Detailed Report: Scratch Detection on Text Images

Author: Rahul Gour

Email: gaurr210@gmail.com

Date: December 2, 2024

Objective:

The objective of this project is to train a machine learning model capable of classifying images of text as either 'Good' (clear text) or 'Bad' (scratched text). The model uses deep learning techniques to detect scratches and can be extended to identify scratches on other surfaces like metallic surfaces or phone screens.

Code Explanation:

1. Dataset Handling:

- The dataset is a zipped file containing two folders: 'good' (clear text) and 'bad' (scratched text).

- The script extracts the dataset to ensure proper structure before proceeding to training.

2. Data Preprocessing:

- Images are resized to 128x128 pixels and rescaled to normalize pixel values to the range [0, 1].

- A training and validation split is applied to ensure robust model evaluation.

3. Model Architecture:

- The model is a Convolutional Neural Network (CNN) built using TensorFlow/Keras.

- Layers include Conv2D, MaxPooling2D, Flatten, Dense, and Dropout to prevent overfitting.

- The output layer uses a sigmoid activation function for binary classification.

4. Training:

- The model is compiled with 'adam' optimizer and 'binary_crossentropy' loss function.

- Early stopping is used to prevent overfitting by monitoring the validation loss.

5. Evaluation and Visualization:

- Training and validation accuracy and loss are plotted to analyze model performance.
- Sample images from the dataset are displayed with their predicted labels.

6. Testing:

- A single image can be tested using the 'predict_image' function, which outputs whether the image is classified as 'Good' or 'Bad'.

7. Submission Notes:

The trained model and scripts are packaged for submission. Ensure that sensitive data like the dataset is not publicly shared.

Setup Instructions:

1. Clone the Repository:

```
git clone <your-repo-link>
cd <your-repo-directory>
```

2. Install Required Dependencies:

```
pip install tensorflow numpy matplotlib
```

3. Dataset Structure:

   - Dataset Directory: Should include 'good' and 'bad' folders.
   - Example structure:

```
dataset/
    good/
    bad/
```

4. Run the Script:

```
   python train_and_evaluate.py
```

5. Test the Model:

```
   python test_model.py --image_path path/to/your/image.jpg
```

Acknowledgments:

This project was authored by Rahul Gour. Special thanks to resources and communities that provided guidance on deep learning and dataset handling.