Deep Fake Detection Using CNN Model

Team Members

Shiva Reddy Dubbaka (16352875)

Varshith Gandhe (16354840)

Karthikeya Arra (16354793)

Naga Sahithi Vunnam (16355633)

Introduction

The fast advancement of artificial intelligence (AI) technology has resulted in a new type of digital manipulation known as "deepfakes." This phrase, derived from the deep learning algorithms used to make them, refers to synthetic media in which a person's appearance is changed in an existing image or video, typically without their permission. These changes are made feasible by breakthrough machine learning and artificial neural network technologies, notably Generative Adversarial Networks (GANs), which can produce photorealistic photos and movies with astounding precision.

Deep Fake Detection is the process of identifying bogus photographs that were created using deep learning methods is known as "deep fake detection." Machine learning algorithms are used to replace or alter certain elements of an original video or image, like a person's face, to create deepfakes. Deepfake detection aims to recognise these kinds of alterations and separate them from authentic photos or videos.

Deepfake technology initially received widespread notice in late 2017, when a Reddit user, who coined the term "deepfake," began uploading realistic-looking fake pornographic videos of celebrities. Since then, technology has not only gotten more sophisticated, but also more accessible, with a plethora of tools and software available that allow even inexperienced users to produce deepfakes. This accessibility poses serious issues since the technology may be used to fabricate evidence, impersonate prominent personalities, manipulate stock markets, or deliberately target private persons.

Related Work

CNN designs: Several research have used different CNN designs to increase detection accuracy. Afchar et al. (2018) introduced MesoNet, a CNN with fewer layers that focuses on the mesoscopic features of pictures. This network is intended to identify mid-level face characteristics, which are commonly distorted in deepfakes.

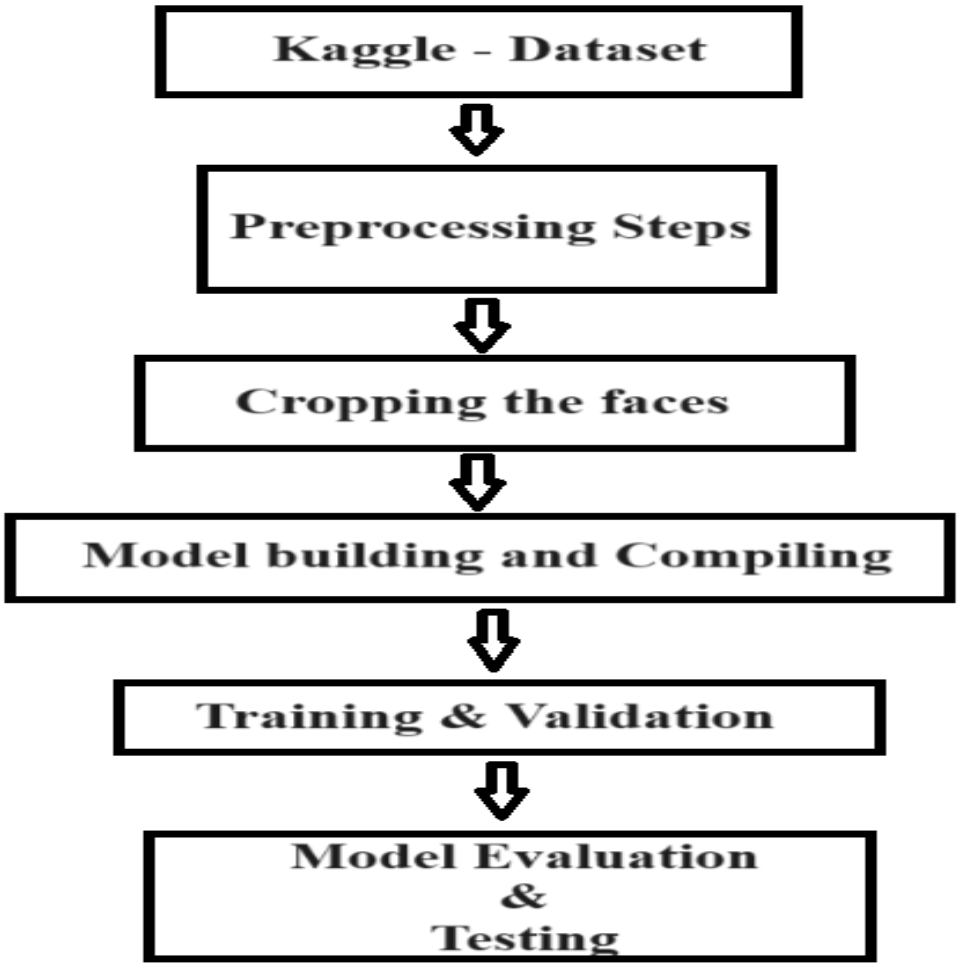
Autoencoder and GAN-based models: Cozzolino et al. (2018) used an autoencoder-based technique, in which a network is trained particularly to rebuild deepfake pictures. The reconstruction mistakes indicate modified content. Similarly, Nguyen et al. (2019) demonstrate how Generative Adversarial Networks (GANs) were used to produce and detect modified pictures, resulting in a self-improving detection system.

Leveraging Pre-trained Models: Researchers have also investigated the use of transfer learning to adapt existing, robust models learned on big datasets to the specific objective of deepfake detection. This strategy eliminates the requirement for large deepfake training sets, which are challenging to assemble. Rozsa et al. (2020) showed that models pre-trained on facial recognition tasks may be fine-tuned to detect alterations with high accuracy.

Few and zero-shot learning approaches: These methods are especially promising for identifying deepfakes that use previously undiscovered manipulation techniques. Few-shot learning includes training models using a small number of instances from a given class, with the goal of generalising effectively to new, comparable classes.

Standardised Dataset Creation: As the field expands, there is a greater demand for comprehensive and diverse datasets that represent the changing nature of deepfake technology. Face Forensics++, Celeb-DF, and the Deepfake Detection Challenge Dataset have all proved useful for training and assessing deepfake detection models.

Methodology



1.Data Preprocessing:

To strengthen the model against tampering outside the training data, picture augmentation techniques were used with Keras' ImageDataGenerator. This includes random transformations such rotations, width and height changes, shearing, zooming, and horizontal flipping. These modifications improve the model's ability to generalise to new, previously unknown data.

Training: The model was trained using the Adam optimizer and binary cross-entropy loss, using a systematic method to tracking training accuracy and loss.

2. Face Detection and Cropping:
The process includes a phase to recognise and crop faces in the photos, which was done using OpenCV's pre-trained Haar Cascade classifier. This emphasis on face areas is critical since

most deepfake algorithms change facial characteristics. Cropped photos ensure that the model focuses on the most important attributes for deepfake identification.

3. Model Architecture: CNN Design
The convolutional neural network created for this project comprised of the following layers:

Convolutional Layers: To extract diverse characteristics from pictures, use many convolutional layers with increasing filter sizes (e.g., 32, 64, 128). Each convolutional layer uses the ReLU activation function to add nonlinearity, allowing the model to learn complicated patterns.
Pooling Layers: Max pooling layers were added after each convolutional layer to minimise the spatial size of the representation, minimising the amount of parameters and computations in the network and thereby reducing overfitting.

4. Training & Validation:
Training: The model was trained using 100 epochs and a batch size of 256. Early stopping monitored the validation loss and stopped training if it did not improve after 5 consecutive epochs.
Validation: A second validation set was utilised to modify the hyperparameters and assess the model's performance throughout training. This recurrent method meant that the model learned to generalise to new data rather than simply memorising the training data.

5. Evaluation and Metrics:
To evaluate the model's performance, many metrics were calculated:

Accuracy: The percentage of properly detected photos (including actual and deepfakes).
Precision, recall, and F1-Score: These measures gave information on the model's capacity to reliably identify deepfakes without misclassifying actual photos.
The Confusion Matrix provided a thorough perspective of the model's performance across many classes, emphasising true positives, true negatives, false positives, and false negatives.

Results and Discussions

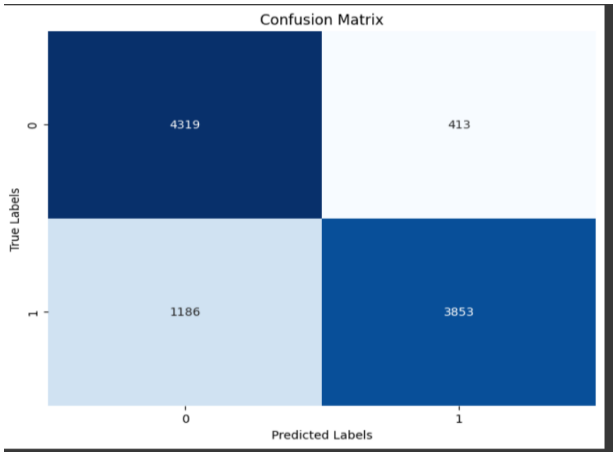
Training outcomes:
Accuracy gains: The model showed constant gains in training accuracy, demonstrating its capacity to learn discriminative features from training data successfully.
Loss Reductions: Loss metrics dropped as training epochs progressed, indicating that the model was becoming more reliable in making accurate predictions.

Testing and Validation:
Generalisation Capability: On previously unknown test data, the model scored a high accuracy rate, indicating that it has excellent generalisation skills.
Performance Metrics: The usage of confusion matrices and classification reports gave thorough insights into the model's performance, with a focus on accuracy, recall, and F1-scores across classes.

```
510/510 [==============================] - 146s 287ms/step - loss: 0.1657 - accuracy: 0.9320 - val_loss: 0.2401 - val_accuracy: 0.9037
Epoch 5/100
510/510 [==============================] - 148s 290ms/step - loss: 0.1440 - accuracy: 0.9415 - val_loss: 0.2072 - val_accuracy: 0.9133
Epoch 6/100
510/510 [==============================] - 149s 291ms/step - loss: 0.1248 - accuracy: 0.9495 - val_loss: 0.2181 - val_accuracy: 0.9074
Epoch 7/100
510/510 [==============================] - 146s 286ms/step - loss: 0.1111 - accuracy: 0.9550 - val_loss: 0.2200 - val_accuracy: 0.9189
Epoch 8/100
510/510 [==============================] - 148s 290ms/step - loss: 0.0958 - accuracy: 0.9616 - val_loss: 0.2587 - val_accuracy: 0.9103
Epoch 9/100
510/510 [==============================] - 148s 291ms/step - loss: 0.0815 - accuracy: 0.9674 - val_loss: 0.2623 - val_accuracy: 0.9151
Epoch 10/100
510/510 [==============================] - 147s 288ms/step - loss: 0.0669 - accuracy: 0.9733 - val_loss: 0.3140 - val_accuracy: 0.9152
Epoch 11/100
510/510 [==============================] - 147s 287ms/step - loss: 0.0599 - accuracy: 0.9767 - val_loss: 0.3528 - val_accuracy: 0.9146
Epoch 12/100
510/510 [==============================] - ETA: 0s - loss: 0.0491 - accuracy: 0.9807Restoring model weights from the end of the best epoch: 7.
510/510 [==============================] - 147s 288ms/step - loss: 0.0491 - accuracy: 0.9807 - val_loss: 0.3160 - val_accuracy: 0.9129
Epoch 12: early stopping
Epoch 1: Accuracy = 0.7877975106239319
Epoch 2: Accuracy = 0.9005042314529419
Epoch 3: Accuracy = 0.9206501245498657
Epoch 4: Accuracy = 0.9320372939109802
Epoch 5: Accuracy = 0.9414933323860168
Epoch 6: Accuracy = 0.9494781494140625
Epoch 7: Accuracy = 0.9550490975379944
Epoch 8: Accuracy = 0.9616009593009949
Epoch 9: Accuracy = 0.9674477577209473
Epoch 10: Accuracy = 0.9732792973518372
Epoch 11: Accuracy = 0.9766663312911987
Epoch 12: Accuracy = 0.9807276725769043
```

Evaluation Techniques

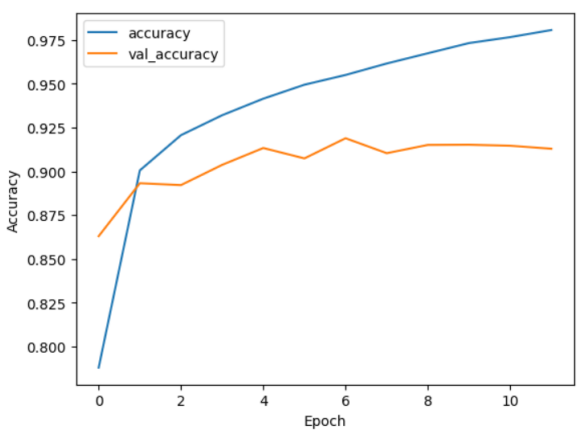
Confusion Matrix Visualisation: Enhanced using graphical libraries, these visualisations aided in analysing the model's strengths and identifying misclassifications.

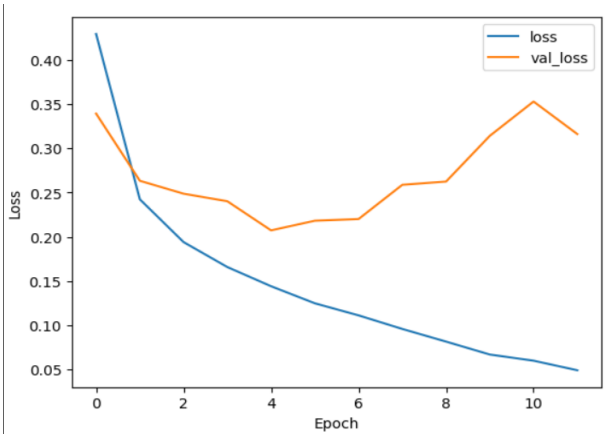



```
Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.91      0.84      4732
           1       0.90      0.76      0.83      5039

    accuracy                           0.84      9771
   macro avg       0.84      0.84      0.84      9771
weighted avg       0.85      0.84      0.84      9771
```

Visualizations

Graphical Representations: Accuracy and loss graphs gave visual insights into the model's learning progress, showing how model changes affected performance over time.





Conclusion:

Our project successfully showed the use of CNNs to identify deepfake pictures, with excellent accuracy and robust performance on a dataset of actual and modified photos. The approaches used, ranging from advanced picture preprocessing to thorough model training, have been beneficial in improving the model's detection skills.