Deliverable 2: Concrete Crack Detection

Problem Statement

This project consists of crack detection of concrete surfaces. It is an image classification problem. Images are classified into positive (crack) and negative (no crack). Crack detection is important for several reasons. Since concrete, widely used in construction, develops cracks over time due to factors like stress, if left unchecked, this can compromise the structural integrity and safety of buildings, bridges and roads. Identifying cracks using machine learning is beneficial for early crack detection to prevent catastrophic failures. It automates and enhances the detection process, improving speed, accuracy, and efficiency. This ultimately benefits maintenance operations and infrastructure management.

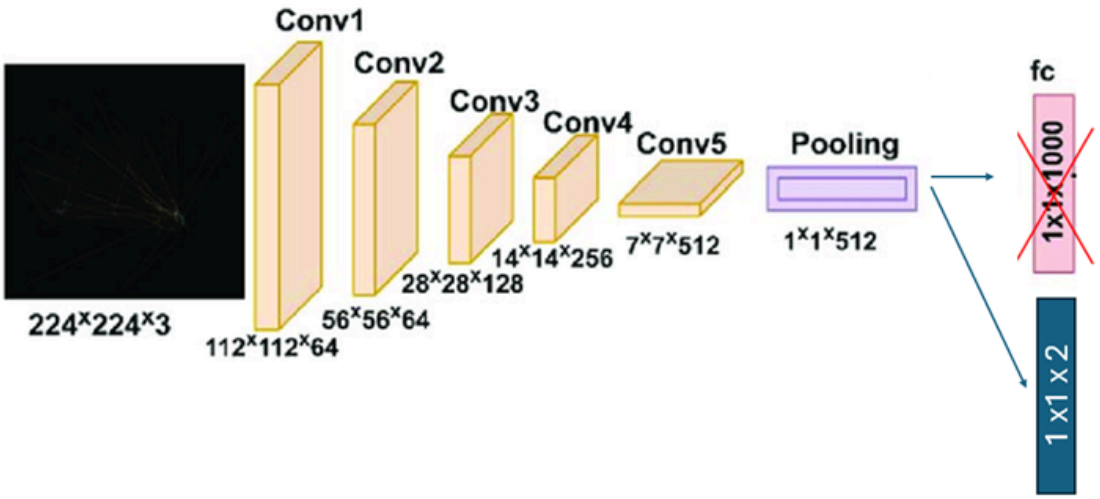
Data preprocessing

The dataset used for training in this project is chosen from Kaggle. It can be accessed through this link. It contains concrete images having cracks. It is divided into two classes as negative and positive crack images. Each class has 20000 images with a total of 40000 images with 227 x 227 pixels with RGB channels.

To ensure the images have the correct properties before being input into the model chosen, data transformations are performed on the images during the preprocessing stage. The images are resized to a fixed size of 224x224 pixels. The pixel values are transformed into a tensor and normalized by adjusting the mean and standard deviation for each channel. These values are selected based on the model chosen (discussed below).

Machine Learning model

The project is developed using the PyTorch library as well as subsidiary libraries such as NumPy and Matplotlib amongst others. The model selected for training is ResNet18, used as a fixed feature extractor, with a fully connected linear layer serving as the classifier. The parameters of the ResNet18 layers are frozen, meaning no backpropagation is applied to them during training. The final output layer consists of two neurons, each corresponding to one of the two classes.

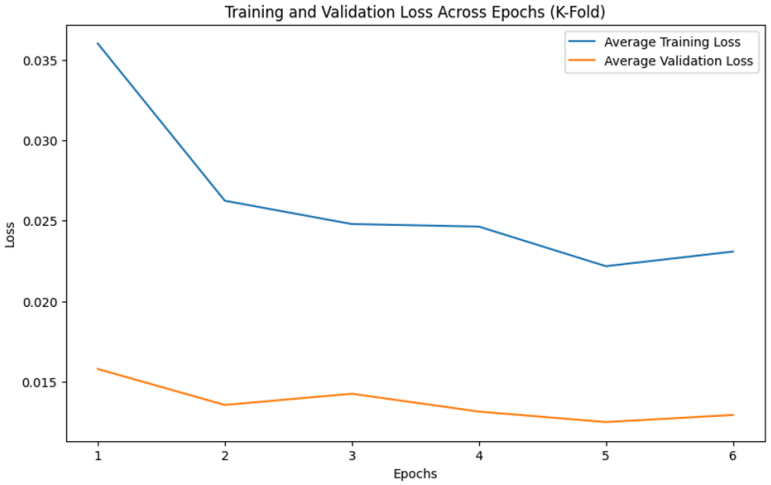


During the training phase, the loss function used is the Cross Entropy loss. For optimization, stochastic gradient descent is used with a learning rate of 0.001.

The dataset is split into two sets, a train set and a test set, with 85-15 split. K-fold cross validation is used with k = 5. Each fold is trained for 6 epochs. This is because the training accuracy and validation accuracy started at quite high values around 99% for fold 1, epoch 1.

Preliminary results

The average training loss continuously decreases with each epoch. However, the validation loss is consistently lower than the training loss, as shown below. This must be further investigated.



When the modified ResNet18 is tested on the test set, the following results are obtained:

- Test Accuracy: 0.9972
- Test Precision: 0.9972
- Test Recall: 0.9972
- Test F1-Score: 0.9972

In addition to testing the model with the test set of the above mentioned dataset (named dataset 1 onwards), another round of testing was used on a new dataset, named dataset 2, which can be accessed here. It consists of two folders: "positive" and "negative", containing images of cracked and non-cracked concrete surfaces, respectively. Each image in the dataset is in JPEG format, with a resolution of 227 x 227 pixels in RGB format.

The effect of transfer learning was studied. For the pretrained ResNet18 model with unmodified weights, a final fully connected layer was added since the model has 1000 output classes and we need 2 output classes. This layer is randomly initialized. It was compared with our model with modified weights for the final layer. The results are shown in the table below. The model with modified weights gives a better performance than the model with unmodified weights.

Performance on test dataset 2

	Pretrained model with unmodified weights	**Pretrained model with modified weights**
Accuracy	0.6802	0.8439
Precision	0.6916	0.8754
Recall	0.6802	0.8439
F1-Score	0.6754	0.8406

Next steps

In the next step, we will be building a CNN model from scratch and training it on dataset 1. The trained model will then be tested on dataset 2. This will allow us to compare the difference between using a pretrained model or a model built from scratch.