|  |  |
| --- | --- |
| Joseph Ambrose | 20142218 |
| Thomas Ludwig | 20246182 |
| Kelly McConvey | 20229337 |
| Hamid Baz Mohammadi | 20252900 |
| Taha Shafique | 20245433 |
| Anuj Singh | 20250333 |
| Jichen Song | 20197262 |
| Selina Wang | 20251610 |

Crack Detection Project

MMAI 894

Contents

[Executive Summary 3](#_Toc68438096)

[Business Opportunity 3](#_Toc68438097)

[Modelling 3](#_Toc68438098)

[Next Steps 3](#_Toc68438099)

[Implementation 4](#_Toc68438100)

[Business Opportunity 5](#_Toc68438101)

[Modelling 6](#_Toc68438102)

[Model Selection 6](#_Toc68438103)

[Primary Dataset 6](#_Toc68438104)

[Secondary Dataset 7](#_Toc68438105)

[Preprocessing 8](#_Toc68438106)

[Data Preparation 8](#_Toc68438107)

[Optimizing Performance 9](#_Toc68438108)

[The Modelling Journey 9](#_Toc68438109)

[High-Level Model Architecture 9](#_Toc68438110)

[Model Building and Selection 10](#_Toc68438111)

[Model Training and Validation 10](#_Toc68438112)

[Model Testing 11](#_Toc68438113)

[Convolution Neural Networks (CNN) 1 12](#_Toc68438114)

[Convolution Neural Networks (CNN) 2 13](#_Toc68438115)

[Convolution Neural Networks (CNN) 3 14](#_Toc68438116)

[Convolution Neural Networks (CNN) 4 & 5 16](#_Toc68438117)

[VGG16 Pretrained Model 17](#_Toc68438118)

[Resnet50 pre-trained model 18](#_Toc68438119)

[Other Pre-Trained Models 19](#_Toc68438120)

[Augmentation of the Dataset 20](#_Toc68438121)

[Summary of All Models and Key Conclusions 22](#_Toc68438122)

[Determining the Size of the Crack 23](#_Toc68438123)

[Next Steps 24](#_Toc68438124)

[Implementation 25](#_Toc68438125)

[Conclusion 26](#_Toc68438126)

[APPENDIX 28](#_Toc68438127)

[Resources Used 36](#_Toc68438128)

[References Used 37](#_Toc68438129)

# Executive Summary

## Business Opportunity

Detecting and repairing cracks in concrete and other materials early can help minimize the cost of repairs and prevent structural failures. Using supervised deep learning image classification, we can create a model to detect surface cracks. This model can be used by construction companies, engineers, and city infrastructure surveyors to examine industrial and road concrete. This solution may be especially useful for developing countries, where construction requirements are less stringent, labour expertise is limited, and civil construction failures are frequent (Fernandez R. H., 2018) (Ker Than, 2013).

Two sets of datasets have been used for this initiative. One set has been used for training; the other set has been used for testing (SDNET). This ensures our model can handle images which it has not been trained on, improving generalizability, and help us understand the feasibility of using this model in a production environment, where images can differ vastly.

## Modelling

Of all the models we tried, the pre-trained ResNet50 model had the best performance. We were able to obtain an average F1 accuracy score of 84.5% on the testing dataset. Attempts to augment the dataset helped increase this score to 84.7%. Additionally, as a proof of concept, a model was developed to obtain the size of the crack. More investigation will be required to understand if our assumptions in this model were valid.

## Next Steps

More investigation will be required to reach our desired F1 score of 95%. This could be due to the quality of SDNET images, their variation from concrete\_data images, or issues with the SDNET images being mislabeled. Further assessment of the model is required using real images collected by the engineering companies we’ve targeted as potential customers. Furthermore, additional business value could be achieved by adding recommendation logic to prioritize crack repair based on urgency.

## Implementation

To scale our model for production, we recommend using a mobile phone, security cameras or a drone to take pictures or video of the site. Special considerations will need to be given on how our training data is stored and catalogued, environmental biases, and whether our model retrieves real-time recommendations or runs a batch job to retrieve the recommendations. Depending on a client’s requirement, both can be considered with a higher price for real time retrievals.

# Business Opportunity

In the construction industry, monitoring the health and integrity of concrete structures is an important preventative maintenance task to ensure the safety of the structure. Concrete provides structures with strength, and resilience from deformation. These characteristics, however, result in concrete structures lacking the flexibility to move in response to volume or environmental changes. Cracking is usually the first sign of distress in concrete, which can occur due to “thermal stress, poor construction practices, weathering, or overloading”. (SCIENTIFIC, 2019) A cracks size ranges from micro-cracks that expose the concrete to salt, to larger cracks caused by external loading conditions. These cracks can cause foundation issues which may lead to structural failures and result in injury and loss of life. (SCIENTIFIC, 2019)

Detecting and repairing cracks early, can help minimize the cost of repairs and prevent structural failures. A periodic manual inspection is normally done to ensure the site is safe. This is a laborious and lengthy process that carries a degree of subjectivity and requires expertise by the inspector to determine whether the crack is structural in nature and what action needs to be taken to fix the problem. Performing crack detection manually requires physical presence of the inspector who would have to manually inspect all areas of the site. A manual assessment has limitations on the volume of concrete that can be examined as some can be inadvertently missed which would potentially allow cracks to spread and threaten the structural integrity of the concrete.

Commercial, residential, and industrial buildings go through regular engineering inspections that include structural assessment of concrete surfaces. Bridges, roads, tarmacs also require visual inspections to ensure safety of those who use these structures and the surrounding environment.

This paper will present a supervised deep learning image classification crack-detection model which uses labeled images of concrete with and without cracks. A variety of deep learning algorithms were experimented with for model development. A fully connected Convolution Neural Network (CNN), ResNet and other pre-trained models were included. To evaluate our model, we primarily used the F1 accuracy metric. We also attempted to minimize false negatives as we did not want to classify images that have cracks incorrectly, as this could lead to structural failure.

# Modelling

## Model Selection

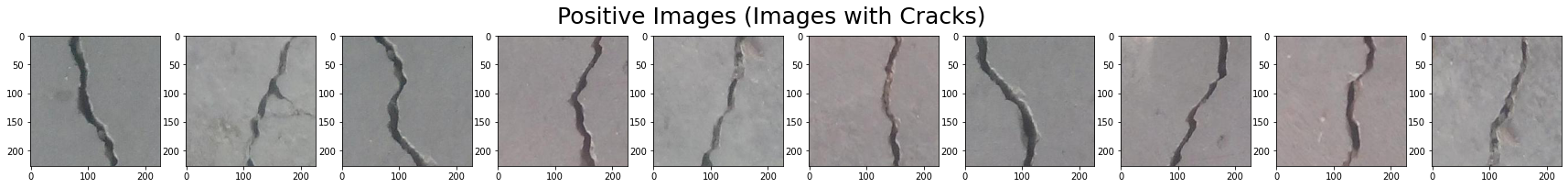
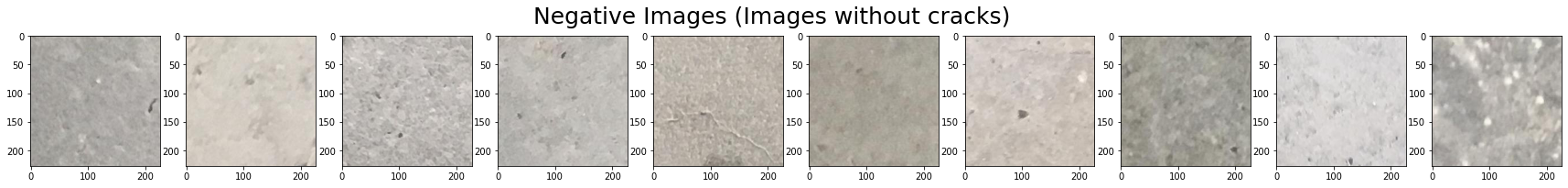
Deep learning is an appropriate solution for real-life problems that involve processing large amounts of information and that require constant oversight, and that have sufficient data available. Additionally, routine tasks with predefined outcomes are good candidates for automated decision making with machine learning.

Crack detection requires the evaluation of many images of concrete surfaces with only a small percentage of positive cases, resulting in many images without cracks. The outcome is clearly defined, as an image will either be categorized as having a crack or not.

## Primary Dataset

Our primary dataset contains 40,000 images of concrete, 20,000 with cracks and 20,000 without cracks. This dataset is publicly available to use at <https://data.mendeley.com/datasets/5y9wdsg2zt/2>. (Özgenel, 2019) The images include concrete surfaces of different colors and cracks of different sizes, shapes, and orientations. Each image has a resolution of 227x227 pixels in RGB. These images represent photos of real concrete taken in high resolution and sliced into smaller images (227×227 pixels).

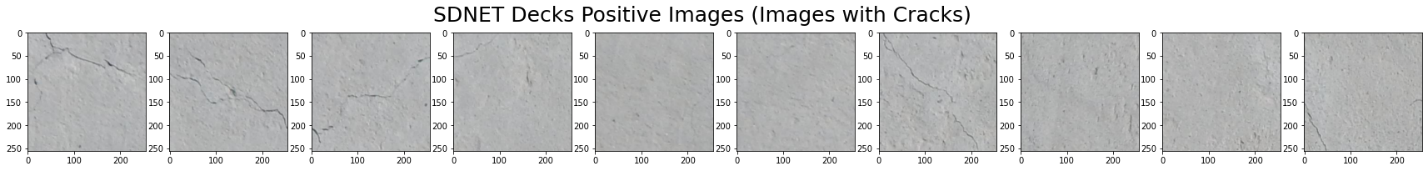
Example – Positive Images with Cracks and Negative Images without Cracks



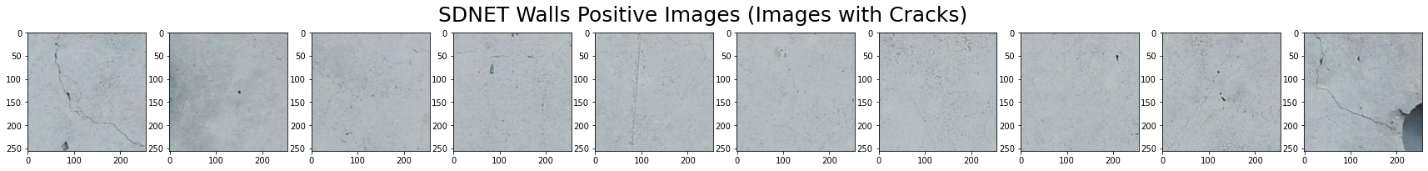
## Secondary Dataset

We also used the SDNET dataset containing 56,092 images representing decks (13,620), pavements (24,334), and walls (18,138). All images are 256×256 pixels in RGB. The dataset includes cracks as thin as 0.06 mm and as thick as 25 mm. The images also contain different types of obstructions including shadows, surface texture, scaling, edges, holes, and background debris. (Maguire, 2018)  With it, we tested the performance of our crack detection model on surfaces beyond which it had been trained. The performance of our model using this dataset aided us in determining the feasibility of using this model in a production environment, where images can vastly differ.

Example Cracked Images (Decks / Pavements/ Walls)







## Preprocessing

All images were resized to have 227×227 pixels. The label was encoded categorically (positive/negative class). Further pre-processing operations were performed using VGG16 preprocess input function, which primarily performs the normalization of colors.

## Data Preparation

The primary dataset of 40,000 concrete data images were randomly split into training (80%) and validation (20%) sets and used to build and validate models. The training set included 31,998 images (15,999 positive and 15,999 negative) and the validation set was comprised of 8,002 images (4,001 positive and 4,001 negative). The SDNET dataset was used in testing to evaluate the true model performance.

## Optimizing Performance

Considering the large volume of data, ImageDataGenerator object and the flow\_from\_directory function were used to optimize performance. This approach allowed the model to iterate through a batch of 100 images loaded online at the same time. The code snippet below demonstrates this approach.



## The Modelling Journey

### High-Level Model Architecture

The diagram below outlines the deep learning neural network model architecture selected for this project. Image classification tasks are typically performed using CNN (Convolution Neural Network) and pre-trained models. After pre-processing, training images enter the neural network model (CNN or pre-trained) in the input tensor with dimensions 227×227 and three channels. The last two layers in the model are a Dense layer and a Softmax layer with two nodes representing the output probabilities of an image being positive or negative.

CNN models have one or more convolutional layers, max or average pooling, flattening, dropout and dense layers. Pre-trained models are integrated within neural networks as the first layer with the calculated weights used as inputs into trainable dense layers. Models are built using the designated training data and weights are adjusted using the validation data. Model performance is assessed using out-of-sample data (SDNET images for testing). The accuracy of the model varies greatly between classification tasks, quality of the image, complexity of the task, etc. Typically, models with 95% accuracy are considered acceptable for implementation and may be considered for production. (8 Top-funded Facial Recognition Startups, 2019)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Re-sizing and  Normalization  of colors,  Augmentation | Input tensor 227×227×3 | Neural  network machine learning model  CNN or Pre-trained | Dense layer | Softmax layer with 2 nodes | Probability of “positive”  image (with crack) |
| Probability of “negative”  image (without crack) |
| **Images of  concrete surface** | **Pre-processing** |  | **Deep learning** | | | **Model outcome** |

### Model Building and Selection

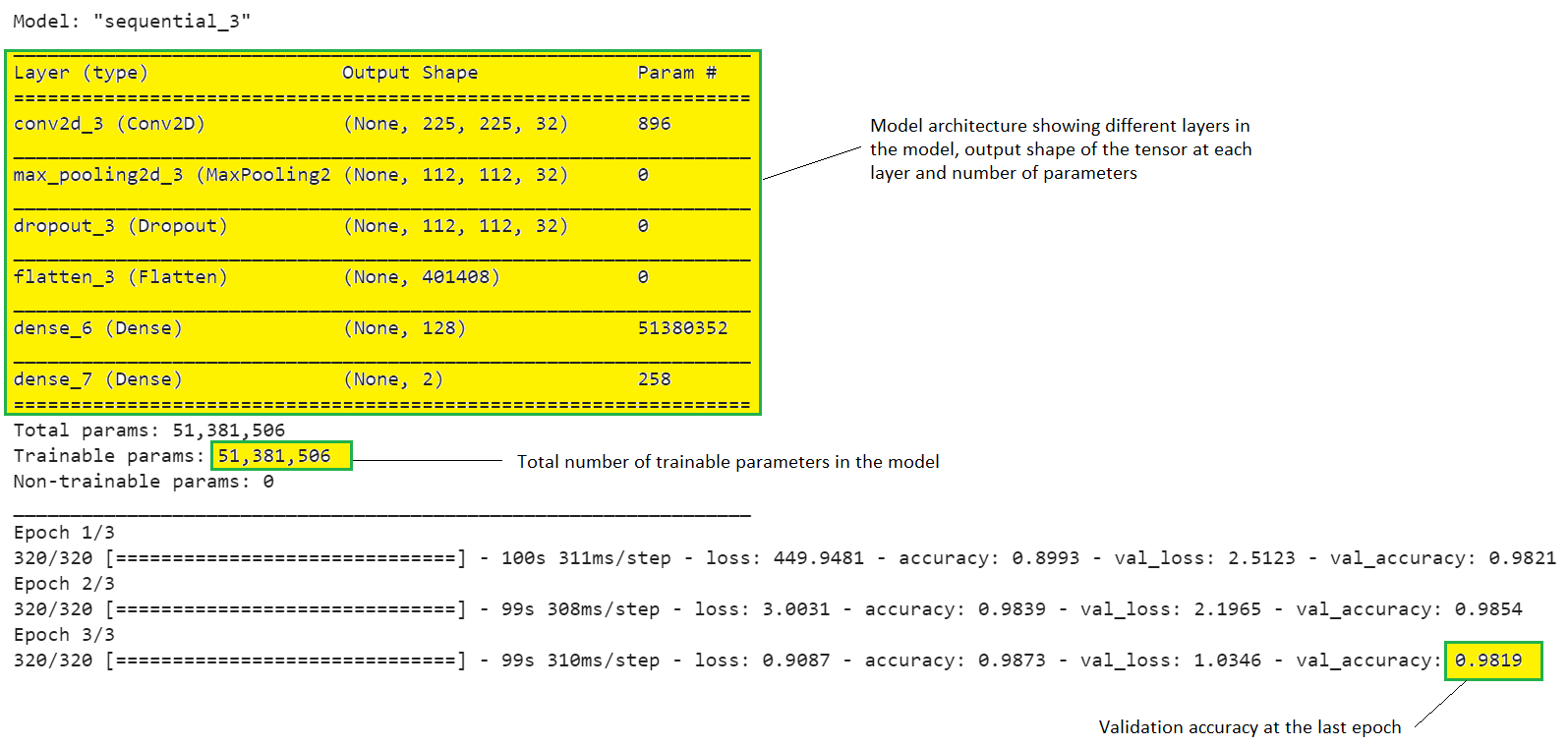
For this project, we examined a variety of ML models. Several CNN variations were run using different hyperparameters, as well as pre-trained models such as VGG16, ResNet50, Xception, InceptionV3, NASNetLarge, and DenseNet. CNN models were built with hyperparameter tuning with batch normalization, different configurations of convolution layers and neurons in the final layers. All models were tuned using the ADAM optimizer, the categorical cross-entropy loss function and accuracy metrics. At the last layer, all models have a Dense layer with two classes and a softmax activation function.

### 

### Model Training and Validation

Training was conducted using batch size 100 with between three and five epochs. Three epochs for CNN models and five epochs for pre-trained models were found sufficient to achieve over 0.97 validation set accuracy.

The example below shows the model architecture and training results.



In the above model, there are six layers: convolution 2D with 896 parameters, max pooling, dropout, flatten, dense layer with 128 nodes (51380352 parameters) and final dense layer with 258 parameters. Total number of trainable parameters is 51381506. The model achieved validation accuracy of 0.9819 at the final epoch.

### Model Testing

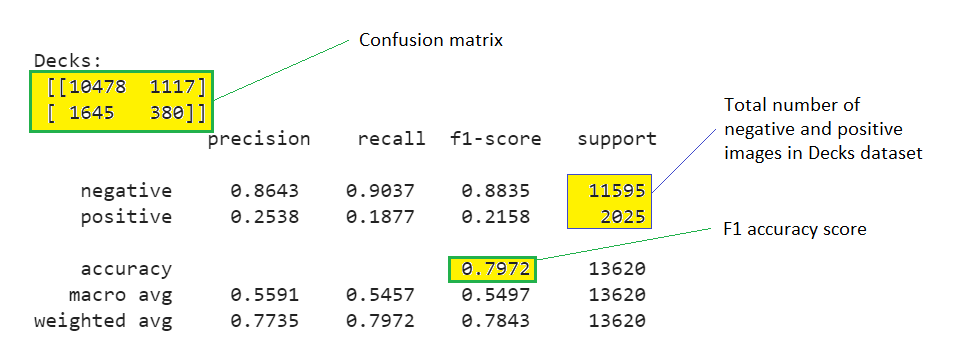
Out of sample performance and accuracy were determined using images from the SDNET dataset. These images represent concrete-like surfaces (decks, pavements, walls). By using this method, we can see how our model performs on images that it did not use for training. This approach gives us a better understanding of the model performance and is more tuned to how the model would perform in the real world.

The Confusion matrix was created for each assessment showing the number of correctly and incorrectly identified images.

|  |  |  |
| --- | --- | --- |
| Confusion matrix | Predicted (classified) as negative | Predicted (classified) as positive |
| Actual Negative images | Correct classification (true negative) | False positive |
| Actual Positive images | False negative | Correct classification (true positive) |

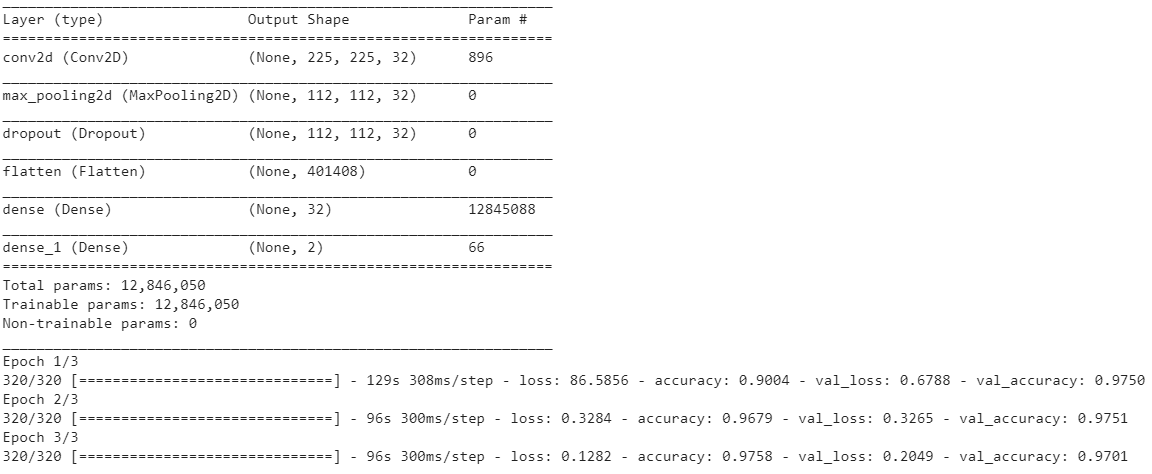
In addition to the confusion matrix, a classification report showing precision, recall rates, F1 score and accuracy was also generated. The F1 accuracy score was given the most weight as it demonstrates the overall ability of the model to correctly classify both negative and positive images. Higher accuracy values correspond to a better model.

Confusion matrix with sections highlighted for interpretation.

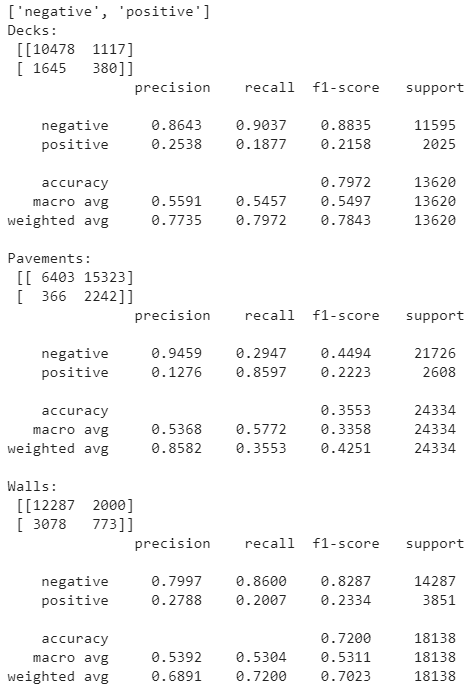


### Convolution Neural Networks (CNN) 1

Our first CNN model includes a 3x3 convolution layer with 32 nodes, 2x2 Max Pooling, 20% Dropout, Flatten layer, Dense layer with 32 nodes and RELU activation. This model has a total of 12,846,050 trainable parameters.

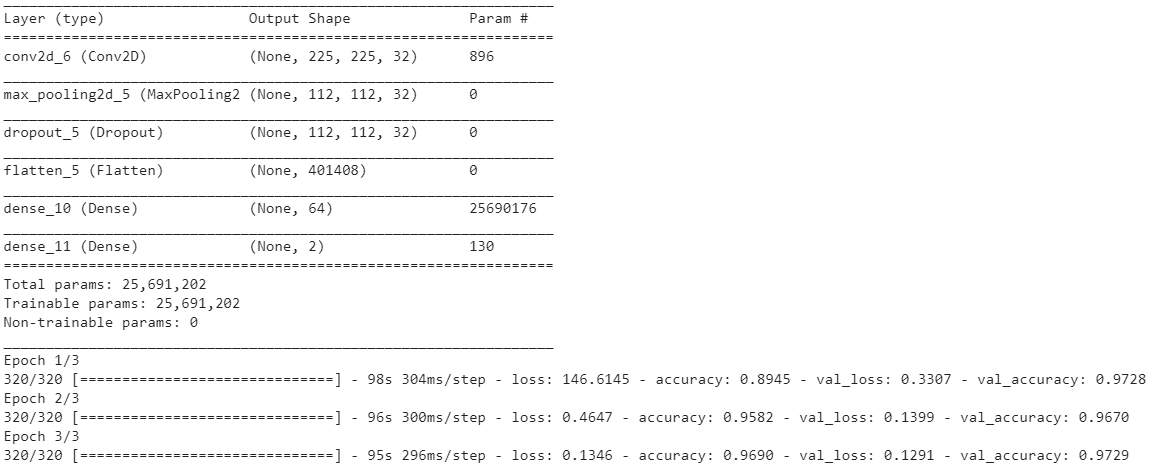


CNN1 model achieved 0.9701 validation accuracy, however, the model demonstrated only average performance on the SDNET test dataset and was particularly poor with pavement images. Pavement is more granular and has a beige/light brown color, which could explain the poor performance.

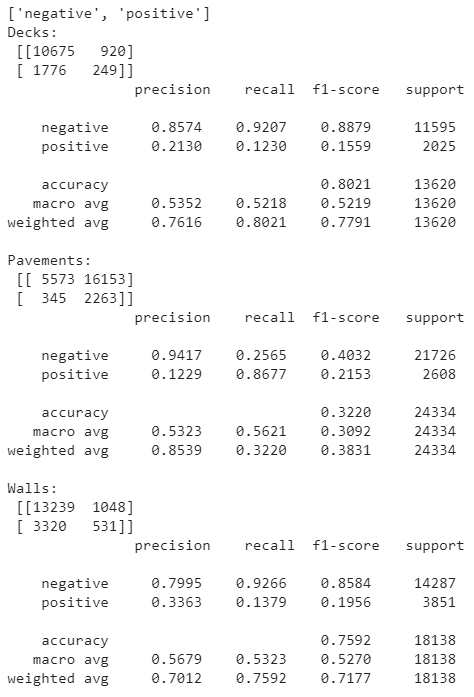


### Convolution Neural Networks (CNN) 2

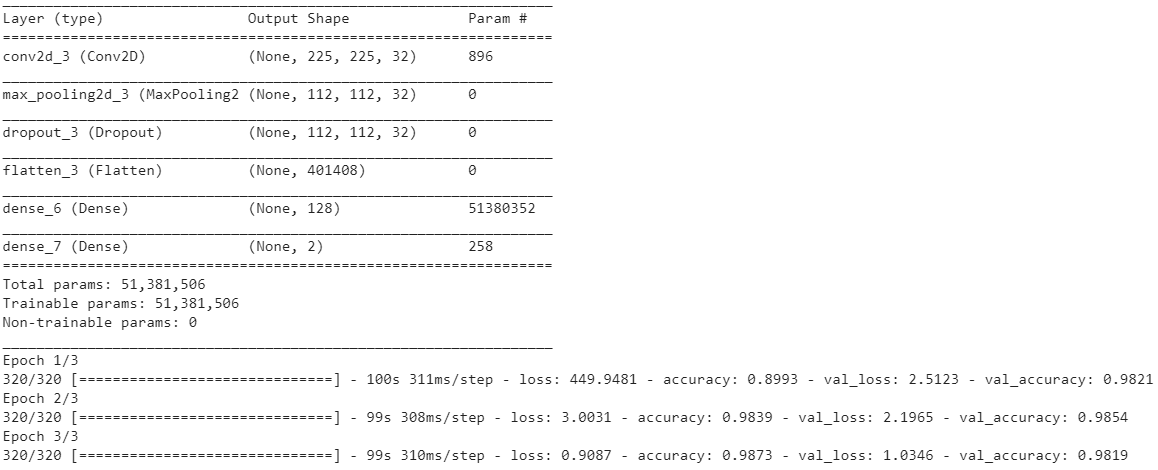
This model was built with a 3x3 convolution layer with 32 nodes, 2x2 Max Pooling, 20% Dropout, Flatten layer, Dense layer with 64 nodes and RELU activation. The main difference between the CNN1 and CNN2 models is CNN2 had double the number of nodes at the layer before the softmax function. This model has a total of 25,691,202 trainable parameters.

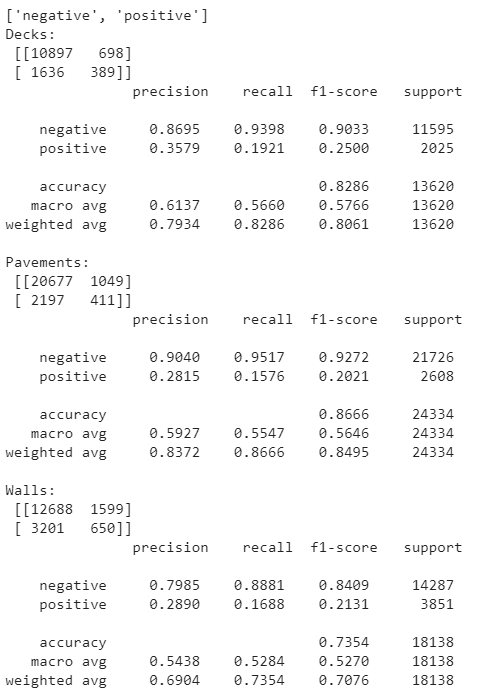
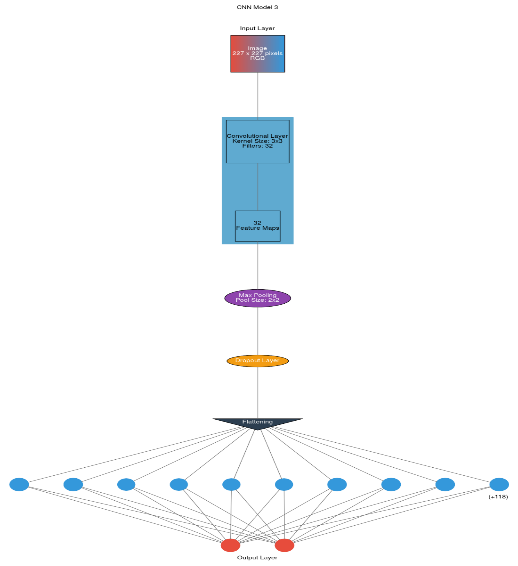


CNN2 achieved 0.9729 validation accuracy, slightly better than the CNN1 model. This model also had an average performance on the SDNET dataset, with poor accuracy on pavement images.



### Convolution Neural Networks (CNN) 3

This model is an extension of the CNN1 and CNN2 sequence and was also built with 3x3 convolution layer with 32 nodes, 2x2 Max Pooling, 20% Dropout, Flatten layer, Dense layer with 128 nodes and RELU activation but had twice as many nodes in the RELU layer compared to CNN2 and consequently the greatest number of trainable parameters (51,381,506). The model architecture can be seen below.

The CNN3 model achieved high validation accuracy (0.9819) and better performance than both the CNN1 and CNN2 models. More importantly, even on the more complex SDNET pavement dataset, we achieved an accuracy of 0.8666. This level of accuracy was not achieved in any other CNN model.

### Convolution Neural Networks (CNN) 4 & 5

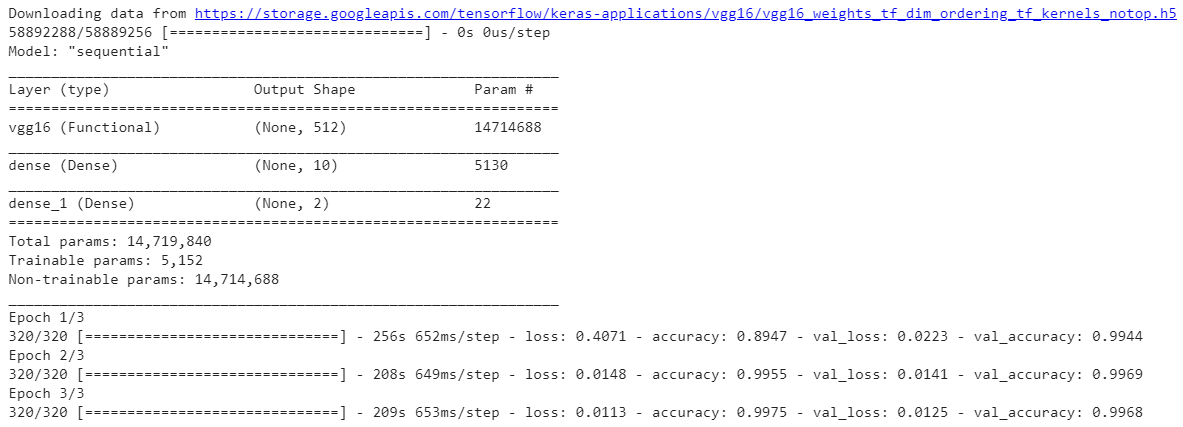
CNN4 is similar to CNN3 but includes a second convolution layer right after the first layer. The overall model architecture includes a 3x3 convolution layer 1 with 32 nodes, 3x3 convolution layer 2 with 32 nodes, 2x2 Max Pooling, 20% Dropout, Flatten layer, Dense layer with 128 nodes and RELU activation. As expected, CNN4 model achieved slightly better validation accuracy (0.9856) compared to CNN3. However, when tested on SDNET images, the performance was very poor on pavement images (0.4907 accuracy score).

For CNN 5, two batch normalizations were added, one after the convolution and one after dense layers with the following architecture: 3x3 convolution layer 1 with 32 nodes, Batch Normalization, 2x2 Max Pooling, 20% Dropout, Flatten layer, Dense layer with 128 nodes and RELU activation, Batch Normalization. Although the model demonstrated the highest validation accuracy among all CNN models at the time of training (0.9865), the testing results were average and not better than CNN1, CNN2 or CNN3.

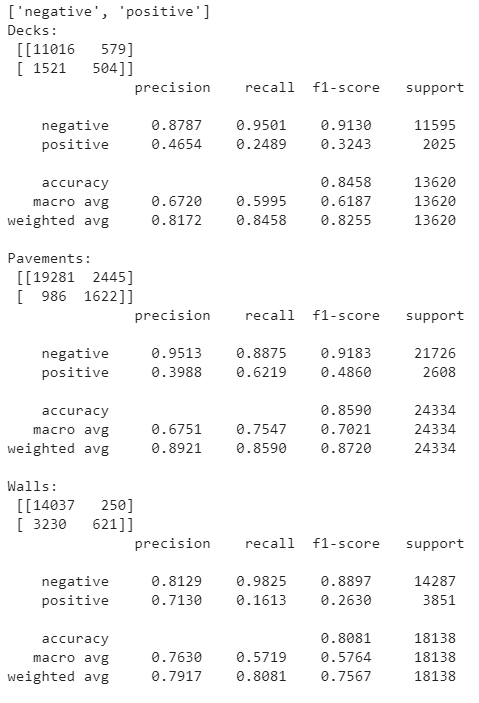
#### CNN Summary and Hyperparameter Tuning

To summarize, five CNN models were built for this project. They differed in the number of convolution layers, use of batch normalization and number of neurons in the dense layer. We performed hyper-parameter tuning with our five CNN models to find the one with highest accuracy.

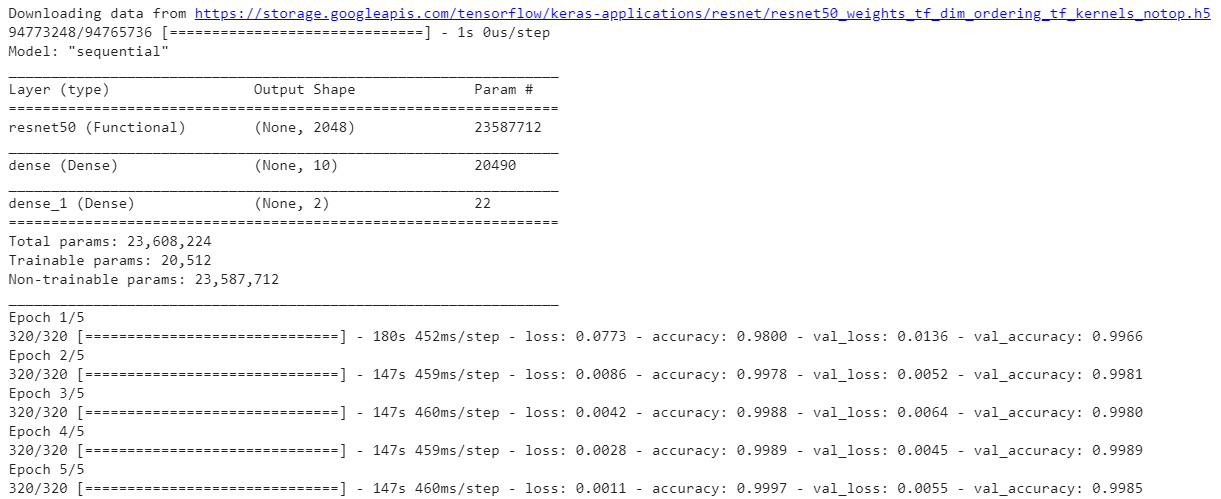
### VGG16 Pretrained Model

The VGG16 Pretrained model includes a pre-trained layer from keras.applications.vgg16 with MAX pooling and weights obtained from IMAGENET training. Output of the pre-trained layer is processed by a trainable Dense layer with 10 nodes and RELU activation, and a final Dense layer with softmax activation. Only the second and third Dense layers of the model are trainable resulting in 5,152 trainable parameters, while the functional vgg16 layer has 14,719,840 parameters that were pre-trained.

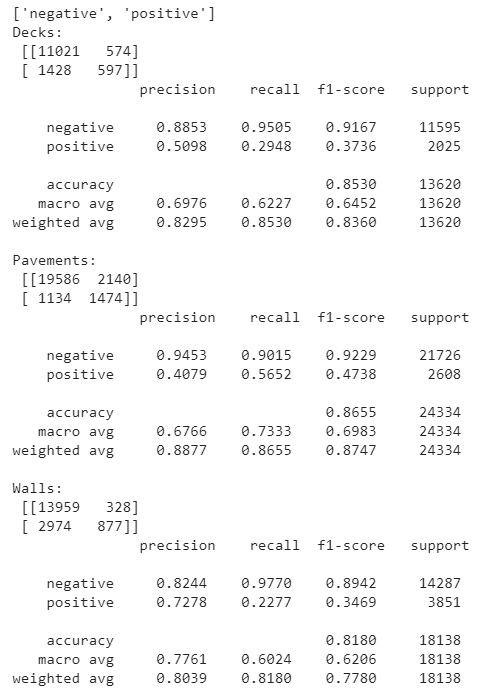
The VGG16 model had excellent validation accuracy (0.9968) and very good performance on the SDNET images. Accuracy on pavement images was 0.8590, one of the highest among all constructed models.



### Resnet50 pre-trained model

The Resnet50 pre-trained model includes a pre-trained layer from keras.applications.resnet50 with MAX pooling and weights obtained from IMAGENET training. Output of the pre-trained layer is processed by a trainable Dense layer with 10 nodes and RELU activation, and a final Dense layer with softmax activation. Only the second and third Dense layers of the model are trainable resulting in 20,512 trainable parameters, while the functional resnet50 layer has 23,608,224 parameters that were pre-trained.

ResNet50 model shows the highest validation accuracy (0.9985) and the best performance on the SDNET images. In all three SDNET categories, the ResNet50 model showed the highest accuracy making this model the best among pre-trained models.



### Other Pre-Trained Models

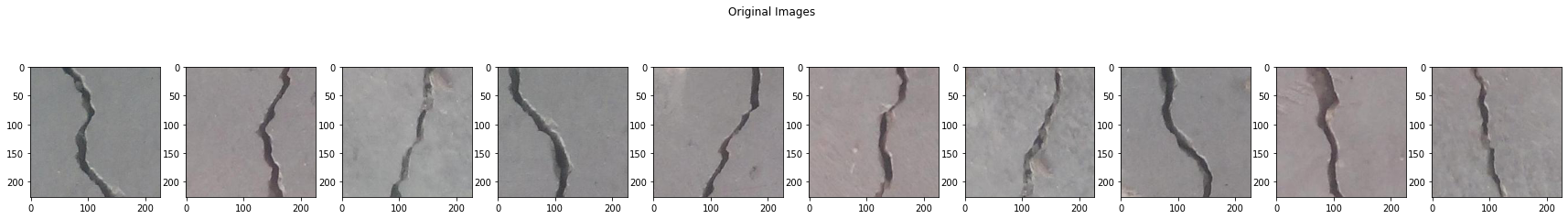
We also tried modelling using Xception, InceptionV3, NASNetLarge and DenseNet. These pre-trained models did not perform as well as ResNet50. Results of each of these models and description of results can be seen in Appendix 4. One interesting note is that while NASNetLarge is the most complex among all pre-trained models with 84,957,170 parameters, it performed the worst.

## Augmentation of the Dataset

One potential challenge with ML deep-learning models is insufficient training data. Although 40,000 images was initially thought to be a sufficiently large dataset, we found additional data was needed to help the model better generalize. We attempted to double the number of training images using data augmentation. Three augmentation methods were used: 90-degree rotation, horizontal flip, and vertical flip. After the training set was doubled with augmented images, our two best-performing models were re-trained and re-tested on the test SDNET images. The new training set included the original images plus the augmented set, 63,996 images total.

#### 90 Degree Rotation

This augmentation rotates each image 90-degree counter clockwise.





#### Horizontal Flip

This augmentation creates a mirror version of the image with “mirror” being positioned horizontally.

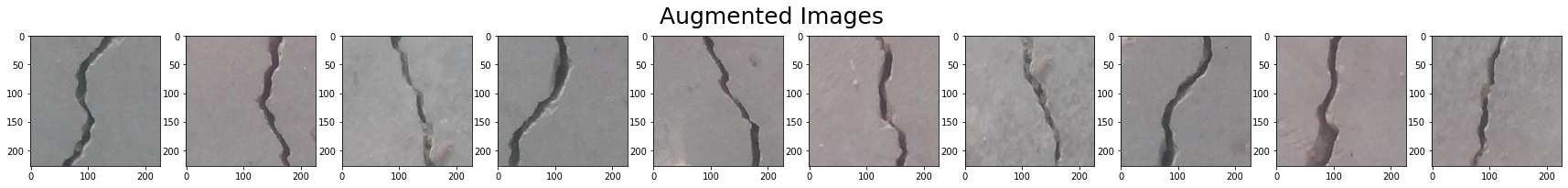




#### Vertical Flip

This augmentation creates a mirror version of the image with “mirror” being positioned vertically.



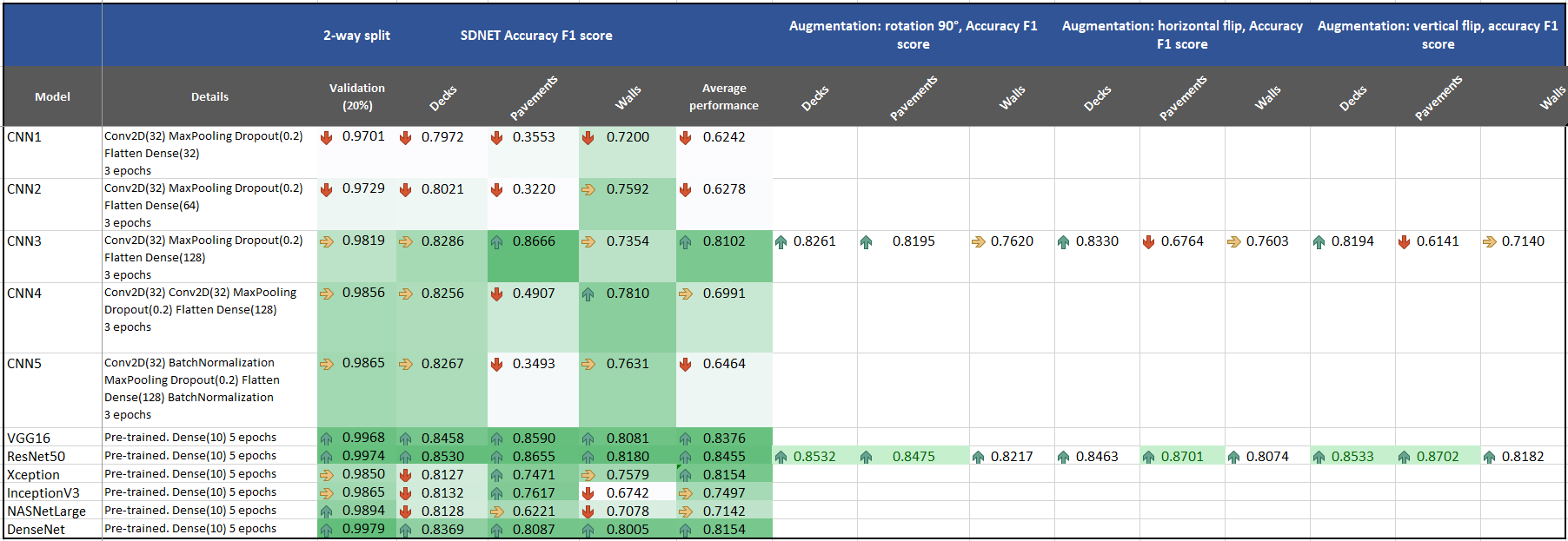


The ResNet50 model once again demonstrated superior performance after re-training using vertical flip augmented images. An average performance of 0.8472 was obtained, which is the best level of accuracy among all models. This suggests augmentation might be helpful to obtain good performance for the ResNet50 model.

Other data augmentation techniques such as horizontal flip and 90-degree rotation did not result in favorable results, in some cases our accuracy remained consistent and in others, it deteriorated.

## Summary of All Models and Key Conclusions

The heat-map below shows performance of different models on SDNET images as well as the results of our augmentations.

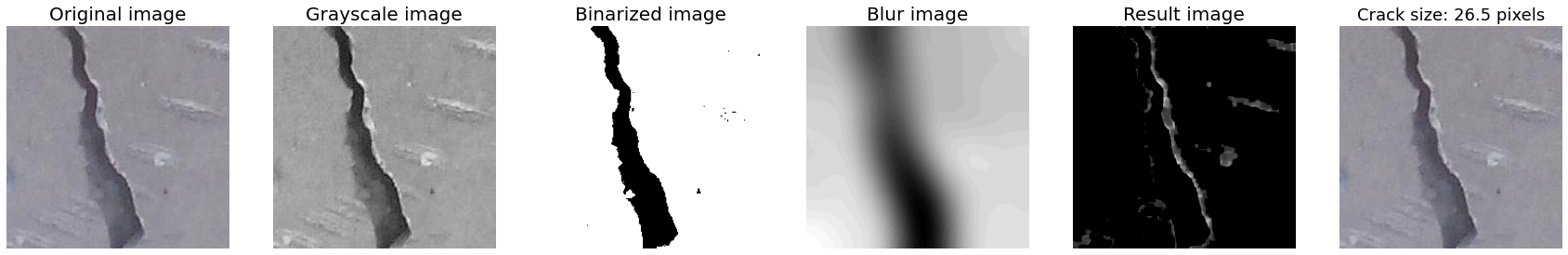


We found the ResNet50 model to have the best performance in comparison to all other models. The second most accurate model was VGG16, also a pre-trained model. ResNet50 showed superior performance when re-trained using augmented images, with the absolute best performance obtained after re-training on vertical flip augmented images.

The SDNET dataset appears to have some mislabeled images, especially within the pavement set. The pavement images have a slightly different color scheme compared to decks and walls and are substantially different from the 40,000 training images.

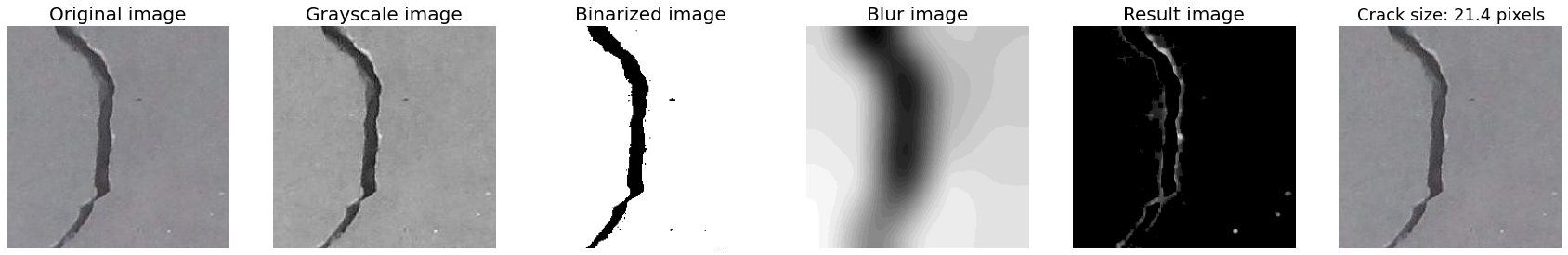
## Determining the Size of the Crack

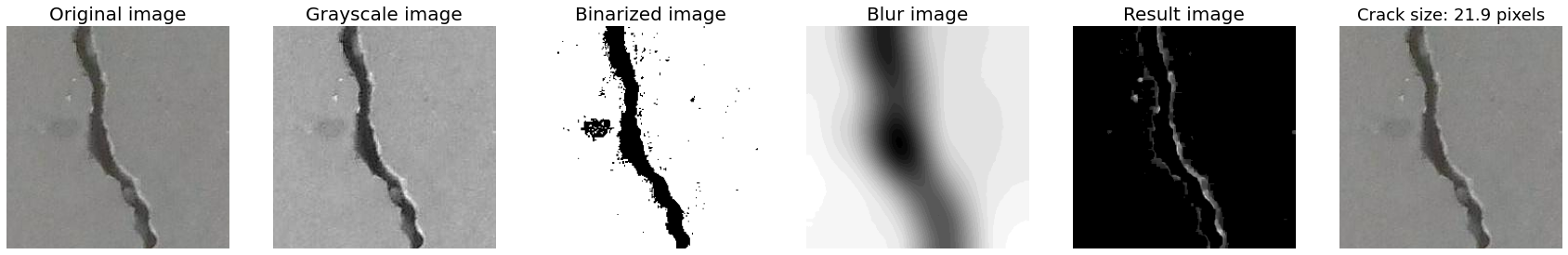
It is additionally valuable to determine the size of the crack as a measure of severity and urgency. A series of image transformations was used to calculate the size of the crack. These transformations are shown in the sequence below.

To compute this, the original image was converted to grayscale and then binarized. The transformations were made using cvtColor and threshold functions from the cv2 library. The image was then blurred using the GaussianBlur function (blur image) and detracted from gray image. The resulting image was processed to apply the Sobel operator (combining Gaussian smoothing and differentiation), calculating hypotenuse, normalizing the image, and applying the morphologyEx function. This produces the image shown as the result above.

As a final step, Euclidean distances was calculated, and mean distance was obtained. Mean distance when multiplied by image rows (227) is the estimated average width of the crack. It is displayed on the final image in pixels. Knowing the actual size of the image (say 10 inches), the width of 26.5 pixels can be converted into inches as (10” × 26.5/227 = 1.17”).

Examples below show cracks of different sizes as calculated by the described algorithm.





Note: this is a proof of concept to calculate the crack size. To obtain the true size of the crack, more investigation would be required to ensure the images are taken at the same height, same distance, and the same aspect ratio.

# Next Steps

The model performance for our SDNET test images did not reach the desired level of 95%, which is generally considered a requirement for obtaining funding and implementation. (8 Top-funded Facial Recognition Startups, 2019) This could be due to the quality of SDNET images, their variation from concrete\_data images, or issues with SDNET images being mislabeled. Further assessment of the model is needed using real industry production images. These images can be used to re-train the model to achieve optimal performance. Additional business value would be achieved by adding recommendation logic to prioritize crack repair based on urgency determined by comparing time series images of the crack location to predict both the path and the rate of crack growth. This data would then be used to assess and prioritize crack repairs based on input collected from subject matter experts. Furthermore, we would also like to add video to our dataset. Video captured by drones or dashcams can add to our model and possibly increase accuracy. Finally, we would also want to test out our model on potholes which opens new business avenues for applications of our model.

## Implementation

To implement our model in industry, we recommend using a mobile phone, security camera or a drone to take pictures or video of the site. To automate the process as much as possible, a wireless link should transmit data to back-end servers in real-time using compression techniques to reduce bandwidth. If a mobile phone is preferred, a mobile app can be created with an option to store images locally for later transfer to the server, or immediate delivery by calling web services to send the data.

We will have to consider how our training data is stored and catalogued. Options such as on-premises or cloud storage can be considered. Given the data is not sensitive, we recommend using a cloud storage solution such as Microsoft Azure, Amazon AWS, or Google Cloud. Cloud storage provides an integrated ecosystem where data can be processed using custom built scripts to extract meta-data, structure data into a relational database, and run the AI model using GPU or TPU for increased compute power. Cloud computing also provides scalability of storage and compute without having to manage on-premises infrastructure.

Environmental conditions should be factored into model training and tuning to prevent environmental bias. Some factors to consider are if people, animals, insects or objects frequently obstruct the surface, if weather affects image clarity, the lighting conditions in which images collected, the impact of shadows visible during certain times of the day, and the different surface types visible.

Each use-case for the model should be fully understood for optimal configuration by defining the acceptable accuracy thresholds, the compute power requirements, and the expected frequency of running the model. We will need to investigate if the model provides real-time recommendations or if a batch job is sufficient to retrieve results. Depending on a client’s requirement, both can be considered with a higher price for real-time retrievals. Finally, we will also need to continue improving upon our model and iterate on it based on the performance of the model in production.

Implementing this model in a business environment will allow companies to be more efficient in concrete crack detection. Inspections can be conducted at greater frequency but at less cost. This can lead to earlier detection of cracks, reducing repair costs and preventing more costly repairs in the future. Additionally, it may help avoid irreparable structural damage. When routine inspections of concrete surfaces are used for infrastructure objects like bridges, tunnels, highways, and tarmacs, active monitoring can prevent accidents and save lives.

# Conclusion

Having a high-accuracy ML-based algorithm as a low-cost preventative maintenance tool will have significant benefits, especially when paired with drone technology. Our solution can be implemented by construction companies, engineers, and city infrastructure surveyors to examine industrial and road concrete to find cracks. Pairing our solution with drones will increase monitoring in hard-to-reach places and situations where manual inspections are not feasible. Phone-based applications can also be created to increase the use among less-trained inspectors. Additionally, it presents opportunities for use in developing countries, where construction requirements are lax, labour expertise is limited, and civil construction failures are frequent. (Fernandez R. H., 2018) (Ker Than, 2013)

# APPENDIX

1. GITHUB: <https://github.com/taha-shafique/mmai894>
2. Both datasets are hosted on a public website (created specifically for this project): <https://www.josephambrose.com/concrete_data.zip>,

<https://www.josephambrose.com/SDNET.zip>

1. Figure 1 – CNN 3 Architecture

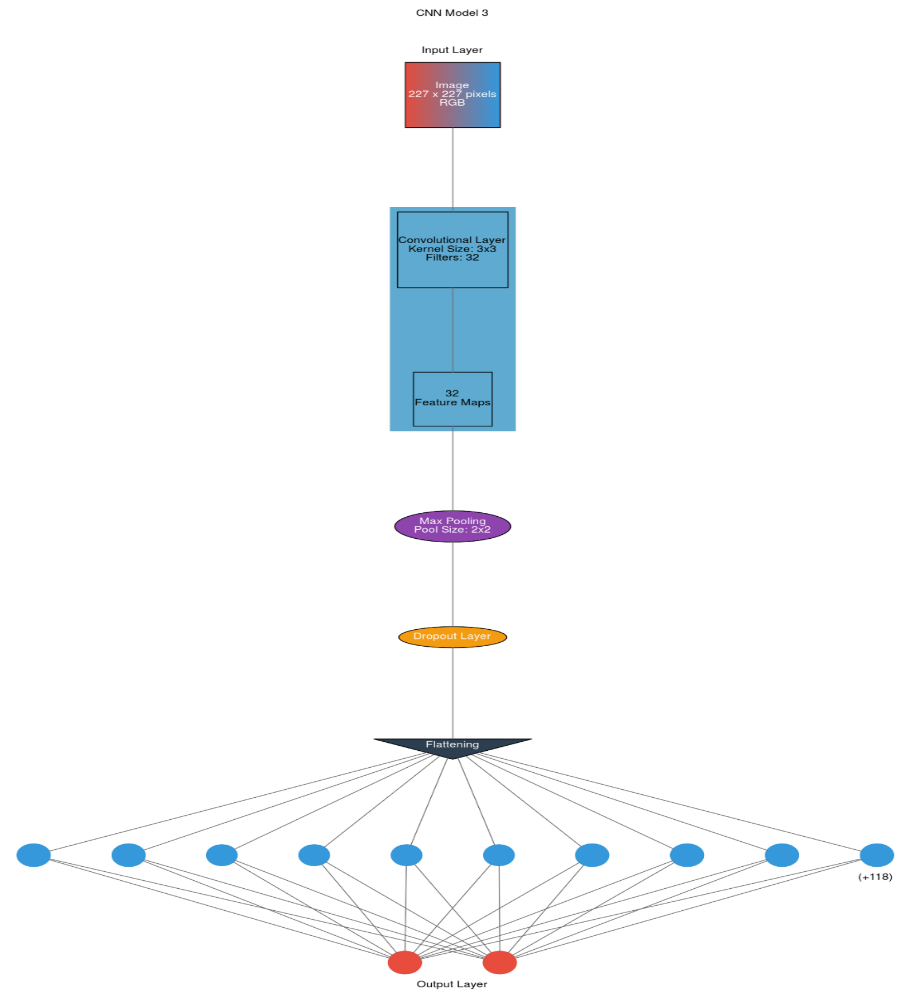
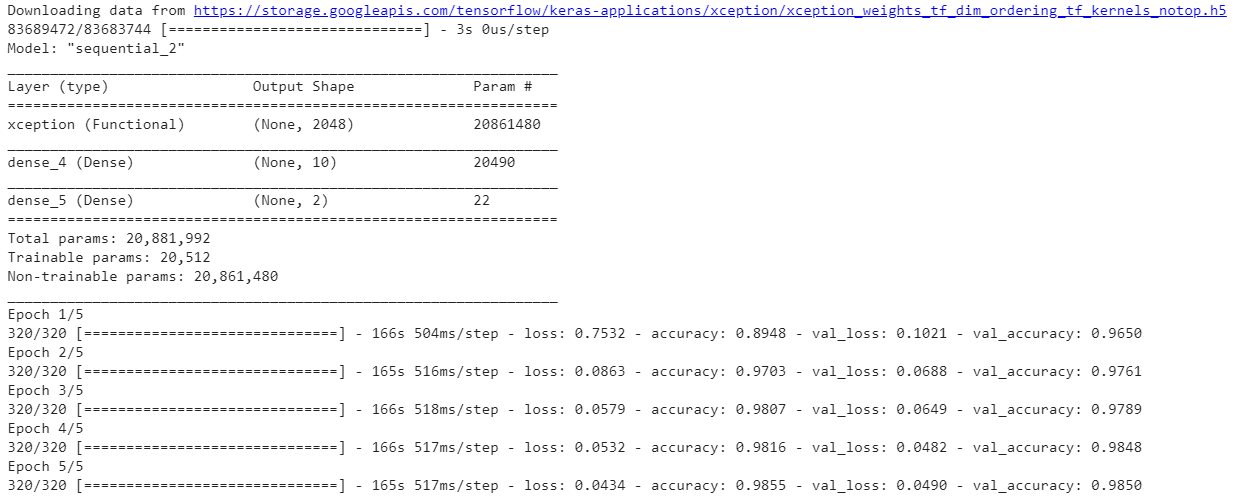


Figure 1 - CNN 3 Architecture visualized using ANN visualizer.

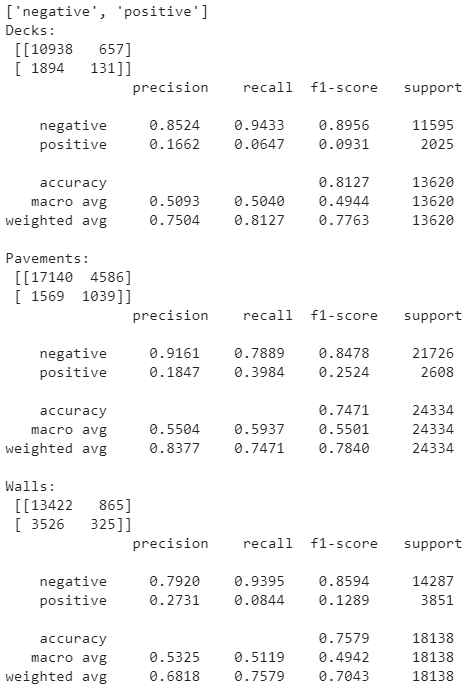
1. Other Pre-Trained Modelling results

XCEPTION

This model includes pre-trained layer from keras.applications.xception with MAX pooling and weights obtained from IMAGENET training. Output of the pre-trained layer, is processed by trainable Dense layer with 10 nodes and RELU activation, and then final Dense layer with softmax activation. Only second and third Dense layers of the model are trainable resulting in 20512 trainable parameters, while the functional Xception layer has 20881992 parameters that were pre-trained.

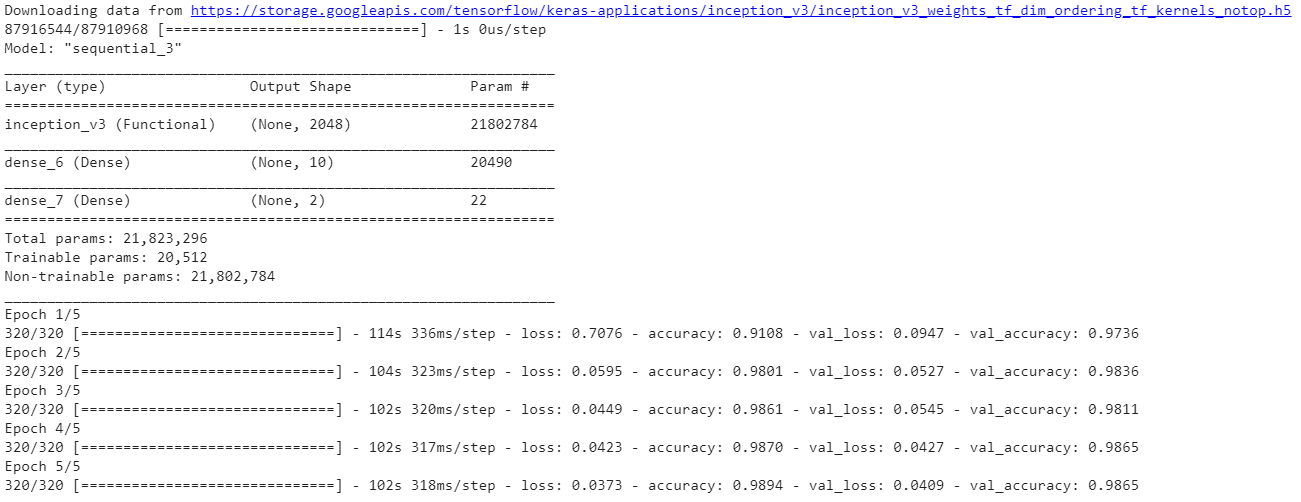


Although the model demonstrated good validation accuracy (0.9850) the performance on SDNET images was average. Pavements and walls images had accuracy of about 0.75 with this model.

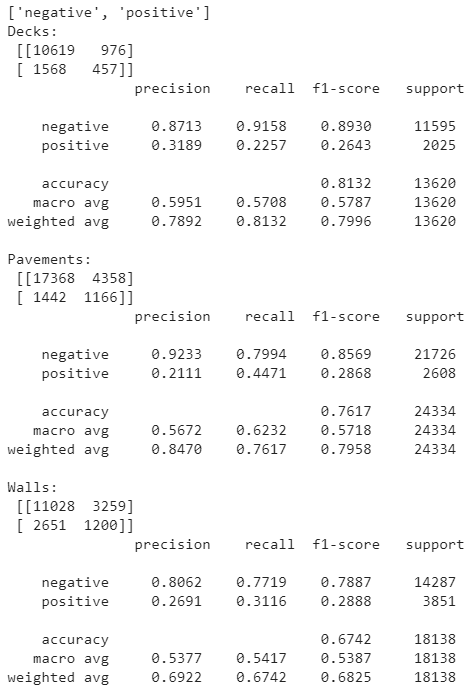


InceptionV3

This model includes pre-trained layer from keras.applications.InceptionV3 with MAX pooling and weights obtained from IMAGENET training. Output of the pre-trained layer, is processes by trainable Dense layer with 10 nodes and RELU activation, and then final Dense layer with softmax activation. Only second and third Dense layers of the model are trainable resulting in 20512 trainable parameters, while the functional Inception\_v3 layer has 21802784 parameters that were pre-trained.

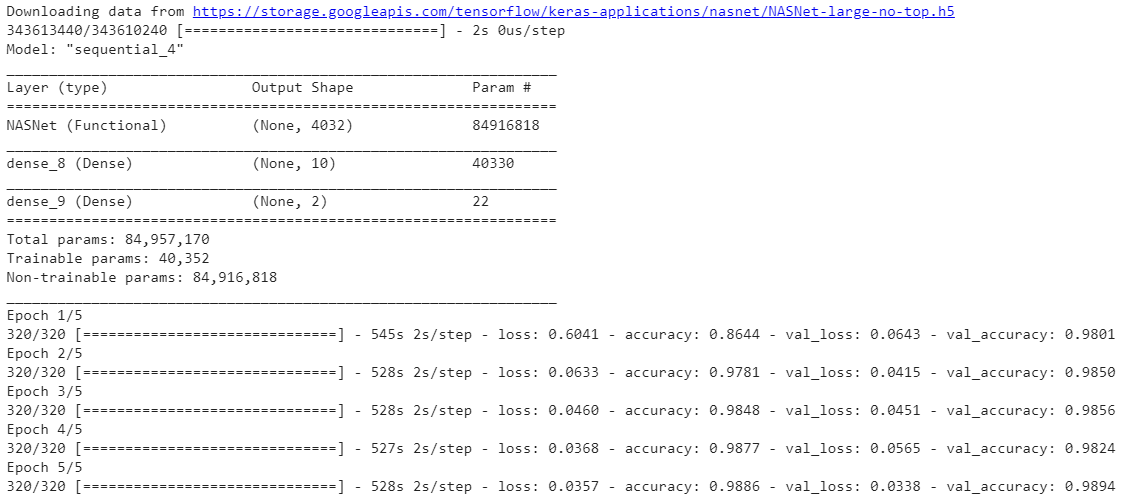


Model has good validation accuracy (0.9865) comparable to Xception model, but also suffered from poor performance on SDNET dataset, in particular walls (performance on walls images was 0.6742).

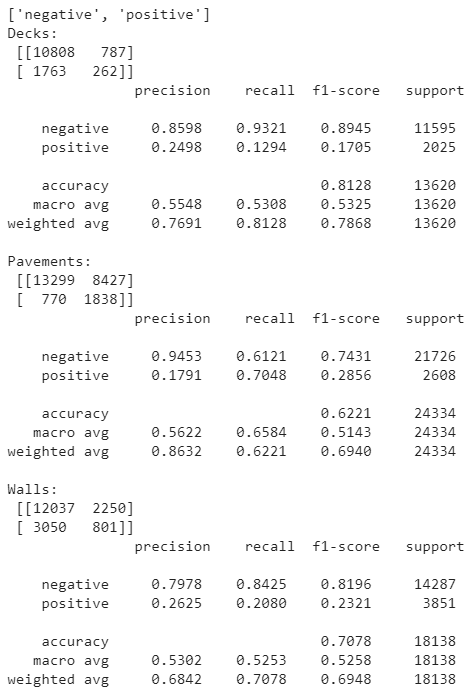


NASNetLarge

This model includes pre-trained layer from keras.applications.NASNetLarge with MAX pooling and weights obtained from IMAGENET training. Output of the pre-trained layer, is processed by trainable Dense layer with 10 nodes and RELU activation, and then final Dense layer with softmax activation. Only second and third Dense layers of the model are trainable resulting in 40352 trainable parameters. The functional NASNet layer is the most complex among all pre-trained models with 84957170 parameters that were pre-calculated.

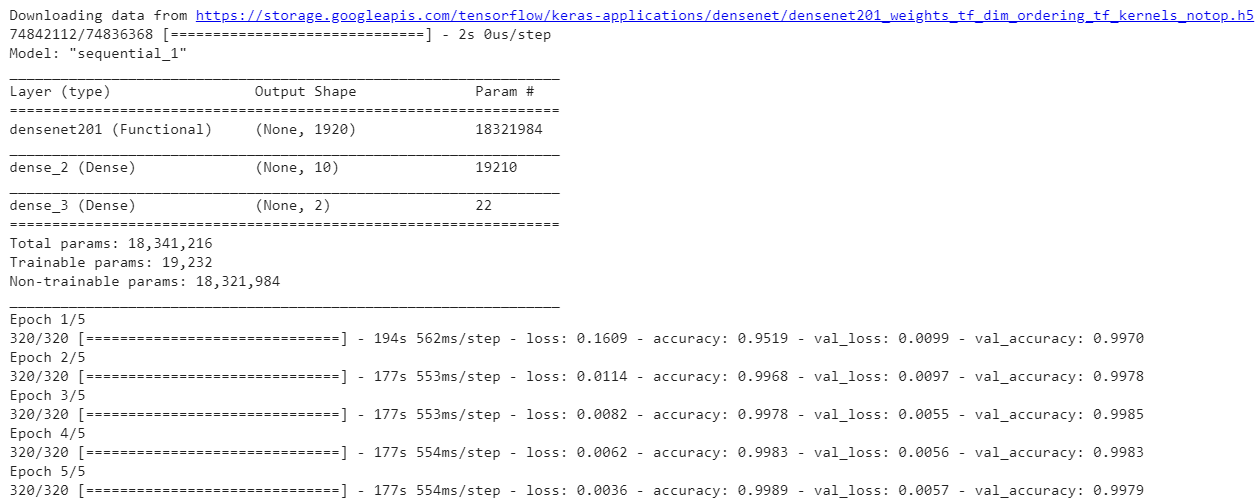


Despite being the most complex (largest) pre-trained model, NASNet model did not show good accuracy processing SDNET images. In particular, model had low accuracy on pavements and walls, the worst accuracy among all pre-trained models.

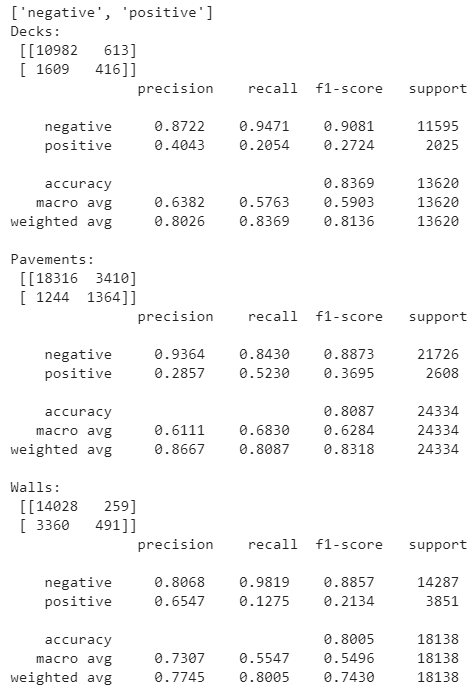


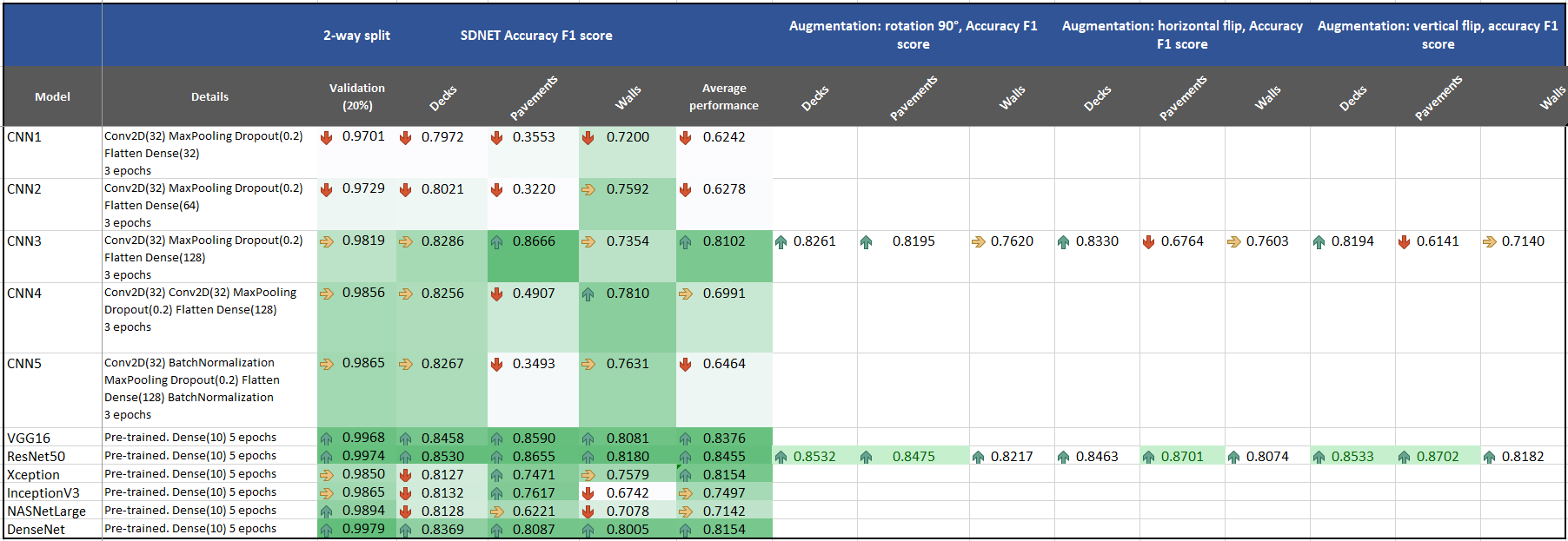
DenseNet

This model includes pre-trained layer from keras.applications.DenseNet201 with MAX pooling and weights obtained from IMAGENET training. Output of the pre-trained layer, is processed by trainable Dense layer with 10 nodes and RELU activation, and then final Dense layer with softmax activation. Only second and third Dense layers of the model are trainable resulting in 19232 trainable parameters, while the functional densenet201 layer has 18341216 pre-determined parameters.



Training the model achieved high validation accuracy (0.9979), however the model was average in ability to detect cracks on SDNET images. Its overall performance was similar to Xception model.



1. Table of All Models with Accuracy F1 Score
2. Group Participation

All members contributed in the development of the code and paper, but were put in designated teams. <https://github.com/taha-shafique/mmai894>

|  |  |
| --- | --- |
| Contribution | Group Member |
| First Lead | Taha Shafique |
| Second Lead | Anuj Singh |
| Modelling Team | Joseph Ambrose  Thomas Ludwig |
| Research Team | Jichen Song  Hamid Baz Mohammadi |
| Paper Team | Kelly McConvey  Selina Wang |

# Resources Used

1. Deep Learning (MIT Press) - [Ian Goodfellow](https://play.google.com/store/books/author?id=Ian+Goodfellow), [Yoshua Bengio](https://play.google.com/store/books/author?id=Yoshua+Bengio" \t "_blank), [Aaron Courville](https://play.google.com/store/books/author?id=Aaron+Courville)
2. Deep Learning with Python - by Francois Cholle
3. The Elements of Statistical Learning - Jerome H. Friedman, Robert Tibshirani, and Trevor Hastie
4. Deep Learning class lecture notes - Prof.Ofer Shai
5. <https://keras.io/>
6. <https://opencv.org/>
7. <https://en.wikipedia.org/wiki/Sobel_operator>
8. <https://en.wikipedia.org/wiki/Gaussian_blur>
9. www.stackoverflow.com
10. www.scikit-learn.org

# References Used

1. Fernandez, Raul H. Figueroa (2018): Strategies to Reduce the Risk of Building Collapse in Developing Countries. Carnegie Mellon University. Thesis. https://doi.org/10.1184/R1/6723218.v1
2. Ker Than (2013): Bangladesh Building Collapse Due to Shoddy Construction

National Geographic News

[Bangladesh Building Collapse Due to Shoddy Construction (nationalgeographic.com)](https://www.nationalgeographic.com/history/article/130425-bangladesh-dhaka-building-collapse-world)

1. Özgenel, Çağlar Fırat (2019), “Concrete Crack Images for Classification”, Mendeley Data, V2, doi: 10.17632/5y9wdsg2zt.2
2. Maguire, M., Dorafshan, S., & Thomas, R. J. (2018). SDNET2018: A concrete crack image dataset for machine learning applications. Utah State University. <https://doi.org/10.15142/T3TD19>
3. Giatech Scientific Inc: Evaluating Cracking in Concrete Procedures

[Evaluating Cracking in Concrete: Procedures | Giatec Scientific Inc.](https://www.giatecscientific.com/education/cracking-in-concrete-procedures/)