Fake News? Deepfake Detection

Yifan Jiang
University of Illinois Urbana-Champaign
yifanj4@illinois.edu

Mingzhi Li
University of Illinois Urbana-Champaign
mingzhi2@illinois.edu

Abstract

Recently, image generation and manipulation are growing rapidly with skyrocketing research being done in the computer vision and artificial intelligence area. [1] Currently, there are many open-source tools and training packages on the internet for people to experience and create images [3] or even videos, ranging from animates to paintings, portraits to landmarks. Deep fake technology, in particular, can potentially spread fake news and misinformation. Together with other digital manipulations like voice or AI text generation, these will lead to more severe ethical problems or crimes. [2] So in this project, our team tried to implement deepfake detection. Though our implementation did not beat state-of-the-art models in terms of performance [6], we still gained various results that can be improved with further study. Our approach involves using a combination of traditional computer vision techniques and deep learning algorithms to detect and classify manipulated images. Specifically, we utilized convolutional neural networks to identify and differentiate between real and manipulated images.

1. Introduction

In this project, we present our approach to detecting deepfake videos/images. Our approach aims to identify and differentiate between real and manipulated images through convolutional neural networks. While our implementation did not surpass the state-of-the-art models in terms of accuracy, it showed interesting results that can be improved with further efforts.

1.1. Table of contents

1.2. Problem statement

The rapid development of image generation and manipulation techniques have brought about significant advancements in the field of computer vision and artificial intelligence. [1] However, it has also led to serious concerns about the credibility of digital content. Deep fake technology, in particular, has the potential to create and spread fake news [4] and misinformation, leading to severe ethical problems or even crimes. Therefore, it has become crucial to develop effective methods for detecting and mitigating the impact of image manipulation.

1.3. Project questions

These are the basic project questions we tend to resolve during our process of this project:

1. What are some current methods for detecting deepfakes?
2. How effective are these methods at detecting different types of deepfakes?
3. Can we come up with our own models that have similar scores to current state-of-art methods?

Besides these questions, here are some interesting side questions related to the topic:

1. What are the ethical implications of deepfake technology, and how can they be addressed?
2. How well are humans able to detect manipulated pictures/videos?

1.4. Significance of the study

Deepfakes is just another challenge to media literacy in the digital age. [1] By developing and promoting effective deepfake detection methods, society could help to promote media literacy and critical thinking, reducing the risk of deception and misinformation. In a world driven by digital technology, this is a crucial and important task to overcome. By developing effective deepfake detection methods, relevant studies can help mitigate such threats and preserve the integrity of digital media. Of course, better deepfake detection could also lead to progress in deepfake itself.

1.5. Overview of the Project

In order to achieve the goal of deepfake detection, we explored various studies/projects related to the topic and concluded our overall process to be:

1. Data collection and preprocessing: Gather the dataset for training and testing our model. Preprocess the data by cropping images from videos in different frames, resizing cropped images, modifying pixel values if necessary .etc.
2. Model selection/training: Choose a suitable pre-trained model as our standard to classify fake and real images. Then, train the model on our preprocessed dataset and fine-tune it to achieve high accuracy. After that, create our own simplified model based on the pre-trained model.
3. Evaluation: Evaluate the performance of our model using metrics like accuracy, precision, recall, and F1-score.
4. Analysis: Analyze the results, provide data overview, and interpretation of results.

2. Proposed Approach

2.1. Dataset and data preprocessing

The dataset we are using is the DeepFake Detection Challenge (DFDC) Dataset, which is the dataset for accompanying DeepFake Detection Challenge (DFDC) from Kaggle competition [5][9]. The DFDC dataset is by far the largest currently and publicly available face swap video dataset, with over 100,000 total clips sourced from 3,426 paid actors, produced with several Deepfake, GAN-based, and non-learned methods.

We download the dataset and use a small portion for training, validation and testing. The preprocessing code loops over each video file in the dataset and extracts frames from the video using OpenCV's VideoCapture() function. The frames are then processed to detect faces using the face_recognition library. [11] Once a face is detected, the code crops the image to include the face with some padding, and saves it to an output directory as a JPEG image. The preprocessing is done and outputs are then fed into the neural network.

2.2. Network Architecture

2.2.1 EfficientNet

For the first network implementation, we choose EfficientNet for baseline, as it's suggested in many discussions for this topic [6]. EfficientNet is a family of convolutional neural network models developed by Google Brain team, designed to achieve state-of-the-art accuracy on image classification tasks with fewer parameters and faster inference times than previous models [7]. The EfficientNet family consists of several different models, labeled B0 to B7, each with a different number of parameters and computational cost. The B0 model is the smallest and lightest and we used the B0 version to achieve a decent accuracy.

We loaded the pretrained model using the efficientnet_pytorch module. The original output feature map has a dimension of 1280, so we replaced the final fully connected layer of the pre-trained model with a new one that has a single output node. This enables the model to perform binary classification by predicting a single output value between 0 and 1. For classification, the

Figure1: Data Preprocessing

output will go through the sigmoid and finally round to 0 or 1. Before training, we froze all the parameters of the pre-trained model to prevent them from being updated during training. This is to prevent overfitting and avoid large computation usage.

2.2.2 Custom CNN

The EfficientNet is very powerful and the result was good. However, the parameter number is too large for us to train. We write and train a tiny convolutional neural network, trying to classify the videos. Our network contains three main parts: Stem, Blocks, and Head.

The Stem part is responsible for processing the input image. It contains a 3*3 convolutional layer with a stride of 2. Then, we add batch normalization and then an activation function. For the activation function, we tried both Rectified Linear Unit (ReLU) and Sigmoid Linear Unit (Swish). For detailed performance differences between these two methods, please check the Conclusion section.

The Blocks part is meant to manipulate the MBConv blocks in EfficientNet-B0. Each block contains a 1*1 convolution layer, batch normalization, ReLu/Swish activation, and finally another 1*1 convolution layer. These blocks will gradually increase the complexity of our network.

The final section is the Head part. The key part of this section is the adaptive average pooling layer at the end. This pooling layer reduces the dimensions of the features ad the pass is through a fully connected layer to help us get the prediction.

Figure 2 shows the architectural schematics of our model. For more details, please see the Figure below.

2.2.3 InceptionResNet

For the third approach, we try to use the InceptionResNet-V2 from PyTorch Image Models Module, which is another pre-trained model fine-tuned for classification. [11] One advantage of it is that the number of classes could be set in the model loading function, so the best parameter could be loaded. Like EfficientNet, we replaced the last fully connected layer to get a classification. The hyperparameters are similar to the EfficientNet. It achieved a similar performance.

2.3. Training Process

To train our deepfake detection model, we use the sample training data provided by Kaggle.[8] This is a small portion of the DFDC dataset that contains 400 different videos, with 77 videos with the REAL tag and others with the FAKE tag. For the validation dataset, we use the first section, or section 0 of the Kaggle DFDC dataset as our testing set. This portion of the dataset contains 1334 videos with 86 REAL videos and the rest of 1248 FAKE videos. Though this seems like a very imbalanced dataset, we still decided to use it. Further detail on how we use this dataset could be found in section 2.4.

For training, we use a learning rate of 0.001 and a learning rate decay factor of 0.1 for 20 epochs. The optimizer is Adam, with the loss function being BCEWithLogitsLoss, which is a combination of the Binary Cross Entropy Loss (BCELoss) and the Sigmoid activation function, which is commonly used in deep learning for binary classification problems.

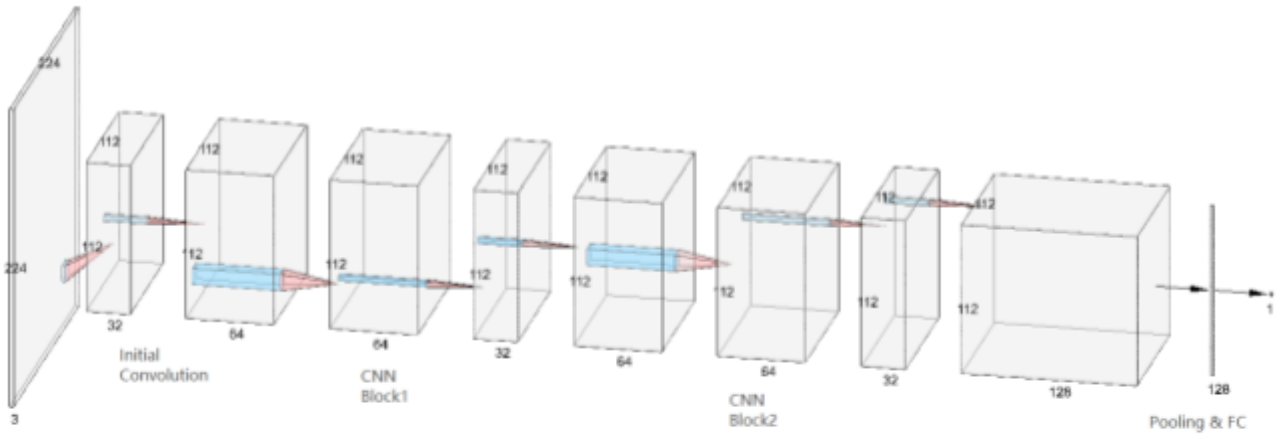

Figure 2: Custom Network

2.4. Evaluation Metrics

Because of equipment limitations and time constraints, we decided to use only the first portion of DFDC dataset provided on Kaggle [9] as our testing set. This portion of the data contains a total of 1334 videos, with 86 of them as REAL data and the rest of them as FAKE data. Though it seems like an extremely imbalanced dataset, in many real-world scenarios, imbalanced datasets are common, where one class has significantly fewer examples than the others. But to have a more general view of the performance, we have tried to augment data specifically on REAL data to make our model more robust. In our final test set, the ratio of REAL to FAKE is about 1:3.

In order to evaluate the performance of our proposed deepfake detection approach, we used the following evaluation metrics: Accuracy, Precision, Recall, and F-1 score.

Accuracy is defined as the proportion of correctly classified samples over the total number of samples. Or as the formula below:

$$Overall\ Accuracy = \frac{Num\ True\ Positive + Num\ True\ Negative}{Total\ Samples}$$

Precision measures the proportion of true positives (the proportion of correctly classified deepfakes over the total number of samples classified as deepfakes) over the total number of samples classified as deepfakes. At the same time, recall measures the proportion of true positives over the total number of actual deepfakes (the proportion of correctly classified deepfakes over the total number of actual deepfakes). Finally, we have the F-1 score, the harmonic mean of precision and recall. This score is used for us to evaluate the overall performance of the model. The F-1 score takes into account both the false positives and false negatives, making it a more balanced measure of a model's accuracy than precision or recall alone.

Of course, we gave up common metrics like runtime and memory usage. But since efficiency was not our main goal, it's reasonable for us to omit these metrics. For a detailed conclusion on the gathered data, please check the next section.

3. Results

3.1. Performance comparison

Table 1 refers to the metric scores gathered from our tests. As we predicted, the two pre-trained methods have higher performance on accuracy. The precision scores of all models are similar, and huge differences exist among their recall rates.

Model	Precision	Recall	F1-Score	Train Accuracy	Test Accuracy
EfficientNet	0.8544	0.9859	0.9165	0.8679	0.8466
Custom CNN (ReLU)	0.8599	0.8382	0.8489	0.8178	0.7469
Custom CNN (Swish)	0.8541	0.7077	0.7741	0.8136	0.6495
Inception ResNet	0.8663	0.9104	0.9178	0.8661	0.8482

Table 1: Metric Scores

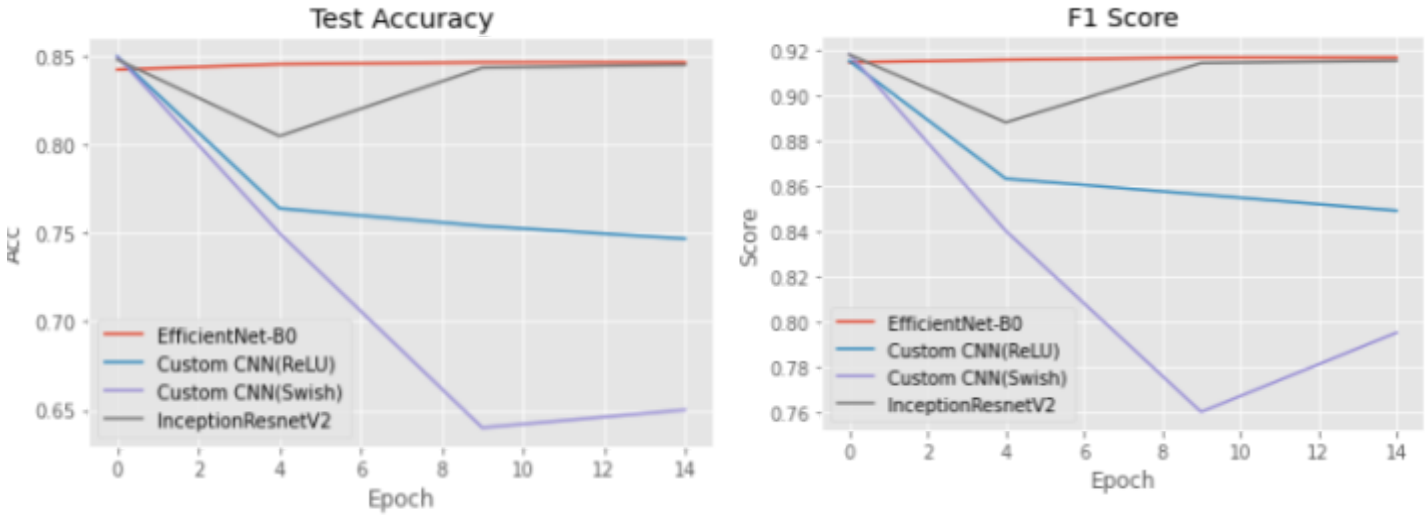

Figure 3: Accuracy and F-1 Score of Different Models

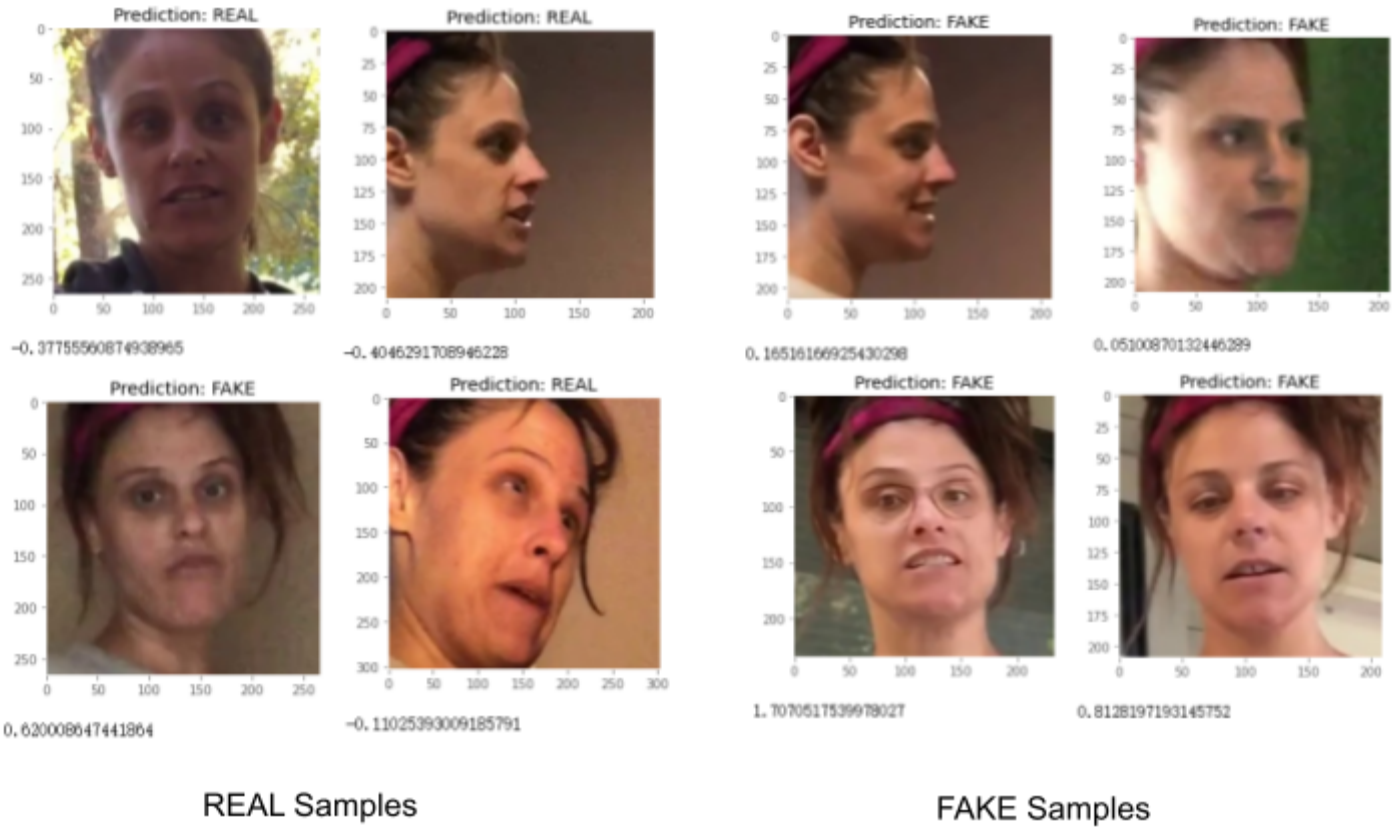

Figure 4: Predictions for Some Samples

3.2. Analysis of the results

Inception ResNet seems to be the best-performing model among the four. As our goal is to find out deepfakes, it's important for our model to have a higher precision, and Inception ResNet performs the best. Figure 4 shows some predictions from Inception ResNet for given labels.

EfficientNet also achieved a high accuracy, but it has an abnormally high recall, which shows that it may be biased towards predicting positive labels.

For our self-defined nets, using ReLU as activation function increased the accuracy, which is surprising as sources show Swish tends to work better than ReLU on deeper models across a number of challenging data sets [9]. The low recall rates show that they are less subject to bias or overfitting, and they even have higher precision than EfficientNet.

3.3. Interesting Findings

During our implementation of our own network, we observed a decreasing performance with increasing epochs. Upon further analysis, we discovered that the main reason for this was a decreasing recall value. At the same time, the Precision value increases slightly as the recall goes down. We believe that this issue may have been due to overfitting. But, this might also be caused by our imbalanced data leaning towards more FAKE attributes during the training and testing process. Since we can see our model increasing its true-positive rate within its selected classifications, it's not 100% safe to claim that overfitting is the reason behind it. Still, we gained valuable experience in implementing and training our own network, which will inform our future work in the field of deepfake detection. If time allows, we would love to explore the overall false-positive rate and other metrics to check the reason behind this.

4. Discussion and Conclusions

4.1. Summary of the project

In this project, we aimed to implement deepfake detection. Although our model did not perform as well as we had hoped, achieving an F-1 Score around 0.85 for the selected data (and we expect it to be lower for a larger dataset), we gained valuable insights into the challenges and limitations of deepfake detection. Specifically, we

found that imbalanced data and limited computational resources were significant obstacles to achieving better performance. Despite these limitations, we believe that our project provides us with a useful exercise for future research on this topic, and we learned a great deal about the intricacies of deepfake detection that will inform our future work.

4.2. Contributions of the work

For this project, works are divided into the following:

Yifan: Background Research, Data Selection, CustomCNN Testing, Report Drafting, Report Refinement

Mingzhi: Background Research, Data Selection, Data-Preprocess, Pre-Trained Network Testing, Report Refinement

Works and data are shared using Github and GoogleDrive [10].

4.3. Discussion of limitations and future directions

Despite our efforts, there were a few limitations to our project that impacted our results, and here's how we think we could do better if there's a chance. First, as mentioned in the summary section, we recognize the imbalanced data and the size of the data we used for training. This data testing may have affected the accuracy of our model. In future work, we plan to use more diverse and representative data to address this issue. Additionally, we found that our feature extraction process could be improved by utilizing the features of the face_recognition library and gathering more useful data during the data-preprocessing stage. [12] Thirdly, we plan to explore the use of pre-trained networks in future work, including unfreezing these networks to potentially improve performance. Finally, in response to our interesting finding in section 3.3, we plan to address the issue in future work by using techniques such as regularization to prevent overfitting and improve our model's recall value. Overall, we believe that addressing these limitations will be crucial for advancing the field of deepfake detection and improving the accuracy of our models.

References

[1] M. Westerlund, "The emergence of Deepfake Technology: A Review," *Technology Innovation Management Review*, vol. 9, no. 11, pp. 39–52, 2019.

[2] V. Karasavva and A. Noorbhai, "The real threat of Deepfake pornography: A review of Canadian policy," *Cyberpsychology, Behavior, and Social Networking*, vol. 24, no. 3, pp. 203–209, 2021.

[3] BA. Borji, "Generated Faces in the Wild: Quantitative Comparison of Stable Diffusion, Midjourney and DALL-E 2," in *ArXiv, 2022*.

[4] B. Han, X. Han, H. Zhang, J. Li, and X. Cao, "Fighting fake news: Two stream network for Deepfake detection via learnable SRM," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 3, pp. 320–331, 2021.

[5] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, C. Ferrer, 2020. "The DeepFake Detection Challenge (DFDC) Dataset". arxiv.org/abs/2006.07397

[6] D. A. Coccomini, N. Messina, C. Gennaro, and F. Falchi, "Combining EfficientNet and vision transformers for video deepfake detection," *Image Analysis and Processing – ICIAP 2022*, pp. 219–229, 2022.

[7] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in *ArXiv*, 2019. arxiv.org/abs/1905.11946.

[8] "Deepfake Detection Challenge," *Kaggle*. [Online]. Available: https://www.kaggle.com/competitions/deepfake-detection-challenge/data.

[9] R. Nerd, "Swish activation function by Google," Available: https://medium.com/@neuralnets/swish-activation-function-by-google-53e1ea86f820

[10] Y. Jiang. https://github.com/o0BB0o/cv_project_deepfake

[11] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. Alemi, "Inception-V4, inception-resnet and the impact of residual connections on learning," https://arxiv.org/abs/1602.07261.

[12] "Face-recognition," *PyPI*. [Online]. Available: https://pypi.org/project/face-recognition/.