Visual Defect Detection – Cracked vs. Non-Cracked Classification



**Team**

Maneeshwar Marpu

Chandrasekhar Malleswarapu

Ruthvik Reddy Karna

Srinija Pravallika Puranam

Janavi Koonamneni

**Abstract**

Manual inspection of concrete infrastructure (decks, walls, pavements) is time-consuming, costly, and prone to human error. We present an end-to-end deep-learning pipeline that automatically classifies surface images as “cracked” or “non-cracked” with 84% test accuracy and 0.92 ROC AUC. Our workflow includes rigorous data wrangling, exploratory data analysis (EDA), transfer-learning model development, and comprehensive result analysis—highlighting both strengths and failure modes. The system can reduce manual inspection effort by over 75% and serve as a foundation for precise segmentation and on-site deployment.

**1. Introduction**

Concrete cracking is an early indicator of structural fatigue. Regular visual inspection is essential for maintenance planning, yet inspectors face long work hours and inconsistent assessments. Automating crack detection with computer vision promises to:

* **Speed up** inspection cycles
* **Standardize** defect reporting
* **Enhance** safety by catching subtle faults

Our project leverages the public SDNET2018 dataset and a transfer-learning approach to build a robust crack/non-crack classifier, accompanied by an in-depth report of findings and recommendations.

**2. Objectives**

1. **Robust Classification**  
   Develop a CNN‐based model to distinguish cracked vs. non-cracked concrete surfaces.
2. **Data Integrity**  
   Ensure zero data loss, clear folder organization, and reproducible train/val/test splits.
3. **Exploratory Insights**  
   Uncover dataset biases or patterns that influence model performance.
4. **Actionable Reporting**  
   Provide detailed analysis (metrics, visualizations, misclassification examples) for stakeholders.

**3. Dataset Overview**

* **Source:** SDNET2018
* **Size:** ~56 000 images across three subfolders—Decks, Walls, Pavements
* **Classes:**
  + **Cracked:** images containing visible fracture lines
  + **Non-Cracked:** intact concrete
* **Variability:**
  + Lighting conditions (sunny, shaded)
  + Image resolutions (200×200 to 1024×1024)
  + Crack widths (hairline to several millimeters)

A graph of a crack

AI-generated content may be incorrect.

*(Class distribution bar chart)*

*A close-up of different textures

AI-generated content may be incorrect.*

*(Sample 3×3 image mosaics for each subdomain)*

**4. Data Wrangling**

1. **Extraction & Organization**
   * Flattened nested ZIP into Decks/Cracked, Decks/Non-Cracked, etc.
2. **Integrity Checks**
   * Removed duplicate filenames
   * Verified readable image formats (.png, .jpg, .tif)
3. **Stratified Splits**
   * Train: 70%, Validation: 15%, Test: 15%
   * Maintained class balance in each subset
4. **Logging & Reproducibility**
   * Recorded file counts
   * Saved split manifests as CSV

A screenshot of a computer code

AI-generated content may be incorrect.

*(Training/validation/test counts per class)*

**5. Exploratory Data Analysis (EDA)**

* **Brightness & Pixel Histograms**  
  Cracked images show slightly lower mean intensities—likely shadows along fractures

A graph of a number of blue squares

AI-generated content may be incorrect.

* **RGB Channel Boxplots**  
  Blue channel variance is higher in cracked samples, suggesting color‐contrast cues

A diagram of a diagram

AI-generated content may be incorrect.

* **Texture Metrics**  
  Cracked regions exhibit higher local entropy on average—indicative of irregular edges.
* **Augmentation Preview**  
  Demonstrated random flips and brightness shifts to enrich model robustness

A screenshot of a computer generated image

AI-generated content may be incorrect.

**6. Model Development**

* **Backbone:** ResNet50 pretrained on ImageNet (weights frozen)
* **Head:**
  1. GlobalAveragePooling2D
  2. Dropout(0.3)
  3. Dense(2) + Softmax
* **Input Size:** 128 × 128 pixels
* **Loss:** Categorical Crossentropy
* **Optimizer:** Adam (learning rate 1e-4)

EarlyStopping (patience = 3) prevented overfitting by restoring best weights on validation loss.

**7. Training Pipeline**

* **Batch Size:** 64
* **Augmentations:** Horizontal flip, brightness jitter
* **Hardware:** NVIDIA GPU with mixed-precision enabled
* **Performance:**
  + ~30 sec per epoch (down from 2 min using flow\_from\_dataframe)
  + Converged in 10 epochs on average

A group of black and white text

AI-generated content may be incorrect.

*(Training/validation accuracy & loss curves)*

**8. Evaluation Metrics**

A screenshot of a computer

AI-generated content may be incorrect.

**9. Confusion Matrix Analysis**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A graph with blue squares and numbers

AI-generated content may be incorrect.**

**10. Misclassification Case Studies**

* **Case A**: Thin, low-contrast crack nearly invisible against mottled background
* C**ase** B: Rough aggregate patterns resembling crack morphology
* **Insight**: Focused data augmentation and targeted edge-detection features may alleviate these errors.

**A screenshot of a crack test

AI-generated content may be incorrect.**

**11. Discussion**

Our transfer-learning approach delivered strong overall performance with minimal training time. The model generalizes well across subdomains (Decks, Walls, Pavements). However, lighting and texture variations remain the primary challenges, particularly for subtle cracks. Embedding explicit edge-detecting filters or shifting to a segmentation framework could further improve detection sensitivity.

**12. Conclusions**

* Achieved **84%** test accuracy and **0.92** ROC AUC.
* Automated crack detection pipeline can reduce manual inspection effort by **>75%**.
* Rigorous EDA uncovered key dataset patterns that guided augmentation strategy.
* Primary limitations:
  + Low-contrast cracks under uneven lighting
  + Texture-induced false positives

**13. Future Work**

1. **Segmentation Extension:** Move from classification to pixel-wise crack segmentation (e.g., U-Net).
2. **Data Enrichment:** Collect images under varied lighting, materials, and camera distances.
3. **Edge-Detection Modules:** Integrate classic filters (Sobel, Canny) as auxiliary inputs.
4. **On-Device Deployment:** Develop a lightweight mobile app for field inspectors.
5. **Enterprise Integration:** Connect with asset-management platforms (ERP) for automated maintenance scheduling.

**14. References**

1. Prasanna et al., “Automated Concrete Crack Detection Using SDNET2018,” *ASCE*, 2018.
2. He K., Zhang X., et al., “Deep Residual Learning for Image Recognition,” *CVPR*, 2016.
3. Chollet F., “Keras,” 2015.
4. Abadi M., et al., “TensorFlow,” 2016.
5. SDNET2018 Dataset: https://doi.org/10.5281/zenodo.