CSEC 520/620 Cyber Analytics and Machine Learning

DETECTING DEEPFAKES WITH MACHINE LEARNING

December 8, 2023

[1Vimal Seshadri Raguraman], [2]Venkata Janaki Vamsiram Gandham,
[3]Vijayavarman Arunasalam Harikrishnan, [4]Sree Alekhya Teerthala
Department of Computing Security
College of Computing and Information Sciences
Rochester Institute of Technology
[1]vr2969@g.rit.edu, [2]vg9377@rit.edu,
[3]va7457@g.rit.edu, [4]st6626@g.rit.edu

1. Abstract

In the realm of digital creativity and artificial intelligence, Deepfakes emerge as sophisticated and deceptive content, harnessing the power of AI to convincingly manipulate and mislead individuals. This project focuses on the development of a machine learning model, specifically tailored for detecting Deepfake images, employing a Convolutional Neural Network (CNN). The core challenge lies in distinguishing authentic content from manipulated ones, where the likeness of one person is seamlessly replaced with another, creating an illusion of reality.

2. Introduction

Deepfakes are highly creative and convincing digital content with Artificial Intelligence technology which is often generated to deceive people and manipulate them. It primarily involves replacing the likeness of one person with another, making it appear as though the newly generated content is real and unaltered. They are generated using Deep Learning techniques, particularly deep Neural Networks. A collection of images and videos of the person one wants to generate a Deepfake is collected and the model uses the data collected to identify tiny details and features like the shape of the face, the curve of the smile, etc.

A deep neural network, such as a Generative Adversarial Network (GAN) or Variational Autoencoder (VAE), can change one face into another. GANs consist of two neural networks, a generator, and a discriminator, engaged in a competitive training process. The generator strives to create realistic content, while the discriminator aims to distinguish between real and generated data. VAEs focus on learning a probabilistic representation of the input data, allowing for the generation of new, similar data points. The deep neural network dissects the images in the dataset, extracting intricate facial features, such as the contour of the face, the shape of the eyes, and the nuances of expressions. Through multiple layers of abstraction, the model learns to represent these features in a high-dimensional latent space. The deepfake model is trained using a loss function that quantifies the difference between the generated output and the real data. Training involves iterative adjustments of the model's parameters to minimize this loss, allowing the network to progressively improve its ability to generate realistic content. Once trained, the model can take an image or video of the source individual and transform it into the likeness of the target individual. The generator network creates synthetic images that, ideally, are indistinguishable from genuine content. Additional post-processing techniques, including color correction, frame blending, and shadow manipulation, are often applied to enhance the realism of the deepfake content.

These techniques address imperfections and inconsistencies, making the manipulated content more convincing to human observers. So, Deepfakes are a bit like computer-generated masks that can make someone's face look like someone else's face in pictures or videos. The rapid advancement in Generative Artificial Intelligence enabled the creation of Deepfakes more accessible and with the evolving nature of the Deepfake technology, there is an alarming need to develop efficient Deepfake detection models.

In Fig.1, we can observe a basic image reconstruction using the encoder and decoder architecture. An encoder is used to encode the latent factors for two different persons. Later, two separate decoders are built to reconstruct the first and the second image respectively. The encoded information is responsible for the subsequent reconstruction of the images and it is necessary to capture all the important information.

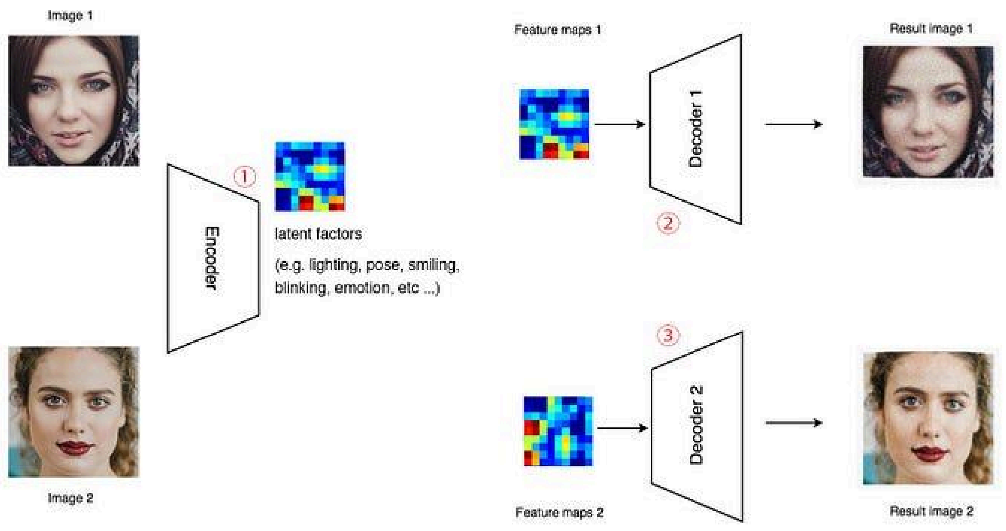


Fig. 1 Image Reconstruction using Encoder-Decoder architecture

In Fig.2, the encoder-decoder architecture is abused to generate a deep fake. Initially, we will capture the features of one image and then reconstruct it using another image decoder.

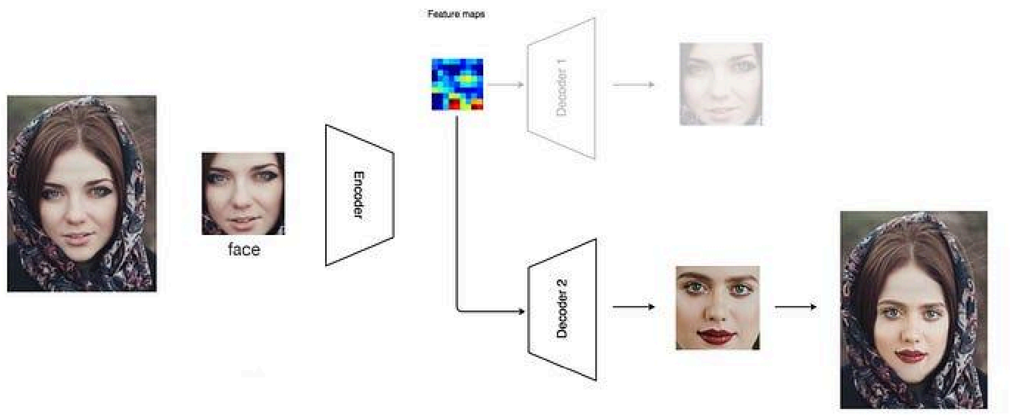


Fig. 2 Deep Fake Generation using Encoder-Decoder architecture

In recent years, Deepfake technology has made significant strides, with real-world implications spanning from political controversies to privacy concerns. One prominent example involves manipulated videos of political figures, disseminated to influence public opinion during pivotal events. Such instances have underscored the potential for Deepfakes to disrupt the political landscape and fuel misinformation campaigns. Additionally, the proliferation of non-consensual deepfake pornography has led to reputational harm and privacy violations for individuals, with instances reported globally since 2017.

Against this backdrop, the "State of Deepfakes 2020" report by Sensity sheds light on the escalating growth of harmful Deepfake videos, particularly those geared towards reputation attacks, emphasizing the urgent need for effective countermeasures and global awareness. The report identifies over 85,000 harmful Deepfake videos, doubling every six months since December 2018, with 93% designed for reputation attacks. The United States is the primary target for attacks on public figures. Notably, 7% of expert-crafted Deepfakes were created for comedy and entertainment. Now more than ever, it's crucial for us to recognize the significance of detecting Deepfakes and to take decisive steps towards developing robust solutions.

Detecting Deepfakes with the naked eye poses a considerable challenge due to the increasingly sophisticated and realistic nature of this technology. While there are some general indicators to watch for, such as unnatural facial movements, inconsistent lighting, or peculiar facial expressions, these visual cues are not foolproof. Deepfake algorithms have become adept at replicating subtle details, making it difficult for the untrained eye to distinguish between genuine and manipulated content, technological solutions are imperative. To enhance our ability to detect Deepfakes visually, it is crucial for individuals to stay informed about the latest advancements in Deepfake technology, remain vigilant for anomalies in content, and be cautious about the potential for misinformation. However, the most reliable and effective means of detection often involve specialized tools and algorithms designed to scrutinize digital content for subtle inconsistencies that elude casual observation.

Developing a detection system is the focus of this project, leveraging Convolutional Neural Networks (CNNs) for enhanced accuracy. CNNs are particularly adept at image recognition tasks, making them a fitting choice for discerning subtle visual cues indicative of deepfake manipulation. By harnessing the power of Deep Learning, this project aims to contribute to the ongoing efforts in fortifying digital media authenticity and mitigating the potential societal harm wrought by deceptive content.

3. Literature Review

Detecting these convincing but deceptive Deepfakes is a challenging task. To uncover these researchers and algorithm have been honed in on specific indicators that reveal inconsistencies between a Deepfake and genuine photo/video. One prominent sign, that might result from inadequate resolution in generated faces, is excessively smooth skin or a lack of skin details. A color mismatch between the real and synthesized faces, head position, temporal flashing, and visible portions of the actual face can be clues as well. These markers can be used to spot variations in facial expressions that point to the usage of DeepFake technology in the alteration of a video or image. It's crucial to remember that some of these signs could be hard to see, particularly if the DeepFake algorithm is really good. [1]

In the domain of Deepfake detection, many approaches have been employed, to distinguish digitized media from authentic content. These techniques encompass deep learning, blockchain-based approaches, and statistical methods. Deep learning-based models have consistently demonstrated superior performance compared to non-deep learning methods when it comes to detecting DeepFakes. [2]

In [3] the researchers proposed a model with the combination of convolutional neural network and transfer learning. They used the Celeb-DF dataset to train the neural network to detect any discrepancies in the facial features. They employed several image preprocessing techniques on the dataset and later passed it to the model for training. The resulting model will detect the real image from the fake one. One research delves into the complexity of detecting deepfake videos, particularly in the domain of social media where videos are often compressed. It puts forward a model that integrates a convolutional neural network with transfer learning techniques. Leveraging the Celeb-DF dataset for training, the model applies advanced image preprocessing to pinpoint inconsistencies in facial features that may indicate tampering. This innovative approach enhances the model's proficiency in distinguishing between authentic and altered images, addressing a significant challenge posed by high-quality deepfake algorithms. [6]

In [7] it introduces a temporal-aware pipeline utilizing a convolutional neural network (CNN) to extract features from video frames, and a recurrent neural network (RNN) to analyze temporal inconsistencies indicative of manipulation. The method showed competitive results with a simple architecture and was tested against a substantial set of deepfake videos from various sources. This approach offers a promising direction for effectively identifying deepfake manipulations using temporal analysis.

Another study outlines a process that involves the extraction and alignment of facial features from video frames, followed by the application of a fine-tuned CNN model for classification. The model is trained and tested using the "Celeb-DF: A New Dataset for DeepFake Forensics", showing promising results for detecting deepfakes, even with low-resolution images where deepfakes are more challenging to identify. The study highlights the significance of continual advancement in deepfake detection methods, given the rapid improvement in deepfake creation technologies.[8]

4. Implementation Details

The dataset was relatively large and diverse, containing a substantial number of videos to ensure a comprehensive evaluation of model performance. The videos varied in terms of resolution, frame rate, and quality, reflecting real-world conditions. To better understand and implement a detection model, we filtered the dataset into subsets and utilized a subset to train and test the model. The number of train and test videos which are considered in our dataset for developing the model can be seen below.

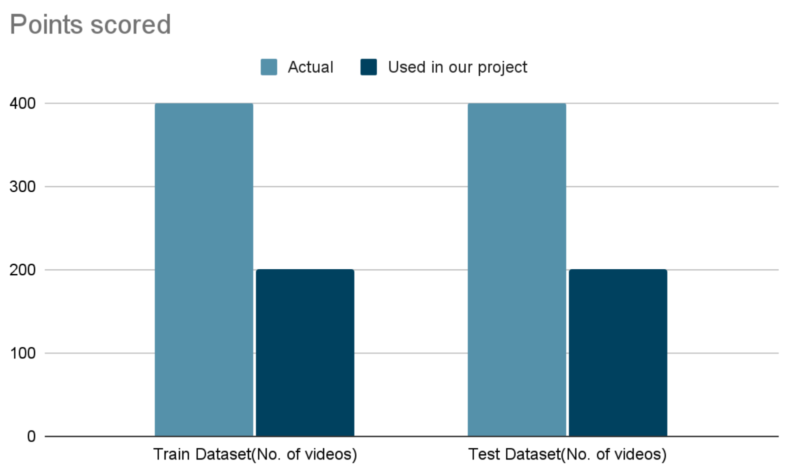


Fig. 3 Training and Testing samples taken from the overall dataset.

4.1 Data Preparation and Loading

The code will initially load the JSON file named **‘metadata.json’** which contains the information about the videos. It then converts this data into a CSV format with the file name as **‘output.csv’.** Using pandas, we can read this CSV file and view the content. It contains 4 columns which are **video_name, label, split, original.** The **video_name** specifies the name of the video, **label** indicates whether the video is fake or original, **split** will give us the information about the video being used as training data.

4.2 Video Frame Extraction

We defined a function called **extract_frames_limit** which is responsible for extracting the specified number of frames from the given input video. This function will iterate through the training and testing video folders and it will return the extracted frames from those videos.

```
from google.colab.patches import cv2_imshow

# Function to extract frames from videos with a limit

def extract_frames_limit(video_path, output_folder, max_frames=2):

    video_capture = cv2.VideoCapture(video_path)

    success, image = video_capture.read()

    count = 0

    while success and count < max_frames:

        frame_path = os.path.join(output_folder, f"frame_{count}.jpg")
        cv2.imwrite(frame_path, image)

        # Display the extracted frame

        cv2_imshow(image)

        success, image = video_capture.read()

        count += 1

    print(f"Generated {count} frames for video: {video_path}")


# Extract frames from training videos

for video_filename in os.listdir(train_videos_path):

    video_path = os.path.join(train_videos_path, video_filename)

    extract_frames_limit(video_path, train_frames_path, max_frames=2)


# Extract frames from testing videos

for video_filename in os.listdir(test_videos_path):

    video_path = os.path.join(test_videos_path, video_filename)

    extract_frames_limit(video_path, test_frames_path, max_frames=10)
```

4.3 Data Preprocessing

In this step, we performed some data preprocessing techniques on the extracted images before, they are used as training data for the model. As a part of this, we defined certain specifications for the images such as the dimensions. The **preprocess_image** function will take each extracted image as its input and it will normalize the image and later return the image array.

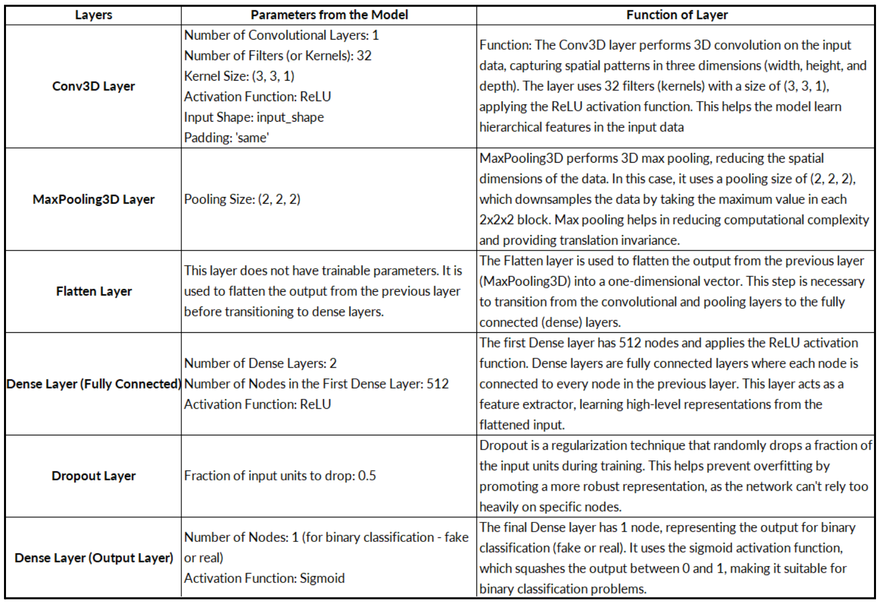
```
# Define image dimensions and other parameters

desired_height = 224

desired_width = 224

num_frames_per_video = 10  # Assuming you extracted 10 frames per video

# Function to load and preprocess an image

def preprocess_image(image_path):

    img = load_img(image_path, target_size=(desired_height, desired_width))
    img_array = img_to_array(img) / 255.0  # Normalize pixel values to [0, 1]

    return img_array
```

Now, once the preprocessing is done, we will split the dataset into training and testing sets using the **train_test_split** method. The training to testing ratio is 80:20.

```
# Split the data into training and testing sets

train_df, test_df = train_test_split(df, test_size=0.2, random_state=42)



# Now you can use the provided data preprocessing code

X_train, y_train = prepare_data(train_df, train_frames_path)

X_test, y_test = prepare_data(test_df, test_frames_path)
```

4.4 Convolutional Neural Network (CNN)

We made use of Convolutional Neural Networks (CNNs) as a means to analyze the frames and identify any manipulative patterns. Using tensorflow and keras, we constructed a 3D CNN model which consists of **convolutional layers**, **max-pooling, flattening and dense layers.** The Conv3D layers were used for spatiotemporal feature extraction and MaxPooling3D layers helped to downsample the data to preserve essential features. The details of the implemented CNN based model can be found in the figure below.



Layers	Parameters from the Model	Function of Layer
Conv3D Layer	Number of Convolutional Layers: 1 Number of Filters (or Kernels): 32 Kernel Size: (3, 3, 1) Activation Function: ReLU Input Shape: input_shape Padding: 'same'	Function: The Conv3D layer performs 3D convolution on the input data, capturing spatial patterns in three dimensions (width, height, and depth). The layer uses 32 filters (kernels) with a size of (3, 3, 1), applying the ReLU activation function. This helps the model learn hierarchical features in the input data
MaxPooling3D Layer	Pooling Size: (2, 2, 2)	MaxPooling3D performs 3D max pooling, reducing the spatial dimensions of the data. In this case, it uses a pooling size of (2, 2, 2), which downsamples the data by taking the maximum value in each 2x2x2 block. Max pooling helps in reducing computational complexity and providing translation invariance.
Flatten Layer	This layer does not have trainable parameters. It is used to flatten the output from the previous layer before transitioning to dense layers.	The Flatten layer is used to flatten the output from the previous layer (MaxPooling3D) into a one-dimensional vector. This step is necessary to transition from the convolutional and pooling layers to the fully connected (dense) layers.
Dense Layer (Fully Connected)	Number of Dense Layers: 2 Number of Nodes in the First Dense Layer: 512 Activation Function: ReLU	The first Dense layer has 512 nodes and applies the ReLU activation function. Dense layers are fully connected layers where each node is connected to every node in the previous layer. This layer acts as a feature extractor, learning high-level representations from the flattened input.
Dropout Layer	Fraction of input units to drop: 0.5	Dropout is a regularization technique that randomly drops a fraction of the input units during training. This helps prevent overfitting by promoting a more robust representation, as the network can't rely too heavily on specific nodes.
Dense Layer (Output Layer)	Number of Nodes: 1 (for binary classification - fake or real) Activation Function: Sigmoid	The final Dense layer has 1 node, representing the output for binary classification (fake or real). It uses the sigmoid activation function, which squashes the output between 0 and 1, making it suitable for binary classification problems.

Fig. 4 Different Layers in the model including parameters and the function of each layer.

5. Performance Discussion

Classification Report

	precision	recall	f1-score	support
0 (REAL)	0.88	0.90	0.89	100
1 (FAKE)	0.90	0.88	0.89	100
Accuracy	0.89			

Confusion Matrix

	0 (Predicted REAL)	1 (Predicted FAKE)
0 (Actual REAL)	90	10
1 (Actual FAKE)	12	88

The model displays a commendable level of accuracy in classifying videos as real or fake, with a slight edge in precision for detecting fake videos. The balance between precision and recall is evident in the F1-scores for both classes, which are very close to the model's overall accuracy. This suggests that the model is well-tuned to perform consistently across both classes. Further analysis could involve examining the model's performance across different subsets of the data or under varying conditions to ensure that it maintains its effectiveness in different scenarios.

6. Future scope

Building on our CNN-based approach for deepfake detection, future research will focus on integrating Siamese Network models to enhance comparative analysis between authentic and synthetic videos. We aim to optimize this architecture for improved feature extraction and fine-tune the output to provide a probabilistic authenticity score.

Efforts will be made to increase the model's processing speed and scalability to enable real-time detection. Additionally, we plan to extend our training datasets to encompass a wider range of deepfake techniques and conduct cross-dataset validations for better generalization. Ultimately, the goal is to apply these advancements not only for deepfake detection but also for broader digital content authentication, addressing critical needs in media and cybersecurity.

7. Conclusion

Our Deepfake Detection AI model demonstrates a promising accuracy of 83.9% on the test set, indicating a strong ability to discern real from manipulated content. The model's performance, as detailed in the confusion matrix, shows high reliability with a perfect recall for Deepfakes. However, it also indicates areas for improvement, particularly in reducing false negatives. Future enhancements will focus on integrating Siamese Network models for advanced comparison, leveraging 3D CNN for robust feature extraction, and improving the model's precision. This technology stands as a critical tool in the fight against digital misinformation, with broad applications in media, online platforms, and security.

8. References

[1] A. A. Maksutov, V. O. Morozov, A. A. Lavrenov and A. S. Smirnov, "Methods of Deepfake Detection Based on Machine Learning," 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus), St. Petersburg and Moscow, Russia, 2020, pp. 408-411, doi: 10.1109/EIConRus49466.2020.9039057.

[2] M. S. Rana, M. N. Nobi, B. Murali and A. H. Sung, "Deepfake Detection: A Systematic Literature Review," in IEEE Access, vol. 10, pp. 25494-25513, 2022, doi: 10.1109/ACCESS.2022.3154404.

[3] A. Karandikar, “Deepfake video detection using Convolutional Neural Network,” International Journal of Advanced Trends in Computer Science and Engineering, vol. 9, no. 2, pp. 1311–1315, 2020. doi:10.30534/ijatcse/2020/62922020

[4] M. Anderson, “The future of generative adversarial networks in deepfakes,” Metaphysic.ai, https://blog.metaphysic.ai/the-future-of-generative-adversarial-networks-in-deepfakes/ (accessed Nov. 30, 2023).

[5] S. Karagiannakos, “The theory behind latent variable models: Formulating a variational autoencoder,” AI Summer, https://theaisummer.com/latent-variable-models/

[6] A. Mitra, S. P. Mohanty, P. Corcoran and E. Kougianos, "A Novel Machine Learning based Method for Deepfake Video Detection in Social Media," 2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS), Chennai, India, 2020, pp. 91-96, doi: 10.1109/iSES50453.2020.00031.

[7] D. Güera and E. J. Delp, "Deepfake Video Detection Using Recurrent Neural Networks," 2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), Auckland, New Zealand, 2018, pp. 1-6, doi: 10.1109/AVSS.2018.8639163.

[8] Karandikar, Aarti. (2020). Deepfake Video Detection Using Convolutional Neural Network. International Journal of Advanced Trends in Computer Science and Engineering. 9. 1311-1315. 10.30534/ijatcse/2020/62922020.

[9] Thanh Thi Nguyen, Quoc Viet Hung Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, Thien Huynh-The, Saeid Nahavandi, Thanh Tam Nguyen, Quoc-Viet Pham, and Cuong M Nguyen. Deep learning for deepfakes creation and detection: A survey. Computer Vision and Image Understanding, 223:103525, 2022. [Fig.1 and Fig.2]

9. Appendix

Github Link: https://github.com/Vimal-Seshadri-Raguraman/CSEC.620-Deepfake-Detection.git

Dataset Link: https://www.kaggle.com/competitions/deepfake-detection-challenge/data