pp. 361–378, 2017.
[4] A. Zhang, K. C. Wang, B. Li, E. Yang, X. Dai, Y. Peng, Y. Fei, Y. Liu, J. Q. Li, and C. Chen, “Automated pixel-level pavement crack detection on 3D asphalt surfaces using a deep-learning network,” Computer-Aided Civil and Infrastructure Engineering, vol. 32, no. 10, pp. 805–819, 2017.
[5] L. Zhang, F. Yang, Y. D. Zhang, and Y. J. Zhu, “Road crack detection using deep convolutional neural networks,” in Proc. Int. Conf. Image Process. IEEE, 2016, pp. 3708–3712.
[6]Krizhevsky, A., Sutskever, I. & Hinton, G. (2012), ImageNet
classification with deep convolutional neural networks, Advances in Neural Information Processing Systems, v2,
[7]Automated Pixel‐Level Pavement Crack Detection on 3D Asphalt Surfaces Using a Deep‐Learning Network Allen Zhang Kelvin C. P. Wang Baoxian Li Enhui Yang Xianxing Dai Yi Peng Yue Fei Yang Liu Joshua Q. Li Cheng Chen
[8]Cha, Y. J., Choi, W. & Buyukozturk, O. (2017), Deep learning-based crack damage detection using convolutional neural networks, Computer-Aided Civil and Infrastructure
Engineering, 32(5), 361–78
[9]Zhang, L., Yang, F., Zhang, Y. D. & Zhu, Y. J. (2016b), Road crack detection using deep convolutional neural network, in Proceedings of International Conference on Image Processing, Phoenix, AZ, v2016, 3708–12.

Data set Reference:

CRACK500:

[10]Zhang, Lei and Yang, Fan and Zhang, Yimin Daniel and Zhu, Ying Julie, Road crack detection using deep convolutional neural network ,Image Processing (ICIP), 2016

[11]Yang, Fan and Zhang, Lei and Yu, Sijia and Prokhorov, Danil and Mei, Xue and Ling, Haibin, Feature Pyramid and Hierarchical Boosting Network for Pavement Crack Detection

GAPs384:

[12]Eisenbach, Markus and Stricker, Ronny and Seichter, Daniel and Amende, Karl and Debes, Klaus and Sesselmann, Maximilian and Ebersbach, Dirk and Stoeckert, Ulrike and Gross, Horst-Michael, How to Get Pavement Distress Detection Ready for Deep Learning? A Systematic Approach.,International Joint Conference on Neural Networks (IJCNN)

CFD:

[13]Shi, Yong and Cui, Limeng and Qi, Zhiquan and Meng, Fan and Chen, Zhensong, Automatic road crack detection using random structured forests,IEEE Transactions on Intelligent Transportation Systems