


<b>Semester (Term, Year)</b>	
<b>Course Code</b>	
<b>Course Section</b>	
<b>Course Title</b>	
<b>Course Instructor</b>	
<b>Submission</b>	
<b>Submission No.</b>	
<b>Submission Due Date</b>	
<b>Title</b>	
<b>Submission Date</b>	

<b>Submission by (Name):</b>	<b>Student ID (XXXX1234)</b>	<b>Signature</b>
		

*By signing the above you attest that you have contributed to this submission and confirm that all work you contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a "0" on the work, and "F" in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Academic Integrity Policy 60, which can be found at [www.torontomu.ca/senate/policies/](http://www.torontomu.ca/senate/policies/)*

# Table of Contents

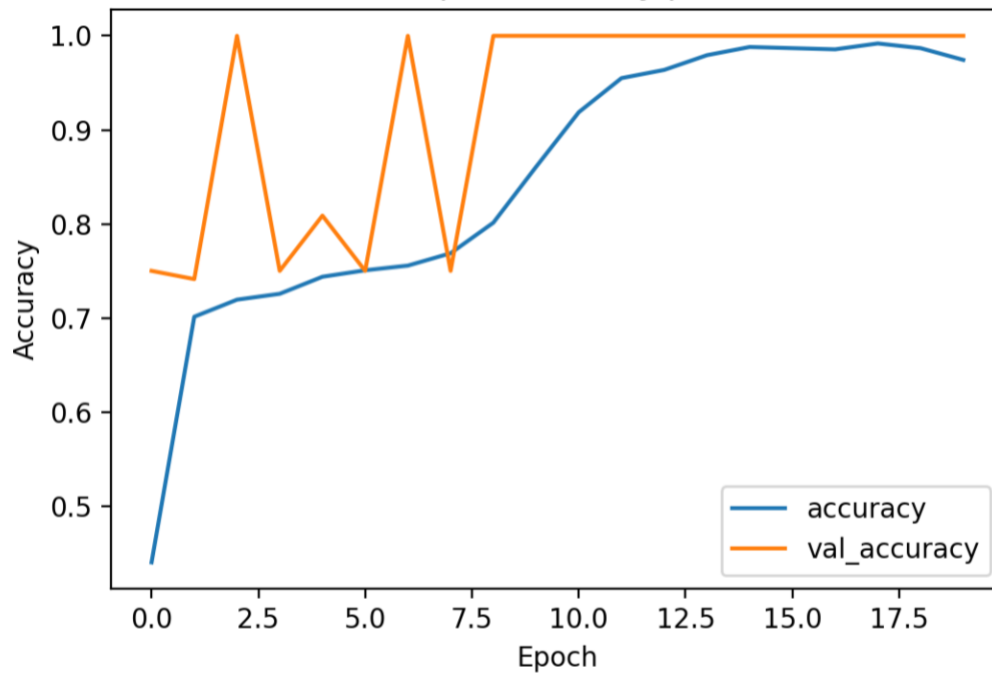
Cover page .....	1
Table of Contents .....	2
List of Tables and Figures .....	2
Results & Discussion .....	3
Further Discussion .....	7
Conclusion .....	10
Appendix .....	11
GitHub Link .....	11
Python Code Step 1 to 4 .....	11
Python Code Step 5 .....	13

## List of Tables and Figures

<b><u>Figure 1:</u></b> The Accuracy plot of the CNN model as Epoch approaches 20 .....	3
<b><u>Figure 2:</u></b> The Loss plot of the CNN model as the Epoch approaches 20 .....	4
<b><u>Figure 3:</u></b> The max probability from the model's prediction - Test/Medium directory .....	5
<b><u>Figure 4:</u></b> The max probability from the model's prediction - Test / Large directory .....	6
<b><u>Figure 5:</u></b> The accuracy plot of the same model but Epoch = 5 .....	7
<b><u>Figure 6:</u></b> The loss plot of the same model but Epoch = 5 .....	7
<b><u>Figure 7:</u></b> The max probability from the Epoch = 5 model - Test / Medium directory .....	8
<b><u>Figure 8:</u></b> The max probability from the Epoch = 5 model - Test / Large directory .....	9

## Results & Discussion

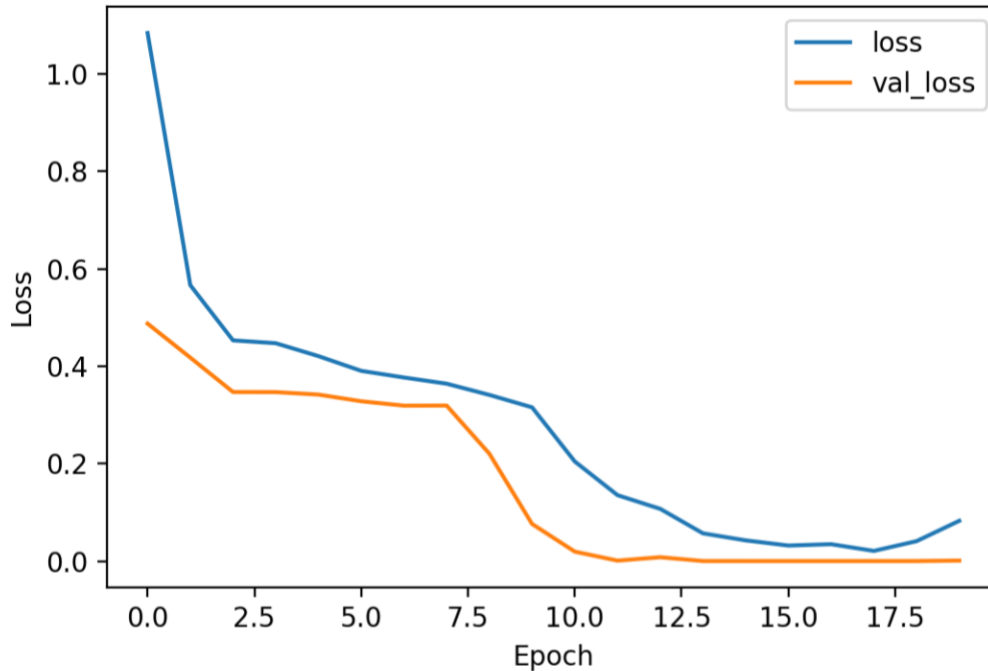
The accuracy data represents the differences in the number of correction predictions while fitting using the validation data set. The Epoch number was set to 20, meaning the neural network was iterated through the whole train set 20 times, higher Epoch number tend to yield more accurate results. The figure below showcases the accuracy of the model based on number of Epoch:



**Figure 1:** The Accuracy plot of the CNN model as Epoch approaches 20

The orange validation accuracy line in Figure 1 has an upward trend as Epoch number increases, roughly corresponding to the blue accuracy line. At low Epoch number, the accuracy is poor, this is because the neural network wasn't trained enough. The accuracy of the model had a value of about 0.75 initially, and at low Epoch number, they were not stable and fluctuated around. However, as the model was being fitted repeatedly, their accuracy increased and eventually stabilized at 100% accuracy around Epoch 8.

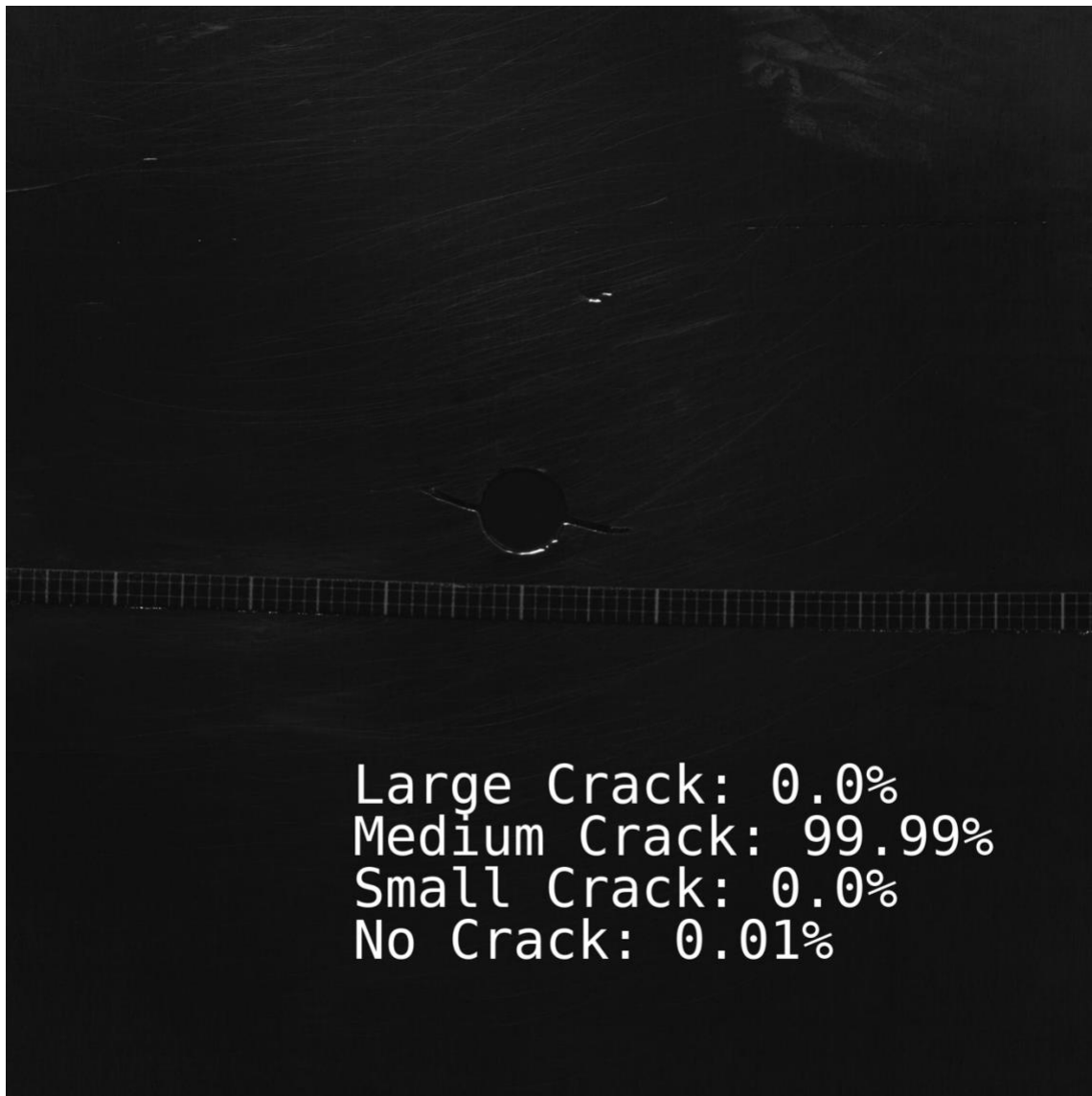
The loss function represents the comparison of the prediction outcome of the neural network and the actual validation results. This means that the loss results of the model must be small and decreases as Epoch number increases. In other words, the prediction must be as close as possible to the target results. The figure below showcases the loss function of the neural network function as the Epoch number approaches 20:



**Figure 2:** The Loss plot of the CNN model as the Epoch approaches 20

According to the Figure 2, the loss function of the model trend to decreases as Epoch number increases. Initially, the loss value started high at a value of about 0.5 meaning only half of the model's predictions were correct. As the model were being fitted repeatedly, the predictions became closer to actual value and eventually stabilized at 0% loss at around Epoch 13. This indicates the importance of adequate number of Epoch or iterations to ensure the model is being trained enough to be as accurate as possible and lose as little as possible.

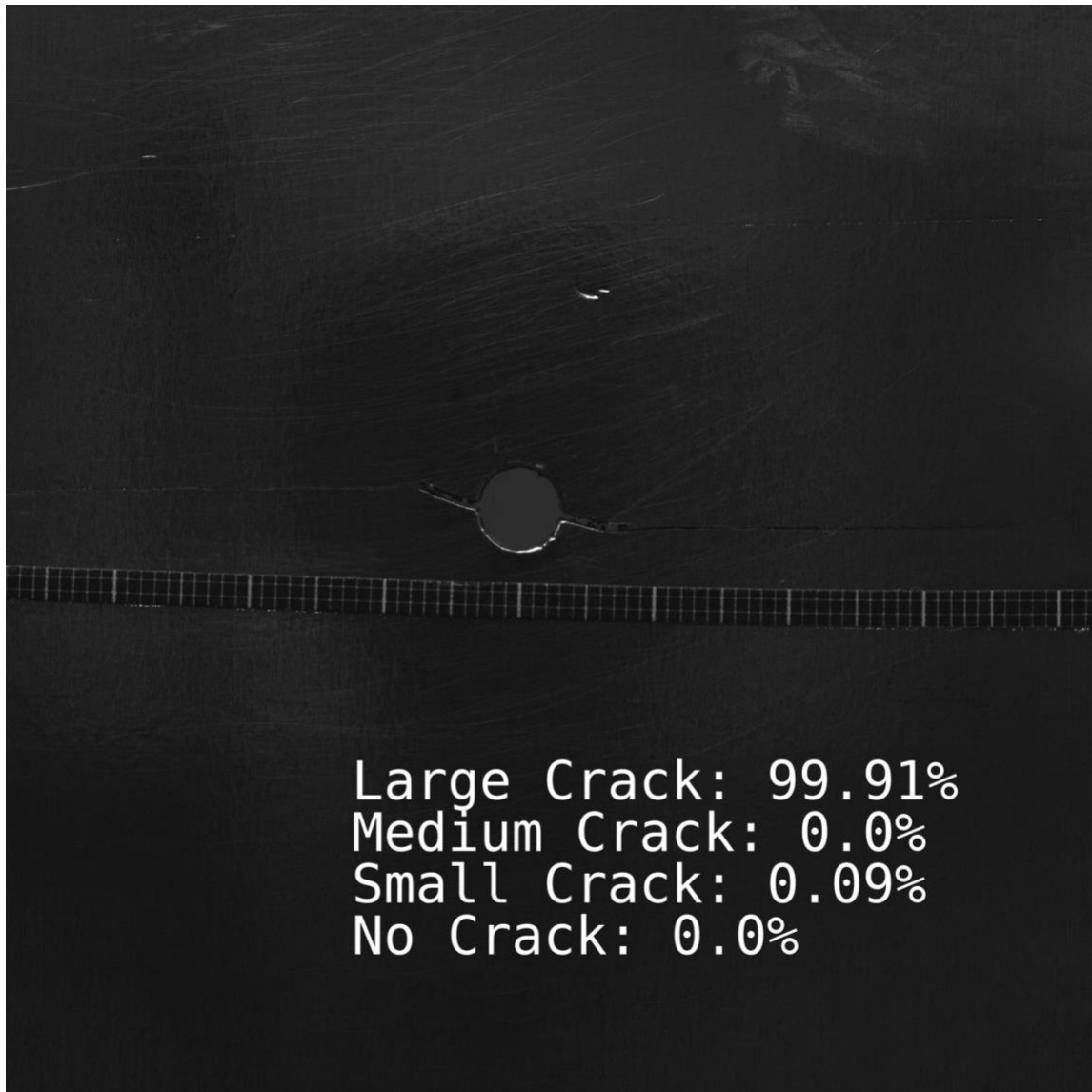
Once the model is trained and fitted, it can be used to predict actual images completely outside of the train and validation dataset. The model would classify the size of the crack based on a probability ranking system. This means when the neural network model is applied to an image, it calculates how likely it is for the image to belong to each class. Subsequently, the model would classify the image based on the class with the highest probability. The model was tested with 2 provided image files. The figure below shows the prediction of the model for the 'Crack\_\_20180419\_06\_19\_09,915.bmp' file, which is under the Test / Medium directory:



**Figure 3:** The max probability from the model's prediction - Test/Medium directory

According to Figure 3, the model yielded the highest probability for the Medium Crack class at a probability of 99.99%, while other classes yield zero or near zero chance of the test image belonging to that class. This proves that the model accurately classifies the test image because the 'Crack\_\_20180419\_06\_19\_09,915.bmp' file is under the Test / Medium directory.

Similarly, the model was tested a second time by applying it to a different file, this file is 'Crack\_\_20180419\_13\_29\_14,846.bmp' which lives under the Test / Large directory.

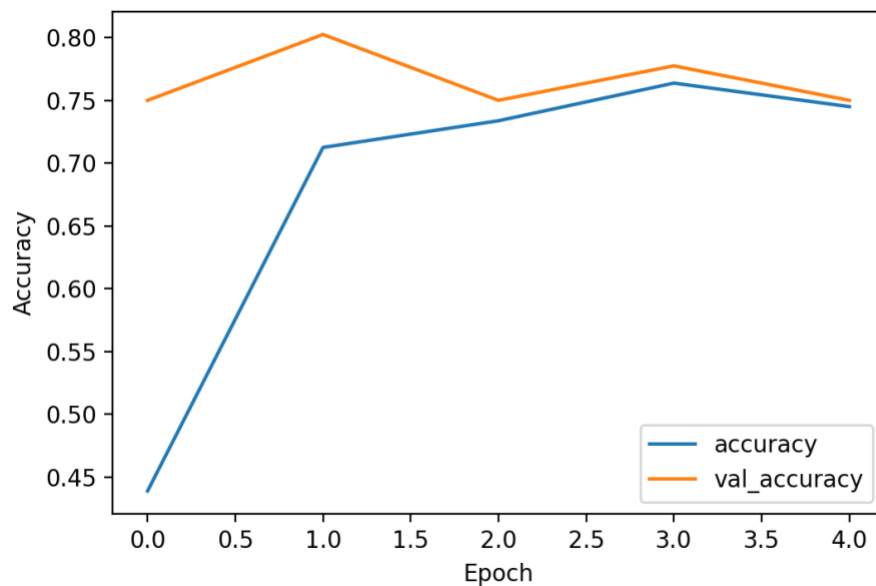


**Figure 4:** The max probability from the model's prediction - Test / Large directory

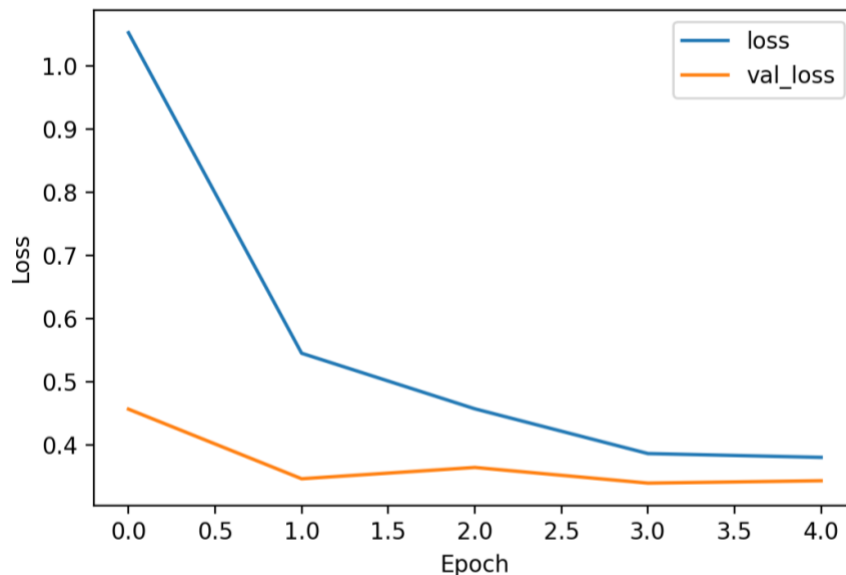
According to Figure 4, the model yielded the highest probability for the Large Crack class at a probability of 99.91%, while other classes yield zero or near zero chance of the test image belonging to that class. This proves that the model accurately classify the test image because the 'Crack\_\_20180419\_13\_29\_14,846.bmp' file is under the Test / Large directory.

## Further Discussion

The neural network in this project used a high Epoch number of 20. This ensures the accuracy function is able to stabilize at 100% and the loss function to stabilize at 0%. However, what would be the outcome if the model used a low Epoch number? In this project, a new neural network was fitted. Every aspect of this new model is the same as the model used for the image classification task above. However, the one crucial difference is the Epoch number being very low, it was set to 5 instead of Epoch 20. The figures below represent the accuracy and loss function of this new model with low Epoch number:



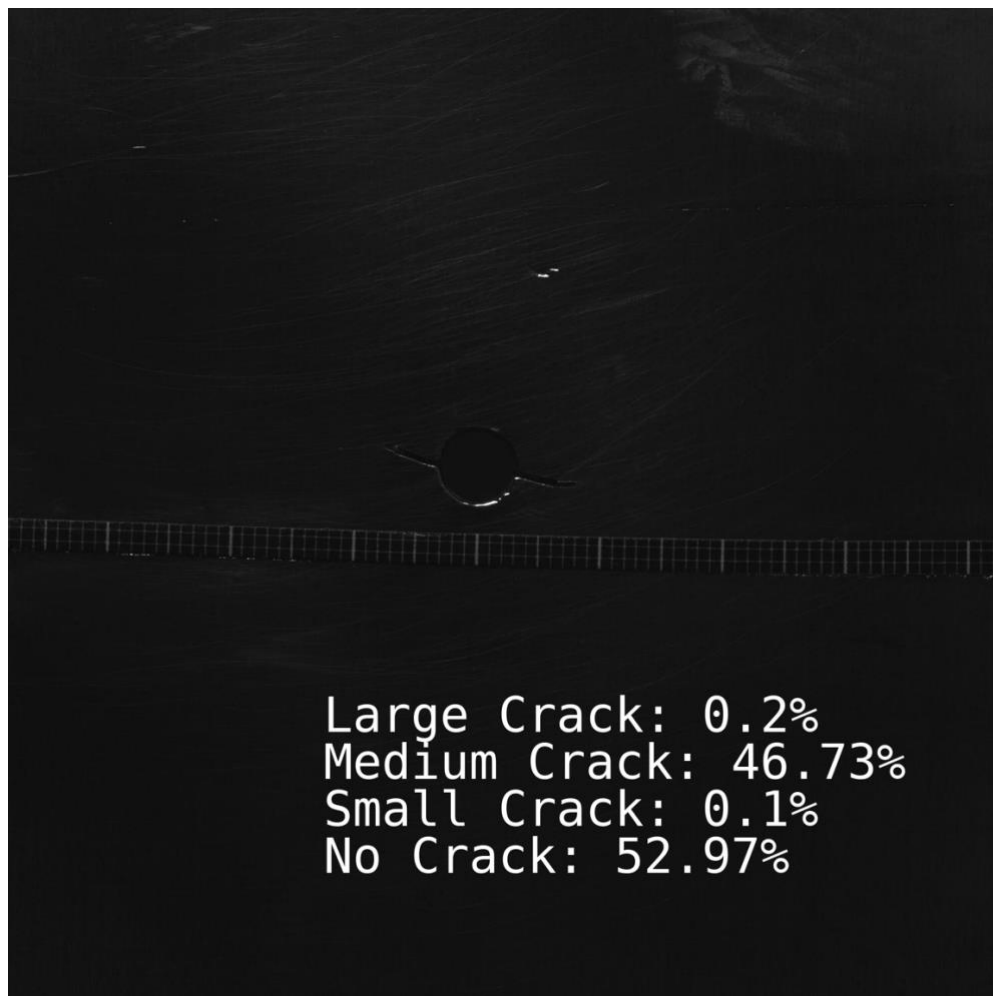
**Figure 5:** The accuracy plot of the same model but Epoch = 5



**Figure 6:** The loss plot of the same model but Epoch = 5

According to Figure 5 and 6, the general trends of the accuracy and loss function are very poor, they do not improve as Epoch number increases. The accuracy of this model started at 0.75 and fluctuated at that level throughout the training process. Similarly, the loss of this model started just below 0.5 and barely decreases to around 0.3 by the end of the training process. This is objectively an inferior model compared to the Epoch 20 model given that the Epoch 20 model yielded a 100% accuracy and 0% loss.

This new model with a low Epoch value of 5 were used to classify the same 2 test images used for the previous model. These 2 image files are 'Crack\_\_20180419\_06\_19\_09,915.bmp' which belongs to the Test / Medium directory and 'Crack\_\_20180419\_13\_29\_14,846.bmp' which is under the Test / Large directory. The figures below show the prediction results being used the new model with a low Epoch value of 5:

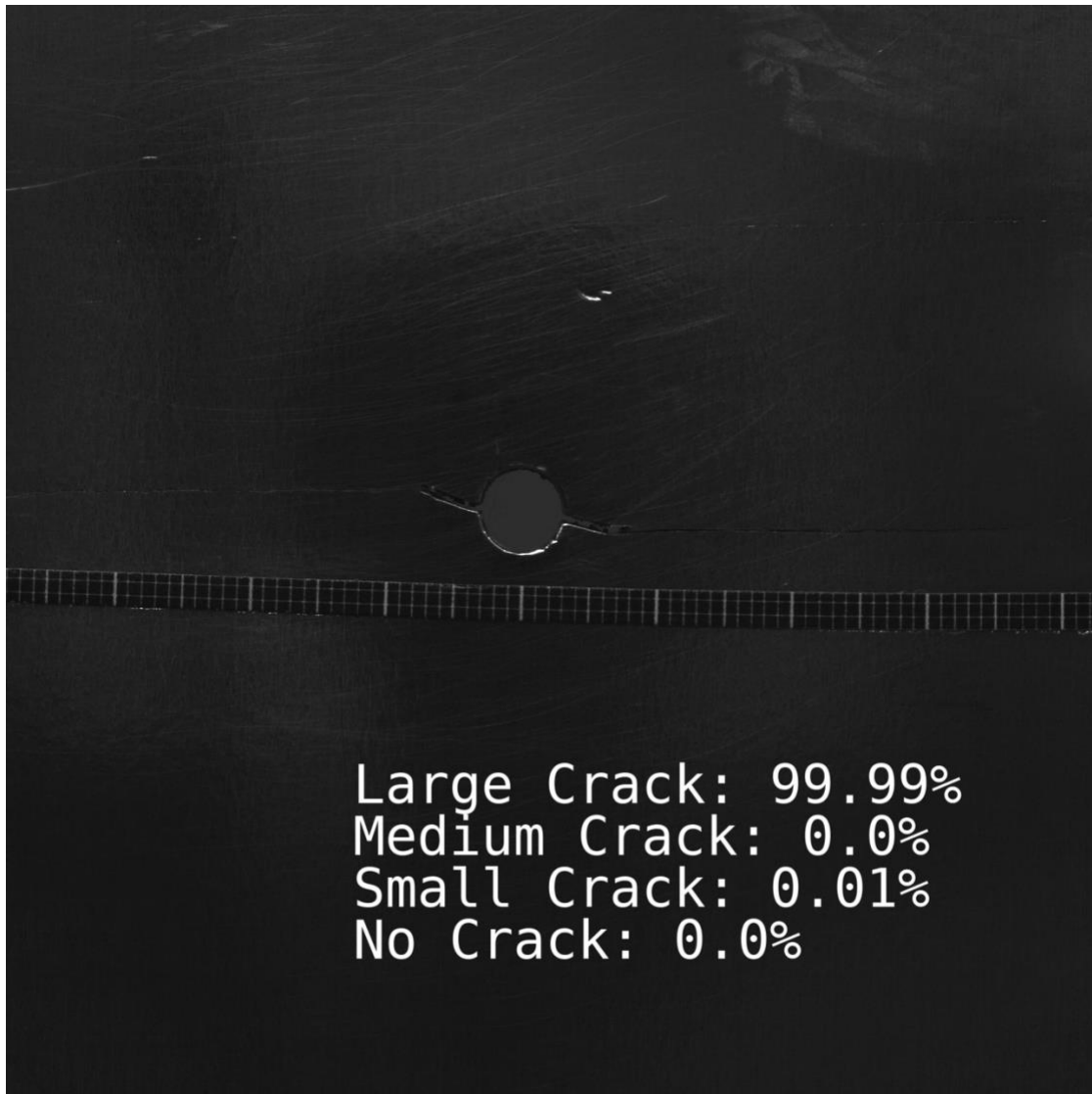


**Figure 7:** The max probability from the Epoch = 5 model - Test / Medium directory

According to Figure 7, this new model yielded a highest probability of 53% for No Crack class, second highest probability for Medium Crack at 47% and neglectable probability for the remaining classes. Since the No Crack class has the highest probability, the model classified this image as



No Crack. This result is incorrect since the image is under the Test / Medium directory. The Epoch model not only classified the image to the wrong class, it also did so with less certainty. This uncertainty can be observed by comparing 53% No Crack probability to the 99% Medium Crack probability from the high Epoch 20 model. The results for the second test image are shown in the figure below:



**Figure 8:** The max probability from the Epoch = 5 model - Test / Large directory

According to Figure 8, this new model yielded a highest probability of 99.99% for the Large Crack, which is inline in with the higher Epoch model. This can explain by this model's accuracy value of 0.75, therefore there is a 3/4 chance of predicting correctly, which is the case for this test image.

## Conclusion

Overall, this project aims to develop a convoluted neural network model to classify images with cracks in to 4 categories of Large, Medium, Small and No Crack. The model used Python TensorFlow platform to train the neural network. The architecture of this model consists of Conv2D, MaxPooling2D, Flatten, Dense and Dropout layers. Specifically, 3 Conv2D layers, 3 MaxPooling2D layers, 1 Flatten layer, 3 Dense layers and 2 Dropout layers. The model was then fitted using an Epoch value of 20. This model yielded an accuracy of 100% and loss of 0% meaning the model perform very well to a high degree. Two test images were applied to the neural network model to classify the size of the crack. In both cases, the model predicted 99.9% probability of the image belonging to their correct respective class. Further analysis was performed on the same model with a lower Epoch value of 5 which yielded vastly inferior performance. This low Epoch model has a 75% accuracy and 30% loss while also incorrectly classify one of the test images with less certainty. It is extremely important to set adequate Epoch number to ensure accurate neural network model fitting. Furthermore, the number of layers inside the architecture of the model and their respective hyperparameters must be tuned appropriately to prevent overfitting or underfitting of the data. Ultimately, the Epoch number and the architectures directly influences the performance of the neural network model.

# Appendix

## GitHub Link

<https://github.com/GetHydrogen/Project-2-Duong-AER850-F2023.git>

## Python Code Step 1 to 4

```
### Import packages
import os
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import datasets, layers, models
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing import image
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

### Step 1: Data Processing

shape = (100,100,3)

script_path = os.path.realpath(__file__)
root = os.path.dirname(script_path)
train_dir = os.path.join(root,'Data','Train')
validation_dir = os.path.join(root,'Data','Validation')

train_aug = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
validation_aug = ImageDataGenerator(rescale=1./255)

train_gen = train_aug.flow_from_directory(
    train_dir,
    target_size=(100,100),
    batch_size=32,
    class_mode='categorical'
)

validation_gen = validation_aug.flow_from_directory(
    validation_dir,
```

```

        target_size=(100,100),
        batch_size=32,
        class_mode='categorical'
    )
    #%% Step 2: Neural Network Architecture Design
    # Step 3: Hyperparameter Analysis

    model = models.Sequential()

    model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=shape))
    model.add(layers.Conv2D(64, (3, 3), activation='relu'))
    model.add(layers.Conv2D(128, (3, 3), activation='relu'))

    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.MaxPooling2D((2, 2)))
    model.add(layers.MaxPooling2D((2, 2)))

    model.add(layers.Flatten())

    model.add(layers.Dense(128, activation='relu'))
    model.add(Dropout(0.5))
    model.add(layers.Dense(128, activation='relu'))
    model.add(Dropout(0.5))

    model.add(layers.Dense(4, activation='softmax'))

    model.compile(optimizer='adam',
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])

    model.summary()

    #%% Step 4: Model Evaluation

    #Fit and save model
    history = model.fit(train_gen, epochs=5, validation_data=(validation_gen))
    model.save('saved_model_Duong_65857.h5')

    #Plot Accuracy
    plt.title('Step 4: Accuracy plot')
    plt.plot(history.history['accuracy'], label='accuracy')
    plt.plot(history.history['val_accuracy'], label='val_accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
    plt.savefig('Step 4 Accuracy plot',dpi=200)

```

```
plt.show()

#Plot Loss
plt.title('Step 4: Loss plot')
plt.plot(history.history['loss'], label='loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.savefig('Step 4 Loss plot',dpi=200)
plt.show()
```

## Python Code Step 5

```
### Import packages
import os
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
import numpy as np
from PIL import Image, ImageDraw, ImageFont

### Step 5: Model Testing

script_path = os.path.realpath(__file__)
root = os.path.dirname(script_path)
test1_path = os.path.join(root,'Data','Test','Medium',
                           'Crack__20180419_06_19_09,915.bmp')
test2_path = os.path.join(root,'Data','Test','Large',
                           'Crack__20180419_13_29_14,846.bmp')

test1 = image.img_to_array(image.load_img(test1_path, target_size=(100, 100)))
test2 = image.img_to_array(image.load_img(test2_path, target_size=(100, 100)))
test1 = np.expand_dims(test1/255.0, axis=0)
test2 = np.expand_dims(test2/255.0, axis=0)

model = tf.keras.models.load_model('saved_model_Duong_65857.h5')

pred1_prob = model.predict(test1)
pred1 = np.argmax(pred1_prob, axis=-1)

pred2_prob = model.predict(test2)
pred2 = np.argmax(pred2_prob, axis=-1)
```

```

text_position = (650, 1400)
text_color = 255
font_path = os.path.join(root, 'DejaVuSansMono.ttf')
font = ImageFont.truetype(font_path, 100)

img1 = Image.open(test1_path)
draw = ImageDraw.Draw(img1)
text1 = 'Large Crack: '+str(round(pred1_prob[0,0]*100,2))+ '%\n'\
        +'Medium Crack: '+str(round(pred1_prob[0,1]*100,2))+ '%\n'\
        +'Small Crack: '+str(round(pred1_prob[0,2]*100,2))+ '%\n'\
        +'No Crack: '+str(round(pred1_prob[0,3]*100,2))+ '%\n'
draw.text(text_position, text1, fill=text_color, font=font)
img1.show()
img1.save(os.path.join(root, 'Test Prediction 1:Crack__20180419_06_19_09,915.bmp'))

img2 = Image.open(test2_path)
draw = ImageDraw.Draw(img2)
text2 = 'Large Crack: '+str(round(pred2_prob[0,0]*100,2))+ '%\n'\
        +'Medium Crack: '+str(round(pred2_prob[0,1]*100,2))+ '%\n'\
        +'Small Crack: '+str(round(pred2_prob[0,2]*100,2))+ '%\n'\
        +'No Crack: '+str(round(pred2_prob[0,3]*100,2))+ '%\n'
draw.text(text_position, text2, fill=text_color, font=font)
img2.show()
img2.save(os.path.join(root, 'Test Prediction 2: Crack__20180419_13_29_14,846.bmp'))

```