Deepfake Detection

Angel Chu
Electrical and Computer Engineering
yuchu@andrew.cmu.edu

Zihong Shi
Electrical and Computer Engineering
zihongs@andrew.cmu.edu

Yanqiao Wang
Electrical and Computer Engineering
yanqiaow@andrew.cmu.edu

1 Introduction

With the rapid progress in synthetic image generation and manipulation, digital content that appears in videos and photos can be used to defame persons, maliciously as a source of misinformation and persuasion [1]. Deepfakes are the result of digital manipulation to forge logistic but fake images or videos. Fake images and videos can be generated using a variety of methods, such as variational autoencoders (VAEs) or generative adversarial networks (GANs), which are becoming more accessible and accurate [2].

Over the years, people have been working on coming up with different types of solutions to detect whether a video has been modified. Irreversible operations leave a special footprint that can be exposed to detect specific edits. However, forensic footprints are often very subtle and difficult to detect. This is a case of video being over-compared or down-sampled. Therefore, modern facial manipulation techniques are very difficult to detect.

Given the negative effects of Deepfake technology and the difficulties to detect fake videos, our team aims to implement a method to classify Deepfakes and identify manipulated media within the subset of the Kaggle Deepfake Detection Challenge Data.

In this project, we present a method to classify Deepfakes, existing video replaced with someone else's likeness, and identify manipulated media based on depthwise separable convolutions in neural computer vision architectures with residual connections.

2 Related Work

Before discussing related work on deepfake detection and classification model, the face manipulation methods need to be addressed first.

Generative adversarial networks (GANs) are a form of deep neural network that is capable to learn from a set of training data sets and create a sample of data with the same features and characteristics [3]. It can apply face aging, generate new viewpoints, or alter face attributes like skin color [4]. CycleGAN is also a deepfake technique which is an unsupervised method that can perform the image-to-image conversion without using paired examples[3].

Traditionally, convolutional neural networks have been used to perform video deepfake detection. However, the majority of previous research uses the same type of dataset to train and test their models. The research from FaceForensics++ leveraged the recent advances in deep learning to learn extremely powerful image features with CNNs and tackle the detection problem by training a neural network in a supervised fashion [4].

Abdulqader M. Almars also proposed a model that first employs CNN for frame features extraction, followed by Long Short-Term Memory (LSTM) layers to analyze a temporal sequence for face manipulation between frames and a softmax function to classify the video as real or fake [4].

In addition to the traditional deepfake detection model, hybrid approaches are also introduced by researchers. A group of researchers has proposed a model combining EfficientNet and Vision Transformers for video deepfake detection and achieved an AUC of 0.951 for their best model [2].

A 3D-attentional inception network model has also been proposed which leverages spatial information from separate frames and values inter-frame temporal information. The inception architecture of convolutional neural networks first demonstrates the advantages of factoring convolutions into multiple branches operating successively on channels and then on space. The proposed model incorporates both spatial and temporal information with the 3D kernels and applies channel and spatial-temporal attention modules to improve the detection ability [5].

3 Methods

The suggested approach is divided into three major phases: dataset pre-processing, data augmentation, and feature extraction.

The dataset was originally picked from Kaggle Deepfake Detection Challenge [6]. Since the full training set Kaggle provided is over 470 GB [6], the pre-processing and training process can be extremely long. Therefore, regarding the limited hardware, a subset of the full training dataset is used. The dataset for the final training is about 10 GB in size and contains 400 videos where each is 10-second long. Each training video happens in a real background with at least one person talking or making facial expressions in it. This project focuses on whether the faces are manipulated, so other factors such as the background and voices will not be included when classifying the videos.

Pre-processing is a procedure of internal face tracking, alignment, and extraction from each video clip. The extracted aligned faces will then be saved as images for future use. Face detection is performed with the retinaface-resnet50-pytorch model which is a PyTorch implementation of medium size RetinaFace model with ResNet50 backbone for Face Localization [7]. After performing the face detection, it will return facial area coordinates and landmarks, such as eyes and mouth, which can apply to the following alignment of the detected face. All detected faces were cropped to 320x320 pixels, and 15 frames were extracted from per 10-second video. The pre-processing continues recursively in the input directory on each mp4 file.

Before the training, there is the step of data augmentation. It contains multiple transforms. After the application of augmentation, the sepia filter is applied to the input RGB images. Lastly, normalization is applied with the formula $\text{img} = \frac{\text{img - mean * max\_pixel\_value}}{\text{std * max\_pixel\_value}}$. All of these are applied within the fast image augmentation library `Albumentations`. The detailed pre-processing procedure is given in Table 1.

We propose a convolutional neural network architecture based on a stack of depthwise separable convolution layers. It is based on the original paper: "Rethinking the Inception Architecture for Computer Vision" by Szegedy, et. al [8]. Figure 1 introduces the conceptual model workflow of depthwise separable convolution that first performs pointwise convolution, a 1x1 convolution, to map cross-channel correlations. Then the result separately maps the spatial correlations of every output channel, which indicates a depthwise convolution. The outputs are concatenated and sent to the next inception module. Compared with conventional convolution, the depthwise separable convolution has fewer number of connections and a lighter model because it does not need to perform convolution across all channels, which makes the model much more efficient in terms of computation time.

The entire model architecture is shown in Figure 2. It contains 14 blocks that are constructed from a total of 36 convolutional layers forming the repeated feature extraction. As indicated in each block, all convolution and separable convolution layers are followed by batch normalization [9], which is to normalize input features on models leading to a substantial reduction in convergence time. The depth multiplier of separable convolution layers is set to 1 at default. By changing depth multiplier, we can change the number of output channels in the depthwise convolution. Here, it is unchanged to be 1. Rectified linear unit is also used after, in order to achieve a non-linear transformation of the data so that the transformed data will be close to linear for regression and linearly separable for classification. Except for the first and last blocks, all other blocks have linear residual connections around them

Table 1: Pre-processing Procedures

Order	Name	Function	Probability of applying the transform (p)
1	ChannelShuffle	Randomly rearrange channels of the input RGB image	0.1
2	GaussNoise	Apply gaussian noise to the input image	0.1
3	GaussianBlur	Blur the input image using a Gaussian filter with a random kernel size	0.1
4	HueSaturationValue	Randomly change hue, saturation and value of the input image	0.1
5	RandomBrightness	Randomly change brightness of the input image	0.8
6	RandomBrightnessContrast	Randomly change brightness and contrast of the input image	0.2
7	ToSepia	Applies sepia filter to the input RGB image	0.1
8	Resize	Resize the input to the given height and width (320x320)	—
5	Normalize	Apply normalization	—

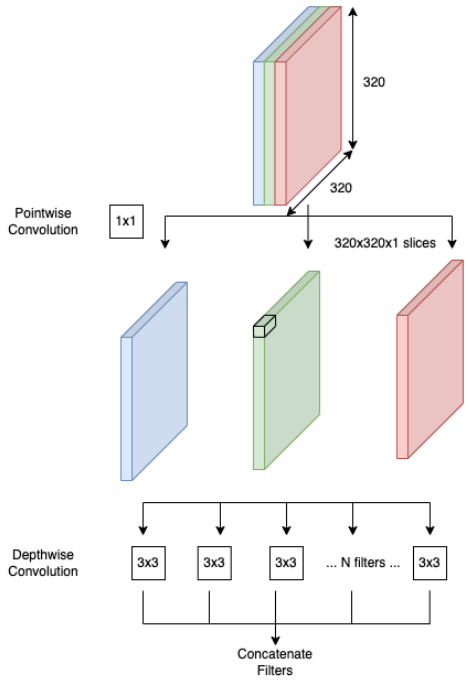

Figure 1: Modified depthwise separable convolution

to make the whole architecture easy to define and modify. The pooling operation at the end of each block is implemented to decrease the grid size of the feature maps. Since the problem is essentially image classification, the convolutional base is followed by a logistic regression layer.

After finishing training for the training dataset, validation is performed. The model generates an average prediction value for each test video. The table with different thresholds is given in Table 4, indicating the performance of the model with the different threshold values. Comparing the accuracy

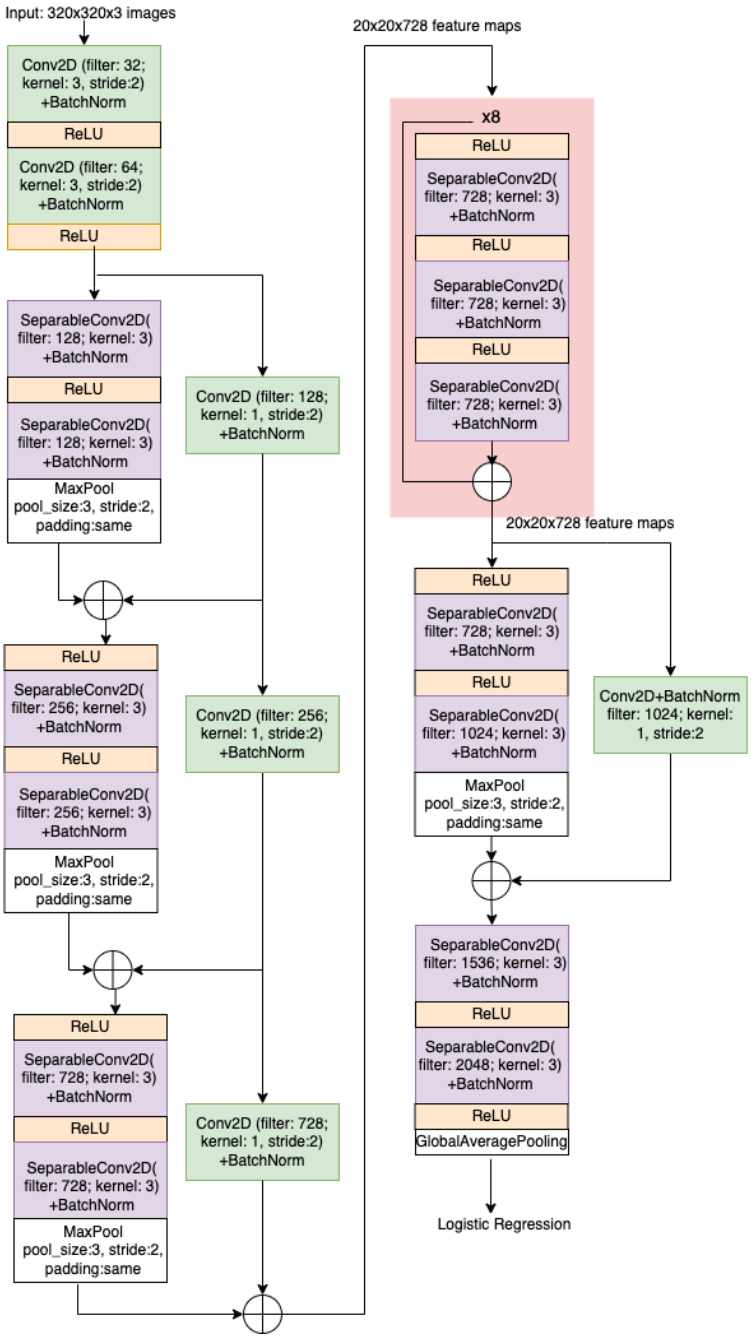

Figure 2: Architecture Diagram

with different thresholds, the final threshold is set to be 0.9 for the highest accuracy. If the predicted value is greater than 0.9, the video is judged to be fake, and if it is less than 0.9, the video is judged to be real.

4 Experiments

4.1 Datasets

We use a part of the Deepfake Detection Challeng(DFDC) dataset built by Facebook as the dataset for this experiment. As mentioned in the previous chapter, we use the dataset containing 400 training videos, capture 15 frames from each video, and performe face classification on each extracted frame using the RetinaFace network and cropped the images to 320x320 size. After extracting the frames from the video, we get a total of 6000 images of human faces. 80 percent of the dataset is divided into the training set and 20 percent into the verification set. We extracte part of our processing results which are shown in Figure 3-4. We use another 400 videos as test set. Of the 400 test videos, 200 are real videos and 200 are fake videos.

Figure 3: face image after cropping

Figure 4: images extracted from a video

4.2 Training of Neural Network

The training process of the neural network is completed using the above 6000 pictures. Data loading is completed by calling the Dataloader function in the pytorch library, batch size is set to 16 and shuffle is set to True to ensure that the training data of each iteration is randomly selected images. We invoke 1 child process to import data at the same time.

The epoch of the model training is set to 10. The Loss function uses the Cross Entropy Loss function, which is often used in classification training. The cross entropy loss function combines the LogSoftmax function and the NLLLoss function to reflect the gap between the output value and the expected label value during the training of the neural network, and updates the parameters of the neural network through the back propagation.

The Stochastic gradient descent algorithm is used in the gradient descent process, and the learning rate is set to 0.01. In order to prevent overfitting caused by excessive free parameters of the model, the weight decay parameter was used and set to 0.0001. The momentum parameter is set to 0.9 to prevent the model from falling into the local optimal solution and enable the model to quickly find the parameter when the objective function reaches the minimum value. The amount of gradient descent in gradient descent algorithm is

$$v = -\frac{\nabla f}{\nabla x} \times lr + v \times momentum \tag{1}$$

the momentum can accelerate the gradient descent and accelerate the speed of finding the optimal solution.

In addition, the learning rate lr adopts the method of dynamic adjustment, and the initial learning rate is 0.01. We use the StepLr function to adjust the learning rate according to the training times as the

following formula

$$new\_lr = initial\_lr \times \gamma \tag{2}$$

The γ is set to 0.2 and step size is set to 2, that is, the learning rate is updated once without two training sessions.

4.3 Model Evaluation

The neural network model evaluates 400 videos in turn, and each video gets a predicted value. In our experiment, two evaluation methods are approached. For each video, we take 30 frames of it, and we put a predicted value on each frame.

The first method averages the predicted values of the 30 frames extracted from each video, and that average value is the predicted value of the video.

The second approach aims to take into account every frame. It looks for the top three largest predicted values and determine if all three values are greater than the threshold. The video is determined as "fake" if they are greater than the threshold, otherwise, we calculate the mean value of 30 frames. The second method takes the several maximum values of all frames to avoid neglecting the high-value outliers.

4.4 Results

In this experiment, the label of the fake video in the training set is 1, and the label of the real video is 0. The classification of the predicted video is a binary classification problem – judging whether the video is true or false. Predicting video as 1(fake video) is Positive, predicting video as 0(True video) is Negative, the correct prediction is True, and the wrong prediction is False.

4.4.1 Detect Accuracy

The correct prediction includes: the false video is judged as "False" and the real video is judged as "True". Therefore, according to the definition of confusion matrix, the accuracy "Acc" can be calculated as follows:

$$Acc = \frac{TP + TN}{TP + FP + FN + TN} \tag{3}$$

In this experiment, prediction value $P_{video} \in (0, 1)$, we artificially delimited a classification threshold t, that is, if the predicted value $P_{video} > t$, the video judged by the model is considered to be false; if the predicted value $P_{video} < t$, the video is considered to be true.

Since we find that the prediction values of fake videos are close to 1 and the values of real videos are close to 0.8, so we decide to set the threshold to 0.9 first. When we set the threshold to 0.9, the test results are shown in the figure below:

Table 2: Comparison of detection results between two methods

method	test result				
	Accuracy	TP	TN	FN	FP
method 1	51.75%	137	70	63	130
method 2	52.50%	149	61	51	139

We can see that the second method achieves a slightly higher detection accuracy. Taking the mean value of videos could disregard the impact of one or two frames and may misclassify the fake videos as real videos. The second approach, however, could increase the True Positive (TP) rate because we take into account the maximum prediction values of the 30 frames in a video. And we can see that the second method detected 12 more fake videos than the first method, though 9 more real videos are incorrectly classified as fake videos. The wrong classification of one or two frames of the video determines the wrong classification of the whole video. The predicted result is that the second approach could better classify the fake and real videos since it can keep track of every frame to avoid

some fake frames being ignored because of averaging.

4.4.2 Logloss Value

A Logloss value is calculated as follows:

$$Logloss = -\frac{1}{n}\sum_{i=1}^{n}[y_i \ln \hat{y}_i + (1 - y_i)\ln(1 - \hat{y}_i)] \tag{4}$$

Where, $\hat{y}_i$ is the predicted value of the video, which can also be regarded as the confidence level for determining that the video is fake. The closer the value of $\hat{y}_i$ is to 1, the higher the confidence level for determining that the video is fake, and y_i is the label value of the video. The Logloss value can see the difference between the overall predicted value of the video and the label value. The smaller Logloss value means a more accurate prediction.

Table 3: Comparison of Logloss values between two models

methods	results	
	accuracy	Logloss
method1	51.75%	2.106
method2	52.50%	2.053

We can see that method 2 has a smaller Logloss value than method 1.

4.4.3 ROC Curve

Roc curves and AUC values of the two methods of average value and maximum value are shown in Figure 5.

By taking the average of the selected frames, there is a prediction value for each video. Figure 5 is the Receiver Operating Characteristic (ROC) curve of all the videos, generated based on 400 test videos. More than half of the videos could be classified correctly. The AUC value for the model is given in the right bottom corner of Figure 5.

The second approach of taking maximum values into account behaves a little better than the method of average values. Method 2 has a 0.586 AUC value while method 1 has a 0.584 AUC value. A false positive value is not as high as a false negative value, meaning that the low accuracy value is because of the misclassified real videos.

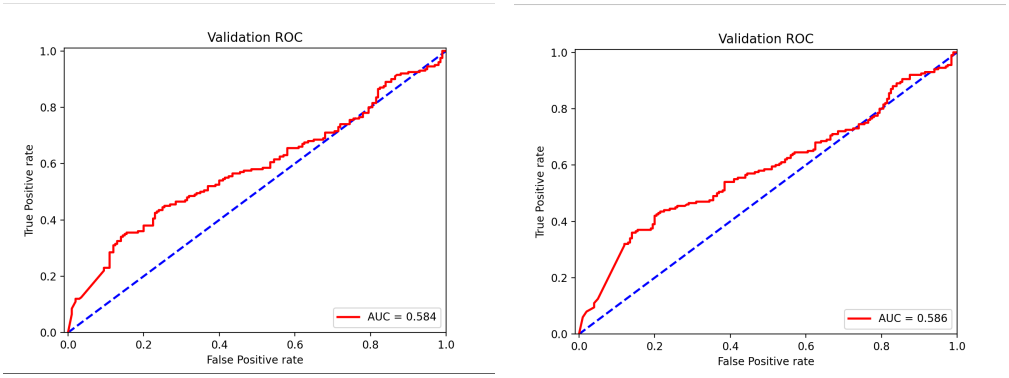

Figure 5: ROC curve with AUC value. (a) method 1; (b) method 2

4.4.4 Different Thresholds

We also compared the detection accuracy under different threshold Settings, and the accuracy results are shown in Table 4.

Table 4: Comparison of accuracy between different thresholds

threshold	results		
	accuracy	TP	TN
0.85	50.00%	152	48
0.88	51.50%	150	56
0.90	52.50%	149	61
0.92	53.50%	131	83
0.95	55.75%	114	109

As can be seen from Table 4, compared with the initial threshold of 0.9, an appropriate increase in the threshold can obtain a higher accuracy rate, but the probability of detected fake videos will also decrease a lot. The reason for the higher accuracy rate is that more real videos are correctly classified.

This shows that the classification effect of our model on the real video needs to be improved. But 0.9 is still an appropriate threshold given that we want to detect as many fake videos as possible.

5 Discussion and Conclusion

By validating our training results with randomly selected sample faces tested with human eyes, the result is satisfying that upon data that even human evaluators have a hard time distinguishing fake videos from real ones, our model performs better than the random chance line as shown in Figure 5.

With the efficient use of model parameters, our project can help classify and prevent manipulated media. With linear stack-of-layer architecture, it is easy to modify and further improved. Thus, it is useful for further research.

Further work could exploit various numbers of independent channel space segments used for performing spatial convolutions or ensembling with other architecture to seek for higher accuracy rate. Plus, the current dataset we are using is about 10 GB, if provided with Graphics Processing Units or other hardware platforms, future work could be done with expanding the scale of the dataset or frames as time permits, which are critical to a detector's performance. Then Fake/Real distinguishing may result in better precision.

References

[1] Advantages and Disadvantages of Deepfake Technology | by Mehmet Emin Masca | Geek Culture | Medium." https://medium.com/geekculture/advantages-and-disadvantages-of-deepfake-technology-ccfa7c12b1ae [accessed Dec. 17, 2022].

[2] Davide Coccomini, Nicola Messina, Claudio Gennaro, and Fabrizio Falchi, "Combining EfficientNet and Vision Transformers for Video Deepfake Detection," Jan, 2022. https://arxiv.org/pdf/2107.02612v2.pdf [accessed Dec. 17, 2022]

[3] Almars, A. (2021) Deepfakes Detection Techniques Using Deep Learning: A Survey. Journal of Computer and Communications, 9, 20-35. doi: 10.4236/jcc.2021.95003. [accessed: Dec. 17 2022].

[4] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "FaceForensics++: Learning to Detect Manipulated Facial Images". [accessed: Dec. 17 2022].

[5] Changlei Lu, Bin Liu, Wenbo Zhou, Qi Chu, Nenghai Yu, "Deepfake video detection using 3D-attentional inception Convolutional Neural Network." https://ieeexplore.ieee.org/document/9506381. [accessed Dec. 17, 2022].

[6] "Deepfake Detection Challenge," Kaggle. [Online]. Available: https://www.kaggle.com/c/deepfake-detection-challenge. [accessed: Dec. 17 2022].

[7] "Use Case and High-Level Description. " https://docs.openvino.ai/latest/omz_models_model_retinaface_resnet50_pytorch.html. [accessed: Dec. 17 2022].

[8] Szegedy, et. al, "Rethinking the Inception Architecture for Computer Vision." https://arxiv.org/abs/1512.00567. [accessed: Dec. 17 2022].

[9] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In Proceedings of The 32nd International Conference on Machine Learning, pages 448–456, 2015. [accessed: Dec. 17 2022].