Concrete Integrity Analysis: A Machine Learning Approach to Crack Detection

1. Introduction

Concrete structures are prone to cracking, which can lead to severe structural integrity issues over time. The timely detection and classification of these cracks are essential for maintaining the safety and durability of infrastructure Cracks in concrete structures are a common phenomenon associated with corrosion, chemical deterioration, and the application of adverse loading. The appearance of these cracks is a sign of stress, weakness, and wear and tear within the structure, leading to possible failure/collapse [2]. Traditional methods of manual inspection are time-consuming, labour-intensive, and prone to errors. With the rise of artificial intelligence, particularly deep learning algorithms, there is a growing interest in automating this process through the use of image classification techniques.

In this project, we address the problem of detecting cracks in concrete structures using a subset of the SDNET2018 dataset, a well-known collection of images created for machine learning applications in crack detection[1] . To overcome the limitations of manual inspection, this study proposes an autonomous crack detection method using deep learning (DL) techniques, specifically convolutional neural networks (CNNs). The goal is to automate and accelerate the detection of cracks while enhancing accuracy and consistency. In this study, a subset of 1,801 images from the SDNET2018 dataset was used to train and evaluate the CNN models. SDNET2018 is a widely recognized dataset that includes over 56,000 images of cracked and non-cracked concrete surfaces, providing a challenging testbed for AI-based crack detection due to its diverse lighting conditions, surface irregularities, and obstructions

The subset of images selected for this project allows us to focus on the development of machine learning model for crack classification using fewer but representative images. By doing so, we aim to streamline the process of crack detection, ultimately improving the speed and accuracy of these models for real-world applications.

The results of this study demonstrate the effectiveness of using a subset of SDNET2018 to train and validate crack detection algorithms. We present an analysis of the performance of these models in terms of best model accuracy. The use of deep learning in this context enables more timely interventions, potentially preventing structural failures and ensuring the long-term safety of critical infrastructure.

2. Related Work

Concrete crack detection has been a critical focus area for researchers working on structural health monitoring and maintenance. Several studies have leveraged machine learning and deep learning approaches to address the problem of detecting cracks in infrastructure, primarily to improve speed, accuracy, and reliability over manual inspection methods.

Dorafshan et al. (2018) explored the potential of using deep learning algorithms for automated crack detection in concrete surfaces. They utilized the SDNET2018 dataset to train convolutional neural networks (CNNs) and demonstrated the potential of these models to outperform traditional image processing techniques. Their approach was designed to handle various obstructions such as shadows and surface roughness, which are commonly present in real-world applications.

Yamaguchi et al. (2017) applied image processing techniques such as thresholding and edge detection to detect cracks in concrete surfaces.Their approach worked well for structured environments with controlled lighting.

Our method focuses solely on CNN architectures, particularly MobileNetV2 , which simplifies the model while achieving similar, if not better, performance on crack detection tasks. By keeping the model structure simple, we maintain computational efficiency and avoid the need for complex model hybridization. our study uses a subset of 1,801 images from SDNET2018, focusing on efficiency in terms of data usage and computational cost. We also simplify preprocessing and focus on fine-tuning CNN parameters such as learning rate and epochs, demonstrating that high precision can be achieved even with a smaller dataset model.

3. Materials and Experimental Evaluation

3.1 Dataset

a) **Dataset Source**:
The dataset used in this project is a subset of the SDNET2018 dataset. SDNET2018 is a comprehensive collection of annotated images intended for training, validation, and benchmarking of artificial intelligence-based crack detection algorithms for concrete structures.

b) **Format and Structure**:
The dataset consists of images captured in the .JPG format, with each image segmented into 256 x 256-pixel sub-images. The images showcase both cracked and non-cracked concrete surfaces from bridge decks, walls, and pavements. For this project, a subset of 1,801 images has been selected.

7005-78

7010-17

70111-186

CRACK IMAGES

7001-67

7002-89

7002-160

NON-CRACK IMAGES

c) **Pre-processing**:
No changes were made to the images during preprocessing. The dataset was used in its original form, with images directly applied to the model without additional modifications.

d) **Number of Classes** and **Class Distribution**:The dataset is divided into two distinct categories:

i. Crack: Images containing visible cracks.
ii. Uncrack**:** Images that do not contain cracks.

The subset contains a total of 1,801 images, distributed across two classes Crack**:** 1,103 images (61% of the data) and Uncrack**:** 698 images (39% of the data).

e) **Data Collection and Processing**:
The dataset images were captured using a 16 MP Nikon digital camera, focusing on concrete surfaces from 54 bridge decks, 72 walls, and 104 pavements. Each full-size image was divided into smaller sub-images (256 x 256 pixels).

f) Training and Testing:

- ImageDataGenerator: This is a powerful utility provided by Keras (a deep learning library) that allows you to preprocess images in real-time while training the model. It can perform operations like rescaling, rotation, zooming, and more.
- preprocessing_function=preprocess_input: The preprocessing_function is used to specify a custom function that will be applied to each image before it is fed into the model. In this case, preprocess_input is likely a function provided by a specific model architecture (e.g., ResNet) that normalizes the input images in the way that model expects (e.g., scaling pixel values between -1 and 1). No changes were made to the images during preprocessing
- validation_split=0.2: This parameter splits the data into training and validation sets. Specifically, 20% of the data will be reserved for validation (to evaluate the model's performance on unseen data during training), and the remaining 80% will be used for training.

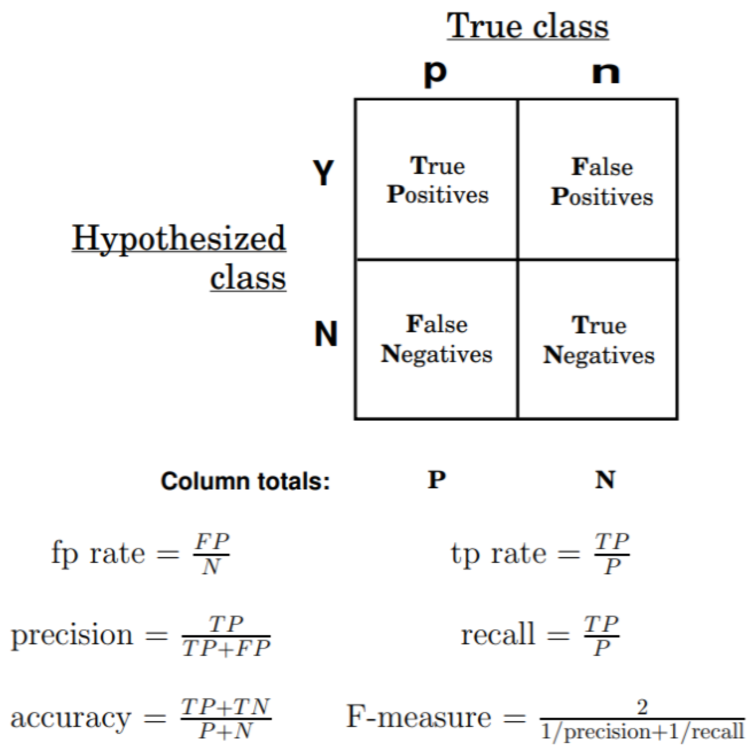
3.2 Methodology

The experiment tests the hypothesis that deep learning models (CNNs), especially those utilizing transfer learning with preprocessing methods, can accurately detect and classify concrete cracks.

Evaluation Criteria:
The performance of the model is evaluated based on the following criteria:

1. Accuracy: Measured on both training and validation datasets.
2. Loss: Training and validation losses are tracked throughout the epochs.
3. Confusion Matrix: Used to evaluate the classification performance by analyzing true positive (TP), true negative (TN), false positive (FP), and false negative (FN) rates.
4. F1 Score: A performance metric that balances precision and recall.
5. Best Model Accuracy: Captured based on validation accuracy with the lowest validation loss.

Given a classifier and an instance, there are four possible outcomes. If the instance is positive and it is classified as positive, it is counted as a true positive; if it is classified as negative, it is counted as a false negative. If the instance is negative and it is classified as negative, it is counted as a true negative; if it is classified as positive, it is counted as a false positive. Given a classifier and a set of instances (the test set), a two-by-two confusion matrix (also called a contingency table) [3].



$$\text{fp rate} = \frac{FP}{N} \qquad \text{tp rate} = \frac{TP}{P}$$

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{P}$$

$$\text{accuracy} = \frac{TP+TN}{P+N} \qquad \text{F-measure} = \frac{2}{1/\text{precision}+1/\text{recall}}$$

Confusion matrix and common performance metrics calculated from it

Experimental Methodology:

In this study, the methodology revolves around training and validating a Convolutional Neural Network (CNN) using the MobileNetV2 model pre-trained on the ImageNet dataset. The experiment was conducted as follows:

1. **Dataset**: A subset of 1,801 images from the SDNET2018 dataset was used, split into two classes: cracked and uncracked concrete surfaces. The dataset was divided into training -1481 (80%) and validation -369 (20%) sets using the ImageDataGenerator.
2. **Model Architecture**: A pre-trained MobileNetV2 was utilized as the base model with non-trainable layers. A custom classifier head was added:
   - GlobalAveragePooling2D layer
   - Dense layer (100 units with ReLU activation)
   - Final softmax layer with two output nodes (for binary classification)
3. **Optimizer ,Training and Validation**:
   The best model was selected using the Adam optimizer with a learning rate of 0.000001 to 0.05 The model was trained for 10 to 50 epochs with a batch size of 32. Early stopping was implemented to prevent overfitting, restoring the best model weights based on validation loss

Training/Test Data:

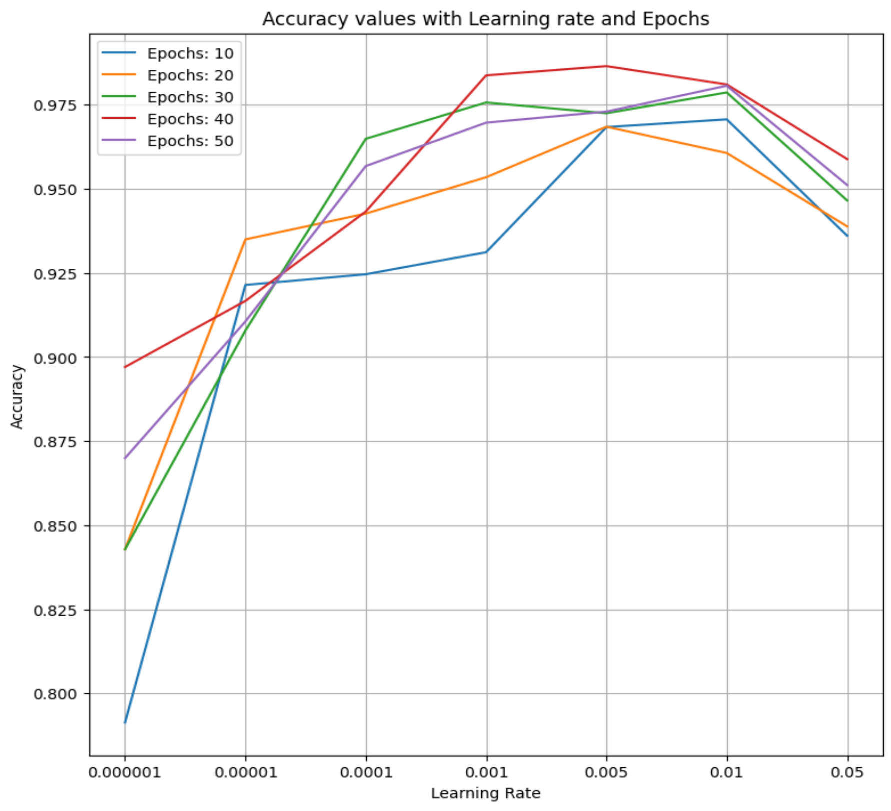
The training data comprised 80% of the 1,801 images (approx. 1,450 images), while the remaining 20% (approx. 350 images) was used for validation.

Performance Data:
The performance data collected during the training process includes:

1. Training and validation accuracy: Monitored throughout the epochs.
2. Training and validation loss: Used to evaluate overfitting or underfitting.
3. Confusion matrix: Shows the distribution of correct and incorrect classifications across both classes.
4. Best model accuracy: Captured when the validation loss reached its minimum value.

Performance analysis involved:

- Epoch-wise comparison: A comparison of model performance across different learning rates (ranging from 0.000001 to 0.05) and epoch counts (10 to 50 epochs).
- Model evaluation: Based on the confusion matrix, the model's ability to correctly classify cracked and uncracked images was scrutinized. .

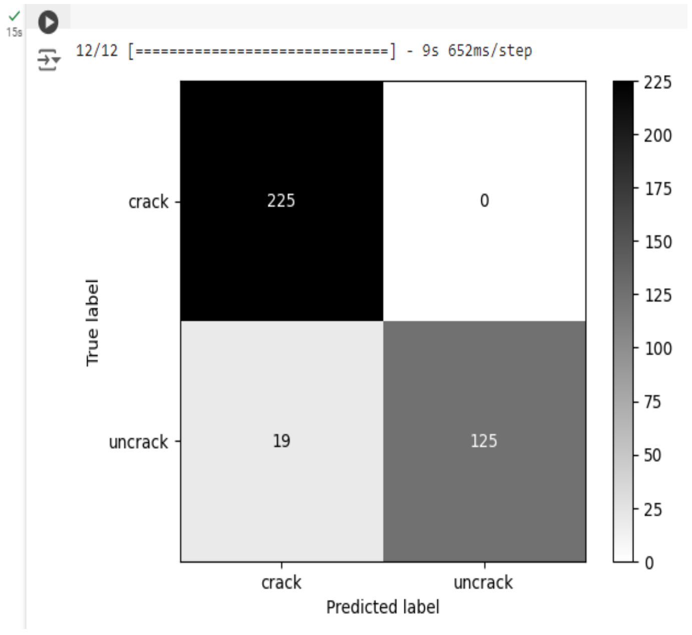


Epochs/learning rate	0.000001	0.00001	0.0001	0.001	0.005	0.01	0.05
10	0.791327894	0.92140919	0.924590135	0.931147564	0.968360658	0.970655725	0.93606559
20	0.842818439	0.934959352	0.942623019	0.953442631	0.968467235	0.960655725	0.938852437
30	0.842818439	0.907859087	0.96486485	0.975675702	0.972459014	0.978655725	0.946557369
40	0.897018969	0.916672108	0.943243265	0.983739853	0.986486495	0.981029809	0.958852437
50	0.869918704	0.910569131	0.956756771	0.969672108	0.972950792	0.980655725	0.951147564

3.3 Results

Confusion Matrix and Classification Report:

1. **Confusion Matrix**: The confusion matrix for the model shows excellent performance, with True Positives (TP) and True Negatives (TN) being dominant compared to False Positives (FP) and False Negatives (FN)[3]. The matrix provides the distribution of correct vs incorrect classifications for each class (cracked and uncracked).


```
12/12 [==============================] - 9s 652ms/step
```

2. **Classification Report**: The classification report provides precision, recall, and F1 scores for both classes:

    - Precision: The model shows average precision (0.95) for both classes, meaning that there are very few false positives.
    - Recall: Recall is also near 0.95, suggesting that most true cracks and non-cracks were correctly identified.
    - F1 Score: F1 scores for the cracked and uncracked classes were both over 95%, indicating the model's high ability to balance precision and recall.

Example: Model (40 Epochs at Learning Rate 0.005)

Class	**Precision**	**Recall**	**F1 Score**
Crack	0.92	1.00	0.96
Uncrack	1.00	0.87	0.93
Average	0.95	0.95	0.95

3.4 Discussion

Impact of Epochs: Increasing the number of epochs generally improved the performance, but the difference in performance between 10 and 20 epochs was significant, while additional epochs (30-50) provided diminishing returns.

4. Future Work

The dataset used (1,801 images) is relatively small, which can limit the generalization capabilities of the model and its robustness when deployed in real-world scenarios. A larger and more diverse dataset could be incorporated to increase the model's robustness. This may include gathering more images of different types of concrete cracks from different environments and lighting conditions.

Exploring advanced preprocessing techniques such as histogram equalization, contrast-limited adaptive histogram equalization (CLAHE), or even using generative adversarial networks (GANs) to create synthetic images could improve model performance. Further experimentation with image filters and multi-channel inputs (e.g., combining RGB, grayscale, and edge detection features) could also enhance the model's feature extraction.

Freezing the MobileNetV2 layers limited the model's ability to fine-tune and adapt to the specific features of the crack detection problem. Future work could involve unfreezing the later layers of MobileNetV2 and allowing fine-tuning of the model's weights, which may lead to improved performance by adapting pre-trained features to this specific task.

While the Adam optimizer was effective, other optimizers Further optimization of learning rates and regularization techniques like weight decay or dropout should be explored.

The current model focuses only on binary classification (cracked vs. uncracked), which may be a simplified problem in practical use cases. Extending the model to classify multiple types of cracks (e.g., longitudinal, transverse) or identifying crack severity (e.g., minor, moderate, severe) could provide more granular insights. This would involve labeling and incorporating more diverse datasets, as well as experimenting with multi-class classification architectures.

5. Conclusion

This project presented a deep learning approach using MobileNetV2 to classify cracked and uncracked concrete surfaces. The model achieved strong validation accuracy while avoiding overfitting through early stopping. The Adam optimizer with a learning rate of 0.005 yielded the best results, though other optimizers offered insights into hyperparameter tuning.The successful application of transfer learning for a practical civil engineering problem, demonstrating that pre-trained models like MobileNetV2 can be effective for concrete crack detection.

This research provides a foundation for future studies aiming to enhance the automated detection of structural damage in concrete, which could lead to faster, more cost-effective maintenance processes in civil engineering. The methods explored here could be extended to other materials and structural health monitoring tasks, further improving infrastructure safety and inspection procedures.

6. References

1. 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
2. Kim, I.-H.; Jeon, H.; Baek, S.-C.; Hong, W.-H.; Jung, H.-J. Application of Crack Identification Techniques for an Aging Concrete Bridge Inspection Using an Unmanned Aerial Vehicle. *Sensors* 2018, *18*, 1881. https://doi.org/10.3390/s18061881
3. Fawcett, T. ROC graphs: Notes and practical considerations for researchers. Mach. Learn. 2004, 31, 1–38
4. Golding, V.P.; Gharineiat, Z.; Munawar, H.S.; Ullah, F. Crack Detection in Concrete Structures Using Deep Learning. *Sustainability* 2022, *14*, 8117. https://doi.org/10.3390/su14138117