Intel Unnati Industrial Training Program 2024

Karunya Institute of Technology and Science

Pixelated Image Detection & Correction

Faculty Mentor: Mr. S. BASIL XAVIER

Team members:

Finney Rajan S (URK22CS1056)

David Rosario Selvaraj (URK22CS1007)

Shannandhirshath T (URK22CS1045)

Melvin Matthew N (URK22CS1157)

Beneesh S (URK22CS1190)

Pixelated Image Detection

Abstract:

In this project, we developed a machine learning model to detect and classify pixelated images using a convolutional neural network (CNN) based on the MobileNetV2 architecture. The project involved preprocessing a dataset of images, training the model for binary classification, and evaluating its performance using various metrics. Additionally, the project included validation on new datasets and calculating the frames per second (FPS) for model inference, demonstrating both the accuracy and efficiency of the model.

The model was built using TensorFlow and Keras, and trained on a custom dataset with pixelated and non-pixelated images. We leveraged transfer learning with the pre-trained MobileNetV2 model, fine-tuning it with additional layers for our classification task. Early stopping was used to prevent overfitting.

After training, the model achieved high validation accuracy, with strong performance metrics including accuracy, F1 score, and confusion matrix analysis. It was further validated on a separate dataset to ensure robustness.

The results demonstrated that the model could efficiently and accurately classify pixelated images, making it valuable for image quality assessment and enhancement applications.

Methodology:

Dataset Collection and Preprocessing:

To train our binary classifier for detecting pixelated images, we utilized a total of 1,140 images from the DIV2K image dataset and the 'Wide Screen Image Dataset' obtained from Kaggle. These datasets encompass a diverse range of content such as natural landscapes, buildings, people, animals, vehicles, and anime scenes. Additionally, we manually included specific anime images to augment the dataset.

Our dataset consisted of 1,140 images. Since our model required both classes—pixelated and non-pixelated images for effective training—we created pixelated versions of our dataset. This involved downsizing the original images by factors of 5x or 6x and subsequently upscaling them by the same factors using either nearest neighbor or bilinear interpolation methods. Each image was preprocessed to fit the model's input size of 224x224 pixels. Instead of resizing, which could distort the image and affect the detection task, we opted for cropping. This method preserved the original texture and details essential for accurate pixelation detection.

Model selection (MobileNetV2), transfer learning, and fine-tuning:

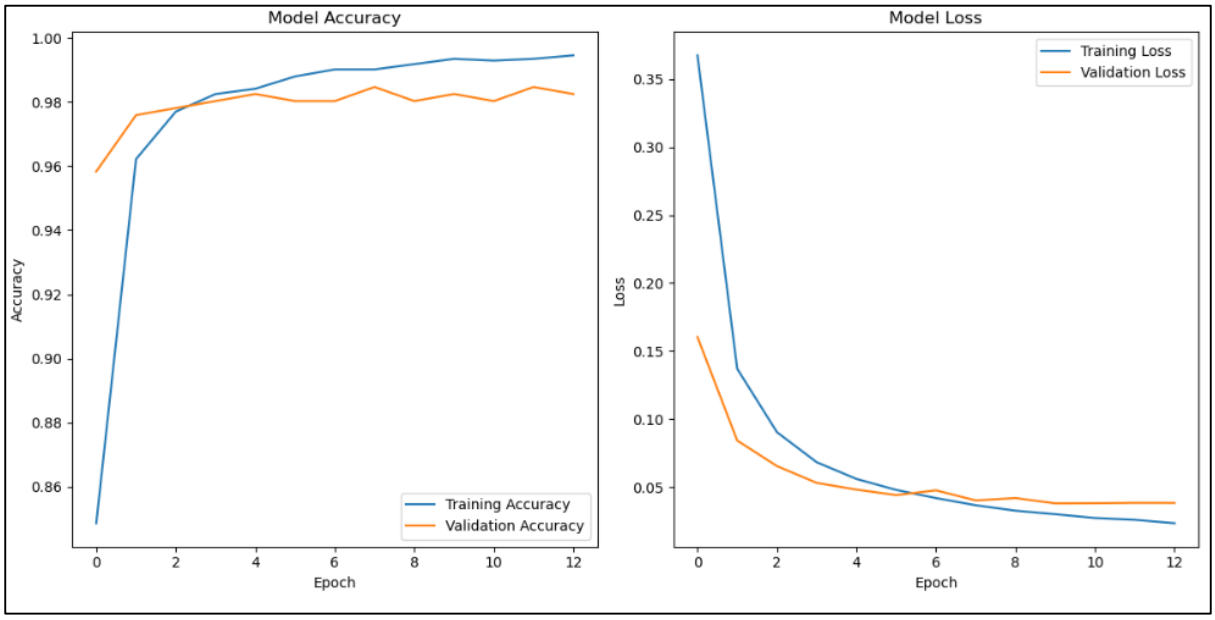
We chose MobileNetV2 for its efficiency and strong performance in image classification. Using transfer learning, we initialized it with pre-trained weights from ImageNet and excluded the top layers (include_top=False). This allowed us to add custom dense layers tailored for our task of distinguishing pixelated from non-pixelated images. We froze the base MobileNetV2 layers (base_model.trainable = False) to retain the pre-trained weights, enhancing model stability and efficiency. After adding a global average pooling layer to reduce feature map

dimensions, a dense layer with 128 units and ReLU activation extracted pertinent features crucial for pixelated image detection.

Training Details:

For training the pixelated image detection model, we utilized the MobileNetV2 architecture pre-trained on ImageNet and fine-tuned it with additional layers. During training, the model was configured with an Adam optimizer, where the learning rate was set to 1e−4. The model was trained over 13 epochs, as early stopping was employed to prevent overfitting and ensure optimal generalization. A batch size of 32 was used for efficient processing, and the training data underwent rescaling to normalize pixel values, ensuring consistent data range. The dataset was split into training and validation sets using an 80-20 split, with stratification to maintain class balance. This setup facilitated robust training and validation, achieving a peak validation accuracy of 98.25%.

```
Found 1824 images belonging to 2 classes.
Found 456 images belonging to 2 classes.
Epoch 1/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 47s 692ms/step - accuracy: 0.7432 - loss: 0.5057 - val_accuracy: 0.9583 - val_loss: 0.1603
Epoch 2/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 37s 650ms/step - accuracy: 0.9550 - loss: 0.1560 - val_accuracy: 0.9759 - val_loss: 0.0842
Epoch 3/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 38s 664ms/step - accuracy: 0.9762 - loss: 0.1005 - val_accuracy: 0.9781 - val_loss: 0.0654
Epoch 4/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 39s 667ms/step - accuracy: 0.9822 - loss: 0.0715 - val_accuracy: 0.9803 - val_loss: 0.0530
Epoch 5/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 38s 661ms/step - accuracy: 0.9868 - loss: 0.0539 - val_accuracy: 0.9825 - val_loss: 0.0481
Epoch 6/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 39s 666ms/step - accuracy: 0.9930 - loss: 0.0452 - val_accuracy: 0.9803 - val_loss: 0.0441
Epoch 7/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 38s 666ms/step - accuracy: 0.9902 - loss: 0.0442 - val_accuracy: 0.9803 - val_loss: 0.0476
Epoch 8/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 39s 682ms/step - accuracy: 0.9888 - loss: 0.0379 - val_accuracy: 0.9846 - val_loss: 0.0401
Epoch 9/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 39s 671ms/step - accuracy: 0.9919 - loss: 0.0357 - val_accuracy: 0.9803 - val_loss: 0.0419
Epoch 10/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 40s 680ms/step - accuracy: 0.9935 - loss: 0.0322 - val_accuracy: 0.9825 - val_loss: 0.0380
Epoch 11/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 42s 734ms/step - accuracy: 0.9894 - loss: 0.0343 - val_accuracy: 0.9803 - val_loss: 0.0381
Epoch 12/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 42s 725ms/step - accuracy: 0.9946 - loss: 0.0281 - val_accuracy: 0.9846 - val_loss: 0.0384
Epoch 13/50
57/57 ━━━━━━━━━━━━━━━━━━━━ 43s 737ms/step - accuracy: 0.9951 - loss: 0.0221 - val_accuracy: 0.9825 - val_loss: 0.0383
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We
recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my_model.keras')`.
15/15 ━━━━━━━━━━━━━━━━━━━━ 9s 590ms/step - accuracy: 0.9843 - loss: 0.0326
Validation Accuracy: 98.25%
```



Validation:

The loaded model achieved a validation accuracy of 98.25% using the evaluate method on a validation subset from the dataset. The evaluation yielded an accuracy score of 97.86% and a loss of 0.0373

Validation ¶

```
# Evaluate the model
loss, accuracy = loaded_model.evaluate(validation_generator)
print(f'Validation Accuracy: {accuracy * 100:.2f}%')
```

```
15/15 ━━━━━━━━━━━━━━━━━━━━ 8s 544ms/step - accuracy: 0.9786 - loss: 0.0373
Validation Accuracy: 98.25%
```

F1 Score:

The model was evaluated on 1200 images from the Caltech 256 image dataset, with 600 images in each class (pixelated and non-pixelated). Predictions were made using the loaded model, resulting in an F1 score of 0.96.

```
Found 1200 images belonging to 2 classes.
38/38 ━━━━━━━━━━━━━━━━━━━━ 22s 534ms/step
F1 Score: 0.96
```

Confusion matrix:

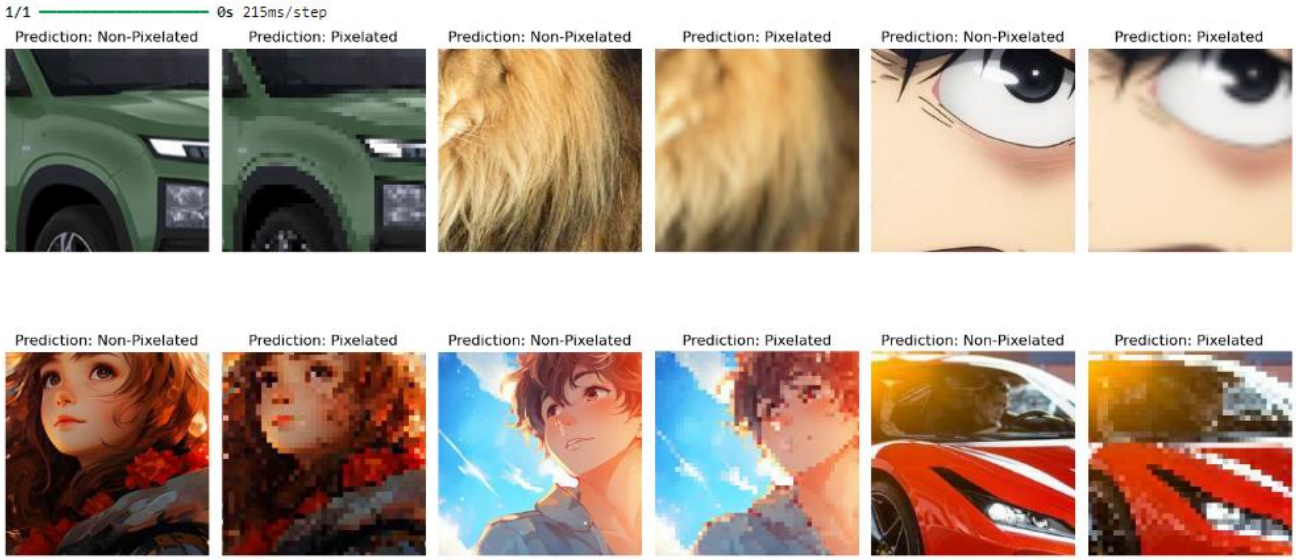
```
Confusion Matrix:
[[599   1]
 [ 49 551]]
Classification Report:
              precision    recall  f1-score   support

Non-Pixelated       0.92      1.00      0.96       600
    Pixelated       1.00      0.92      0.96       600

     accuracy                           0.96      1200
    macro avg       0.96      0.96      0.96      1200
 weighted avg       0.96      0.96      0.96      1200
```

FPS Calculation:

The FPS calculation was performed on a batch of 400 images, measuring the time taken for the model to process all images collectively

```
13/13 ━━━━━━━━━━━━━━━━━━━━ 10s 634ms/step
Number of images: 400
Elapsed time: 9.76 seconds
FPS: 41.00
```

Sample Output:



Conclusion:

In this project, we developed a machine learning model using MobileNetV2 to detect and classify pixelated images. Through preprocessing, transfer learning, and fine-tuning, we achieved high validation accuracy and robust performance metrics. The model effectively distinguished pixelated from non-pixelated images, validating its accuracy and efficiency on new datasets.