International Journal of Trend in Scientific Research and Development (IJTSRD)
Special Issue on Smart Innovations in Computer Science and Applications
Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

Computer Visioned Based Face Recognition Attendence System

Nishant Umrao Kale
Department of Science and Technology,
G. H. Raisoni Skill Tech University, Nagpur, India

Abstract

Attendance management is a fundamental administrative task in educational institutions and organizations; however, conventional methods such as manual roll calls, sign sheets, and card-based systems are time-consuming, error-prone, and vulnerable to proxy attendance. To overcome these limitations, this research paper presents the design and implementation of a computer vision–based face recognition attendance system that automatically identifies individuals and records attendance in real time using facial features.

The proposed system integrates image processing and deep learning techniques to detect, recognize, and verify human faces from live video streams captured through a camera. Initially, face detection is performed to localize facial regions from input frames, followed by preprocessing steps such as normalization, alignment, and noise reduction to improve recognition accuracy. A convolutional neural network (CNN)-based face recognition model is employed to extract discriminative facial embeddings, which are then compared with a pre-trained facial database using similarity metrics. Upon successful recognition, the system automatically logs attendance along with date, time, and confidence score into a centralized database.

The system is designed to operate in real-world environments and is robust to variations in illumination, facial expressions, pose, and minor occlusions. Experimental results demonstrate that the proposed approach significantly improves accuracy and efficiency compared to traditional attendance systems, while minimizing human intervention. The automation of attendance tracking not only saves time but also enhances reliability, security, and scalability. This research highlights the potential of computer vision and deep learning technologies in developing intelligent, contactless, and efficient attendance management solutions suitable for modern smart environments.

Keywords: Artificial Intelligence, Information Retrieval, Natural Language Processing, Machine Learning, Algorithms

1. Introduction

Attendance monitoring is an essential administrative process in educational institutions, corporate organizations, and industrial environments. It serves as a critical parameter for evaluating participation, discipline, productivity, and compliance. Traditionally, attendance is recorded through manual methods such as roll calls, sign-in sheets, or card-based systems. Although widely used, these approaches are inefficient, time-consuming, and prone to errors such as false marking, data redundancy, and proxy attendance. As organizations grow in size, managing attendance manually becomes increasingly impractical and unreliable.

With the rapid advancement of computer vision and artificial intelligence, automated biometric systems have emerged as effective alternatives to conventional attendance methods. Among various biometric techniques—such as fingerprint recognition, iris scanning, and voice recognition—face recognition has gained significant attention due to its non-intrusive, contactless, and user-friendly nature. Unlike other biometric systems, face recognition does not require physical interaction or specialized hardware, making it suitable for real-time and large-scale deployment.

Computer vision–based face recognition systems leverage image processing and deep learning algorithms to identify individuals based on unique facial features. These systems typically involve face detection, feature extraction, and classification stages. The integration of deep learning models, particularly convolutional neural networks (CNNs), has significantly improved recognition accuracy by enabling robust feature learning under varying conditions such as changes in illumination, facial expressions, pose variations, and partial occlusions.

The motivation behind developing a face recognition attendance system is to create an automated, efficient, and secure solution that minimizes human intervention while ensuring attendance recording. By using live video feeds from cameras, the system can identify individuals in real time and automatically mark their attendance in a digital database along with timestamps. This approach not only saves time but also eliminates fraudulent practices such as proxy attendance and manual data manipulation.

Furthermore, the proposed system aligns with the growing demand for smart and intelligent environments, including smart classrooms, smart offices, and automated access control systems. The adoption of such technology enhances operational efficiency and provides a scalable solution capable of handling large populations without compromising accuracy. The system can also be integrated with existing information management platforms, making it adaptable to various institutional requirements.

In this research paper, we present the design, development, and evaluation of a computer vision–based face recognition attendance system that utilizes deep learning techniques for accurate face identification. The system is designed to operate in real-world conditions and aims to address the limitations of traditional attendance systems by providing a reliable, secure, and fully automated solution.

1.1. Motivation

Attendance management is a fundamental requirement in educational institutions, corporate offices, and industrial organizations. Despite its importance, many institutions still rely on traditional attendance systems such as manual roll calls, register-based signatures, or card-based methods. These approaches are not only time-consuming but also prone to human errors, data inconsistency, and fraudulent practices such as proxy attendance. In large classrooms or organizations, manually recording attendance significantly reduces productive time and increases administrative workload.

Existing biometric attendance systems such as fingerprint and iris recognition provide better security compared to manual methods; however, they suffer from several limitations. These systems require physical contact, which leads to hygiene concerns, hardware wear and tear, and increased maintenance costs. Moreover, contact-based systems are inconvenient in high-traffic environments and may fail due to sensor damage or improper usage. These challenges became more evident during situations requiring contactless solutions, such as public health emergencies.

1.2. Contribution

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

Design of a Fully Automated Attendance System

Application of Deep Learning for Robust Face Recognition

Contactless and Non-Intrusive Attendance Mechanism

Reduction of Proxy Attendance and Human Errors

Efficient Attendance Logging and Data Management

Scalability and Adaptability

Foundation for Future Enhancements

2. Related work

The research on automated attendance systems has evolved significantly over the past two decades. Traditional manual methods have largely been replaced in many institutions by digital or semi-automated techniques such as barcode scanning, RFID cards, and biometric sensors. However, the advent of computer vision and deep learning has accelerated development in face recognition–based attendance systems. The following review examines key contributions and limitations in related research.

3. Research Methodology

3.1. Problem statement

Traditional attendance management systems—such as manual registers, ID cards, punch cards, or fingerprint scanners—suffer from several limitations including time consumption, proxy attendance, human error, lack of scalability, hygiene concerns, and high administrative overhead. In educational institutions and organizations with large populations, these systems become inefficient and unreliable.

Manual attendance recording is prone to errors and manipulation, while biometric systems like fingerprint scanners require physical contact, leading to wear-and-tear and health risks. RFID and card-based systems are vulnerable to misuse, as one person can mark attendance for another. Moreover, these methods often require dedicated hardware and constant human supervision.

With advancements in computer vision and deep learning, facial recognition has emerged as a non-intrusive, contactless, and automated solution for identity verification. However, designing a real-time face recognition attendance system presents challenges such as:

Variations in lighting conditions

Pose and facial expression changes

Occlusion (masks, glasses)

Real-time performance constraints

Accuracy vs. computational efficiency trade-offs

3.1.1. 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.

3.1.2. 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.

3.1.3. 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.

3.1.4. 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.

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.

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions\ made}$$

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:

$$Logarithmic\ Loss = {}^{-1}\sum\sum Y\ * (\log\ ) \qquad (3)$$

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.

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.

$$Accuracy = \frac{True\ Positive + True\ NeNNative}{Total\ Sample}$$

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.

$$True\ Positive\ Rate = \frac{True\ Positive}{False\ NeNNative+True\ Positive}$$

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.

$$True\ Negative\ Role = \frac{True\ \ NeNNative}{True\ NeNNative+False\ NeNNative}$$

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.

$$False\ Positive\ Rate = \frac{False\ \ Positive}{True\ NeNNative+False\ Positive}$$

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. It 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 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 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.

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

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.

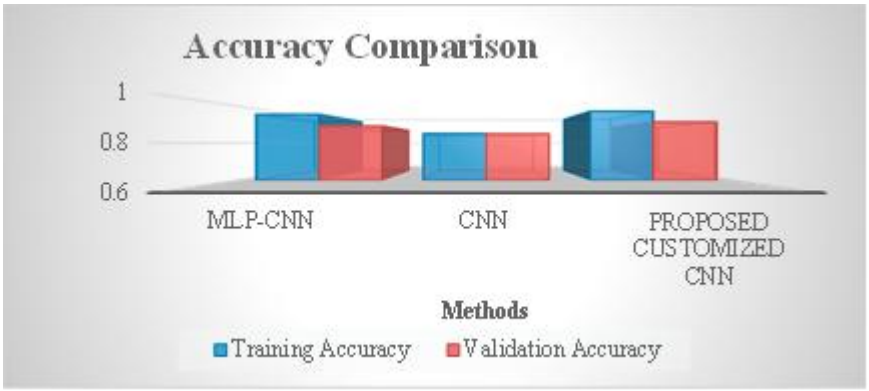

Fig. 9. Bar graph for accuracy comparison

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.

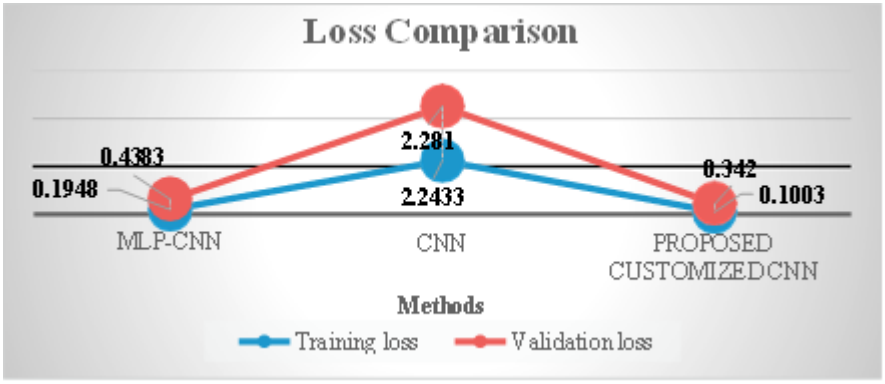

Fig. 10. Line graph for loss comparison

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.

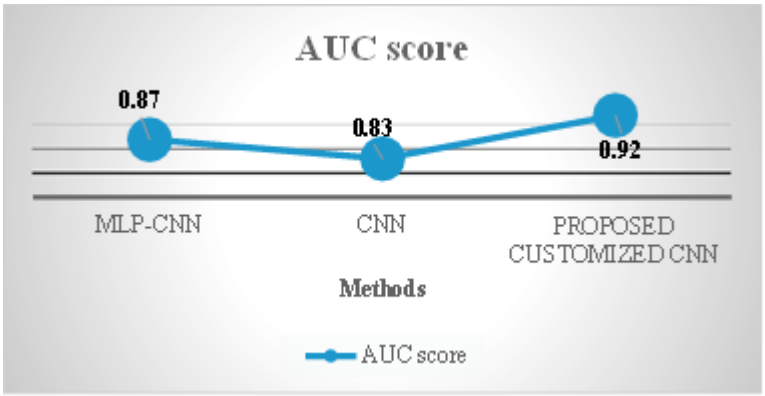

Fig. 11. Line graph for AUC score comparison

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*,