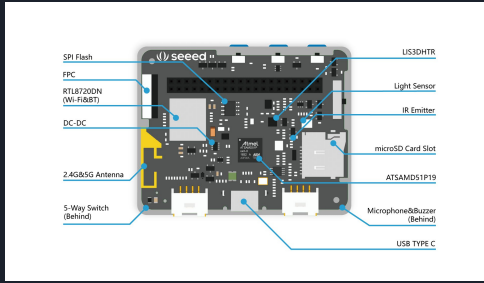
Foundations of Artificial Intelligence - TinyML Sound Processing/Speech to intent Project

Jordan Washburn

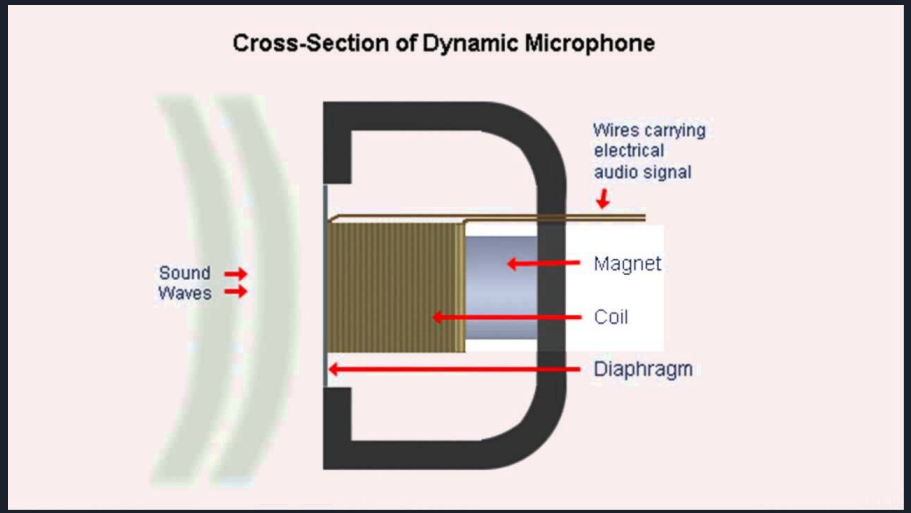
Speech to Intent on MCU Project Goals

- Limited support for FOSS that facilitates speech recognition tasks on microcontrollers
  - TensorFlow Lite and Arduino IDE allow one to train models and deploy them to MCUs
- Microcontrollers are relatively inexpensive and consume little energy
  - For example, I bought an Arm Cortex-M4 core for less than $20 and it operates on a power supply of 3.6 - 1.8 V (I will be using the WIO Terminal from Seeedstudio)
  - Low footprint - CPU efficient, small code size (fits limited flash memory)
  - Small memory (RAM) footprint
- While there are many use cases for speech recognition, this one focuses on speech to intent used in home automation or voice commands to control robots
- Works on embedded devices, not reliant on cloud like commercially available systems



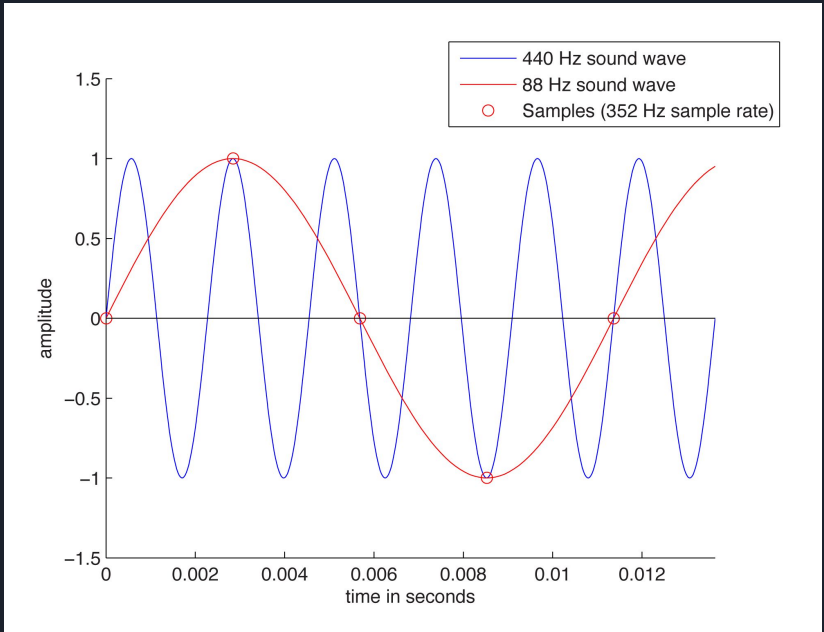
Lesson in Physics - How Does Sound Work?

- Sound is nothing more than a vibration that propagates through a transmission medium (gas, liquid, solid)
- One molecule “pushes” another molecule, which pushes another molecule and so on and so forth
- Once it reaches another object, that molecule then vibrates
- Microphones use a diaphragm that is pushed by sound waves, compresses, and then returns to its original state



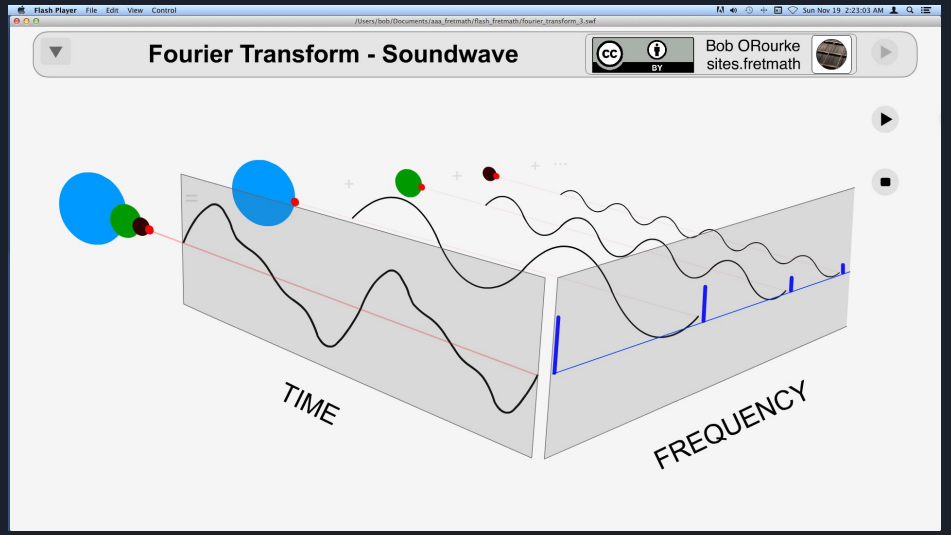
Lesson in Physics - How Does Sound Work? (cont.)

- We can record the movement from the microphone's diagram to convert it into an electrical signal (alternating current) and convert it from analog to digital
- The electrical signal is proportional to the amplitude of the sound wave i.e. the higher the amplitude of the sound wave, the greater the voltage of the electrical signal
- The number of times we want to record the signal is measured in a Sampling Rate (Hz) - the number of times a reading is taken in one second (Hz is defined as cycle/second.)
- We can visualize audio signal as a graph of amplitude vs. time



Lesson in Physics - How Does Sound Work? (cont.)

- We cannot do much with data represented in Amplitude vs. Time - here is where Fourier transforms come in.
- Fourier transforms are what facilitate decomposition of a signal into individual frequencies and frequency amplitude[1].

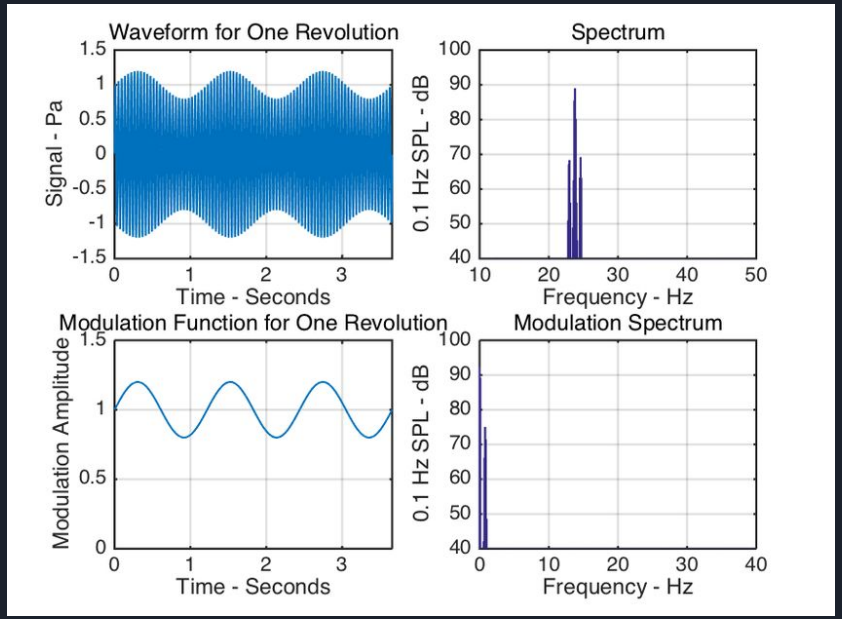


Source:https://youtu.be/ZvLcv7ekgUc?list=PLzTESaPWJUtYecRo8DaDPHI1EyVmnhTjF&t=170

[1] (Kulkarni et al., 2020)

Lesson in Physics - How Does Sound Work? (cont.)

- The higher bars correspond to higher frequencies. We can now do more with the signals.
- We can eliminate the least important frequencies, compress the audio, and remove noise[2].
- This, however, is still not good enough for speech recognition.

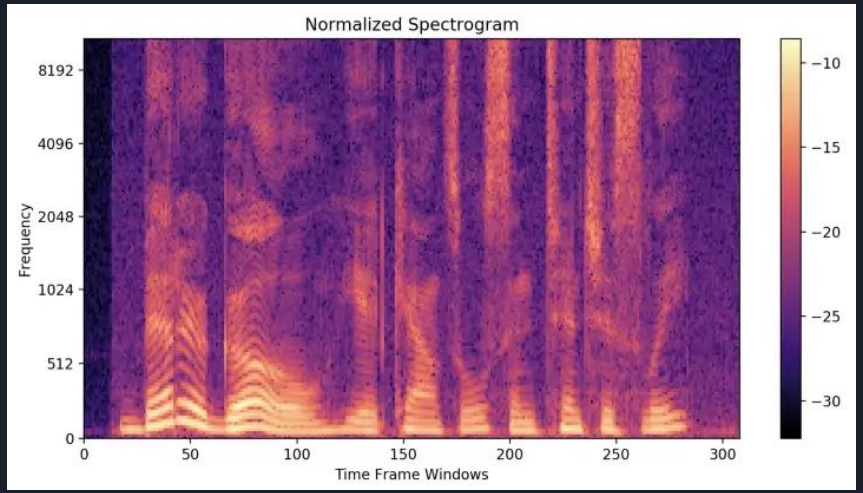


Source:https://www.researchgate.net/publication/326874266_A_Review_of_the_Possible_Perceptual_and_Physiological_Effects_of_Wind_Turbine_Noise

[2](Carlile et al., 2018)

Lesson in Physics - How Does Sound Work? (cont.)

- We have the frequencies, but where is the time information?
  - This is where spectrograms come in
- We can take the fourier transform multiple times in a time window and essentially append each measurement together[3]
- Then, we can visualize this in a Spectrogram - smaller frequencies (0-1kHz) are bright.

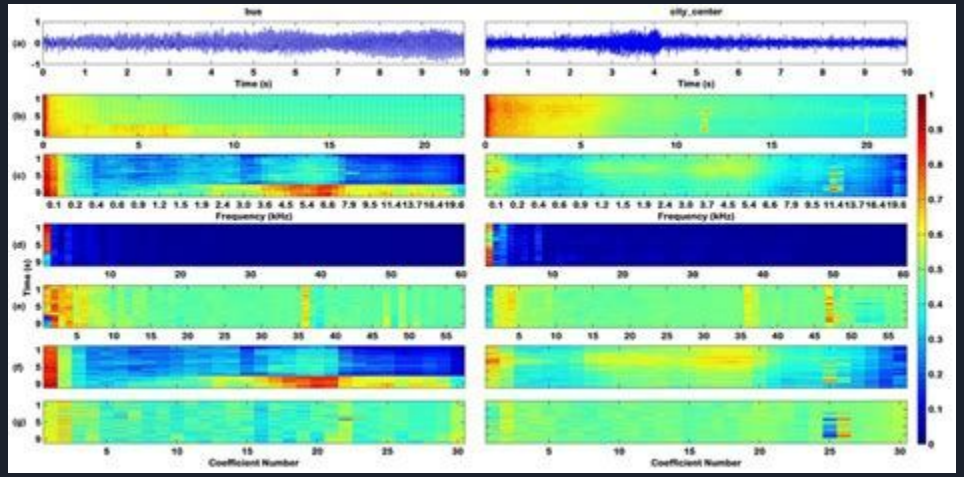


Source:https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

[3](Chaudhary, 2020)

Lesson in Physics - How Does Sound Work? (cont.)

- A normal Spectrogram still contains too much information in the range that humans can perceive and understand - human hearing range is about 20 Hz to 20,000 Hz[4]
    - Humans do not perceive frequencies in the linear scale - the difference between 500 Hz and 1000 Hz is easier to detect than the difference between 10,000 Hz and 10,500 Hz
- The Mel scale overcomes this limitation by specifying a unit of pitch so that equal distances in pitch sound equally distant to the listener. This places more emphasis on the range of frequencies that humans can hear.

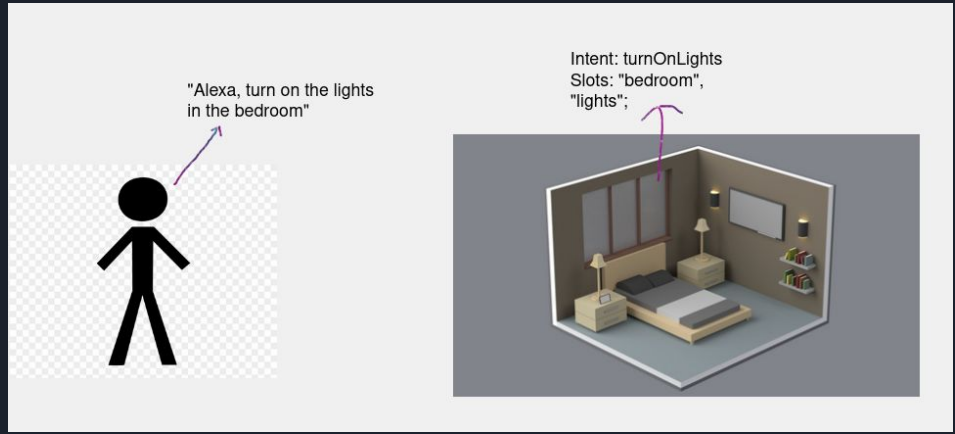


Source:https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520

[4](Perry, 2021)

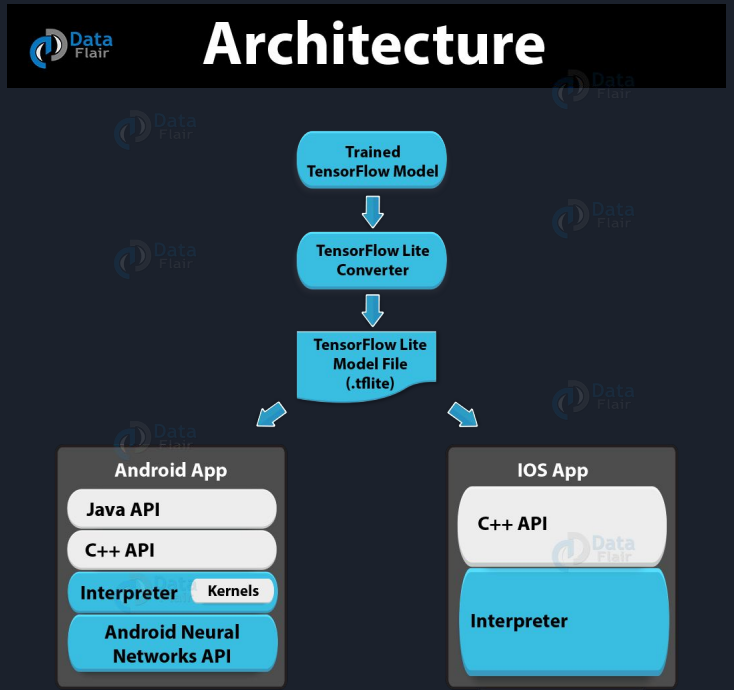
Speech to Intent - How does it work?

- In general, speech recognition is widely used in virtual assistants like Alexa and Siri
    - These cloud based systems use continuous speech recognition models and text based natural language parsers
    - Microcontrollers struggle with these models since they require large vocabularies/dictionaries and demand considerable amounts of computing power (MCUs limited by RAM and flash memory)
- To overcome these constraints, speech to intent can be directly converted to speech to parsed intent if based on a specific, predefined domain vocabulary.



Speech to Intent - Software

- My model was trained using Python, and specifically inside of a Jupyter Notebook.
- My training data consisted of TinyML's Fluent Speech Commands Dataset which contains 97 speakers, 248 unique phrases which are mapped to 31 different intents, that are divided into three slots: action, object, and location.



Code - Functionality Under the Hood

- Prominent imports used - pandas, numpy, librosa, matplotlib, tensorflow
- First, I needed to specify my file paths where I kept the training folder directories I carefully set up which contained the .csv files and .wav files

```
53 train_dataset_path = os.path.join(data_path, 'data/csv/train_data.csv'
54 valid_dataset_path = os.path.join(data_path, 'data/csv/valid_data.csv'
55 #test_dataset_path = os.path.join(data_path, 'data/csv/test_data.csv')
56 test_dataset_path = os.path.join(data_path, 'data/csv/wt_data.csv')
57
58 SAMPLING_RATE = 16000
59 MIN_FREQ = 100
60 MAX_FREQ = SAMPLING_RATE//2
61 WIN_SIZE_MS = 0.02
62 WIN_INCREASE_MS = 0.02
63 NUM_CEPSTRAL = 10
```

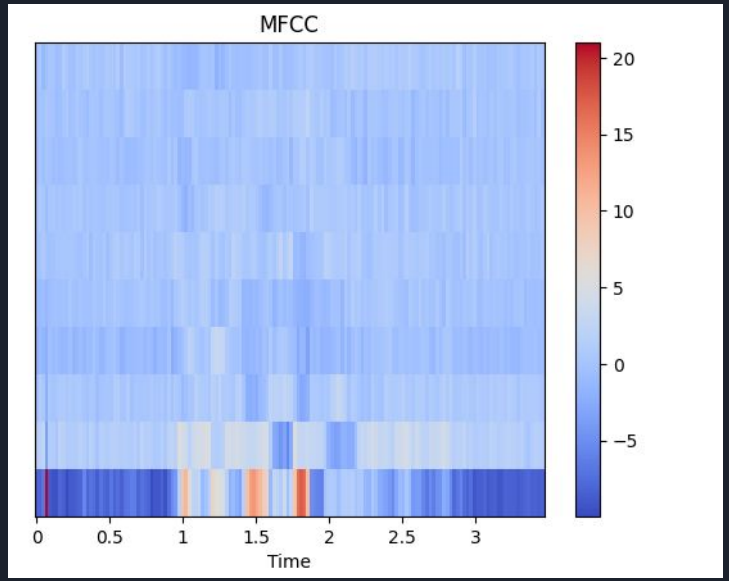
Code - Functionality Under the Hood (cont.)

```
In [4]: test_data = pd.read_csv(test_dataset_path)
        test_data.head()
```

Out[4]:

	Unnamed: 0	path	speakerId	transcription	action	object	location
0	0	wavs/wt_data/Bring_me_my_socks_325311.wav	NaN	Bring me my socks	bring	socks	none
1	1	wavs/wt_data/Bring_me_my_socks_872611.wav	NaN	Bring me my socks	bring	socks	none
2	2	wavs/wt_data/Bring_me_some_juice_147104.wav	NaN	Bring me some juice	bring	juice	none
3	3	wavs/wt_data/Bring_me_some_juice_184605.wav	NaN	Bring me some juice	bring	juice	none
4	4	wavs/wt_data/Bring_me_some_juice_336204.wav	NaN	Bring me some juice	bring	juice	none

- The files are then read and displayed
- Then, in order to match MFCC processing parameters on my device, audio_spectrogram and mfcc functions from TensorFlow are implemented.
- Next, generate_features creates a spectrogram and is then converted to mel frequency



Code - Functionality Under the Hood (cont.)

```
dict_values(["change language", "activate", "bring", "decrease", "deactivate", "increase"])
dict_values(["socks", "korean", "newspaper", "washroom", "english", "lights", "kitchen", "shoes", "heat", "juice", "g
erman", "bedroom", "music", "volume", "chinese", "lamp", "none"])
```

- The .csv files are then labeled, slots include words and locations and intents include only words for actions
- The training dataset consisted of audio that was very clean, had no background noise, etc.
    - To replicate real-world scenarios, AddGausianNoise and AddBackgroundNoise are implemented to a random number of samples

```
def create_aug_pipeline():

    aug_pipeline = Compose([
    AddGaussianNoise(min_amplitude=0.001, max_amplitude=0.015, p=0.1)
    AddBackgroundNoise(sounds_path=os.path.join(data_path, "data/wavs
    ClippingDistortion(p=0.3),
    PitchShift(min_semitones=-4, max_semitones=4, p=0.2),
    Shift(min_fraction=-0.5, max_fraction=0.5, p=0.1),
    Gain(p=0.2),
    TimeStretch(p=0.05)
    ])

```

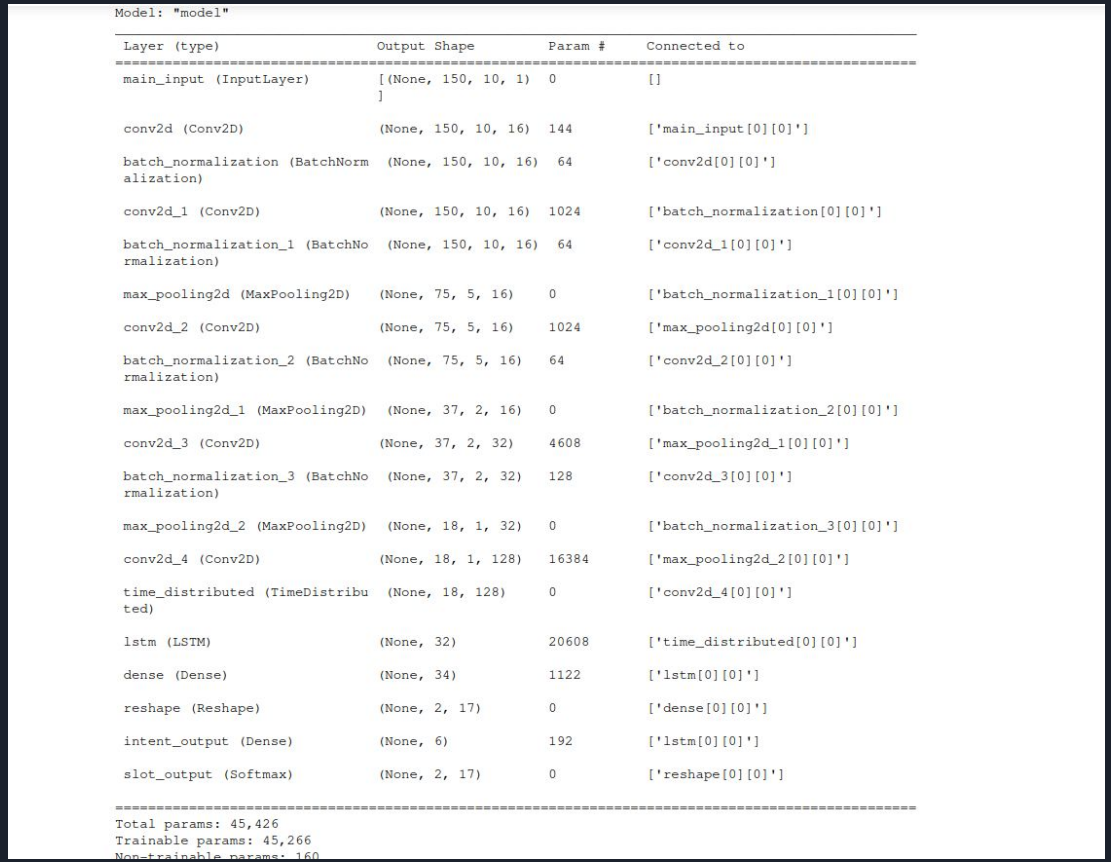
Code - Model Results/Accuracy

- Ultimately, the best model achieved an intent accuracy of 93.75%, with a slot accuracy of 96.87%
- There were multiple implementations and variations of the model and about 30 epochs were performed
- If using TensorFlow, it is best to have an NVIDIA GPU run the calculations. Otherwise, TensorFlow is limited to your CPU and it may take quite some time to train a model

```
Model ../checkpoints/2023-01-14_17-45-19/slu_model.h5

        Accuracy Intent 0.875 %
        Accuracy Slot 0.859375 %

        Random sample num:157

        Ground truth
        Intent:deactivate
        Slot1: lights  Slot2: washroom


        Prediction
        Intent:deactivate
        Slot1: lights  Slot2: bedroom


Model ../checkpoints/2023-01-14_18-32-15/slu_model.h5

        Accuracy Intent 0.96875 %
        Accuracy Slot 0.9375 %

        Random sample num:99

        Ground truth
        Intent:activate
        Slot1: lights  Slot2: washroom


        Prediction
        Intent:activate
        Slot1: lights  Slot2: washroom


----------------------------
Best model is ../checkpoints/2023-01-14_18-32-15/slu_model.h5 Accuracy 0.953125 %
----------------------------
```

Code - Model Results/Accuracy (cont.)

- The best performing model consisted of an LSTM layer with 2D Convolution layers, Batch Normalization, and Max Pooling 2D layers.
- Global Max Pooling is implemented and features are fed into a dense layer which then maps intent and slot outputs.



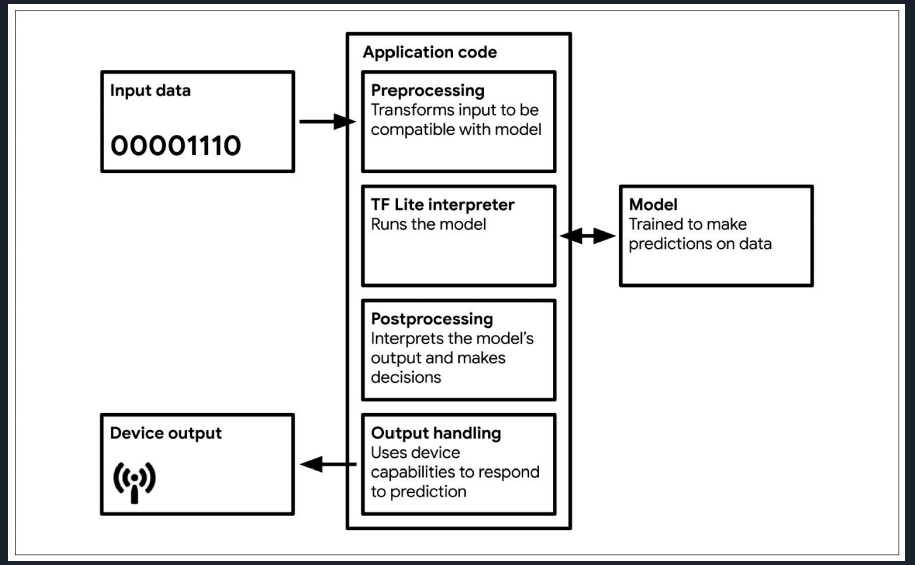
Model: "model"

Layer (type)	Output Shape	Param #	Connected to
main_input (InputLayer)	[(None, 150, 10, 1)]	0	[]
conv2d (Conv2D)	(None, 150, 10, 16)	144	['main_input[0][0]']
batch_normalization (BatchNormalization)	(None, 150, 10, 16)	64	['conv2d[0][0]']
conv2d_1 (Conv2D)	(None, 150, 10, 16)	1024	['batch_normalization[0][0]']
batch_normalization_1 (BatchNormalization)	(None, 150, 10, 16)	64	['conv2d_1[0][0]']
max_pooling2d (MaxPooling2D)	(None, 75, 5, 16)	0	['batch_normalization_1[0][0]']
conv2d_2 (Conv2D)	(None, 75, 5, 16)	1024	['max_pooling2d[0][0]']
batch_normalization_2 (BatchNormalization)	(None, 75, 5, 16)	64	['conv2d_2[0][0]']
max_pooling2d_1 (MaxPooling2D)	(None, 37, 2, 16)	0	['batch_normalization_2[0][0]']
conv2d_3 (Conv2D)	(None, 37, 2, 32)	4608	['max_pooling2d_1[0][0]']
batch_normalization_3 (BatchNormalization)	(None, 37, 2, 32)	128	['conv2d_3[0][0]']
max_pooling2d_2 (MaxPooling2D)	(None, 18, 1, 32)	0	['batch_normalization_3[0][0]']
conv2d_4 (Conv2D)	(None, 18, 1, 128)	16384	['max_pooling2d_2[0][0]']
time_distributed (TimeDistributed)	(None, 18, 128)	0	['conv2d_4[0][0]']
lstm (LSTM)	(None, 32)	20608	['time_distributed[0][0]']
dense (Dense)	(None, 34)	1122	['lstm[0][0]']
reshape (Reshape)	(None, 2, 17)	0	['dense[0][0]']
intent_output (Dense)	(None, 6)	192	['lstm[0][0]']
slot_output (Softmax)	(None, 2, 17)	0	['reshape[0][0]']

Total params: 45,426
Trainable params: 45,266
Non-trainable params: 160

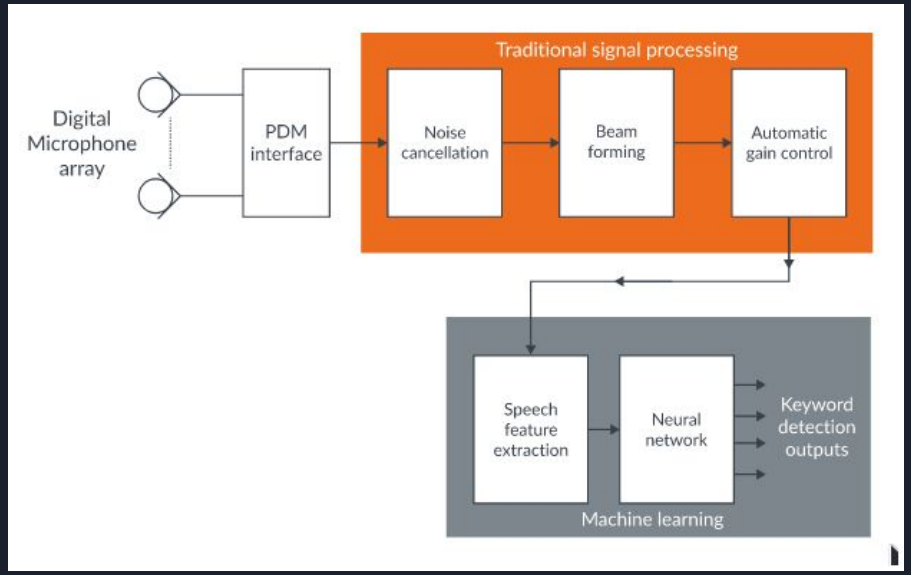
Hardware Integration - Arduino IDE

- Once the best model is found and an accuracy of at least in the 80s is achieved, there will be a .h file in the “checkpoints” folder that the code creates with a timestamp.
- The model.h file can be opened in Arduino IDE
    - I have found the best hardware with the best support from tensorflow and arduino IDE is the Arduino Nano 33 BLE Sense.
- In general, the overview of how TensoFlow integrates with hardware/microcontrollers can be found in the schematic.



Hardware Integration - Arduino IDE

- After the model is developed and the best model.h file is found, inference code can be generally implemented on MCUs in the following phases:
    - Audio acquisition
    - MFCC calculations
    - Inference on MFCC features
- Arduino IDE can import and compile TensorFlow Lite library
- In order to perform audio acquisition, the WIO Terminal has a built in microphone and Direct Memory Access (DMA) is used to pipe data from one location (internal memory, I2C, ADC, etc) to another.



Hardware Integration - Arduino IDE

- There are many similar projects that run on Arduino boards that have wide community support on GitHub which contain code in C/C++ that can be tweaked to run on any Cortex M4 core.
- Full credit - The MFCC calculations are performed by a project by Edge Impulse

Further Use Cases

- Wake word - A unique word can be used for low power modes using keyword spotting
- Robotics and Industrial IoT
- Wearables - smart watches
- Car Infotainment
- Voice remotes and TVs

Works Cited

- Carlile, Simon & Davy, John & Hillman, David & Burgemeister, Kym. (2018). A Review of the Possible Perceptual and Physiological Effects of Wind Turbine Noise. *Trends in Hearing*. 22. 233121651878955. 10.1177/2331216518789551. https://doi.org/10.1177/2331216518789551
- Chaudhary, K. (2020, March 10). *Understanding Audio data, Fourier Transform, FFT, Spectrogram and Speech Recognition*. Medium.https://towardsdatascience.com/understanding-audio-data-fourier-transform-fft-spectrogram-and-speech-recognition-a4072d228520
- Kulkarni, U., Kaushik, S., Lobo, L., & Sonkusare, R. (2020). Comparative Study of Digital Signal Processing Techniques for Tuning an Acoustic Guitar. *2020 7th International Conference on Smart Structures and Systems (ICSSS)*. https://doi.org/10.1109/icsss49621.2020.9202368
- Perry. (2021, October 2). *Human Hearing Range: Hearing Testing Frequency, Pitch, and What's Normal*. Audiology Research. https://www.audiologyresearch.org/human-hearing-range
- *TensorFlow Lite guide*. (n.d.). TensorFlow. https://www.tensorflow.org/lite/guide
- Waldekar, S., & Saha, G. (2020). Analysis and classification of acoustic scenes with wavelet transform-based mel-scaled features. *Multimedia Tools and Applications*, *79*(11-12), 7911–7926. https://doi.org/10.1007/s11042-019-08279-5