SHIV NADAR
UNIVERSITY
CHENNAI

SHIV NADAR UNIVERSITY

KALAVAKKAM-603110

PROJECT REPORT

FakEYE: A Deepfake Detection System

BY: TRISHAA S

Third Year, B. Tech Artificial Intelligence and Data Science

OBJECTIVE:

Convincing manipulations of digital images and videos have been demonstrated for several decades using visual effects and the recent advancements in the deep learning domain have led to a dramatic increase in the realism of fake content and the accessibility in which it can be created.

Creation of these so-called Al-synthesized media (popularly referred to as deepfakes) using the already available Artificially intelligent tools is a simple task. But, when it comes to detection of these Deep Fakes, it is posing to be a major challenge. Till now there are no fool proof solutions which have been created. Already in the history there are many examples where the deepfakes are used as powerful way to create political tension, fake terrorism events, revenge porn, blackmail peoples etc. So, it becomes very important to detect these deepfake media. My DeepFake detection system works on the above problem and aims to detect and remove the circulation of DeepFake media from the internet.

KEYWORDS:

- Deepfake Detection
- Computer Vision
- Machine Learning
- Convolutional Neural Networks (CNN)
- Recurrent Neural Network (RNN)
- Generative Adversarial Networks (GANs)
- Image Analysis
- Video Forensics
- Face Recognition

SCOPE AND AVAILABILITY OF THE PROJECT:

There are many tools available for creating DeepFake media but hardly any tools are available for the DeepFake Detection. Hence this Deep Fake detection project can be used to identify and mitigate the spread of manipulated media.

- This project will focus on Deepfake detection involving the use of various techniques, such as image analysis, video forensics, machine learning, and artificial intelligence.
- A real-time deepfake detection system can be created that analyses content as it is being produced/shared which will drastically reduce the amount of DeepFake being circulated.
- It can also be integrated into major social media platforms like Facebook, Twitter, and Instagram to automatically identify and remove fake/manipulated content.

DATASET DESCRIPTION AND LINK:

For training and evaluating the deep fake detection system, there are multiple datasets available online. Among them I decided used the following dataset:

- DeepFake Detection Challenge (DFDC) dataset: The DFDC dataset consists of a diverse collection of real and deep fake videos. It includes various facial expressions, lighting conditions, and backgrounds to mimic real-world scenarios. The dataset can be accessed at:
  [DFDC Dataset] (https://www.kaggle.com/c/deepfake-detection-challenge/data)

RECENT BASE ARCHITECTURE:

- The proposed architecture is given below. In General, CNN architecture consists of both convolutional and pooling layers. Convolutional layers extract deep features from input images, whereas pooling layers reduce the dimensionality of the input feature maps.

- The input video/Image files are processed by a set of CNNs that extract spatial features. These CNNs capture low-level features such as textures, edges, and patterns, which are essential for differentiating between real and fake faces.

- After convolutional layers, the output from the CNNs is then fed into a sequence of bidirectional LSTM/GRU layers. This enables the model to analyse temporal dependencies and capture subtle dynamics that may be indicative of deep fake videos. This layer is called RNN layers

- Next the outputs from the CNNs and RNNs are combined in a fusion layer. This allows the model to leverage both spatial and temporal information effectively.

- Batch Normalization has been used after certain layers to stabilize the training process.

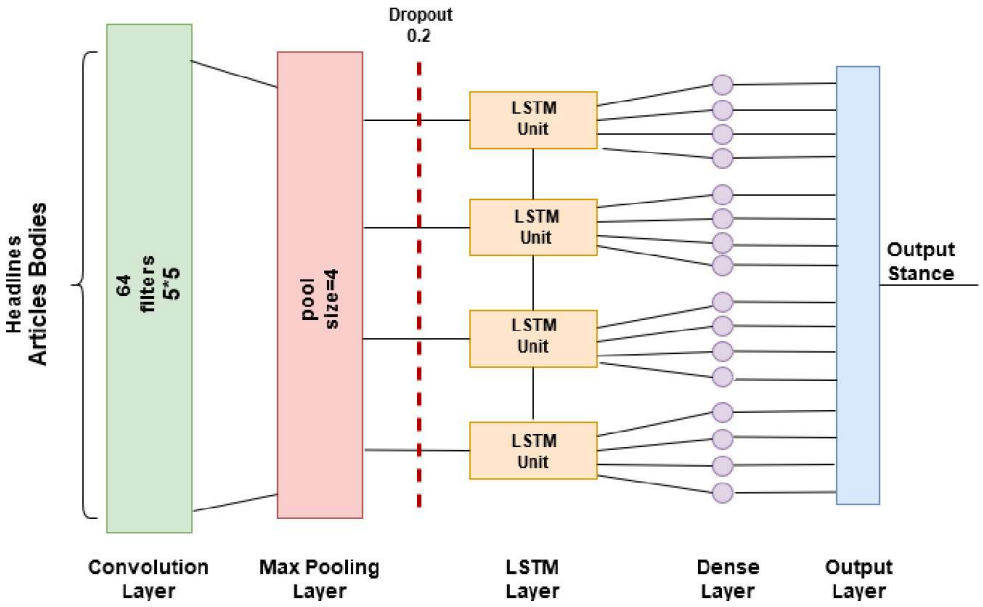
- Average pooling has decreased the dimensionality of the feature maps over the proceeding layers.

- All the feature maps produced as a result of both CNN and RNN are made into a one-dimensional array using a flattened layer and given as input to the fully connected layer.

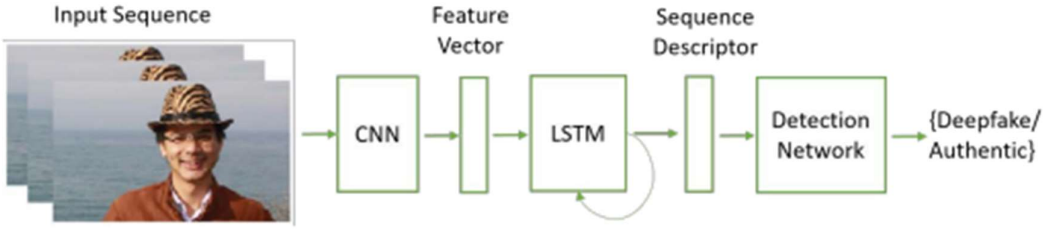
- After the fully connected layer, in the output layer the sigmoid function predicts the subsequent class (whether its deepfake or real) based on the input image.

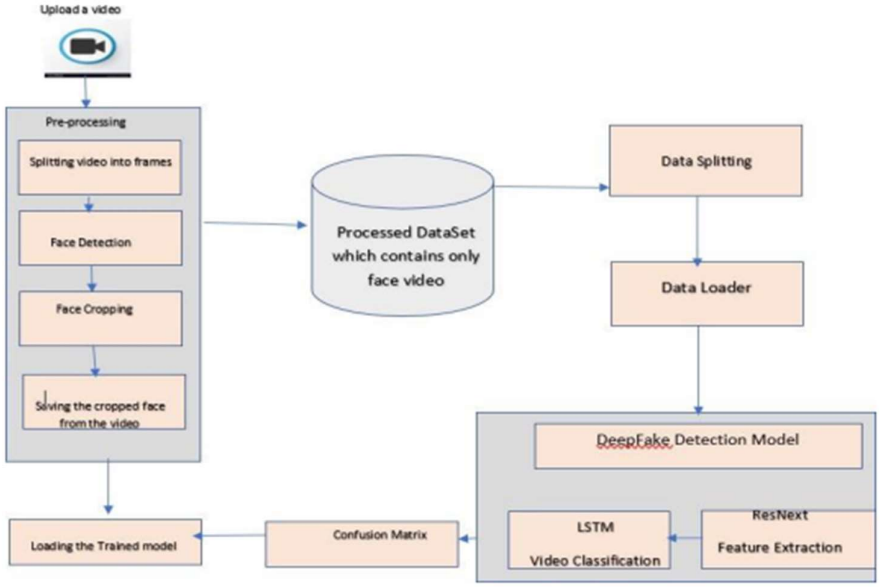
ARCHITECTURE DIAGRAM AND EXPLANATION:

BASIC NEURAL NETWORK ARCHITECHTURE:



PROPOSED BASIC IMPLEMENTATIONAL MODEL EXAMPLE:



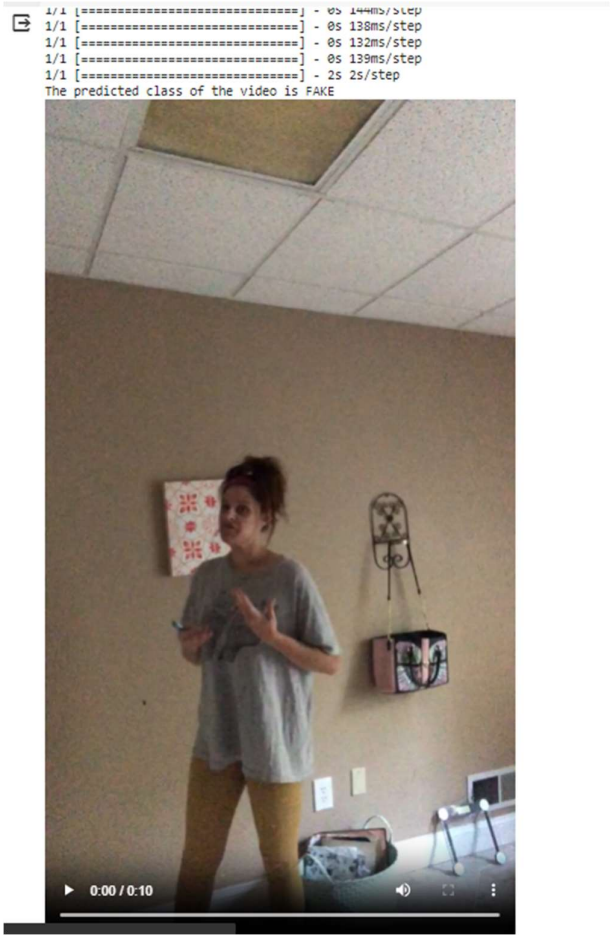


TRAINING THE BASE MODEL:

```
Epoch 1/15
45/45 [==============================] - 7s 50ms/step - loss: 0.6872 - accuracy: 0.7861 - val_loss: 0.6816 - val_accuracy: 0.8000
Epoch 2/15
45/45 [==============================] - 1s 15ms/step - loss: 0.6754 - accuracy: 0.8083 - val_loss: 0.6700 - val_accuracy: 0.8000
Epoch 3/15
45/45 [==============================] - 1s 18ms/step - loss: 0.6640 - accuracy: 0.8083 - val_loss: 0.6593 - val_accuracy: 0.8000
Epoch 4/15
45/45 [==============================] - 1s 23ms/step - loss: 0.6532 - accuracy: 0.8083 - val_loss: 0.6491 - val_accuracy: 0.8000
Epoch 5/15
45/45 [==============================] - 1s 18ms/step - loss: 0.6430 - accuracy: 0.8083 - val_loss: 0.6394 - val_accuracy: 0.8000
Epoch 6/15
45/45 [==============================] - 1s 14ms/step - loss: 0.6333 - accuracy: 0.8083 - val_loss: 0.6303 - val_accuracy: 0.8000
Epoch 7/15
45/45 [==============================] - 1s 16ms/step - loss: 0.6242 - accuracy: 0.8083 - val_loss: 0.6219 - val_accuracy: 0.8000
Epoch 8/15
45/45 [==============================] - 1s 14ms/step - loss: 0.6155 - accuracy: 0.8083 - val_loss: 0.6139 - val_accuracy: 0.8000
Epoch 9/15
45/45 [==============================] - 1s 15ms/step - loss: 0.6074 - accuracy: 0.8083 - val_loss: 0.6061 - val_accuracy: 0.8000
Epoch 10/15
45/45 [==============================] - 1s 14ms/step - loss: 0.5996 - accuracy: 0.8083 - val_loss: 0.5990 - val_accuracy: 0.8000
Epoch 11/15
45/45 [==============================] - 1s 16ms/step - loss: 0.5923 - accuracy: 0.8083 - val_loss: 0.5922 - val_accuracy: 0.8000
Epoch 12/15
45/45 [==============================] - 1s 14ms/step - loss: 0.5855 - accuracy: 0.8083 - val_loss: 0.5859 - val_accuracy: 0.8000
Epoch 13/15
45/45 [==============================] - 1s 14ms/step - loss: 0.5790 - accuracy: 0.8083 - val_loss: 0.5798 - val_accuracy: 0.8000
Epoch 14/15
45/45 [==============================] - 1s 14ms/step - loss: 0.5731 - accuracy: 0.8083 - val_loss: 0.5739 - val_accuracy: 0.8000
Epoch 15/15
45/45 [==============================] - 1s 15ms/step - loss: 0.5672 - accuracy: 0.8083 - val_loss: 0.5691 - val_accuracy: 0.8000
```

+ Code + Text

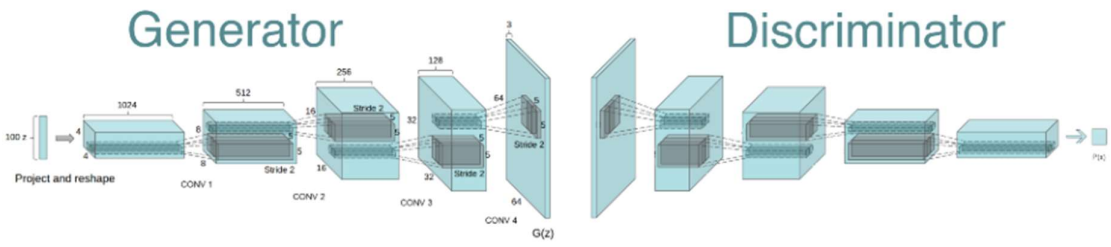
TESTING THE BASE MODEL:



REFINEMENTS:

Adversarial Training:

The term "adversarial training" is key to this approach. This training strategy involves a constant battle between the discriminator and the generator. The generator strives to generate increasingly convincing deepfake content, while the discriminator endeavours to sharpen its ability to distinguish between real and fake data. The generator's task becomes progressively challenging as it aims to produce deepfakes that are nearly indistinguishable from real data.



The accuracy obtained using GAN's is around 90%. Which is a huge increase from the previously seen 85%.

```
generator.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 8192)              827392
_________________________________________________________________
reshape_2 (Reshape)          (None, 4, 4, 512)         0
_________________________________________________________________
conv2d_transpose_5 (Conv2DTr (None, 8, 8, 256)         2097408
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 8, 8, 256)         0
_________________________________________________________________
batch_normalization_6 (Batch (None, 8, 8, 256)         1024
_________________________________________________________________
conv2d_transpose_6 (Conv2DTr (None, 16, 16, 128)       524416
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 16, 16, 128)       0
_________________________________________________________________
batch_normalization_7 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
conv2d_transpose_7 (Conv2DTr (None, 32, 32, 64)        131136
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 32, 32, 64)        0
_________________________________________________________________
batch_normalization_8 (Batch (None, 32, 32, 64)        256
_________________________________________________________________
conv2d_transpose_8 (Conv2DTr (None, 64, 64, 3)         3075
=================================================================
Total params: 3,585,219
Trainable params: 3,584,323
Non-trainable params: 896
_________________________________________________________________
```

```
discriminator.summary()
```

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 32, 32, 32)        1568
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 16, 16, 64)        32832
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 16, 16, 64)        0
_________________________________________________________________
batch_normalization_9 (Batch (None, 16, 16, 64)        256
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 8, 8, 128)         131200
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 8, 8, 128)         0
_________________________________________________________________
batch_normalization_10 (Batc (None, 8, 8, 128)         512
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 4, 4, 256)         524544
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 4, 4, 256)         0
_________________________________________________________________
flatten_2 (Flatten)          (None, 4096)              0
_________________________________________________________________
dropout_2 (Dropout)          (None, 4096)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 4097
=================================================================
Total params: 695,009
Trainable params: 694,625
Non-trainable params: 384
_________________________________________________________________
```

RESULT ANALYSIS:

Many deep learning models are able to achieve almost perfect level of accuracy (approx. 90%) when done on DeepFake detection challenge dataset. But the performance in the above case entirely relies on the type of dataset, the selected features, and the alignment between the train and test sets.

FUTURE SCOPE:

My aim in developing this deepfake detection project is to develop such a system which is able to perform with a high accuracy even when working with an unknown dataset.

In the future my system aims to demonstrate high accuracy and robustness in detecting deep fake videos and images, thus effectively mitigating the potential risks associated with the spread of manipulated content.

REFERENCE PAPERS:

- https://ieeexplore.ieee.org/document/10029095
- https://arxiv.org/pdf/1909.11573.pdf
- https://ieeexplore.ieee.org/document/9302547