Deepfake Detection using Deep Learning

Aditya Kumar
Department of Computational and Mathematical Sciences
University of Toronto
Mississauga, ON L5L 1C6
aditya.kumar@mail.utoronto.ca

Harsh Patel
Department of Computational and Mathematical Sciences
University of Toronto
Mississauga, ON L5L 1C6
harshd.patel@mail.utoronto.ca

Isha Juneja
Department of Computational and Mathematical Sciences
University of Toronto
Mississauga, ON L5L 1C6
isha.juneja@mail.utoronto.ca

Nipun Rustagi
Department of Computational and Mathematical Sciences
University of Toronto
Mississauga, ON L5L 1C6
nipun.rustagi@utoronto.ca

Abstract

Advancements in generative AI have facilitated the creation of deepfakes, synthetic videos, and images generated using techniques such as generative adversarial networks and auto encoders, often for malicious purposes such as defamation. Creating effective ways to detect deepfakes is critical to preventing their potential misuse. This research presents a detection approach that uses a deep neural network with a Long Short Term Memory Recurrent Neural Network to identify if a person's face in a video has been modified(i.e. deepfake). First, we used a deep neural network, implemented directly within OpenCV, to extract the sub-section of a frame containing a face. We then employed a convolutional neural network to create a feature map of relevant and important features in each frame. We compile the feature maps for each frame and use them as input for a long short-term memory model, allowing us to analyze consecutive frames to identify irregularities over a short sequence of frames, which is indicative of video tampering. This approach identifies subtle signs of manipulation by first using the Convolutional Neural Network to find important features and then using the Long Short Term Memory model to analyze inconsistencies and visual distortions across video frames. We need this approach because deepfake alterations may be indiscernible to the human eye. The model is trained on the Deepfake Detection Challenge dataset provided by Kaggle benpflaum et al. [2019]. , containing authentic and tampered videos. Videos used in the training, validation and testing sets are 10 seconds long with a 720p resolution, running at 30 frames per second. Each deepfake contains

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

visual modifications, and all images contain images of people in them. The model will be evaluated on its ability to accurately predict if a video has been modified. Enhancing our ability to detect deepfakes contributes to the ongoing fight against misinformation and helps safeguard the integrity of digital media.

Introduction

Generative AI and deep learning have made considerable strides in the 2020s, and this allowed for the creation of advancements in models capable of editing videos with high quality - often indiscernible at first glance. These tampered videos, known as deepfakes, can be used maliciously in many ways, such as spreading misinformation by falsely suggesting information is coming from a credible source and creating defamatory videos that falsely depict individuals in damaging scenarios. Mitigating these issues is a step towards ensuring AI technology is not misused on a broad scale, and to combat misinformation. To protect against these attacks, it is important to develop strategies to identify falsified videos, since they often cannot be identified by humans due to the subtle nature of their editing.

Our strategy to identify deepfakes leverages existing deep learning libraries to develop a model that can identify if a video is tampered with. As input, the model takes a video that should be analyzed and outputs an identification of either authentic or tampered. A deep learning model is well-suited for deepfake classification due to its ability to handle complex pattern recognition within a sequence of frames. Furthermore, large data sets available for training ensure deep learning models can be scaled to train to robust levels, allowing deep learning models to identify deepfakes with greater accuracy than other methods.

The proposed model will make use of Python-based Open Computer Vision to take a video and convert it into a format usable by the mode. The model will first identify faces in individual frames of the video using a deep neural network. This information is then given to a convolutional neural network, which produces a feature map of all the relevant details in each frame. A Convolutional Neural Network (CNN) can identify intricate patterns in an image which makes it uniquely suited to identify fine-grained visual distortions that the naked eye cannot discern. Given a compiled set of feature maps for each video as input, a type of recurrent neural network known as a Long-Short Term Memory (LSTM) model is employed to capture relationships between consecutive frames. This allows us to look at chunks of frames at a time to identify distortions frame by frame, providing identifying features for video tampering. By leveraging deep neural networks and long short term memory models, this project aims to provide a powerful solution for identifying deepfakes, helping to protect individuals and organizations from the damaging effects of AI-driven media manipulation.

Background and related work

Recent studies have effectively combined CNNs and Recurrent Neural Networks (RNN) to detect deepfake videos. Typically, CNNs first extract important details from each video frame, such as facial features, while RNNs analyze the sequence of frames to detect any unusual patterns. For example, a combined approach using a CNN and an RNN has been proposed a model called InceptionV3 (Güera and Delp [2018]), where a CNN is used for feature extraction and an RNN, specifically an LSTM, is used for sequence processing.

Another CNN and LSTM approach proposed by Priya and Manisha [2024], improves detection by adding biometric cues, such as eye blinks and lip movements, which are often less consistent in deepfake videos. This approach was tested using the eye-blinking dataset, which was collected from the internet and is the first dataset specifically designed for eye-blinking detection. The experimental results discussed in the paper demonstrate the effectiveness of the proposed approach in identifying false images. We believe by focusing on these details and looking for unusual frame-to-frame movements, the model can better catch signs of manipulation.

Additionally, deepfake generation techniques for detection have also been explored. Al-Dhabi and Zhang [2021] proposed a method combining CNNs and RNNs to detect face-warping artifacts introduced by GANs. A pre-trained ResNeXt50_32x4d extracts features from video frames, while an LSTM captures temporal inconsistencies between frames. By simulating resolution inconsistencies

in affine face wrappings during training, this approach effectively identifies manipulation, offering a promising solution for improving deepfake detection.

Related works have produced encouraging results using a CNN and an LSTM approach, and we will also use a similar method. However, we will specifically focus on deepfake detection for human faces, utilizing a pre-trained CNN to extract facial features and an LSTM to determine if the video is a deepfake.

Data

The purpose of our model is to take a video as input and predict whether this video is a deepfake or real. Some noticeable features that indicate that a video is a deepfake include unnatural body movements, odd colouration, strange eye movements, awkward facial expressions, unnatural teeth/hair, and blurry visual alignment SoSafe

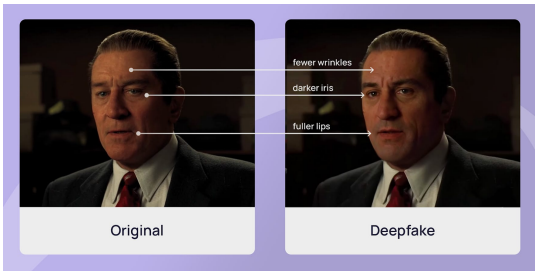

Figure 1: DeepFake of Robert De Niro of the purpose of digital de-aging Shamook [2020]

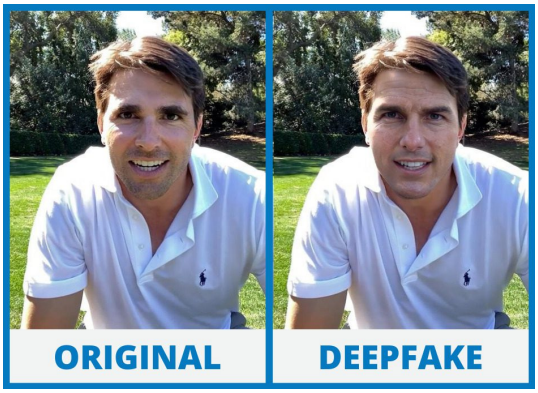

Figure 2: Deepfake of Tom Cruise on a different person Maverick [2023]

In Figure 1, a deepfake version of a de-aged Robert De Niro from the movie, The Irishman is quite similar to the original but has noticeable differences. The figure is taken from a frame of a deepfake video. In Figure 2, we show an example of how a deepfake can be used to mimic another person. These deepfake examples highlight the complexity that comes with these problems as each of the individual frames looks very realistic to the human eye. However, looking at the videos as a whole it is much more noticeable that a video is deepfake. Thus, while training our datasets, it is important for us to take each data point as a series of frames from the video rather than singular frames.

The dataset that we used to train, validate, and test our model is available on Kaggle: benpflaum et al. [2019]. The full dataset is 470 GB but it has been split into 50 smaller files of 10 GB each. The data is in the form of .mp4 files and also consists of a metadata.json that labels each of the files as real or fake and if the file is a fake, it indicates the name of the original file which was altered. We make sure to put the original and altered file in the same dataset (i.e. training set) to prevent the model from developing biases toward specific data points. Without loss of generality, we explored the data in the smaller file of 10GB. As shown in Figure 3, the count of fake videos is much greater than the

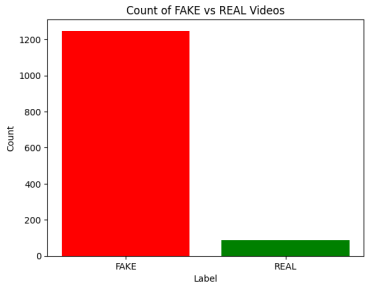

Figure 3: Comparing the count of FAKE vs REAL Videos in a dataset

number of real videos. After doing further inspection of the data, we realized that each real video has multiple fake counterparts. We also noticed that every fake video has its real counterpart in the same dataset and vice versa. In addition, every video in the dataset is 10 seconds long. For our model, we will aim to balance the distribution of original and altered samples to avoid over-representation of one type, enhancing the model's robustness and fairness across varied inputs.

Transforming The Data
We want to be able to transform .mp4 files into vector representations so we can train our model. To do this, we split .mp4 files into a series of frames using OpenCV, an open-source computer science library. Each frame produced will be a 3D array to represent the BGR pixels of the frame. It is a 3D array because the height and width represent the frame dimensions and the depth will be three due to the Blue, Green, and Red channels of BGR.

Since we are analyzing deepfakes regarding human faces, we detect the human faces in the frame and crop them out. To implement this, we use the pre-trained OpenCv face_detector model which returns a list of sections of the frames consisting of human faces. This is a Caffe model and we obtain the .prototxt (detailing the model architecture) from OpenCV [2018] and the model weights from sr6033 [2018]

Thus, for each video we obtain an array of cropped-out frames for each human face detected among the video frames, making it a 4D vector representation for every .mp4 file. Having cropped out frames for humans from each frame in the .mp4 file allows our model to analyze motion and changes over time, enabling the model to better catch features that indicate it to be a deepfake video. To ensure consistency, we ensure that the size of the vector representation for every .mp4 file is the same. If the size of the vector representation is less than the standardized size, the vector representation is padded with zeroes to increase the size. The zeroes will not affect the evaluation of the model but ensure that the sizes of the vectors are the same. If the face detector model is unable to detect any faces in the video, the input for that video will consist of all the frames in their entirety.

We use a subset of 20GB from the full dataset to train, validate, and test our model. We split the data such that 60% of the data was allotted to the training data set, 20% was allocated to the validation data set, and 20% to the test set in order to get the results for our model. The model will be trained to output the probability that the .mp4 file is real.

Since there are specific features of each video such as coloration and eye movements to notice while determining if a video is deepfake, we will use a CNN as the first layer of our model. The CNN model will take the series of frames of the .mp4 video as input and output a series of feature maps representing each frame. By creating a feature map for each frame, we will be able to learn and extract spatial structures like texture and objects. These feature maps can then be supplied to our model to evaluate/train on.

Model architecture

For our project, Deepfake Detection in Video Streams, a model architecture that combines a CNN and an LSTM. This hybrid approach is chosen to enhance the model's ability to detect subtle, frame-by-frame inconsistencies indicative of deepfake manipulation.

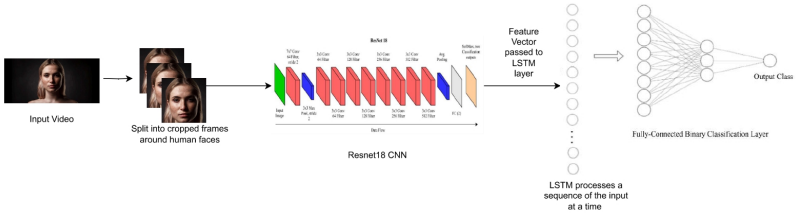

Figure 4: Model Architecture

Convolutional Neural Network (CNN) for facial feature extraction
The first component of the model is a CNN designed to process individual frames of the video. CNNs are well-suited for extracting spatial features due to their ability to capture complex patterns in images. In this model, we use ResNet-18, which is a CNN model that is 18 layers deep. We loaded a pre-trained version of the network that was trained on the ImageNet dataset. We remove the final fully connected (classification) layers of ResNet-18, keeping only the convolutional and pooling layers for feature extraction. In the context of deepfake detection, the CNN will be given preprocessed data that highlights relevant information by focusing on facial features as an input, and the CNNs will be used to identify frame-level anomalies such as texture inconsistencies, unnatural lighting, and blending artifacts, which are common indicators of manipulated content. Each frame in the video is independently processed by the CNN, resulting in a high-dimensional feature vector that captures spatial details, critical for deepfake detection. Therefore, the CNN architecture is based on a series of convolutional layers, and activation functions. Additionally, since the task is primarily based on frames (image processing), we utilize pooling layers Goodfellow et al. [2016] which work by summarizing small sections of each feature map produced by the convolutional layers by keeping only the most significant value in each section. This pooling helps reduce the overall computation for the model and helps focus on the most prominent features and patterns, such as edges, textures, and shapes, rather than the irrelevant fine-grained noise in each frame.

Long Short Term Memory (LSTM)for temporal feature extraction
While the CNN component captures facial features within each frame, identifying deepfakes would require an understanding of temporal patterns across consecutive frames. To address this need, we use an LSTM network, a type of RNN that processes sequences by "remembering" important information from earlier in the sequence and "forgetting" less relevant details. This ability to retain and pass information over time makes LSTMs particularly well-suited for identifying abrupt changes, unnatural movements, or temporal artifacts that are often present in deepfake videos. The LSTM we built has a single layer to process a sequence of features from the CNN used for feature extraction. The input and hidden layers have 512 nodes because it is the output size of the ResNet-18 model.

Classification layer
The final component of our proposed model is a fully connected layer that combines the temporal representations produced by the LSTM to generate a classification output. Specifically, after the LSTM processes the temporal sequence of feature vectors, a mean pooling operation is applied across the sequence length to condense the sequence into a single feature vector for each video. This aggregated representation is then passed through the fully connected layer, which maps it to a probability score for each class (e.g., real or fake).By integrating both spatial and temporal information through the CNN and LSTM components, the model aims to provide a reliable and robust deepfake detection solution

Results

After hyperparameter tuning which involved determining the optimal number layers in the LSTM model, our model is an LSTM with a single layer. During the validation phase, we implemented an early stopping technique, which allowed us to stop training when the improvement in the validation set stopped. This would allow us to get the best results and lighten the computational load. The following sections will discuss the metrics we used to evaluate our model performance quantitatively.

The accuracy we achieved using the model on the training set was 91.76% and on the validation set was 87.80%. The training loss value was 0.2177 and the validation loss value was 0.4123. During the

training process, early stopping was triggered on the last epoch indicating that validation accuracy would not improve any further. We achieved a high training and validation accuracy, shown in figure 5, as well as a low training and validation loss, indicating that the model can be used to classify deep fakes well and should generalize well as well. Once we were satisfied with this progress, we ran the model with the test set to evaluate our model performance.

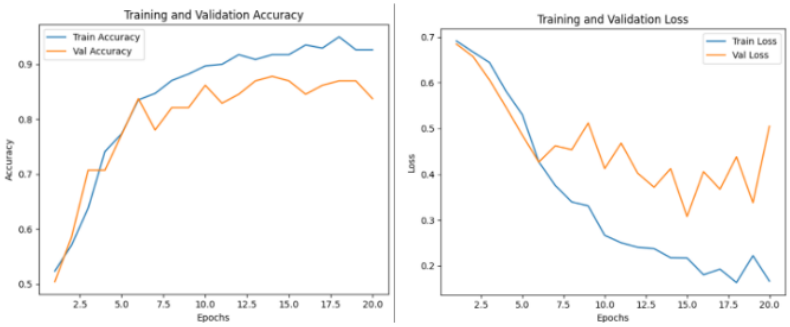

Figure 5: Training and Validation Accuracies and Losses

The test accuracy for this model is 87.72% and the test loss value is 0.2942. These are excellent results given the time and computational load constraints faced in this project. As discussed in the next section, this result is comparable to our baseline model and other models available online. These results can be visualized with the confusion matrix below from figure 6:

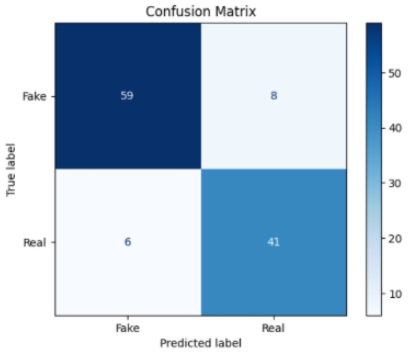

Figure 6: Confusion Matrix

Discussion

Interpretation of Model Performance Based
Using all of these technologies, the model achieved a validation set accuracy of 87.80%, a marginal improvement over the original iterations with 50% validation accuracy. The test accuracy for the model was given as 87.72%, with a loss value of 0.2942. A significant source of improvement was our updated method of computing the feature map from the CNN. Our original method had no concrete focus that we wanted CNN to focus on. Including a deep neural network to identify faces, and having the CNN focus on facial features allowed the model to make considerable improvements as this was the primary location of videographic distortions that the model looked for. The high accuracy scores limited the amount of false positives/negatives as the total number of incorrect guesses was minimized. We can interpret another dimension of our model performance using the confusion matrix. The results seen in the matrix are derived from the test set. The ratio of false positives to false negatives is very similar (6:8), indicating that the model is not biased toward one label or another. Although there are technically more false positives (8), we attribute this to a limitation in our test set size, as we would expect a larger margin of difference had there been a clear bias.

Methodology & Process
Prior to developing the final model, we had several previous iterations which we improved on.

These iterations were improved on by gauging the training and validation accuracies along with the confusion matrix. After establishing our initial setup and defining the initial model architecture (accessible in the previous project proposal), we trained the model for 10 epochs. The training loss was 0.6865, and the validation loss was 0.6925. The initial training and validation accuracy were 48.72% and 53.85%, respectively. These metrics serve as our baseline for comparison and cause us to rethink some of our approaches to the problem.

One such approach was in the data preprocessing. Our initial approach involved representing the video as a set of frames. Rather than storing the whole frame, we stored subsections of the frames focusing on human faces. The face detection strategy was employed using a Deep Neural Network to identify faces within each frame of the video. The Deep Neural Network was a pre-trained model provided by OpenCV for face detection, and we made no modifications to this model. This approach allowed our model to focus on identifying patterns and changes in the human faces which is where the deepfake was applied.

Another aspect of data preprocessing we looked at was the CNN. The CNN of our initial model was ResNet50 which includes more layers which increases the accuracy but also requires more computational power and memory. We ran our model with ResNet18, which uses less layers, and achieved the same training and validation accuracy (negligible difference). This motivated us to use ResNet18 because it provided the same results but required less computational power and memory.

Apart from data preprocessing, we made key changes to the model architecture. The initial model processes each frame one at a time, while the final model processes an entire batch of frames simultaneously, improving computational efficiency. In addition, the initial model does not apply average pooling after feature extraction, whereas the final model leverages pooling to condense spatial dimensions and focus on prominent features.

These adjustments in the final architecture led to improved performance in the training and validation accuracy. It also significantly reduced the number of false positives and false negatives. However, we noticed that our model overfit to the training data as the validation accuracy began to decline quickly. This led us to implement early stopping, which improved our post-training accuracy.

Comparison To Baseline Model
We found a baseline model within the Kaggle discussion page. Kirushikesh [2022]

We believed this to be a good baseline model because it followed a similar model architecture. Similar to our approach, it chose to divide each .mp4 file into frames and attempts to extract features from the frame. It uses the InceptionV3 CNN along with keras, a Python Neural Network API with Tensorflow to build a feature extractor for the frames. After extracting the features, it supplies this input to an RNN using GRU (Gated Recurrent Unit). The RNN also includes a dropout layer and utilizes the binary cross entropy loss function. This model also obtains its training, validation, and test datasets from the same data source.

Key Differences in our Model
We use the ResNet18 CNN which is similar to InceptionV3 except that it is more robust and allows for faster training time. We also implement an RNN using LSTM which includes an additional cell state that acts as a memory to store long-term dependencies along with three gates compared to the two gates of GRU. LSTM is known to perform better than GRU and better capture long-term relationships. We also have a more focused data preprocessing. This includes ensuring an equal number of fake and real videos in our training set which prevents our model from emphasizing fake videos over real videos. We also use pre-trained Face Detection to crop out every human face in the frame rather than looking at the whole frame. This enables our model to focus more on the human faces which is what the deepfake is applied on.

Limitations

The biggest limitation we faced was in computational capacity, wherein we did not have the time nor capability to train a model utilizing the entirety of the provided dataset. With additional time and computational capabilities, we anticipate a stronger ability to generalize the model as we will be able to make use of additional videos to train from. Additionally, having a more varied training set would allow us to make more accurate predictions for videos outside the biases of the Kaggle dataset such as the video size, length, and featured demographic.

The model was trained on a dataset with limited diversity in gender and race, leading to potential biases in its predictions. As a result, the model's performance may be less reliable for underrepresented demographics. For example, In the data used for training, validation and testing, approximately 80% of videos featured a female. Due to time constraints, acquiring additional datasets to create a more balanced representation was not feasible.

Furthermore, the model struggles with videos where facial features are obscured. Since the model relies heavily on facial features for classification, its performance is significantly reduced in such scenarios. Future work should address these limitations by sourcing more diverse datasets and developing methods to improve robustness when facial features are not prominently visible.

Ethical consideration

This research aims to mitigate the creation and spread of misinformation and protect individuals from identity-based attacks. Erroneous predictions in legal settings, such as flagging genuine videos as tampered with or vice versa, could harm reputations or lead to wrongful accusations. Collecting data for a deepfake detection model has several ethical considerations to make in data collection revolving around the privacy & consent of people in videos.

This model's ability to predict deepfake content makes it uniquely suited to making decisions in security and legal scenarios, which carry significant implications for erroneous judgments. Incorrectly classifying a real video as fake (false negative) actively discredits real and valid information. Conversely, incorrectly classifying a modified video as real and unmodified (false positive) allows an incorrect piece of information to be misused to support fraudulent claims, by representing falsified information as legitimate. Correct predictions (both true positives and negatives) ensure that information that is reliable and legitimate is portrayed as such, and misinformation can be identified and therefore minimized. To mitigate the issues arising from false positives and false negatives, we advise that this model be used to aid in decision-making rather than making final judgments.

Videos, both original and tampered, contain potentially sensitive information about individuals. The use of someone's videos to train this model requires that we ensure we have consent and accreditation for their works. Individuals should also be informed of exactly how their videos will be used. Furthermore, individuals must have minimal exposure to personally identifiable information to protect their privacy. For this reason, we use public datasets where explicit permission for machine learning training has been guaranteed. The use of a public dataset ensures that data privacy is maintained, as all data has been permitted for public sharing, negating the risk of identifying individual data that should be kept confidential. The model will also make no inferences based on personally identifying metadata. This model introduces the possibility of reinforcing biases for marginalized demographics due to imbalanced datasets. To this extent, training and testing will be done on diversified datasets, and limitations that reinforce biases will be communicated clearly.

By adopting these protocols and implementing these safeguards, this model can responsibly aid in the detection of deepfakes without compromising individual rights or contributing to social harm. The model is designed for use with human oversight and is designed only to provide supporting information and aid in decision-making.

Conclusion

Our final model was a neural network implementing strategies from a DNN for facial recognition, a CNN to identify visual distortions in images and an RNN to identify distortions in sequences of frames. This approach ensures that the model can identify meaningful patterns indicative of tampering, even if these distortions are imperceptible to the human eye. Through training and optimization of hyper-parameters, and the implementation of pooling and early stopping, the final test accuracy of the model was 87.72%. This performance underscores the robustness of our approach in detecting manipulations across a diverse range of scenarios and video qualities. While promising, further refinements are necessary to approach real-world deployment, where higher precision and recall rates are critical to minimize false predictions By advancing the state of deepfake detection, this research contributes to the ongoing battle against misinformation and malicious media manipulation, safeguarding the integrity of digital content and fostering trust in an era of rapid AI-driven innovation.

GitHub Repo

Please see attached the link to our Github Repository. This contains the code for running the initial, final, and baseline model. Link: https://github.com/Nipun0309/413_Group_Proj

References

Y. Al-Dhabi and S. Zhang. Deepfake video detection by combining convolutional neural network (cnn) and recurrent neural network (rnn). In *2021 IEEE International Conference on Computer Science, Artificial Intelligence and Electronic Engineering (CSAIEE)*, pages 236–241, 2021. doi: 10.1109/CSAIEE54046.2021.9543264.

benpflaum, B. G, djdj, I. Kofman, J. Tester, JLElliott, J. Metherd, J. Elliott, Mozaic, P. Culliton, S. Dane, and W. Kim. Deepfake detection challenge. https://kaggle.com/competitions/deepfake-detection-challenge, 2019. Kaggle.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

D. Güera and E. J. Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6, 2018. doi: 10.1109/AVSS.2018.8639163.

Kirushikesh. Deep fake detection on images and videos, 2022. URL https://www.kaggle.com/code/krooz0/deep-fake-detection-on-images-and-videos/notebook#Deep-Fake-Video-Classification.

Maverick. Video personalization | personalized video email marketing | maverick videos. https://www.trymaverick.com/blog-posts/are-deep-fakes-all-evil-when-can-they-be-used-for-good, 2023. Accessed: 2023-05-09.

OpenCV. Opencv github. https://github.com/opencv/opencv/blob/4.x/samples/dnn/face_detector/deploy.prototxt/, 2018.

A. S. Priya and T. Manisha. Cnn and rnn using deepfake detection. *International Journal of Science and Research Archive*, 11(02):613–618, 2024. doi: 10.30574/ijsra.2024.11.2.0460.

Shamook. De-aging robert deniro in the irishman [deepfake]. https://www.youtube.com/watch?v=dHSTWepkp_M, 2020. Accessed: 2023-05-09.

SoSafe. How to spot a deepfake. https://www.sosafe.com/en/how-to-spot-a-deepfake. Accessed: 11-8-2024.

sr6033. Face detection with opencv and deep neural network, 2018. URL https://github.com/sr6033/face-detection-with-OpenCV-and-DNN.