# Deepfake Detection Using CNN Trained on Eye Region

David Johnson [1][0000-0001-8504-4501], Tony Gwyn[1][0000-0003-4625-0040], Letu Qingge[1], and Kaushik Roy[1]

[1] North Carolina A&T State University, Greensboro NC 27411, USA
dmjohns8@aggies.ncat.edu, tgwyn@aggies.ncat.edu,
lqingge@ncat.edu, kroy@ncat.edu

**Abstract.** In this work, we will develop a simple convolutional neural network to detect deepfakes in videos on a frame-by-frame level, focusing on the region around the eyes. Since deepfakes are increasingly being created using forms of CNN, it should be possible to also detect deepfakes using CNN. OpenCV allows for frame extraction from videos, while also allowing image cropping. The well-developed Multitask Cascade Neural Network (MTCNN) is a stacked neural network for face detection and alignment. MTCNN is used for high accuracy face detection to greatly reduce false positive images in the dataset. Finally, a region around both eyes are cropped, with extra padding, to be used as input to train a CNN, using returned coordinates from MTCNN for the eyes. This research will focus on measuring if the eye region can be a useful area of interest for comparing original videos to deepfake videos.

**Keywords:** Deep Learning, Deepfake, deepfake detection, CNN, computer vision, eye region

## 1 Introduction

Deepfakes are manipulated images or videos that usually are an attempt to affect an individual's image or reputation, or deepfakes are used to spread disinformation. While they are somewhat easy to distinguish based on the viewer, they are becoming rampant because there are many times where they are difficult to tell if the video is real or fake. The goal of this research is to use deep learning to create a convolutional neural network (CNN) trained in distinguishing between real and fake images to then be able to detect manipulated image sequences in videos. There are a few public datasets that can be used to train a model. For this research, the dataset DeepFakeDetection [1] from the FaceForensics++ collection [1] will be used. DeepFakeDetection [1] consists of one thousand original videos, and three thousand manipulated videos of the originals. While deepfakes can be used to alter both visual and auditorial information, this research will only focus on the visual information from the videos. There exist patterns within deepfakes that can be learned by a CNN [2] [3] to be able to detect potential manipulation, such as the boundary where a fake face is placed over an original.

**Fig. 1.** An original extracted face (right) and the same face, manipulated at the same frame (left). The left image is an example of a deepfake from the DeepFakeDetection dataset [1] [4].

## 2 Background

Deepfakes are popularly created using Generative Adversarial Networks (GAN), which creates data after training, and validates against itself to increase realism [5]. In some of the more rudimentary forms, deepfakes present themselves as a false face superimposed on an original, unmanipulated face. Oftentimes, there is some brushing involved to allow better edge blending, this can be seen in Figure 1. This style of deepfake creation is famously used online with the actor Nicolas Cage's face being superimposed on various people [6], such as other celebrities or world leaders. The increase in hyperrealism from deepfakes also leads to the potential for mis-, or disinformation, to spread. An example of a deepfake created using improved GAN technology would be the video of U.S. President Barack Obama [7] manipulated by the director and comedian Jordan Peele. While these examples show the humor and potential behind deepfakes, oftentimes, they are used as propaganda. Jordan Peele used his deepfake as a way to show how videos on the Internet may not always be original, "good faith" videos, but rather disinformation. Some states have gone to pass laws that make the use of deepfakes for political purposes, although enforcement remains problematic. Deepfakes are created using a type of CNN [5], and typically includes artifacts in the manipulated video that can be detected either by a person or by machine. Below are previous works that have also used CNN detect anomalies and artifacts based on the features from extracted images. The datasets used in the related works are not the same use here, although some use datasets from the same collection, FaceForensics++ [1], others use datasets created from different deepfake methods.

# 3 Related Works

Deepfakes, in their fundamental nature, are vectors for misinformation. Given their potential, it would be good to have ways to detect various types of deepfakes and their respective patterns in images. The following sections include various works related to the discussed problem.

## 3.1 Convolutional Neural Networks

Convolutional Neural Networks (CNN) are deep learning models that are commonly used on image data, or data that can be processed as image data. There have been previous works at constructing CNNs to detect deepfakes on different datasets, either by classifying against entire faces [5] or by focusing on specific extracted regions [2] [3]. In building a simple CNN based around the face region on the Celeb-DF dataset, classification close to 70% could be achieved [8]. Other methods of deepfake detection, such as using Scale-Invariant Feature Transform (SIFT) with CNN, which focus on specific regions of the face using SIFT [2], then using those results as input for a CNN, produced classification accuracies closer to 93%. By focusing on a specific region of the extracted face, a CNN can be constructed to detect deepfakes with greater accuracy than training on the entire face.

## 3.2 Generative Adversarial Networks

Generative adversarial networks (GAN) are deep learning models consisting of paired neural networks, or sub-models, that work against each other to produce realistic images [5]. The paired neural networks, the generator and the discriminator, tend to be CNNs, where the generator creates fake data that looks real, while the discriminator classifies real and fake data. When training with paired sub-models, a generator can create fake images, which are passed to the discriminator, where a classification is predicted. Based on the result, the generator will adjust to try to "fool" the discriminator into classifying fake data as real. This tandem generation and classifying leads to realistic images that are difficult to distinguish from original, unmanipulated images.

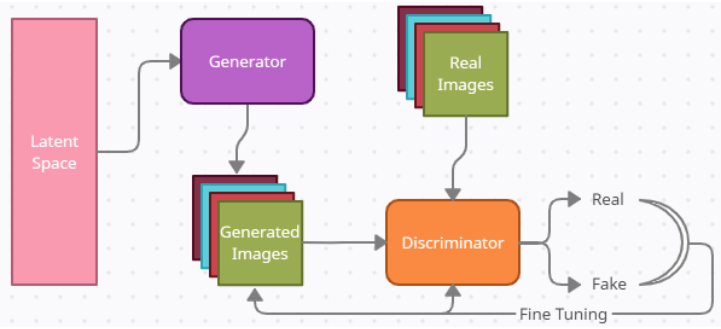


**Fig. 2.** Deepfake creation using GAN [8].

# 4 Materials & Methodology

Here, the tools used and methodology will be outlined to classify deepfakes using a CNN.

## 4.1 Dataset

The dataset used comes from the FaceForensics++ collection of deepfakes [1]. The current collection contains six sets of videos produced in different styles of manipulation. A few of the datasets are Face2Face, DeepFakeDetection, FaceSwap, and Deepfakes. The DeepFakeDetection [1] dataset will be used as the source for videos. The FaceForensics++ collection [1] consists of 1000 original videos that have been manipulated to produced thousands more. In the DeepFakeDetection dataset [1], there are 1000 original videos, and 3000 manipulated videos. To balance original and manipulated videos, every third manipulated video is used for every original video.

## 4.2 Frame Extraction

Next, using Python [9] and the OpenCV [10] library to load the video frame-by-frame, a frame is extracted every 100ms and 40 frames are extracted per video. The frames are then used as input to a face detection neural network for high accuracy face detection in each frame. Frames without a high-confidence face are ignored.

## 4.3 Face and Eye Region Extraction

The face detection neural network is Multitask Cascaded Neural Network (MTCNN) [11], which is a model consisting of three stacked neural networks for bounding box regression, face detection and alignment, and facial landmark location. MTCNN's face detection component is used to detect faces in extracted frames. Faces with a confidence greater than 99% are retained in the extracted faces database, while other images are ignored. MTCNN returns coordinates for facial landmarks, such as the eyes, nose, and mouth. Using the coordinates of the eyes, a bounding box is created to contain both eyes at either end. The bounding boxes width and height are padded with 15 and 10 pixels, respectively, to increase the extracted region to also include potential regions of manipulation and the bounding box edges from face swapping. After extraction, the image database to be used as input to train a CNN consists of shape (40498, 20, 100, 3), where 20,249 RGB images are original and 20,249 RGB images are manipulated.

## 4.4 Construct Convolutional Neural Network

CNNs are great tools for classifying image data. *Keras*, within the TensorFlow platform [12], is used to build the model. In creating a CNN, the input dataset must be split into training and testing sets. The dataset is split with a ratio of 90% training and 10% testing, or 36,449 training images and 4,049 testing images. Training is set to 200 epochs,

using the Adam optimizer with a learning rate of 0.0001. The proposed CNN model consists of six layers: an input layer, three hidden layers, a dense layer, and finally an output layer. To reduce overfitting, dropout is utilized between each layer, with a rate of 0.3. Max pooling layers also proceed every convolutional layer. Accuracies should exceed 80% to be considered statistically relevant.

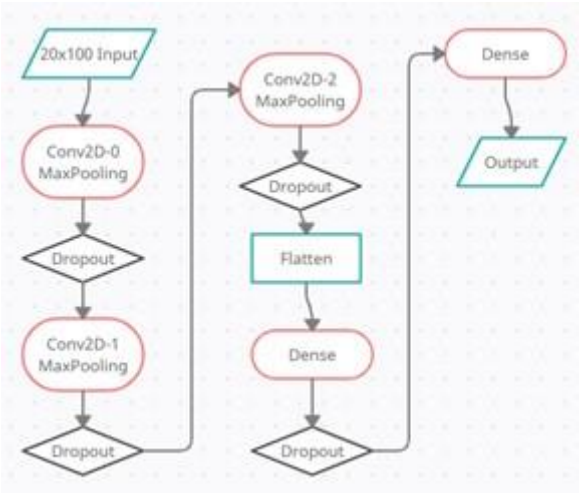


**Fig. 3.** Architecture of CNN model used

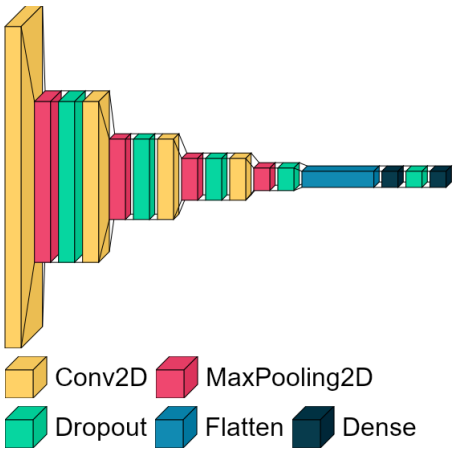


**Fig. 4.** Visualization of proposed CNN model

### 4.5 Compare Test Results

It is worth noting that the construction of a CNN model does not have to be limited to the dataset originally designed in mind. Given a collection of datasets, where each dataset is derived from the same set of original videos, in the case of FaceForensics++ [1], it would be fairy simple to retrain the model against similar, but different data. Since manipulated data come from the same source, the original data does not need to be reextracted, just the manipulated data. The new, manipulated data, along with the original data, are used to train the same CNN model-concept to compare metrics.

## 5 Results & Conclusion

Using the proposed CNN architecture, a model was created on the DeepFakeDetection dataset [1] with accuracy of about 98.3% at epoch 200. At epoch 50, validation accuracy tends to be around 90% but continues to increase, even beyond 200 epochs. Although training could continue beyond 200 epochs, accuracy and loss were not improving enough to warrant the extended training. Considering deepfakes are created using technologies based on CNN, it would seem possible to construct a CNN that could detect deepfakes, and this research shows that it is possible to detect deepfakes with strong confidence. The proposed CNN also provided significant results on other datasets from the FaceForensics++ collection [1], such as Face2Face [1], FaceSwap [1], and NeuralTextures [1]. Some datasets may need additional work on the CNN to increase positive detection rates, such as with detection in the NeuralTextures dataset [1] within FaceForensics++ [1].

This research focused on the creation of a simple CNN, which could be a helpful tool to determine if an image has been manipulated. By restricting images to specific, feature-containing regions, such as around the eyes, a high-accuracy CNN can be created to detect and potentially help mitigate the use of false media for disinformation, while still allowing the creative use of generative adversarial networks.

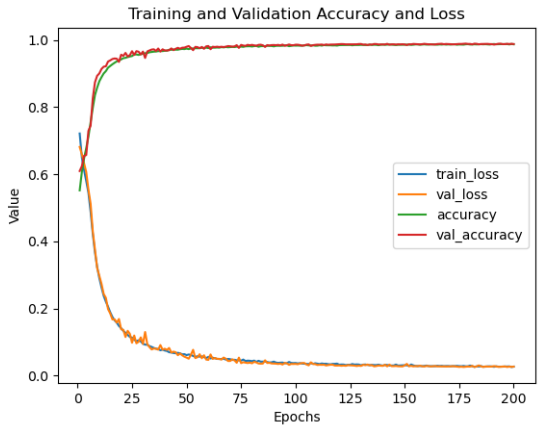

**Fig. 5.** Training and Validation accuracy/loss curves

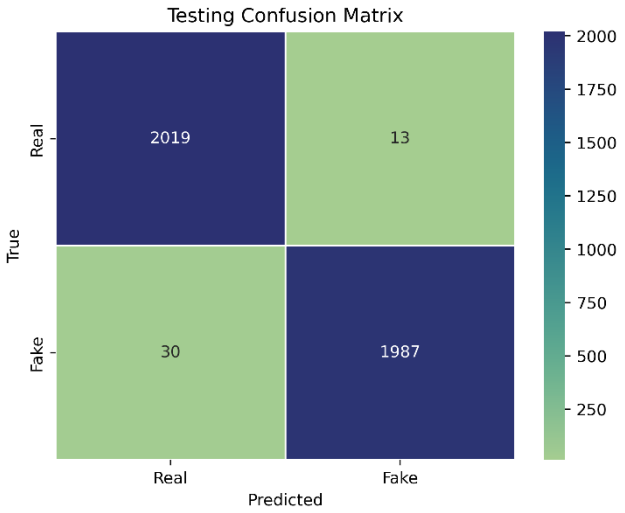


**Fig. 6.** Resulting Confusion Matrix

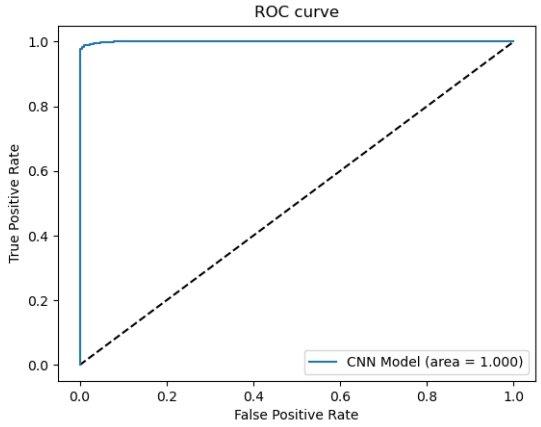


**Fig. 7.** ROC curve

| K-Fold Accuracies | |
|---|---|
| 0 | 98.07 |
| 1 | 98.32 |
| 2 | 98.44 |
| 3 | 98.05 |
| 4 | 98.27 |

| | |
|---|---|
| 5 | 98.52 |
| 6 | 98.47 |
| 7 | 98.02 |
| 8 | 98.59 |
| 9 | 97.68 |
| | |
| Mean | 98.24 |
| St. Dev. | 0.27 |

**Table 1.** Accuracy results from k-fold validation, k=10

| Dataset Average Accuracies | |
|---|---|
| DeepFakeDetection | 98.3% |
| FaceSwap | 97.8% |
| Face2Face | 86.5% |
| NeuralTextures | 67.2% |

**Table 2.** Accuracy results comparison between datasets on same CNN model architecture

## 6 Acknowledgements

This research is supported by National Science Foundation (NSF). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF.

## References

1. *Face{F}orensics++: Learning to Detect Manipulated Facial Images.* Rössler, Andreas, et al. s.l. : International Conference on Computer Vision (ICCV), 2019.
2. Burroughs, Sonya, Roy, Kaushik and Gokaraju, Balakrishna. Detection Analysis of DeepFake Technology by Reverse Engineering Approach (DREA) of Feature Matching. *Springer in Algorithm for Intelligent System (AIS).* s.l. : International Conference on Machine Intelligence and Smart Systems, 2020.
3. Wodajo, Deressa and Atnafu, Solomon. Deepfake Video Detection Using Convolutional Vision Transformer. *arxiv.org.* March 11, 2021. https://arxiv.org/pdf/2102.11126.pdf.
4. (Jigsaw), Nick Dufour (Google Research) and Andrew Gully. Contributing Data to Deepfake Detection Research. *Google Blog.* Google, September 24, 2049. https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html.
5. *Fake Faces Identification via Convolutional Neural Network.* Huaxiao Mo, Bolin Chen, Weiqi Luo. Innsbruck : Association for Computing Machinery, 2018. IH&MMSec '18: Proceedings of the 6th ACM Workshop on Information Hiding and Multimedia Security. pp. 43-47.
6. derpfakes. Nicolas Cage | Mega Mix Two | Derpfakes. *YouTube.* February 02, 2019. https://www.youtube.com/watch?v=_Kuf1DLcXeo.
7. Buzzfeed Video, Jordan Peele. You Won't Believe What Obama Says In This Video! 😉. *YouTube.* April 17, 2018. https://www.youtube.com/watch?v=cQ54GDm1eL0.
8. *Deepfake Video Detection Using Convolutional Neural Network.* A. Karandikar, V. Deshpande, S. Singh, S. Nagbhidkar, S. Agrawal. 2020, International Journal of Advanced Trends in Computer Science and Engineering.
9. *Python.* https://www.python.org/.
10. *OpenCV.* https://opencv.org/.
11. Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks.* 2016.
12. *Tensorflow.* https://www.tensorflow.org/.
13. Rossler, Andreas, et al. FaceForensics++: Learning to Detect Manipulated Facial Images. *The CVF.* 2019. https://openaccess.thecvf.com/content_ICCV_2019/papers/Rossler_FaceForensics_Learning_to_Detect_Manipulated_Facial_Images_ICCV_2019_paper.pdf.
14. Dataset, FaceForensics++. *GitHub.* https://github.com/ondyari/FaceForensics.