Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 218 (2023) 2636–2652

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Machine Learning and Data Engineering

Revealing and Classification of Deepfakes Video's Images using a Customize Convolution Neural Network Model

Usha Kosarkar[a], Gopal Sarkarkar[b], Shilpa Gedam[c]

[a]*Department of Science and Technology, GHRIET, Nagpur, India*
[b]*Department of Artificial Intelligence, G H Raisoni College of Engineering, Nagpur, India*
[c]*Department of Computer Science, Shivaji Science College, Nagpur, India*

Abstract

Deepfake has been exploited in recent years despite its widespread usage in a variety of areas to create dangerous material such as fake movies, rumors, and false news by changing or substituting the face information of the sources and so poses enormous security concerns to society. Research on active detection & prevention technologies is critical as deepfake continues to evolve. Deepfake has been a blessing, but we've taken advantage of it by utilizing it to swap faces. Deepfake is a new subdomain of Artificial Intelligence (AI) technology in which one person's face is layered over another person's face, which is becoming more and more popular on social networking sites. Deepfake pictures and videos can now be created much more quickly and cheaply due to ML (Machine Learning), which is a primary component of deepfakes. Despite negative connotations attached to the term "deepfakes," technology is increasingly being used in commercial & individual contexts. New technical advancements have made it more difficult to distinguish between deepfakes and images that have been digitally manipulated. The rise of deepfake technologies has sparked a growing sense of unease. The primary goal of this project is to properly distinguish deepfake pictures from real images using deep learning techniques.

In this study, we implemented a customized CNN algorithm to identify deepfake pictures from a video dataset and conducted a comparative analysis with two other methods to determine which way was superior. The Kaggle dataset was used to train & test our model. Convolutional neural networks (CNNs) have been used in this research to distinguish authentic & deepfake images by training three distinct CNN models. A customized CNN model, which includes several additional layers such as a dense layer, MaxPooling, as well as a dropout layer, has also been developed and implemented. This method follows the frames extraction, face feature extraction, data preprocessing, and classification phases in determining whether Real or Fake images in the video reflect the objectives. Accuracy, loss, and the area under the receiver operating characteristic (ROC) curve were used to characterize the data. Customized CNN outperformed all other models, achieving 91.4% accuracy, a reduced loss value of 0.342, as well as an AUC of 0.92. Besides, we obtained 85.2% testing accuracy from the CNN and 95.5% testing accuracy from the MLP-CNN model.

Keywords: Deepfake detection; Deep learning; Customize CNN; Deepfake Detection Challenge Dataset; Classification

10.1016/j.procs.2023.01.237

1. Introduction

Many deepfake videos have been [1] shared on social media as a result of the ease with which new technologies may be accessed. An example of a "deepfake" is a picture or video in which that person's likeness is substituted by that of another person. Deepfake is becoming one of the most severe concerns facing modern society. Many pornographic videos have included celebrities' faces being swiped over their photos using Deepfake. As well as spreading misinformation for politicians, deepfake was also utilized in this role[2][3][4]. In 2018, a fake video for Barack Obama was made to include comments he had never said. With Joe Biden's tongue out, deepfakes have been employed already in the US 2020 presidential election. Deepfakes may have a negative influence on our society and propagate misinformation, particularly on social media, as a consequence of these detrimental applications of the technology [5]. The term [6]"deepfake" is a fusion of terms "deep learning" and "fake." It is mostly the result of neural networks in machine learning and relates specifically to the forgeries of images, videos, and audio created by GANs (Generative Adversarial Networks) [7]. Deepfakes have the potential to accelerate the growth of the entertainment, cultural exchange, as well as education industries, therefore enhancing not only the teaching level in the area of education but also the whole quality of life. However, deepfake is frequently used to create fake news and fabricate electronic evidence, misinforming the public and disrupting societal order. This technique has developed into the most sophisticated method of network attack. Deepfake [8]is capable of creating bogus images/videos that are difficult to discern with human eyes, resulting in social chaos [9]. For a long time, academics and the film industry have been fascinated by the idea of synthesizing pictures or videos. The majority of early false pictures and videos were created using graphical algorithms, as opposed to other approaches. And it was not until later that a technology called Deepfake was made available to the general public that it became widely recognized. A few clicks are all it takes to create a video recording of someone doing or saying something they did not want to use Deepfake, which is heavily reliant on DNN (Deep Neural Networks). Deepfake videos [10] have had a considerably greater effect than anybody could have predicted, and this might influence how people evaluate the veracity of media reports in the future. The idea that "seeing is believing" has been disproven. While the technology is harmless when used for recreational reasons, some individuals may utilize it for political or other nefarious goals, which might have major repercussions[9].

In response to this scenario, researchers have begun to investigate several approaches for distinguishing deepfake videos from real ones. In most cases, machine learning is used in conjunction with other detection approaches. It is fairly uncommon for researchers to use NN [11] of various designs to look for differences among fake and real videos, but in other cases, they have turned to handcrafted features that may be used to uncover semantic differences like the pattern of eye blinking [12] or head postures or face warping traces or particular habits of facial expression and movement while speaking[13]. To the best of our knowledge, maximum approaches either rely only on features of individual frames, neglect temporal data, or are too reliant on training datasets, preventing them from being generalized. Most of the time, study on deepfake video forensics is still in its start [9].

In several domains, with computer vision, Natural Language Processing (NLP) [14], & machine vision [15], deep learning (DL) [16] has shown to be a powerful and valuable approach. Deepfakes employs DL technology to create fake photos and videos of people that are impossible to tell apart from the genuine thing when seen by a human eye. Recently, several research has been undertaken to better understand how deepfakes function, as well as numerous new algorithms using deep learning, have been developed to identify these fakes. Computer vision, big data analytics, and human-level control are all examples of complicated issues for which DL has been effectively used. As a result, DL technologies have also been used to construct software that might pose a danger to the security of the United States and its people's privacy. Deepfake is a recent example of a DL-powered application.

1.1. Motivation

For the most part, the Deepfake toolkit works by modifying the face's most important traits while leaving the rest of the face unchanged. With a few changes, the video's character has changed. As a result of this, the video's message may be entirely different. Such alterations are carried out frame-by-frame, which is seen to be the most

significant limitation of Deepfake so far. A well-designed system can fine-tune each image to make it more realistic, but coordination of realism of huge series of pictures on which multiple temporal limits are imposed is very challenging.

We employ a CNN (Convolutional Neural Network) learning model in our approach. Using facial landmarks detection, structured data is retrieved from video frames and supplied as input to the model, as illustrated in Fig. 1. Automatic feature extraction is performed using video image frames fed directly into the CNN. To get the final result, the CNN output is coupled to a dense neural layer as well as an activation layer.

1.2. Contribution

The following is a list of the paper's key contributions:

1) To detect deepfake face images using CNN deep learning techniques.
2) Use of customized deep learning method to detect deepfake face image to get the best accuracy.
3) Detect all face features (eyes, lips, nose, etc.) using the Facial landmark predictor model.
4) Achieved the best accuracy as an important factor that kept in mind is the different facial expressions of faces.
5) To develop a universal frame model to detect deepfake faces with greater accuracy.
6) To perform comparative analysis with three methods.

The paper is organized as follows. Deepfake's development status and our contributions are briefly discussed in Section 1, and related efforts are discussed in Section 2. In Section 3, we describe the architecture, innovation, and benefits of our deepfake detection approach in great detail; in Section 4, we explain our experimental setup and provide our detection performance results; in Section 5, we summarize and introduce future work.

2. Related work

In this part, we are going to look at some of the research that has been done in the field of Deepfake detection & production. Deepfake films like these are becoming more popular on social media networks, prompting the international community to take the threat seriously and, as a result, encouraging academics throughout the globe to build sophisticated deepfake detection tools. It is possible to find a variety of ways in the recent literature.

Unsupervised contrastive learning is used to build a novel deepfake detection algorithm in this study [17]. We first make 2 different versions of an image and feed them into 2 separate networks, one of which is an encoder and the other is a projection head. Maximizing the projection head's outputs' degree of correspondence is how the unsupervised training is accomplished. To test the detection efficiency of our unsupervised technique, we train an efficient linear classification network using the unsupervised features. Unsupervised learning may achieve equivalent recognition efficiency to current advanced supervised algorithms in both inter & Intra database contexts, according to several experiments. In addition, they carry out ablation tests to test the efficacy of our approach.

In this article, [18] CNN facial recognition models, such as Alex Net & Shuffle Net, are applied to distinguish between authentic and fraudulent images of people. Fake/real face recognition collection from Yonsei University's Computational Intelligence Photography Lab is used to evaluate the performance & operation of all separate algorithms After normalizing the images, Error Level Analysis (ELA) is performed before the images are used in a variety of CNN models, which are all unique. Next, the K-NN (K-Nearest Neighbour) & SVM (Support Vector Machine) techniques are used to extract the detailed features from the CNN models An accuracy of 88.2% was found for Shuffle Net's KNN, compared to an accuracy of 86.8% for Alex Net's vector.

In this article [19], multilayer hybrid recurrent DL models for deepfake video detection are presented. Noise-based temporal face convolutional features & temporal learning of hybrid recurrent DL models are used to create the models proposed in this article. These models' performance against stacked recurrent DL models has been shown via experiments.

A deep ensemble learning method called DeepfakeStack has been proposed thru [20] to address the issues given by Deepfake multimedia. The proposed method generates a better composite classifier by combining several state-of-the-art classification models based on deep learning. With an accuracy of 99.65 percent and an AUROC score of 1.0, our tests reveal that DeepfakeStack beats the competition in identifying Deepfake. A Real-time Deepfake detector might be built using our technique.

This work [21] provides a method for exposing such fake videos, which makes usage of CNN architecture & Transfer Learning approach. Every video frame is fed into a CNN, which then utilizes the extracted feature information to train an effective binary classifier that can tell the difference between genuine and altered videos. A comprehensive collection of deepfake videos collected from a variety of datasets is used to test the method's accuracy. All of the models had good training accuracy, with each model achieving more than 98%. It was the inceptionV3-based model that had the best accuracy.

In this study [22], the proposed strategy is depending upon utilizing residual noise, which is the difference between the original picture as well as its denoised form. Research of residual noise has shown that it is efficient in deepfake detection due to the unique and discriminative properties that it has, which can be captured successfully by CNNs with TL (Transfer Learning). To test the effectiveness of our technique, we used low-resolution video clips from FaceForensics++ & high-resolution video clips from Kaggle DFDC (Deepfake Detection challenge). When compared to other competing methodologies, the acquired findings demonstrate a high degree of accuracy.

[16] examine deepfake detection algorithms Xception &MobileNet as 2 techniques for classification tasks to repeatedly identify deepfake videos in this research. Four fake video creation techniques & 2 advanced NNs were used to train, test, and compare a total of 8 deepfake video classification models, which were then associated & reviewed. Each model demonstrated adequate classification performance when applied to the corresponding dataset that was utilized in its development. This article makes use of four datasets created using four distinct deepfake technologies that were used in the development of FaceForensics++ to train and evaluate the algorithm. The accuracy of the findings is high across all datasets, with accuracy ranging between 91 and 98 percent depending upon deepfake technologies used. In addition, we built a voting process that can identify fake videos by aggregating results of all 4 approaches, rather than just one.

3. Research Methodology

3.1. Problem statement

Building systems that conduct facial identification poses several challenges for face detection, making it one of the most difficult tasks in image processing. Face detection is the first issue to be addressed. The following two causes are to blame for the difficulties in Face Detection[23]. face expressions and diverse facial traits are included. People communicate their feelings and intentions via their facial expressions, making them one of the most impactful and immediate temperaments. It is important to note that facial emotions, such as anger or enjoyment, may directly change the look of a person's face. Many individuals wear spectacles, while others have a beard or mustache, and yet others exhibit scars from an earlier life. These features are known as facial features. There has been a steady rise in the quality of deepfakes, which has increased the need for new detection approaches. To deepfake detection, deep classifiers & shallow classifiers are the two major classifier types. Fake images and videos may be distinguished from genuine ones using shallow classifiers because of the irregularity of their features. As an example, the reflections in the eyes may be lost, as well as other details. Similar discrepancies may exist in the teeth, which may be exploited in the same way.

To identify fake videos, this study presents a DL (Deep Learning) [20]strategy based on customized CNN (Convolutional Neural Networks) [24]. Fig. 1 depicts the proposed system's steps. Video is first given as an input, from which individual image frames may be retrieved. The location of the eyes, nose, and lips may be determined with the use of a facial landmarks detector. Eye blinks and other facial features may also be deduced from this information. Before feeding the model with this data, it is necessary to do some kind of preprocessing. However, pre-processing step transforms the pictures into their numerical form. In this first, it crops the face region of interest

and resizes the input image frames into 224 x 224. Now ensures that all images are in the RGB channel. Training, validation, and testing sets have been separated after completing the preprocessing phase. Then, the customized CNN model conducts feature extraction and trains on features that are extracted. Classification stage can predict whether given video is deepfake or not using this customized deep learning technique based on CNN model.

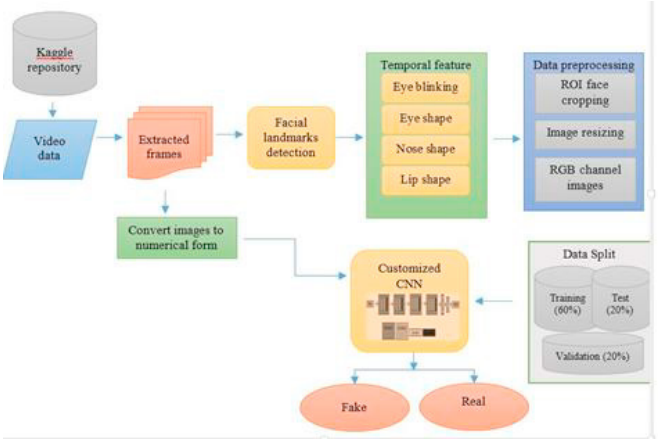


Fig 1. Block diagram of the proposed model

This section describes a proposed mechanism for categorizing video as REAL or FAKE by following a series of processes outlined in the previous section.

3.1.1. Frame Extraction & Facial Landmarks Detection

Image frames are created by slicing a video into individual images. There are a total of 3735 frames in this video. The face is identifiable in each image. The positions of 68 facial landmarks (x, y coordinates) are retrieved from the face area. The dlib library includes a facial landmark detector that has already been trained. To identify a face in a picture, the dlib library is used to first identify 68 facial landmarks. This landmark detector uses an ensemble of regression trees to predict the face's landmark placements based on pixel intensities in images.

3.1.2. Temporal Facial Feature Analysis

1) Eyeblink detection

As most novice deepfakes have either no blinking or quick unnatural blinking, deepfakes need a higher level of complexity to avoid compromising on eye blinking. The blinking pattern of the subject in the shot is picked up in this step. Eye coordinates (points 37–46 in list of retrieved facial landmarks) are derived from the face landmarks and provided as input to the blink detector. Detection of blinking is done by calculating the eye's aspect ratio (EAR). It is possible to depict each eye individually using six distinct landmark locations. The eye is signified by six (x, y) coordinates, beginning with the left corner (p1) & progressing clockwise from there by plotting points (p2, p3, p4, p5, p6). The EAR for a single eye is determined using the following equation:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 + p_4\|} \quad (1)$$

where ||p2-p6|| means the distance between points p2 and p6.

When the eye is open, the value of EAR stays constant; however, when the eye is closed, the value of EAR drops to zero. The frequency of blinks is increased if the average of the EAR values of both eyes falls below a threshold

(EYE_AR_THRESH) in defined no. of consecutive frames (EYE_AR_CONSEC_FRAMES). We utilized parameters 0.3 & 3 for EYE_AR_THRESH and EYE_AR_CONSEC_FRAMES, respectively, in this study.

The facial landmarks detector is used to extract face features including eye, nose, & lip coordinates. To capture the vast range of eye forms produced by the face-swapping approach prevalent in most deepfake videos, features from the eyes were extracted. A person's eye shape in a real video is quite stable, according to our research. Regardless of how the video was modified to make it seem false, this was never the case. Additionally, the majority of facial abnormalities, including facial warping, occur around the mouth area. Because of deepfake manipulation, there are different lip forms. This is why we want to use the variations in the face characteristics' shapes crosswise frames to train our classifier.

The eye shape detector uses eye coordinates (points 37–46) derived from facial landmarks as input. Using an eye form detector, the eye shape detector calculates Euclidean distance (d1) among endpoints of the left eye & Euclidean distance (d2) among endpoints of the right eye, respectively. The Lip shape detector uses lip coordinates (points 49–68) retrieved from face landmarks. Using d1 among inner lip coordinates, the length of inner lips (d3) is calculated. Similar to this, the length of outer lips (d4) may be determined by computing Euclidean distance among outer lip coordinates. When the Nose Shape detector receives facial landmark data (points 28–36), it uses that information to determine the nose's shape. The Euclidean distance (d5) among a base of the nose's two edges is used to calculate the nose's base width. A similar calculation is made using Euclidean distance (d6) to determine the nose's highest point. Consequently, the widths of the eyes (d1, d2), the inner & outer lip coordinates (d3, d4), and the top & base width of the nose (d3, d4) were retrieved as form features (d5, d6).

3.1.3. Data Pre-processing

Before analysis, we must increase the image's quality so that we can do so more effectively. Preprocessing allows us to remove unwanted artefacts and improve some aspects that are critical to the application we're working on. Depending on the application, some of these aspects may be different. We need to establish a baseline size for all pictures that are fed into our AI algorithms since the size of certain images captured by a camera and put into our AI algorithms might change.

1) Crop the face region of interest (ROI)

Using computer vision, Face Crop can automatically identify faces in photos. A rectangle crop is then applied, focusing on either all of the faces of the largest face. Select a DNN (Deep Neural Network) for better accuracy and to set the degree of confidence in face identification before implementing the algorithm. At 300x300 pixel scales, faces can be recognized by DNNs even when the picture has been scaled up or down. By "cropping" a picture, we mean choosing and removing the area of interest (also known as the ROI) from the image. A face-detection application, for example, may need cropping the face out of a picture. When we crop a picture, we are attempting to eliminate the areas of the image that are outside of our interest range. Selecting a Region of Interest, or ROI is a frequent term for this step. NumPy array slicing may be used to crop a picture after it has been captured.

2) Image resize

Resizing is the process of increasing or decreasing the size of a picture without removing any content. In essence, resizing a picture means making it larger or smaller. Because in certain cases, size does play a role. As a result, resizing is most effective when used to reduce the photo's size to meet a certain dimension or reduce the file size. In this case, we have resized the photos to 224 × 224 pixels in resolution.

3.1.4. Data split: Training, Validation, and Testing

Training, validation, & testing subsets are included in this dataset. 60 percent of the data was utilized for training, 20 percent for validation, and 20 percent for testing in every database. This is the equivalent of 2399, 750, and 600

photos in the dataset. In each subset, the percentage of an actual and fraudulent video was the same. To identify the optimum CNN architecture, the validation phase was employed in this process. To train the model, the validation set was utilized to choose the best-performing architecture, & training &test sets were combined to assess the model once it had been trained.

3.1.5. Customized Convolutional Neural Network (CNN)

To identify patterns in pictures, a convolutional neural network(also known as a ConvNet or CNN) [25] is a DNN[26] that is frequently utilized. In the first stage, the video's visual frames are retrieved and converted to numerical data (NumPy arrays). Additionally, each picture is resized to fit within the dimensions of 220px by 3px for easier processing & consistency. The CNN is supplied this information about the visual frames. As an input parameter, 32, 64, & 128 increasingly bigger filters are used so that the network may learn more distinct features. It is made up of 40 epochs of layers such as Conv2D → ReLU → BatchNormalization → MaxPooling2D, as well as a densely connected layer, and it is trained using a customized CNN. Afterward, we apply a thick layer on top of the previous one, along with the requisite Batch Normalization & Dropout layers. The overall model summary for the proposed Customized CNN is given in table 1. In this model, we have used a total of 20 layers to make the CNN as customized CNN that included four convolution layers (conv2d), six batch normalization layers, three max-pooling layers, four drop-out layers, one flatten layer, and two dense layers.

Table 1. Summary of the proposed Customized CNN.

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 32)	896
batch_normalization (BatchNormalization)	(None, 224, 224, 32)	128
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
dropout (Dropout)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 74, 74, 64)	18496
batch_normalization_1 (BatchNormalization)	(None, 74, 74, 64)	256
conv2d_2 (Conv2D)	(None, 74, 74, 64)	36928
batch_normalization_2 (BatchNormalization)	(None, 74, 74, 64)	256
max_pooling2d_1 (MaxPooling 2D)	(None, 37, 37, 64)	0
dropout_1 (Dropout)	(None, 37, 37, 64)	0
conv2d_3 (Conv2D)	(None, 37, 37, 128)	73856
batch_normalization_3 (Batc hNormalization)	(None, 37, 37, 128)	512
conv2d_4 (Conv2D)	(None, 37, 37, 128)	147584
batch_normalization_4 (BatchNormalization)	(None, 37, 37, 128)	512
max_pooling2d_2 (MaxPooling 2D)	(None, 18, 18, 128)	0
dropout_2 (Dropout)	(None, 18, 18, 128)	0
flatten (Flatten)	(None, 41472)	0
dense (Dense)	(None, 1024)	42468352
batch_normalization_5 (BatchNormalization)	(None, 1024)	4096

dropout_3 (Dropout)	(None, 1024)	0
dense_1 (Dense)	(None, 2)	2050

==========

Total params: 42,753,922
Trainable params: 42,751,042
Non-trainable params: 2,880

_____

An explanation of CNNs is provided in this section. We use a filter to examine the impact of surrounding pixels. We yield a filter of a size set via user (a rule of thumb is 3x3 or 5x5) and slide it over the image from the top left to the bottom right, which is precisely what you would expect. A convolutional filter is used to determine a value for every pixel in the picture. A feature map is created for each filter once it has passed over the picture. An activation function is used to determine if a certain feature is extant at a specific position in the picture. Add additional filtering layers & construct more feature maps to create a deeper CNN that becomes abstract as we go deeper into the network. Even though pooling layers may theoretically do any sort of operation, only max-pooling is employed since we need to locate outliers – these are the points at which our network perceives the features.

Several filters are applied to the picture to extract various information using a convolutional layer. When it comes to CNN's, the ReLU (Rectified Non-Linear Unit) [27]is the most effective non-linearity since it combats sigmoidal gradients. ReLU is simpler to calculate & provides sparsity.

When building a CNN, there are three kinds of layers to consider: convolutional layer, pooling layer, & fully connected layers/dense layer, as well as an additional layer known as the dropout layer. There are a variety of factors that may be tweaked in each of these levels, and they each have a specific job to do with the supplied data.

1) Convolution Layers

These are the filtering layers in a deep CNN, where the original picture is filtered or additional feature maps are applied. This is where the vast majority of the network's user-defined parameters are located. No. of kernels & also size of kernels are the two most critical characteristics to consider while creating a model.

The 2D convolution layer is the most often used kind of convolution and is sometimes abbreviated as conv2D. In a conv2D layer, a filter or kernel performs elementwise multiplication on two-dimensional input data. As a consequence, all of the data will be condensed into a single display pixel. When the kernel slides over a site, it will do the identical action at each position, changing a 2D matrix of features into another 2D matrix of features.

2) Pooling layers

There are several different types of convolutional layers, however, pooling layers serve a particular function like max-pooling, which takes the largest value in the filter area, or average pooling, which takes the average value in the filter region. In most cases, they are utilized to minimize the network's dimensions.

3) Dropout layers

In most cases, dropout is applied to the fully connected layers solely since these layers have the most parameters and are thus more prone to excessively co-adapt, leading to overfitting. Convolutional layers (for example, Conv2D) & pooling layers may both be employed before or after dropout (for example, MaxPooling2D). A basic heuristic says that dropout should only be applied after the pooling layers, however, this isn't always true. It is possible to apply dropout to every feature map cell or element.

4) Fully connected layers/ Dense layers

Dense layers or FCLs are added before the classification output of CNN and also utilized to flatten findings before classification. It is comparable to an MLP's output layer.

3.1.6. Proposed Algorithm

Input: Deepfake video dataset
Output: Good classification results

Strategy:

Step 1. Input video dataset from Kaggle
Step 2. Frames extraction from videos
Step 3. Detection of the all-face features using Facial landmark predictor model
  a. Eyes blink detection
  b. Eyes shape detection
  c. Lips shape detection
  d. Nose shape detection
Step 4. Perform data preprocessing on frames
  a. Crop the face region of interest
  b. Image resize into 224 x 224
  c. Ensure all images in the RGB channel
Step 5. Data splitting into three parts
  a. Training set (60%)
  b. Validation set (20%)
  c. Testing test (20%)
Step 6. Hyper Parameters setting
Step 7. Apply Customized CNN model by adding some layer for training
  a. conv2d layer
  b. Batch normalization
  c. Max pooling layer
  d. Drop out layer
  e. Flatten
  f. Dense layer
Step 8. Perform testing on a test set
Step 9. Calculate performance metrics
Step 10. Classification results whether it is fake or real

4. Research Methodology

Python and also its libraries were used to conduct this research. The initial learning rate was set to 0.0001 as well as batch size was set to 32 for purpose of testing. This process was stopped after 40 epochs. To extract the frames from the videos, a 10-frame-per-10-second frame rate is used. CNN training convergence led to the selection of this maximum value. The accuracy, loss, & ROC AUC metrics are used in a quantitative examination of the performance of the designs in the study. The percentage of rectified pixels in each class is referred to as the accuracy.

4.1. Data description

From the dataset collected by the Deepfake Detection Challenge, we employ 242 videos, 199 of which are fictitious, while the remaining 53 are authentic. A single video lasts for ten seconds. To get a more even distribution of actual and fraudulent videos, we have included 66 videos from the YouTube dataset acquired from Dessa that

contains real videos to achieve a more balanced distribution of real and fake videos. In all, 318 videos were utilized, 199 of which were fraudulent and 119 of which were authentic.

The material is made up of.mp4 files that have been compressed to a total of ~10GB each. In addition to a filename, label (REAL or FAKE), original and split columns, described below under Columns, the metadata.json accompanying each pair of.mp4 files include the following:

Columns

- filename - filename of the video
- label - whether a video is FAKE/REAL
- original - in case that train set video is FAKE, the original video is enumerated here
- split - It is always equal to "train".

Figure 2 shows collection of sample images of both type fake and real.

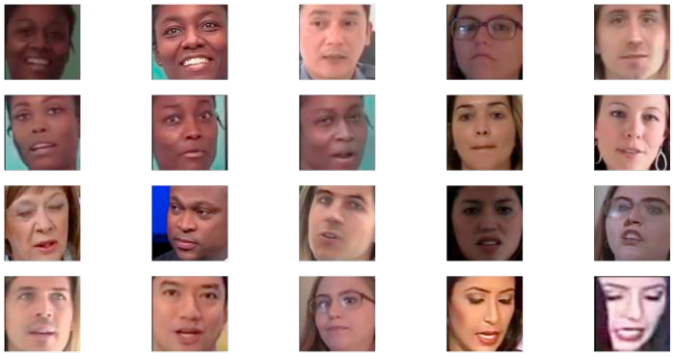


Fig. 2. Sample images

4.2. Data description

An important element of every project is testing our ML method. A metric such as accuracy_score may provide satisfactory results when testing this model, however other metrics like logarithmic_loss or any other of its kind may yield bad results. The majority of the time, classification accuracy is utilized to evaluate the efficiency of the proposed model; nevertheless, this is not adequate to evaluate the proposed model. Various forms of assessment metrics have been discussed in this section.

- Logarithmic Loss
- Classification Accuracy
- Area under Curve
- Confusion Matrix

4.2.1. Classification Accuracy

When we use the word "accuracy", we most often refer to classification accuracy. A good measure of accuracy is the ratio of accurate predictions to the total no. of input samples.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made} \quad (2)$$

Only if there are equal numbers of examples from every class can it be used effectively.

Considering the following description: 98% of the samples in our training set to come from class A, while the remaining 2% come from class B. Our model can easily reach 98 percent training accuracy by correctly predicting every training sample in class A. " The test accuracy would be 60% on a set of samples with 60% from class A and only 40% from class B. However, classification accuracy offers us a false perception that we have attained great accuracy levels.

4.2.2. Binarycross entropy/Logarithmic Loss (LL)

A logarithmic loss, often known as a log loss, is used to penalize false classifications in data. The multi-class classification works well with it. Using LL[28], the classifier must assign probabilities to every class for all data. LL is computed as follows if there are N examples belonging to M classes:

$$Logarithmic\ Loss = \frac{-1}{N}\sum\sum Y_{ij} * \left(\log P_{ij}\right) \quad (3)$$

where,

p_ij, shows the probability of sample i belonging to class j
LL has no upper bound also it occurs on a range [0, ∞)
y_ij, shows whether sample i goes to class j or not.

Log Loss that is closer to 0 suggests better accuracy, whereas LL that is farther away from 0 specifies less accuracy. As a general rule, minimizing LL results in better classification.

4.2.3. Confusion Matrix

This is, as its name implies, generates a matrix as output that summarizes the overall performance of the model.

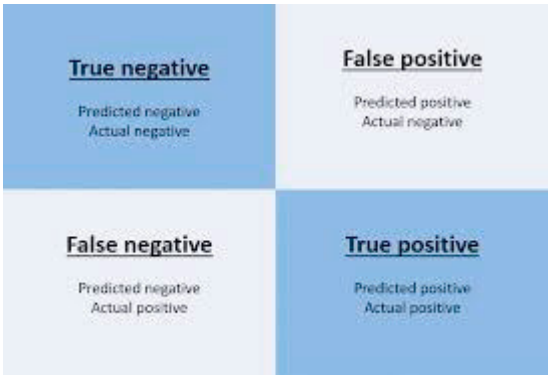


Fig. 3. Confusion Matrix

There are four significant terms:

- **True Positives:** TP
- **True Negatives:** TN
- **False Positives:** FP
- **False Negatives:** FN

Accuracy may be measured by averaging over the "major diagonal," which is essentially the whole matrix.

$$Accuracy = \frac{True\ Positive + True\ Negative}{Total\ Sample} \quad (4)$$

The Confusion Matrix serves as the foundation for all other measurements.

4.2.4. Area Under Curve

An assessment statistic called AUC (Area Under the Curve) is extensively utilized. Classification problems are solved with it. This means that a classifier's AUC is equal to how likely it is to rank a randomly picked positive sample more than an equally randomly generated negative sample in terms of AUC. Before determining AUC, we need to grasp a couple of fundamental terms:

True Positive Rate (Sensitivity): TPR (TP/FN+TP) is calculated as TP/(FN+TP). The TPR is a percentage of all positive data points that are taken into account when determining whether or not a sample is positive.

$$True\ Positive\ Rate = \frac{True\ Positive}{False\ Negative + True\ Positive} \quad (5)$$

True Negative Rate (Specificity): TNR is calculated as follows: TN / (FP+TN). In other words, the FPR is a percentage of negative data values that are accurately classified as negative data points.

$$True\ Negative\ Role = \frac{True\ Negative}{True\ Negative + False\ Negative} \quad (6)$$

False Positive Rate (FPR): FPR is calculated as follows: *FP / (FP+TN)*. In the context of all negative data points, FPR is a percentage of negative data points that are wrongly labelled positive.

$$False\ Positive\ Rate = \frac{False\ Positive}{True\ Negative + False\ Positive} \quad (7)$$

4.3. Result Evaluation & Analysis

This research has been able to tell if a video is a deepfake or not. A voice or a face swap may be used in a deepfake (or both). In training data, it is indicated by the label "FAKE" or "REAL" in the label column. Here, we have projected the probability of the video being fraudulent.

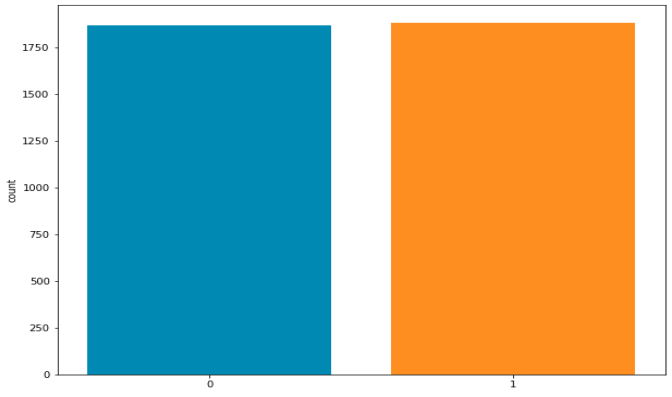


Fig. 4 shows a dataset distribution graph for the deep fake video dataset

Fig. 4 shows a dataset distribution graph for the deep fake video dataset. Here, the x-axis shows video class & the y-axis shows total counts. In this plot, the video class is categorized as 0 and 1, where 0 for Real and 1 for Fake. From this graph found that there is the almost same number of counts for both classes.

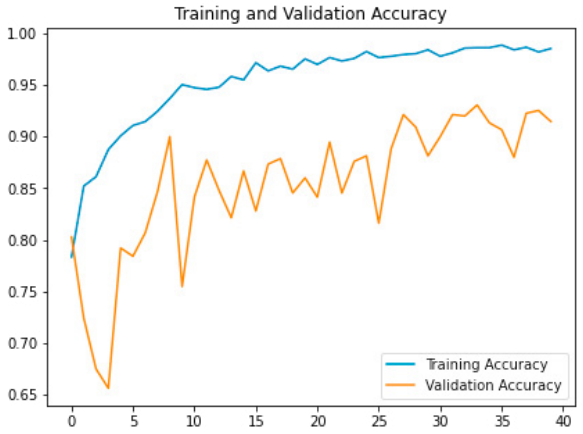


Fig. 5. Model accuracy graph

Fig. 5 depicts the proposed Customized CNN model's model accuracy with blue and orange lines denoting training and validation accuracy, respectively. There are epochs on the x-axis & percent accuracy on the y-axis. This plot found that training accuracy is very high with an increased number of epochs while validation accuracy is minimized in comparison to training accuracy however it has also achieved a great accuracy level and there are many variations during the testing.

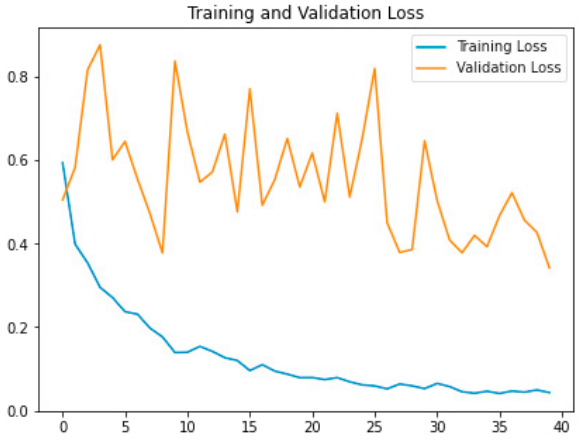
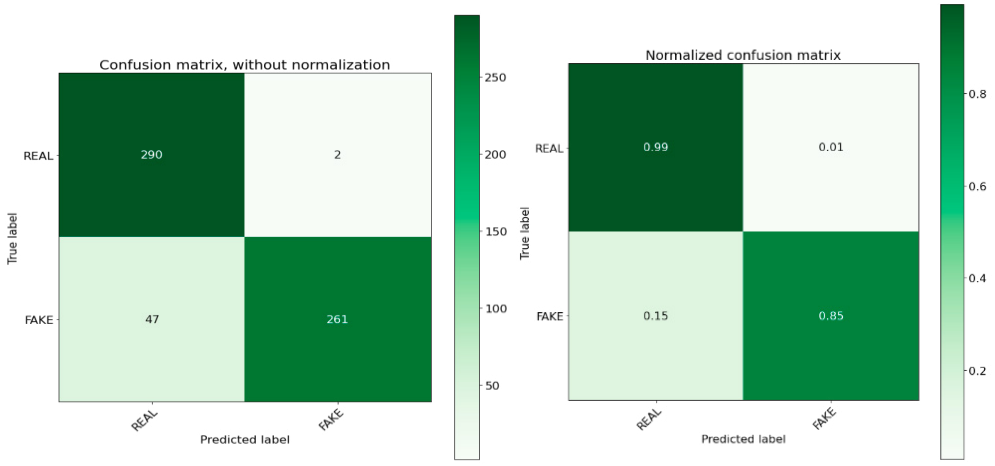


Fig. 6. Model loss graph

Fig. 6 depicts the proposed Customized CNN model's model loss graph, with orange & blue lines denoting training and validation losses, respectively. As a similar way of accuracy graph, if accuracy is high then obviously loss will be minimized. Thus the training loss is high in the training data but the validation loss is minimized with many variations during testing.



(a) (b)

Fig. 7. Confusion matrix for test data

Fig. 7 shows the confusion matrix for test data in which fig. 7 (a) plotted the confusion matrix without normalization whereas fig. 7 (b) plotted the normalized confusion matrix for calculating the video is fake or real. Let's suppose we have a binary classification problem. We have several examples that fall into 2 categories: Fake and Real. In addition, we have our particular classifier that predicts class for provided input example based on data. This is what we found after running our model through 600 tests.

The 4 important terms are represented as :

- **TP**: A total of 261 examples were identified in which we predicted Fake, as well as the actual output, was likewise Fake.
- **TN**: The instances when we predicted Real, as well as the actual output, was Real, that is, 290.
- **FP**: The instances when predicted Fake, as well as actual output, was Real, that is, 2.
- **FN**: The instances when we predicted Real, as well as actual output, was Fake, that is, 47.

Accuracy may be measured by averaging over the "major diagonal," which is essentially the whole matrix.
Accuracy = (TP+TN)/ total sample
= (261+290)/600
= 0.918

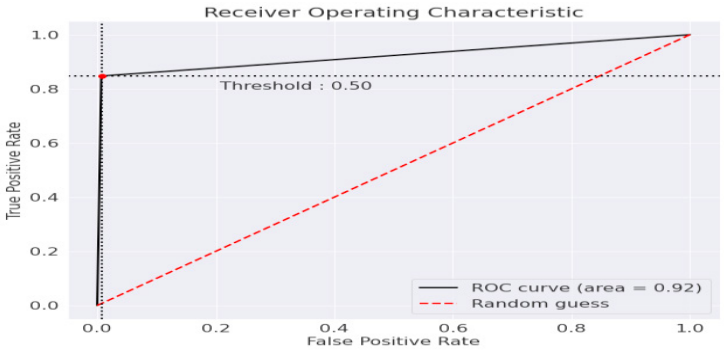


Fig. 8. ROC curve (customized CNN)

Figure 8 depicts the ROC curve. AUC) is derived from the ROC, and it is an essential measure. AUC is an appropriate statistic to utilize due to the imbalance of the dataset. There is a [0, 1] range of values for the FPR & TPR. The FPR and TPR are both calculated and a graph is created at numerous threshold values like (0.00, 0.02, 0.04,...., 1.00). AUC denotes the area under the curve of a plot of FPR versus TPR at various locations in the interval [0, 1]. Our model's performance improves as the value increases. This model's AUC score is 0.92, indicating that it has a 92% probability of successfully identifying a fake video.

Table 2. Performance results evaluation of the proposed customized CNN with two models

Model	Training loss	Training Acc	Validation loss	Validation Acc	AUC score
Base (MLP-CNN)	0.1948	0.9552	0.4383	0.8929	0.87
CNN	2.2433	0.8523	2.2810	0.8501	0.83
Proposed	0.1003	0.9721	0.3420	0.9147	0.92

Table 2 represents the accuracies of three models along with their AUC scores, in this proposed CNN model is compared with both existing methods.

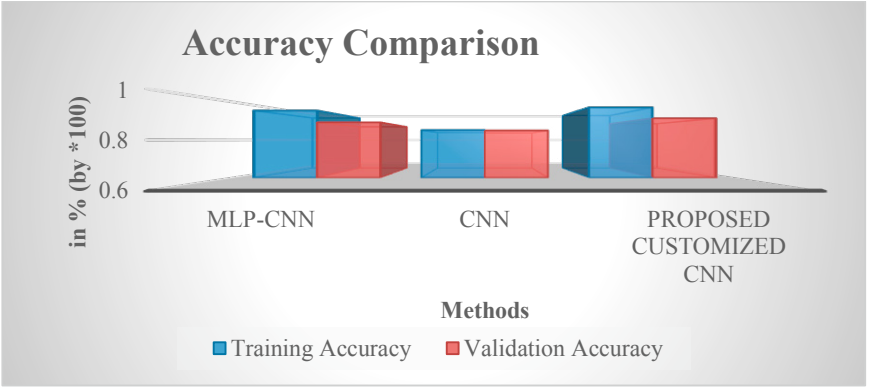


Fig. 9. Bar graph for accuracy comparison

Fig. 9 visualized the comparison bar graph for accuracy among three methods. This comparative graph shows that the training and validation accuracy of CNN only is approximately equal but it is very minimal to another method MLP-CNN which has achieved higher training accuracy but reduced validation accuracy (compared to training data). However, MLP-CNN achieved good classification results but proposed Customized CNN outperform for both training and validation data accuracy over these two methods.

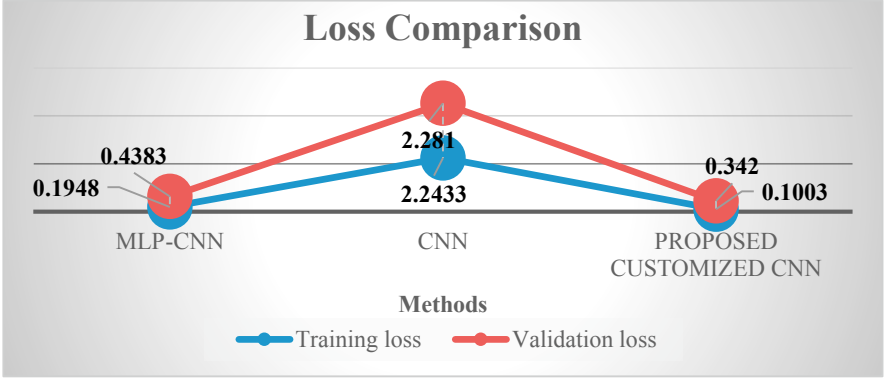


Fig. 10. Line graph for loss comparison

Fig. 10 visualizes the comparative line graph for displaying loss value (binary cross-entropy) differences among all three methods. This comparative visualization shows that training and validation loss of CNN only is 2.243 and 2.281, respectively but it is very high to another method MLP-CNN which has achieved training and validation loss 0.194 and 0.438, respectively, which is minimal (compared to CNN only method). However, MLP-CNN achieved decreased loss values but proposed Customized CNN outperform by achieving reduced loss value for both training and validation data, which are 0.1 and 0.342, respectively, over these two methods.

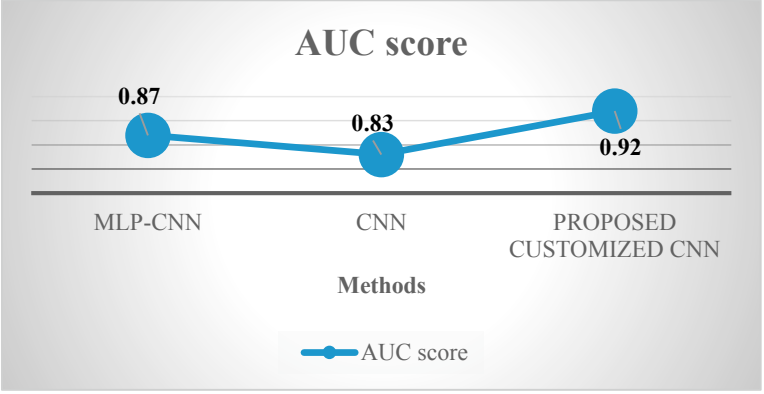


Fig. 11. Line graph for AUC score comparison

Fig. 11 visualized the comparison line graph for comparing the AUC score for all three methods. This comparative line graph shows that the AUC score for MLP-CNN is 0.87 but it is higher than another existing method CNN only that achieved AUC score is 0.83. As we can see that MLP-CNN achieved good AUC score value than CNN only results but it is also has minimized AUC score value than proposed Customized CNN that achieved a 0.92 AUC score value.

5. Conclusion and Future work

As part of this study, a unique approach has been developed to reveal AI-generated deepfake video together with powerful feature extraction & classification utilizing customized CNNs. The proposed method exhibits testing accuracy of 91.47%, loss of 0.342, and AUC score of 0.92 despite of training on a small subset of data. The comparative analysis found that the proposed customized CNN outperform two existing methods like CNN and MLP-CNN.

The future scope of this study might include identifying strategies to broaden the variety of people who can be reliably detected by the algorithm, like people of color, in order to guarantee justice and minimized prejudice in the system. More spatial & temporal face data should be included in a reasonable mix as well. In addition, it is required to evaluate better models using additional DL approaches on larger, more balanced datasets.

References

[1] A. M. Almars, “Deepfakes Detection Techniques Using Deep Learning: A Survey,” *J. Comput. Commun.*, 2021, doi: 10.4236/jcc.2021.95003.
[2] L. Nataraj *et al.*, “Detecting GAN generated Fake Images using Co-occurrence Matrices,” 2019, doi: 10.2352/ISSN.2470-1173.2019.5.MWSF-532.
[3] S. Y. Wang, O. Wang, R. Zhang, A. Owens, and A. A. Efros, “CNN-Generated Images Are Surprisingly Easy to Spot.. For Now,” 2020, doi: 10.1109/CVPR42600.2020.00872.
[4] C. C. Hsu, C. Y. Lee, and Y. X. Zhuang, “Learning to detect fake face images in the wild,” 2019, doi: 10.1109/IS3C.2018.00104.
[5] D. Guera and E. J. Delp, “Deepfake Video Detection Using Recurrent Neural Networks,” 2019, doi: 10.1109/AVSS.2018.8639163.
[6] F. Sun, N. Zhang, P. Xu, and Z. Song, “Deepfake Detection Method Based on Cross-Domain Fusion,” *Secur. Commun. Networks*, vol. 2021, no. 2, 2021, doi: 10.1155/2021/2482942.
[7] A. Aggarwal, M. Mittal, and G. Battineni, “Generative adversarial network: An overview of theory and applications,” *Int. J. Inf. Manag.*

Data Insights, 2021, doi: 10.1016/j.jjimei.2020.100004.
[8] J. C. Dheeraj, K. Nandakumar, A. V. Aditya, B. S. Chethan, and G. C. R. Kartheek, "Detecting Deepfakes Using Deep Learning," 2021, doi: 10.1109/RTEICT52294.2021.9573740.
[9] M. Li, B. Liu, Y. Hu, and Y. Wang, "Exposing deepfake videos by tracking eye movements," 2020, doi: 10.1109/ICPR48806.2021.9413139.
[10] Y. Al-Dhabi and S. Zhang, "Deepfake Video Detection by Combining Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN)," 2021, doi: 10.1109/CSAIEE54046.2021.9543264.
[11] A. Badale, L. Castelino, and J. Gomes, "Deep Fake Detection using Neural Networks," vol. 9, no. 3, pp. 349–354, 2021.
[12] Y. Li, M. C. Chang, and S. Lyu, "In Ictu Oculi: Exposing AI created fake videos by detecting eye blinking," 2019, doi: 10.1109/WIFS.2018.8630787.
[13] S. Agarwal, H. Farid, Y. Gu, M. He, K. Nagano, and H. Li, "Protecting world leaders against deep fakes," 2019.
[14] M. Nagao, "Natural language processing and knowledge," in *2005 International Conference on Natural Language Processing and Knowledge Engineering*, 2005, pp. 1-, doi: 10.1109/NLPKE.2005.1598694.
[15] G. Lee and M. Kim, "Deepfake detection using the rate of change between frames based on computer vision," *Sensors*, 2021, doi: 10.3390/s21217367.
[16] D. Pan, L. Sun, R. Wang, X. Zhang, and R. O. Sinnott, "Deepfake Detection through Deep Learning," 2020, doi: 10.1109/BDCAT50828.2020.00001.
[17] S. Fung, X. Lu, C. Zhang, and C. T. Li, "DeepfakeUCL: Deepfake Detection via Unsupervised Contrastive Learning," 2021, doi: 10.1109/IJCNN52387.2021.9534089.
[18] R. Rafique, M. Nawaz, H. Kibriya, and M. Masood, "DeepFake Detection Using Error Level Analysis and Deep Learning," in *2021 4th International Conference on Computing Information Sciences (ICCIS)*, 2021, pp. 1–4, doi: 10.1109/ICCIS54243.2021.9676375.
[19] G. Jaiswal, "Hybrid Recurrent Deep Learning Model for DeepFake Video Detection," in *2021 IEEE 8th Uttar Pradesh Section International Conference on Electrical, Electronics and Computer Engineering (UPCON)*, 2021, pp. 1–5, doi: 10.1109/UPCON52273.2021.9667632.
[20] M. S. Rana and A. H. Sung, "DeepfakeStack: A Deep Ensemble-based Learning Technique for Deepfake Detection," 2020, doi: 10.1109/CSCloud-EdgeCom49738.2020.00021.
[21] S. Suratkar, E. Johnson, K. Variyambat, M. Panchal, and F. Kazi, "Employing Transfer-Learning based CNN architectures to Enhance the Generalizability of Deepfake Detection," 2020, doi: 10.1109/ICCCNT49239.2020.9225400.
[22] M. C. El Rai, H. Al Ahmad, O. Gouda, D. Jamal, M. A. Talib, and Q. Nasir, "Fighting Deepfake by Residual Noise Using Convolutional Neural Networks," 2020, doi: 10.1109/ICSPIS51252.2020.9340138.
[23] S. Kolagati, T. Priyadharshini, and V. Mary Anita Rajam, "Exposing deepfakes using a deep multilayer perceptron – convolutional neural network model," *Int. J. Inf. Manag. Data Insights*, vol. 2, no. 1, p. 100054, 2022, doi: 10.1016/j.jjimei.2021.100054.
[24] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," 2018, doi: 10.1109/ICEngTechnol.2017.8308186.
[25] G. Botelho De Souza, D. F. Da Silva Santos, R. Goncalves Pires, J. P. Papa, and A. N. Marana, "Efficient width-extended convolutional neural network for robust face spoofing detection," 2018, doi: 10.1109/BRACIS.2018.00047.
[26] R. K. Kaliyar, A. Goswami, and P. Narang, "DeepFakE: improving fake news detection using tensor decomposition-based deep neural network," *J. Supercomput.*, 2021, doi: 10.1007/s11227-020-03294-y.
[27] R. Arora, A. Basu, P. Mianjy, and A. Mukherjee, "Understanding deep neural networks with rectified linear units," 2018.
[28] D. Godoy, "Understanding binary cross-entropy / log loss: a visual explanation," *towardsdatascience*, 2018. https://towardsdatascience.com/understanding-binary-cross-entropy-log-loss-a-visual-explanation-a3ac6025181a.