Documentation

for

DeepFake Face Identification

Using CNN and Explainable AI Techniques

Prepared by Dhruvi Sachin Shah

and Sanjana Gummuluru

Submitted to Dr. Rajeswari Sridhar

CSPC54 Project, Fall 2024

National Institute of Technology, Tiruchirappalli

Table of Contents

Project Overview

Abstract

This project leverages a Convolutional Neural Network (CNN) to detect DeepFake images and applies LIME, an Explainable AI (XAI) method, to make the model's predictions interpretable. Deepfake detection is a crucial task given the rise in AI-generated media, and understanding how the model makes its decisions is critical to ensure reliability and trust. The CNN is trained to classify real and fake images, and XAI methods help visualise the key features influencing each prediction.

Motivation

DeepFake, which is a portmanteau of the terms ‘deep learning’ and ‘fake’, is a new vein of AI generated fake videos synthesised using generative ML models. They can achieve high degrees of realism and have thus been used in malignant ways, manipulating people into believing something is real when it is not. They can thus have numerous negative implications, especially in media, necessitating the need for a reliable DeepFake detection model.

However, one major limitation in current DeepFake detection methods is that they operate as ‘black boxes’. Although these models achieve high accuracy, they offer little insight into how decisions are made, which can be problematic in real-world applications where interpretability and trust are essential. The usage of XAI techniques would make the decision-making process of AI systems transparent and understandable to humans. Additionally, we can ensure that the model is detecting relevant artefacts in DeepFake images rather than unrelated patterns in the data, making the model more robust and reliable in real-world applications.

Objectives

1. Train a CNN model to classify images as real or DeepFake.
2. Implement the LIME explainability techniques to visualise model predictions.
3. Use these techniques to understand and debug model performance.

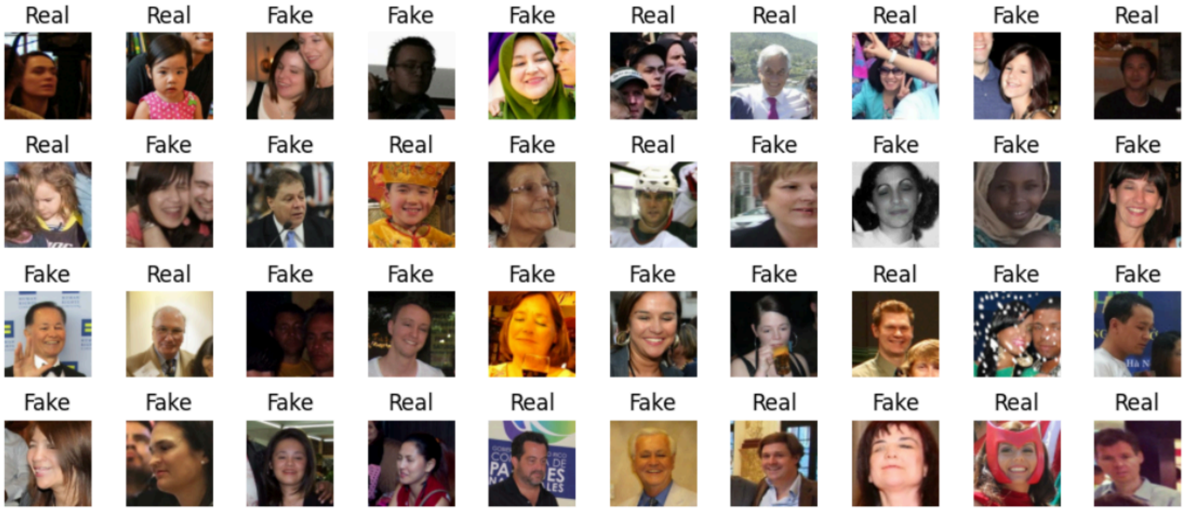
Dataset

We have utilised the OpenForensics Dataset, which contains over 190k images (both real and fake) split into train, test, and validation sets.

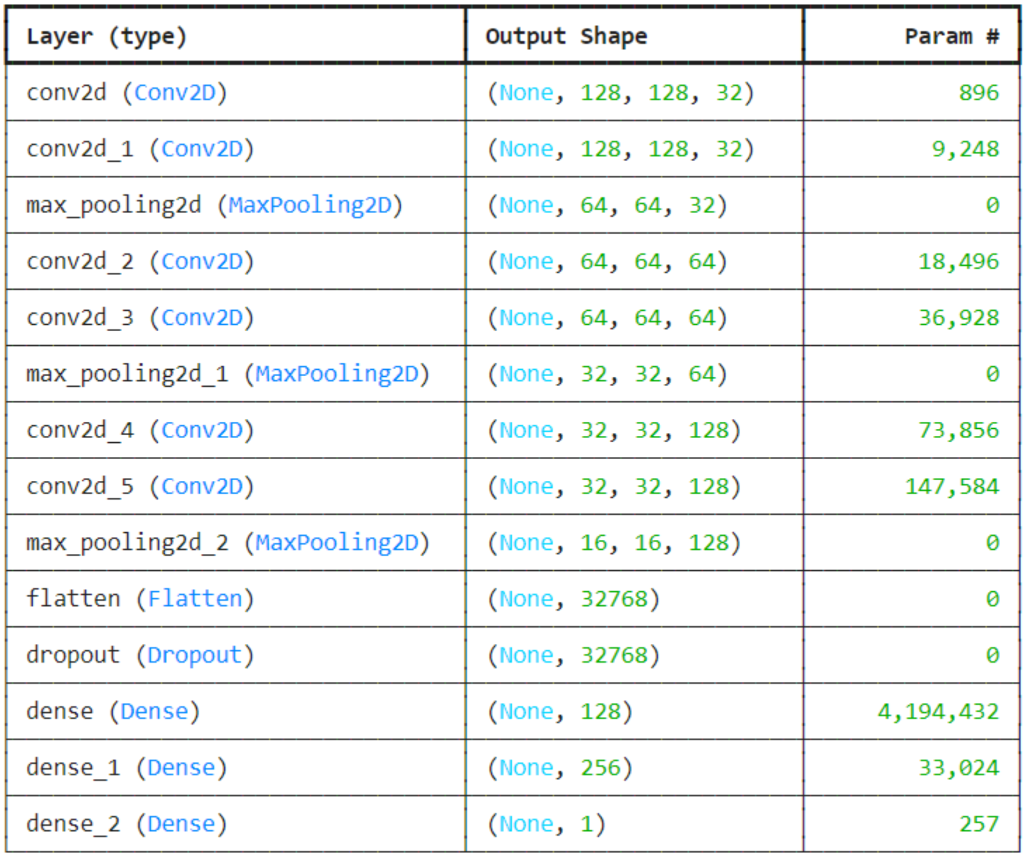
Preprocessing steps:

1. Resized images to 128x128 pixels.
2. Normalised pixel values to the range [0,1].

Below is a snippet of 40 images from the dataset:



CNN Architecture



Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 128, 128, 32)	896
conv2d_1 (Conv2D)	(None, 128, 128, 32)	9,248
max_pooling2d (MaxPooling2D)	(None, 64, 64, 32)	0
conv2d_2 (Conv2D)	(None, 64, 64, 64)	18,496
conv2d_3 (Conv2D)	(None, 64, 64, 64)	36,928
max_pooling2d_1 (MaxPooling2D)	(None, 32, 32, 64)	0
conv2d_4 (Conv2D)	(None, 32, 32, 128)	73,856
conv2d_5 (Conv2D)	(None, 32, 32, 128)	147,584
max_pooling2d_2 (MaxPooling2D)	(None, 16, 16, 128)	0
flatten (Flatten)	(None, 32768)	0
dropout (Dropout)	(None, 32768)	0
dense (Dense)	(None, 128)	4,194,432
dense_1 (Dense)	(None, 256)	33,024
dense_2 (Dense)	(None, 1)	257

1. Conv2D

The convolutional layers are designed to automatically learn features from images (like edges, patterns, textures). The activation function used here is ReLU (Rectified Linear Unit), which helps the model learn faster by making all negative values zero and leaving positive values as they are.

2. Dropout

This layer randomly "turns off" half of the neurons during each training step by 50%. This helps prevent the model from overfitting.

3. MaxPooling2D

These layers reduce the spatial size (width and height) of the feature maps, which helps to reduce the number of parameters and computations in the network. It takes the maximum value from every 2x2 block of pixels.

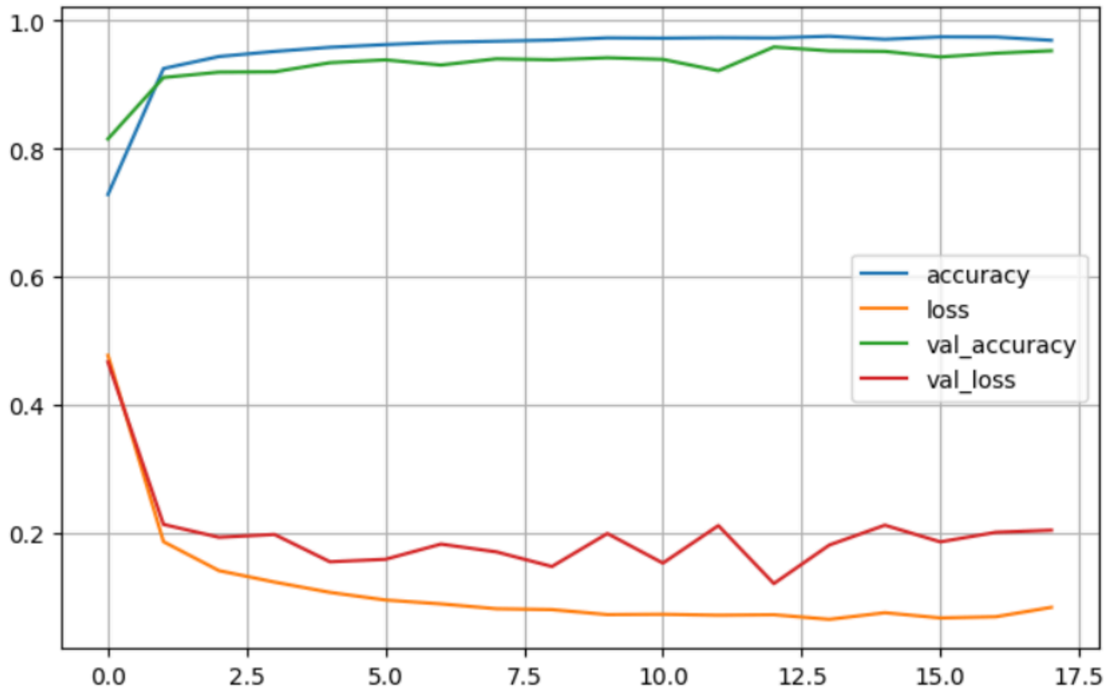
4. Flatten

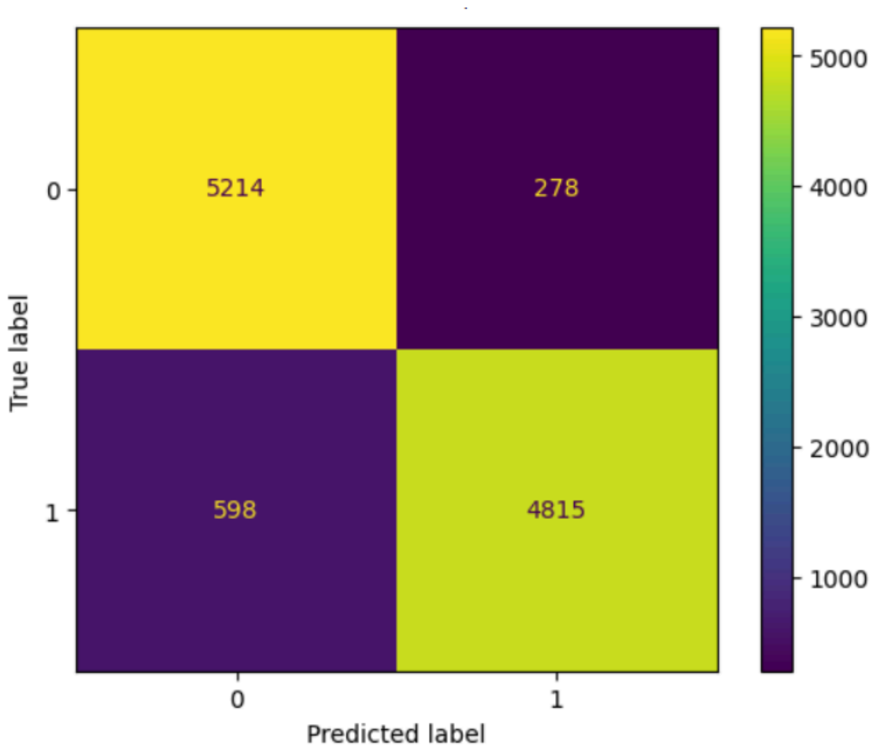
This layer converts the 2D feature maps into a 1D vector. This is necessary because the next layers are fully connected (dense) layers, which require a flat input.

5. Dense

These are fully connected layers. Each neuron is connected to every neuron in the previous layer. These layers help the model learn complex combinations of the features extracted by the convolutional layers. ReLU activation is used again here to introduce non-linearity, enabling the model to learn complex patterns.

Model Performance





True Positives (TP) = 5214 (top left)

False Positives (FP) = 278 (top right)

False Negatives (FN) = 598 (bottom left)

True Negatives (TN) = 4815 (bottom right)

Hence,

Accuracy = 0.920

Precision = 0.949

Recall = 0.897

F1 Score = 0.922

Specificity = 0.945

Explainable AI (XAI) Methods

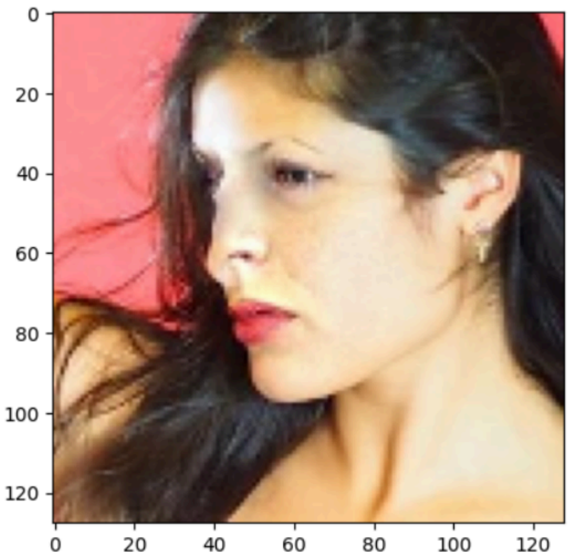
Deep learning models like CNNs are powerful but often opaque, making it difficult to understand how predictions are made. XAI techniques such as LIME are employed to explain individual predictions and provide feature importance, helping to ensure the model makes decisions based on meaningful image features.

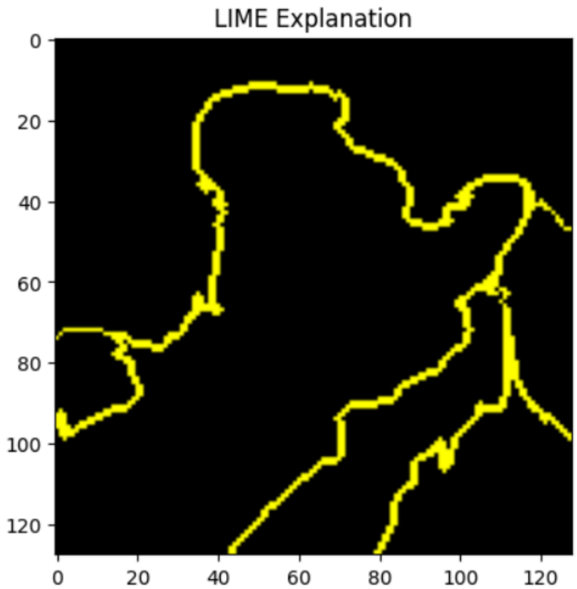
LIME (Local Interpretable Model-agnostic Explanations) is used to explain the CNN's predictions by creating perturbed versions of the input image and observing the changes in the model's output. This provides a local explanation for each prediction by highlighting regions of the image that influenced the decision.

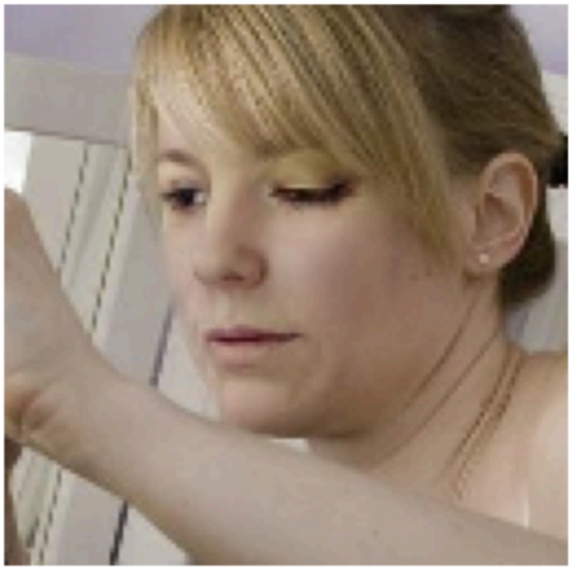
LIME was able to explain the predictions by highlighting the areas in the image that most influenced the CNN's decision. For example, in one case, LIME highlighted the edges of a person's face, where the artefacts introduced by deepfake algorithms were prominent.

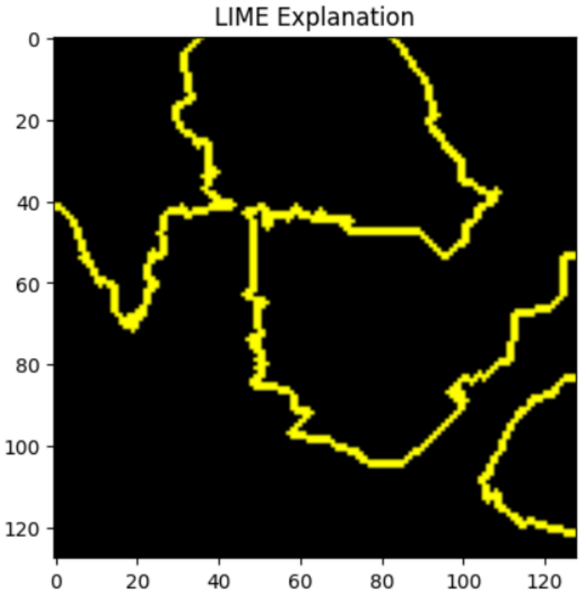
Below are 2 such examples of LIME explanations for images:

The yellow regions in the images correspond to the parts most influential in the model's prediction of whether the image is real or a DeepFake.




LIME Explanation




LIME Explanation

Code and Repository

The GitHub repository for this project can be found here.

It contains the following:

1. **cnn-deepfake.ipynb**, where the CNN model was built.

2. **deepfake-detector-model.keras**, the resultant model.

3. **xai-deepfake.ipynb**, the Explainable AI using LIME.

4. **references**, a folder containing all the papers referenced.

5. **review_reports**, a folder containing all reports for the project reviews throughout the semester.

6. **Documentation.pdf**, this document.

7. **README.md**, a short explanation of the project.

References

[1] Alben Richards, Kaaviya Varshini, Diviya N et al, “Deep Fake Face Detection using Convolutional Neural Networks”, IEEE, 2023.

[2] Patel, Y., Tanwar, S. et al, “An Improved Dense CNN Architecture for Deepfake Image Detection”, IEEE, 2023.

[3] Shraddha Suratkar, Faruk Kazi et al, “Exposing DeepFakes Using Convolutional Neural Networks and Transfer Learning Approaches”, IEEE, 2020.

[4] Hayat Al-Dmour, Afaf Tareef et al, “Masked Face Detection and Recognition System Based on Deep Learning Algorithms”, Journal of Advances in Information Technology, 2023.

[5] Jifeng Dai, Haozhi Qi et al, “Deformable Convolutional Networks”, Microsoft Research, 2017.

[6] Donnelly, J., Barnett, A.J. et al, “Deformable ProtoPNet: An Interpretable Image Classifier Using Deformable Prototypes”, IEEE.