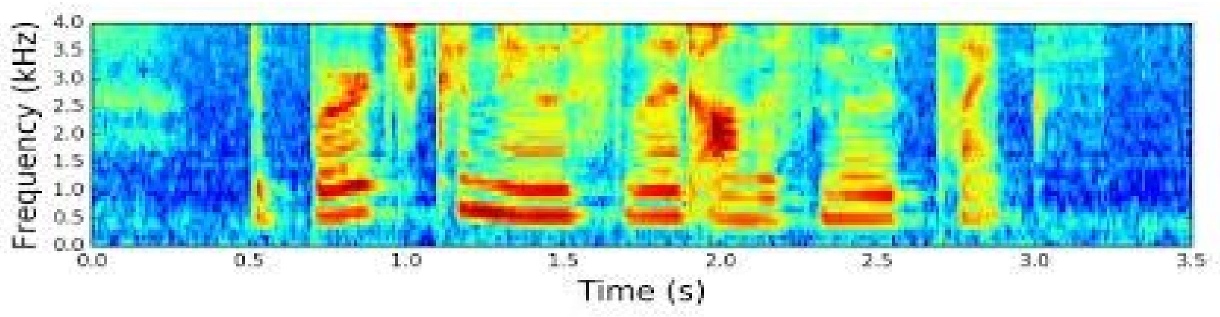
End-to-end deep learning based voice cloning

General Definition of Voice cloning: To mimik one's voice with tools and techniques, either manual or automated is known as voice cloning.

Deep learning based definition: **Voice cloning** is a deep-learning algorithm that takes in **voice** recordings of an individual and is able to synthesize a **voice** is very similar to the original **voice**.

On this platform we will discuss some state-of-the-art techniques to clone voice with deep learning. We will also learn the concepts of mel-spectrograms, frequency-time graphs, noise and synthesis. Let's start with the basic definitions of mentioned keywords.

Mel-spectrograms:

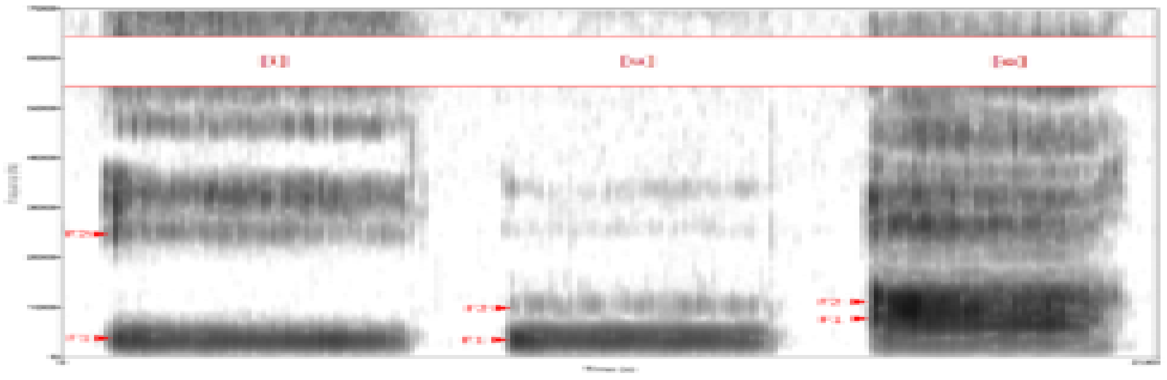


Mel-spectrograms is the way to extract the most useful information from voice or noise signals as visualizing them in time domain only will not be enough and can cause loss of information.

"In sound processing, the **mel-frequency cepstrum** (**MFC**) is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency." [Mel Spectrograms]

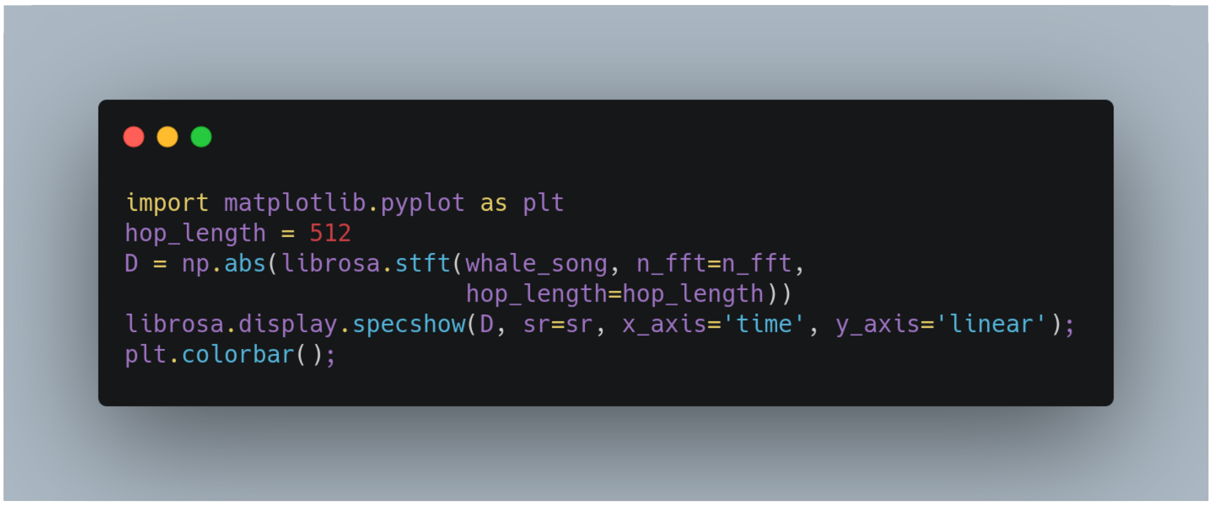
It is a visual representation of the spectrum of frequencies of a signal as it varies with time. When applied to an audio signal, spectrograms are sometimes called **sonography**, **voiceprints**, or **voicegrams**. When the data is represented in a 3D plot they may be called **waterfalls**.

Color segments in mel-spectrograms show the events of voice signal, the louder the event the brighter the colors.The lines in spectrogram is also known as Formants, (a **formant** is the spectral shaping that results from an acoustic resonance of the human vocal tract.).



[Formant]

These mel-spectrograms can be generated with many tools through sound files (e.g. .wav, .mp3 etc) like Audacity etc.
This can also be generated with Python Matplotlib and librosa (powerful sound analysis library) as follows,



```
import matplotlib.pyplot as plt
hop_length = 512
D = np.abs(librosa.stft(whale_song, n_fft=n_fft,
                        hop_length=hop_length))
librosa.display.specshow(D, sr=sr, x_axis='time', y_axis='linear');
plt.colorbar();
```

To learn more about mel-spectrograms, [Mel-Spectrograms Tutorial]

Let’s dive a bit more and understand how to generate, preprocess and train data for voice cloning.

Regarding voice data few questions may arise.

1. How much data is required to get good speaking models ?
2. Do we need to split files or process a whole big file in a neural network ?
3. What must be the quality level of files ?
4. Do only voice files be sufficient ?
5. How to preprocess voice and text files ?
6. Any data set available for this task ?
7. Will this data generation process be automated or manual ?
8. Etc

Let’s answer all questions one-by-one .
How much data is required to get good speaking models ?
To get good quality output one must gather at least half an hour (30 minutes) data set of voice. I have conducted so many tests with less data and no success.

Do we need to split files or process a whole big file in a neural network ?
Yes, the length of voice file depends on how long a sentence one wants to generate during inferencing the model.

Long sequences can be harder to get learnt by neural network, as the architecture we are going to use is composed on Encoder-decoder architecture and thus long voice clips are very computationally expensive to train, but as per my research if the quality of data set good enough, the continuation of different clips generated at the same time gives very good listening.

What must be the quality level of files ?
There are certain parameters which have to be met to generate good quality data set, to understand it better, one needs to understand already online available data set as LJSpeech [https://keithito.com/LJ-Speech-Dataset/]. This is a very good quality data set for voice cloning tasks. Each audio file is a single-channel 16-bit PCM WAV with a sample rate of 22050 Hz, more details of parameters can be found on the above mentioned site.

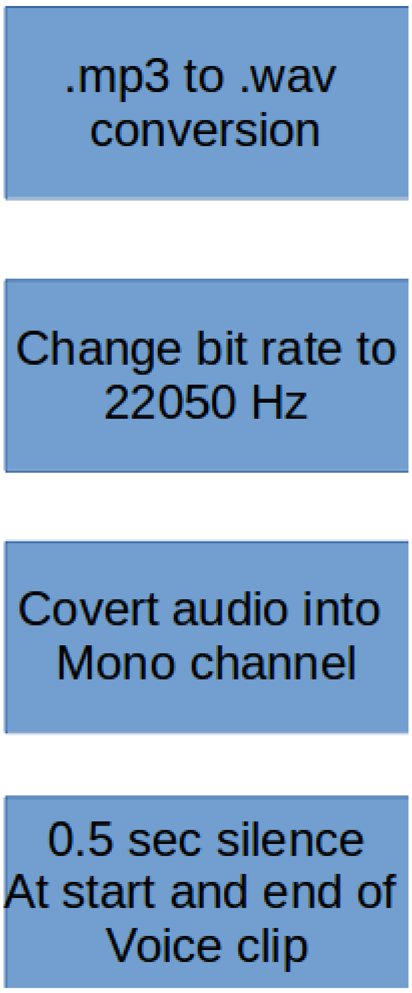
Do only voice files be sufficient ?
No, one also needs text transcript as features to data set. Both voice clips and text transcripts are required.

How to preprocess voice and text files ?
All voice clips and text transcript should be preprocessed as per the instructions in LJSpeech site, each line in text file should be a text script of voice clip.

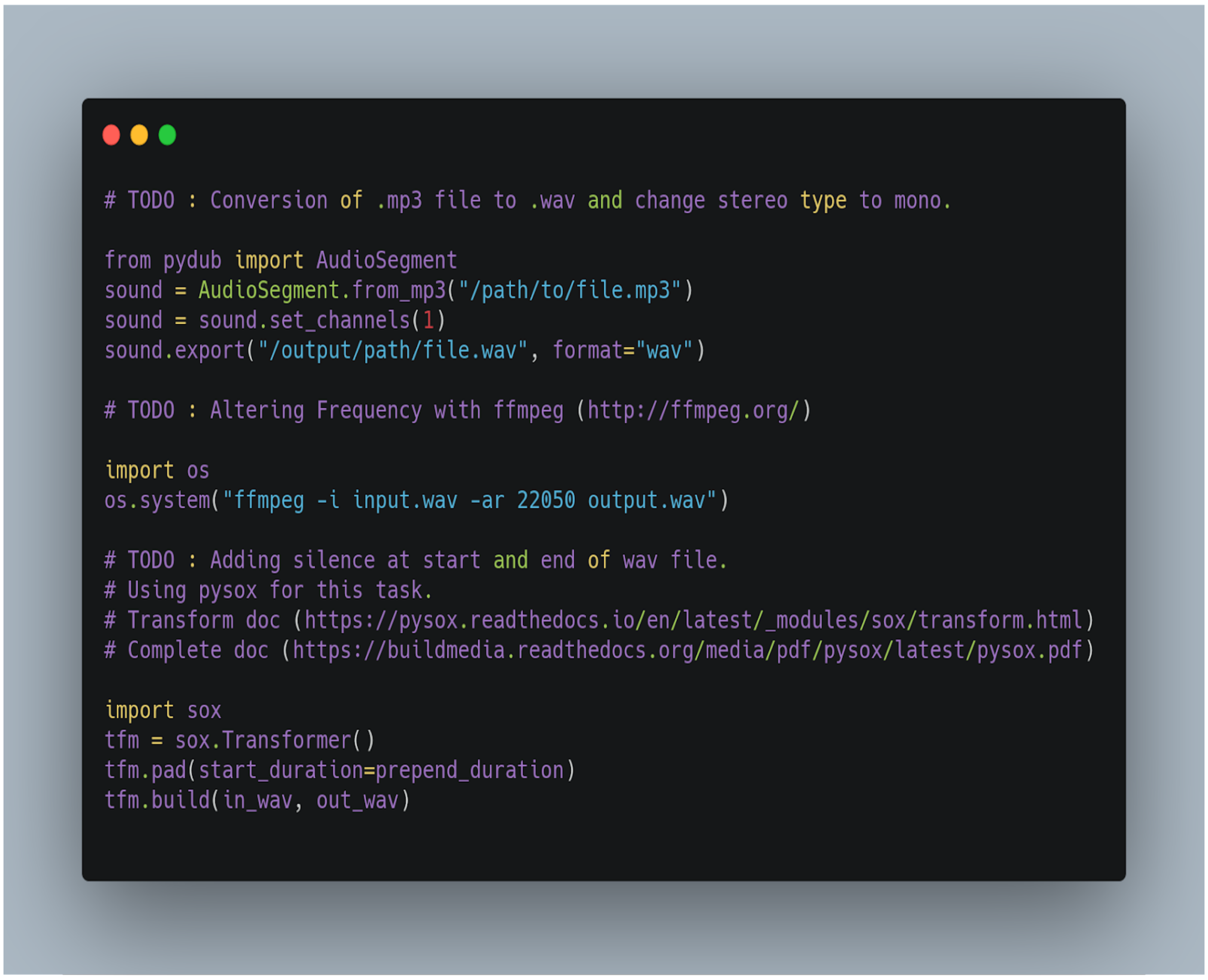
E.g. sampleVoice.wav → sample Voice | this is test voice
More can be found on LJSpeech site.

Any data set available for this task ?
LJSpeech, LibriSpeech, mozilla,VCTK etc
[Voice Data sets]

Will this data generation process be automated or manual ?
Python comes with very powerful and enrich sound libraries like librosa, wave, ffmpeg etc the whole process of voice transformation and clipping can be done automatically.


.mp3 to .wav conversion
Change bit rate to 22050 Hz
Covert audio into Mono channel
0.5 sec silence At start and end of Voice clip

Below attached are code references for the above mentioned preprocess techniques.



```
# TODO : Conversion of .mp3 file to .wav and change stereo type to mono.

from pydub import AudioSegment
sound = AudioSegment.from_mp3("/path/to/file.mp3")
sound = sound.set_channels(1)
sound.export("/output/path/file.wav", format="wav")

# TODO : Altering Frequency with ffmpeg (http://ffmpeg.org/)

import os
os.system("ffmpeg -i input.wav -ar 22050 output.wav")

# TODO : Adding silence at start and end of wav file.
# Using pysox for this task.
# Transform doc (https://pysox.readthedocs.io/en/latest/_modules/sox/transform.html)
# Complete doc (https://buildmedia.readthedocs.org/media/pdf/pysox/latest/pysox.pdf)

import sox
tfm = sox.Transformer()
tfm.pad(start_duration=prepend_duration)
tfm.build(in_wav, out_wav)
```

Let's dive deeper and evaluate the deep learning based models we are going to use.

This section is divided into two main parts.

1. Mel generation.
2. Synthesis.

Mel Generation.

This is a section in which your voice will be trained against the text you provide, I have tested so many architectures for this task like,

1. Deep Voice (PyTorch) [Deep Voice]
2. Mozilla TTS (PyTorch) [Mozilla TTS]
3. Tacotron-2 (TensorFlow) [Tacotron-2]

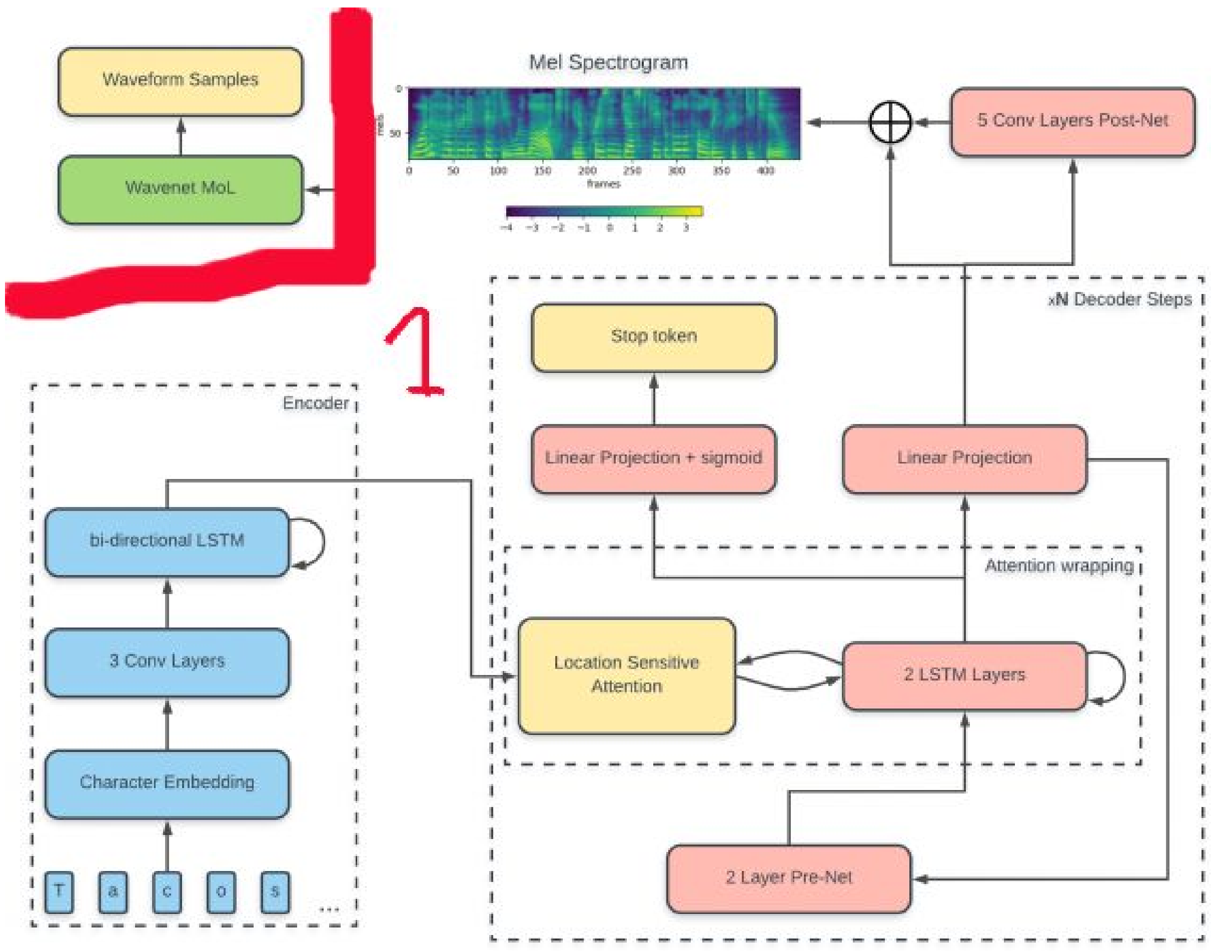
And found Tacotron-2 to be very promising and effective.

All necessary installation commands and guides are present on GitHub repo with complete directory structure and requirements.

All one needs to do, go to this link [Rayhane-mamah Tacotron-2] and follow the instructions. Here, I am sharing my steps of operations on Linux OS. I haven't tried on Windows due to a lack of libraries but if someone can find them please share.

1. Clone the repo.
    - Git clone "https://github.com/Rayhane-mamah/Tacotron-2.git".
2. Install system libraries.
    - "apt-get install -y libasound-dev portaudio19-dev libportaudio2 libportaudiocpp0 ffmpeg libav-tools".
3. Install Python dependencies. (Use any Python 3 version => 3.5)
    - Pip install -r requirements.
4. Make dir for data set.
    - Mkdir "LJSpeech-1.1" and cd LJSpeech-1.1
    - Download data set from LJSpeech here, or put your custom data here.
    - Put all .wav files in LJSpeech-1.1/wavs and text script as metadata.csv in LJSpeech-1.1/
5. Run peprocess.py file.
    - python preprocess.py
6. Run Tactron only training (Reason will be discussed later)
    - python train.py --model='Tacotron'
7. Overfit your data training (set training steps to 250,000) and evaluate model file after 100,000 time steps, you can change "tacotron_train_steps" param in train.py line 114, as per my experience good model files start generating after 100,000 and before 250,000 also depends on the pitch of voice data set.
8. Your model files will be generated in "logs-Tacotron/taco_pretrained".
9. Evaluate your model by changing the model number as per in taco_pretrained in checkpoint file present in the same dir.
10. Evaluate model after changing its name as follows,
    - python synthesize.py --model='Tacotron'
11. Your generated wav file be placed in "tacotron_output/logs-eval/wavs".
12. There will be two types of files in this dir, 1- linear and 2-mel, use mel generated files.
13. Continue training and use best model files.

Tacotron Architecture:



The architecture of Tacotron is composed of Encoder-Decoder layers, with three (3) encoder layers each of 256 LSTM units, and two (2) decoder layers each of 1024 LSTM layers. Let's talk about the architecture a bit and we will be using only part 1 of this architecture.
As in Tacotron, mel spectrograms are computed through a short time Fourier transform (STFT) using a 50 ms frame size, 12.5 ms
frame hop, and a Hann window function.
It is composed of an encoder and a decoder with attention mechanism. The encoder is designed to convert input characters into hidden feature representation embedding, it then passed to stack of three (3) convolution layers each of 512 filters with shape of 5x1 with batch normalization and ReLU activations.These conv layers help in modeling long-term context in input characters sequence. The output of these conv layers is then passed to single bi-directional LSTM layer of 512 uits (256 in each direction) in order to generate encoded features representation.Then this

encoded feature set is forwarded to attention network(location-sensitive attention mechanism is used) converts this into fixed-length context vector for each output step of decoder.
The decode side is of an autoregressive recurrent neural network which predict a mel spectrogram from encoded input sequence one frame at a time. Small pre-net of 2 fully connected layers of 256 hidden ReLU units is used to access the prediction from previous time step, then pre-net output and attention context vector are concatenated and forwarded to 2 uni-directional LSTM layers of 1024 units. Finally the predicted mel spectrogram is passed to 5-layer convolutional post-net of 512 filters with shape 5x1 with batch normalization and tanh activations except final layer which predicts residual to add to the prediction to improve the overall reconstruction.

More can be found in Tacotron Paper [Tacotron Paper].

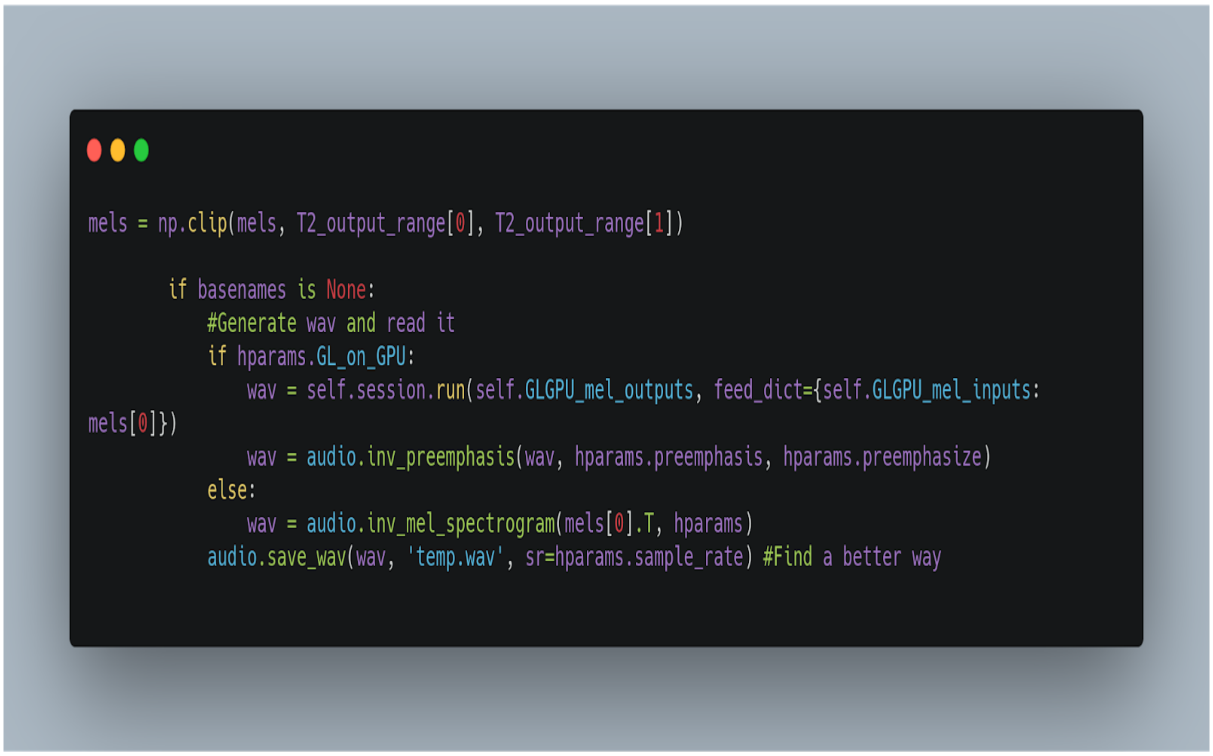
Congrats we have completed the second and very important stage of creating voice from text, with the help of Tacotron we are able now to generate mels with input text. Let's talk about the next and last stage of this tutorial, i.e. Synthesizer.

Synthesizer helps in generating speech from from mel-spectrograms as it converts Frequency domain signals back into time-domain signals. First we need to understand following questions,

1. How currently Tacotron is generating speech with mels ?
2. Why are we not using WaveNet ?
3. What are other better options ?

How currently Tacotron is generating speech with mels ?
Tacotron is using very basic source of synthesizer which is as same as Griffin-lim, but they claim it is exactly Griffin-lim [Griffin Lim]

The code snippet used for this purpose is below.



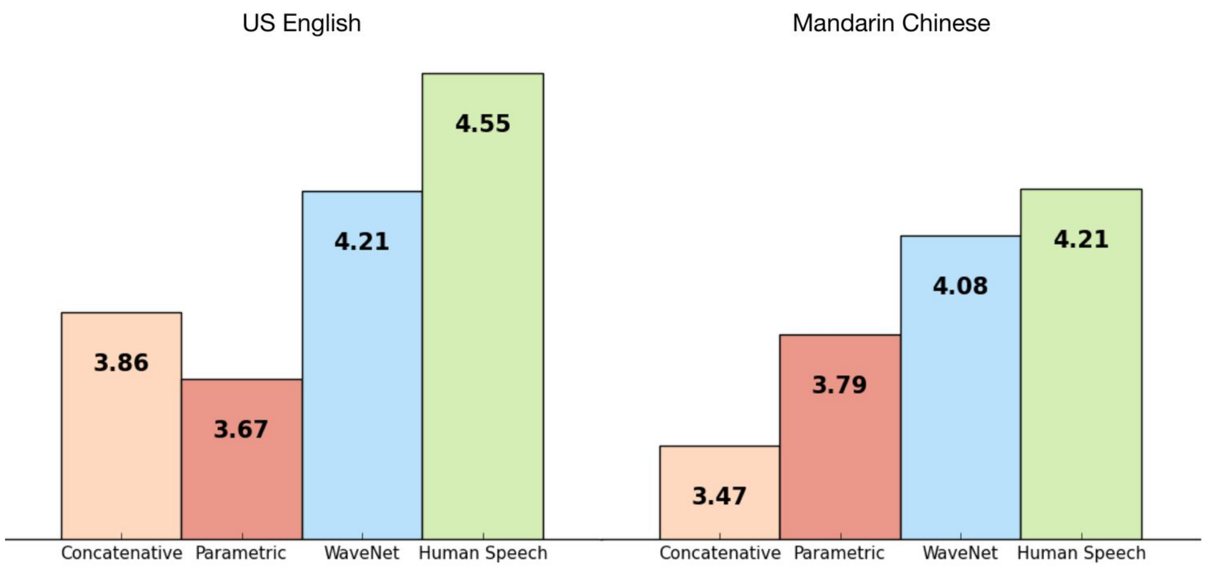
```
mels = np.clip(mels, T2_output_range[0], T2_output_range[1])

        if basenames is None:
            #Generate wav and read it
            if hparams.GL_on_GPU:
                wav = self.session.run(self.GLGPU_mel_outputs, feed_dict={self.GLGPU_mel_inputs:
mels[0]})
                wav = audio.inv_preemphasis(wav, hparams.preemphasis, hparams.preemphasize)
            else:
                wav = audio.inv_mel_spectrogram(mels[0].T, hparams)
            audio.save_wav(wav, 'temp.wav', sr=hparams.sample_rate) #Find a better way
```

You can find this code segment in Tacotron in tacotron/synthesizer.py between lines 160 and 170.

Why are we not using WaveNet ?

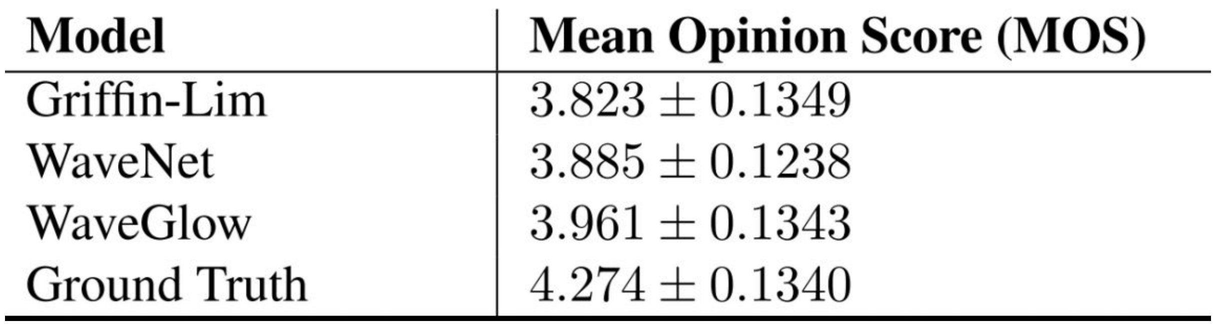
In order to draw a comparison between WaveNet and existing speech synthesizing approaches, subjective *5-scale Mean Opinion Score (MOS) tests* were conducted. In the MOS tests, subjects (humans) were presented with speech samples generated from either of the speech synthesizing systems and were asked to rate the naturalness of the speech sample in a five-point scale score (1: Bad, 2: Poor, 3: Fair, 4: Good, 5: Excellent).



It's clearly visible from the bar-charts that WaveNet achieved naturalness above 4.0 in the 5-scale MOS tests, which were significantly better than those from the other baseline systems and very close to actual human speech.

What are other better options ?

As AI is growing platform and every day we get better algorithms,
Let's discuss very challenging synthesizer i.e. WaveGlow.
According to my experiments and experiments conducted by others as well
WaveGlow is proved to be better than WaveNet, and MOS speaks about it.



Model	**Mean Opinion Score (MOS)**
Griffin-Lim	3.823 ± 0.1349
WaveNet	3.885 ± 0.1238
WaveGlow	3.961 ± 0.1343
Ground Truth	4.274 ± 0.1340

According to NVIDIA "WaveGlow: a flow-based network capable of generating high quality speech from mel-spectrograms.Our PyTorch implementation produces audio samples at a rate of 4850 kHz on an NVIDIA V100 GPU. Mean Opinion Scores show that it delivers audio quality as good as the best publicly available WaveNet implementation."

Find out more in GitHub repo. [NVIDIA Wave Glow]

Follow the instructions for installation and training if you want to train your own model, as per my experiments as it is a way to convert frequency-domain signals to time-domain signals the mels which lies in the same range of pitch can use the same trained synthesizer and experimentally it is also proved. Thus for my mels for light female voices I used the same LJSpeech trained synthesizer models.

For synthesizing put your mel files name in test_files.txt, then run the following command which will create .pt files to be processed by model.

python mel2samp.py -f test_files.txt -o . -c config.json

Then run following commands and replace model name if different,

```
ls *.pt > mel_files.txt
python3 inference.py -f mel_files.txt -w checkpoints/waveglow_10000 -o .
--is_fp16 -s 0.6
```

You can check my implementations on following link,

http://151.253.132.152:7000/voiceover

In which I have also solved long sentence continuation issues and fasten complete Tacotron-WaveGlow process also I completed the whole process in one step.