

Deepfake Video Classifiers – Capture Images and Texts from Deepfake & Real YouTube Videos as Features

Yunting Chiu

M.S. in Data Science, Department of Mathematics & Statistics at American University

INTRODUCTION

Motivation

Deepfake Videos may fool people nowadays and they are causing trouble.

Contributions

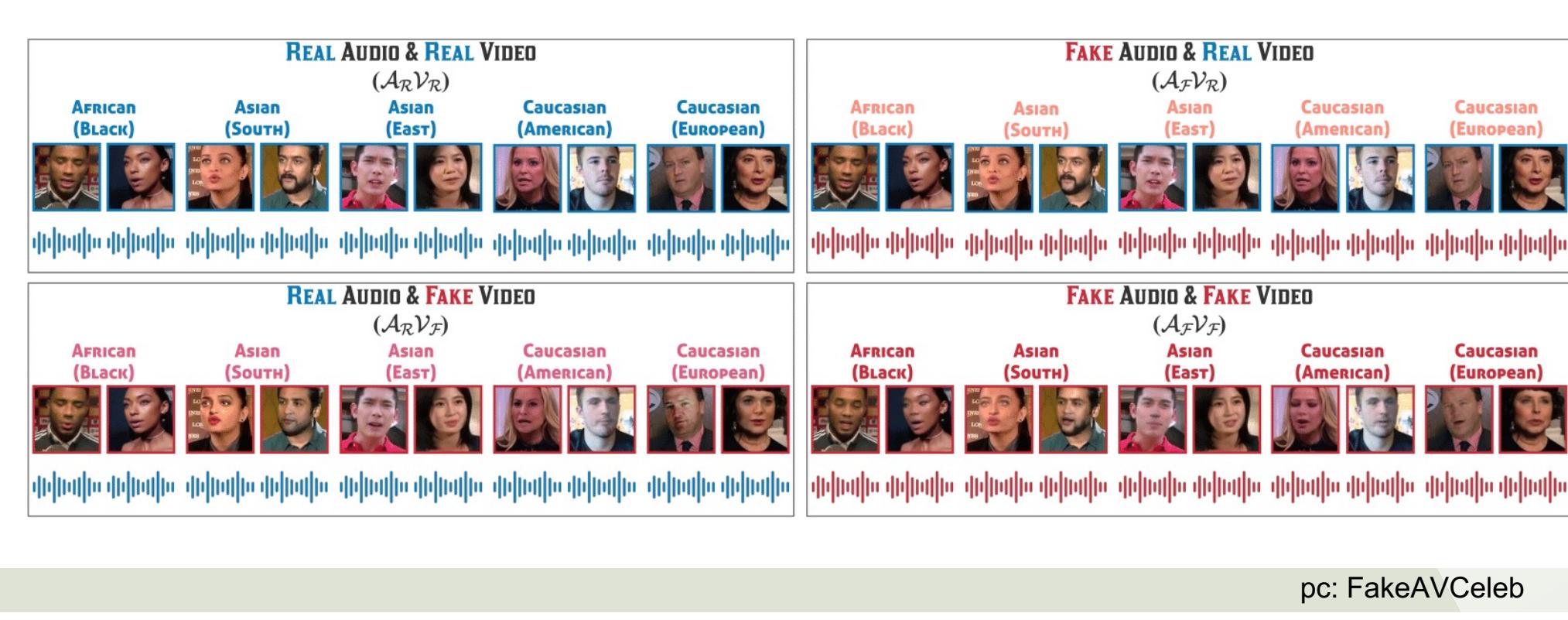
- I successfully extract two important video features: images and texts.
Then, I concatenate them as a data frame for model training.
- I construct two different classification models to classify Deepfake videos with 97 % and 88 % accuracy scores.

Keywords

Computer Vision, Natural Language Processing, Machine Learning, Neural Network

FakeAVCeleb DATASET

The authors propose a novel Audio-Video Deepfake dataset (FakeAVCeleb [1]) that contains not only deepfake videos but also respective synthesized lip-synced fake audios. FakeAVCeleb is generated using recent most popular deepfake generation methods. To generate a more realistic dataset, the authors selected real YouTube videos of celebrities having four racial backgrounds (Caucasian, Black, East Asian, and South Asian) to counter the racial bias issue.



I choose RTVC as the fake subset in terms of real-video and fake-audio

I choose FSGAN as the fake subset in terms of fake-video and real-audio

I choose Faceswap as the fake subset in terms of fake-video and fake-audio

I choose VoxCeleb2 as the real subset in terms of real-video and

DATA PREPROCESSING

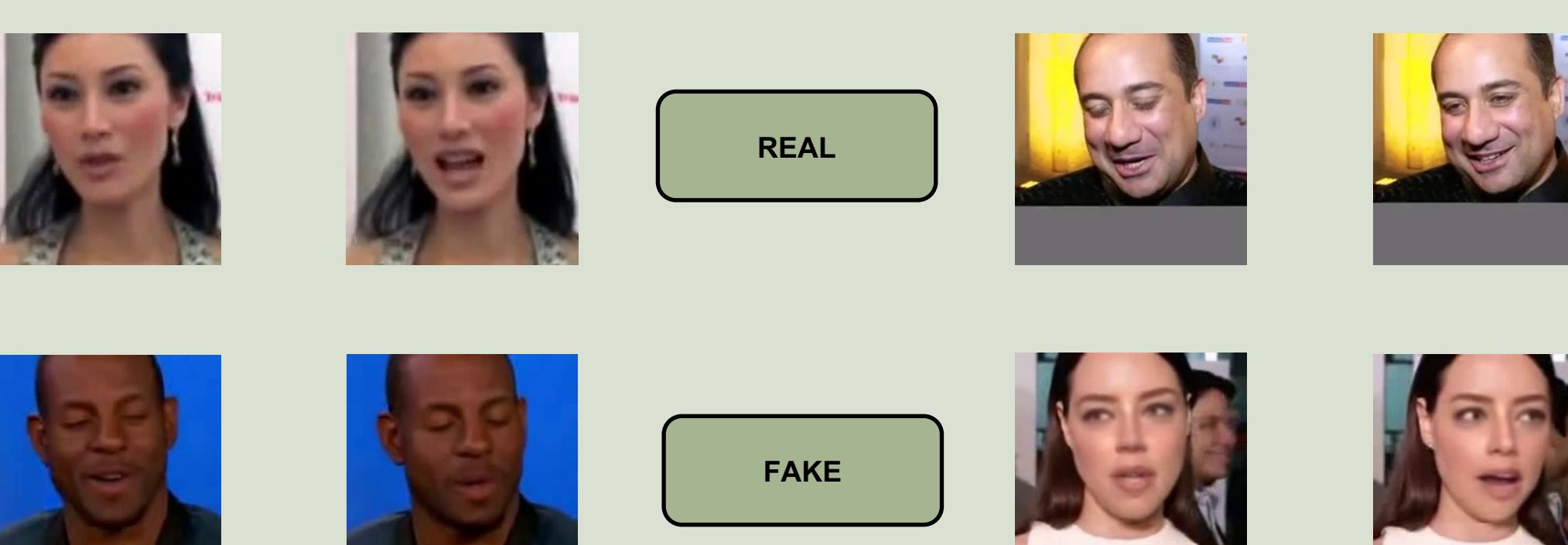
1. Capture Images from Videos

Fake Images :

- RTVC: 360 videos x 4 frames = 1440 frames
- FSGAN: 1884 videos x 2 frames = 3768 frames
- Faceswap: 2256 videos x 2 frames = 4512 frames

Real Images:

- VoxCeleb2 : 390 videos x 25 frames = 9750 frames



2. Capture Audios from Videos and then Transfer Them to Texts

Leverage the *SpeechRecognition* package to convert audios to texts.

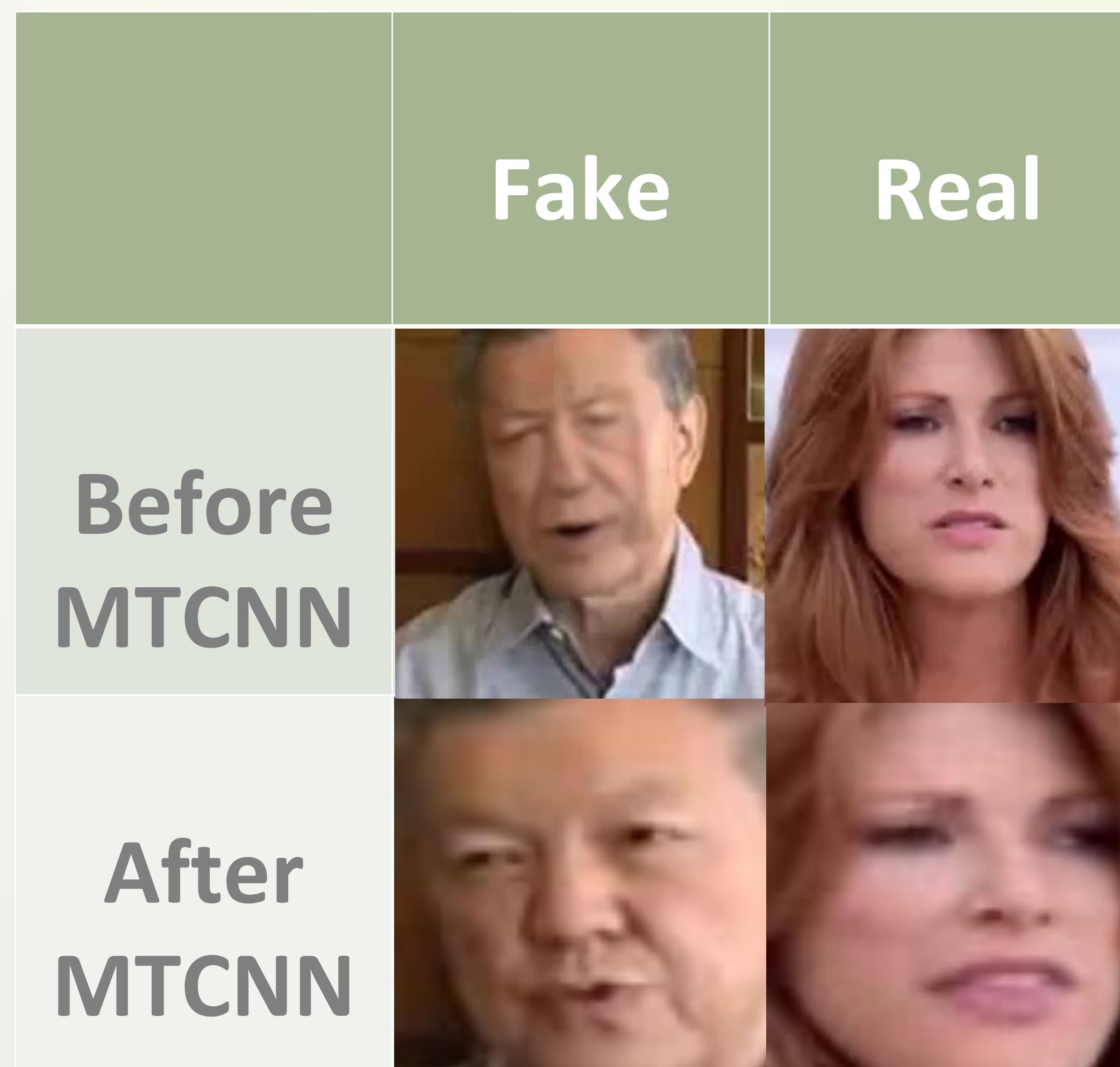
Fake Texts: "it's just be widespread casted like 12 cast members", "attraction carries not that she's being that and then that's address and everything"

Real Texts: "beautiful I love the weather here I love like the warm climate

I've loved every everywhere we go out to eat and", "I said that you know God so had this plan where he thought I want"

3. Extract Facial Features Using a Pre-Trained MTCNN Model

MTCNN (Multi-Task Cascaded Convolutional Neural Networks) is a type of neural network that recognizes faces and facial landmarks in images. It was published by Zhang et al. in 2016. MTCNN is a good face detector that provides exactly pixel positions of precise nose, mouth, left eye, right eye, and face boundary. Thanks for the prior research, I can now simply install a 'MTCNN' package to capture the facial features.



My task for this session is to save each image as a one-dimensional array. The images are three-dimensional in nature (RGB channels). I need to change the dimensions from 64 x 64 x 3 to 1 x 12288. I flatten each pixel as 1D array to each observation. Also, all data points have normalized from 0-255 to 0-1.

3. Vectorize Texts Using a Pre-Trained word2vec Model

After tokenizing, converting to lowercase, and removing stop words and numbers, the texts now are cleaned. I keep punctuation and emoticon symbols since they might be necessary for the vectorization procedure that follows. A popular pre-trained option is the Google News dataset model, containing 300-dimensional embeddings for 3 millions words and phrases. For constructing vector representations of words, this tool provides an efficient implementation of the continuous bag-of-words and skip-gram designs. A single video of texts is now a 1 x 300 array after vectorization.

4. Concatenate Image Feature and Text Features

Finally, I column-wise concatenate image and text features and save them as a new data frame from which we will begin building the models.

EXPLORATORY DATA ANALYSIS

X Features

Data dimension: 19470 rows x 12288 columns

- 9720 observations from the fake set, 9750 observations from the real set
- 12288 features from the images, 300 features from the texts

Y Labels

Label 0 as fake videos, label 1 as real videos. Each observation is labeled, so this is supervised learning.

MACHINE LEARNING MODEL

Random Forest

- Train-test split: 80 % training, 20 % testing
- Cross validation: (1) K-fold cross-validation, k = 5
(2) Utilize bootstrap samples to build trees

Why Random Forest

- It can handle large features efficiently
- It can reduce computation time

Hyperparameters Tuning the Random Forest Model

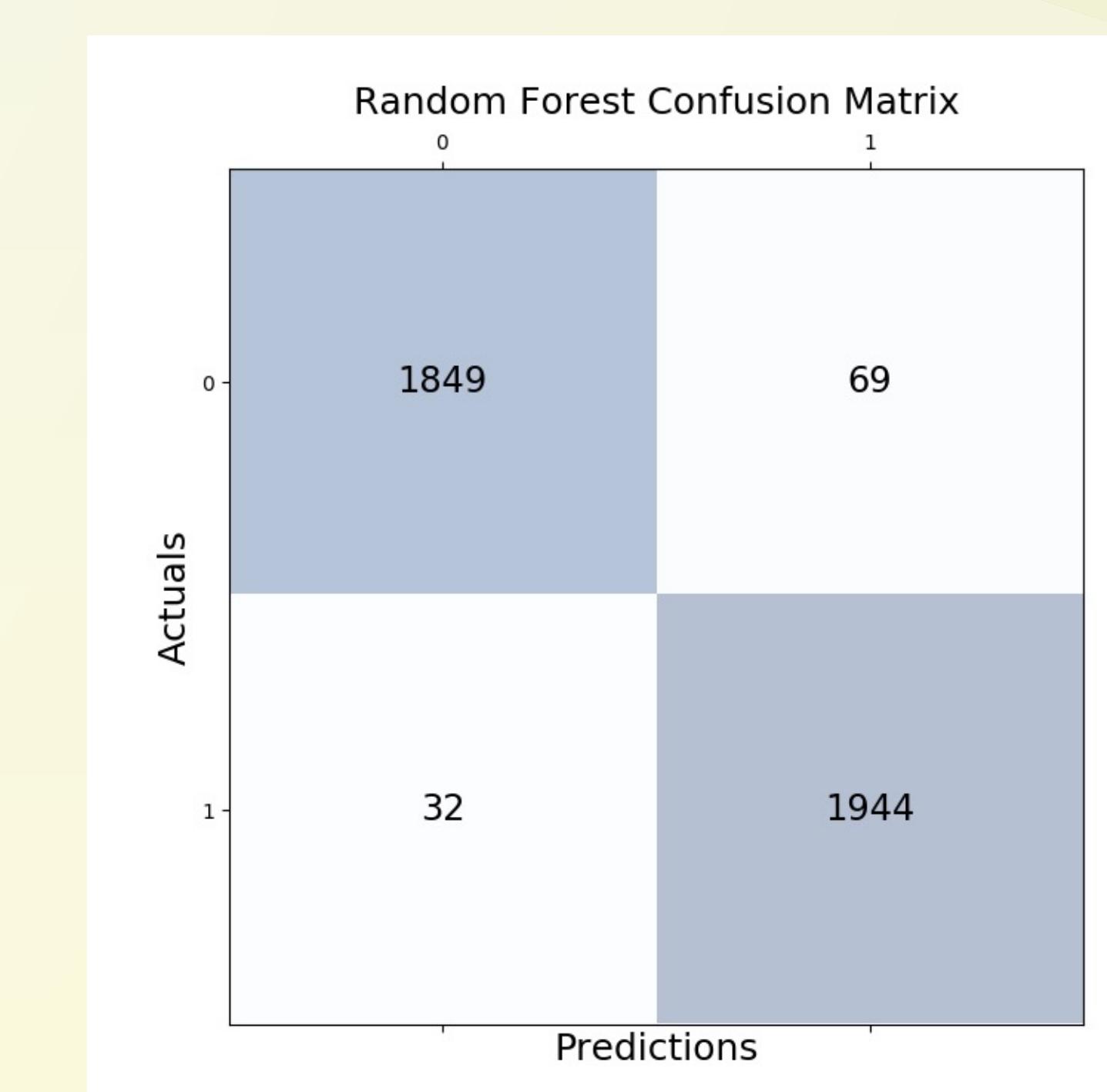
I leverage grid search in training set, the selected parameters are:

- grid['n_estimators'] = [100, 500]
- grid['max_depth'] = [5, 10, 50, 100]

Random Forest Model Results

Testing Accuracy: 97.41%

Random Forest Model Confusion Matrix



Random Forest Model Classification Report

	precision	recall	F1-score	support
fake	0.98	0.96	0.97	1918
real	0.97	0.98	0.97	1976
accuracy			0.97	3894
macro avg	0.97	0.97	0.97	3894
weighted avg	0.97	0.97	0.97	3894

DEEP LEARNING MODEL

Why Deep Learning

I would like to experiment with incrementally learning high-level features from data in order to capture more underlying relationships in the data.

Neuron Network Architecture

Model: "sequential"

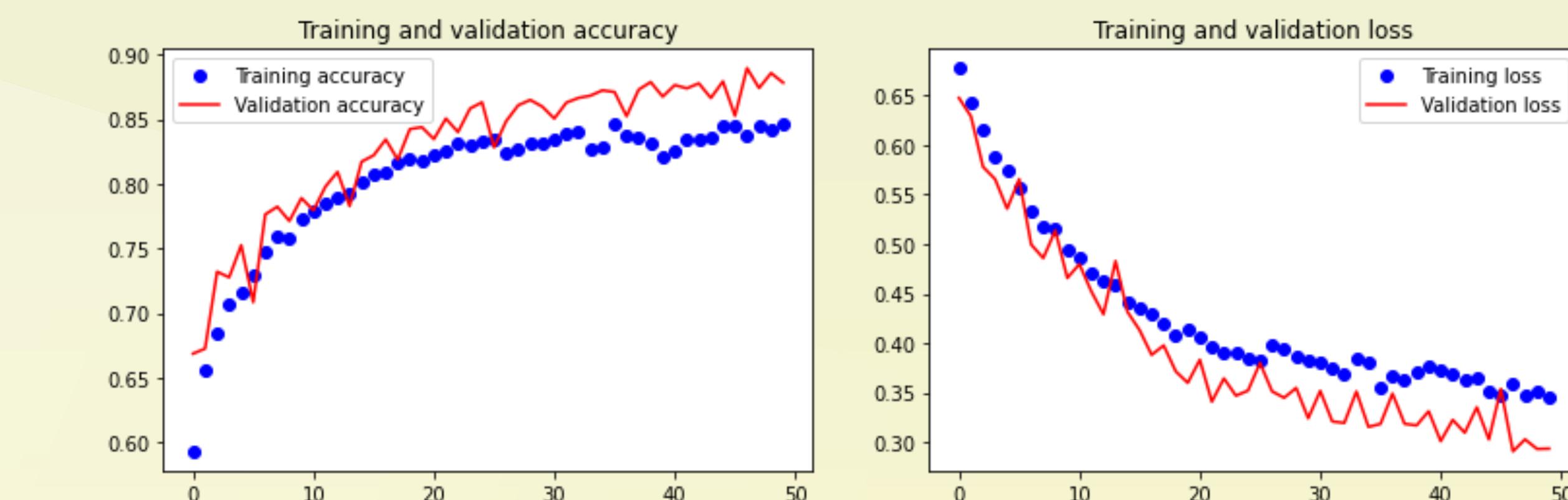
Layer (type)	Output Shape	Param #	Activation
=====			
dense (Dense)	(None, 8192)	103129088	[relu]
dropout (Dropout)	(None, 8192)	0	
dense_1 (Dense)	(None, 4096)	33558528	[relu]
dropout_1 (Dropout)	(None, 4096)	0	
dense_2 (Dense)	(None, 1024)	4195328	
dense_3 (Dense)	(None, 512)	524800	[sigmoid]
dense_4 (Dense)	(None, 256)	131328	[softmax]
dense_5 (Dense)	(None, 64)	16448	
dense_6 (Dense)	(None, 1)	65	[sigmoid]
=====			
Total params:	141,555,585		
Trainable params:	141,555,585		
Non-trainable params:	0		

Hyperparameters Tuning the Deep Learning Model

• Train-test split: 80 % training, 20 % testing. I also use the testing set for validation during training.

- Learning Rate: 0.0001
- Optimizer: Adam
- Loss Function: Binary Crossentropy
- Grid Search: batch size = [64, 128], epochs = [10, 20, 50], 3-fold cv

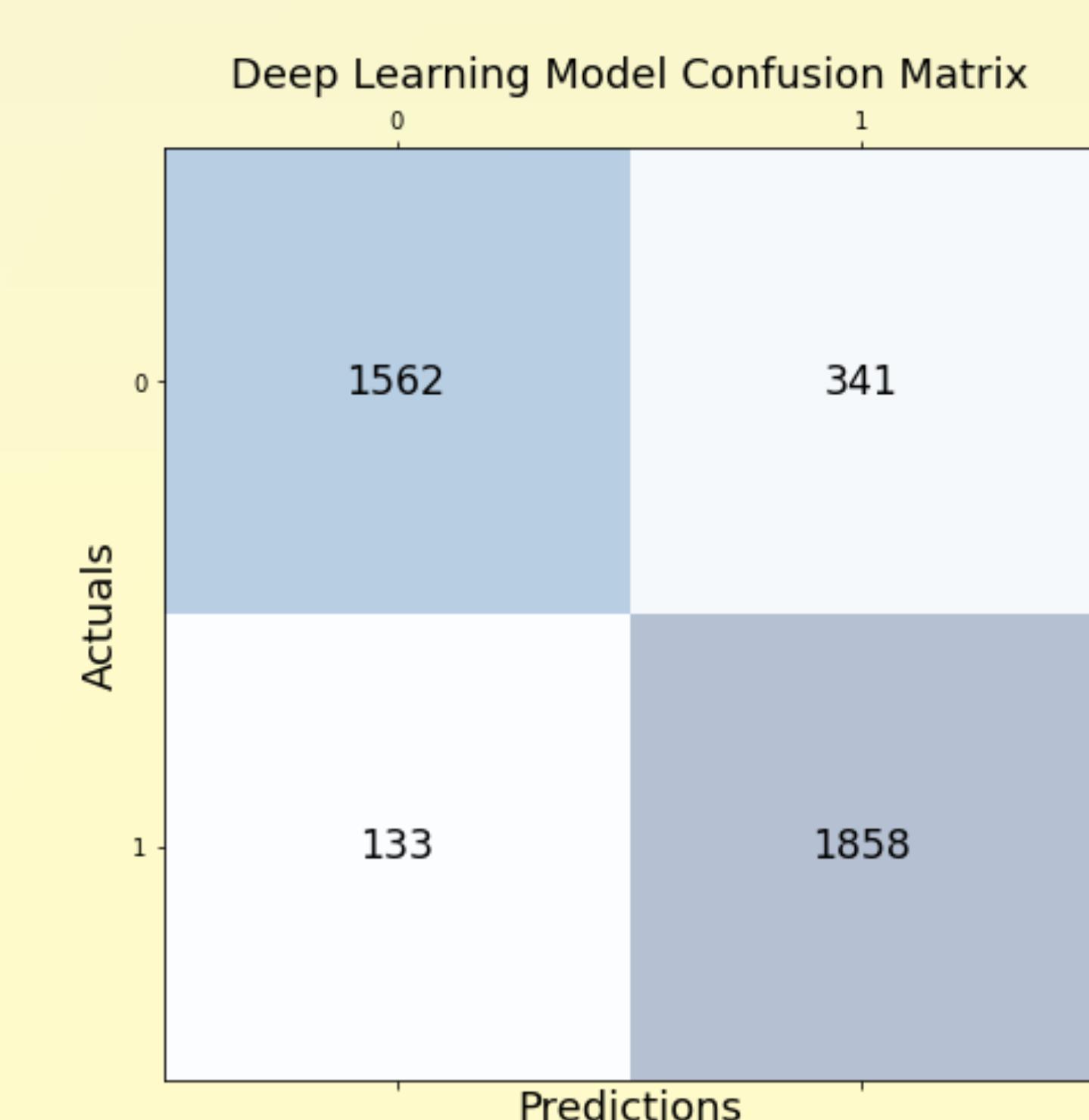
Training and Validation



Deep Learning Model Results

Testing Accuracy: 87.83%

Deep Learning Model Confusion Matrix



Deep Learning Model Classification Report

	precision	recall	F1-score	support
fake	0.92	0.82	0.87	1903
real	0.84	0.93	0.89	1991
accuracy			0.88	3894
macro avg	0.88	0.88	0.88	3894
weighted avg	0.88	0.88	0.88	3894

IMPROVEMENTS AND FURTHER WORKS

Data Fusion

One thing I can do is use a CNN model to directly capture 3D image features at the beginning. After flattening the image data in a CNN model, extract these features and concatenate them to text features to save as a new data.

More Hyperparameters Tuning

If I have an abundance of computation resources, I will use grid or random search with more combinations to find the best estimators in both models.

Source Code

Please refer to my GitHub repo: [twyunting/Deepfake_Video_Classifier2.0](https://github.com/twyunting/Deepfake_Video_Classifier2.0)