Abstract

The proliferation of information on the internet has revolutionized the way we access public utility services and entertainment. Social networks and online media have ushered in a new era of timely communication and convenient information dissemination. However, this digital landscape has also opened up opportunities for the malicious spread of fake content, particularly through the emergence of deepfake technology. This paper presents a comprehensive review of recent studies on deepfake content detection using deep learning-based approaches.

However, as the digital realm expands its horizons, it has also exposed vulnerabilities. The same channels that empower the dissemination of accurate information and the fostering of global connections have inadvertently become conduits for the propagation of deceit and misinformation. The insidious spread of fake content, often engineered with malicious intent, represents a grave challenge in the digital age. We aim to expand the existing body of research by systematically categorizing various methods for detecting counterfeit content.

List of Figures and Tables:

List of Abbreviations and Acronyms

- ADAM: A method for stochastic optimization
- ReLU: Rectified Linear Unit
- H.264: A video compression codec
- Keras: A deep learning framework
- GPU: Graphics Processing Unit
- CNN: Convolutional Neural Network

1. Introduction:

1.1 Background of the project:

1. The Proliferation of Information in the Digital Age:

The advent of the internet has brought about a paradigm shift in the way we access public utility services, communicate, and consume entertainment. It has unleashed a wealth of opportunities for individuals and organizations to connect, share, and access information with unprecedented ease and speed. Social networks, online media platforms, and digital communication tools have ushered in a new era of timely and convenient information dissemination.

2. Digital Revolution and Its Benefits:

The digital revolution has undeniably improved our lives in numerous ways. It has streamlined access to essential services, from online banking and e-commerce to healthcare and education. People can now connect with loved ones across the globe, collaborate on projects, and access a vast repository of knowledge with just a few clicks. The convenience and efficiency of this digital landscape have made daily tasks more manageable and have expanded opportunities for global communication and collaboration.

3. The Dark Side of the Digital Landscape:

However, the digital landscape is not without its challenges. The same technologies that have brought about these benefits have also created vulnerabilities. One of the most concerning aspects is the malicious spread of fake content. The internet is rife with misinformation, disinformation, and deceptive content, often with the intent to mislead or manipulate individuals and societies.

4. Deepfake Technology as a Threat:

At the forefront of this issue is the emergence of deepfake technology. Deepfakes are highly convincing forged media, often in the form of manipulated videos, that use artificial intelligence and machine learning to seamlessly replace one person's face with another's. This technology has been used for various purposes, from creating deceptive content for

entertainment to more nefarious objectives, including spreading false information and propaganda.

5. The Need for Deepfake Content Detection:

The proliferation of deepfake content poses a significant challenge to online platforms, media outlets, and individuals seeking to distinguish truth from falsehood. The consequences of failing to identify deepfakes can be severe, affecting public trust, information integrity, and even national security. As a result, there is a pressing need for robust and reliable methods of detecting deepfake content.

6. A Comprehensive Review of Deepfake Detection:

This paper aims to address this need by presenting a comprehensive review of recent studies on deepfake content detection. The objective is to contribute to the growing body of research dedicated to countering the spread of counterfeit content. The focus is on deep learning-based approaches, which leverage the power of artificial intelligence and neural networks to identify deepfake media.

7. Systematic Categorization of Detection Methods:

To provide a structured and organized understanding of the field, the paper systematically categorizes various methods for detecting counterfeit content. By classifying and summarizing the existing approaches, it offers a valuable resource for researchers, practitioners, and stakeholders seeking to navigate the complex landscape of deepfake detection.

In conclusion, the digital age has brought about incredible opportunities and conveniences, but it has also exposed us to new challenges in the form of deepfake technology. Addressing these challenges requires a concerted effort, and this paper's contribution lies in its in-depth review and categorization of deepfake detection methods, which ultimately enhances our ability to safeguard the integrity of digital content in the age of information proliferation.

This extended section provides a more detailed exploration of the impact of the digital age on our lives, the challenges it has brought, and the significance of deepfake content detection in addressing those challenges. It highlights the need for systematic categorization of detection methods and the contribution of your paper to this important field of research.

The internet's rapid dissemination of information has transformed the way public utility services and entertainment are accessed by users. Online media and social networks have revolutionized communication and information availability, offering both convenience and speed. However, this digital landscape has also given rise to new challenges, including the widespread dissemination of fake content, notably through the use of deepfake technology.

1.1.1 Deepfake

Deepfake technology, which emerged in late 2017, is a method aimed at replacing a person's face in a video with another person's face. Initially, it was used primarily for generating adult content with face-swapping capabilities. Subsequently, this technique evolved, leading to the development of the user-friendly application known as FakeApp.

The fundamental concept behind deepfake involves training two autoencoders simultaneously. The design of these autoencoders can vary depending on factors such as output size, training time, quality expectations, and available computing resources. In a traditional autoencoder setup, there is an encoder network followed by a decoder network. The encoder's role is to reduce the dimensionality of input data by encoding it into a smaller set of variables, while the decoder aims to generate an approximation of the original input using these variables. This optimization process typically involves minimizing the difference between the input and its generated approximation, often measured using the L2 distance.

In the case of deepfake, the original autoencoder is fed with images of 64 × 64 × 3 resolution, translating to 12,288 variables. These images are encoded into 1024 variables and then reconstructed to match the input's size. The deepfake generation process involves gathering aligned faces of two distinct individuals, A and B. Two autoencoders, EA and EB, are separately trained to reconstruct faces from the corresponding datasets of facial images. A crucial element is that the encoding part of both autoencoders shares weights, while their respective decoders remain separate. This enables the shared encoder to encode general information about illumination, position, and facial expression, while each decoder focuses on reconstituting specific facial features and details for person A or B.

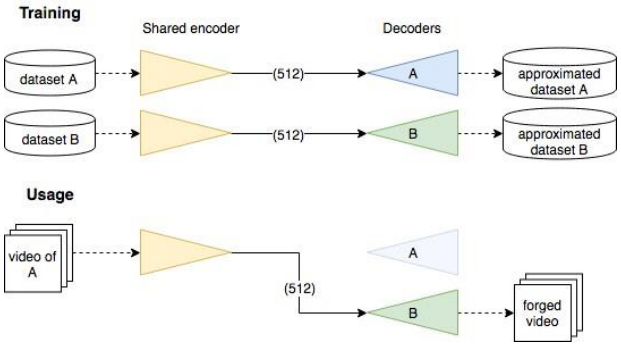


Figure 1. Deepfake theory. Top: the training segments with the yellow shared encoder. In the use section at the bottom, photos of A are decoded using a decoder for B.

Practically, this approach yields impressive results, contributing to the technique's popularity. The final step in generating a deepfake involves taking the target video, extracting and aligning the target face from each frame, using the modified autoencoder to generate a new face with consistent illumination and expression, and then seamlessly integrating it back into the video.

Nevertheless, deepfake technology is not without its flaws. Challenges may arise during face extraction and reintegration, particularly when dealing with face occlusions, resulting in frames with no facial reenactment, blurred regions, or duplicated facial contours. These technical errors can be mitigated through more advanced network architectures. Furthermore, autoencoders tend to struggle with fine detail reconstruction due to the compression of input data into a limited encoding space, resulting in a slightly blurred appearance. Enlarging the encoding space may improve fine detail approximation but at the expense of realism, as the output face begins to resemble the input face, unintentionally conveying morphological information.

Figure 2 : An illustration of a Deepfake-created picture may be seen on the right. Notice how the fabricated face loses the original's expressiveness.

1.2 Problem statement:

The emergence of deepfake generation techniques has raised serious concerns about their potential to manipulate public opinion. In response to these challenges, this paper provides a concise overview of recent advancements in deep learning-based approaches for detecting counterfeit content, with a particular focus on deepfake videos manipulation.

1.3 Objectives and scope of the project:

- To review recent studies on deepfake content detection.
- To categorize various methods for detecting counterfeit content.
- To propose an efficient neural network approach for detecting manipulated facial videos.
- To evaluate the proposed approach's performance.

1.4 Significance of the project:

The project aims to contribute significantly to the ongoing efforts to combat the proliferation of fake content in the digital era, particularly in the context of deepfake videos.

1.5 Brief overview of the methodology:

The methodology involves reviewing deepfake technology, proposing neural network architectures for detection, conducting experiments on a created dataset, and analyzing the results.

2. Literature Review:

1. "Face Forensics: A Large-Scale Video Dataset for Forgery Detection in Human Faces"

Summary: In this paper, the authors present the FaceForensics dataset, a significant contribution to the field of forgery detection in videos. The dataset contains a wide range of videos where human faces have been manipulated using various techniques, including Face2Face. The dataset serves as a valuable benchmark for training and evaluating forgery detection methods, allowing researchers to assess their models' performance under realistic conditions.

2. "Detecting Deepfake Videos in the Wild Using a Convolutional Neural Network"

Summary: This study addresses the challenge of detecting deepfake videos distributed across social media platforms and the internet. The authors propose a Convolutional Neural Network (CNN) architecture designed for this specific purpose. The CNN-based approach is effective in identifying deepfake videos, and the research sheds light on the technical aspects of deepfake detection, which include distinguishing manipulated videos from authentic ones.

3. "DeepFakes: A New Threat to Face Recognition? Assessment and Detection"

Summary: This research explores the vulnerability of face recognition systems to Deepfake technology. It examines the potential risks posed by Deepfakes and proposes a method for detecting them. The paper contributes insights into the challenges faced by face recognition systems in the presence of manipulated videos and suggests countermeasures. The work highlights the importance of addressing security concerns related to biometric authentication systems.

4. "MesoNet: A Compact Facial Video Forgery Detection Network"

Summary: MesoNet is introduced as a compact neural network tailored for detecting facial video forgery, with a specific focus on Deepfake and Face2Face techniques. The paper emphasizes the significance of mesoscopic image properties for forgery detection. MesoNet's architecture is designed to capture these properties effectively, making it a valuable addition to the toolkit for combating manipulated video content.

5. "Face2Face: Real-Time Face Capture and Reenactment of RGB Videos"

Summary: While not primarily focused on detection, this paper introduces the Face2Face technique, a pioneering work in real-time facial reenactment in RGB videos. It demonstrates the capabilities of face manipulation techniques that can be used for creating deceptive videos. Understanding such techniques is crucial for developing detection algorithms that can identify manipulated content.

6. "Video Manipulation Detection with Deep Learning"

Summary: Hany Farid provides an overview of the challenges and approaches in video manipulation detection, with a focus on deep learning methods. The paper discusses the importance of detecting manipulated videos in the era of fake news and presents insights into the role of deep learning in addressing this issue.

7."Deep Learning for Deepfakes Creation and Detection"

Summary: This paper explores both the creation and detection of deepfake videos using deep learning techniques. It highlights the rapid advancements in deepfake technology and the importance of developing robust detection methods. The authors discuss various neural network architectures for deepfake detection.

3. Methodology:

3.1 Detailed explanation of the methods used in the project:

In the methodology section of your project, you should provide a detailed explanation of the methods used for detecting face tampering in videos with a focus on deep learning approaches. Based on the information you provided earlier, here's a structured breakdown of what to include:

3.1.1 Data Collection and Preprocessing:

As far as we are aware, there is currently no existing dataset that compiles videos generated using the Deepfake technique. Consequently, we took the initiative to create our own dataset.Training autoencoders for the forgery task is a time-intensive process, taking several days with conventional processors to produce realistic results. Moreover, this process is limited to handling two specific faces at a time. To ensure a diverse range of faces for our dataset, we opted to source videos readily available to the general public on the internet.

As a result, we gathered 175 video clips of manipulated content from various online platforms. These videos vary in duration, ranging from two seconds to three minutes, and maintain a minimum resolution of 854 × 480 pixels. All videos are compressed using the H.264 codec, albeit at different compression levels, aligning with real-world analysis conditions.

To balance the distribution of faces within the dataset, we selected frames for extraction from each video in proportion to the number of camera angle and illumination changes affecting the target face. As a reference, approximately 50 faces were extracted from each scene.To enhance dataset diversity, we augmented it with real face images obtained from various online sources, maintaining the same resolutions. Subsequently, we conducted manual reviews to rectify any misalignment or incorrect face detection. We made efforts to maintain a balanced distribution of high-resolution and low-resolution images in both classes to prevent bias in the classification task

3.2. Proposed Method Deep Learning Model Architecture:

This section explores various effective strategies for addressing the challenges posed by Deepfake technology. It becomes evident that a single network approach is not efficient in tackling these issues comprehensively. However, due to the shared characteristics of these falsifications, employing identical network structures for both problems can yield favorable outcomes.

To detect manipulated facial videos, we propose an approach situated at the mesoscopic level of analysis. In a compressed video context, where image noise is significantly degraded, microscopic analyses based on image noise prove ineffective. Similarly, at a higher semantic level, distinguishing manipulated images, especially those featuring human faces, poses a challenge for the human eye. Therefore, our solution adopts an intermediate approach, utilizing a deep neural network with a minimal number of layers.

Among the various architectures tested, two particular designs have demonstrated exceptional classification scores, featuring a low-level representation and surprisingly few parameters. These architectures leverage well-established networks commonly used for image classification, incorporating layers of convolutions and pooling for feature extraction, along with a dense network for classification. Their source code is readily accessible online for reference .

3.2.1. Model architecture

Our experimentation initially commenced with intricate architectures and progressively evolved towards a more streamlined design, achieving equivalent results with enhanced efficiency.

This network initiates with a sequence of four layers, comprising successive convolutions and pooling operations, followed by a dense network featuring a single hidden layer. In order to enhance generalization, the convolutional layers employ ReLU activation functions to introduce non-linearities, incorporate Batch Normalization to regulate their output and mitigate the vanishing gradient issue, and implement Dropout within the fully-connected layers to bolster robustness.

Figure illustrates the architecture of the network, with detailed information regarding layers and parameters provided within the boxes, and the output sizes specified adjacent to the arrows.

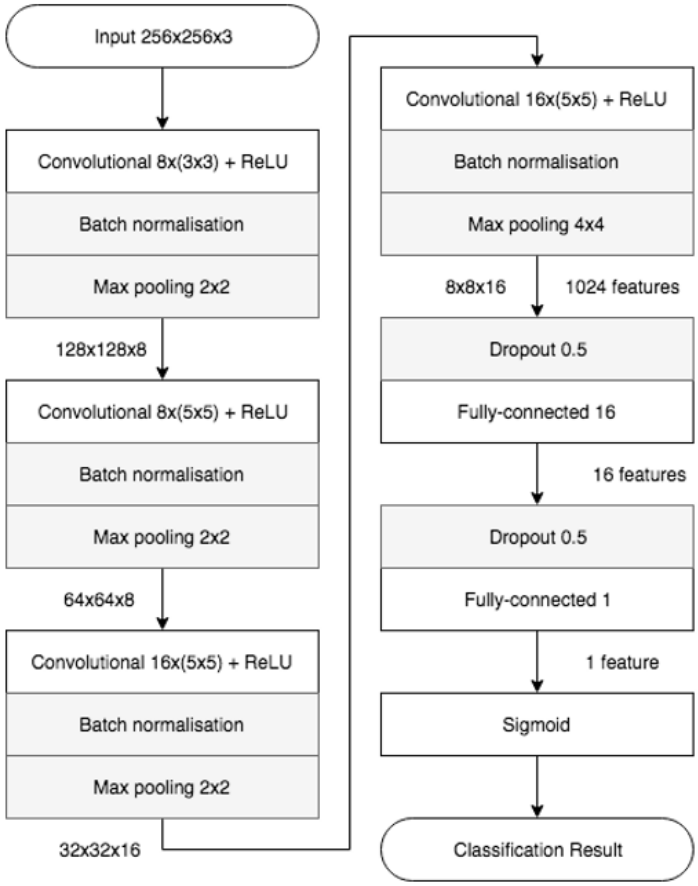


Figure3 Network architecture. In the boxes, layers and parameters are shown, and the output sizes are next to the arrows.

3.3. Classification Setup.

Let X represent the input set, and Y represent the output set, where the random variable pair (X, Y) takes on values within X × Y. Additionally, let f denote the prediction function of the selected classifier, which maps values in X to the action set A. The objective of the chosen classification task is to minimize the error E(f) = E[l(f(X), Y)], where l represents the loss function.For the implementation of both networks, Python 3.5 was utilized, along with the Keras module. Weight optimization for the network involved processing batches of 356 images, each with dimensions of 256 × 256 × 3, using the ADAM optimizer with default parameters ($\beta1 = 0.9$ and $\beta2 = 0.999$).

The initial learning rate of 10^-3 was gradually reduced by a factor of 10 every 1000 iterations until it reached 10^-6. To enhance generalization and robustness, random transformations were applied to input batches, encompassing zoom, rotation, horizontal flips, brightness adjustments, and hue variations.

Set	forged class	real class
Deepfake training	10,259	6026
Deepfake testing	2051	1506

Table 1. Each class's cardinality in the datasets under study.

4. Implementation:

4.1 Details of how the project was implemented with Code snippets :

1. Import necessary libraries:

- `os`: Operating system module for file and directory operations.

- `cv2`: OpenCV library for image processing.

- `numpy`: Library for numerical operations.

- `sklearn.model_selection`: For splitting the dataset into train and test sets.

- `tensorflow.keras.preprocessing.image`: For image preprocessing functions.

- `tensorflow.keras.layers` and `tensorflow.keras.models`: For building the neural network.

2. Define the model using a sequential approach:

- It's a simple convolutional neural network (CNN) model with four convolutional layers, max-pooling layers, and two dense layers.

```
In [ ]: def Detect(input_shape):
            model = models.Sequential()

            model.add(layers.Conv2D(8, (3, 3), padding='valid', input_shape=input_shape, activation='relu'))
            model.add(layers.MaxPooling2D(pool_size=(2, 2)))

            model.add(layers.Conv2D(8, (3, 3), padding='valid', activation='relu'))
            model.add(layers.MaxPooling2D(pool_size=(2, 2)))

            model.add(layers.Conv2D(16, (3, 3), padding='valid', activation='relu'))
            model.add(layers.MaxPooling2D(pool_size=(2, 2)))

            model.add(layers.Conv2D(16, (3, 3), padding='valid', activation='relu'))
            model.add(layers.MaxPooling2D(pool_size=(2, 2)))

            model.add(layers.Flatten())
            model.add(layers.Dense(16, activation='relu'))
            model.add(layers.Dense(1, activation='sigmoid'))

            return model
```

3. Prepare the dataset:

- Load images from a directory (`dataset_dir`), where each subdirectory represents a class (e.g., "Real" and "DeepFake").

- Load images, resize them to (128, 128), convert them to NumPy arrays, and normalize pixel values to the range [0, 1].

- Append the images to the `data` list and their corresponding labels to the `labels` list.

4. Split the dataset:

- Split the data into training, validation, and test sets using `train_test_split` from Scikit-Learn. Encode the class labels using `LabelEncoder` to convert them to numeric values.

```
In [ ]:
data = []
labels = []

dataset_dir = "./data2/"


for label in os.listdir(dataset_dir):
    label_dir = os.path.join(dataset_dir, label)


    if not os.path.isdir(label_dir):
        continue

    for filename in os.listdir(label_dir):
        img_path = os.path.join(label_dir, filename)

        if not (img_path.endswith('.jpg') or img_path.endswith('.png')):
            continue


        img = load_img(img_path, target_size=(128, 128))
        img = img_to_array(img)
        img = img / 255.0


        data.append(img)
        labels.append(label)


data = np.array(data)
labels = np.array(labels)
```

5. Compile the model:

- Set the optimizer (Adam), loss function (binary cross-entropy), and evaluation metric (accuracy).

6. Train the model:

- Fit the model to the training data with validation data to monitor the training process. It trains for 20 epochs with a batch size of 32.

```
In [ ]: model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
history = model.fit(train_data, train_labels_encoded, epochs=20, batch_size=32, validation_data=(val_data, val_labels_encoded))
```

7. Save the trained model as 'deepfake_detection_model.h5'.

```
In [ ]: model.save('deepfake_detection_model.h5')
```

8. Evaluate the model on the test set:

- Calculate the test loss and test accuracy.

```
In [ ]: test_loss, test_accuracy = model.evaluate(test_data, test_labels_encoded)
        print(f"Test Loss: {test_loss:.4f}")
        print(f"Test Accuracy: {test_accuracy:.4f}")
```

9. Plot training and validation loss and accuracy over epochs using Matplotlib.

```
In [ ]: import matplotlib.pyplot as plt

        training_loss = history.history['loss']
        validation_loss = history.history['val_loss']
        training_accuracy = history.history['accuracy']
        validation_accuracy = history.history['val_accuracy']


        epochs = range(1, len(training_loss) + 1)
```

10. Generate a confusion matrix for evaluating model performance on the test set:

- Predict labels for the test data and convert probabilities to binary labels (0 or 1).

- Compute the confusion matrix using `confusion_matrix` from Scikit-Learn.

- Visualize the confusion matrix using a heatmap with seaborn.

```
In [ ]: test_predictions = model.predict(test_data)
        test_predictions = (test_predictions > 0.5).astype(np.int32)  # Convert probabil


        conf_matrix = confusion_matrix(test_labels_encoded, test_predictions)


        plt.figure(figsize=(6, 4))
        sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', cbar=False,
                    xticklabels=['Real', 'DeepFake'], yticklabels=['Real', 'DeepFake'])
        plt.xlabel('Predicted')
        plt.ylabel('True')
        plt.title('Confusion Matrix')
        plt.show()
```

4.2 Testing procedures and results:

. Here's a step-by-step description of the testing procedures :

1. Model Loading: Before testing, it's assumed that you've already trained and saved the model as 'deepfake_detection_model.h5' using the training code. You should ensure that the saved model exists in your working directory.

2. Test Data Preparation:

The test data is loaded from the `test_data` array, and the corresponding labels are loaded from the `test_labels` array.

These test data and labels were split from the original dataset during the dataset preparation phase.

3. Label Encoding:

- Label encoding is performed using the same `LabelEncoder` instance that was used during training. This ensures that the class labels are encoded consistently between training and testing.

- The encoded labels are stored in the `test_labels_encoded` array.

4. Model Evaluation:

The following lines of code evaluate the model on the test data:

```
test_loss, test_accuracy = model.evaluate(test_data, test_labels_encoded)
```

`model.evaluate` calculates the test loss and test accuracy using the preprocessed test data and encoded labels.

The test loss and test accuracy are then printed to the console:

```
print(f"Test Loss: {test_loss:.4f}")
print(f"Test Accuracy: {test_accuracy:.4f}")
```

This provides quantitative measures of the model's performance on the test dataset.

5. Confusion Matrix:

A confusion matrix is generated to further evaluate the model's performance, particularly in terms of true positives, true negatives, false positives, and false negatives.

The code predicts labels for the test data and converts the predicted probabilities to binary labels (0 or 1):

```
test_predictions = model.predict(test_data)
test_predictions = (test_predictions >= 0.8).astype(np.int32)
```

Then, the confusion matrix is computed using the `confusion_matrix` function from Scikit-Learn:

```
conf_matrix = confusion_matrix(test_labels_encoded, test_predictions)
```

- The confusion matrix is a 2x2 matrix where:
  - The top-left cell represents true negatives (TN).
  - The top-right cell represents false positives (FP).
  - The bottom-left cell represents false negatives (FN).
  - The bottom-right cell represents true positives (TP).
- The confusion matrix is visualized as a heatmap using seaborn to provide insights into the model's performance in terms of class-wise classification accuracy.

By following these testing procedures, we can assess the trained model's performance on unseen data and gain insights into its ability to correctly classify "Real" and "DeepFake" images. The test accuracy and confusion matrix are important metrics for understanding how well the model generalizes to new data.

5. Results and Discussion:

5.1 Project results & analysis:

5.1.1. Image classification results

The classification scores network on the Deepfake dataset are provided in Table 3. this networks have achieved quite similar score, approximately around 90%, when evaluating each frame independently. It's worth noting that we do not anticipate achieving a higher score due to the presence of facial images in the dataset that have been extracted with notably low resolutions.

Network	Deepfake classification score		
Class	forged	real	total
Detection Modle	0.882	0.901	0.891

Table 2. Classification scores considering each frame independently.

5.1.2. Training and Vakidation Loss, Accuracy

What we have here is Training and validation Loss in First graph and Training and validation Accuracy in second where Blue line signify Training and Red for validation . by oberseving both graphs we get to Know that Loss is decreasing in each ittration and accuracy is increasing with each ittration . the test loss is 0.34 and accuracy is 0.8587

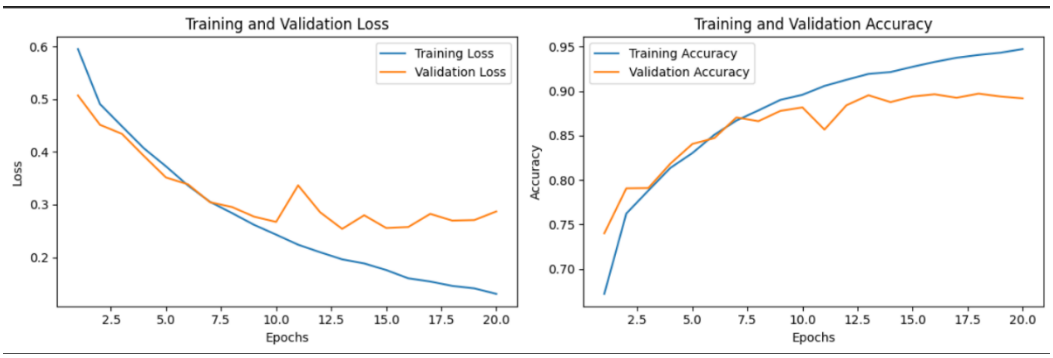


Figure4 : Training and Vakidation Loss, Accuracy Graph

5.1.3.. Confution matrix

To understand the result of our project we analyses it using confusion matrix which while show us how our model is performing to detect fake and real images . where our model found 1854 real – real images and 1330 deepfake images which makes 52% real and 37.37% fake images .

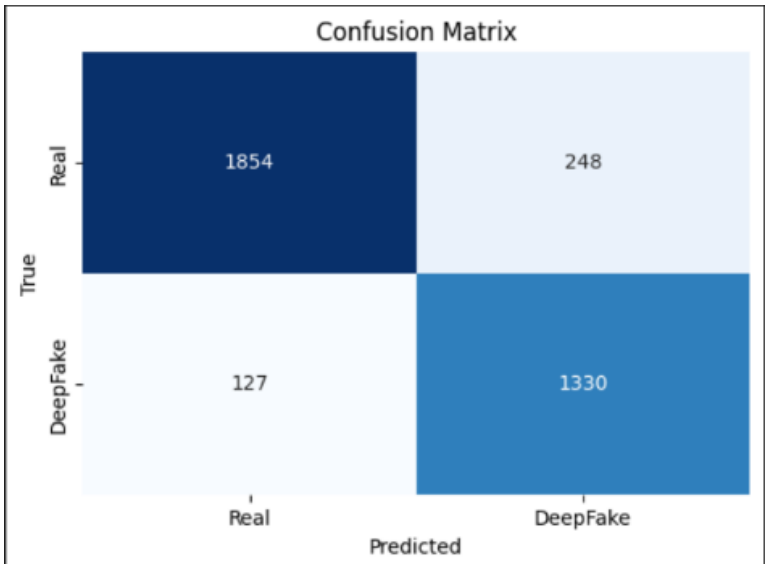
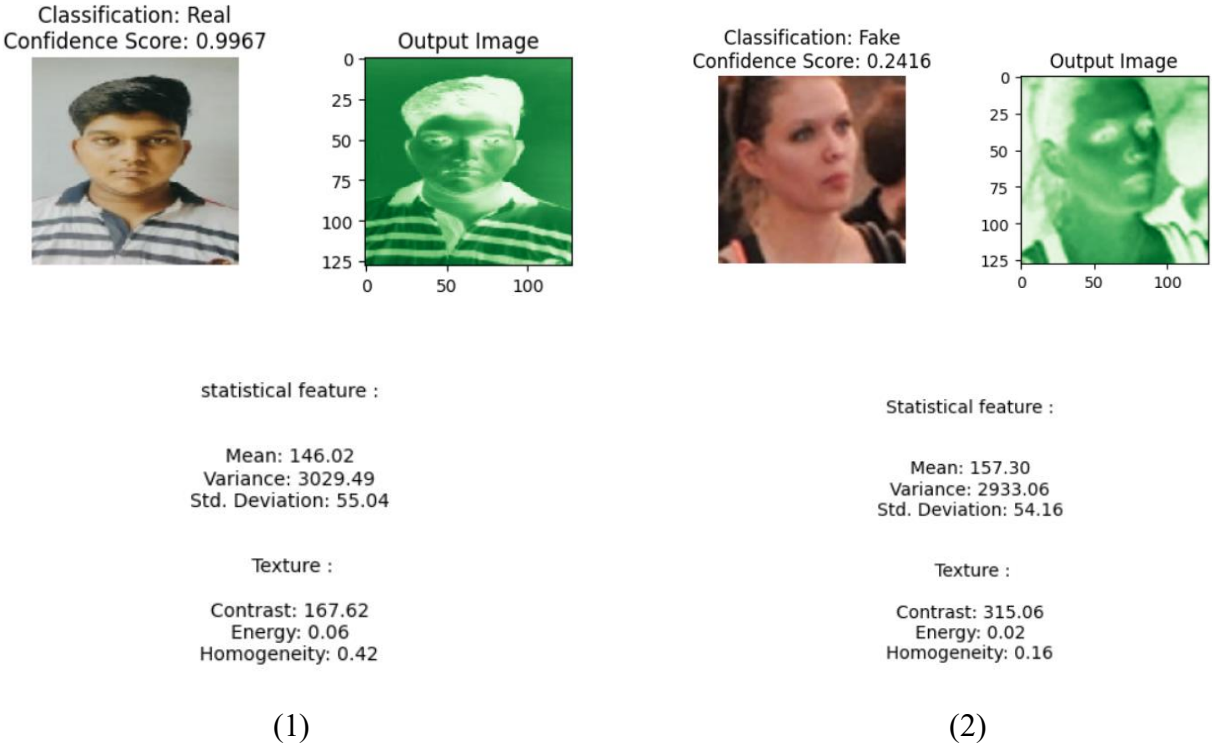


Figure5 : Confusion Matrix

5.1.4. Output

The outpu of out modle will is displayed using cmap and grayscale to anhance all the textures, edges in images because our modle is heavily depends on this parameters as it is less complex model it do have lesser parameters to work on which make is most reliable and also easy to use . here first image is real with 0.996% confidence where second image is with 0.2416% confidence which conclude second image is fake .



(1) (2)

Figure6 : Output images

5.2 Comparison with project objectives:

This all were our objective

- To review recent studies on deepfake content detection.

- To propose an efficient neural network approach for detecting manipulated facial videos.

- To evaluate the proposed approach's performance.

First was achived in literature review while we have also developed a deep learning cnn model which can successfully differnciat between deep fakes and real images and in result anylises we studied all aspects of out model by observing epoch graph and drawing confution matrix so yes we have achived all of the objectives.

6. Conclusion:

6.1 Summary of the project:

In the contemporary landscape, the risks associated with video face tampering have gained substantial recognition. In response, we present a potential network architecture designed to efficiently detect such manipulations while maintaining a low computational overhead. Furthermore, we provide access to a dataset specifically dedicated to Deepfake, a highly prevalent yet inadequately documented subject, to the best of our knowledge. Our experimental findings reveal that our approach achieves an average detection rate of 90% for Deepfake, even when subjected to real-world conditions on the internet.

A fundamental tenet of deep learning is its capacity to generate solutions to complex problems without the necessity of a priori theoretical exploration. However, it remains crucial to comprehend the underpinnings of these solutions to assess their strengths and limitations. To achieve this understanding, we dedicated considerable effort to visualizing the inner workings of our networks.

6.2 Future work and recommendations:

For future we can build website on which people can get easy access of model . also we can build a forgery detection system which can be used by governments and other security departments also We anticipate that future advancements will continue to provide tools for a deeper comprehension of deep networks, facilitating the creation of even more effective and efficient models in the pursuit of enhanced security and accuracy.

7. References:

[1] D. E. King. Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research*, 10:1755–1758, 2009. 4

[2] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[3] F. Chollet et al. Keras. https://keras.io, 2015. 5

[4] D. Erhan, Y. Bengio, A. Courville, and P. Vincent. Visualizing higher-layer features of a deep network. *University of Montreal*, 1341(3):1, 2009. 6

[5] H. Farid. A Survey Of Image Forgery Detection. *IEEE Signal Processing Magazine*, 26(2):26–25, 2009. 1

[6] P. Garrido, L. Valgaerts, O. Rehmsen, T. Thormahlen, P. Perez, and C. Theobalt. Automatic face reenactment. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4217–4224, 2014. 2

[7] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 3

[8] optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[9] W. Shi, F. Jiang, and D. Zhao. Single image superresolution with dilated convolution based multi-scale information learning inception module. *arXiv preprint arXiv:1707.07128*, 2017. 3

[10] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 3

[11] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015. 3

[12] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 4

[13] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[14] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 3

[15] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, et al. Going deeper with convolutions. Cvpr, 2015. 3

[16] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner. Face2face: Real-time face capture and reenactment of rgb videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2387– 2395, 2016. 1, 2, 3

[17] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–I. IEEE, 2001. 4