42028: Deep Learning and Convolutional Neural Network

Assignent-3 Project Report

Project Title:

Identifying cracks and damages in concrete structures

Team Name and Project Number:

Concrete Team, 23

Team Members:

Andres Felipe Lagos	13092248
Carlos Mario Carvajal	13144148
Ernest Ilustre	12763239

Date:

June 18th, 2020

Table of Contents

Abstract

Deep learning and convolutional neural networks have demonstrated to be highly applicable to many industries and a variety of problems. Despite the general belief that sectors like construction only require workforce and machinery, this project has demonstrated that computer vision can provide significant benefits in the identification of damages in concrete structures. Hence, this study was developed to address the identification of cracks in structures to enable workers to focus on more critical tasks. Four different models were developed using classic Convolutional Neural Network architectures such as Inception ResNet v2 and VGG16. Initially, these models were implemented in the identification of the type of structure such as Pavement, Wall or Bridge Deck, and finally, the classification of cracked or uncracked structures.

1. Introduction and Background

Construction is one of the industries that contributes the most to the gross domestic product of many countries, as governments and private companies understand that good infrastructure is the baseline for economic activity. Similarly, it is equally essential to maintain construction structures in good health to prevent a tear or, in the worst-case scenario, a collapse of the structure and impact on lives or economic activity. For this reason, it is necessary to monitor continuously the state of a structure to identify damages and prevent subsequent problems if the structure collapses. The first sign of an unhealthy structure is the presence of cracks which can deteriorate over time if they are not attended. However, the constant monitoring of concrete structures can be overwhelming, time-consuming and extremely expensive when performed by humans. Therefore, it becomes necessary to find a way to automate this process, and computer vision is presented as the solution to this problem.

1.1 The problem

It is possible to observe that in large cities like Sydney and Melbourne, are many structures (Pavement, Walls, Bridge Decks) that present damages and require repair. Damaged structures are easily noticed when going around the CBD and walking through the streets and parks or looking at some buildings. Moreover, a high number of motorways around Australia will wear or tear over time and get cracks on the surface, due to high traffic or environmental conditions, such as heat and rain. These damages are aligned with a high cost of maintenance if they are not prevented on time, or they cause an accident that can take on lives or high costs. To avoid these problems, constant monitoring is required, and current methodologies in the identification and classification of cracked structures use a large variety of resources such as time and labour force that are very expensive and sometimes inaccurate. This because of the enormous amount of concrete structures built in cities that are highly overwhelming and also because the scarce availability of resources to monitor them can result inefficient. Furthermore, on many occasions, the size of the cracks can be so small that the human eye can easily miss it.

Therefore, it becomes necessary to automate the monitoring of these structures, to rapidly identify damages and act on them to prevent further damage or collapse. Thus, Computer Vision and Deep Learning become powerful tools to build an approach that addresses the need of replacing humans with machines for the repetitive and overwhelming task of identifying the type of structure and if it is cracked or uncracked.

1.2 Motivations

Deep learning algorithms have been widely used in the solution of different tasks, proving benefits such as the reduction of resources allocated to them. Furthermore, high accuracy in identification of something specific in an image space presents another benefit that in several occasions can outperform humans. Therefore, an image classification algorithm that implements state-of-the-art Convolutional Neuronal Networks for this problem presents itself as a potential solution that can enhance the construction industry. By integrating a classification algorithm in a system that identifies the structure type and whether it is cracked or uncracked, it is believed that millions of dollars can be saved every year in workforce and time spent in this activity. Hence, the main objective of this project is the elaboration of a robust model and system able to classify surface images (Bridge Deck, Wall and concrete) into uncracked or cracked surfaces to enable to automate the monitoring process.

The implementation of this system will use state-of-the-art techniques in Deep Learning algorithms such as ass Convolutional Neuronal Networks that will deliver outstanding results in the classification of cracked images, that are believed able to outperform humans.

1.3 Application

This system will be applicable in the classification of small, medium and large concrete structures, in any construction project that requires the identification of damages. For instance, if the government wants to invest in the repairing of pavements that have been damaged and identify them efficiently and with no labour force, the system developed by this project is very appropriate. Besides, the system can be used for newly-built structures to rapidly identify structural failures and act on them before the construction is delivered. As an example, this system would have been very appropriate for the real-life case presented in a new building in Mascot, NSW, where people had to be evacuated because of the presence of cracks, according to Stewart & Kontominas (2019). As cracks start small and grow over time, it is essential to identify them before they become damaging, which is addressed by the system developed.

It is recommended that this system will be implemented in drone technologies, in order to replace the human participation in the process of identifying the type of structure and the presence of cracks. Moreover, to allow the identification of damages in structures where it is not easy for workers to get to, such as high buildings or underneath bridges. The objective to address is the early prevention of structure collapse or damage and the reduction of the use of expensive resources employed in this task. Thus, a drone controller can take a picture that he observes when flying the drone and pass it to the system to identify whether the image is cracked or uncracked. The system will also provide a classification of the type of structure, so the controller can determine what kind of material is being evaluated and take appropriate action with the relevant stakeholders.

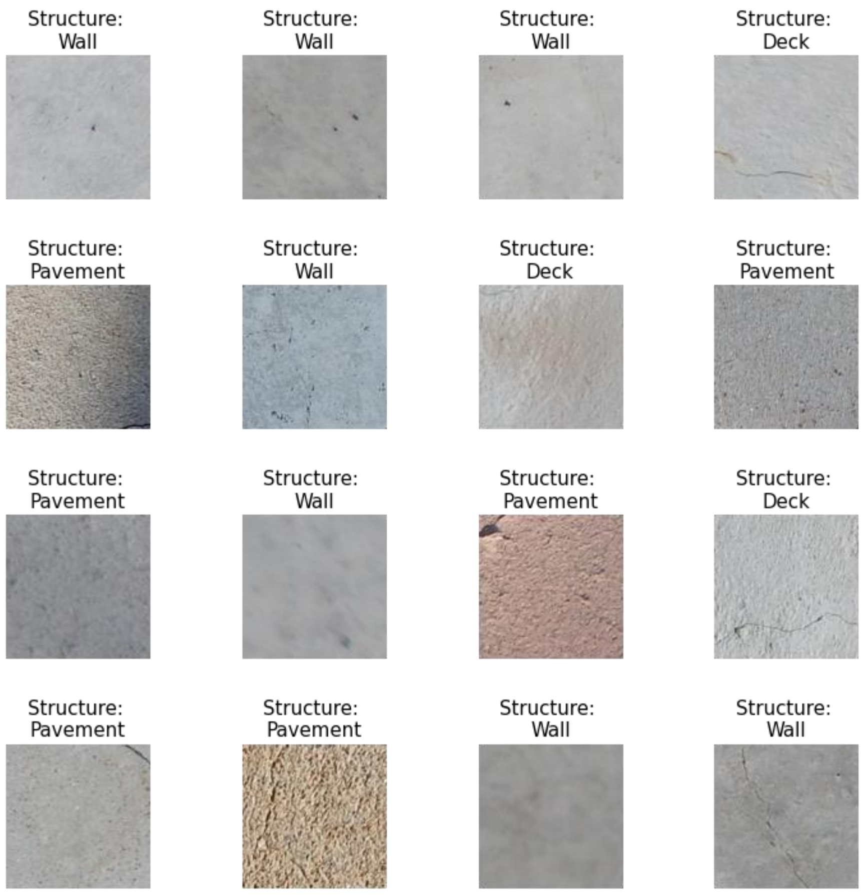
1.4 Dataset

The dataset used was chosen for this study is SDNET2018 dataset published by the Utah State University Library by the authors Maguire, Dorafshan & Thomas (2018) and publicly available in this link. The dataset contains 56,092 sized 256 x 256 images of three different types of structures, and each one of these structures includes images of a damaged (crack) or undamaged. Moreover, the cracked images contain cracks that vary from 0.06mm to 25mm, generating a complexed task for a machine learning model. The following table (Table 1) describes the distribution of classes in the dataset.

Structure	Cracked	Un-cracked	Total
Bridge Deck	2,025	11,595	**13,620**
Wall	3,851	14,287	**18,138**
Pavement	2,608	21,726	**24,334**
Total	**8,484**	**47,608**	**56,092**

Table 1: distribution of classes in the dataset.

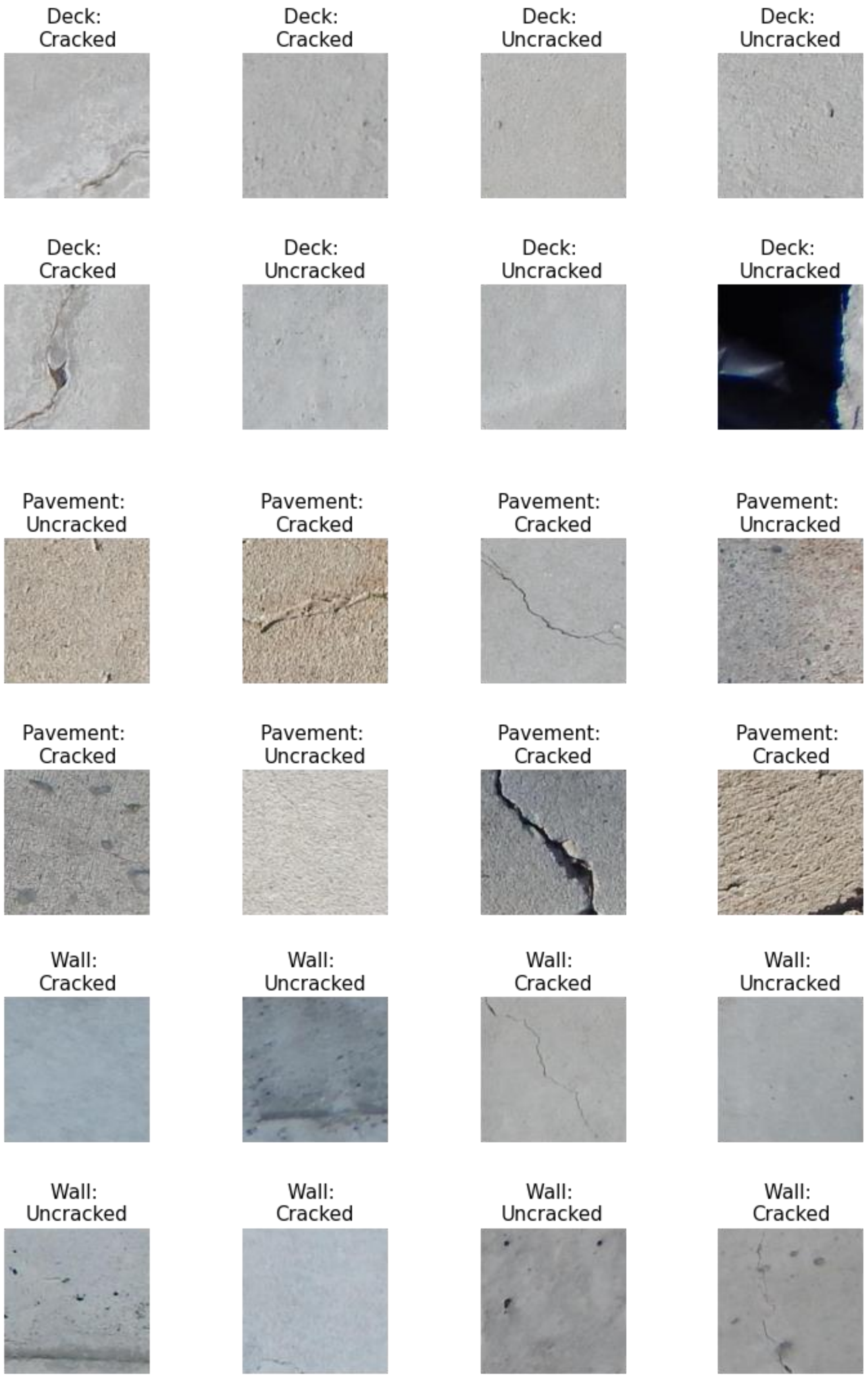
Furthermore, the following graph (graph 1) depicts a randomly chosen sample of 16 images from the dataset containing the three types of structures:



Graph 1: Multiclass random sample from the dataset.

It is possible to observe in a high-level that the structures present variations within the classes, which is considered to add a level of complexity to the machine learning task.

For the binary classification task, the following graph (graph 2) depicts a randomly selected sample of each one of the structures with both cracked and uncracked pictures.



Graph 2: Binary class (crack/uncracked) random sample from the dataset.

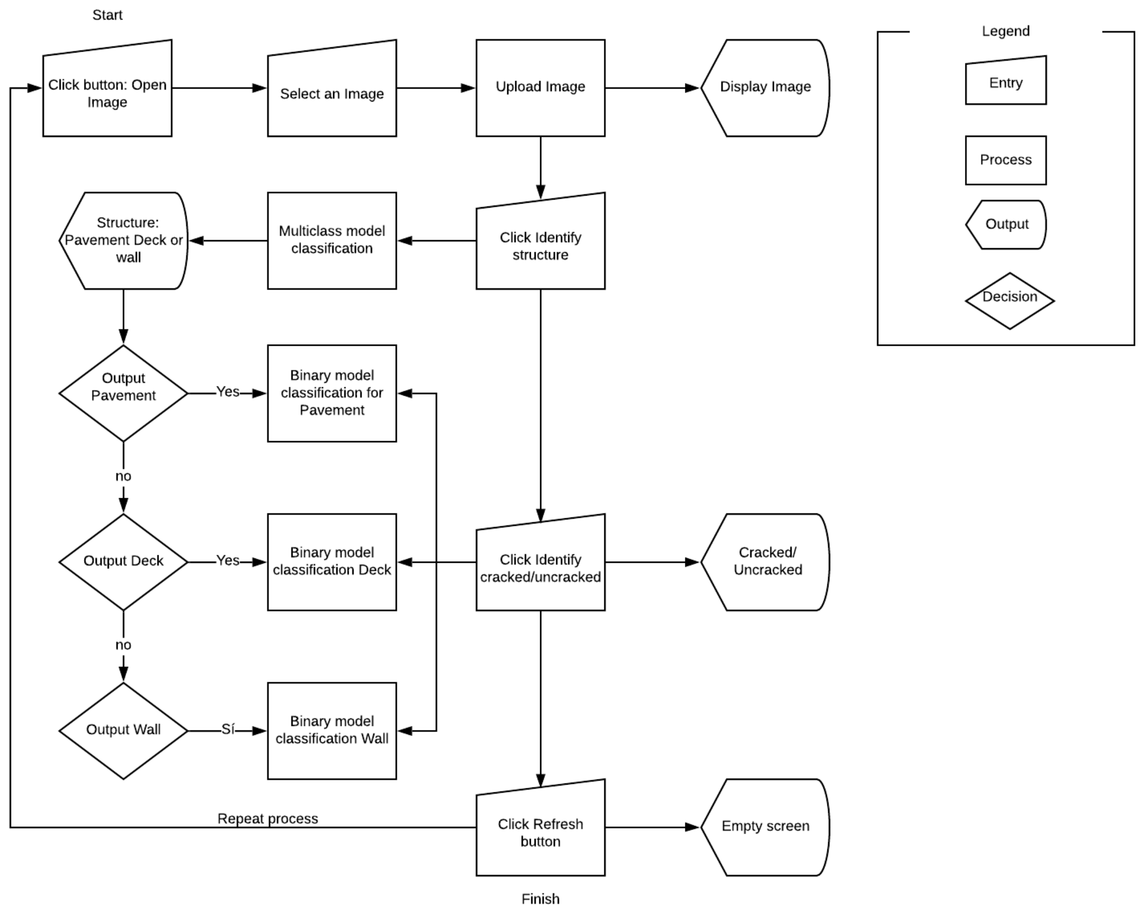
It is also possible to observe that there is a lot of noise present in the images, and also it is unclear in some of them whether this noise represents a crack or not. Moreover, for some of the structures, there are additional factors to consider, such as tiles separator lines that can appear to crack when in fact they are not. This was taken into consideration when developing the model's architectures because of the complexity of the task, pointing to complex CNN architectures.

2 Overview of the architecture/system

The system is composed of 4 trained models that have been trained with the classic architectures Inception ResNet v2 and VGG16, along with transfer learning from the ImageNet weights. It initially takes an image as an input that is given by the user; it displays it and allows the user to click on two buttons. One to perform the identification of the type of structure and the other one to identify if the structure is cracked or uncracked. The following section describes how the system works by using a flow diagram and then introduces the architectures that were used to train the models that perform the classifications in the system.

2.1 Flow Diagram

The following Flow Diagram depicted in graph 3 shows the different processes performed by the system.



Graph 3. Flow Diagram Concrete System

After training and imputing the models into the system, the process starts by requiring the user to click the button open image and select an image from a folder.

Subsequently, the system uploads the image displayed it on the screen, and transform the image into a NumPy array of size (1,256,256,3) for the models to perform classification.

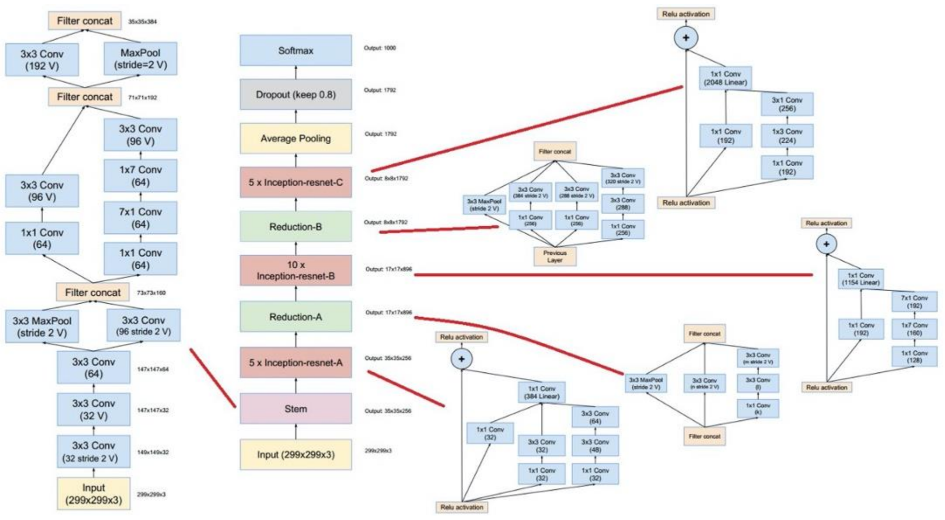
The user is prompted to click Identify structure, which will call the multiclass classification model with Inception ResNet V2 trained architecture for feature extraction, and a 4-layer fully connected layer architecture to perform the classification. If the classification is pavement, the system will use the model trained for the pavement class; if the classification is Deck the system will use the model trained for the deck class; and finally, if the classification is wall, the model will use the model trained for the wall class.

Thereafter, to obtain the binary classification, the user clicks identify cracked/uncracked, and the system will appropriately use the corresponding model for the binary classification, according to the multiclass result. After, it will display the result cracked or uncracked.

Finally, the user can refresh the system and environment by clicking the refresh button, that will delete all information to start with the classification of a new image.

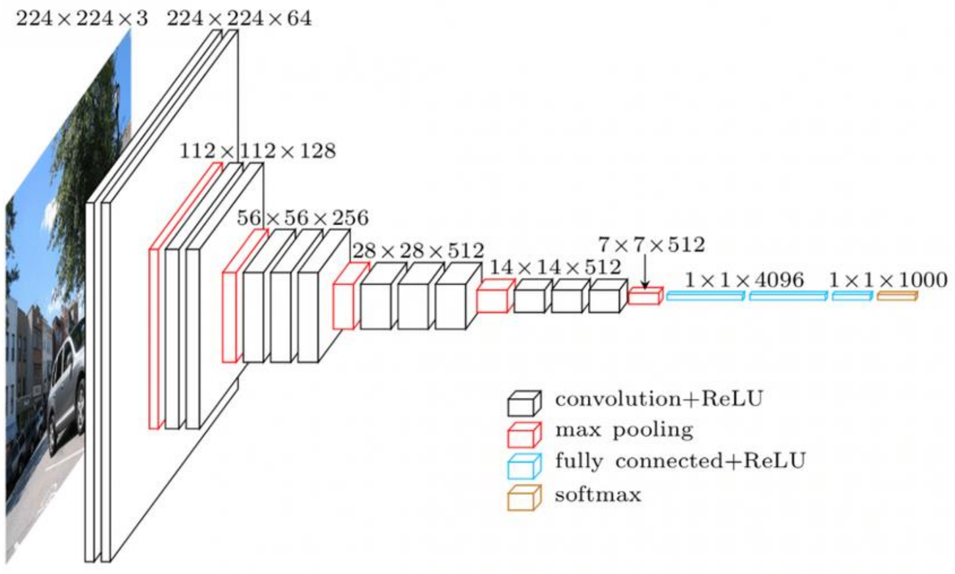
2.2 CNN Architecture Design

The team has initially tested six different types of architectures in order to achieve the best results for this project; these are ResNet101, ResNet152, VGG16, VGG19, Inception v3, and Inception ResNet v2. All six architectures were tested on our multiclass image classification wherein Inception v3, ResNet101, and ResNet 152, averaged accuracy scores of around 45-50, wherein VGG16, VGG19, and Inception ResNet v2 results from average in about 73-80 accuracy score. The team then decided to start fine-tuning these architectures, and out of the three of them, Inception ResNet v2 gave the team the best outcome. Hence, for the final architecture for our multi-image classification, the team chose the Inception of ResNet v2 architecture.



Graph 4. Inception ResNet v2 architecture [1]

With the last three architectures, the team then proceeded to apply it to our binary image classification problem where VGG16 outperformed the other two architectures, and VGG16 gave the team the best scores, at the same time since the team needed to create three models for our binary classification, VGG16 was a much lighter architecture as compared to the two and it was easier and faster to train, fine-tune, and improve to have the finest scores the team could achieve. Consequently, VGG16 was chosen by the team as the architecture for out binary classification models.



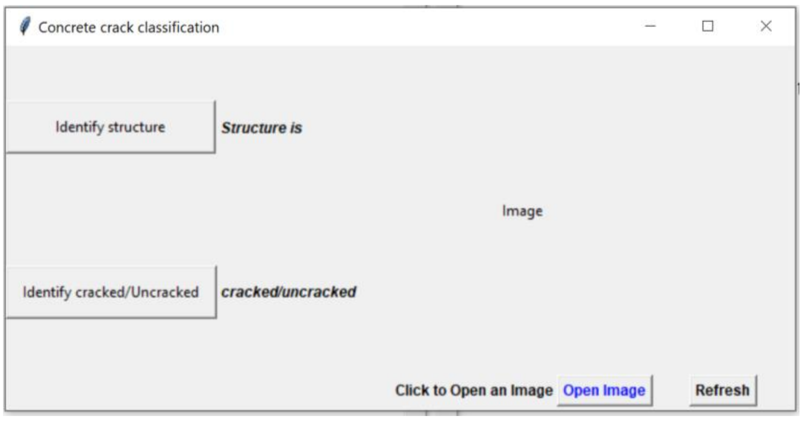
Graph 5. VGG16 architecture [2]

The results and settings from the team's final chosen architectures are presented in section 3 of this report.

2.3 GUI Design

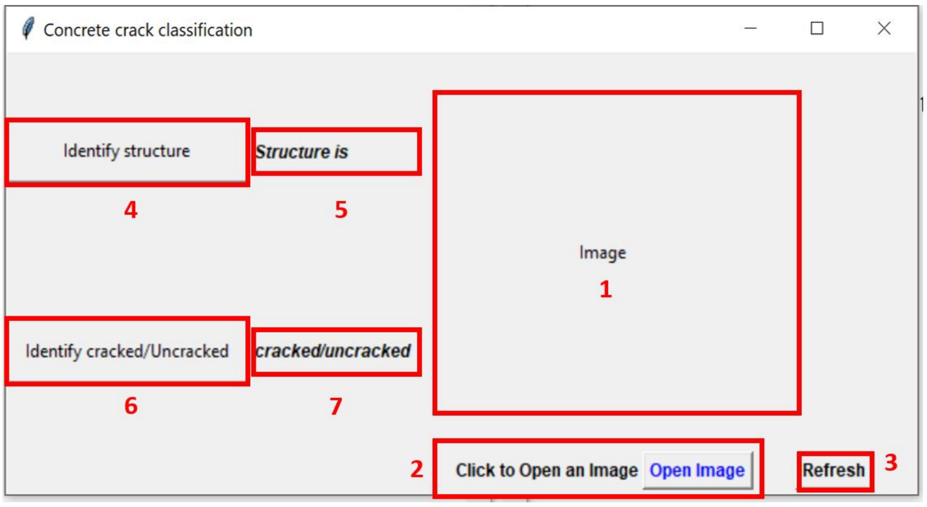
The Graphical User Interface GUI implemented in this project will facilitate the classification and visualisation of Bridge Deck, Wall and concrete surfaces.

Initially, the GUI will pop up as follow:



Graph 6. The initial representation of GUI

The user will be able to select an image and run actions "Identify Structures" and "Identify cracked/Uncracked."



Graph 7. Buttons and labels

GUI's functions of buttons and labels are defined as follow:

1: Image selected will be displayed in the area "1."

2: Open image button: This button will allow opening an image. The image selected will be displayed in square number 1.

3: Refresh button: This button will refresh the screen and will be ready to classify another image.

4: Identify Structure button: This button will identify the concrete structure (Deck, Pavements, Wall). Once the structure is identified, it will display in the area "4".

5: Label: Will display the structure identified.

6: Identify the cracked/Uncracked button: This button will identify if the structure identified is cracked or uncracked. Once this classification is done will be displayed in the area "6".

7: Label: Will display cracked or uncracked structure.

3 Results and Evaluation

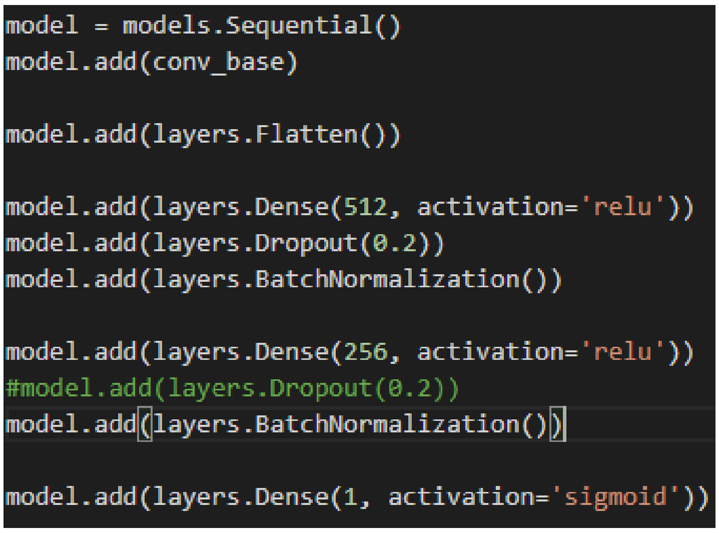
In this section, the team will be giving a brief explanation of the team's settings for each of the four models, as well as their respective results. As mentioned in the previous section, the team made use of the Inception ResNet v2 model for the multiclass image classification to detect different structures and the VGG16 architecture for the binary classification in order to detect if the structure classified in the multiclass classification is damaged or undamaged.

3.1 Experimental Settings

All 4 model experiments had to train with 20 epochs, and each model had its own steps per epoch and validation steps vary depending on the size of the data being trained under each model.

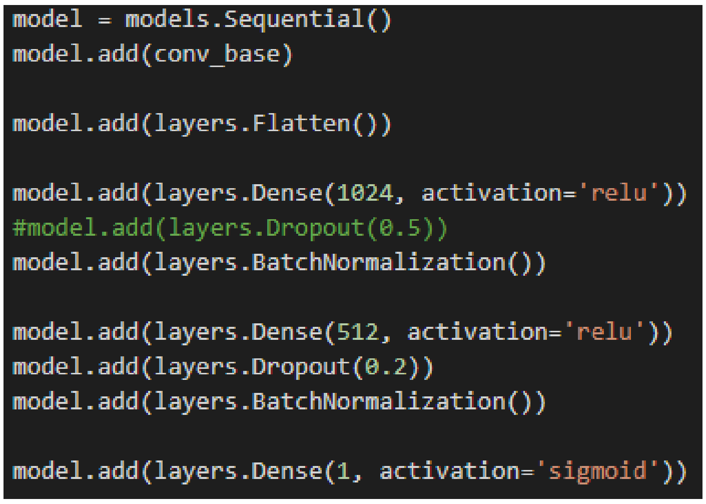
To improve the performance and accuracy of the multiclass image classification using the Inception ResNet v2 architecture, the team has fine-tuned the model by means of using transfer learning and set the weight to imagenet, freezing all the layers and using the last 7, and added fully connected layers. The team added at a total of 5 fully connected layers, with the first layer having a Convolutional 2D layer with 1024 neurons, a kernel size of (1,1) and activation set to ReLU, a flatten layer to transform the dimensions, a dropout layer set to 0.1 to avoid overfitting and underfitting of the model, and finally a batch normalisation layer to improve the performance. The last 4 fully connected layers consist of Dense layers, with the first 2 containing 1024 neurons, the 4th fully connected layer containing 256 neurons, and the output layer consisting of 3 neurons. The dense layers consisting of neurons of 1024, additionally have dropout layers set at 0.1 as well as a batch normalisation layer, the dense layer with 256 neurons also has an additional batch normalisation layer and a dropout layer set to 0.1 as well. All 3 Dense layers except for the last layer had activation set to ReLU and the last having activation set as softmax. Finally, the optimiser used for the multiclass model was Adamax, as it gave the team better results as compared to Adam, RMSprop, and SGD.

The team used the same steps from the multiclass image classification model for each of the binary class models, but each binary class model had different fully connected layers implemented to it for when the team was experimenting and running all models, the accuracy scores were totally different from one another. The screenshots below will be the final fully-connected layers the team used for each binary class model.



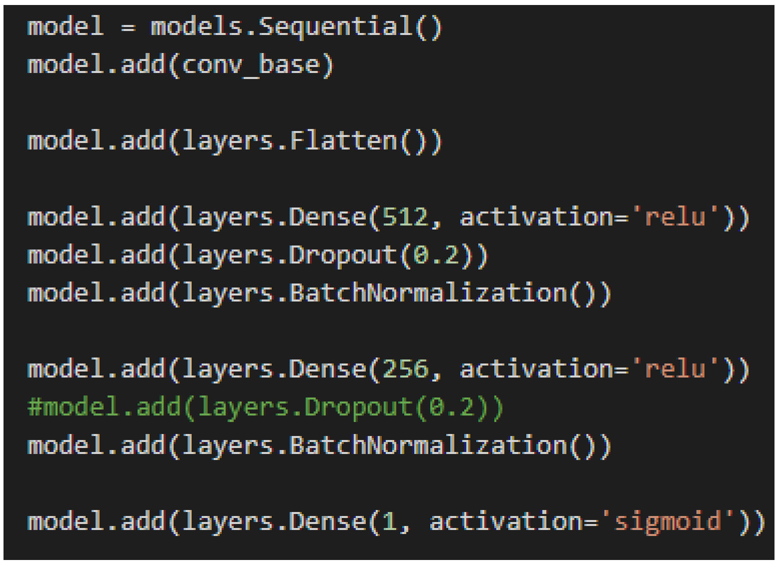
```
model = models.Sequential()
model.add(conv_base)

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.BatchNormalization())

model.add(layers.Dense(256, activation='relu'))
#model.add(layers.Dropout(0.2))
model.add(layers.BatchNormalization())

model.add(layers.Dense(1, activation='sigmoid'))
```

Graph 8. Screenshot of fully connected layers for Wall



```
model = models.Sequential()
model.add(conv_base)

model.add(layers.Flatten())

model.add(layers.Dense(1024, activation='relu'))
#model.add(layers.Dropout(0.5))
model.add(layers.BatchNormalization())

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.BatchNormalization())

model.add(layers.Dense(1, activation='sigmoid'))
```

Graph 9. Screenshot of fully connected layers for Pavement



```
model = models.Sequential()
model.add(conv_base)

model.add(layers.Flatten())

model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dropout(0.2))
model.add(layers.BatchNormalization())

model.add(layers.Dense(256, activation='relu'))
#model.add(layers.Dropout(0.2))
model.add(layers.BatchNormalization())

model.add(layers.Dense(1, activation='sigmoid'))
```

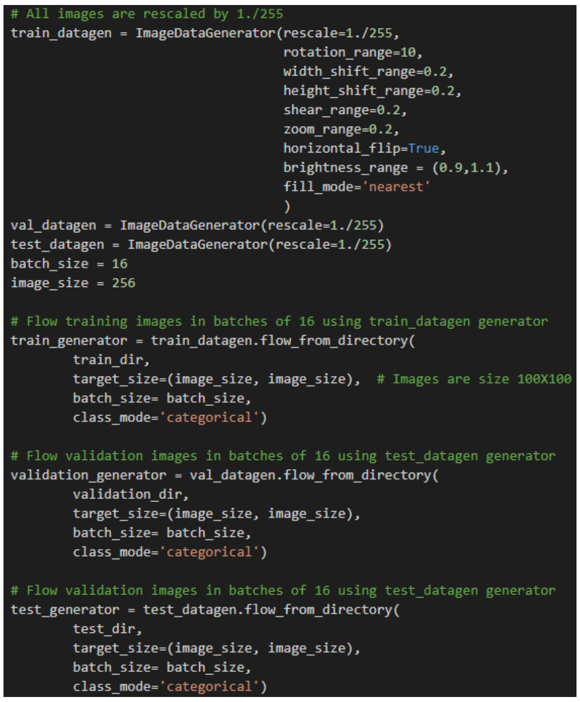
Graph 10 Screenshot of fully connected layers for Bridge Deck

As seen above are the screenshots of the fully connected layers used for each of the binary class models. For all VGG16 models, the team froze all layers except for the last 4, as well as instead of using softmax set for activation for the output layer of our binary class models; the team set sigmoid as the activation. For wall and pavement, the optimiser used was RMSprop, and as for the bridge deck, the team implemented Adam for the team wasn't obtaining the desired output accuracy when running the model using RMSprop.

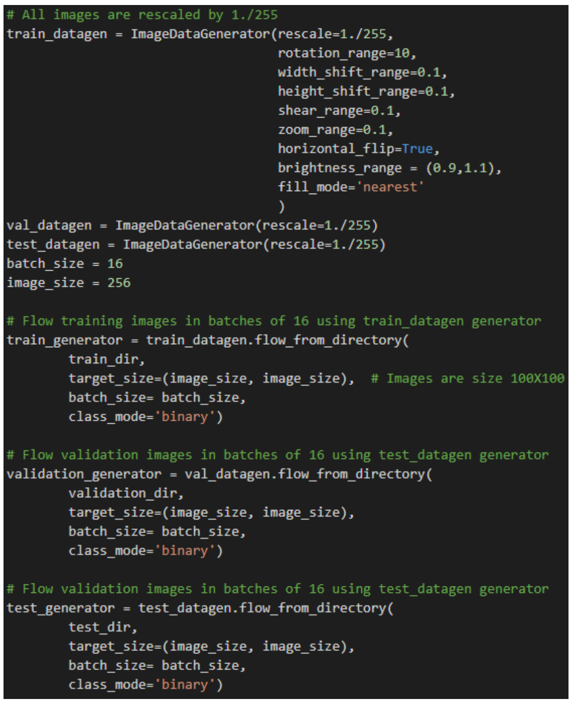
3.2 Pre-Processing

During the pre-processing of the initial dataset, the team encountered a few problems, such as data imbalance which was the biggest issue, with regard to undamaged images having way more data as compared to damaged ones. The team then resolved this issue by splitting the dataset to undamaged having 5% more images than the damaged class. By doing this, the team achieved of having a more balanced dataset for the modelling phase of this project.

After finalising the dataset to be used for the modelling process, the team deemed that the data to be used needed to be augmented, to achieve this the team then implemented Image Data Generator to all the models for both multiclass image classification and binary classification. The screenshots below will present the hyperparameters using Image Data Generator the team set for each of the types of classification.



```
# All images are rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=10,
                                   width_shift_range=0.2,
                                   height_shift_range=0.2,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True,
                                   brightness_range = (0.9,1.1),
                                   fill_mode='nearest'
                                   )
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
batch_size = 16
image_size = 256

# Flow training images in batches of 16 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(image_size, image_size),  # Images are size 100X100
        batch_size= batch_size,
        class_mode='categorical')

# Flow validation images in batches of 16 using test_datagen generator
validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(image_size, image_size),
        batch_size= batch_size,
        class_mode='categorical')

# Flow validation images in batches of 16 using test_datagen generator
test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(image_size, image_size),
        batch_size= batch_size,
        class_mode='categorical')
```



```
# All images are rescaled by 1./255
train_datagen = ImageDataGenerator(rescale=1./255,
                                   rotation_range=10,
                                   width_shift_range=0.1,
                                   height_shift_range=0.1,
                                   shear_range=0.1,
                                   zoom_range=0.1,
                                   horizontal_flip=True,
                                   brightness_range = (0.9,1.1),
                                   fill_mode='nearest'
                                   )
val_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)
batch_size = 16
image_size = 256

# Flow training images in batches of 16 using train_datagen generator
train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(image_size, image_size),  # Images are size 100X100
        batch_size= batch_size,
        class_mode='binary')

# Flow validation images in batches of 16 using test_datagen generator
validation_generator = val_datagen.flow_from_directory(
        validation_dir,
        target_size=(image_size, image_size),
        batch_size= batch_size,
        class_mode='binary')

# Flow validation images in batches of 16 using test_datagen generator
test_generator = test_datagen.flow_from_directory(
        test_dir,
        target_size=(image_size, image_size),
        batch_size= batch_size,
        class_mode='binary')
```

Graph 11 Screenshot of Image Data Generator implemented for multiclass (Left)and for binary classification (right)

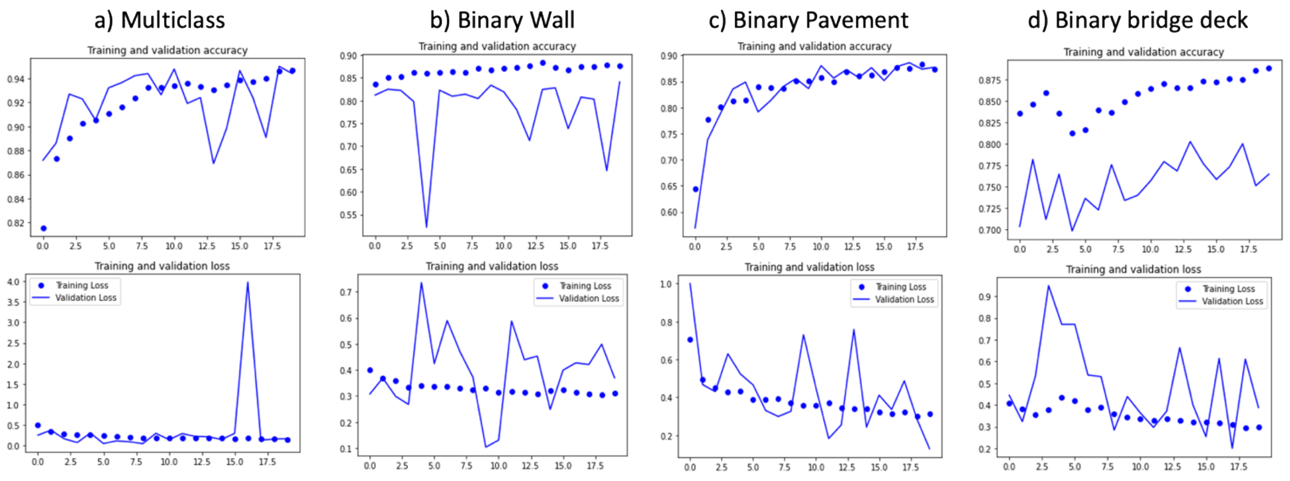
3.3 Experimental Results

The team's desired accuracy for each model was to be achieving a score of above 85 in order for this project to be a success, and having the GUI perform with minimal problems. After a number of runs and experimentations from our fine-tuning, each model, as explained above the team was able to obtain these results. To represent these results, the next images will be screenshots of graphs and a table of each of the models.

Model	Train acc/loss	Val acc/loss	Test acc
Multi-class	Acc: 0.9465 Loss: 0.1548	Acc: 0.9439 Loss: 0.1710	Acc: 0.929
Binary (Wall)	Acc: 0.876 Loss: 0.309	Acc: 0.841 Loss: 0.369	Acc: 0.924
Binary (Pavement)	Acc: 0.873 Loss: 0.312	Acc: 0.877 Loss: 0.129	Acc: 0.919
Binary (Bridge Deck)	Acc: 0.888 Loss: 0.299	Acc: 0.764 Loss: 0.388	Acc: 0.952

Table 2. Results for each model

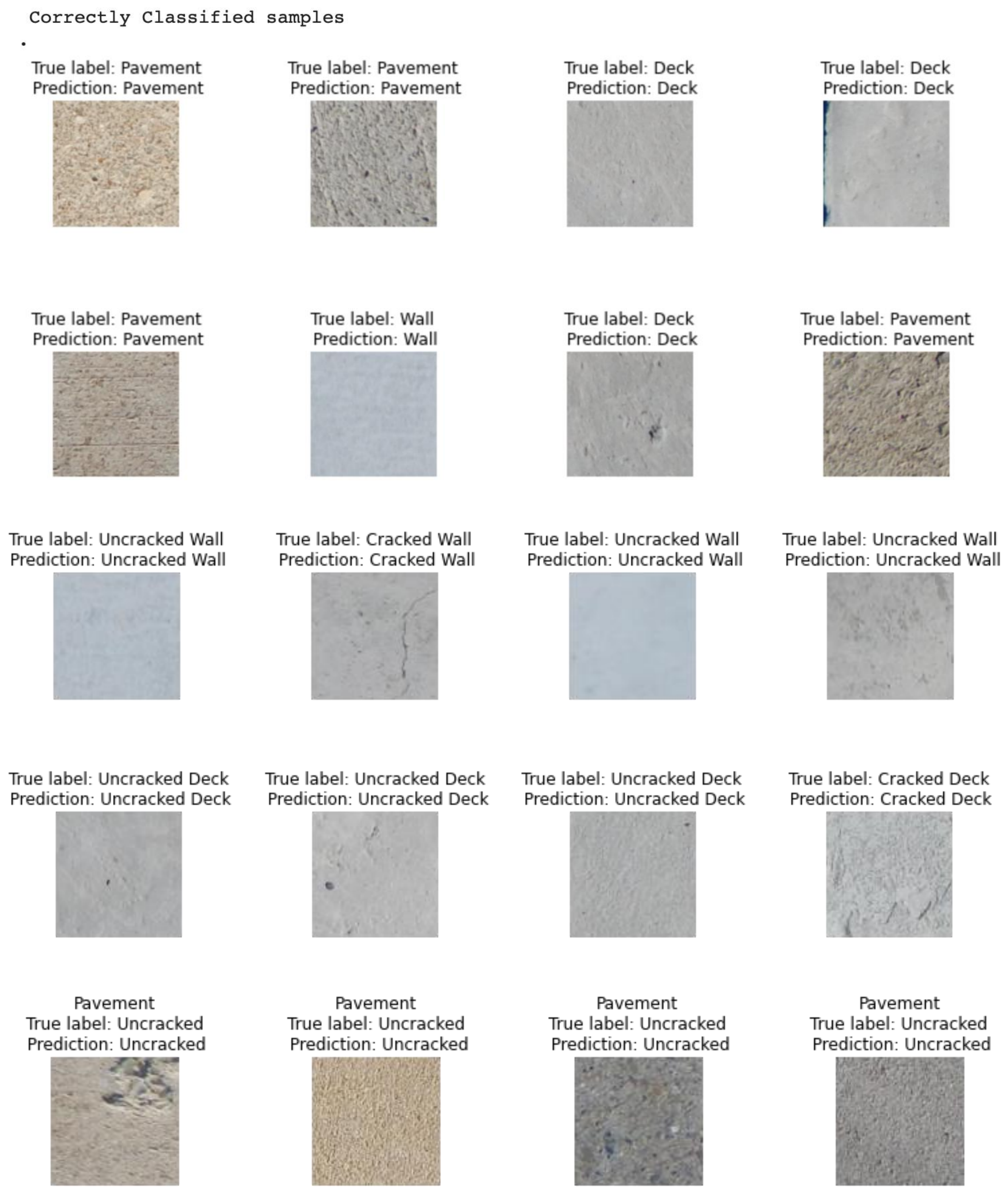
The results for training and validation accuracy and loss shown in table 2 are from the last or 20th epoch running and evaluating the model and the test accuracy for each model are from the best weights that were saved while training and running the model. The following images will represent the graphs of each model's training and validation accuracy and loss.



Graph 13. Training and validation accuracy and loss for the models trained.

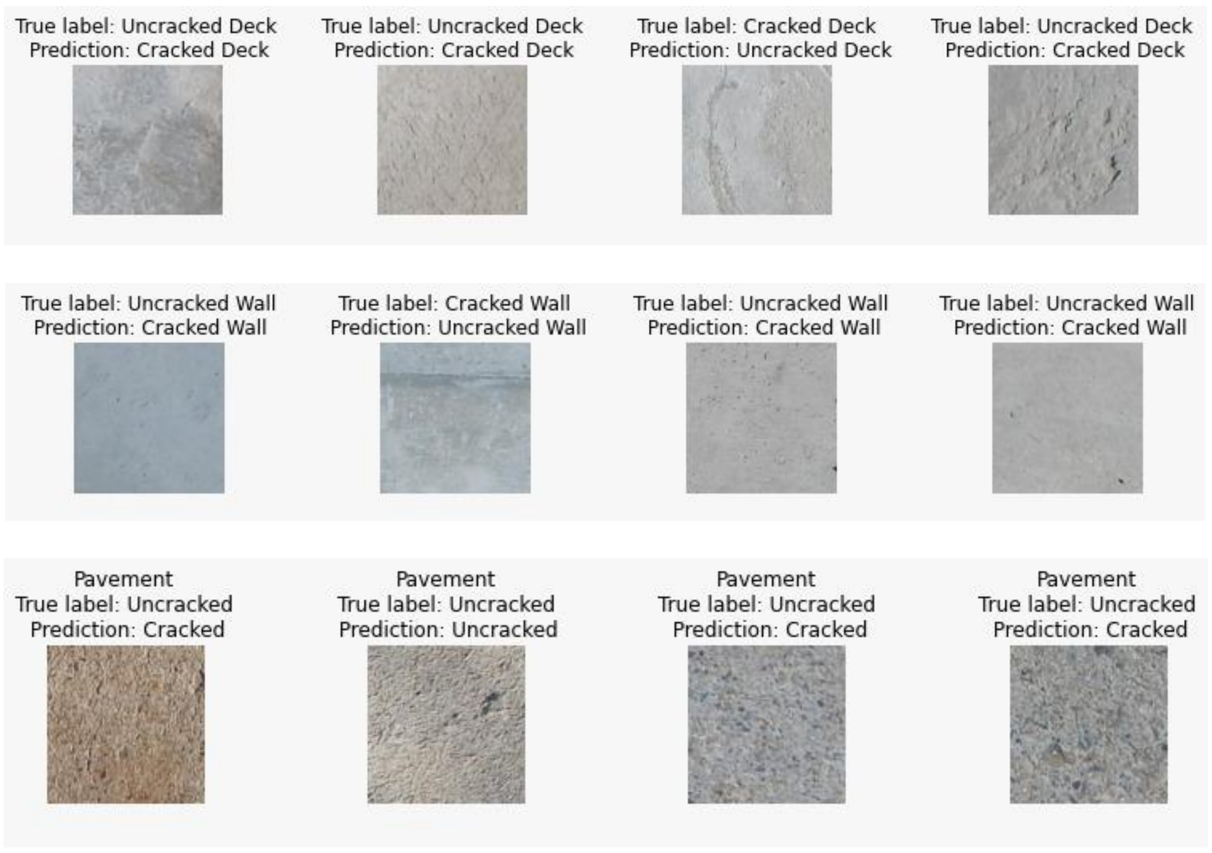
Graph 13 is the results of training each model after the fine-tuning and evaluation of our models. As seen in these graphs, the validation accuracies and losses are erratic as compared to all models' training accuracies and losses having a more stable movement for all the models.

Additionally, there were 36.954 correctly classified images by the best multiclass model, out of 39.797 test samples. Regarding the binary class Deck, the experiment best model correctly classified 8.795 out of 9.242 test samples, or only 447 misclassifications. Moreover, the best model for binary classification of the class Wall classified correctly 10192 out of 11032 test samples and had only 840 misclassified samples. Lastly, the best model for the binary classification in the class Pavement, the model correctly classified 17547 out of 19523 test samples, and only presented 1976 misclassifications. The following graph (graph 6) shows a small randomly chosen sample of the correctly classified images



Graph 14. Correctly classified images

Although the model achieves high accuracy in the four models presented, improvement in the model will highly encourage in order to achieve higher accuracy in the classification of the cracked images. Some Misclassified samples are shown in graph 14.



Graph 14. Misclassified samples

3.4 Limitations

As the system is required to be implemented in a drone, it would be still necessary to use a human to control the drone and take pictures. Although the system reduces the need for the physical presence of humans at the spot where the evaluation of the structure is required, it is still not completely automated. Moreover, only three types of structures were given into the models, which indicates that the system would only address Pavement structures, Wall structures and Bridge Deck structures. Different structures will require new training and most likely new models with different architecture settings. In addition, to have a fully automated drone that does not require human controlling, it would be necessary to develop a model that addresses object detection (which in this case would be crack detection) in a developed application that is efficient like SSD MobileNet or YOLO. If developed and implemented in an autonomous drone, the monitoring of the different structures would be completely automated.

4 Discussion and Conclusions

The implementation of deep learning algorithms in the classification of cracked structures have demonstrated one more time a large number of applications where computing vision techniques are able to reduce cost and increase the accuracy in the solution of different and complex tasks. The model implemented in this project is applicable in various sectors such as construction and government, where the classification of cracked structures is essential for the maintenance of small, medium and large engineering structures. Current methodologies in the classification of cracked images are costly and inaccurate; therefore, this project analysed a Convolutional Neural Network architecture able to reduce the cost in the identification of cracked images and increase the accuracy. The application of this model will be widely used and will bring a large number of benefits such as early classification of possible cracked and the reduction of resources in the execution of this task.

The architecture presented in this project showed high accuracy in the test data. However, further, improvement will be acceptable in order to achieve an outstanding solution. Furthermore, this project classifies three different structures, adding more structures will increase the application of the model on a broader number of industries.

5 References

[1] Hewage, R. 2020, '*Extract Features, Visualise Filters and Feature Maps in VGG16 and VGG19 CNN Models'*, viewed June 17th, 2020, <https://towardsdatascience.com/extract-features-visualize-filters-and-feature-maps-in-vgg16-and-vgg19-cnn-models-d2da6333edd0>

[2] Koehrsen, W. 2017, Object Recognition with Google's Convolutional Neural Networks, viewed June 17th, 2020, <https://medium.com/@williamkoehrsen/object-recognition-with-googles-convolutional-neural-networks-2fe65657ff90>

Maguire, M., Dorafshan, S. & Thomas, R.J. 2018, 'SDNET2018: A concrete crack image dataset for machine learning applications. ', Utah State University.

Stewart, S. & Kontominas, B. 2019, 'Sydney high-rise in Mascot evacuated after residents spot cracks in building', *ABC News*.

Appendices

A. Individual Contribution

Each group member should write half-page (max) about his or her contribution to the project development activities.

Andres: I contributed to the group's progress by experimenting, running and improving the multi-classification model and the identification of the cracked or uncracked models. Besides, I helped to design and code the GUI, that was able to create a user-friendly interface in the classification of cracked structures and the elaboration of the final report. I believe that the final result of this project has been achieved by the collaboration of the three members of the group. Without the cooperation of any of the members, we would not have been able to achieve the final result of this project.

Carlos: I contributed to the group's progress by building the models' initial infrastructures, as well as setting up the environments and experimental models for training. I did the initial training of the models and gave recommendations for training to my team mates. I helped debugging the GUI after it was developed and helped introducing the models into the GUI for correct performance. Communication was always great and the team was always available for zoom calls to show results, test, perform debugging remotely as only Ernest had the TensorFlow environment installed, etc. It was great working with my teammates.

Ernest: I contributed to the group's progress by finding the dataset to be used in the project, experimenting, running, and improving the four models that were used, as well as helping with the GUI and documenting the video for the GUI demonstration. I believe that each member did all the parts equally and always helped each other out, communicated correctly and that we work well as a team that we were able to be successful in completing the project.

B. Individual Contribution Split

Team Member Name	**Percentage**
Andres Lagos	**33.3%**
Carlos Carvajal	**33.3%**
Ernest Ilustre	**33.3%**

All the team members have discussed and agreed on the above individual contribution to the project.

Carlos Carvajal: Carlos

Andres Lagos: Andres

Ernest Ilustre: Ernest