# SINGAPOREAN SPEECH CLASSIFICATION WITH NEURAL NETWORKS

BY ANDY CHAN, DSI17

Machines understand meh?

# AGENDA

- Problem Statement
- Data Collection and Scoping
- Demystifying Audio Processing
- EDA and Findings
- Model Performance and Evaluation
- Conclusion, Limitations and Suggestions for Further Research

# LISTENING GAME!

*What did you hear?*

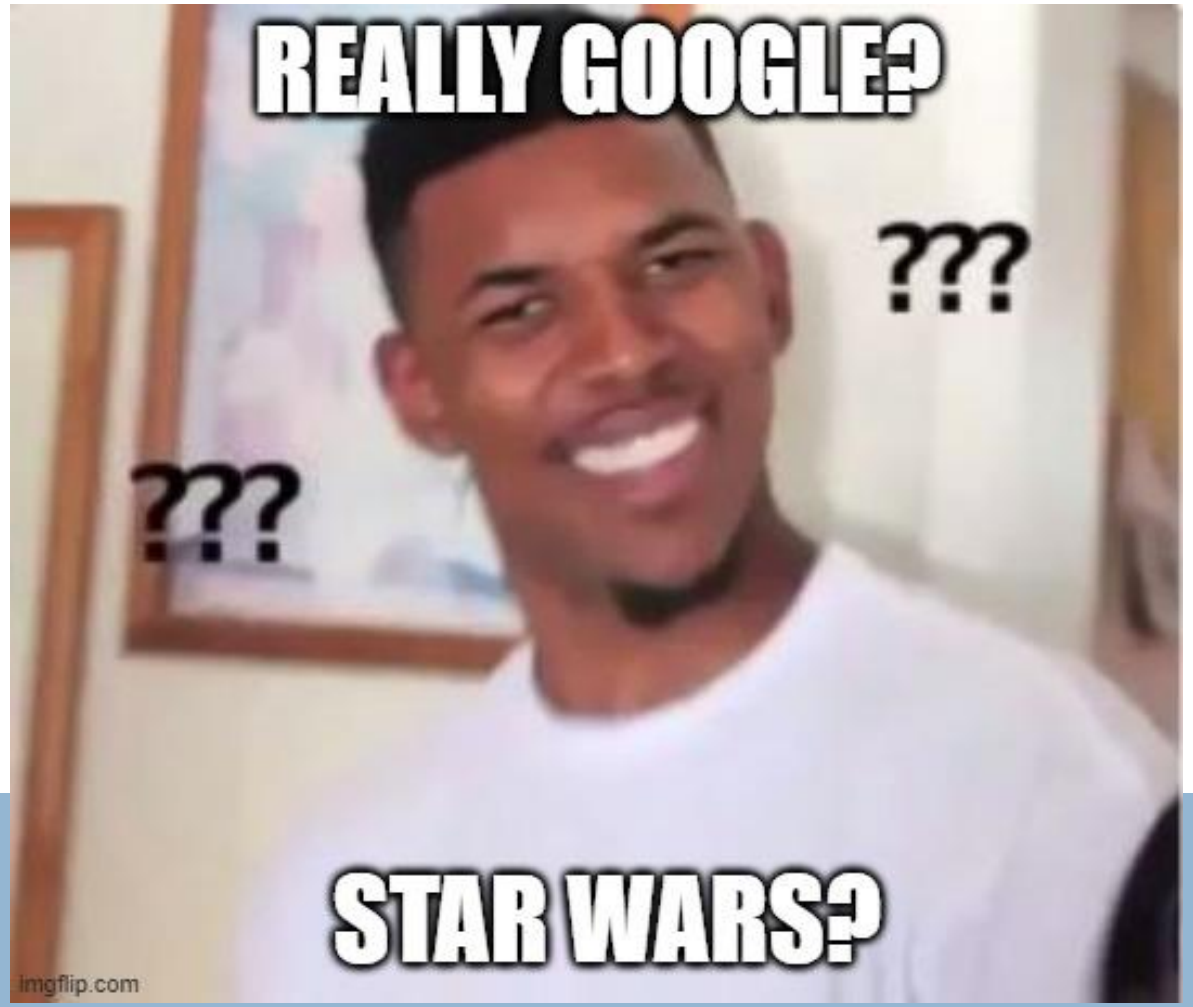
**Apples**

```
In [5]: import speech_recognition as sr

In [34]: from os import path

         AUDIO_FILE = "./4740062_flowers.wav"

In [35]: r = sr.Recognizer()
         with sr.AudioFile(AUDIO_FILE) as source:
             audio = r.record(source)

         r.recognize_google(audio)

Out[35]: 'Star Wars'
```

**Flowers?**

**Star Wars!**

# PROBLEM STATEMENT



# PROBLEM STATEMENT

- How well can machine learning algorithms classify Singaporean-accented English?
- How feasible is it as a speech-to-text engine (based on computation speed)?
- Multiclassification model, using neural networks
- Dataset: IMDA National Speech Corpus
- Metric: Accuracy and Speed
- Stakeholder: Business (e.g. call centres, restaurants), visually impaired people

NDP organisers apologise for Tamil language errors in NDP 2020 evening show


A line of lyrics with a Tamil phrase that was supposed to read "My Singapore" had strokes and letters in wrong places. PHOTO: KARTHIKEYAN SOMASUNDARAM/FACEBOOK

# DATA COLLECTION AND SCOPING OF PROJECT

# ABOUT THE DATA SOURCE

INFOCOMM MEDIA DEVELOPMENT AUTHORITY

**IMDA National Speech Corpus (NSC)**

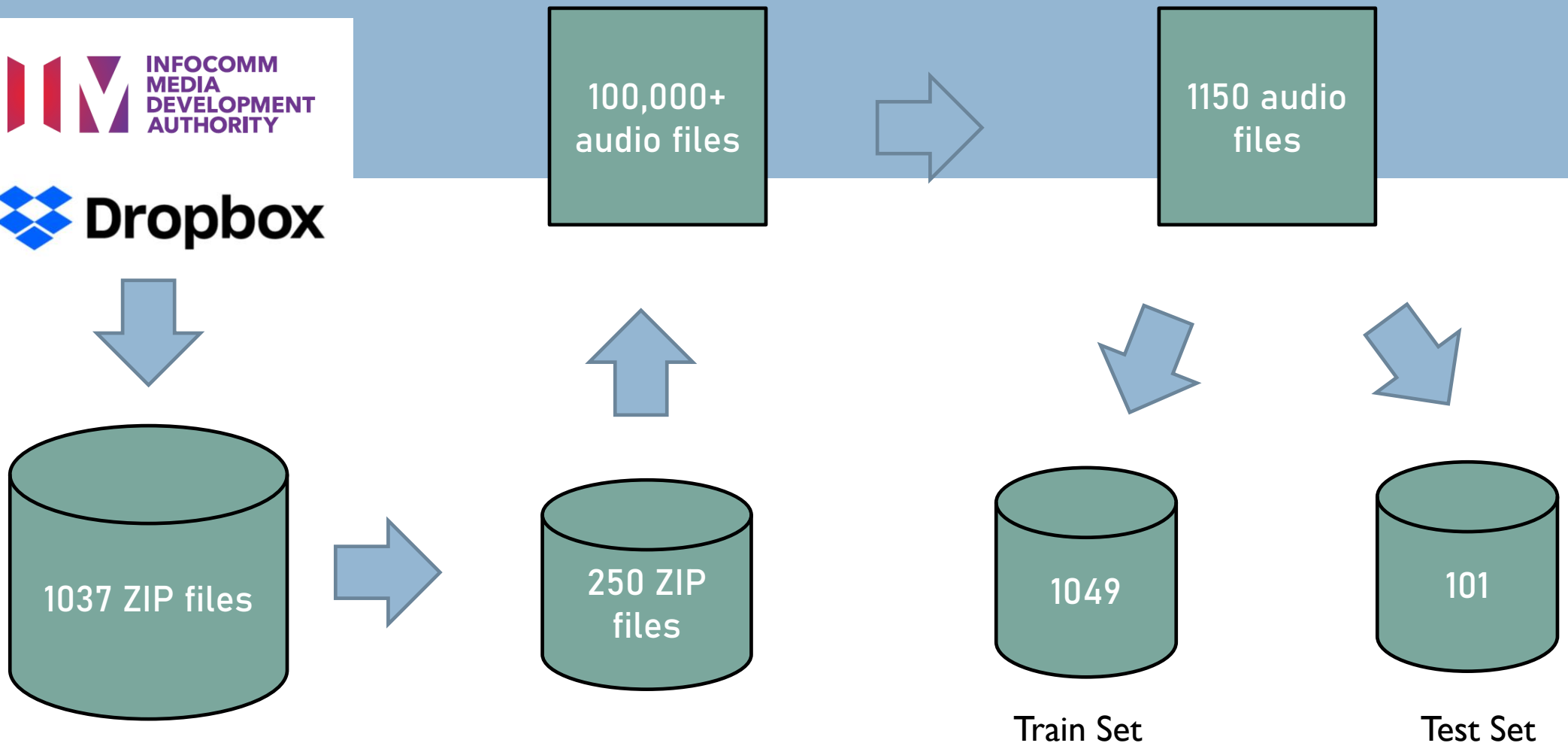
- >1 TB of Singaporean Speeches Recorded in various environments and with different settings
- Aim:
  - Help improve the recognition of locally accented English
  - Help start-ups and tech companies develop speech models

**Challenges**

- Storage Space
- Time

INFOCOMM MEDIA DEVELOPMENT AUTHORITY

Dropbox




**All these with just 3 blocks of code and a few minutes!**

# Delete Unwanted ZIP Files

```
# where I keep all the audio files
path = 'C:\\Users\\Andy Chan\\Desktop\\audio_original'

# get the name of the zip file of each speaker
zip_names = []
for f in selected_speakers_2:
    if len(str(f)) == 1:
        zip_names.append(f'SPEAKER000{f}.zip')
    elif len(str(f)) == 2:
        zip_names.append(f'SPEAKER00{f}.zip')
    elif len(str(f)) == 3:
        zip_names.append(f'SPEAKER0{f}.zip')
    else:
        zip_names.append(f'SPEAKER{f}.zip')

# for each file in "path"
for f in os.listdir(path):
    # if file is not in zip_names
    if f not in zip_names:
        full_file_path = os.path.join(path, f)
        # remove the file, keeps the files which are in zip_names
        os.remove(full_file_path)
```

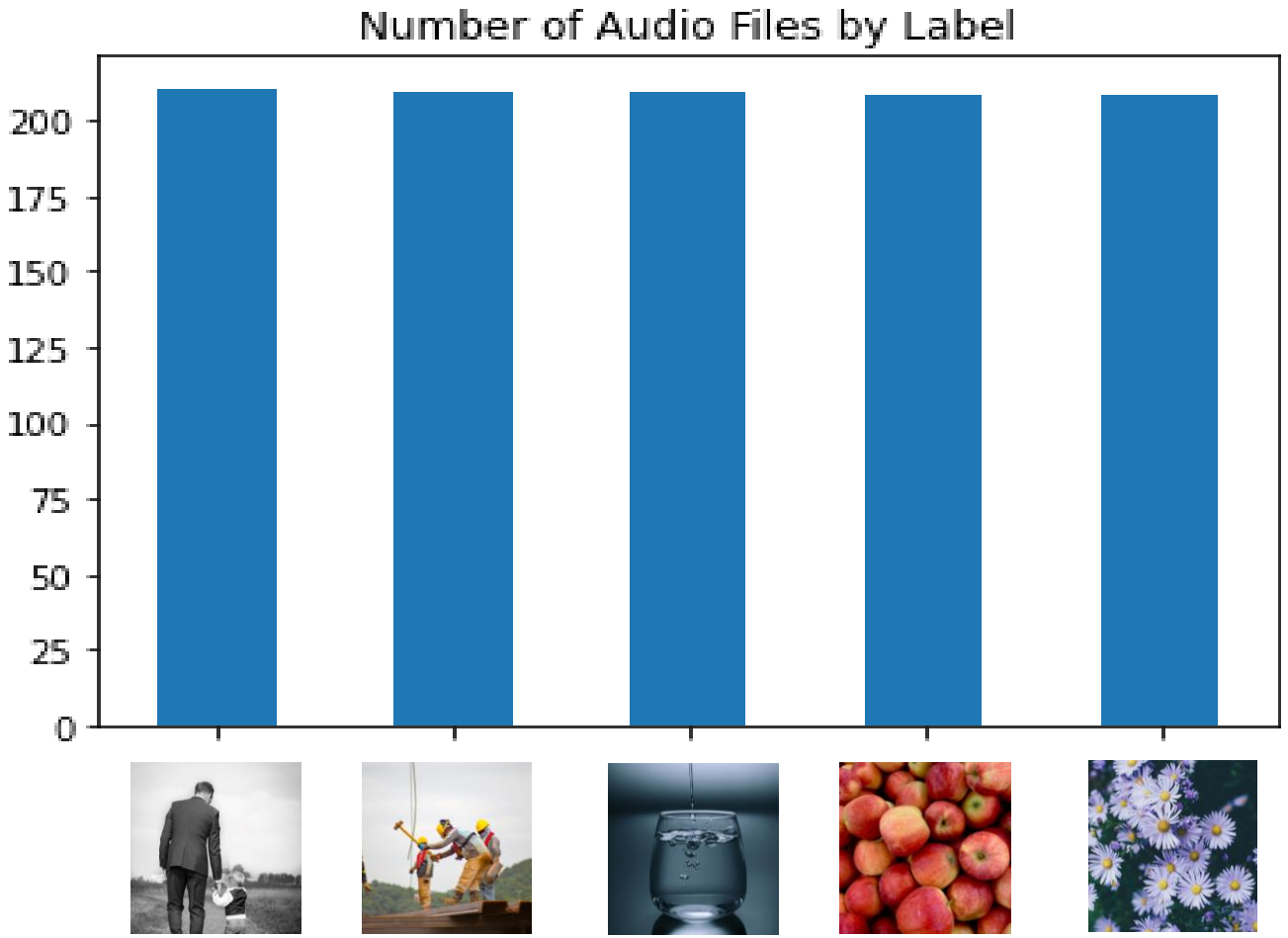
## Git Bash to Unzip Remaining Zip Files

```
unzip -j SPEAKER0003.zip 'SPEAKER0003/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0006.zip 'SPEAKER0006/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0007.zip 'SPEAKER0007/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0008.zip 'SPEAKER0008/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0059.zip 'SPEAKER0059/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0116.zip 'SPEAKER0116/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0144.zip 'SPEAKER0144/SESSION0/*' -d capstone_audios/
unzip -j SPEAKER0147.zip 'SPEAKER0147/SESSION0/*' -d capstone_audios/
```

## Delete Unwanted Audio Files

```
# where I keep all the audio files
path = 'C:\\Users\\Andy Chan\\Desktop\\audio_original\\capstone_audios'

# create a list of file names which I am interested in
file_list_2 = [i.tolist() for i, _ in speaker_list_2]

# file path has 9 digits, where the first digit will be zero, followed b
# audio code of 3 digits. Hence, they need to pad with zeros for if they

audio_paths = []
for f in file_list_2:
    if len(str(f)) == 5:
        audio_paths.append(f'0000{f}.WAV')
    elif len(str(f)) == 6:
        audio_paths.append(f'000{f}.WAV')
    elif len(str(f)) == 7:
        audio_paths.append(f'00{f}.WAV')
    else:
        audio_paths.append(f'0{f}.WAV')

for f in os.listdir(path):
    if f not in audio_paths:
        full_file_path = os.path.join(path, f)
        os.remove(full_file_path)
```

# CLASS COMPOSITION




**Father**

**Worker**

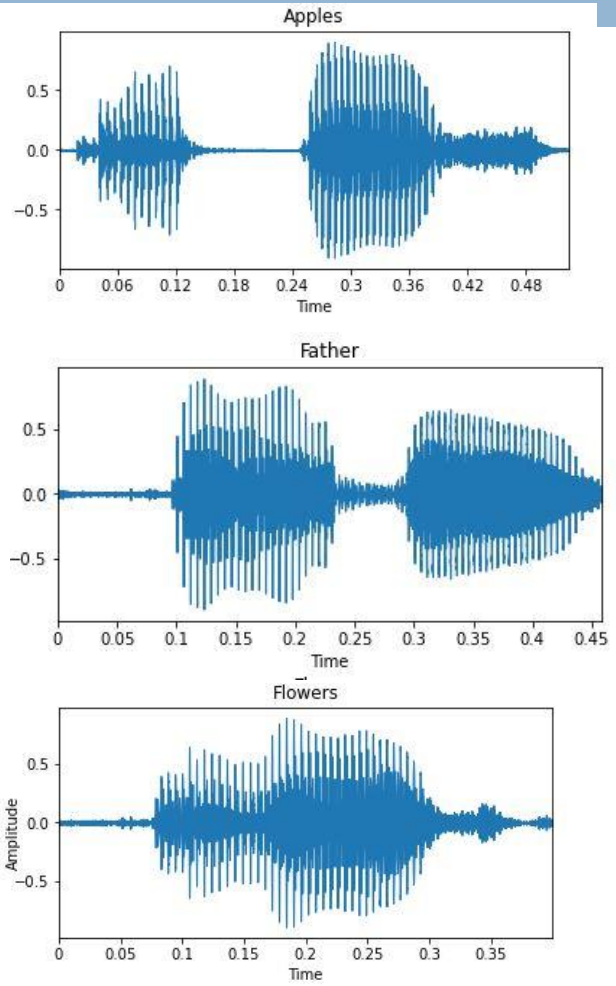
**Water**

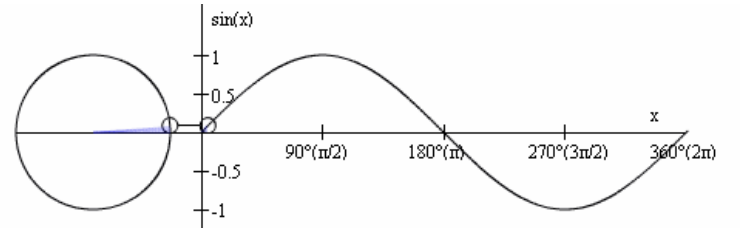
**Apples**

**Flowers**

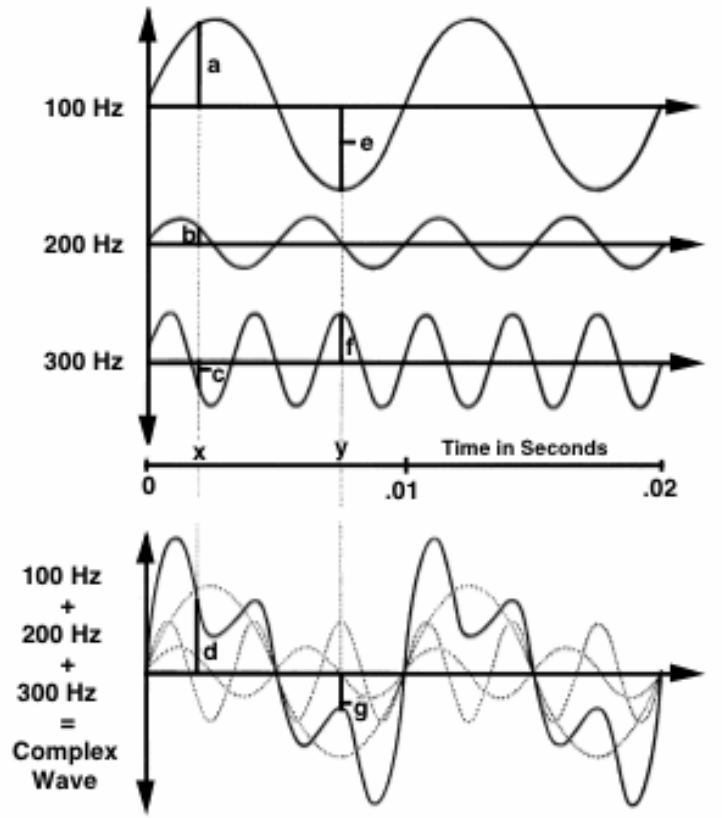
# DEMYSTIFYING AUDIO PROCESSING

# WHAT ACTUALLY ARE SOUNDWAVES?



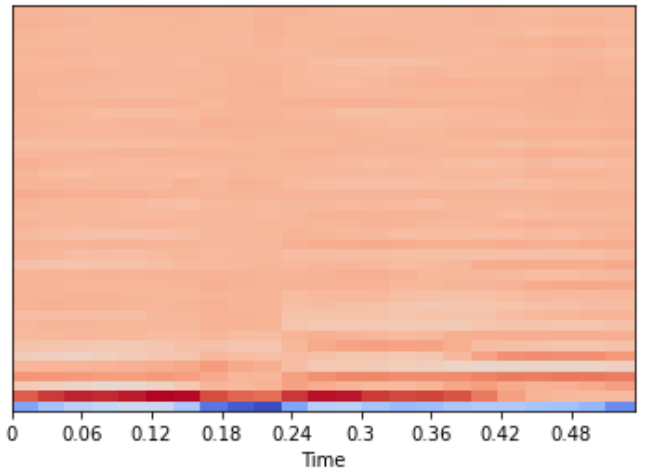



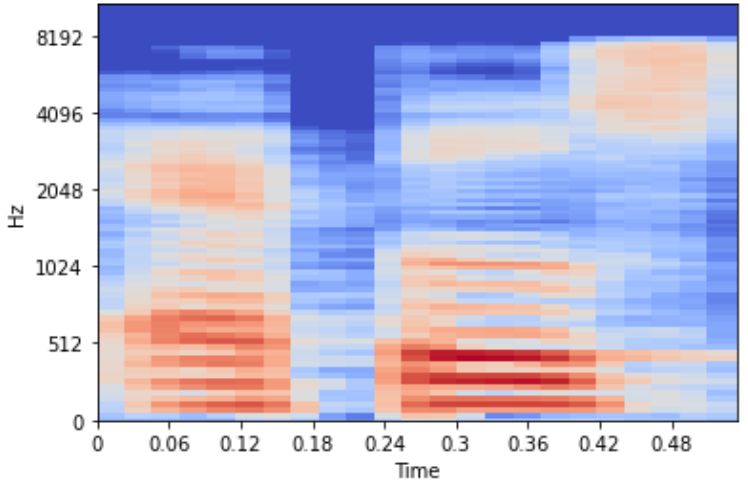

Audio are made up of many Sine waves!



# MAIN FEATURES IN AUDIO PROCESSING

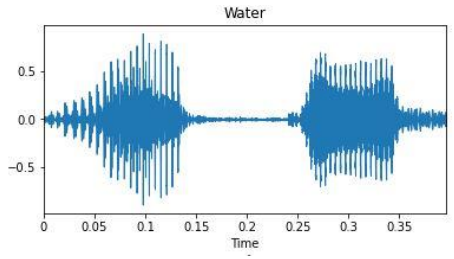
- Mel-Frequency Cepstral Coefficients (MFCCs)
  - The coefficients of a cepstrum
  - "Identity" of the audio file
  - Widely used in speech recognition due to its accuracy
- Mel-Spectrogram
  - Spectrogram – visual representation of amplitude of frequencies changes across time
  - Converted to Mel-scale (how human perceive audio vs frequency)
  - Think Heatmap!



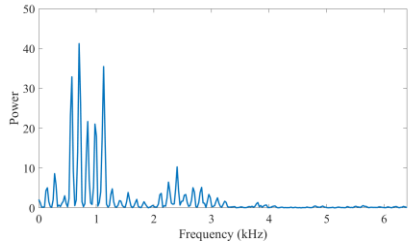
# HOW TO GET MFCC

**Audio Signal**



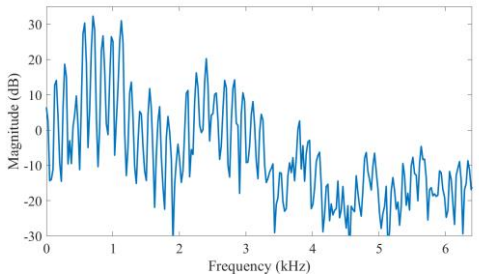

DFT

**Spectrum**



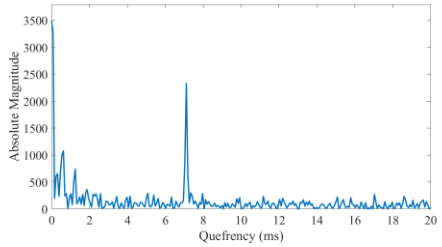

Log

**Log-scale Spectrum**



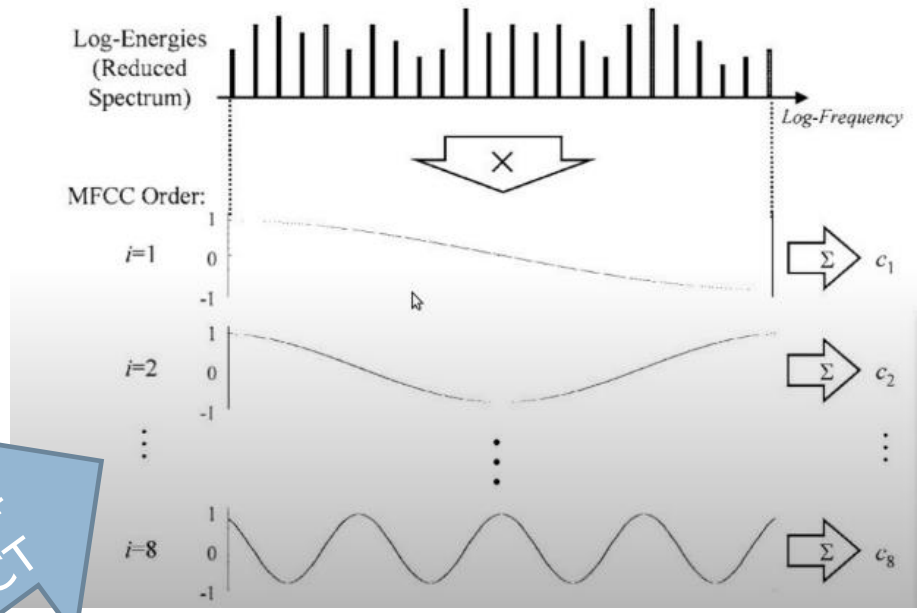

Inverse DFT

**Cepstrum**



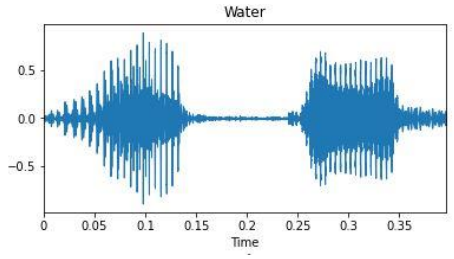

Mel + DCT

**MFCCs**



# HOW TO GET MEL-SPECTROGRAM

**Audio Signal**



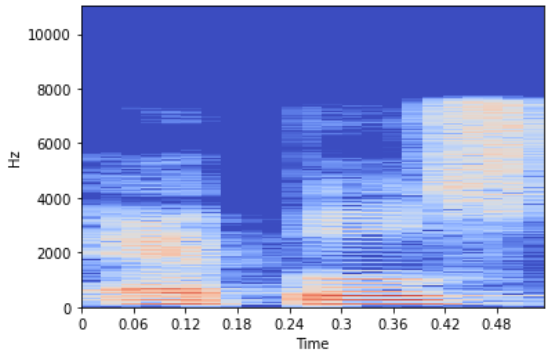

STFT

**Spectrum**

Complex Matrix

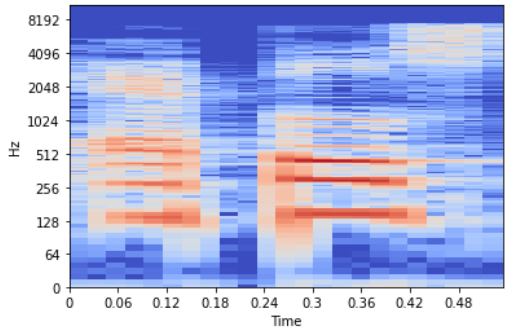
Square it

**Spectrogram**



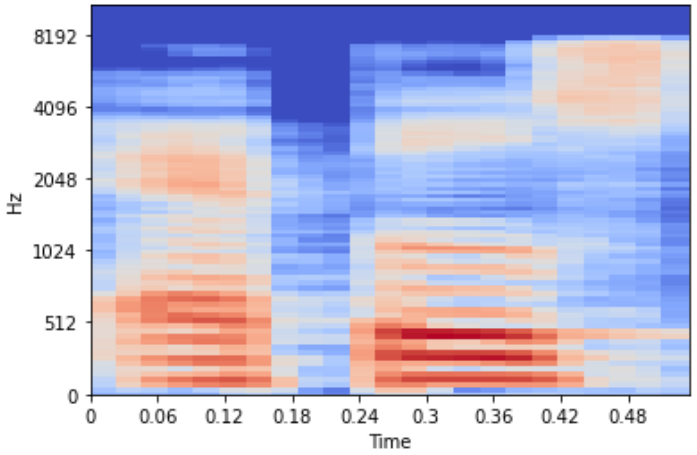

Log

**Log-Spectrogram**




Mel

**Mel-Spectrogram**



# INTRODUCING LIBROSA

## Extract MFCC

```
def get_mfcc_combined(file, n_mfcc=40):
    """
    extracts the mfcc, the delta and delta2 across timestep and return a transposed concatenation.
    """

    audio, sr = librosa.load(file)
    mfccs = librosa.feature.mfcc(y=audio, sr=sr, n_mfcc=n_mfcc)
    delta = librosa.feature.delta(mfccs)
    delta2 = librosa.feature.delta(mfccs, order = 2)
    combined = np.concatenate((mfccs, delta, delta2))
    return combined.T
```

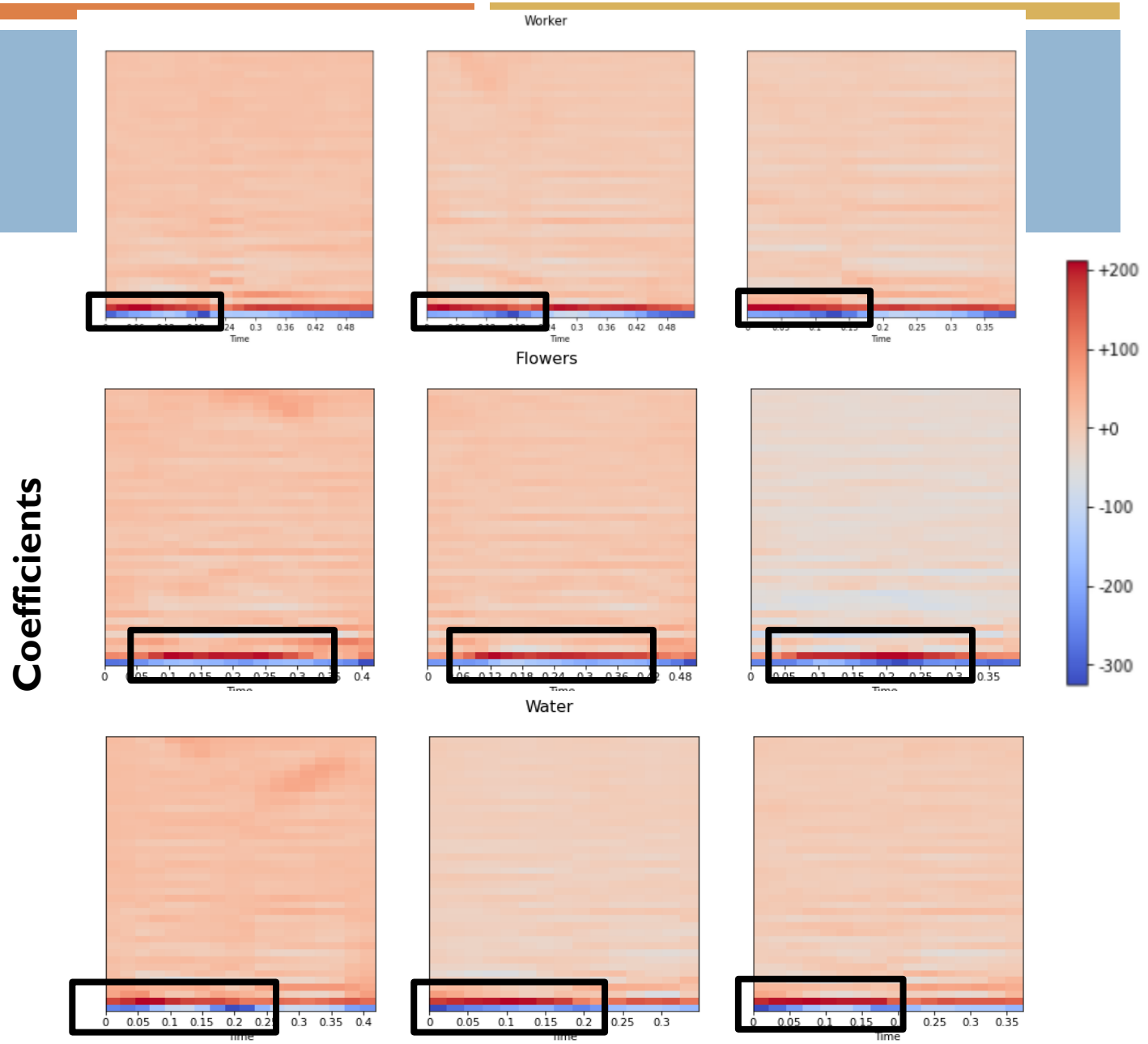
## Extract Mel-Spectrogram

```
audio, sr = librosa.load(f)
# 20 mel bands
mel_spectrogram = librosa.feature.melspectrogram(audio, sr = sr, n_mels=20)
# log scale
log_mel_spectrogram = librosa.power_to_db(mel_spectrogram)
# plot
librosa.display.specshow(log_mel_spectrogram,sr = sr, ax=ax[n],x_axis = 'time', y_axis = 'mel')
```

# EDA AND FINDINGS

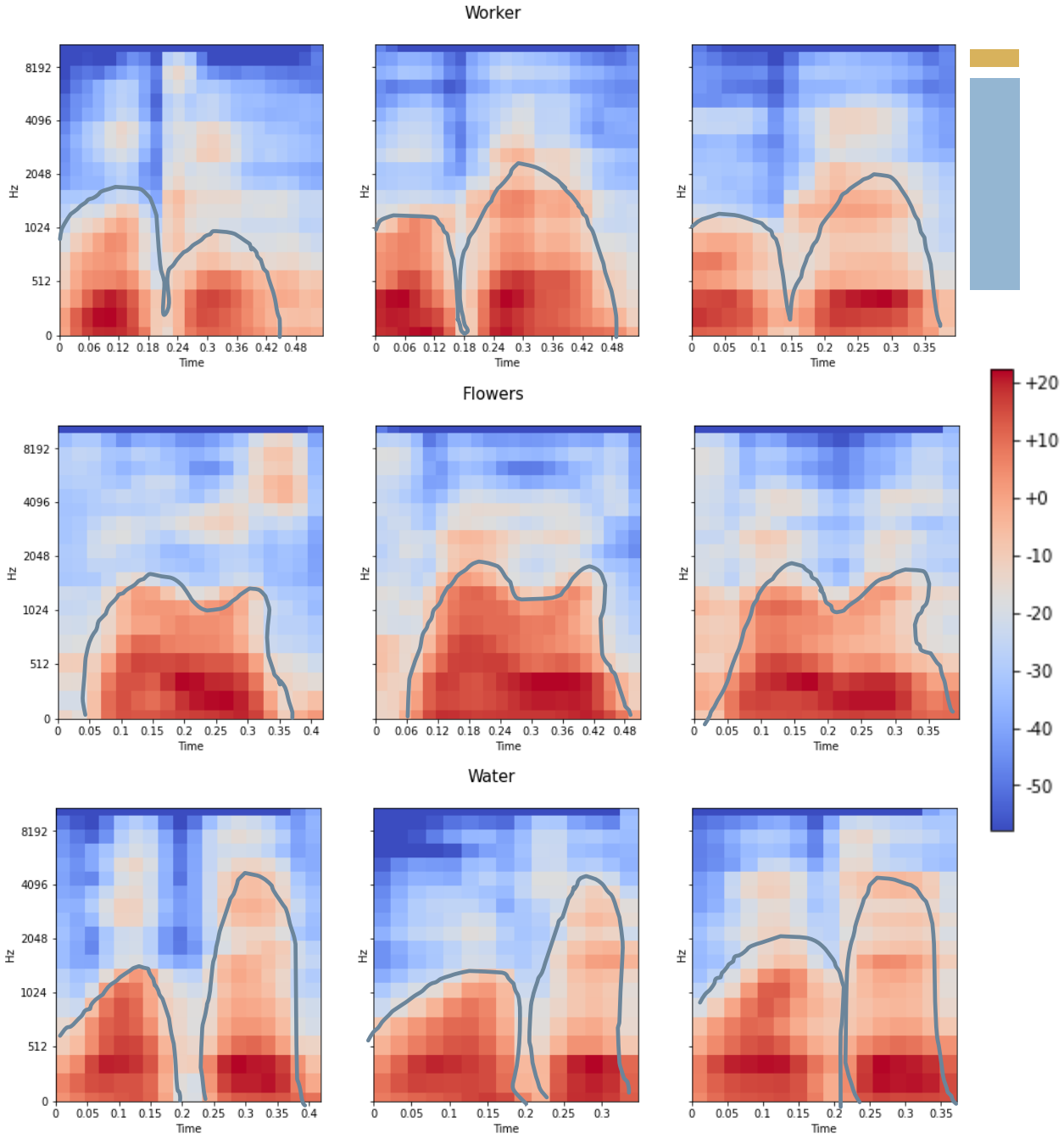
# MFCCS

- First few coefficients most significant
- Patterns among each class
- Some distinct patterns between classes
- CNN? Not ideal
- RNN? Most probable



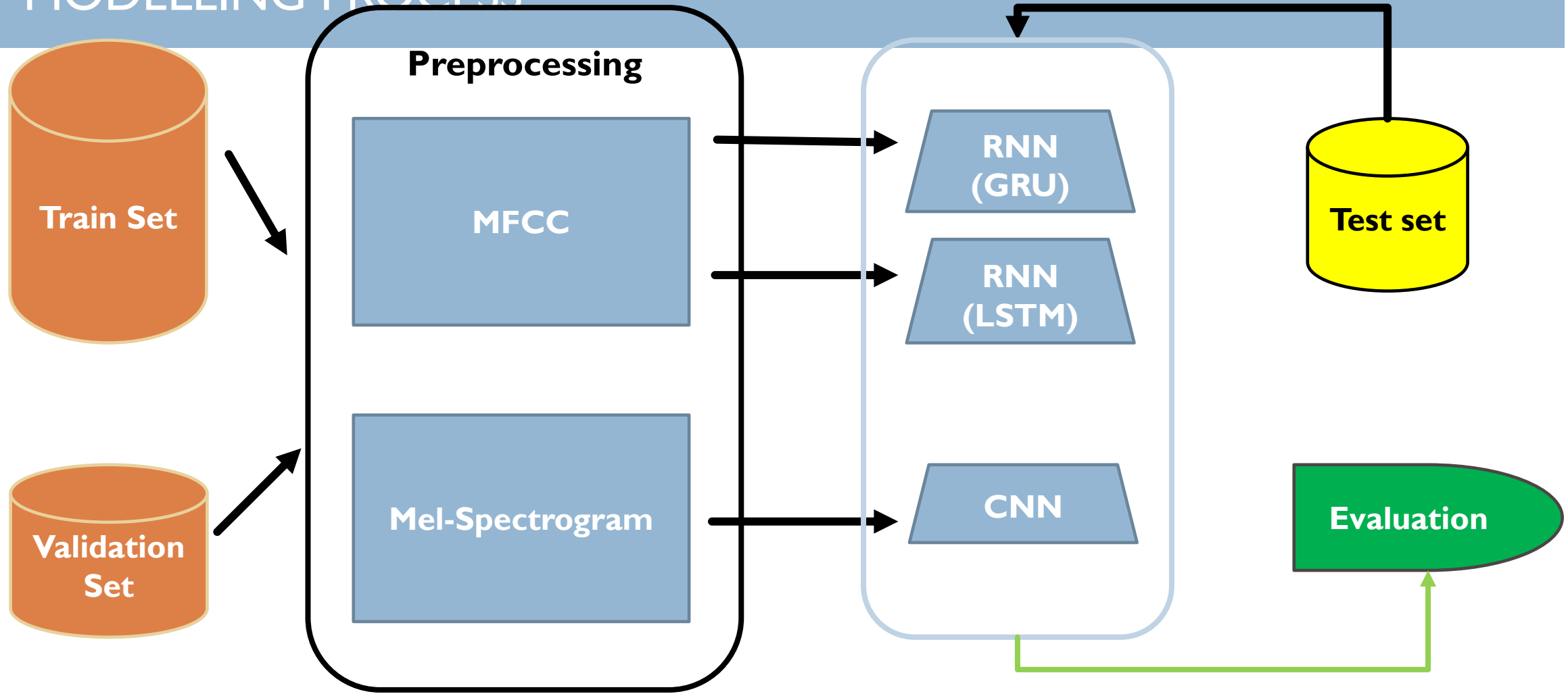
# MEL-SPECTROGRAM

- Features are distinct between classes
- Similarity among classes are also much more obvious
- CNN? Seems very possible!



# MODEL PERFORMANCE AND EVALUATION

# MODELLING PROCESS



# MODEL PERFORMANCE AND EVALUATION

| Model | Features | Val Accuracy | Test Accuracy | Training Time | Rank |
|---|---|---|---|---|---|
| CNN | Mel-Spectrograms Images | 99% | 96% | 6 mins | 2 |
| RNN (GRU) | MFCCs | 98% | 97% | 4 mins | 1 |
| RNN (LSTM) | MFCCs | 97% | 93% | 3 mins | 3 |
| FNN | MFCCs, Delta, Delta2 | 96% | | | |
| FNN | MFCC Mean, Std Dev, Delta Mean, Delta Std Dev | 94% | | | |
| CNN | Mel-Spectrogram Arrays | 90% | | | |
| FNN (Baseline) | MFCC Mean | 70% | | | |

# GREATEST TAKEAWAY

- Variable sequence length makes preprocessing difficult
  - Input shape (ts, coef)
- Luckily, there's padding and masking!

```
# apply function to all files
mfcc_combined = train_df['filepath'].map(get_mfcc_combined)

# padding
mfcc_combined = pad_sequences(mfcc_combined, padding='post')

# add into dataframe
train_df['mfcc_pad_combined'] = mfcc_combined.tolist()
```

```
model_7 = Sequential()

# add masking layer so that it would not use the zeroes
model_7.add(Masking(mask_value=0, input_shape=(42, 40)))

# two GRU layers
model_7.add(GRU(64, input_shape = (42,40), return_sequences=True))
model_7.add(Dropout(0.5))
model_7.add(GRU(32, return_sequences=False ))
model_7.add(Dropout(0.5))

# one dense layer
model_7.add(Dense(64, activation = 'relu'))
model_7.add(Dropout(0.5))

# output layer
model_7.add(Dense(5, activation = 'softmax'))
```

# CONCLUSION

- RNN is able to learn audio data very well
  - GRU might be better for shorter sequence
- Audio classification by CNN Image classification is also highly possible and accurate.
- MFCCs are strong identifiers for audio

Answering Problem Statement

- Machine learning algorithm can classify Singaporean-accented speech very well, at more than 90% accuracy on unseen data.
- RNN definitely has potential to be scaled up as a business solution due to its efficiency.
  - Voice command to interact with menus

# LAST BUT NOT LEAST…

Limitations

- MFCC and dataset not robust to noise
- Speakers in both train and test set are generally quite proficient speakers
  - Might score worse with speakers with lower proficiency

Suggestions for Further Research

- Other preprocessing techniques beside MFCC: Spectral Centroid
- How accuracy might be affected with less proficient English speakers;
- Processing speeches in noisy environment

# THE END

- Thank you! AMA!

# SOURCES


- https://hearinghealthmatters.org/wp-content/uploads/sites/9/files/2012/06/Fourier-Analysis.gif
- https://wiki.aalto.fi/display/ITSP/Cepstrum+and+MFCC
- https://unsplash.com/