Convolutional Neural Networks for Crack Detection Classification

Using Convolutional Neural Networks for binary classification of concrete with and without cracks

Megan Arnold
DTSA 5511 Final
September 30th, 2025

Problem Overview

- Images contain very useful information, but are difficult to classify as there are potentially many features within a single image.
- Detecting cracks and deviations can enable early detection; thus, enabling timely repairs or replacement before catastrophic and expensive failure
- Challenge: Many features not relevant to cracks are present
- Challenge: Labels are binary for the images, nothing spatial is encode

Cracked Examples:

Uncracked Examples:

Project Goals

- Explore dataset to **understand data diversity** and guide model selections and identifying potential pitfalls and confounding factors
- **Build Baseline CNN Model** based on exploratory data analysis.
- **Evaluate model performance** and attempt to improve architecture and parameters for subsequent models
- **Visualize CNN gradient heatmaps** to determine the strengths and weaknesses of models; thus, guiding future architecture decisions
- Use **Data Augmentation** on the best performing model to determine if that will help the model generalize
- **Evaluate hyperparameter** tuning outcomes on test dataset, via Kaggle Submission
- Make future work **recommendations**

Dataset Overview

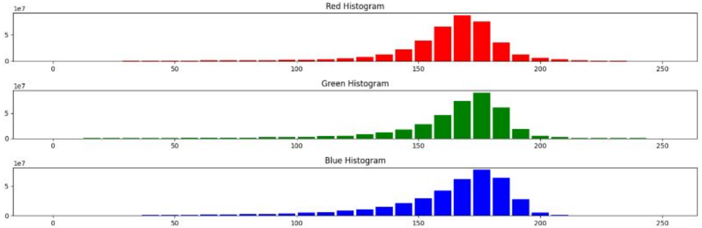
- About 7500 cracked and uncracked images
- Balanced dataset between binary classification
- Observations
    - Cracked:
        - Thick and thin cracks
        - Some are hardly visible to the naked eye
        - No spatial labeling
    - Uncracked:
        - Many uncracked images have significant craters

Examples of Data:

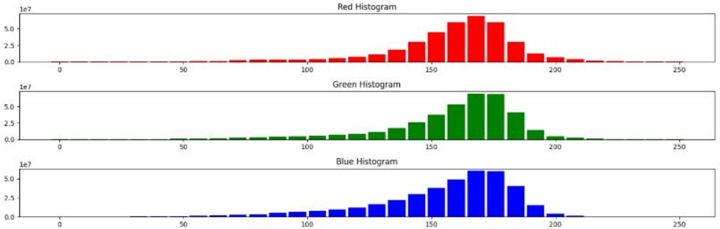
Exploratory Data Analysis

- 256 by 256 pixels with 3 color channels
- Balanced Dataset
- Consistent histograms on pixel values. Reduces the probability of the model learning a shortcut
- Challenges:
    - There appears to be a potential confounding factor of craters in uncracked images
    - *Note: My first baseline model used the craters as an important feature for classification, as observed by the heatmaps

Cracked Histograms:

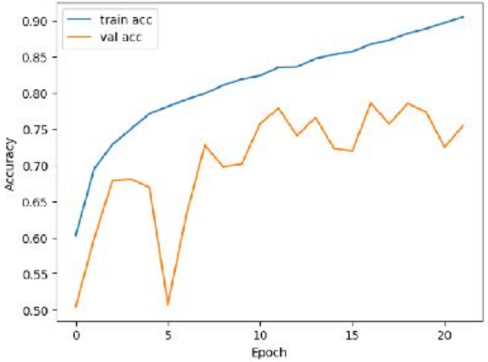


Uncracked Histograms:



Baseline Convolutional Model

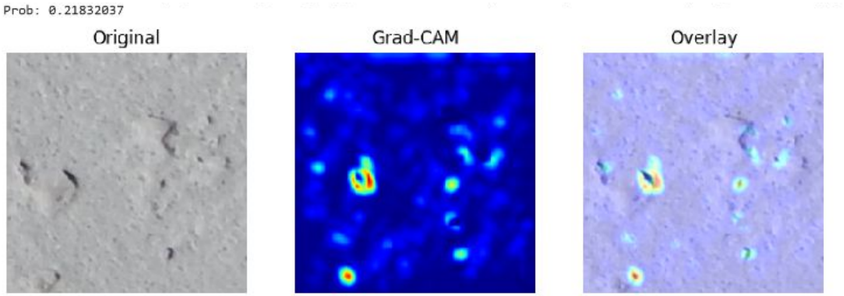
- Rescaling to pixel value range of [0,1]
- 7 Convolutional Layers
    - 32,32,64, 64, 128, 128, 256
- Batch Normalization between odd and even layers
- MaxPooling between even and odd layers
- Global Average Pooling between CNN and classification blocks
- Dense layer of 64 neurons
- Sigmoid activation neuron

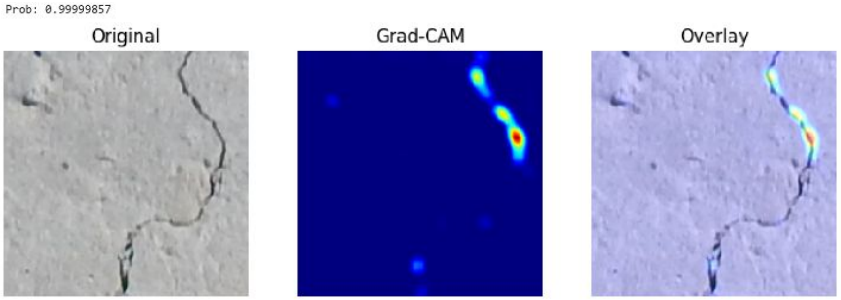


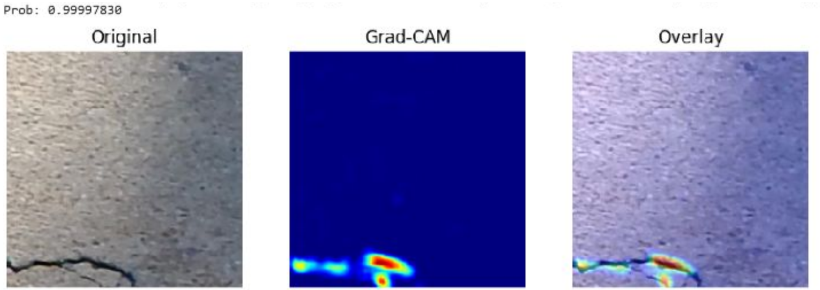

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling (Rescaling)       (None, 256, 256, 3)       0

 conv2d (Conv2D)             (None, 256, 256, 32)      896

 batch_normalization (Batch  (None, 256, 256, 32)      128
 Normalization)

 conv2d_1 (Conv2D)           (None, 256, 256, 32)      9248

 max_pooling2d (MaxPooling2  (None, 128, 128, 32)      0
 D)

 conv2d_2 (Conv2D)           (None, 128, 128, 64)      18496

 batch_normalization_1 (Bat  (None, 128, 128, 64)      256
 chNormalization)

 conv2d_3 (Conv2D)           (None, 128, 128, 64)      36928

 max_pooling2d_1 (MaxPoolin  (None, 64, 64, 64)        0
 g2D)

 conv2d_4 (Conv2D)           (None, 64, 64, 128)       73856

 batch_normalization_2 (Bat  (None, 64, 64, 128)       512
 chNormalization)

 conv2d_5 (Conv2D)           (None, 64, 64, 128)       147584

 conv2d_6 (Conv2D)           (None, 64, 64, 256)       295168

 global_average_pooling2d (  (None, 256)               0
 GlobalAveragePooling2D)

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 64)                16448

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 599585 (2.29 MB)
Trainable params: 599137 (2.29 MB)
Non-trainable params: 448 (1.75 KB)
_________________________________________________________________
```

Baseline Observations

- Test Accuracy: 0.78
- General Observations
    - Started overfitting around epoch 10
    - Significant focus on craters instead of cracks
    - Focus on small local areas instead of global context
    - Some focus on parts of a crack that look like craters
    - No edge artifacts seen

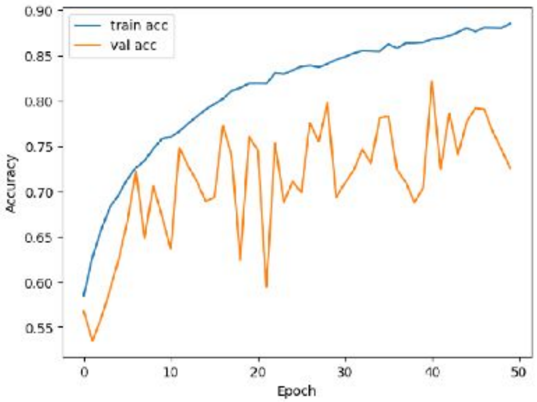






Model and Hyperparameter Tuning Version 2

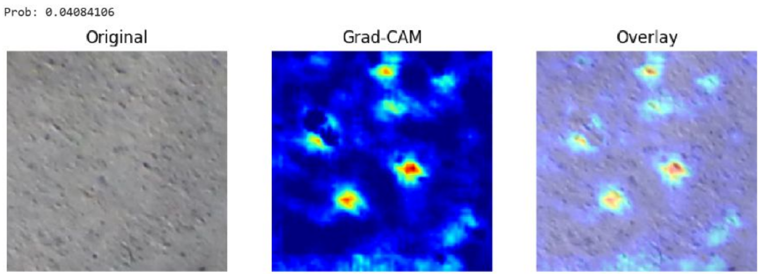
- Overfitting Fixes:
  - Decrease Dense layer in classification from 64 to 16
  - Increased the dropout before classification
- Global vs Local Context Fixes:
  - Added Dilation rate of 2 to the final two convolutional layer
  - Changed padding of first convolutional layer to “valid” instead of “same”

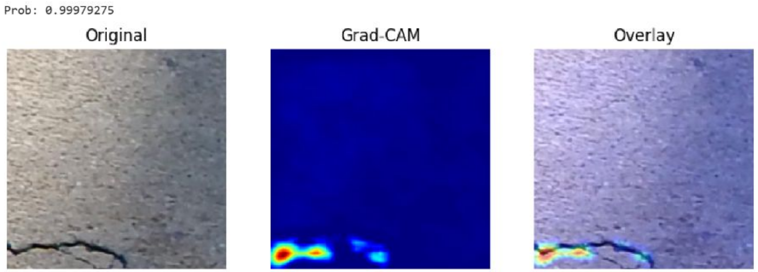


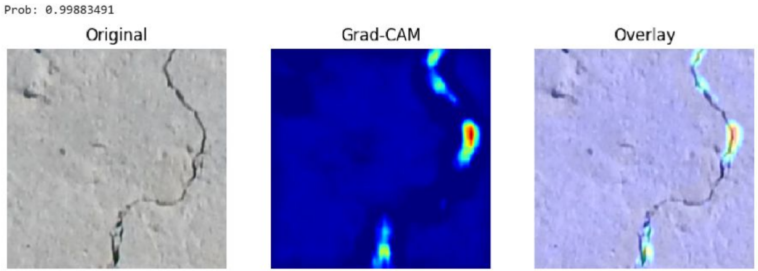
Model Description	Validation Accuracy	Public Kaggle Accuracy	Private Kaggle Accuracy
Model Baseline	~0.85	0.780	0.773
Version 2	~0.85	0.721	0.699

Version 2 Observations

- Test Results: 0.721
- Observations:
    - Edge artifacts on visualization heat maps
        - Suggests “valid” isn’t an ideal choice here
    - Focus on craters still
    - Focus on small local contexts still
- Significant changes need to occur for the third version

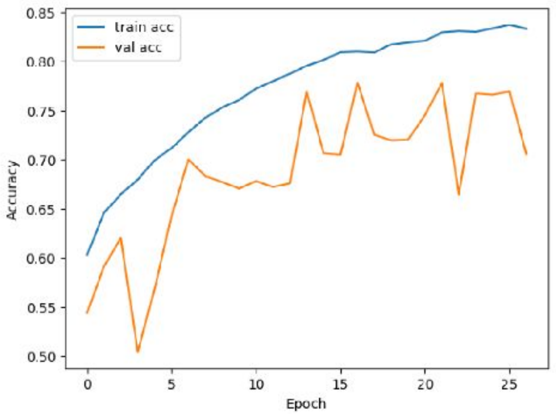






Model and Hyperparameter Tuning Version 3

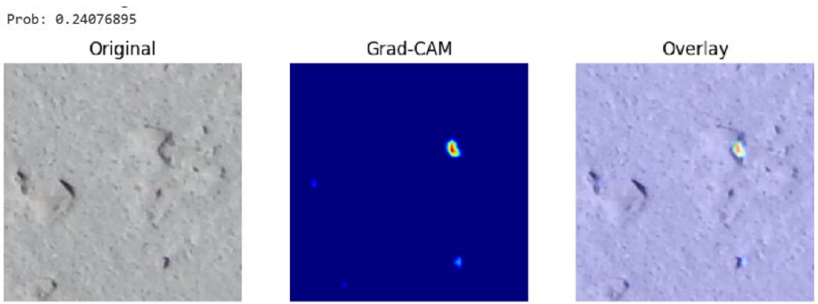
- Global vs Local Context Fixes
  - Increased kernel size from 3 to 5 of first convolutional layer
  - Added LeakyReLU as activation instead of ReLU. Allows negative values to propagate, although at a decreased rate
  - Average Pooling Instead of Max Pooling
    - Max Pooling was hiding the darker cracks
- Overfitting
  - Added regularization to the convolutional layers
  - Kept the decreased Dense Layer size
- Edge Artifacts
  - Change back to “same” padding

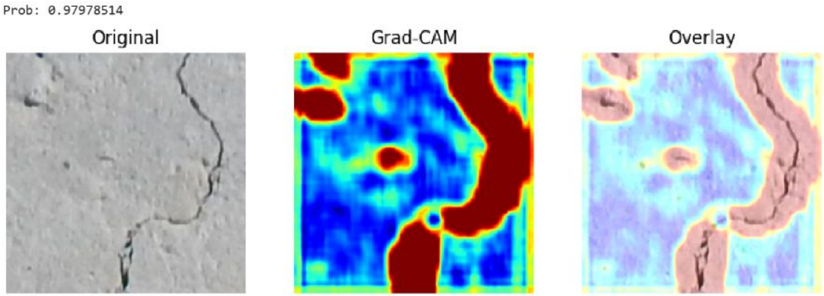


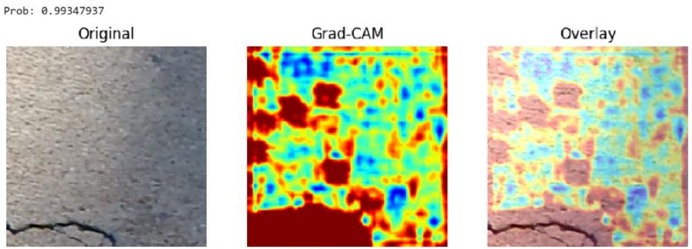
Model Description	Validation Accuracy	Public Kaggle Accuracy	Private Kaggle Accuracy
Model Baseline	~0.85	0.780	0.773
Version 2	~0.85	0.721	0.699
Version 3	~0.86	0.795	0.771

Version 3 Observations and Results

- Test Results: 0.795
- Observations
  - Smaller overfitting gap between train and validation metric
    - Regularization
  - Ignores non-relevant craters/features
  - Has global context on cracks
    - Kernel size increase
  - Better at identifying thin cracks
    - Average Pooling
- Overall, best model so far and doesn't highlight confounding factors, only cracks.
- Explainable model when GradCAM is applied

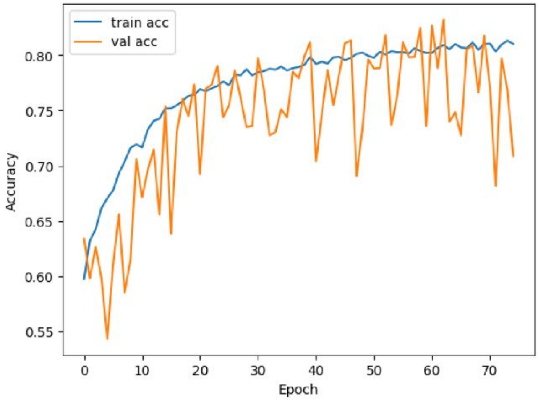






Data Augmentation

- Augmentation Layers
  - Random Contrast
  - Random Flip
  - Random Rotation
  - Random Zoom
  - Random Translation
  - Gaussian Noise
- Test Result: 0.713
- Hypothesis
  - Augmentation was too aggressive and cause the cracks to potentially be removed if they were near the edge.
  - Future work should tune this portion of the model



Model Description	Validation Accuracy	Public Kaggle Accuracy	Private Kaggle Accuracy
Model Baseline	~0.85	0.780	0.773
Version 2	~0.85	0.721	0.699
Version 3	~0.86	0.795	0.771
Augmented Version 3	~0.77	0.713	0.704

Recommendations, Use Cases, and Next Steps

- Results
  - Convolutional Neural Network for Binary Classification of images
  - Explainable model with additional GradCAM visualization
  - Model is capable of identifying the entire crack in the global context
  - Model is capable of ignoring the non-relevant, but confounding factors of craters
- Benefits
  - Improves the ability to automate defect identification
  - Less expensive than having human inspectors
  - Enables earlier identification of defects; thus, potentially mitigating costly repairs and replacements

- Next Steps
  - Have **spatially labeled** images with either bounding boxes or individual pixels
  - Further hyperparameter tuning with the architecture and classification head to increase **accuracy** and **explainability**
  - Further work with **data augmentation** to decrease overfitting and reduce the amount of training data needed
  - Explore transfer learning of commonly used models

Conclusion

- Binary Classification Convolutional Neural Networks are able to still highlight the important features in the original image that is important to the classification
- The model was relatively fast to train and inference on a commercially available GPU
- Future work:
  - Improved spatial labeling with bounding boxes instead of binary labeling for entire image
  - Data Augmentation to further reduce overfitting
  - Hyperparameter tuning to further improve performance