Fake Audio Speech Detection

1Hardik Varma
1Student
1Rashtriya Raksha University

Abstract - **Today advantage of deep learning is very vast we can train our images,videos,audio according our need and give view as we want But as like always comes with some disadvantages and risk so in this we discuss about audio that created by deep fakes it is very popular word in latest technology fake audio not only terrifying but actually beginning to happen .Fake audio can be used for malicious purposes it affect directly or indirectly human life. Example:google map use deep learning based navigation so if it modified it will misdirect to us. So in this project I read many papers how to differentiate real or fake audio and I take overview that is I want to implement using Python and deep learning .we can use audio files or video file those are input of our project then we trained our model for uniquely identify features for voice creation and voice detection and apply deep learning technique and find accuracy between real and fake.**

I. Introduction

Now a day's deep leaning uses as generate realistic images audio, videos, etc.if we take real example of deep learning is **Our News Channel.** This gave us some real or fake news we actually don't know where it is original or fake. So, achieve our goal require two main things that one is trained data set like fake audio clips (trained data using AI or Deep Learning) or ASV Proof Challenge can help to provide different language with different voice tone as a example And second thing is Any deep learning and python technique that calculate function for differentiate real or fake audio speech. So, If we use ASV dataset as a input there two types of ASV dataset, first one is text dependent and second one is text-independent. In text dependent we require language tone and content with audio characteristics. It uses i-vector Probabilistic Linear Discriminate Analysis. Or if we use text independent work with different languages it extract feature using Mel Frequency coefficients that will extract features of audio. Not always possible these model gives 100% accuracy for identify real or fake audio clip it gives 80 to 85 percent accuracy because any system is vulnerable so we can use some python library like numpy,pandas or librosa for audio analysis and for train audio data we will use deep learning.

II. Literature Review

In this project mainly we focus on python library numpy,subprocess and some built in tool like realtalk from github for data proceesing purpose and deep fake for train model to identify ration between audio is real or not

Python Numpy

NumPy is a Python package. It stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array. **Numeric**, the ancestor of NumPy, was developed by Jim Hugunin. Another package Numarray was also developed, having some additional functionalities. In 2005, Travis Oliphant created NumPy package by incorporating the features of Numarray into Numeric package. There are many contributors to this open source project

Operations using NumPy

Using NumPy, a developer can perform the following operations

- Mathematical and logical operations on arrays.
- Fourier transforms and routines for shape manipulation.
- Operations related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Librosa

Librosa is powerful Python library built to work with audio and perform analysis on it. It is the starting point towards working with audio data at scale for a wide range of applications such as detecting voice from a person to finding personal characteristics from an audio

It help us to implement:

•Audio signal analysis for music.
•Reference implementation of common methods.
•Building blocks for Music information retrieval (MIR).

SciPy

SciPy, pronounced as Sigh Pi, is a scientific python open source, distributed under the BSD licensed library to perform Mathematical, Scientific and Engineering Computations.

The SciPy library depends on NumPy, which provides convenient and fast N-dimensional array manipulation. The SciPy library is built to work with NumPy arrays and provides many user-friendly and efficient numerical practices such as routines for numerical integration and optimization. Together, they run on all popular operating systems, are quick to install and are free of charge. NumPy and SciPy are easy to use, but powerful enough to depend on by some of the world's leading scientists and engineers.

TensorFlow

TensorFlow is a software library or framework, designed by the Google team to implement machine learning and deep learning concepts in the easiest manner. It combines the computational algebra of optimization techniques for easy calculation of many mathematical expressions.

Features

- It includes a feature of that defines, optimize and calculates mathematical expressions easily with the help of multi-dimensional arrays called tensors.
- It includes a programming support of deep neural network and machine learning techniques.
- I includes a high scalable feature of computation with various data sets.
- TensorFlow uses GPU computing, automating management. It also includes a unique feature of optimization of same memory and the data used.

Deep Learning

Deep learning is a subfield of machine learning where concerned algorithms are inspired by the structure and function of the brain called artificial neural networks.

All the value today of deep learning is through supervised learning or learning from labelled data and algorithms. Each algorithm in deep learning goes through the same process. It includes a hierarchy of nonlinear transformation of input that can be used to generate a statistical model as output.

Deep neural networks, deep belief networks and recurrent neural networks have been applied to fields such as computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, and bioinformatics where they produced results comparable to and in some cases better than human experts have.

How To Train a Model With Python And Deep learning Neural Network

We have to find the optimal values of the weights of a neural network to get the desired output. To train a neural network, we use the iterative gradient descent method. We start initially with random initialization of the weights. After random initialization, we make predictions on some subset of the data with forward-propagation process, compute the corresponding cost function C, and update each weight w by an amount proportional to dC/dw, i.e., the derivative of the cost functions w.r.t. the weight. The proportionality constant is known as the learning rate.The gradients can be calculated efficiently using the back-propagation algorithm. The key observation of backward propagation or backward prop is that because of the chain rule of differentiation, the gradient at each neuron in the neural network can be calculated using the gradient at the neurons, it has outgoing edges to. Hence, we calculate the gradients backwards, i.e., first calculate the gradients of the output layer, then the top-most hidden layer, followed by the preceding hidden layer, and so on, ending at the input layer.

The back-propagation algorithm is implemented mostly using the idea of a computational graph, where each neuron is expanded to many nodes in the computational graph and performs a simple mathematical operation like addition, multiplication. The computational graph does not have any weights on the edges; all weights are assigned to the nodes, so the weights become their own nodes. The backward propagation algorithm is then run on the computational graph. Once the calculation is complete, only the gradients of the weight nodes are required for update. The rest of the gradients can be discarded.

Times is specified, Times Roman or Times New Roman may be used. If neither is available on your word processor, please use the font closest in appearance to Times. Avoid using bit-mapped fonts. True Type 1 or Open Type fonts are required. Please embed all fonts, in particular symbol fonts, as well, for math, etc.

III. Experiment Details

DeepFake Voice Generation

1. The state-of-the-art AI approaches proved to be successful in capturing the linguistic details and producing a smooth, natural human sound(eg. DeepVoice3 or Tacotron2). By the help of holy deep learning, it is possible to extract the language into a meta representation, and synthesize the audio by using this representation.
2. Voice cloning technology is relatively accessible on the Internet today. Montreal-based AI startup Lyrebird provides an online platform that can mimic a person's mimics speech when trained on 30 or more recordings. Baidu last year introduced a new neural voice cloning system that synthesizes a person's voice from only a few audio samples.
3. **Text to speech synthesis (TTS)**: A text-to-speech synthesis system typically consists of multiple stages, such as a text analysis frontend, an acoustic model and an audio synthesis module. There is not yet a consensus on the optimal neural

network architecture for TTS. However, sequence-to-sequence models (Wang et al., 2017; Sotelo et al., 2017; Arık et al., 2017) have shown promising results. An end-to-end generative text-to-speech model synthesizes speech directly from characters.

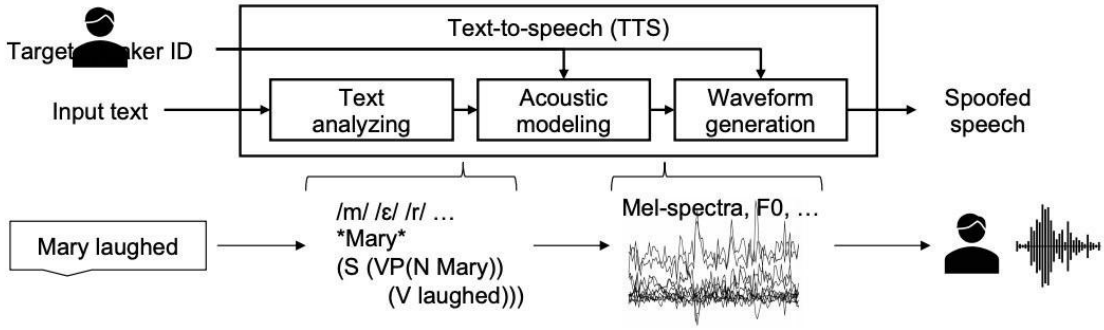


Fig:1 Text To Speech Synthesis

Input format: <text, audio> pairs for single speaker TTS systems and <text, audio, speaker> pairs for multi-speaker TTS systems.

Text Preprocessing

Uppercase all characters in the input text.2.Remove all intermediate punctuation marks.3.End every utterance with a period or question mark.4.Replace spaces between words with special separator characters which indicate the duration of pauses inserted by the speaker between words.

Four different word separators, indicating (i) slurred-together words, (ii) standard pronunciation and space characters, (iii) a short pause between words, and (iv) a long pause between words. For example, the sentence "Either way, you should shoot very slowly," with a long pause after "way" and a short pause after "shoot", would be written as "Either way%you should shoot/very slowly%." with % representing a long pause and / representing a short pause for encoding convenience.

The pause durations can be obtained through either manual labeling or by estimated by a text-audio aligner such as Gentle (Ochshorn & Hawkins, 2017).

4. Voice conversion

VC is the process of converting the source speaker's voice to a sound similar to the target speaker's voice. VC deals with the information that relates to the segmental and suprasegmental features and keep the language content similar. Earlier studies include statistical techniques, such as Gaussian Mixture Model (GMM), Hidden Markov Model (HMM) , unit selection [58], principal component analysis (PCA) [59], and Non-negative matrix factorization (NMF) [60] for VC task. Recently, DNN [61], Wavenet [50], and GAN [48] represent a technology leap.

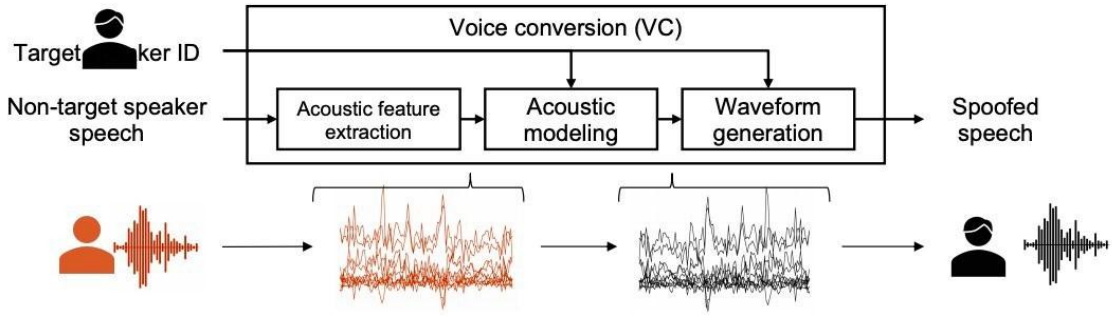


Fig.2 Voice Conversation

5. Neural Voice Cloning

Method 1: Whole Model Adpatation

Fine-tuning a multi-speaker generative model

Method 2: Speaker Embedding Adaptation

Train a model to directly predict new speaker's embedding

Apply these embeddings to multi-speaker generative model

Method 3: Voice Conversion

Non-parallel data requirement

Wave A (speaker dependent) → Phoneme Classifier → Phenomes (speaker independent) → Synthesizer → Wave B

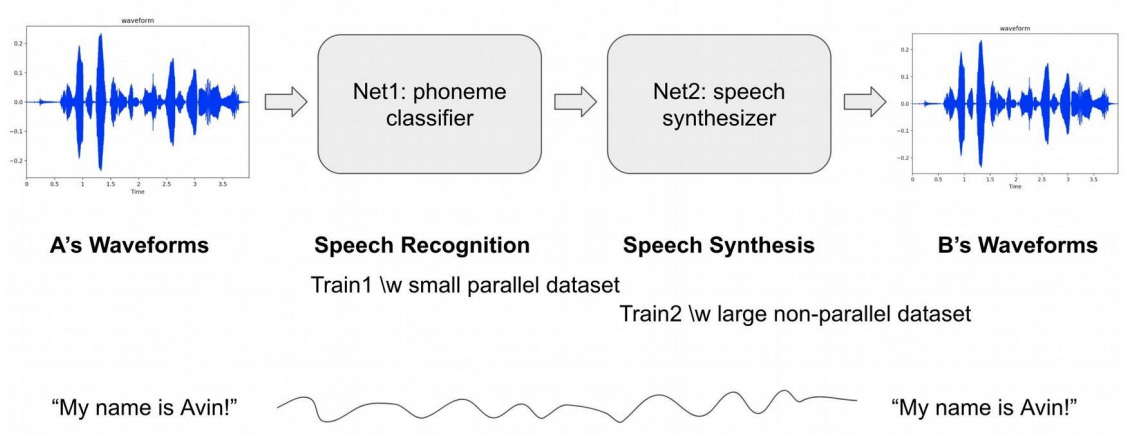


Fig.3 Voice Conversation (Method 2)

Net1 is a classifier

1. Process: wav -> spectrogram -> mfccs -> phoneme dist.
2. Net1 classifies spectrogram to phonemes that consists of 60 English phonemes at every timestep.For each timestep, the input is log magnitude spectrogram and the target is phoneme dist.
3. Objective function is cross entropy loss.
4. TIMIT dataset used.contains 630 speakers' utterances and corresponding phones that speaks similar sentences.
5. *Over 70% test accuracy iv.*

Net2 is a synthesizer

1. Net2 contains Net1 as a sub-network.
2. Process: net1(wav -> spectrogram -> mfccs -> phoneme dist.) -> spectrogram -> wav
3. Net2 synthesizes the target speaker's speeches.The input/target is a set of target speaker's utterances.
4. Since Net1 is already trained in previous step, the remaining part only should be trained in this step.
5. Loss is reconstruction error between input and target. (L2 distance)
6. Datasets
   a) Target1(anonymous female): Arctic dataset (public)
   b) Target2(Kate Winslet): over 2 hours of audio book sentences read by her (private)
7. Griffin-Lim reconstruction when reverting wav from spectrogram.

While a generative model can be trained from scratch with a large amount of audio samples, here, focus is on voice cloning of a new speaker with a few minutes or even few seconds data. It is challenging as the model has to learn the speaker characteristics from very limited amount of data, and still generalize to unseen texts.

Speaker adaptation

a. Speaker adpatation on whole model
   i. For example, If we want to create a fake voice of President Trump (PT), We can use Deep Learning based Text to Speech synthesis models like Tacotron 1, Tacotron 2, DeepVoice1, DeepVoice2, DeepVoice3, WaveRNN, WaveNet, WaveGlow, Char2Wav, VoiceLoop etc.
   ii. We have to pass the (Text, Utterance) pairs of PT as input to these models.
   iii. Acoustic model will learn to generate the spectrograph form character text and vocoder model will learn to generate audio wave from the given spectrograph.
b. Speaker adaptation to a portion of the model
   i. We only train selected layers of the pre-trained TTS model with the PT data.
   ii. This method reduces the training time and also reduces the requirement of large size of training data
c. Speaker adaptation only to the embeddings of PT
   i. Here, we will get speech embedding vector of PT from an encoder
   ii. The TTS model is trained on the (Text, Utterance, Speaker_Embedding) pairs
   iii. We pass PT embeddings to the model and model synthesize the voice conditioned on PT embeddings

Essential Components

a. **Synthesizer**: It is also known as acoustic model. Sequence-to-sequence models (Sutskever et al., 2014; Cho et al., 2014) encode a variable-length input into hidden states, which are then processed by a decoder to produce a target sequence. An attention mechanism allows a decoder to adaptively select encoder hidden states to focus on while generating the target sequence (Bahdanau et al., 2015). Attention-based sequence-to-sequence models are widely applied in machine translation (Bahdanau et al., 2015), speech recognition

(Chorowski et al., 2015), and text summarization (Rush et al., 2015). In speech synthesis, generative models can be conditioned on text and speaker identity [e.g., Arik et al., 2017b]. While text carries linguistic information and controls the content of the generated speech, speaker identity captures characteristics such as pitch, speech rate and accent.

i. Tacotron 2
ii. Deep Speech 3

b. **Vocoder**: It is also known as waveform generator. Neural vocoder has the function of taking the lossy spectrograms (the ones produced by the first system) and overlaying an additional layer of naturalness. This process transforms the spectrograms from something utilitarian into something that's a work of art. In technical terms, the neural vocoder is tasked to invert these spectrograms back to audio while reconstructing the phase.

i. WORLD (Morise et al., 2016)
ii. Griffin-Lim (Griffin & Lim, 1984)
iii. WaveNet (Oord et al., 2016) iv. SampleRNN
v. WaveRNN
vi. WaveGlow

c. Encoder

i. GE2E

2.2 DeepFake Voice Detector

a. Automatic Speaker Verification context

i. Speaker recognition usually refers to both speaker identification and speaker verification. A speaker identification system identifies who the speaker is, while an automatic speaker verification (ASV) system decides if an identity claim is true or false.
ii. The ASV systems are vulnerable to various kinds of spoofing attacks, namely, synthetic speech (SS), voice conversion (VC), replay, twins, and impersonation.
iii. A general ASV system is robust to zero-effort impostors, they are vulnerable to more sophisticated attacks. Such vulnerability represents one of the security concerns of ASV systems. Spoofing involves an adversary (attacker) who masquerades as the target speaker to gain the access to a system. Such spoofing attacks can happen to various biometric traits, such as fingerprints, iris, face, and voice patterns. We are focusing only on the voice-based spoofing and antispoofing techniques for ASV system. The spoofed speech samples can be obtained through speech synthesis, voice conversion, or replay of recorded speech.

b. Imagine the scenario…Your phone rings, you pick up. It's your spouse asking you for details about your savings account — they don't have the account information on hand, but want to deposit money there this afternoon. Later, you realize a bunch of money has went missing! After investigating, you find out that the person masquerading as them on the other line was a voice 100% generated with AI. You've
just been scammed, and on top of that, can't believe the voice you thought belonged to your spouse was actually a fake.

c. To discern between real and fake audio, the detector uses visual representations of audio clips called spectrograms, which are also used to train speech synthesis models. d. Datasets

i. Google's 2019 AVSSpoof dataset contains over 25,000 clips of audio, featuring both real and fake clips of a variety of male and female speakers.

d. Models

i. First, raw audio is preprocessed and converted into a mel-frequency spectrogram this is the input for the model. The model performs convolutions over the time dimension of the spectrogram, then uses masked pooling to prevent overfitting. Finally, the output is passed into a dense layer and a sigmoid activation function, which ultimately outputs a predicted probability between 0 (fake) and 1 (real).

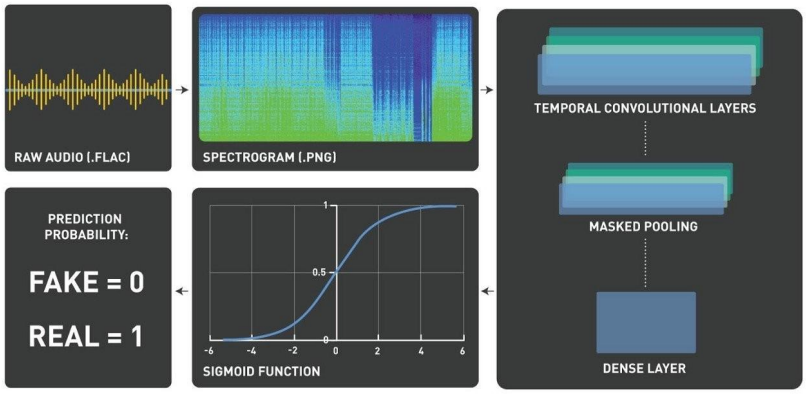


Fig.4 Model (for find fake and real)

ii. T0he baseline model achieved 99%, 95%, and 85% accuracy on the train, validation, and test sets respectively. The differing performance is caused by differences between the three datasets. While all three datasets feature distinct and different speakers, the test set uses a different set of fake audio generating algorithms that were not present in the train or validation set.Temporal Convolution, ResNet, GMM, Light CNN, Fusion, SincNet, ASSERT, HOSA, CVAE

IV. IMPLEMENTATION

1] Envrionment Setup Process

Environment Setup Process

```
[ ] # Installation
    !git clone https://github.com/dessa-oss/fake-voice-detection.git
    %cd /content/fake-voice-detection/
    !pip install -r '/content/fake-voice-detection/code/requirements.txt'
```

```
Cloning into 'fake-voice-detection'...
remote: Enumerating objects: 73, done.
remote: Counting objects: 100% (73/73), done.
remote: Compressing objects: 100% (50/50), done.
remote: Total 73 (delta 27), reused 66 (delta 20), pack-reused 0
Unpacking objects: 100% (73/73), done.
/content/fake-voice-detection
Collecting librosa==0.7.0
  Downloading https://files.pythonhosted.org/packages/ad/6e/0eb0de1c9c4e02df0b40e56f258eb79bd957be79b918511a184268e01720/librosa-0.7.0.tar
     |████████████████████████████████| 1.6MB 2.5MB/s
Collecting numpy==1.15.1
  Downloading https://files.pythonhosted.org/packages/fe/94/7049fed8373c52839c8cde619acaf2c9b83082b935e5aa8c0fa27a4a8bcc/numpy-1.15.1-cp36
     |████████████████████████████████| 13.9MB 315kB/s
Collecting joblib==0.13.0
  Downloading https://files.pythonhosted.org/packages/0d/1b/995167f6c66848d4eb7eabc386aebe07a1571b397629b2eac3b7bebdc343/joblib-0.13.0-py2
     |████████████████████████████████| 276kB 47.8MB/s
Collecting scipy==1.1.0
  Downloading https://files.pythonhosted.org/packages/a8/0b/f163da98d3a01b3e0ef1cab8dd2123c34aee2bafbb1c5bffa354cc8a1730/scipy-1.1.0-cp36-
     |████████████████████████████████| 31.2MB 144kB/s
Collecting scikit_optimize==0.5.2
  Downloading https://files.pythonhosted.org/packages/f4/44/60f82c97d1caa98752c7da2c1681cab5c7a390a0fdd3a55fac672b321cac/scikit_optimize-0
     |████████████████████████████████| 81kB 10.3MB/s
```

2]Uploading Files for processing

```
[ ] # Load data
    !wget https://asv-audio-data-atlas.s3.amazonaws.com/realtalk.zip
    !unzip realtalk.zip
```

```
--2020-05-31 11:38:56--  https://asv-audio-data-atlas.s3.amazonaws.com/realtalk.zip
Resolving asv-audio-data-atlas.s3.amazonaws.com (asv-audio-data-atlas.s3.amazonaws.com)... 52.216.240.60
Connecting to asv-audio-data-atlas.s3.amazonaws.com (asv-audio-data-atlas.s3.amazonaws.com)|52.216.240.60|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 13434461 (13M) [application/zip]
Saving to: 'realtalk.zip'

realtalk.zip        100%[===================>]  12.81M  49.6MB/s    in 0.3s

2020-05-31 11:38:57 (49.6 MB/s) - 'realtalk.zip' saved [13434461/13434461]

Archive:  realtalk.zip
   creating: realtalk/
   creating: realtalk/fake/
  inflating: realtalk/fake/JREa633-0023.wav
  inflating: realtalk/fake/JREa631-0030.wav
  inflating: realtalk/fake/JREa633-0021.wav
  inflating: realtalk/fake/JREa619-0010.wav
  inflating: realtalk/fake/JREa626-0019.wav
  inflating: realtalk/fake/JREa588-0032.wav
  inflating: realtalk/fake/JREa570-0043.wav
  inflating: realtalk/fake/JREa613-0033.wav
  inflating: realtalk/fake/JREa586-0023.wav
  inflating: realtalk/fake/JREa610-0040.wav
  inflating: realtalk/fake/JREa628-0022.wav
  inflating: realtalk/fake/JREa564-0038.wav
   creating: realtalk/real/
  inflating: realtalk/real/JRE1169-0025.wav
```

3]Use Pretrained Model

```
[ ] # Load pretrained model
    %cd /content/fake-voice-detection/code
    !mkdir fitted_objects
    %cd fitted_objects
    !wget https://asv-audio-data-atlas.s3.amazonaws.com/saved_model_240_8_32_0.05_1_50_0_0.0001_100_156_2_True_True_fitted_objects.h5
```

```
/content/fake-voice-detection/code
/content/fake-voice-detection/code/fitted_objects
--2020-05-31 11:39:47--  https://asv-audio-data-atlas.s3.amazonaws.com/saved_model_240_8_32_0.05_1_50_0_0.0001_100_156_2_True_True_fitted_ob
Resolving asv-audio-data-atlas.s3.amazonaws.com (asv-audio-data-atlas.s3.amazonaws.com)... 52.217.16.212
Connecting to asv-audio-data-atlas.s3.amazonaws.com (asv-audio-data-atlas.s3.amazonaws.com)|52.217.16.212|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 98311072 (94M) [binary/octet-stream]
Saving to: 'saved_model_240_8_32_0.05_1_50_0_0.0001_100_156_2_True_True_fitted_objects.h5'

saved_model_240_8_3 100%[===================>]  93.76M  47.2MB/s    in 2.0s

2020-05-31 11:39:49 (47.2 MB/s) - 'saved_model_240_8_32_0.05_1_50_0_0.0001_100_156_2_True_True_fitted_objects.h5' saved [98311072/98311072]
```

4]Loading testing Audio with label

```
[ ] # # Loading testing audio file
    %cd /content/fake-voice-detection/data
    !mkdir inference_data
    %cd inference_data
    !mkdir unlabeled
    !cp /content/fake-voice-detection/realtalk/fake/JREa564-0038.wav /content/fake-voice-detection/data/inference_data/unlabeled
    !cp /content/realtalk/fake/JREa564-0038.wav /content/fake-voice-detection/data/inference_data/unlabeled
    %cd /content/fake-voice-detection/code/
```

```
/content/fake-voice-detection/data
/content/fake-voice-detection/data/inference_data
cp: cannot stat '/content/realtalk/fake/JREa564-0038.wav': No such file or directory
/content/fake-voice-detection/code
```

```
[ ] import os
    import numpy as np
    import subprocess
    import sys
    sys.path.append('/content/fake-voice-detection/code')
    from sklearn.metrics import f1_score, accuracy_score
    from utils import *

    os.environ["CUDA_DEVICE_ORDER"] = "PCI_BUS_ID"  # see issue #152
    os.environ["CUDA_VISIBLE_DEVICES"] = "0"
```

```
WARNING:tensorflow:
The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
```

5]Detect Deepfake

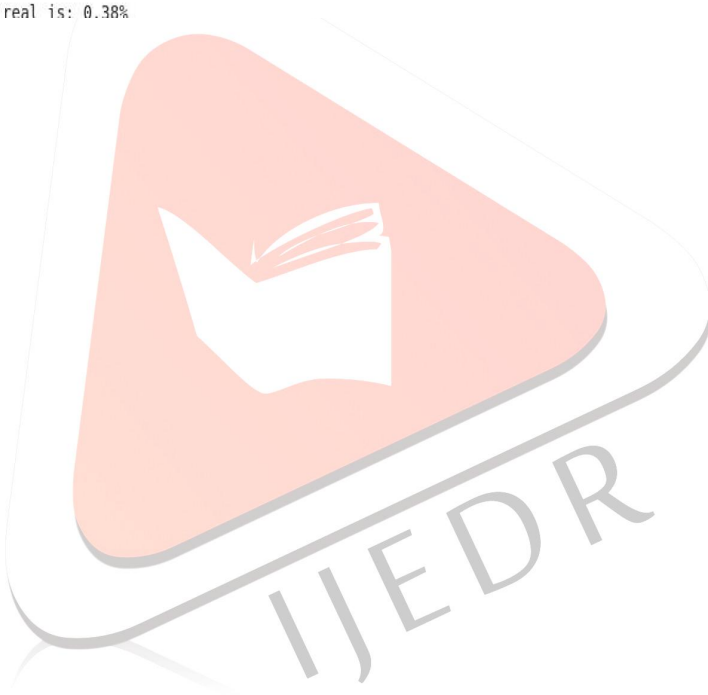
```
[ ] # Step 1: Set the location where your files are stored
    my_file_path = '/content/fake-voice-detection/data/inference_data/'

    # Step 2: Run the model
    real_probability = detect_deepfake(my_file_path)
```

```
  0%|          | 0/1 [00:00<?, ?it/s]Loading inference data from /content/fake-voice-detection/data/inference_data/unlabeled
Loading pretrained model saved_model_240_8_32_0.05_1_50_0_0.0001_100_156_2_True_True_fitted_objects.h5
100%|██████████| 1/1 [00:00<00:00, 17.46it/s]
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:517: The name tf.placeholder is deprecate

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:4138: The name tf.random_uniform is depre

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:131: The name tf.get_default_graph is dep

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:133: The name tf.placeholder_with_default

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:3445: calling dropout (from tensorflow.py
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to `rate = 1 - keep_prob`.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:174: The name tf.get_default_session is c

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:181: The name tf.ConfigProto is deprecate

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:186: The name tf.Session is deprecated. F

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:190: The name tf.global_variables is depr

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:199: The name tf.is_variable_initialized

WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/keras/backend/tensorflow_backend.py:206: The name tf.variables_initializer is
```

6]Get Result

```
[ ] # Step 3: Get the result
    print("The probability of the clip being real is: {:.2%}".format(real_probability))
```

```
The probability of the clip being real is: 0.38%
```



V. RESULTS

Text to Speech Module Results

	Model	Synthesizer	Vocoder	Link
1	TTS-M1	Tacotron2	WaveGlow	https://github.com/NVIDIA/tacotron2/
2	TTS-M2	Tacotron	Griffin-Lim	https://github.com/keithito/tacotron
3	TTS-M3	Tacotron	WaveRNN	https://github.com/mozilla/TTS/
4	TTS-M4	Tacotron2	WaveRNN	https://github.com/mozilla/TTS/
5	TTS-M5	Tacotron2	PWGAN	https://github.com/mozilla/TTS/
6	TTS-M6	Tacotron2	MelGAN	https://github.com/mozilla/TTS/
7	TTS-M7	DeepVoice 3	Griffin-Lim	https://github.com/r9y9/deepvoice3_pytorch
8	TTS-M8	DeepVoice 3	WORLD	https://github.com/hash2430/dv3_world
9	TTS-M9	Tacotron2	WaveNet	https://nvidia.github.io/OpenSeq2Seq/html/index.htm l
10	TTS-M10	DC-TTS	Griffin-Lim	https://github.com/Kyubyong/dc_tts

Audio Style Transfer (VST) Module Results

	Model	Architecture	Link
1	AST-M1	RandomCNN	https://github.com/mazzzystar/randomCNN-voicetransfer
2	AST-M2	VGG16-1DCNN	https://dmitryulyanov.github.io/audio-texturesynthesis-and-style-transfer/
3	AST-M3	CBHG-DNN	https://github.com/andabi/deep-voice-conversion
4	SV2TTS	GE2E (encoder) + Tacotron (synthesizer) + WaveRNN (vocoder)	https://github.com/CorentinJ/Real-Time-VoiceCloning
5	AST-M4	DeepVoice3 Adapter	https://sforaidl.github.io/Neural-Voice-Cloning-With-Few-Samples/

Voice Conversion Module Results

	Model	Architecture	Link
1	VC-M1	MelGAN -VC	https://github.com/marcoppasini/MelGAN-VC
2	VC-M2	ASR-TTS	https://github.com/espnet/interspeech2019-tutorial
3	VC-M3	ESPNet Merlin	https://github.com/r9y9/icassp2020-espnet-ttsmerlin-baseline

Deepfake Audio Detection

	Model	Architecture	Link
1	AD-M1	Temporal Convolution	https://github.com/dessa-oss/fake-voice-detection
2	AD-M2	Encoding Similarity Match	https://github.com/resemble-ai/Resemblyzer
3	AD-M3	CQCC GMM + MFCC ResNet + CQCC ResNet	https://github.com/BhusanChettri/ASVspoof2019
4	ASSERT	Squeeze-Excitation and Residual neTworks	https://github.com/jefflai108/ASSERT
5	AD-M4	ResNet34	https://github.com/rahul-t-p/ASVspoof-2019
6	AD-M5	CPC	https://github.com/jefflai108/Contrastive-PredictiveCoding-PyTorch
7	AD-M6	GMM-MFCC	https://github.com/elleros/spoofed-speech-detection
8	AD-M7	CycleGAN	https://github.com/kstoneriv3/Fake-Voice-Detection

VI. Conclusion

We can use any audio clip as a input or directly copy YouTube video id and put in module it will give us accuracy between real and fake content. All achieved by deep learning technique and python library.

REFERENCES

[1] Massimiliano Todisco and Xin Wang and Ville Vestman and Md Sahidullah and Hector Delgado and Andreas Nautsch and Junichi Yamagishi and Nicholas Evans and Tomi Kinnunen and Kong Aik Lee, "ASVspoof 2019: Future Horizons in Spoofed and Fake Audio Detection", arXiv:1904.05441, 2019.

[2] Wei Ping and Kainan Peng and Andrew Gibiansky and Sercan O. Arik and Ajay Kannan and Sharan Narang and Jonathan Raiman and John Miller, "Deep Voice 3: Scaling Text-to-Speech with Convolutional Sequence Learning", arXiv:1710.07654, 2017.

[3] Jonathan Shen and Ruoming Pang and Ron J. Weiss and Mike Schuster and Navdeep Jaitly and Zongheng Yang and Zhifeng Chen and Yu Zhang and Yuxuan Wang and RJ Skerry-Ryan and Rif A. Saurous and Yannis Agiomyrgiannakis and Yonghui Wu, "Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions", arXiv:1712.05884, 2017.

[4] Francesca Crist. WIRED Insider. (2018, October). How Lyrebird Uses AI to Find Its (Artificial) Voice. Retrieved from https://www.wired.com/brandlab/2018/10/lyrebird-uses-ai-find-artificialvoice.

[5] Sercan O. Arik and Jitong Chen and Kainan Peng and Wei Ping and Yanqi Zhou, "Neural Voice Cloning with a Few Samples", arXiv:1802.06006, 2018.

[6] McAuliffe, Michael & Socolof, Michaela & Mihuc, Sarah & Wagner, Michael & Sonderegger, Morgan. (2017). Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi. 498502. 10.21437/Interspeech.2017-1386.

[7] Yaniv Taigman and Lior Wolf and Adam Polyak and Eliya Nachmani, "VoiceLoop: Voice Fitting and Synthesis via a Phonological Loop", arXiv:1707.06588, 2017.

[8] Sercan O. Arik and Mike Chrzanowski and Adam Coates and Gregory Diamos and Andrew Gibiansky and Yongguo Kang and Xian Li and John Miller and Andrew Ng and Jonathan Raiman and Shubho Sengupta and Mohammad Shoeybi, "Deep Voice: Real-time Neural Text-to-Speech", arXiv:1702.07825, 2017.

[9] Yuxuan Wang and RJ Skerry-Ryan and Daisy Stanton and Yonghui Wu and Ron J. Weiss and Navdeep Jaitly and Zongheng Yang and Ying Xiao and Zhifeng Chen and Samy Bengio and Quoc Le and Yannis Agiomyrgiannakis and Rob Clark and Rif A. Saurous, "Tacotron: Towards End-to-End Speech Synthesis", arXiv:1703.10135, 2017.

[10] Tomoki Hayashi and Ryuichi Yamamoto and Katsuki Inoue and Takenori Yoshimura and Shinji Watanabe and Tomoki Toda and Kazuya Takeda and Yu Zhang and Xu Tan, "ESPnet-TTS: Unified, Reproducible, and Integratable Open Source End-to-End Text-to-Speech Toolkit", arXiv:1910.10909, 2019.

[11] Jemine, Corentin, "Automatic Multispeaker Voice Cloning" (Master thesis), http://hdl.handle.net/ 2268.2/6801, 2019.

[12] Nal Kalchbrenner and Erich Elsen and Karen Simonyan and Seb Noury and Norman Casagrande and Edward Lockhart and Florian Stimberg and Aaron van den Oord and Sander Dieleman and Koray Kavukcuoglu, "Efficient Neural Audio Synthesis", arXiv:1802.08435, 2018.

[13] Ye Jia and Yu Zhang and Ron J. Weiss and Quan Wang and Jonathan Shen and Fei Ren and Zhifeng Chen and Patrick Nguyen and Ruoming Pang and Ignacio Lopez Moreno and Yonghui Wu, "Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis", arXiv:1806.04558, 2018.

[14] Li Wan and Quan Wang and Alan Papir and Ignacio Lopez Moreno, "Generalized End-to-End Loss for Speaker Verification", arXiv:1710.10467, 2017.

[15] T. Kinnunen, K.-A. Lee, H. Delgado, N. Evans, M. Todisco, M. Sahidullah, J. Yamagishi, D.-A. Reynolds, "t-DCF: a Detection Cost Function for the Tandem Assessment of Spoofing Countermeasures and Automatic Speaker Verification", Proc. Odyssey 2018 - The Speaker and Language Recognition Workshop.

[16] Kamble, Madhu & Patil, Hemant. (2018). A Survey on Replay Attack Detection for Automatic Speaker Verification (ASV) System. 10.23919/APSIPA.2018.8659666.