Detailed Documentation - Scratch Detection Project

1. Project Overview
This project focuses on detecting scratches on images containing printed text. The objective is to classify images as 'good' or 'bad', with bad images having scratches on the text, and also localize these scratches using masks and bounding boxes.

2. Dataset Description
- Total images: ~5000+ (good and bad combined)
- Good images: Images with clear, unscratched text.
- Bad images: Contain various degrees of scratches over the text.
- Masks: Optional binary images indicating scratch regions in bad images (255 = scratch, 0 = background).

Dataset Statistics:

- Total images: ~5200
- Good images: ~4178
- Bad images: ~1028
- Image format: .jpg or .png
- Image size: Varies (resized to 224x224 during processing)

Environment and Libraries Used

- Python (Anaconda/Jupyter Notebook)
- TensorFlow / Keras
- OpenCV
- NumPy
- Matplotlib

3. Data Preparation
- Images resized to 224x224.
- Used ImageDataGenerator for data augmentation and rescaling.
- Train-test split of 90% for training, 10% for validation.
- All data loaded via Keras flow_from_directory for binary classification.

4. Classification Models Implemented

Five different CNN-based architectures were implemented and trained:

1. Custom CNN – Lightweight 2-layer CNN model.
2. MobileNetV2 – Efficient model suitable for mobile deployment.
3. ResNet50 – Deep residual network known for high accuracy.
4. VGG16 – Large model with deep feature extraction.
5. InceptionV3 – Wide network with mixed convolutions for efficient detection.

5. Model Training & Evaluation

- Each model was trained for 10 epochs using Adam optimizer.
- Binary crossentropy used as the loss function.
- Accuracy and loss were tracked for both training and validation sets.
- Training histories were plotted for model comparison.

6. Mask Generation & Scratch Localization

- OpenCV was used to read grayscale masks and detect contours.
- Bounding boxes were drawn using cv2.boundingRect over contours with significant area (>50 px).
- Visualizations included original image, mask, and box overlay for easy inspection.
- For good images, if mask was empty, a message indicating 'no scratch detected' was shown.

7. Visualizations

- Used matplotlib to visualize the following:
  - Original image
  - Masked image
  - Bounding box overlay
- Graphs for training and validation accuracy/loss for all models.
- Overlay options for mask + bounding box for presentation quality.

Test data: 560 images (split from the original set)

Here are the results for each model:

A. Custom CNN:
- Accuracy: 87.5%
- Precision: 82.3%
- Recall: 78.1%

- F1 Score: 80.1%

B. MobileNetV2:
- Accuracy: 92.1%
- Precision: 89.4%
- Recall: 85.2%
- F1 Score: 87.2%

C. ResNet50:
- Accuracy: 94.3%
- Precision: 92.1%
- Recall: 90.4%
- F1 Score: 91.2%

D. VGG16:
- Accuracy: 91.2%
- Precision: 88.3%
- Recall: 86.7%
- F1 Score: 87.5%

E. InceptionV3:
- Accuracy: 93.5%
- Precision: 91.7%
- Recall: 88.1%
- F1 Score: 89.9%

Why It Worked / Didn’t Work

- Custom CNN worked decently but lacked deep feature extraction.

- MobileNetV2 gave good results with minimal computation and size.

- ResNet50 was best due to deep residual learning.

- VGG16 gave stable results but was slower.

- InceptionV3 handled variability in scratch shapes well due to mixed kernels.

- Scratch localization via mask/bounding box was accurate for clean masks; less effective on simulated ones with noise.

8. Bonus Features Implemented
- Bounding box and scratch mask generation for bad images.
- Simulated scratch generator for generating synthetic bad images from good ones.
- Bounding box overlay with contour filtering.
- Ready to extend to object detection models like YOLO, U-Net, etc.

9. Suggested Extensions
- Use YOLOv8 for real-time scratch detection.
- Train a U-Net or Mask R-CNN for pixel-level segmentation.
- Add GUI for user scratch threshold control.
- Generalize to different scratch types: screen, glass, metal.

Future Work
- Train U-Net or Mask R-CNN for better scratch segmentation.

- Add threshold tuning GUI for scratch severity.

- Extend to other types of scratches: screen cracks, metallic scratches.

- Add a web-based dashboard for demo.

10. Author & Credits
Created by: Nandha Kumar Vijayan
Role: Developer & Researcher
Platform: Jupyter Notebook (Anaconda)
Contact: nandhav6@gmail.com