Speech To Text
Deep Learning Overview

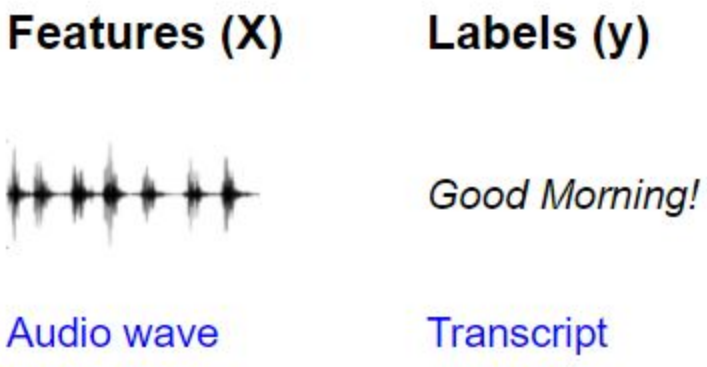
Sub topics

- Speech To Text using Deep learning model
- Understand the Speech To Text Deep learning model development using a simple Voice Recognition to Text

Speech To Text using Deep Learning Model

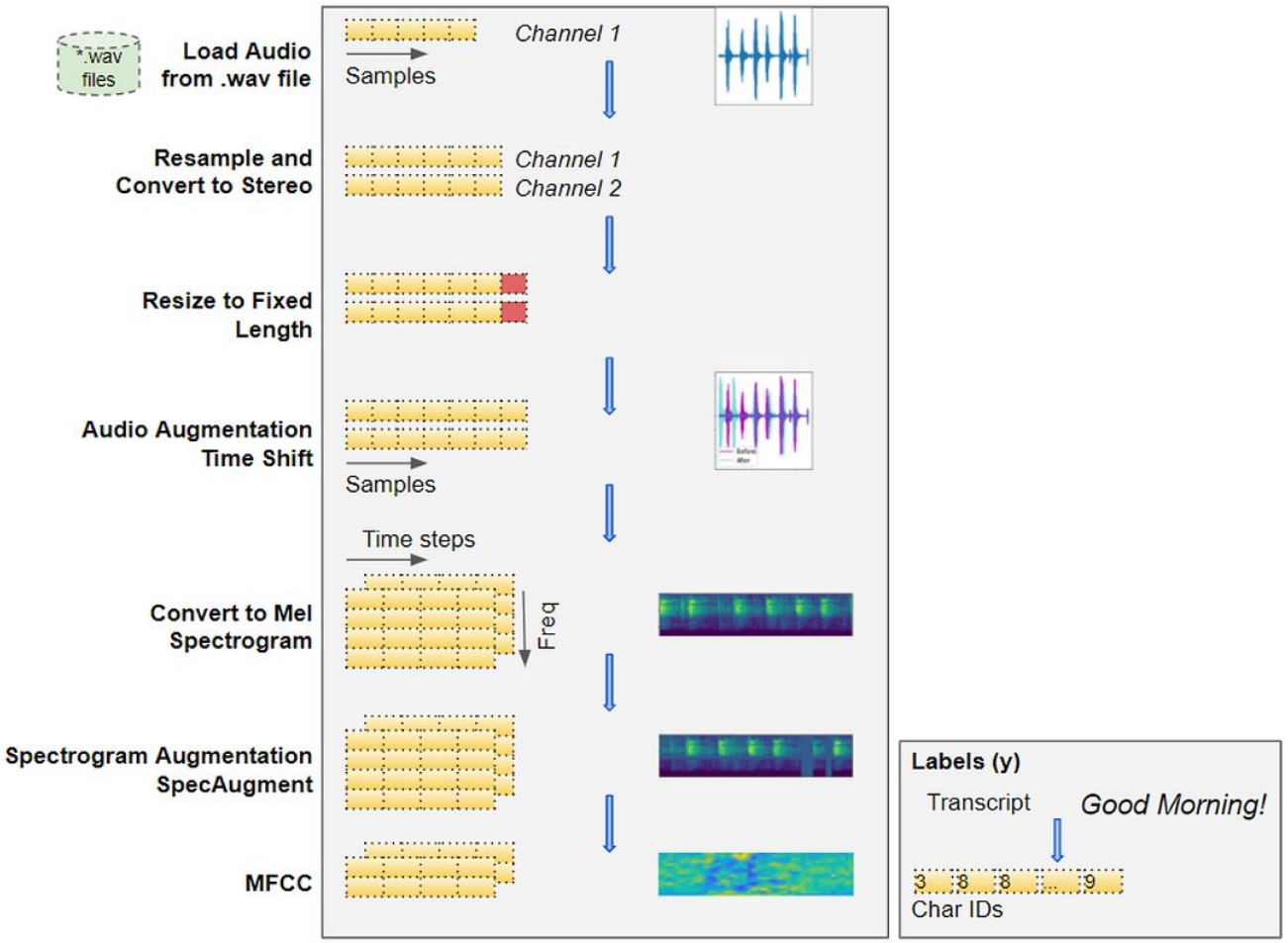
Data Set

- audio classification start with a sound clip and predict which class that sound belongs to, from a given set of classes. For Speech-to-Text problems, your training data consists of:
- Input features (*X*): audio clips of spoken words
- Target labels (*y*): a text transcript of what was spoken



Data Engineering

- The data transforms methods are used to process audio data for deep learning models. There are several Python libraries that provide the functionality to do this, with librosa being one of the most popular.



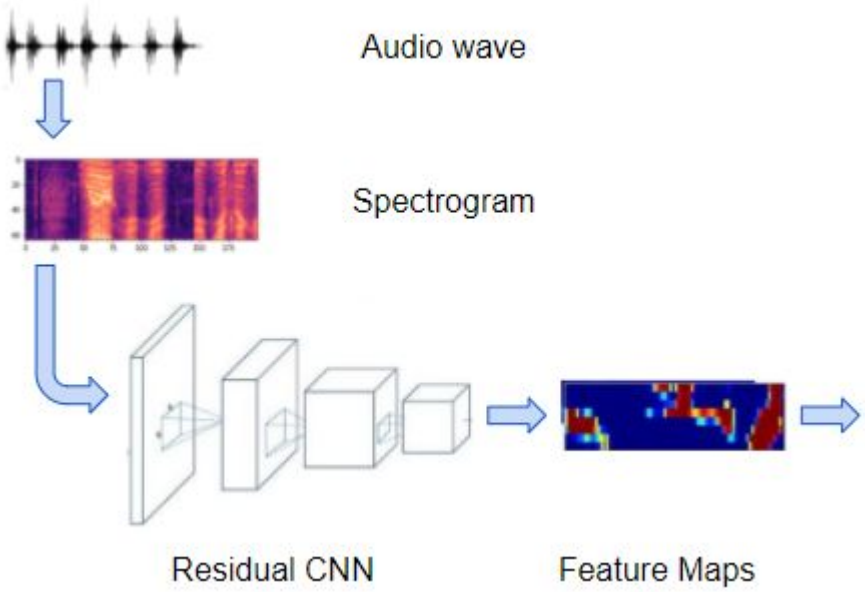
Deep Learning Model

There are many variations of deep learning architecture for ASR. Two commonly used approaches are:

- A CNN (Convolutional Neural Network) plus RNN-based (Recurrent Neural Network) architecture that uses the CTC Loss algorithm to demarcate each character of the words in the speech. eg. Baidu's Deep Speech model.
- An RNN-based sequence-to-sequence network that treats each 'slice' of the spectrogram as one element in a sequence eg. Google's Listen Attend Spell (LAS) model.

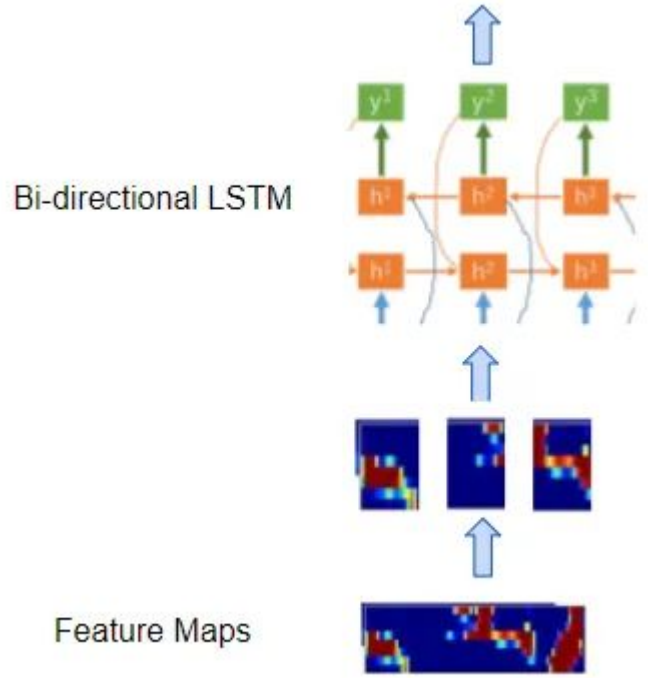
Deep Learning Model

A regular convolutional network consisting of a few Residual CNN layers that process the input spectrogram images and output feature maps of those images.



Deep Learning Model

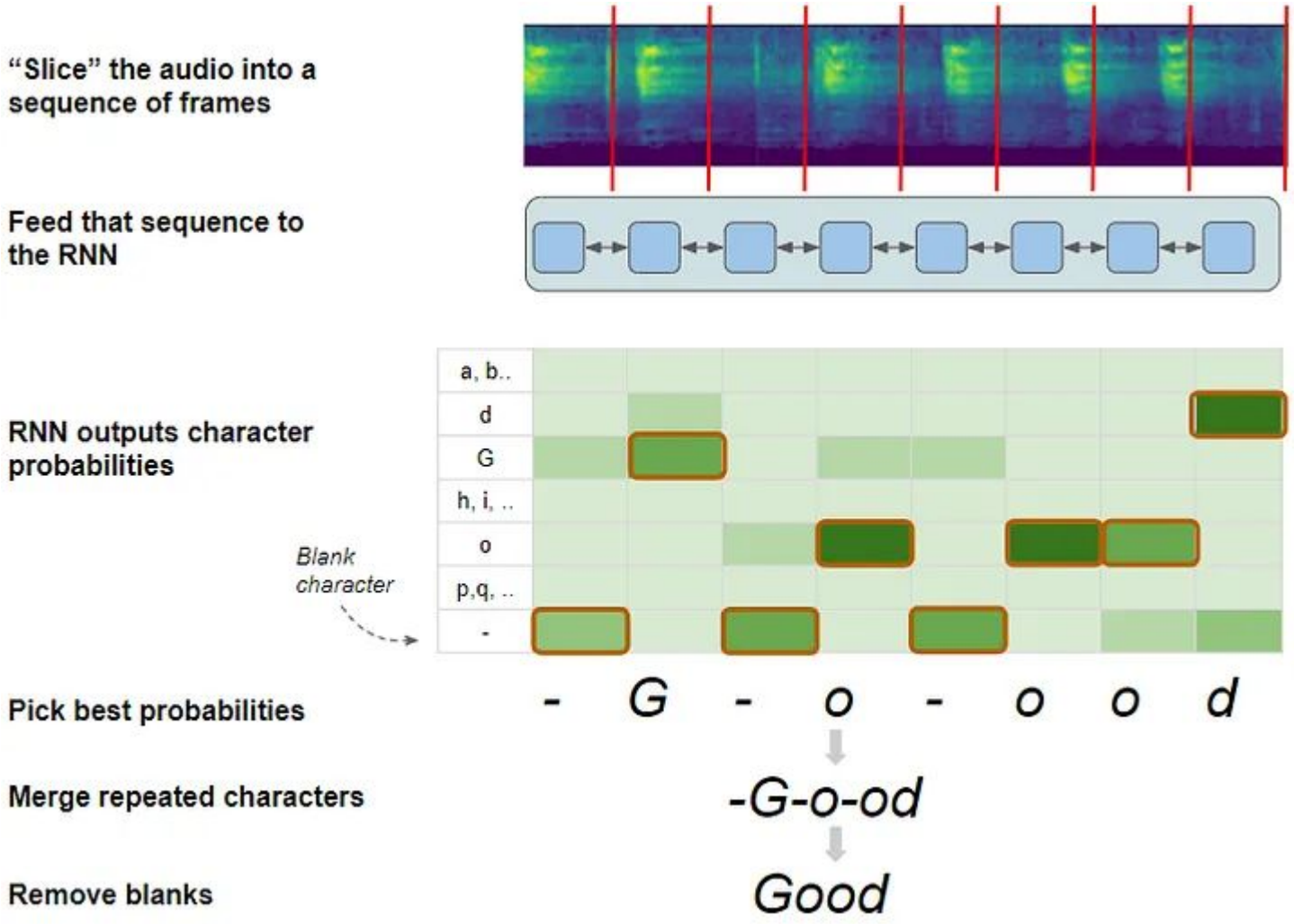
A regular recurrent network consisting of a few Bidirectional LSTM layers that process the feature maps as a series of distinct timesteps or ‘frames’ that correspond to our desired sequence of output characters. (An LSTM is a very commonly used type of recurrent layer, whose full form is Long Short Term Memory). In other words, it takes the feature maps which are a continuous representation of the audio, and converts them into a discrete representation.



Deep Learning Model

Decoding

Use the character probabilities to pick the most likely character for each frame, including blanks. eg. “-G-o-ood”



Data Engineering for Speech To Text Dataset

SIGNAL PROCESSING to Images

Audio Signal Processing

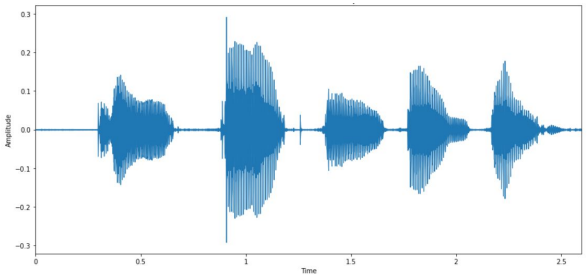
- Live Demo:
  - https://musiclab.chromeexperiments.com/Spectrogram/

- Try to use your Microphone and headset to “see” how the spectrogram looks like (in 3D) when you:
  - Whistle a tune
  - Say “YES” and “NO” repeatedly

- Can you “**see**” the difference / similarities of the YES/NO spectrograms?

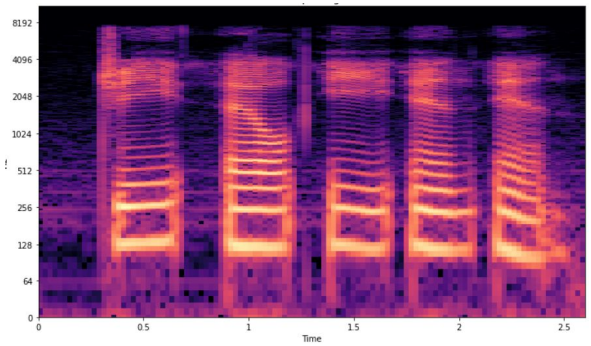
Audio Signal Processing

Waveform of a person spelling out “G L E N N”



Fourier Transform

Spectrogram of the waveform above



This axis here is the frequency

The colours represent the strength / amplitude of the frequency at a specific time

This axis is time

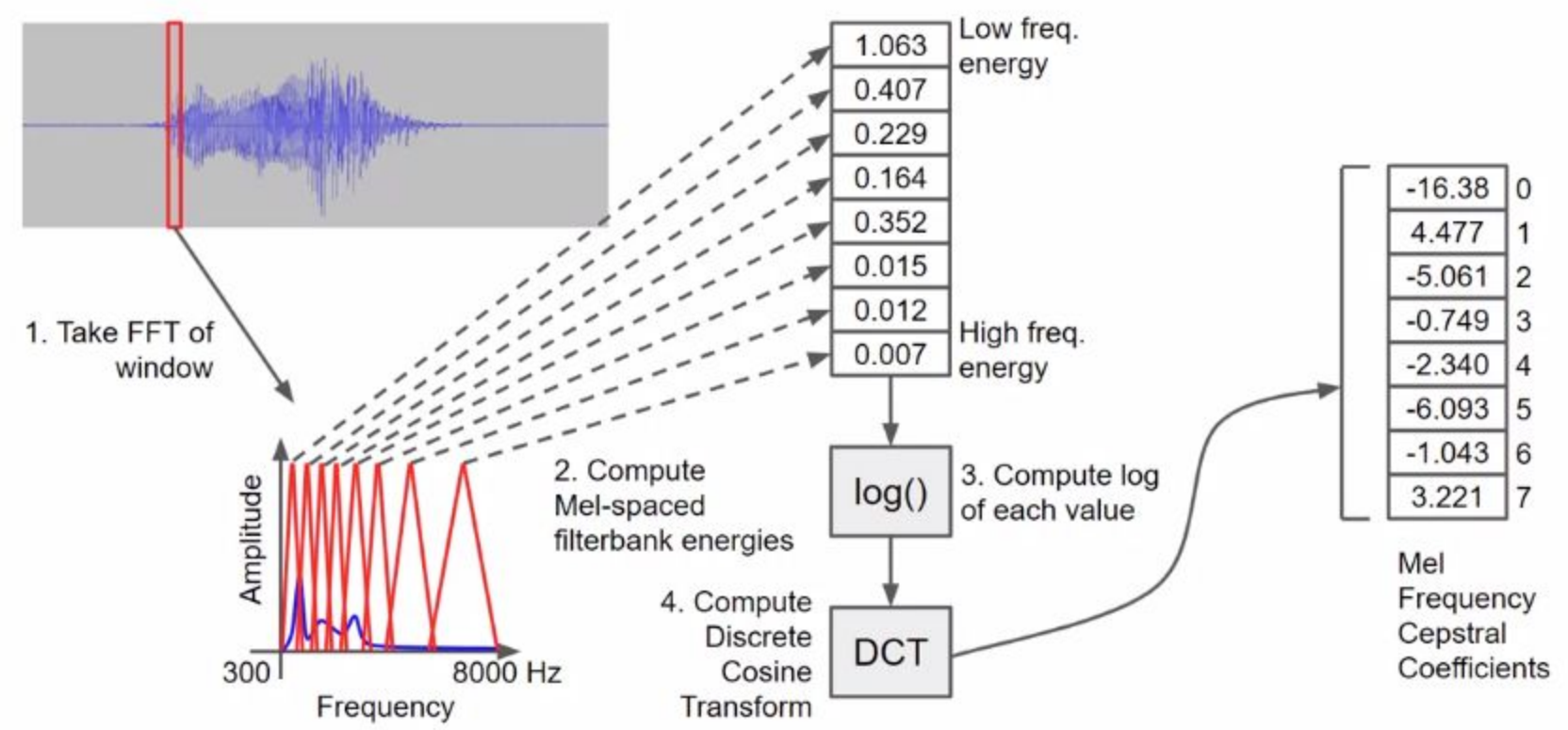
Audio Signal Processing

- **Mel-Frequency Cepstral Coefficients** is another well-researched signal processing algorithm that converts audio signals into another set of features.

- Defined in Wikipedia:
  - 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-frequency cepstral coefficients** (**MFCCs**) are coefficients that collectively make up an MFC.

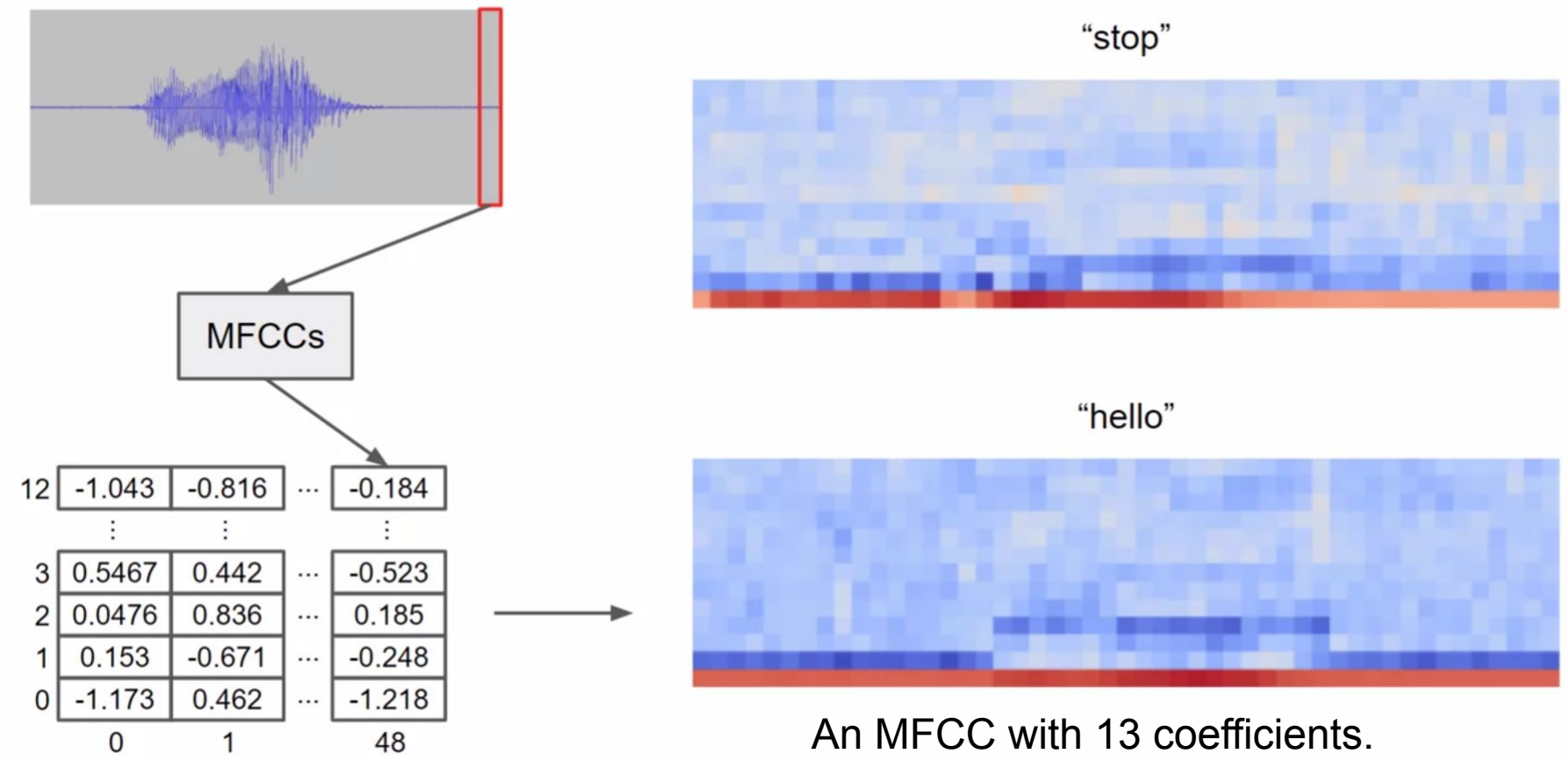
Audio Signal Processing

Mel Frequency Cepstral Coefficients



Audio Signal Processing

Mel Frequency Cepstral Coefficients

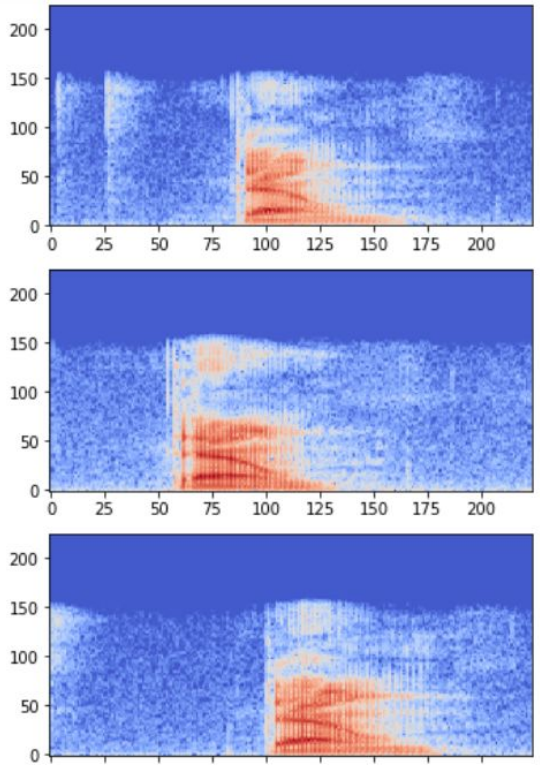


An MFCC with 13 coefficients.

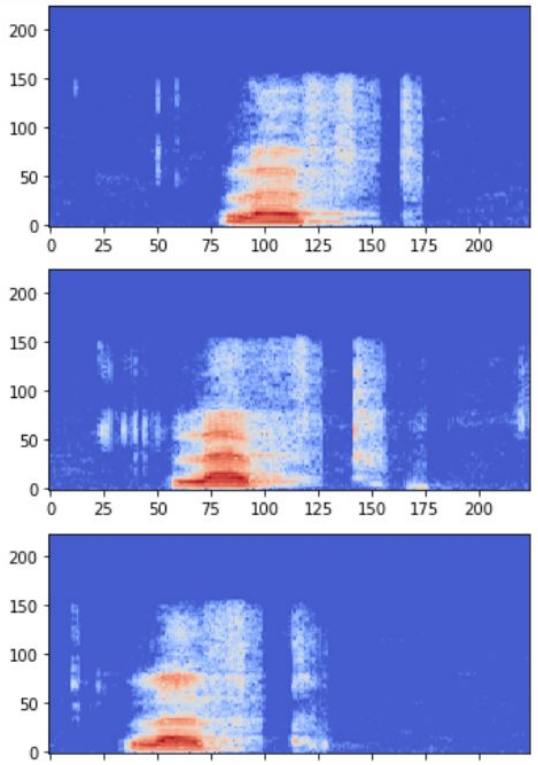
Using CNNs

- Why can CNNs help?

Spectrogram of the word “Down”

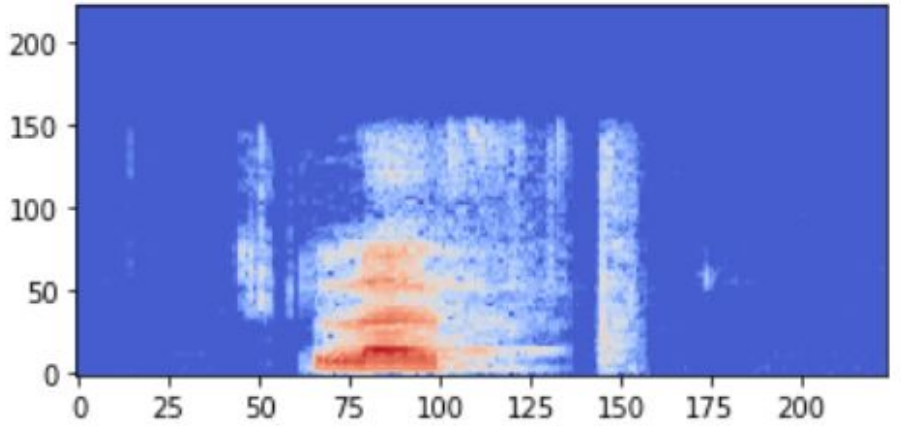


Spectrogram of the word “Left”



Does this look like a spectrogram of:

(a)“Down”
(b)“Left”?



Using CNNs

- Big idea for using CNN:
  - You are able to **see** the difference in the spectrograms (or MFCCs) produced from different words.
  - Therefore, we can theoretically treat spectrograms (or MFCCs) as if they were images, and use 2D CNNs to recognize them!

Using CNNs

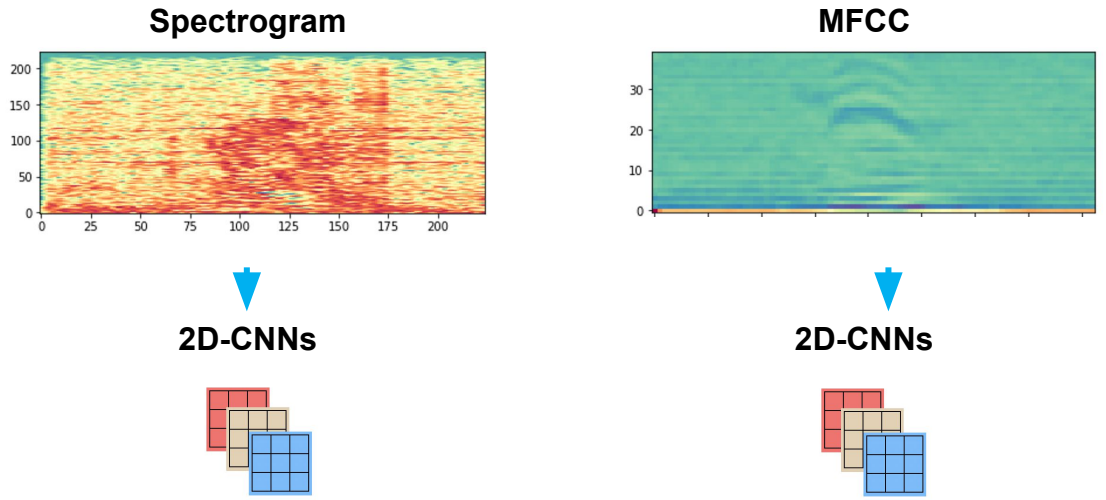
- Audio and Signals may be of unknown length

- Two solutions:
  - Train a CNN to accept a max size, if inputs are smaller => pad
    this is what we do in our practical
  - Stretch/Compress data to match input size
    this is what we do in our practical

How to Use CNNs on Audio Signals?

- Big idea:
  - Treat the spectrogram / MFCC output as an image with x (time), y (frequency) dimensions

- Train the CNN to recognize the image.



Non-Image Classification with CNN

EXERCISE

Step 1 – Preparing Our Dataset

- Use the Google Speech Commands Dataset for our exercise:
  - https://www.tensorflow.org/datasets/catalog/speech_commands
- Process all waveforms into 2D images of spectrograms
- Zip and upload our processed files into Google Drive

Step 2 – Creating Model / Hyper-Parameter Tuning

- We will create and train our CNN model on Colab.

- We will have to decide:
  - Conv2D layers, kernel size, strides, padding
  - Average/Max Pooling size
  - Dropouts
  - And how many of the above layers we need?
  - Flattening / Global Average/Max Pooling
  - Dense Layer

- Once we find a good model, we will save the model

Step 3 - Model Loading and Testing

- Keras allows you to save and load:
  - Model architecture (layers)
  - Model weights
  - Model architecture + weights together

- Injected HTML / IPyWebRTC:
  - Allows you to launch a recording button in Colab / Jupyter Notebook (Classic only) to record voices and save them into audio files.
  - We will load this audio file, convert to a spectrogram and send it to our Keras model for inference.