Project Report: Deep Fake Detection in Videos

Introduction

The "Deep Fake Detection in Videos" project aims to develop a robust system for identifying deepfake videos using deep learning techniques. The project leverages Convolutional Neural Networks (CNNs) and a custom graphical user interface (GUI) built with Tkinter to provide a user-friendly platform for uploading and analyzing videos. The system is designed to classify videos as either real or fake, contributing to efforts in combating misinformation and enhancing digital media integrity.

Methodology

Data Collection

The dataset for this project comprises two categories of videos:

- **Real Videos**: Authentic videos without any manipulations.
- **Fake Videos**: Videos altered using deepfake techniques.

The videos were preprocessed and divided into frames, which were then used for training the deep learning model.

Data Preprocessing

The preprocess_video function was created to convert video files into a series of frames, resizing them to a consistent shape of 128x128 pixels. This preprocessing step ensures that the input data is standardized for model training.

Model Architecture

A 3D Convolutional Neural Network (CNN) was implemented to detect deepfakes. The model architecture includes multiple layers of 3D convolutions, pooling, flattening, dense layers, and a final sigmoid activation for binary classification. The model was compiled using the Adam optimizer and binary cross-entropy loss function.

```
model = Sequential([

   Conv3D(32, (3, 3, 3), activation='relu', input_shape=(30, 128, 128, 3)),

   MaxPooling3D((2, 2, 2)),

   Conv3D(64, (3, 3, 3), activation='relu'),

   MaxPooling3D((2, 2, 2)),

   Conv3D(128, (3, 3, 3), activation='relu'),

   MaxPooling3D((2, 2, 2)),

   Flatten(),

   Dense(128, activation='relu'),

   Dropout(0.5),

   Dense(1, activation='sigmoid') ])
```

Training and Evaluation

The dataset was split into training, validation, and test sets. The model was trained for 10 epochs, and its performance was evaluated on the test set. The trained model was then saved for future use.

```
history = model.fit(X_train, y_train, epochs=10, validation_data=(X_val, y_val))

test_loss, test_accuracy = model.evaluate(X_test, y_test)

model.save('deepfake_detection_model.h5')
```

User Interface

A Tkinter-based GUI was developed to allow users to upload videos and check their authenticity. The interface includes buttons for uploading and analyzing videos, displaying the first frame of the uploaded video, and showing the analysis result.

```
# Function to handle video upload
def upload_video():
    global video_frames

    # Clear the previous video frames
    video_frames = []
    # Get the selected video file
    video_path = filedialog.askopenfilename()

    # Preprocess the video frames
    frames = preprocess_video(video_path)
    video_frames.append(frames)
    video_frames = np.array(video_frames)

    # Display the first frame of the video
    if len(video_frames) > 0:
        frame_image = Image.fromarray(video_frames[0][0])
        frame_image.thumbnail((300, 300))  # Resize the image
        frame_photo = ImageTk.PhotoImage(image=frame_image)
        video_label.config(image=frame_photo)
        video_label.image = frame_photo
```

```
# Function to check if the video is real or fake
def check_video():
    global video_frames


    predictions = []
    # Make predictions using the model
    Deepfakedetection = load_model('deepfake_detection_model.h5')
    predictions = Deepfakedetection.predict(video_frames)


    # Post-process the predictions
    processed_predictions = (predictions > 0.5).astype(int)


    # Display the result
    result_label.config(
        text="The uploaded video is Real" if np.all(processed_predictions == 1) else "The uploaded
video is Fake")
```

Results

The trained model achieved satisfactory performance on the validation and test datasets, demonstrating its capability to distinguish between real and fake videos. The GUI allows users to interact with the model seamlessly, providing real-time analysis of video authenticity.

Conclusion

The "Deep Fake Detection in Videos" project successfully implemented a deep learning-based solution to detect deepfake videos. By integrating a CNN model with a user-friendly GUI, the system offers an accessible tool for identifying manipulated video content, thereby contributing to digital media verification efforts. Future work could involve expanding the dataset, refining the model, and enhancing the GUI for better user experience.