Crack Detection: Convolutional Neural Network

Customized CNN and VGG16

Neal Xu

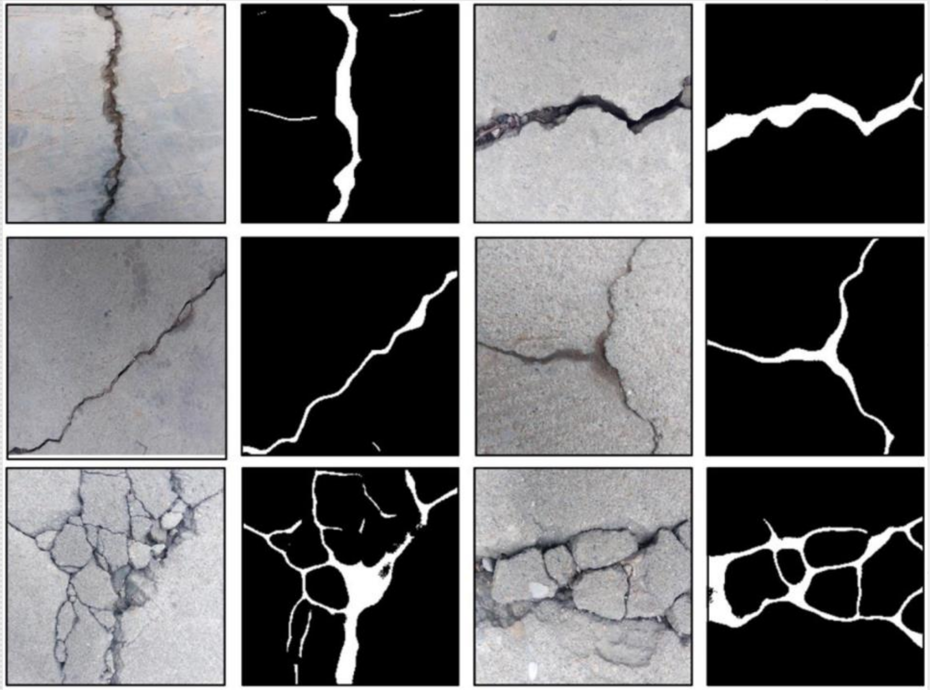
Problem Statement

- In civil structures, concrete or infrastructure surface cracks pose a significant threat as they undermine the rigidity and tensile strength of buildings.
- Building inspection is crucial for evaluating the health of a structure, and crack detection plays a pivotal role in this process.
- However, manual identification of cracks is time-consuming and prone to human error.

Project Goal

- There is a need for an automated system that can accurately detect and classify cracks in concrete surfaces, enabling efficient building inspections and facilitating the assessment of building health.

- In this project, the **GOAL** is to use deep learning image classification model to identify cracks in different types of material surfaces.



Assumptions & Hypotheses for CNN

- **Localized features**: CNNs excel at capturing local features due to their convolutional layers. It is assumed that surface cracks exhibit distinct patterns and textures that can be learned by the CNN's filters.
- **Spatial invariance**: CNNs are invariant to spatial transformations, meaning they can detect cracks regardless of their location, orientation, or scale on the surface. This hypothesis assumes that cracks exhibit similar patterns and textures regardless of their specific position on the surface.
- **Transferability**: CNNs trained on one type of surface (e.g., concrete) can generalize well to other similar surfaces (e.g., asphalt). This hypothesis assumes that cracks on different surfaces share common visual characteristics that can be learned by the network.

Dataset I

Surface photos for single material

- Cracks:
  - 20000 images with 227 x 227 pixels with RGB channels
- Non-Cracks:
  - 20000 images with 227 x 227 pixels with RGB channels
- These surface cracks image are retrieved from various civil structures.

Well **Balanced** Data for Positive and Negative cases

Dataset II

Surface photos for multiple materials

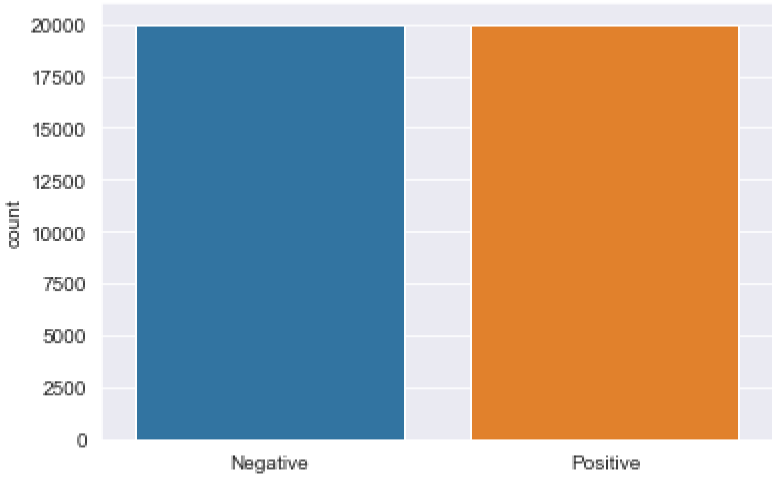
- Concrete Bridge Walls
  - 3851Cracked Images
  - 143000Non-Cracked Images
- Concrete Bridge Pavements
  - 2608 Cracked Images
  - 217000 Non-Cracked Images
- Concrete Bridge Decks
  - 2025 Cracked Images
  - 116000 Non-Cracked Images

Extremely **Imbalanced** and **Unclear** Data for Positive and Negative cases

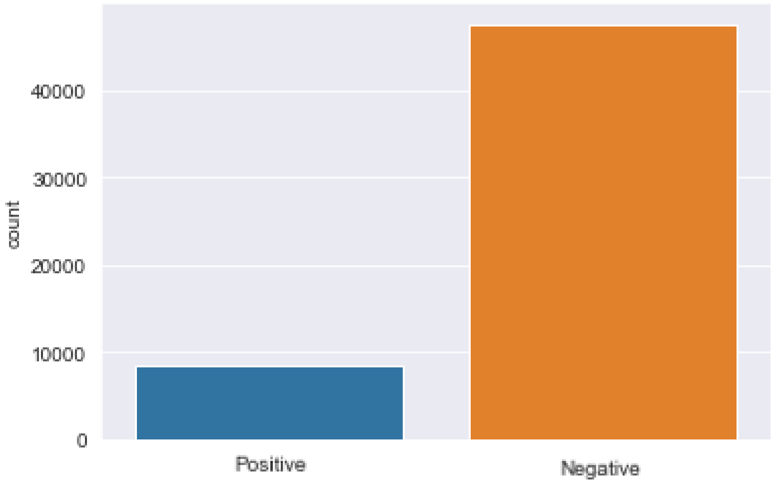
Exploratory Data Analysis

Dataset I: single material

Dataset II: multiple materials



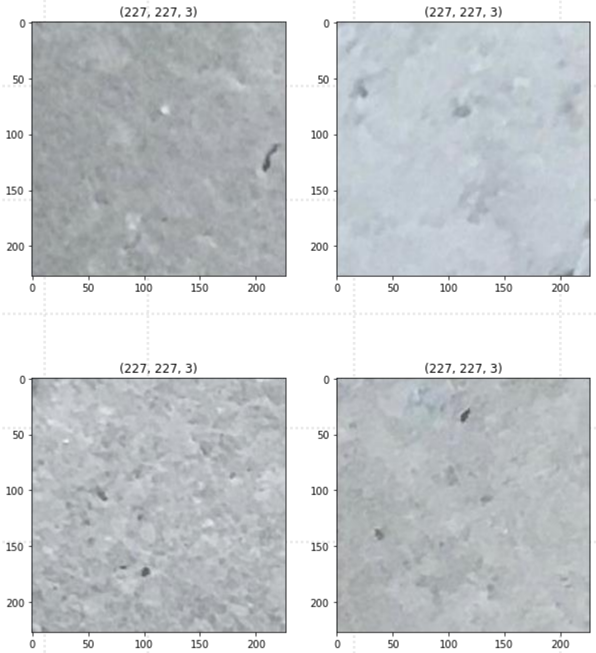
Dataset I (single material kind) has a balanced training and testing data of 20000 images for each category.



Dataset II (multiple material kind) has an extremely imbalanced training and testing data of 8000 images for cracked and 45000 for non-cracked.

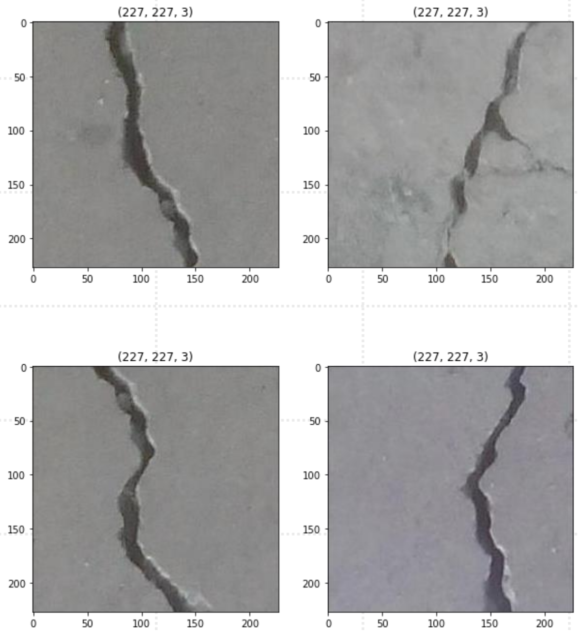
Data Example (Dataset I)

Non-Cracked



Clear image with no cracks and consistent images for non-cracked datasets.

Cracked

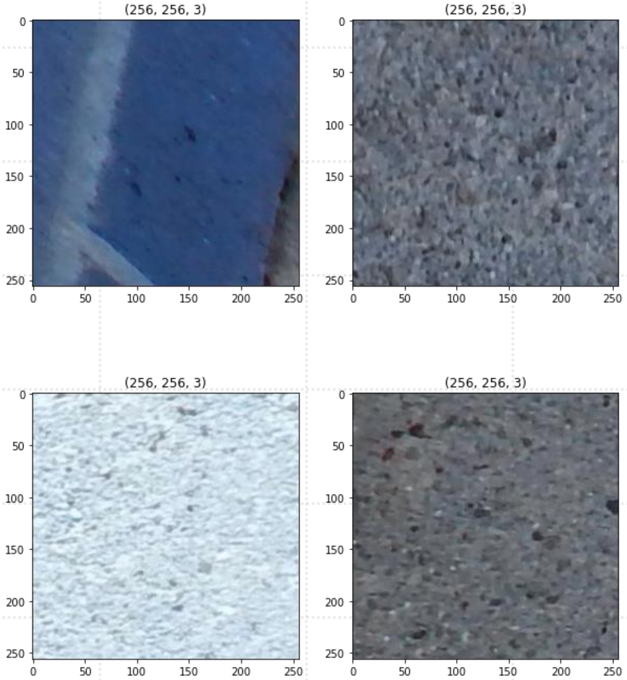


Clear image with clear cracks that is clear to be identified for cracked images.

Data Example (Dataset II)

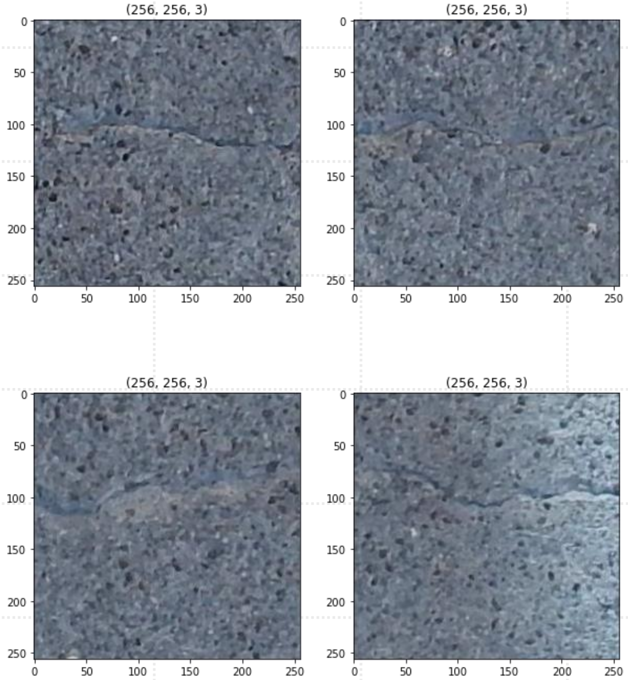
Uncleaned data with unclear cracks

Non-Cracked



Clear image with no cracks, BUT with inconsistent images for non-cracked datasets from different materials as well as with image capturing error.

Cracked (May Cause Problems)



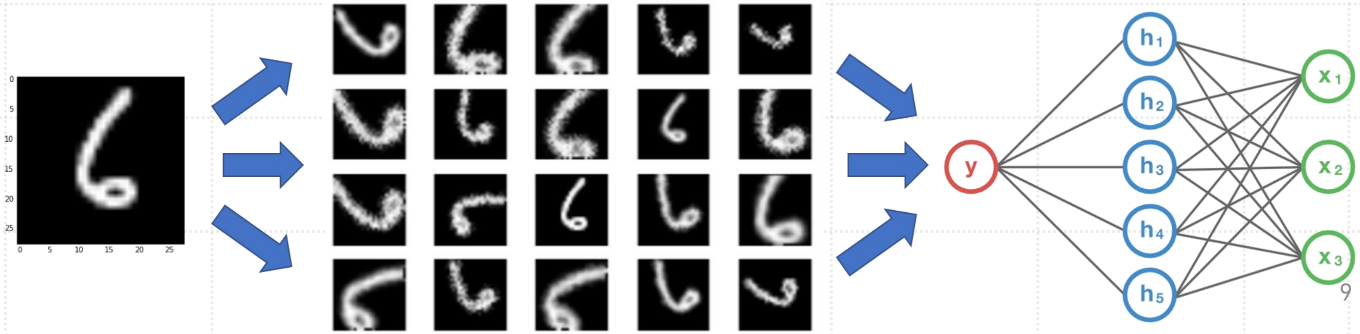
Clear image with cracks, BUT hard to be identified even with human eyes. Also, different materials have different crack types.

Feature Engineering: Data Augmentation (only need for Dataset II)

```
datagen = ImageDataGenerator(
        rotation_range=365,
        zoom_range = 0.2,
        width_shift_range=0.2,
        height_shift_range=0.2,
        horizontal_flip=True,
        vertical_flip=True)
```

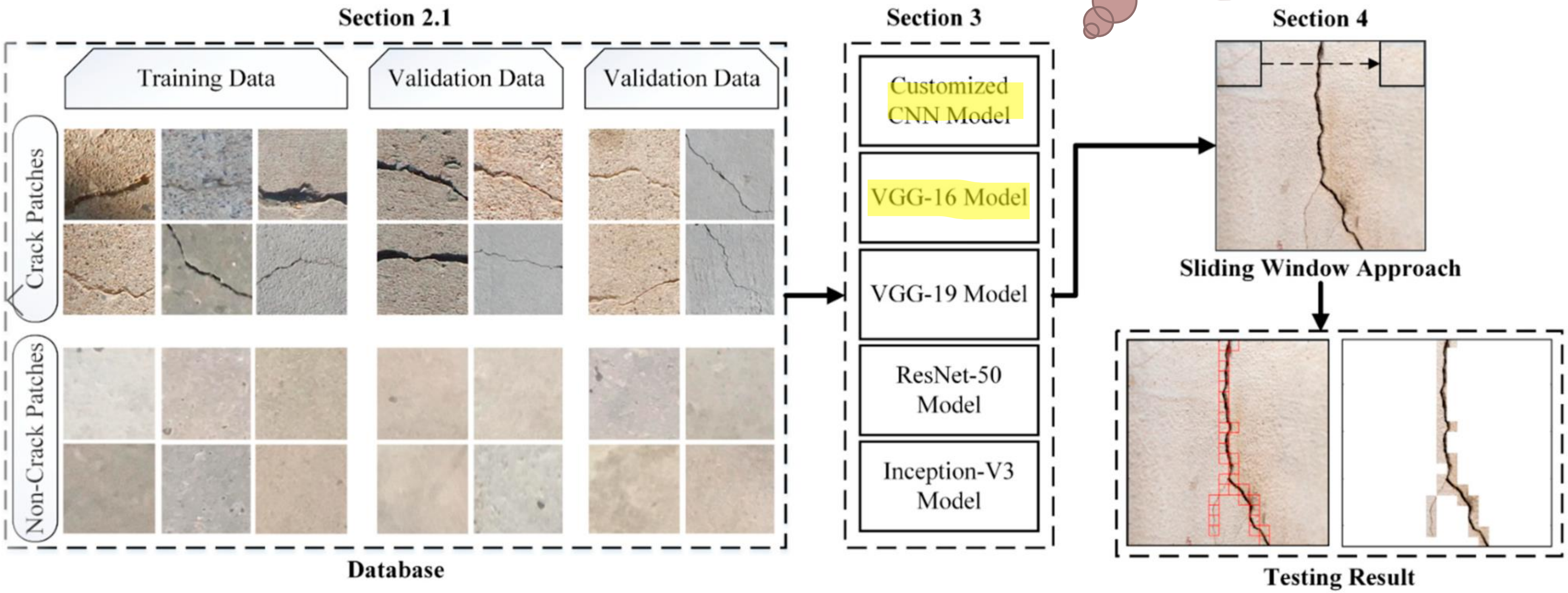
The code on the left-hand side would **increase the training dataset** by doing various operations shown below:

- Randomly **rotate** images in the range [-365, 365] degrees
- Randomly **zoom** images by a factor of up to 0.2
- Randomly **shift** images horizontally by up to 20% of the image width
- Randomly **shift** images vertically by up to 20% of the image height
- Randomly **flip** images horizontally
- Randomly **flip** images vertically



Model Selection

For this project, we choose Customized CNN model and the VGG-16 Model



Modeling (CNN)

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 64)      1792

 max_pooling2d (MaxPooling2D  (None, 75, 75, 64)       0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        36928

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 128)       73856

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 flatten (Flatten)           (None, 41472)             0

 dense (Dense)               (None, 256)               10617088

 dropout (Dropout)           (None, 256)               0

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 dense_1 (Dense)             (None, 2)                 514
```

On the left hand, is the screenshot of the CNN model used to train for the dataset I with **single-kind surface material ONLY**.

```
================================
Total params: 10,731,202
Trainable params: 10,730,690
Non-trainable params: 512
```

The customized CNN did a poor job with the Dataset II which has multiple surface materials and unclear images.

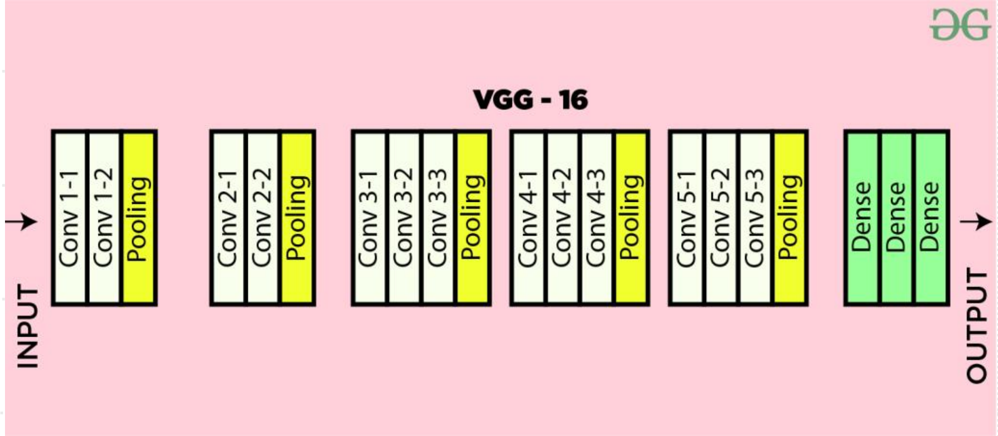
Here is an illustration of why the CNN on the left does not work:

Low Accuracy for both training and testing datasets.

```
Training Accuracy: 0.8496
Testing Accuracy: 0.8454
```

Modeling (Pre-Trained – VGG16)

As a solution, we apply a more sophisticated model for Dataset II:



Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150, 150, 3)]	0
block1_conv1 (Conv2D)	(None, 150, 150, 64)	1792
block1_conv2 (Conv2D)	(None, 150, 150, 64)	36928
block1_pool (MaxPooling2D)	(None, 75, 75, 64)	0
block2_conv1 (Conv2D)	(None, 75, 75, 128)	73856
block2_conv2 (Conv2D)	(None, 75, 75, 128)	147584
block2_pool (MaxPooling2D)	(None, 37, 37, 128)	0
block3_conv1 (Conv2D)	(None, 37, 37, 256)	295168
block3_conv2 (Conv2D)	(None, 37, 37, 256)	590080
block3_conv3 (Conv2D)	(None, 37, 37, 256)	590080
block3_pool (MaxPooling2D)	(None, 18, 18, 256)	0
block4_conv1 (Conv2D)	(None, 18, 18, 512)	1180160
block4_conv2 (Conv2D)	(None, 18, 18, 512)	2359808
block4_conv3 (Conv2D)	(None, 18, 18, 512)	2359808
block4_pool (MaxPooling2D)	(None, 9, 9, 512)	0
block5_conv1 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv2 (Conv2D)	(None, 9, 9, 512)	2359808
block5_conv3 (Conv2D)	(None, 9, 9, 512)	2359808
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	0

Total params: 14,714,688
Trainable params: 0
Non-trainable params: 14,714,688

Proposed Approaches (Model Comparison)

For Dataset I :

- CNN (Convolutional Neural Network)

For Dataset II:

- VGG16 (a pre-trained CNN architecture)

Aspect	**CNN**	**VGG16**
Architecture	Customized CNN architecture	Specific deep CNN architecture
Depth	7 layers	16 layers
Convolutional Filters	64, 64, 128	3x3 filters
Pooling Strategy	Max pooling (2x2 window, stride of 2)	Max pooling (2x2 window, stride of 2)
Model Capacity	Moderate capacity	High capacity due to depth
Transfer Learning	Limited to its own architecture	Commonly used for transfer learning
Computational Complexity	Moderate	Relatively high due to depth and parameters
Overfitting	Requires monitoring and regularization techniques	Requires monitoring and regularization techniques
Performance	Performance dependent on design and optimization	Performance influenced by pre-training and architecture
Implementation	Custom implementation based on requirements	Pre-trained model available for u

Proposed Solution (Model Selection)

- Based on the model performance result, CNN (Convolutional Neural Network) is chosen for detecting the cracks for **Dataset I**.
- It has several reasons:
  - High accuracy
  - High precision, recall, f-1 score
  - Fast training time
  - Grey scall data is enough

- Based on the model performance result, VGG16 (Very Deep Convolutional Networks for Large-Scale Image Recognition) is chosen for detecting cracks for **Dataset II**.
- VGG16 has a better ability due to its pre-trained nature. It has been pre-trained on large-scale image classification tasks, such as the ImageNet dataset. This pre-training allows the network to learn general features from a diverse set of images.

Checks for Overfitting/Underfitting

For CNN on Dataset I

There are **NO** signs showing overfitting or underfitting since both the testing and training dataset has extremely high scores.

Training Accuracy: 0.9978

Testing Accuracy: 0.9983

For testing dataset:

- **Precision**: 0.9982522146073458
- **Recall**: 0.9982482285407996
- **F1-score**: 0.998249971999552

For VGG16 on Dataset II

If only check the accuracy of the training and testing. There is also no sign of overfitting or underfitting as shown below:

Training Accuracy: 0.8496

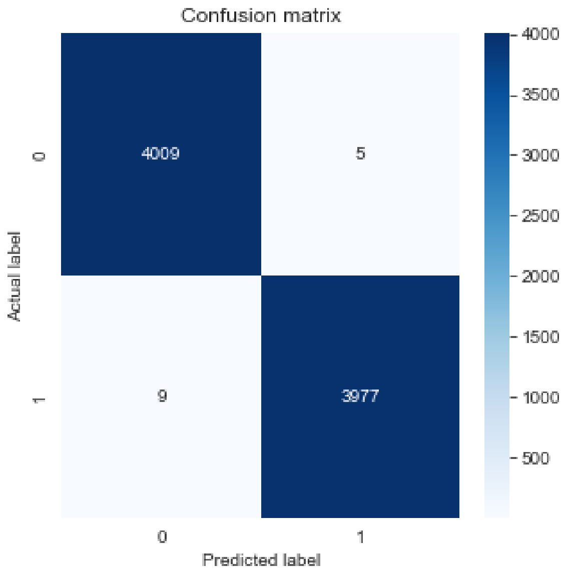
Testing Accuracy: 0.8454

However, the model has another problem it has a low capability of detecting cracks possibly due to the quality of Dataset II.

See next Slide

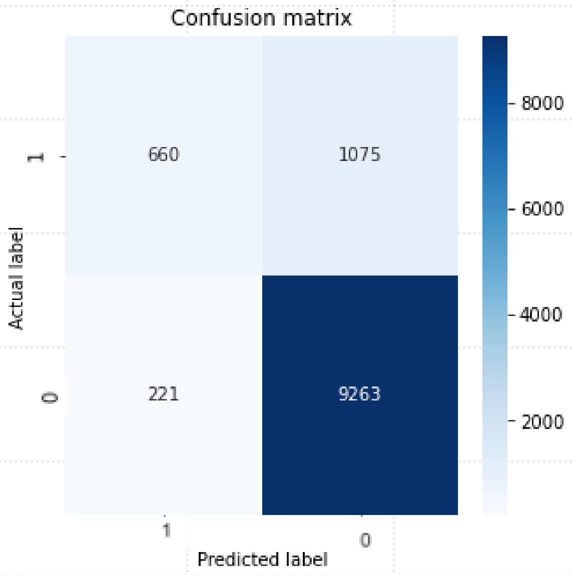
Results (Confusion Matrix)

Confusion matrix for single material (dataset I)



Perfect Confusion matrix showing that the CNN model has an extremely <u>high</u> capability of correctly identifying cracked and non-cracked images.

Confusion matrix for multiple materials (dataset II)



The VGG16 we applied to the Dataset II images with different surface materials shows that it has a <u>high</u> capability to identify non-crack surfaces but has a <u>low</u> ability to identify positive cases (Due to low positive image quality as shown on slide No.8).

Future Work

- For the CNN model applied to the single material surface cracks (Dataset I) detection, there is no need for improvement and the model is ready to use.
- However, the images from Dataset I is not a real representation of the actual situation that is free of the errors of capturing images including but not limited to light, vibration, camera, and so on.
- Dataset II is a better representation of the real-life application of the algorithm that has many image errors on various surface materials.
- Therefore, future work should focus o:
  - Improving training image pre-processing;
  - Increasing the amount and quality of training data (data augmentation is not sufficient here);
  - Fine-tuning the VGG16 to minor cracks detection to increase crack detection ability;
  - Apply more Hyperparameter Tuning, since the current hardware limitation, not enough hyperparameters are tuned resulting in no improvement of the model.