Acta Polytechnica **63**(5):305–319, 2023
https://doi.org/10.14311/AP.2023.63.0305

Published by the Czech Technical University in Prague

FAKE FACE DETECTION BASED ON A MULTI DISCRIMINATOR DEEP CNN ARCHITECTURE (MDD-CNN)

Chemesse Ennehar Bencheriet[a,*], Hiba Abdelmoumène[b], Abdennour Sebbagh[c], Abdennour Yahiyaoui[d], Zahra Taba[d]

[a] *8 Mai 1945-Guelma University, Computer Science Department/LAIG Laboratory, 24000 Guelma, Algeria*

[b] *8 Mai 1945-Guelma University, Computer Science Department/LabGED Laboratory, 24000 Guelma, Algeria*

[c] *8 Mai 1945-Guelma University, Department of Electrical Atomatic Engineering/LAIG Laboratory, 24000 Guelma, Algeria*

[d] *8 Mai 1945-Guelma University, Computer Science Department, 24000 Guelma, Algeria*

[*] corresponding author: cbencheriet@yahoo.fr

Abstract. Due to the robustness of the deep learning tools used to design these applications, fakes are becoming increasingly common as these applications become more widely available and accessible to the general public. These fakes are typically fake faces or even fake people, which are difficult to distinguish from real individuals. Therefore, we need more efficient applications for fraud detection. In this work, we propose a new multi-discriminator architecture to distinguish fake faces from real ones. The architecture consists of three deep networks (discriminators) competing with each other, each trained differently. The final decision is made by voting based on the decisions of the three discriminators. The core element of our architecture is the proposed new adversarial deep network discriminator (NDGAN), which is trained in three different ways, resulting in three distinct discriminators. Discriminator 1 undergoes adversarial training, discriminator 2 is trained using transfer learning, and the third discriminator undergoes supervised training with a standard CNN using examples and counterexamples. Training and testing were performed on 70 000 real faces from the Flickr-Face-HQ (FFHQ) dataset, while 70 000 fake faces were generated using Nvidia's StyleGAN. The tests conducted on the three networks produced significant results, with accuracy ranging from 79 % to 98 % for fake faces, and from 80 % to 98 % for real faces. The reliability of the discriminators contributes significantly to the overall performance of the multi-discriminator system, achieving an accuracy of 96 % for fake faces and 98 % for real faces.

Keywords: Fake face, real face, discriminator, MDD-CNN architecture adversarial training, transfer learning, deep learning.

1. Introduction

False images have become a major problem in recent years. They can easily deceive people and inevitably lead to serious social risks, such as false evidence and false information. They can also damage reputations and harm people. Today, fake faces can be generated with minimal effort and by anyone with sufficient equipment. Detecting deppfakes is tberefore becoming increasingly important.

The act of image forgery is not new, in 1997 [1], a project called "Video Rewrite" altered existing video footage of people to make them appear to be saying words that appeared on an altered audio track: essentially, putting words in their mouths. The early 2000s were relatively quiet as computer development moved deeper into the world of facial recognition. Developments in this field led to dramatic improvements in motion tracking that makes today's deepfakes more convincing. In 2001 [2], the active appearance models algorithm made its debut. Using a thorough statistical model to match a shape to an image proved to be a big step forward. This made face matching and tracking much more efficient.

Face manipulation is generally divided into two categories, faceswap (*facial change*), and facial expression reconstruction. Faceswap and DeepFakes belong to the same category: a target face replaces the face in the source image (see Figure 1). Faceswap is a graphical approach, while DeepFakes is based on GANs.

Mainly the Face2face and Neural Textures methods transfer facial expressions from the target image to the source image. The Face2Face method transfers the expression from the source video to the target video using blend shape coefficients. In contrast, the Neural Textures approach uses GANs to learn the texture of the target person [3]. New technological advances made it easier to create Deepfakes, which are hyper-realistic videos or images using face swaps that leave little trace of manipulation (see Figure 2). These Deepfakes are the result of artificial intelligence (AI) applications that mix and match images and video clips to create fake videos that appear real and natural. This technology can generate, for example, a humorous or political video of a person without the consent or permission of the person whose image and voice are involved. Deepfakes can now be created

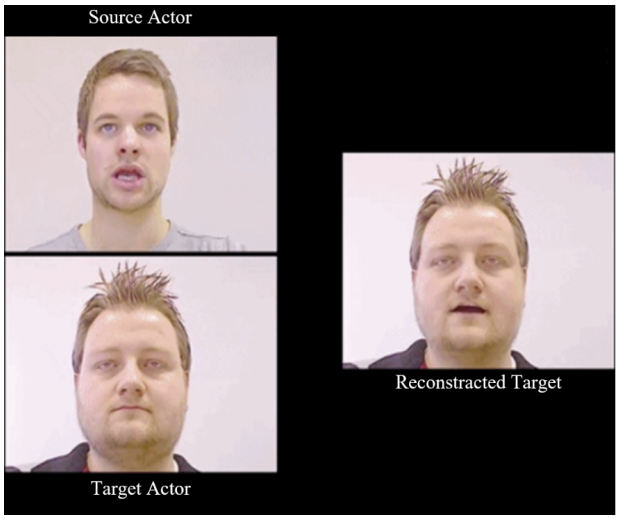
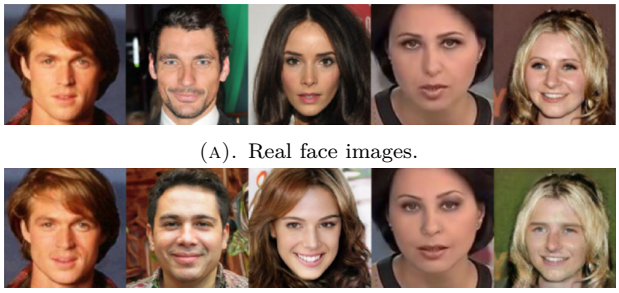


Figure 1. Example of facial reconstitution.



(a). Real face images.

(b). Fake face images.

Figure 2. Deepfake example [12].

using consumer-grade hardware as demonstrated by two papers published in 2016 and 2017 [4, 5]. The Face2Face project from the Technical University of Munich and the Synthesizing Obama project from the University of Washington established the feasibility of generating deepfakes. They have improved computing and rendering times while updating graphical fidelity for a photorealistic look. The game-changing factors for forgeries are the scope, scale, and complexity of the technology involved, as today anyone with a computer can create fake videos that are almost indistinguishable from real videos [6].

The GAN and Autoencoder networks have become very famous for their ability to produce forgeries of remarkable quality and undetectable by human beings. Recently, many architectures have been developed to generate fakes. A severe competition between these architectures was measured by the quality of the false images generated by each one. Several fake datasets consisting mainly of fake faces have been developed, such as the GAN collection [7], FaceForensics++ [8], Celeb-DF [9], DEFACTO [10], FaceSwap [11].

In this paper, we introduce a new architecture called NDCGAN, which stands for "Neural Discriminator-Generator Confrontation Generative Adversarial Network". This architecture consists of three discriminator modules that work together to distinguish between real and fake faces.

The first module, Discriminator 1, is trained against its opponent, the generator, to outperform it and accurately identify the fake faces generated by the generator. The second module, Discriminator 2, is the NDCGAN discriminator (Discriminator 1) trained using transfer learning techniques. This module leverages the pre-trained knowledge of Discriminator 1 to enhance its ability to distinguish between real and fake faces.

The third module, Discriminator 3, is a separate discriminator trained from scratch on both real and fake faces. It is designed to work independently of its associated generator, focusing solely on accurately classifying faces. In the final decision-making process, the three discriminators vote together on whether a given face belongs to the real face class or the fake face class. This voting mechanism ensures a more robust and reliable determination of a face's authenticity. Through the utilisation of NDCGAN and the collaboration of these three discriminator modules, our proposed architecture offers improved performance and accuracy in distinguishing between real and fake faces.

The rest of the paper is organised as follows: Section 2 presents existing work regarding fake detection based on deep neural networks. Section 3 provides an overview of the proposed architecture, along with a brief explanation of the baseline modules. In Section 4, we present the proposed model, along with the training parameters, implementation details, classification results, and discussions. Finally, Section 5 presents the conclusions and discusses future work.

2. Related work

Although deepfakes were initially created for entertainment purposes, such as allowing people to experience things that no longer exist or to attract extraordinary attention from online audiences, making a website popular in search engines. Later, their applications were exploited for counterfeiting and spreading misinformation, including the creation of false news and deceptive advertising, as well as the malicious use of video Deepfakes for blackmail. Faced with the emergence of fakes in our environment that are undetectable by humans, the interest of researchers has subsequently been directed towards finding robust algorithms for the detection of fake images and videos [13, 14] based on various deep-learning approaches, such as long short-term memory (LSTM), recurrent neural network (RNN), Convolutional Neural Network (CNN), and even the hybrid approaches have been proposed.

Guera and Delp [15] proposed a video Deepfake detection system that uses a recurrent convolutional neural network to extract features from each frame, which are then concatenated and sent to the LSTM for analysis. They produce an estimate of the probability that the sequence is a Deepfake or an unmanipulated video. The authors collected 600 video Deepfakes,

including 300 videos from various websites and another 300 videos from the HOHA database [16]. They presented the performance of their system in terms of accuracy using sub-sequences with lengths of 20, 40, and 80 frames. They achieved 96.7 % accuracy for 20 frames, 97.1 % for 40 frames, and 97.1 % for 80 frames.

Liu et al. [17], showed that CNN models are strongly texture-based rather than shape-based. They found that these models are focus predominantly on texture regions, such as skin and hair, while giving less importance to other regions. To investigate this further, the researchers conducted experiments specifically targeting skin regions, which are rich in texture information, but relatively poor in structural information. Based on their findings, they concluded that the texture of fake faces differs from that of real faces. Based on these observations, Liu et al. [17] introduced a novel architecture called Gram-Net, which improves the robustness and generalisation capabilities of CNNs in the detection of fake faces. The evaluation of their approach using styleGan [18] and CelebA-HQ [19] databases during the test phase yielded an estimated accuracy of 80.72 % in fake face detection. These results show a promising and encouraging direction for understanding false images from GANs.

The detection of the source of deepfake videos was performed by analysing the "heartbeat" of deep fakes and spatial and temporal feature analysis. For the detection of deepfake images, a hybrid approach was introduced, which uses a pairwise-learning; the approach first uses GANs to create and generate a fake image. Then, on the popular fake feature network (CFFN) generated by GANs, a pairwise learning model is used to capture the discriminative information between the fake and real images.

A. Deshmukh and S. B. Wankhade [20] conducted a study on the tools and algorithms used to create and detect deepfakes and the strategies associated with them. They also present the ideologies of deepfake algorithms and how deep learning has been used to enable such technologies. According to their findings, the most commonly used deepface tools are dfaker, FaceSwap-GAN, Faceswap, deepfakerlab, DeepFaker-tk.

Deepfake detection methods are divided into two dominant classes. The first class comprises false image detection approaches that rely on deep learning techniques such as CNN, SVM, random forest, multilayer perceptron, and GAN [21]. These methods generate images that are noticeable as fake. The second class, known as the False Video Detection Approach, can also be divided into two groups. The first group uses chronological features to identify deepfake videos, while the second group focuses on exploring visual artfacts. The researchers have proposed a pipeline technique that uses CNN and LSTM to detect deepfake videos from blinking eyes. The authors verified the proposed techniques on the following datasets: Forensics++ dataset, the UADFV, and DeepfakeTIMIT datasets. During the tests conducted, the authors found that general facial recognition systems, such as VGG and Facenet, are unable to successfully detect deepfakes. In addition, the use of SVM to assess image quality and lip synchronisation resulted in a significantly high error rate.

The aim of the work by Jung et al. [22] is to detect deepfakes generated by generative adversarial networks (GANs) using blink analysis. The authors propose a method called DeepVision that incorporates machine learning techniques. The DeepVision architecture has a pre-processor that receives input information. This input data includes parameters such as gender, age, activity, and time, which play a crucial role in assessing changes in the human eye blink. To perform the detection process, DeepVision uses the Target Detector algorithm for face detection and then uses the Eye Tracker algorithm to track and measure the number of blink repetitions and their duration. These measurements are then compared to a database of natural movement to determine whether the blink is natural. Deepvision achieves 87.5 % accuracy in detecting deepfake videos. It should be noted, however, this method may be less effective for people with mental illness, as abnormal blinking is often observed in such cases.

Korshunov and Marcel [23] presented two sets of Deepfake videos generated from the VidTIMIT database. First, they generated videos of different resolution qualities: (64×64) and (128×128). Then, they evaluated two neural network based systems, namely VGG [24] and facenet 6 [25], to distinguish between Deepfake and real videos. However, both systems showed a high error rate of 95 %, indicating their inability to effectively distinguish between the two.

To overcome this limitation, the authors investigated an audiovisual approach to detecting inconsistencies between lip movements and speech in the audio track. They found that lip-synchronisation-based methods failed to detect inconsistencies between lip movements and speech, while pure image-based approaches proved effective in detecting video Deepfakes. By using IQM techniques together with an SVM classifier, they achieved a significantly lower error rate of 8.97 % in detecting Deepfakes.

Li and Lyu [26] aimed to describe a new method based on deep learning that can effectively distinguish Deepfake videos from real ones. Due to computational resources and production time limitations, the Deepfake algorithm cannot distinguish between real and fake videos. The images undergo an affine deformation corresponding to the facial configuration of the source. Therefore, the authors proposed a convolutional neural network (CNN) model to detect the presence of artifacts from the face regions and surrounding areas. The CNN training is based on images collected from the internet (24442 JPEG face

images). To increase the diversity of the training, the authors modified the colour information, brightness, contrast, and sharpness for all training examples; they validated their work on the Deepfake UADFV video dataset [12]. This dataset contains 98 videos (32752 frames), including 49 real and 49 fake videos. The tests performed on the Deepfake videos demonstrate the effectiveness of the proposed method in practice with an accuracy rate of up to 97 %.

Yang et al. [27] introduced a pre-processing module called AMTEN, which uses the convolutional layer as a predictor to detect image manipulation. This module can be integrated into CNN-based models to improve their ability to detect image manipulation. By applying some post-processing operations, (including lossy compression), the authors simulated a real analysis of face images in various complex scenarios.

To demonstrate the effectiveness of the AMTENnet project, a series of experiments were conducted to achieve a higher detection rate. Yang et al. [27] used a large HFF database of 116 000 face images for training including real images of different resolutions and five types of false images. The dataset was divided into three subsets: 75 % for training, 5 % for validation, and 20 % for testing. As a result, AMTENnet achieved an accuracy of 95.17 %.

In their work, Chintha et al. [28] proposed an XcepTemporal convolutional recurrent neural network for Deepfake detection. They used a face detector called dlib to determine the main face in each frame of the video. The faces were encoded using the xception-Net architecture [29]. The spatial features extracted from xceptionNet were fed into a first bidirectional LSTM layer and the outputs from this layer were fed to a second bidirectional LSTM layer to produce secondary features. The feature vector obtained from the last LSTM unit in the second bidirectional layer was then fed through a fully connected layer and a classification layer. A dropout was added to the fully connected layer to improve regularisation. The model was trained using the traditional cross-entropy and KL divergence loss functions. In addition, the researchers introduced a complementary architecture for audio analysis by using multiple convolution modules to obtain audio feature representations. These audio embeddings were also fed through a bidirectional recursive layer. Training the model on the FaceForensics++ [9] and Celeb-DF [10] databases resulted in visual detection accuracies ranging from 84.8 % to 100 %. For audio detection, the authors demonstrated the robustness of their method, achieving a t-DCF (tandem detection cost function) of 0.1424. This indicates the generalisability of their approach to different types of attacks.

Lee et al. [30] aimed to detect a new manually produced facial manipulation by introducing a new dataset containing photoshopped face images created with tools such as Adobe Photoshop to detect both human-manipulated and machine-generated images. This dataset contains 1527 images developed with several levels of editing complexity and 621 original images used to create the fake faces. To address this challenge, the researchers developed a CNN-based model called Shallow-FakeFace (SFFN). This model shows promising results in detecting human-created fake images, achieving a 72.52 % OCR using less than 2500 fake images for training. The effectiveness of the SFFN model was alos evaluated on other GAN-generated fake faces, demonstrating a recognition rate of 93.99 % for low-resolution images. These results highlight the model's ability to generalise well against potential attackers.

In this study, we propose a new multi-discriminator architecture for distinguishing fake faces from real ones. The proposed architecture is a competition between three deep networks (discriminators), each having been trained differently. The final decision is made by voting on the decisions of the three discriminators. In summary, the paper presents the following contributions:

- Proposes a new architecture of DCGAN called ND-CGAN where the Discriminator 1 trains with its opponent (the generator) to get an advantage and recognise the fake faces generated by the latter with a considerable rate.
- Presents a Multi Discriminator Deep CNN (MDD-CNN) architecture based on three competing discriminators. The main advantage is that the three networks are involved in the final decision by voting on the predictions made by each. This significantly minimises prediction errors.

3. Proposed system architecture

The multi-discriminator architecture proposed in this work is based on the Deep Convolutional Generative Adversarial Network (DCGAN), which is a direct extension of the GAN. Using the same distribution of training data, GANs are used to train deep learning models to produce new data. Created in 2014 by Ian Goodfellow [31] and published in the paper Generative Adversarial Nets. These networks consist of two different models: the generator and the discriminator. The generator creates artificial or synthetic visuals that resemble training images. The discriminator examines an image and the output to determine whether it is authentic or fraudulent. During the training process, the discriminator strives to get better at detecting and categorising whether an image is real or fake by learning to create better fake images that trick the discriminator into categorising them as real images.

3.1. Deep convolutional generative adversarial network (DCGAN)

As an extension, DCGAN incorporates convolutional and convolutional-transposed layers explicitly in both the discriminator and the generator components. Convolutional layers preserve the spatial structure of an

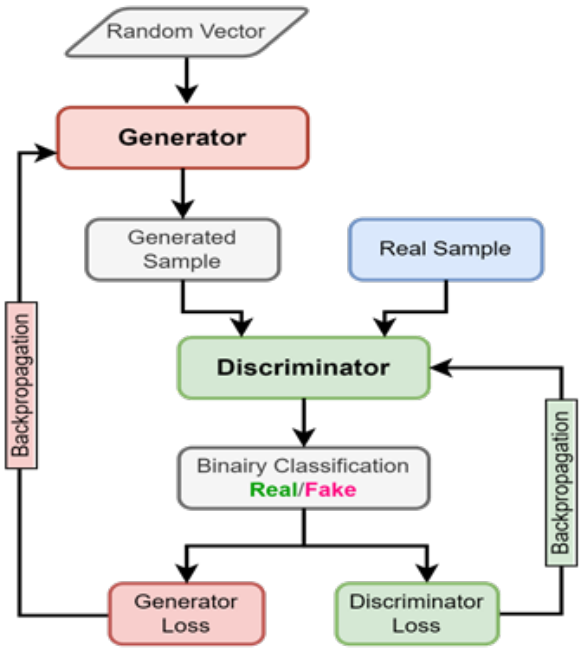


FIGURE 3. Deep convolutional GAN network.

image, which means that the most accurate and detailed features are extracted from an image. The generator and discriminator will, therefore, have more advanced spatial reasoning capabilities about the output they will generate and how to distinguish between the features of the real images and the fake images.

Improving the extracted features' quality is usually the reason why DCGANs are used for image processing [32]. Like all GANs, the DCGAN consists of a generator and a discriminator (Figure 3).

3.2. MULTI DISCRIMINATOR DEEP CNN ARCHITECTURE (MDD-CNN)

Our proposed architecture is based on three discriminators. The architecture is described in Figure 4. Where the main modules are:

- A new DCGAN architecture called NDCGAN, where the Discriminator 1 trains with its opponent (the generator) to overtake it and recognise the fake faces generated by the latter with a significant rate.
- The second module of our architecture is the Discriminator 2: this is the NDCGAN discriminator (Discriminator 1) trained by transfer learning.
- The third module is the Discriminator 3: which is the discriminator of the NDCGAN trained from scratch on faces and fake faces, without its generator.

The final decision of whether the face belongs to the real face class or the fake face class is the voting result of the three discriminators.

3.2.1. NEW DCGAN ARCHITECTURE

In the following, we describe the DCGAN architecture that consists of a generator network and a discriminator network, which are the main parts of the MDD-CNN architecture.

A. Architecture of generator network The generator consists of transposed convolutional layers that allow the noise vector to be unsampled and transformed into an image. In a classical CNN network, the convolutional layers will try to extract smaller and smaller features which are then be classified. In a generator, the transposed convolutional layers are designed to reverse the operations of the convolutional layer. This upsampling process enlarges the image and adds details to the final result. This is the part of the generator that "draws" the actual image. The transposed convolution layers of the generator will decompress the noise so that it becomes a 128×128 image with all the details in the right place in the image. The details of each layer are shown in Figure 5 and Table 1. Note that the activation of all layers is LeakyReLU (given by Equation (1)), except for the output layer which uses tanh (Equation (2)).

$$\varnothing(x) = \begin{cases} \alpha x & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}, \tag{1}$$

$$\tanh(x) = \frac{2}{1 + \mathrm{e}^{-2x}} - 1\,. \tag{2}$$

B. Discriminator network architecture The discriminator is composed of strided convolutional layers and LeakyReLU activations (Table 2, Figure 6). The input is a $128 \times 128 \times 3$ image, and the output is a probability between 0 and 1.

3.2.2. STABILITY CONDITIONS

To have a stable network, the following is necessary:

- Replacing pooling by stride convolution. This allows the network to learn its own spatial downsampling (by changing the size of the input). CNNs use pooling layers to reduce dimensions. For example, a 2×2 max pooling layer would take a 2×2 array of pixels and map it to a number that is the maximum among them. Stride convolution can reduce the dimension by skipping several pixels between convolutions instead of dragging the kernel one by one. It can also increase the dimension by adding empty pixels between the actual pixels.
- No fully connected layer at the end of the CNN. The generator is not a classifier, so this part is not needed.
- Use leakyReLU (Figure 7) in the generator except for the output, which uses tanh. The symmetry of the tanh function allows the model to learn faster and leakyReLU for the discriminator.

The output of the Leaky ReLU activation function is positive if the input is positive, and a controlled negative value if the input is negative. The negative value

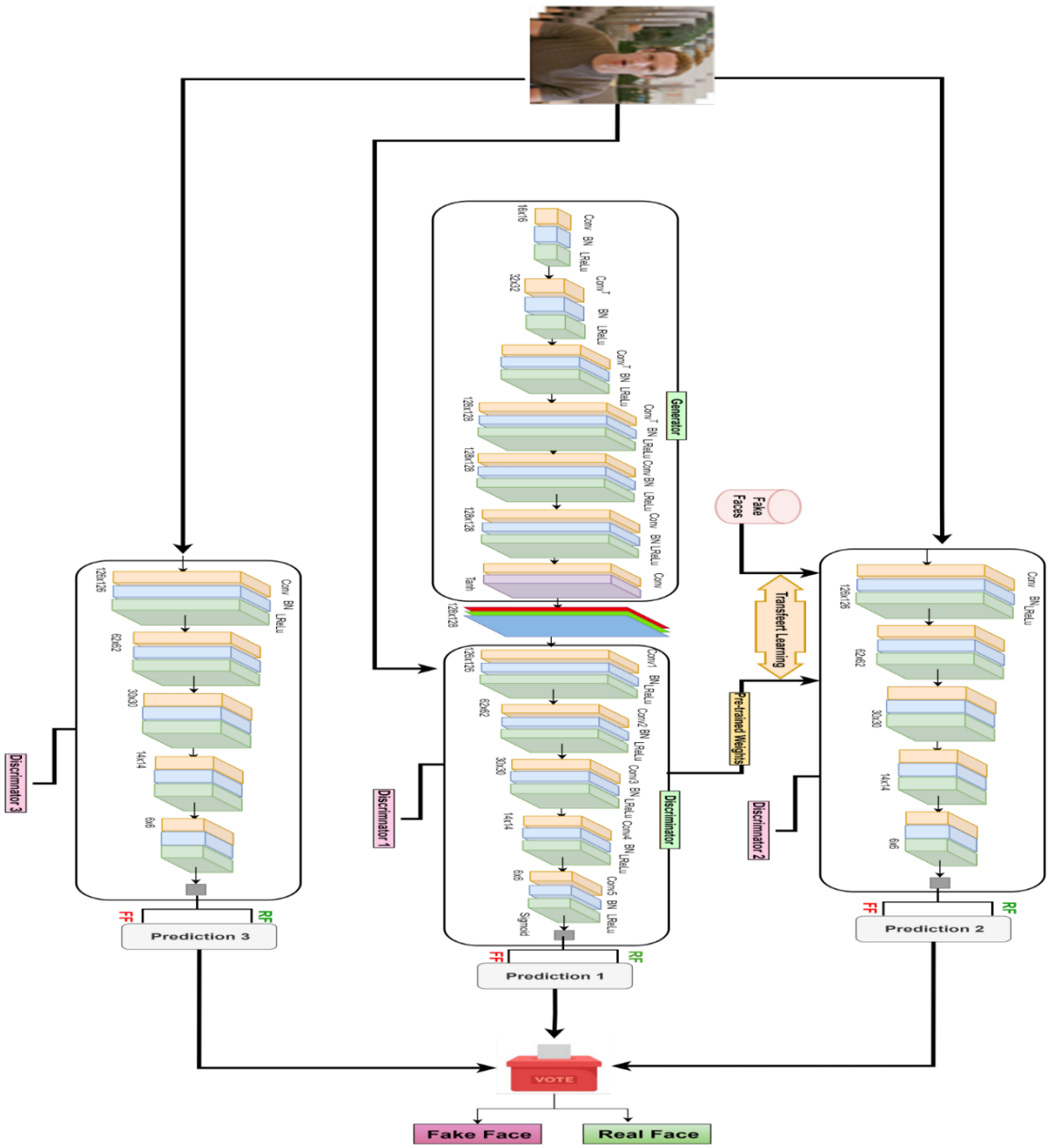


Figure 4. Architecture of the proposed system.

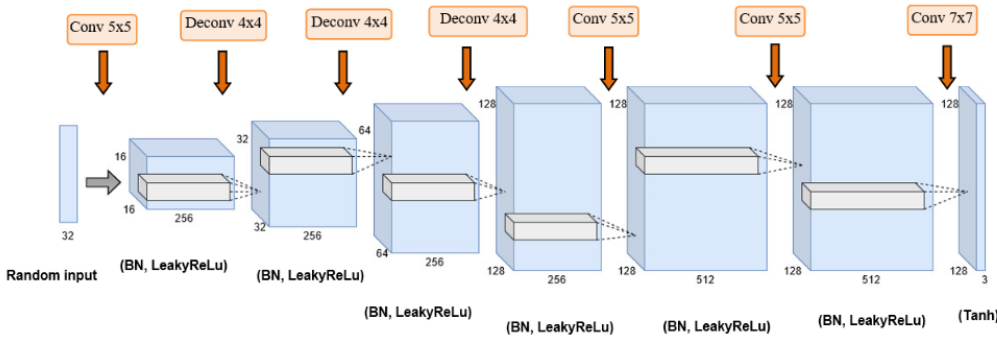


Figure 5. Architecture of the proposed NDCGAN generator.

Layer type	Dimension	Kernel size	Stride	Activation
Input	32	–	–	–
Dense	–	–	–	LeakyReLU
Conv2D	(16, 16, 256)	5	1	LeakyReLU
Conv2DTranspose	(32, 32, 256)	4	2	LeakyReLU
Conv2DTranspose	(64, 64, 256)	4	2	LeakyReLU
Conv2DTranspose	(128, 128, 256)	4	2	LeakyReLU
Conv2D	(128, 128, 512)	5	1	LeakyReLU
Conv2D	(128, 128, 512)	5	1	LeakyReLU
Conv2D	(128, 128, 3)	7	1	tanh

TABLE 1. Architecture of the generator.

Layer type	Dimension	Kernel size	Stride	Activation
Input	(128,128,3)	–	–	–
Conv2D	(126, 126, 256)	3	1	LeakyReLU
Conv2D	(62, 62, 256)	4	2	LeakyReLU
Conv2D	(30, 30, 256)	4	2	LeakyReLU
Conv2D	(14, 14, 256)	4	2	LeakyReLU
Conv2D	(6, 6, 256)	4	2	LeakyReLU
Dense	1	–	–	sigmoid

TABLE 2. Architecture of Discriminator 1.

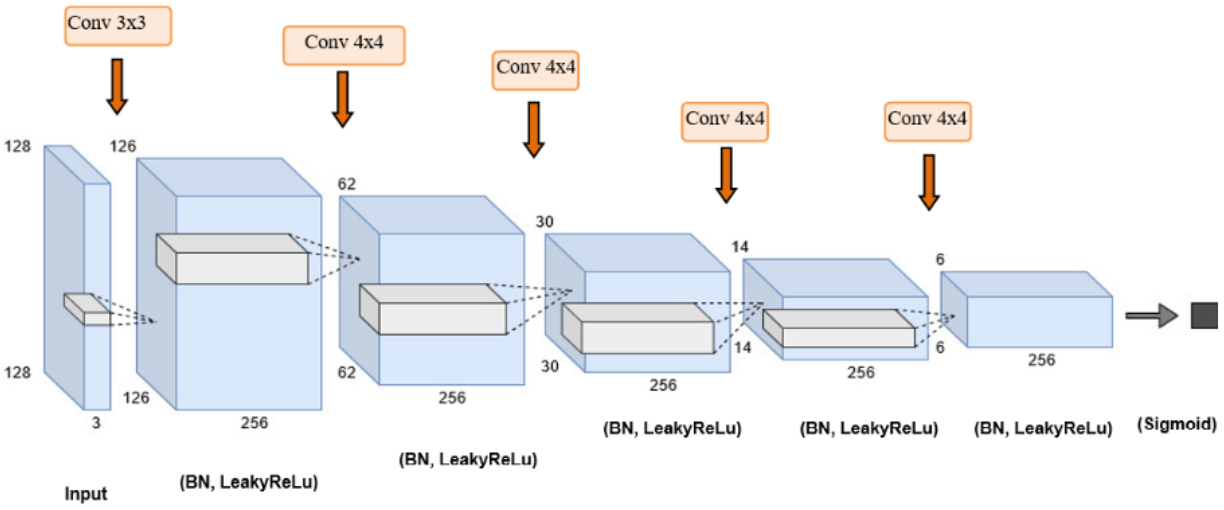
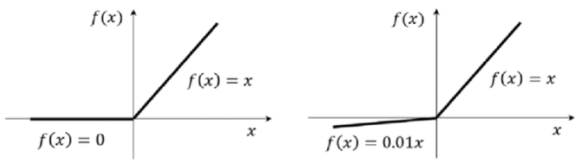


FIGURE 6. Architecture of the proposed NDCGAN discriminator.



(a). ReLU. (b). leakyReLU.

Figure 7. Activation function.

is controlled by a parameter called alpha, which introduces network tolerance by allowing certain negative values to pass.

4. Experimental results

In this section, we provide an overview of the network training and testing process. For our purposes, we chose to use Kaggle [33] rather than Google Colab [34]. We made this choice because of Kaggle's extensive collection of importable datasets and the availability of up to forty GPU/TPU hours per week, all provided free of charge. In contrast, Google Colab does not offer the same level of resource availability at no additional cost.

To train our model, we used the TESLA P100 GPUs available on Kaggle as our training environment, along with Tensorflow; these GPUs are useful for training deep learning models and can speed up training 13 times faster than traditional training on a CPU.

The specifications of CPU runtime provided by Kaggle are Intel Xeon Processor with two cores @2.30 GHz and 16 GB RAM, and 17 GPU RAM. Python was used as the programming language and PyCharm was used as the IDE for coding and development.

After network training and validation, the trained model is imported to run all the tests on a laptop with an Intel Core i7 and 16 Go RAM.

4.1. Dataset

In the training phase, we used a database of 70,000 real faces from the Flickr-Faces-HQ dataset [18] (Figure 8), collected by Nvidia, and 70 000 fake faces (Figure 9), generated using Nvidia's StyleGAN [19]. In this dataset, all images have a dimension of 256 px and are divided into training, validation, and test sets.

4.2. Evaluation metrics

We have implemented the discriminators of our proposed system to compare the overall accuracy and loss. Additionally, to evaluate the effectiveness of our proposed architecture (MDD-CNN), the following conventional evaluation metrics were used:

True Positives (**TP**), which refer to examples correctly labelled as positive;

True Negatives (**TN**), which refer to negative examples correctly labelled as negative

False Positives (**FP**), which refer to negative examples mislabelled as positive;

Figure 8. Flickr-Face-HQ (FFHQ) dataset.

Figure 9. Example of Fake Face dataset.

False Negatives (**FN**), which refer to positive examples mislabelled as negative.

Accuracy (precision) given by Equation (3): is the proportion of observations correctly predicted to the total number of observations.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}. \qquad (3)$$

Loss of discriminator: is the difference between the predicted output and the actual output. It measures the mistakes made by the network in predicting the output.

TruePositiveRate/Recall/Sensitivity: is the proportion of actual positive results that were correctly identified given by Formula (4):

$$TPR = Recall = \frac{TP}{TP + FN}. \qquad (4)$$

Receiver Operating Characteristic Curve (ROC curve) is a graph that shows the performance of a classification model at all classification thresholds.

The curve represents two parameters: True Positive Rate (TPR) (Equation (4)) and False Positive Rate (FPR):

$$FPR = \frac{FP}{FP + VN}\,. \tag{5}$$

4.3. Training and validation

The proposed multi-discriminator architecture (MDD-CNN) requires training in three steps, with each discriminator undergoing a different training. Discriminator 1 undergoes adversarial training, Discriminator 2 undergoes training by transfer learning, and Discriminator 3 undergoes a standard training of a CNN with examples and counterexamples.

During training, certain parameters that are crucial for convergence to the optimal solution were fixed after a series of tests. These parameters include the batch size and the number of epochs.

Batch Size: The batch size refers to the number of training examples presented to the model in each iteration. It has a significant influence on the training process.

- *Computational Efficiency*: A larger batch size enables parallelism and can better utilise hardware acceleration such as GPUs. Training with larger batch sizes can speed up the training process.
- *Generalisation*: Smaller batch sizes often lead to better generalisation but may require more iterations to converge.
- *Memory Constraints*: The batch size must fit within the available memory of the training hardware. Using excessively large batch sizes can exceed the memory capacity and result in out-of-memory errors.

Number of Epochs: An epoch signifies a complete pass through the entire training dataset. The number of epochs determines how many times the model will go through the entire dataset during training. The number of epochs can influence training in various ways:

- *Underfitting vs. Overfitting*: Training for too few epochs can result in underfitting, where the model fails to learn enough from the data. However, training for too many epochs can lead to overfitting, where the model becomes overly specialised to the training data and fails to generalise well to new data.
- *Training Time*: Increasing the number of epochs increases the overall training time. Training deep learning models can be time consuming, so there is typically a trade-off between computational resources and achieving the optimal performance.
- *Learning Rate Schedule*: The number of epochs can also impact the learning rate schedule. Certain training strategies, such as learning rate decay or learning rate warm-up, adjust the learning rate as the number of epochs progresses.

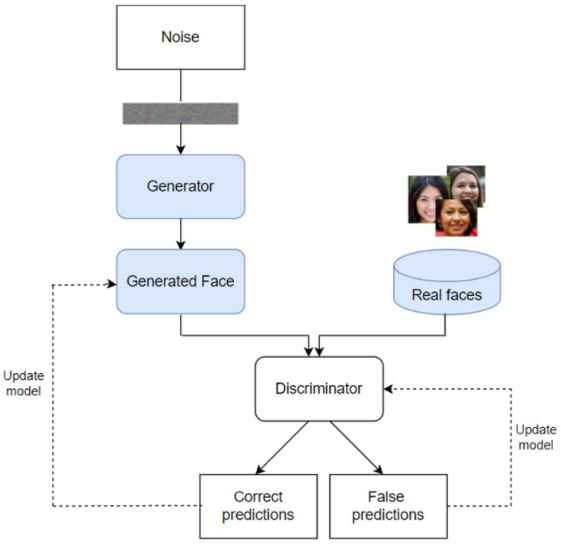


Figure 10. Training of Discriminator 1.

Class	Training	Test
Real Faces	50 000	10 000
Fake Faces	–	10 000

Table 3. Separation of training and test data sets.

Determining the optimal batch size and number of epochs often requires experimentation and tuning based on the specific dataset, model architecture, and problem domain. It is common to begin with default values and iteratively adjust them based on the observed performance.

4.3.1. Training of Discriminator 1

To develop the model of Discriminator 1, we used adversarial learning between generator and discriminator networks (Figure 10). Where the real faces used (60 000) are from the Flickr dataset [18] and the Fake faces are from the StyleGAN dataset [19]. The datasets are divided into training and test as shown in Table 3.

During the training of this network, we kept the following configuration:

- Optimiser = RMSProp which generates more realistic fake images compared to Adam.
- Batch size = 128.
- Number of iterations = 10 000.

During the training, the discriminator tries to maximise the logarithmic probability that the data is real or false. Conversely, the generator tries to minimise the logarithmic probability that the discriminator is correct.

It is therefore considered a min-max game.

The standard GAN loss function, also called the min-max loss function, is given by the following Formula (6) [31]:

$$E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))]\,, \qquad (6)$$

where:

$D(x)$ is an estimate of the probability that the actual data image x is real.

E_x is the expected value over all actual data instances.

$G(z)$ is the output of the generator when it receives a value of z for the noise.

$D(G(z))$ is the estimate of the probability that a false instance is real.

E_z is the expected value over all random inputs of the generator (in fact, it is the expected value over all generated false instances, $G(z)$).

The standard GAN loss function can be further divided into two parts: **Discriminator loss** and **Generator loss**.

Discriminator loss During training, the discriminator classifies both the real face and the fake face from the generator and penalises itself for misclassifying a real face as fake, or a fake face as real, by maximising the Function (7):

$$\nabla_{\theta d}\frac{1}{m}\sum_{i=1}^{m}\left[\log D(x^{(i)}) + \log(1 - D(G(z^{(i)})))\right]\,, \quad (7)$$

where:

- $\log(D(x))$ refers to the probability that the generator is correctly classifying the real face,
- maximising $\log(1 - D(G(z)))$ would help it to correctly label the fake face that comes from the generator.

Generator loss The discriminator's classification is used to calculate the generator loss. If succeeds in fooling the discriminator, it is rewarded; if not, it is penalised.

Equation (8) is minimised to train the generator.

$$\nabla_{\theta g}\frac{1}{m}\sum_{i=1}^{m}\log(1 - D(G(z^{(i)})))\,. \qquad (8)$$

Figure 11 shows the training results of the generator and the discriminator. We can see that the discriminator network achieved a loss for real and fake samples of around 47 % and the generator of around 96 %. The discriminator achieved an accuracy of between 70 % and 79 % for real and fake faces.

The learning results are acceptable because the final decision is a contribution by voting of the decisions of the three discriminators. Therefore, the decision of Discriminator 1 is either confirmed or contradicted by the other two discriminators, which form the overall architecture of our MDD-CNN system.

Class	Training	Validation	Test
Real Faces	50 000	10 000	10 000
Fake Faces	50 000	10 000	10 000
Total	100 000	20 000	20 000

Table 4. Discriminator 2 training and test data separation.

4.3.2. Training of Discriminator 2

Discriminator 2 is the model of Discriminator 1, previously trained with its opponent, the generator, and further trained by transfer learning on a dataset made up of real and fake faces (Figure 12).

During the training of this network, we kept the following configuration:

Optimizer = RMSProp.

Batch size = 128.

Number of epochs = 30.

Figure 12 shows the main training phases of the discriminator. The distribution of the dataset for training and test is described according to Table 4.

Figure 13 shows the training results of the discriminator. We can see that the network achieved an accuracy rate of 98 % during training and 94 % during testing, with a loss of 0.05 % during training and 0.12 % during testing, which is considered as acceptable for the detection of fake faces.

4.3.3. Training of Discriminator 3

The architecture of Discriminator 3 is the same as that of Discriminator 1, trained from scratch on real faces and fake faces without its generator.

During the training of this network, we kept the following configuration:

Optimizer = RMSProp.

Batch size = 128.

Number of epochs = 45.

Figure 14 shows the main training phases of the Discriminator 3. The distribution of the training dataset is the same as for the Discriminator 2 (Table 4).

From the graphs in Figure 15, we can see that the network achieved an accuracy rate of 97 % during training and 94 % during testing, with a loss of 0.06 % during training while the estimated test loss remained within an acceptable range of 0.14 % for effective detection of fake faces.

4.4. Test and analysis

The tests with the three discriminators were carried out with datasets of 20 000 images (fake and real), that had not been presented to the networks before. The results obtained by each network are shown in Tables 5.

Table 6 shows the true and false positive and negative examples. We observed that Discriminator 2 is

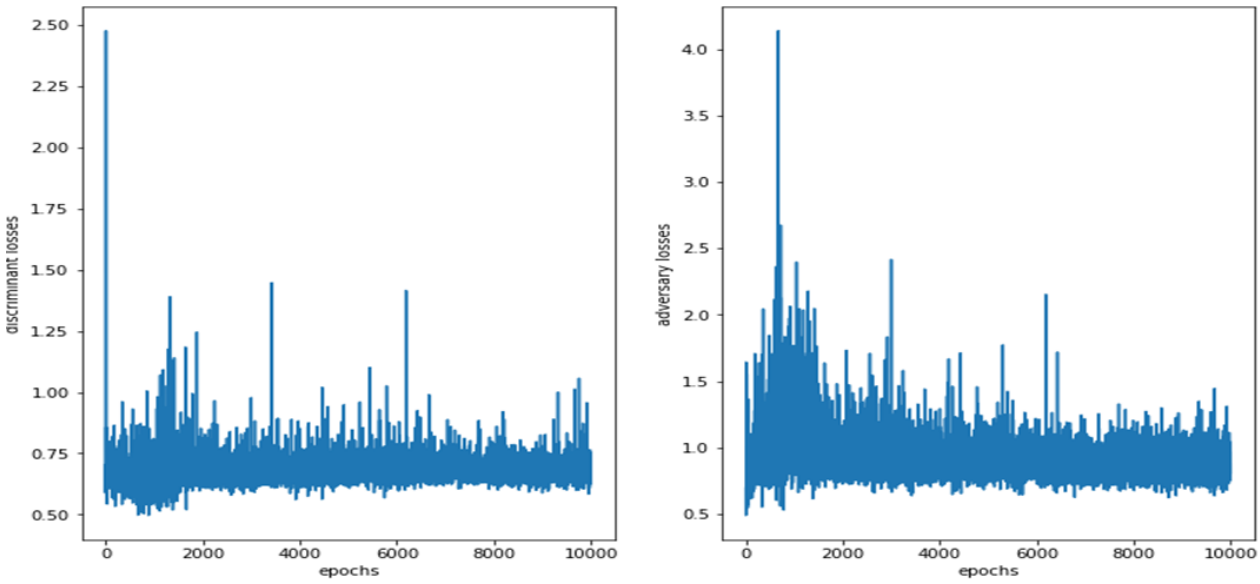


FIGURE 11. Discriminator and generator Loss.

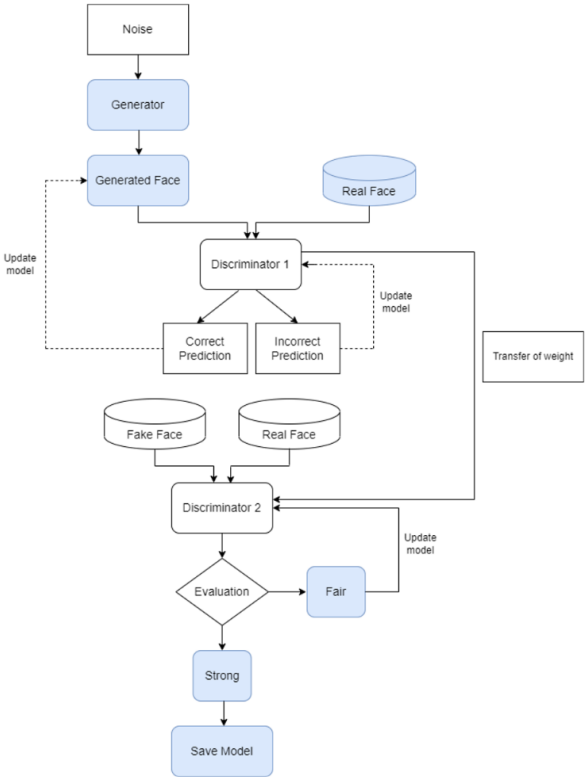


FIGURE 12. Training of Discriminator 2.

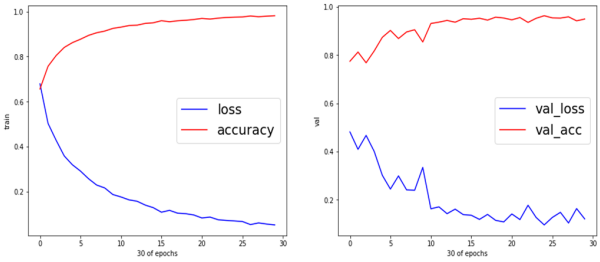


FIGURE 13. Accuracy and Loss during training and validation of Discriminator 2.

better at discriminating negative examples (fake face) with a TN of 98 % than positive examples (real face) with a TP of 92 %. Discriminator 3 is then better at discriminating real faces, with a TP of 98 %, than fake faces, with a TN of 93 %.

Table 7 shows the precision and recall of the three networks obtained with real and fake faces datasets.

The Discriminator 2 recognises the real faces with an accuracy of 0.98, exceeding that of fake faces by 0.05. Conversely, for Discriminator 3, the recognition accuracy of fake faces (0.98) exceeds that of the real faces (0.05).

With the accuracies obtained, the discriminators of our system manage to recognise more than 93 % of the examples presented. It should be noted that 5 % of the examples not recognised by one of the discriminators are recognised by the others, which is advantageous for our multi-discriminator because its final decision is obtained by voting between the decisions of the three discriminators, where the final result is obtained by a majority vote which is not always possible in the case of an even number of discriminators.

Figure 16 shows the ROC curves of the three discriminators and the multi-discriminator (MDD-CNN) allowing the performances of the four architectures to be compared. We can see that the MDD-CNN curve deviates slightly from Discriminator 2 (best discriminator of fakes) and occupies the upper part of the ROC space indicating the best classifier among the four with a TPR around 97.8 %.

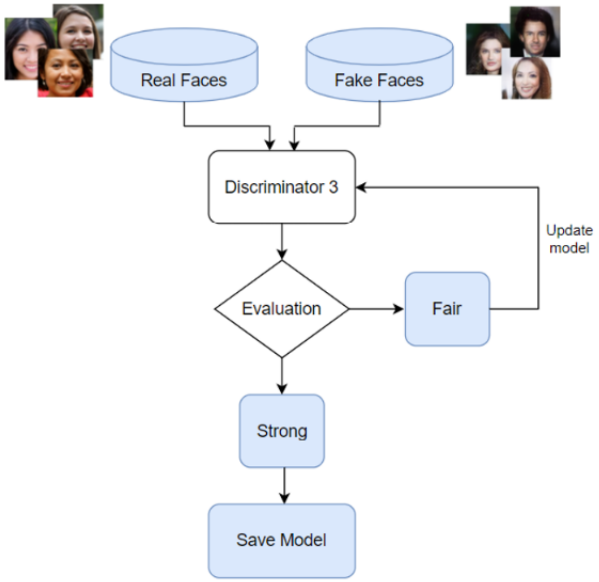


Figure 14. Training of Discriminator 3.

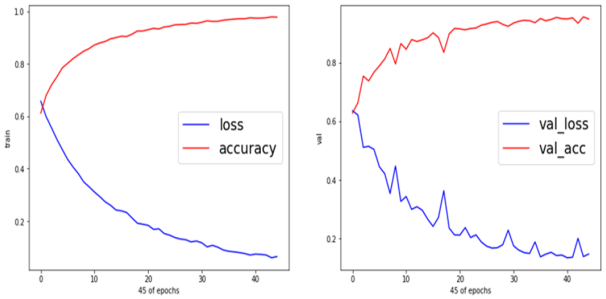


Figure 15. Accuracy and Loss during training and validation of Discriminator 3.

	TP	FN	TN	FP
Discriminator 1				
Image count	8483	1517	7013	2987
Ratio on total count	0.8483	0.1517	0.7013	0.2987
Percentage	84.8 %	15.2 %	70.1 %	29.9 %
Discriminator 2				
Image count	9823	177	9205	795
Ratio on total count	0.9823	0.0177	0.9205	0.0795
Percentage	98 %	2 %	92 %	8 %
Discriminator 3				
Image count	9763	237	9277	723
Ratio on total count	0.9763	0.02	0.9277	0.0723
Percentage	98 %	2 %	93 %	7 %

Table 5. Performance of Discriminators 1, 2 and 3.

	TP	TN	FP	FN
Discriminator 1	84.8 %	15.2 %	70.1 %	29.9 %
Discriminator 2	92 %	98 %	2 %	8 %
Discriminator 3	98 %	93 %	7 %	2 %

Table 6. Performance comparisons of the three discriminators.

In order to test the performance of our MDD-CNN architecture, we have carried out several tests on images of real faces and false faces randomly introduced into the system, the results are shown in Figure 17 with an indication for each image of the prediction and the ground truth class (true or false faces).

Thanks to the combination of the performance of the three discriminators, the MDD-CNN architecture has given high discrimination rates (97.8 %) thus exceeding the state of the art with a minimum deviation of 0.8 % and a maximum of 17 % (Table 8).

5. Conclusion

In recent years, deepfake technologies have gained significant popularity and pose a threat to our security as they can be exploited by criminals for fraudulent activities. Consequently, the detection of fake content has become a crucial research area, given the proliferation of manipulated or fabricated data in various forms, such as images, videos, and sounds.

Distinguishing fake faces from real ones is both a challenging and a delicate task. It requires a high-performance discrimination system with considerable accuracy to distinguish fake faces from real ones. This need led to the development of the multi-discriminator architecture proposed in this study. This architecture consists mainly of three discriminators whose main task is to detect false faces. Each network provides a decision weighted by the probability of belonging to the class of real or fake faces. The decision of each network represents a vote that will contribute to the voting of all the networks to provide a final decision (an opinion) on the class of the image.

The three discriminators have been trained differently, whith Discriminator 1 being part of the architecture that we have proposed for the new DCGAN resulting from the transformation of a GAN network from a generator of fake faces, with a generator that manages to deceive its discriminator to a detector of fakes, with a discriminator that manages to recognise the fake faces of the generator. Although the discriminator of the NDCGAN network gave good results with a precision of 79 %, it is still insufficient as the loss exceeds 40 %. The loss was significantly reduced by transfer learning, which was used in the case of Discriminator 2, resulting in the model having the

		Support	Precision	Recall
Discriminator 1	Fake	10 000	0.79	0.77
	Real	10 000	0.80	0.82
Discriminator 2	Fake	10 000	0.93	0.98
	Real	10 000	0.98	0.92
Discriminator 3	Fake	10 000	0.98	0.93
	Real	10 000	0.93	0.98

TABLE 7. Precision and recall of the three discriminators.

Reference	Dataset	Accuracy
Z. Liu et al. [17]	StyleGAN CelebA-HQ	80.72 %
T. Jung et al. [22]	GAN DF	87.5 %
P. Korshunov & S. Marcel [23]	Vid TIMIT	91.03 %
Y. Li & S. Lyu [26]	Deep Fake UADFV	97 %
X. Yang et al. [27]	HFF	95.17 %
A. Chintha et al. [28]	Face forensics Celeb DF	84.8 %
S. Lee et al. [30]	GAN DF	93.99 %
Our work (MDD-CNN)	Flickr-Faces-HQ Nvidia Style GAN FF	97.8 %

TABLE 8. Comparison of MDD-CNN performance with state of the art.

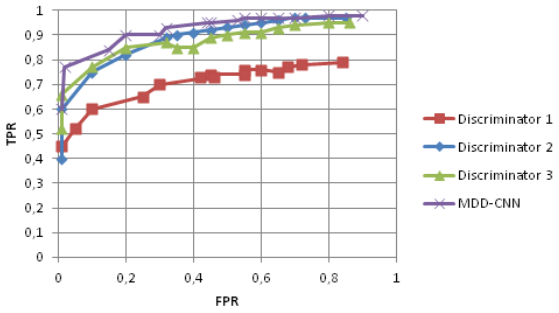


FIGURE 16. ROC Curve of the four classifiers.

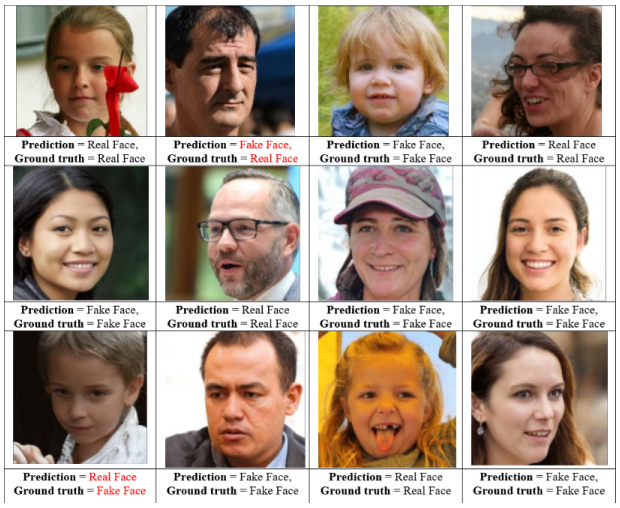


FIGURE 17. Obtained results of our MDD-CNN.

best fake face detection rate (98 %). For the detection of real faces, the best rates were obtained by Discriminator 3 (98 %), which underwent supervised training with positive and negative examples.

The performance of the three discriminators combined in the multi-discriminator architecture allowed us to obtain remarkable detection rates (around 98 %) for the two classes.

The results obtained with the proposed architecture (MDD-CNN) outperform the state of the art by 0.8 %–17 %. These results remain preliminary and need to be deepened by testing our architecture on other fake datasets.

In addition to this study, in the future we plan to conduct various tests on more datasets of images and videos containing fake faces in order to implement a system for detecting fakes in images and videos manipulated in social media.

ACKNOWLEDGEMENTS

This project is supported by the LAIG laboratory and the 8 Mai 1945-University of Guelma – `http://www.univ-guelma.dz`.

REFERENCES

[1] C. Bregler, M. Covell, M. Slaney. Video rewrite: Driving visual speech with audio. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pp. 353–360. 1997. `https://doi.org/10.1145/258734.258880`

[2] T. F. Cootes, G. J. Edwards, C. J. Taylor. Active appearance models. *IEEE Transactions on Pattern*

Analysis and Machine Intelligence **23**(6):681–685, 2001. https://doi.org/10.1109/34.927467

[3] A. Santha. Deepfakes generation using LSTM based generative adversarial networks. Master Thesis, Rochester Institute of Technology, 2020.

[4] J. Thies, M. Zollhofer, M. Stamminger, et al. Face2Face: Real-time face capture and reenactment of RGB videos. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2387–2395. 2016. https://doi.org/10.1109/CVPR.2016.262

[5] S. Suwajanakorn, S. M. Seitz, I. Kemelmacher-Shlizerman. Synthesizing Obama: Learning lip sync from audio. *ACM Transactions on Graphics* **36**(4):1–13, 2017. https://doi.org/10.1145/3072959.3073640

[6] M. Westerlund. The emergence of deepfake technology: A review. *Technology Innovation Management Review* **9**(11):39–52, 2019. https://doi.org/10.22215/timreview/1282

[7] F. Marra, D. Gragnaniello, L. Verdoliva, G. Poggi. Do GANs leave artificial fingerprints? In *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 506–511. 2019. https://doi.org/10.1109/MIPR.2019.00103

[8] A. Rössler, D. Cozzolino, L. Verdoliva, et al. FaceForensics++: Learning to detect manipulated facial images. [2023-05-31]. arXiv:1901.08971

[9] Y. Li, X. Yang, P. Sun, et al. Celeb-DF: A large-scale challenging dataset for deepfake forensics. [2023-05-31]. arXiv:1909.12962v4

[10] G. Mahfoudi, B. Tajini, F. Retraint, et al. DEFACTO: Image and face manipulation dataset. In *27th European Signal Processing Conference (EUSIPCO)*, pp. 1–5. 2019. https://doi.org/10.23919/EUSIPCO.2019.8903181

[11] P. Zhou, X. Han, V. I. Morariu, L. S. Davis. Two-stream neural networks for tampered face detection. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1831–1839. 2017. https://doi.org/10.1109/CVPRW.2017.229

[12] Z. Guo, G. Yang, J. Chen, X. Sun. Fake face detection via adaptive manipulation traces extraction network. *Computer Vision and Image Understanding* **204**:103170, 2021. https://doi.org/10.1016/j.cviu.2021.103170

[13] Z. Alom, T. M. Taha, C. Yakopcic, et al. A state-of-the-art survey on deep learning theory and architectures. *Electronics* **8**(3):292, 2019. https://doi.org/10.3390/electronics8030292

[14] K. G. Kim. Book review: Deep learning. *Healthcare Informatics Research* **22**(4):351–354, 2016. https://doi.org/10.4258/hir.2016.22.4.351

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

[16] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld. Learning realistic human actions from movies. In *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8. 2008. https://doi.org/10.1109/CVPR.2008.4587756

[17] Z. Liu, X. Qi, P. H. S. Torr. Global texture enhancement for fake face detection in the wild. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8057–8066. 2020. https://doi.org/10.1109/CVPR42600.2020.00808

[18] T. Karras, S. Laine, T. Aila. A style-based generator architecture for generative adversarial networks. [2023-05-31]. arXiv:1812.04948

[19] T. Karras, T. Aila, S. Laine, J. Lehtinen. Progressive growing of gans for improved quality, stability, and variation. [2023-05-31]. arXiv:1710.10196

[20] A. Deshmukh, S. B. Wankhade. Deepfake detection approaches using deep learning: A systematic review. In *Intelligent Computing and Networking. Lecture Notes in Networks and Systems*, vol. 146. 2021. https://doi.org/10.1007/978-981-15-7421-4_27

[21] S. Albawi, T. A. Mohammed, S. Al-Zawi. Understanding of a convolutional neural network. In *International Conference on Engineering and Technology (ICET)*, pp. 1–6. 2017. https://doi.org/10.1109/ICEngTechnol.2017.8308186

[22] T. Jung, S. Kim, K. Kim. DeepVision: Deepfakes detection using human eye blinking pattern. *IEEE Access* **8**:83144–83154, 2020. https://doi.org/10.1109/ACCESS.2020.2988660

[23] P. Korshunov, S. Marcel. Vulnerability assessment and detection of deepfake videos. In *International Conference on Biometrics (ICB)*, pp. 1–6. 2019. https://doi.org/10.1109/ICB45273.2019.8987375

[24] O. Parkhi, A. Vedaldi, A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, pp. 1–12. 2015. https://doi.org/10.1109/ICB45273.2019.8987375

[25] F. Schroff, D. Kalenichenko, J. Philbin. FaceNet: A unified embedding for face recognition and clustering. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823. 2015. https://doi.org/10.1109/CVPR.2015.7298682

[26] Y. Li, S. Lyu. Exposing DeepFake videos by detecting face warping artifacts. [2023-05-31]. arXiv:1811.00656

[27] X. Yang, Y. Li, S. Lyu. Exposing deep fakes using inconsistent head poses. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8261–8265. 2019. https://doi.org/10.1109/ICASSP.2019.8683164

[28] A. Chintha, B. Thai, S. J. Sohrawardi, et al. Recurrent convolutional structures for audio spoof and video deepfake detection. *IEEE Journal of Selected Topics in Signal Processing* **14**(5):1024–1037, 2020. https://doi.org/10.1109/JSTSP.2020.2999185

[29] F. Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807. 2017. https://doi.org/10.1109/CVPR.2017.195

[30] S. Lee, S. Tariq, Y. Shin, S. S. Woo. Detecting handcrafted facial image manipulations and gan-generated facial images using shallow-fakefacenet. *Applied Soft Computing* **105**:107256, 2021. https://doi.org/10.1016/j.asoc.2021.107256

[31] I. J. Goodfellow, J. Shlens, C. Szegedy. Explaining and harnessing adversarial examples. [2023-05-31]. arXiv:1412.6572

[32] A. Radford, L. Metz, S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. [2023-05-31]. arXiv:1511.06434

[33] A. Mangal, N. Kumar. Using big data to enhance the bosch production line performance: A Kaggle challenge. In *IEEE International Conference on Big Data (Big Data)*, pp. 2029–2035. 2016. https://doi.org/10.1109/BigData.2016.7840826

[34] T. Carneiro, R. V. Medeiros Da NóBrega, T. Nepomuceno, et al. Performance analysis of Google colaboratory as a tool for accelerating deep learning applications. *IEEE Access* **6**:61677–61685, 2018. https://doi.org/10.1109/ACCESS.2018.2874767