EEP 596: TinyML
Lecture 3: Keyword Spotting Using TinyML

Dept. of Electrical and Computer Engineering
University of Washington

Instructor: Dinuka Sahabandu (sdinuka@uw.edu)

ELECTRICAL & COMPUTER ENGINEERING
UNIVERSITY of WASHINGTON

NSL

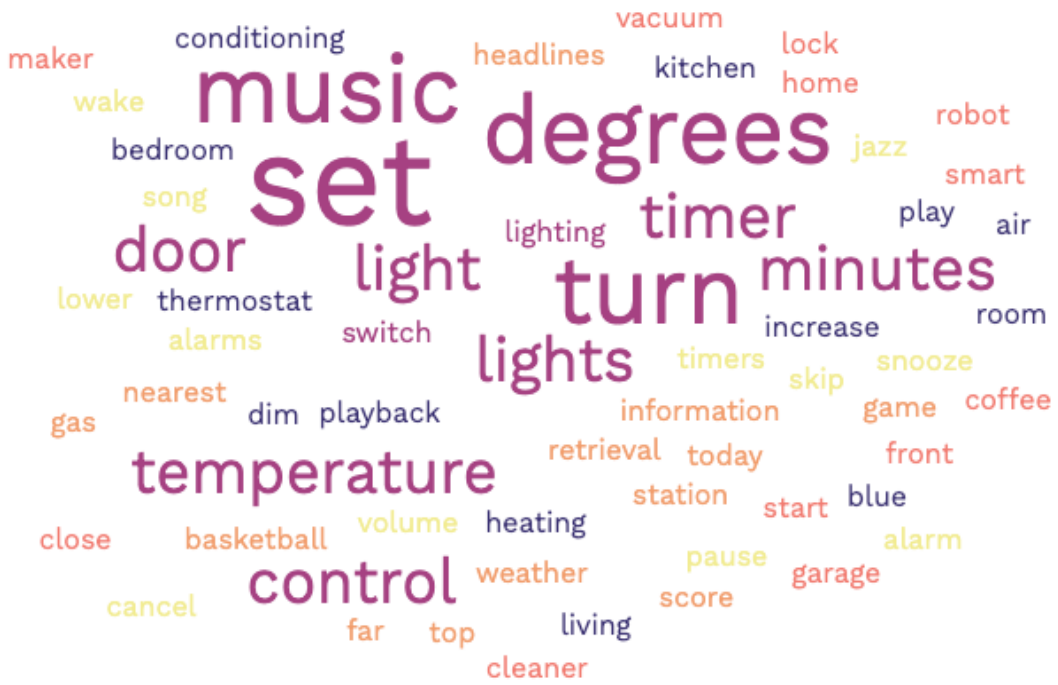
Topics Covered

- Introduction to Key Word Spotting (KWS)
- Applications and Challenges of KWS
- Anatomy and Architectures of KWS Applications
- Analog to Digital Audio Conversion and Representations (FFT, Spectrogram, MFCC)
- KWS Datasets and Data Pre-processing for KWS
- ML model for KWS
- Security Issues in KWS - Dolphin Attack
- Lab 3: Processing Audio Data and Training ML Models for KWS

❖Relevant reading from the Textbook [Warden and Situnayake; O'REILLY Publisher] is Chapter 7

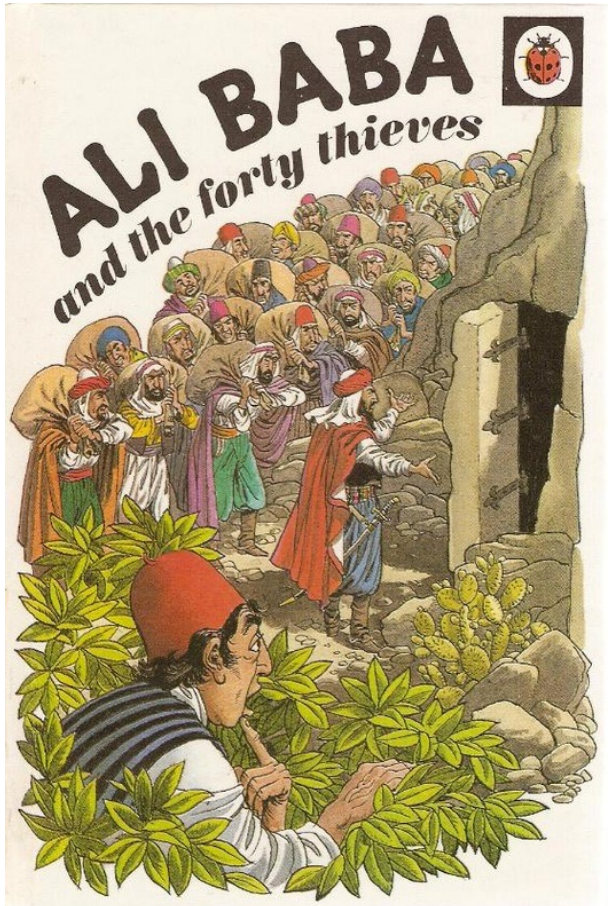
What is Key Word Spotting (KWS)?

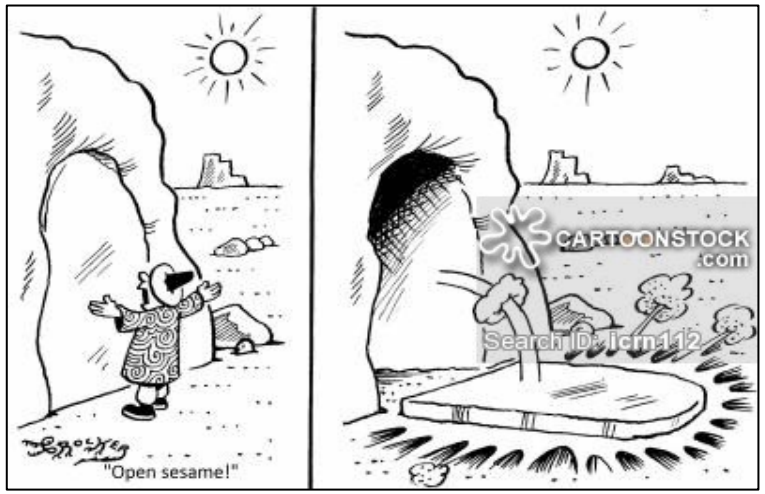
- Identify and extract **specific words or phrases** from spoken audio, then using them to trigger specific actions or responses
- Low-power, continuous, on-device mode
- The keywords are “region of interest”
- We can have region of interest in images and audio as well

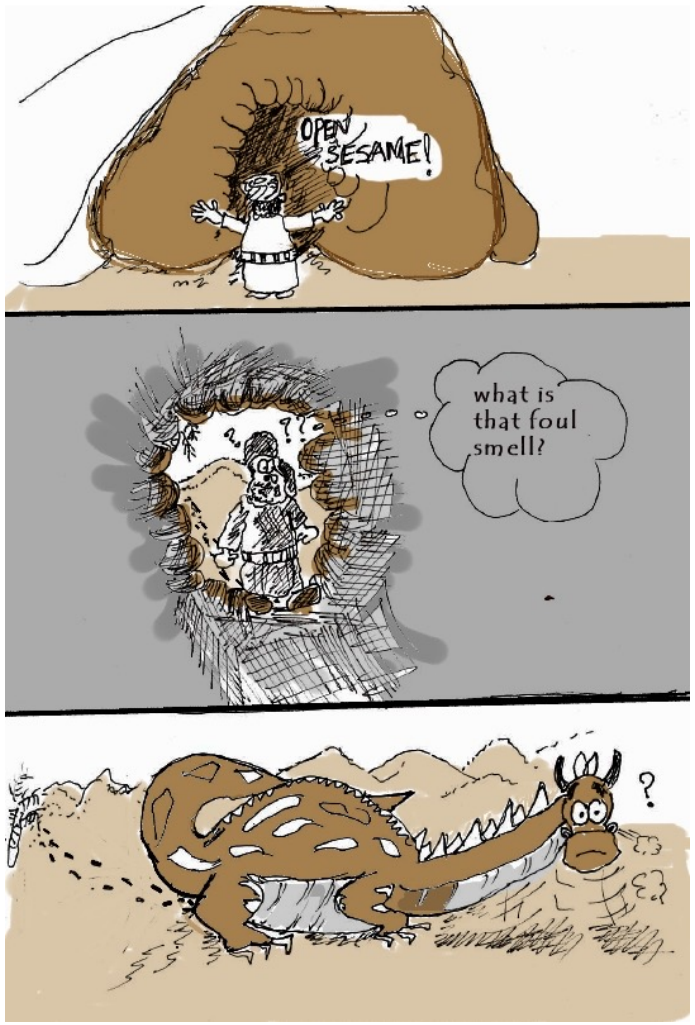


KWS is as Old as Time

What is the oldest example of KWS?

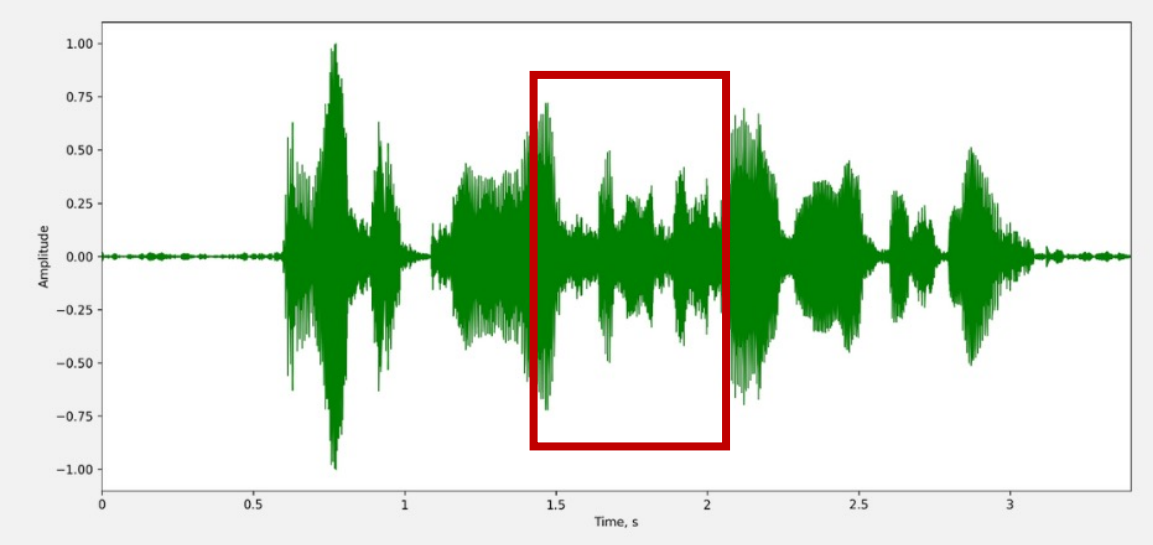


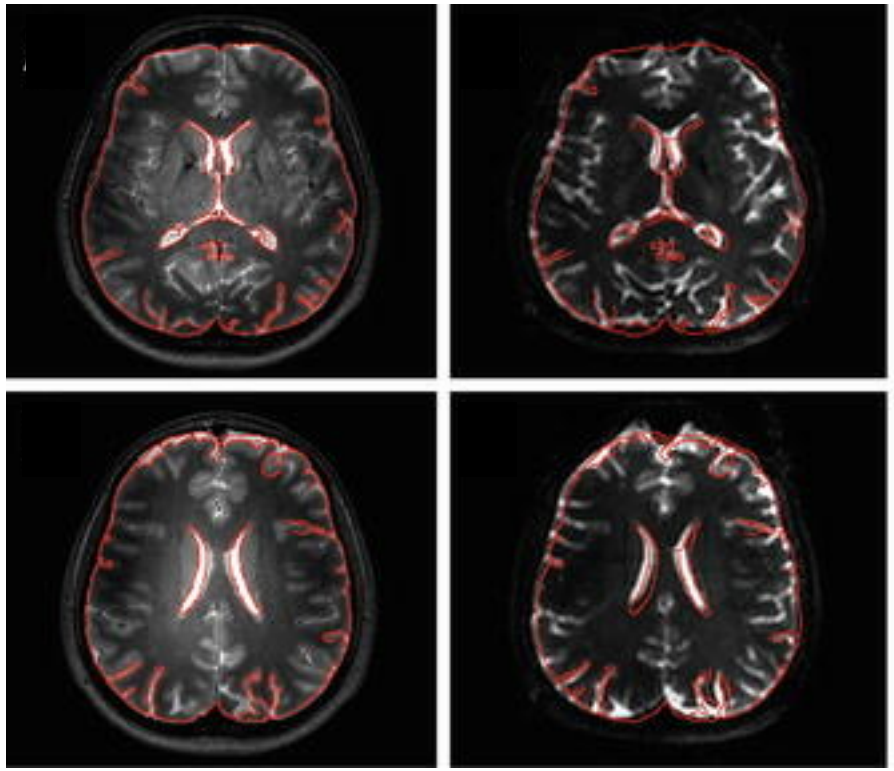




Region of Interest (ROI) in Images and Audios

ROI





Medical Analysis ROI

KWS

What is a salient property of all these devices that is related to KWS?

Why do we care to look at this problem?

Can we use standard language models for KWS recognition?

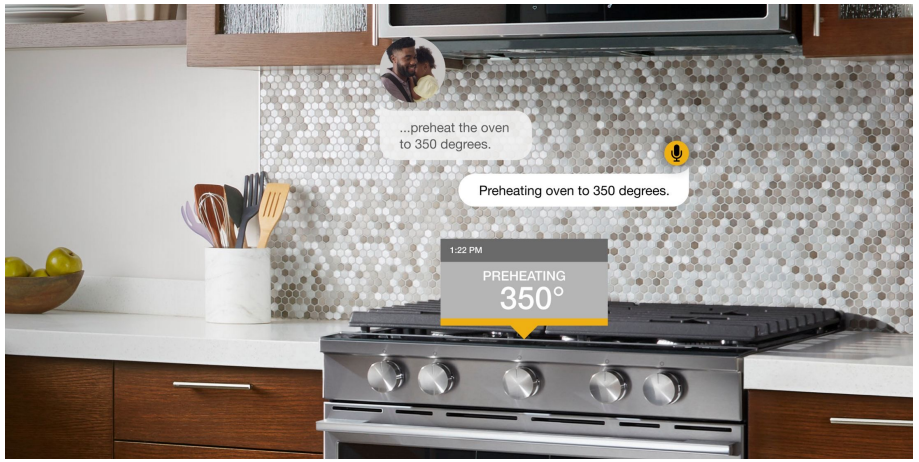
KWS vs. Speech Recognition

- **Key Word Spotting** vs. **General Speech Recognition**

	Key Word Spotting	General Speech Recognition
Focus	Predetermined Keywords	Entire spoken language
Speed	Fast, operate in real-time	Slower than KWS
Resource Assumption	Less resource-intensive, suitable for systems with limited power	Require more computational power and memory
Use Cases	Voice assistants, home automation systems, (specific voice command)	transcription services, real-time translation services (deep understanding of spoken language)

Applications of KWS

Voice Assistant



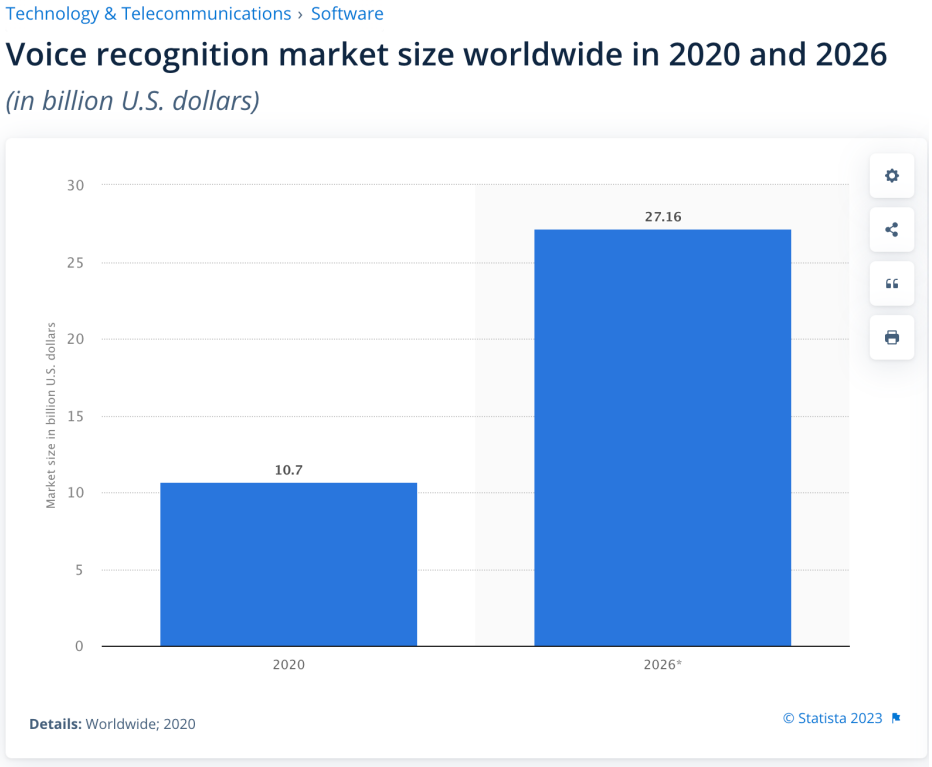
Smart kitchen

Voice Control in Cars

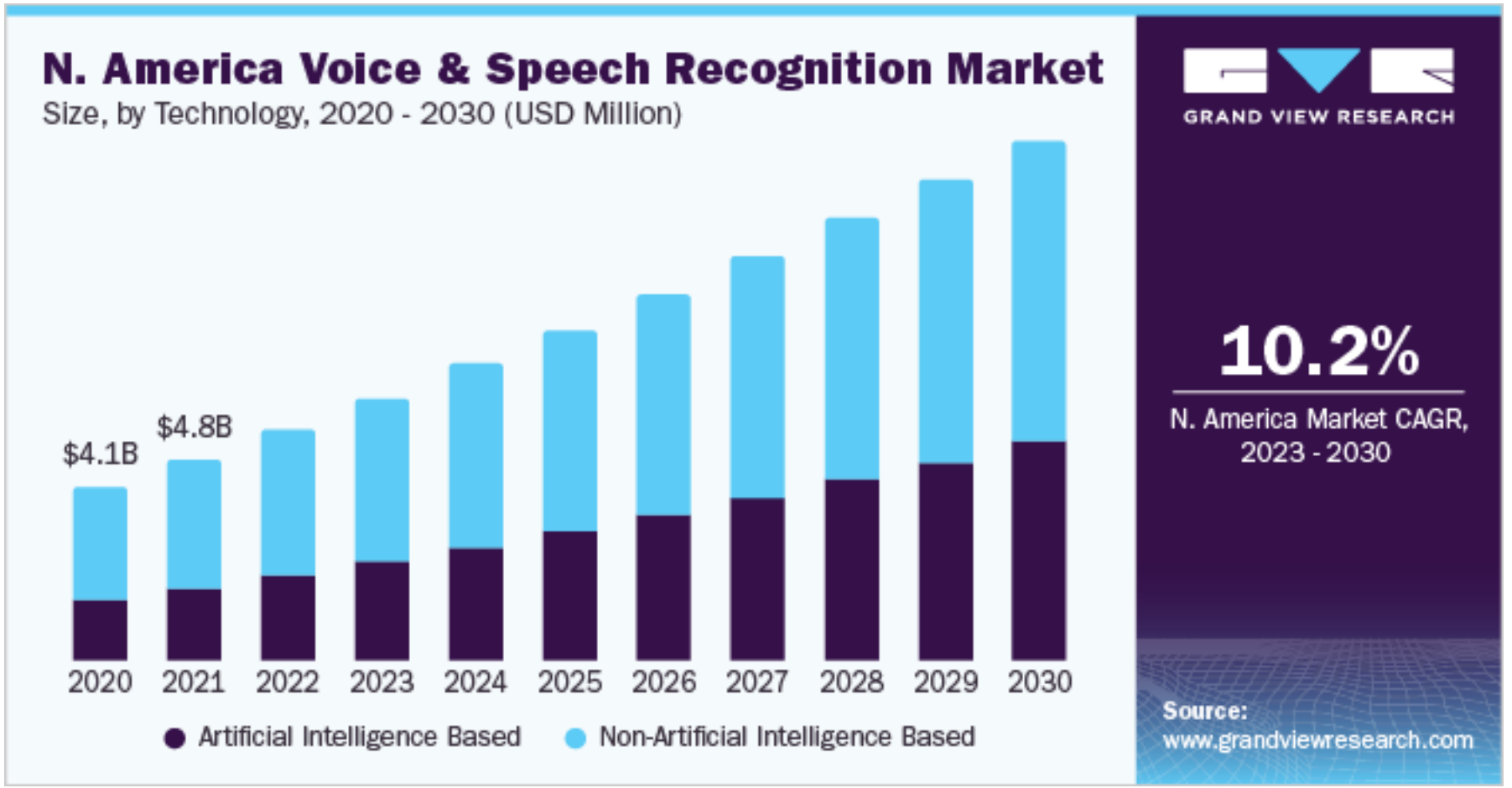
Smart glasses

Applications of KWS

- 62% of adults in the U.S. have a voice assistant on some device.
- Each month more than 1 Billion voice search is carried out.
- Amazon, Google and Apple have added the words “okay, Google,” “Alexa” and “Hey, Siri” to the global lexicon.
- The market size is projected to be about $9Billion in 2023 and to grow to be closer to $20Billion in 2030.



Applications of KWS



Challenges with KWS

Jim

Jim's Alexa

Challenges with KWS

Jim

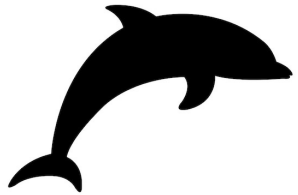
Jim's Alexa

Challenges with KWS

Jim

Jim's Alexa

Challenges with KWS



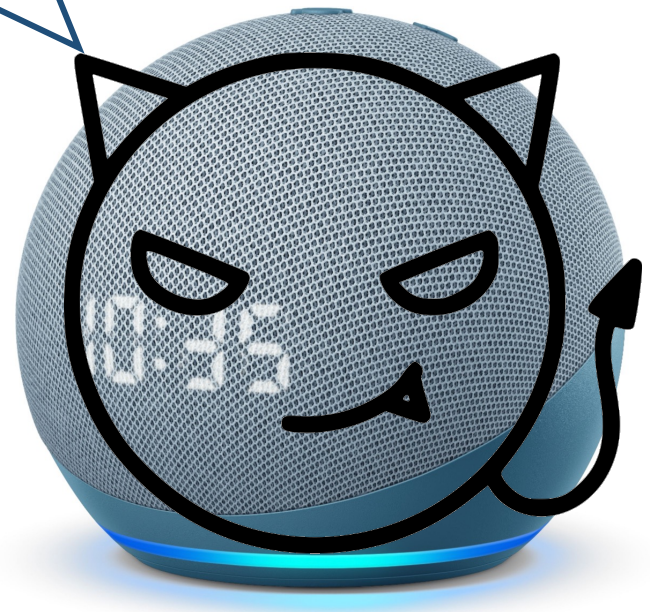
Alexa!

Outdoor Dolphin

I am not talking to you, Alexa!

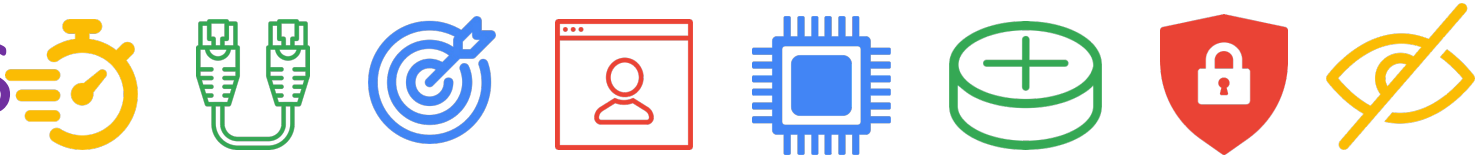
Jim

Yes, I am listening!



Jim's Alexa

Challenges with KWS



Security

- Safeguarding during the data that is being sent to the cloud

Bandwidth

- Minimize data sent over the network

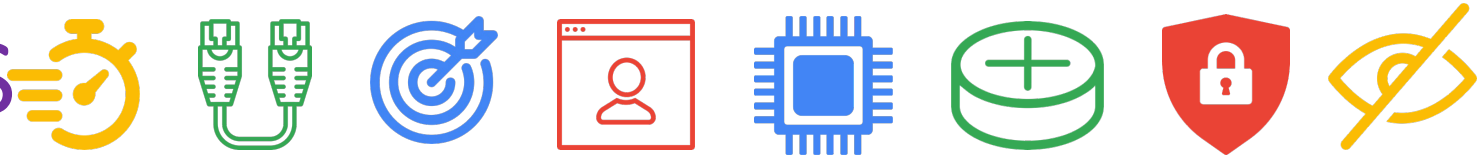
Battery

- Limited energy, operate on coin-cell type batteries

Memory

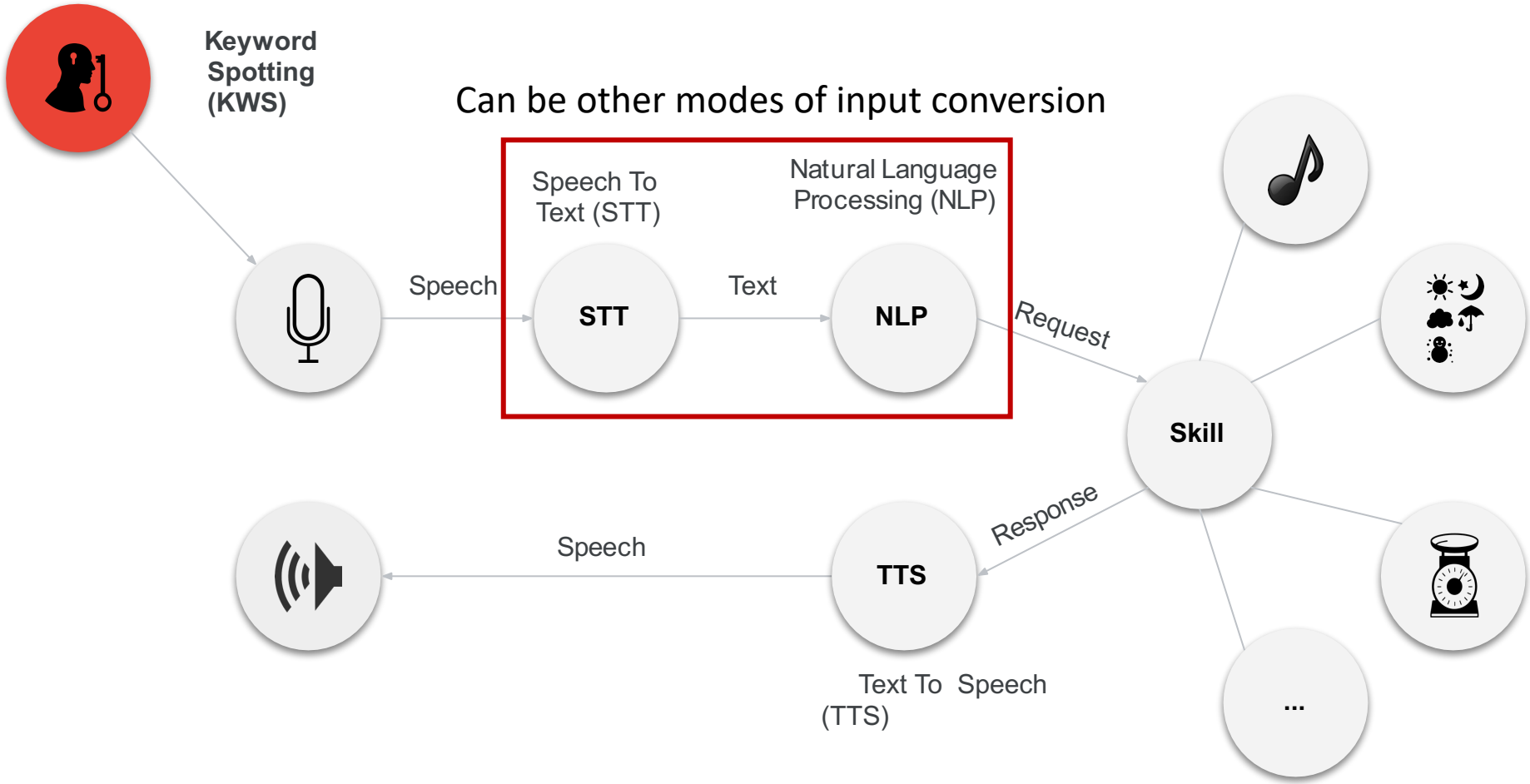
- Run on resource constrained devices

Challenges with KWS



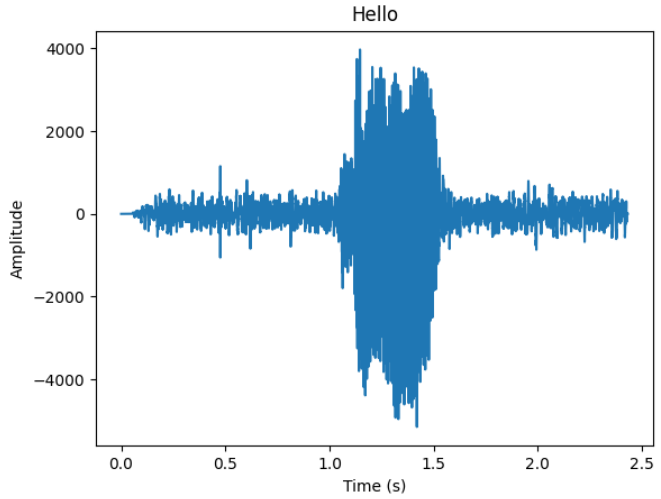
	Amazon Echo	Google Home Mini	Apple HomePod Mini
Memory	2 GB	256 MB	1 GB
Storage	8 GB	2 GB	32 GB
Processor	MT8163V (4 * ARM® Cortex-A53)	88DE3006 (4 * ARM® Cortex-A7)	S5 (Apple Watch)

End-to-End Flow of a General KWS System

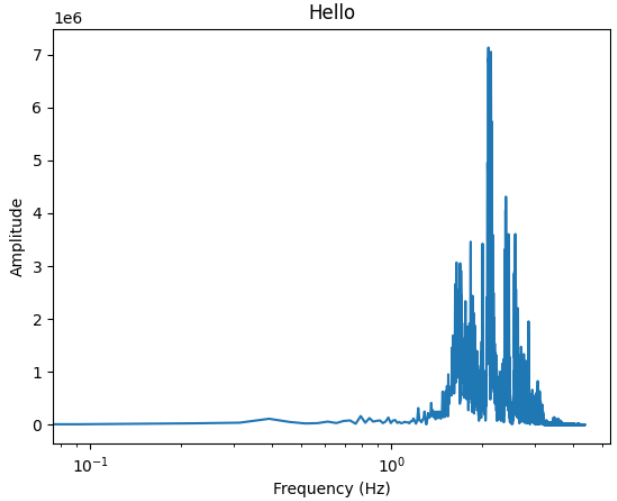


Audio Processing for KWS

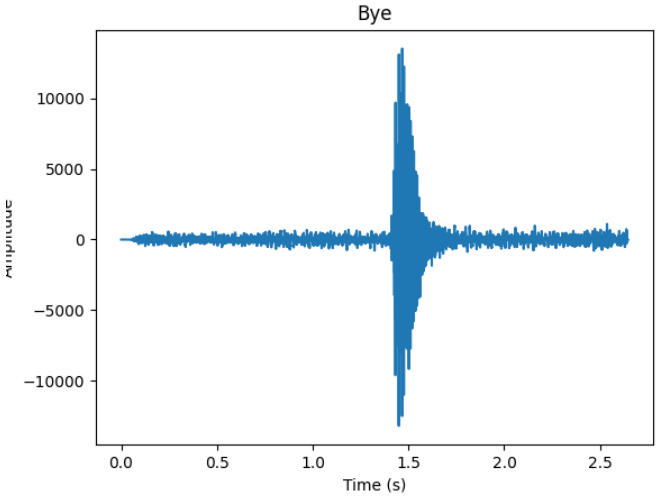
“Hello”

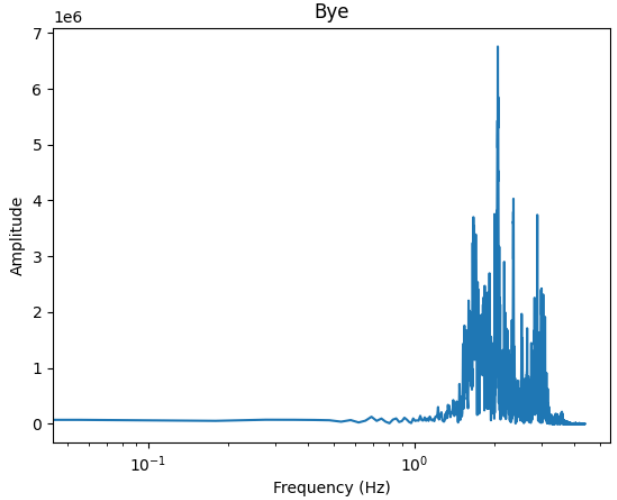


Fourier Transform



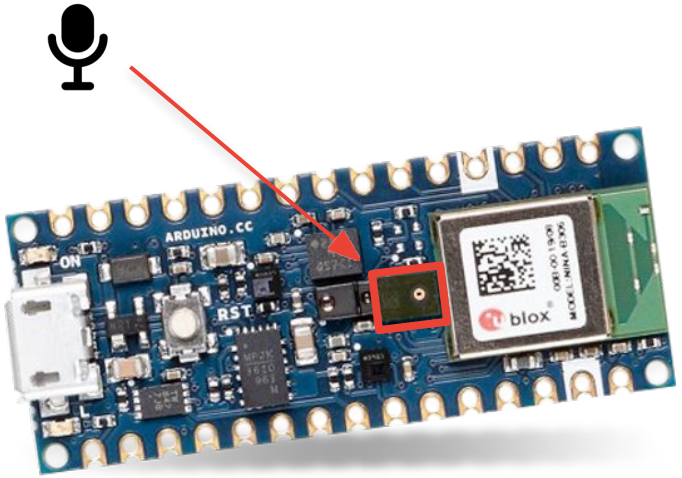
“Bye”



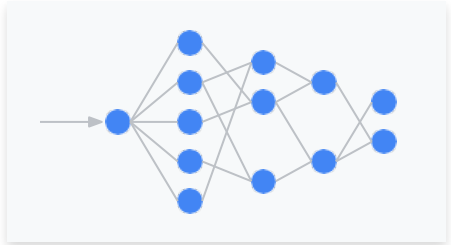


Anatomy of KWS Applications

1 Continuously listen on the microcontroller

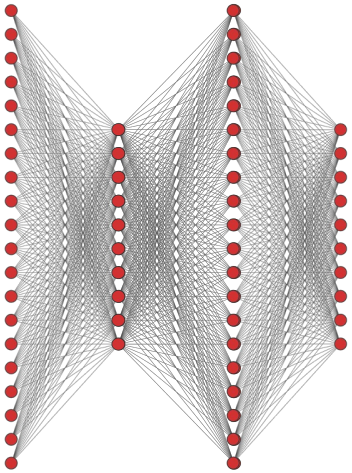


2 Process the data with **TinyML** at the edge

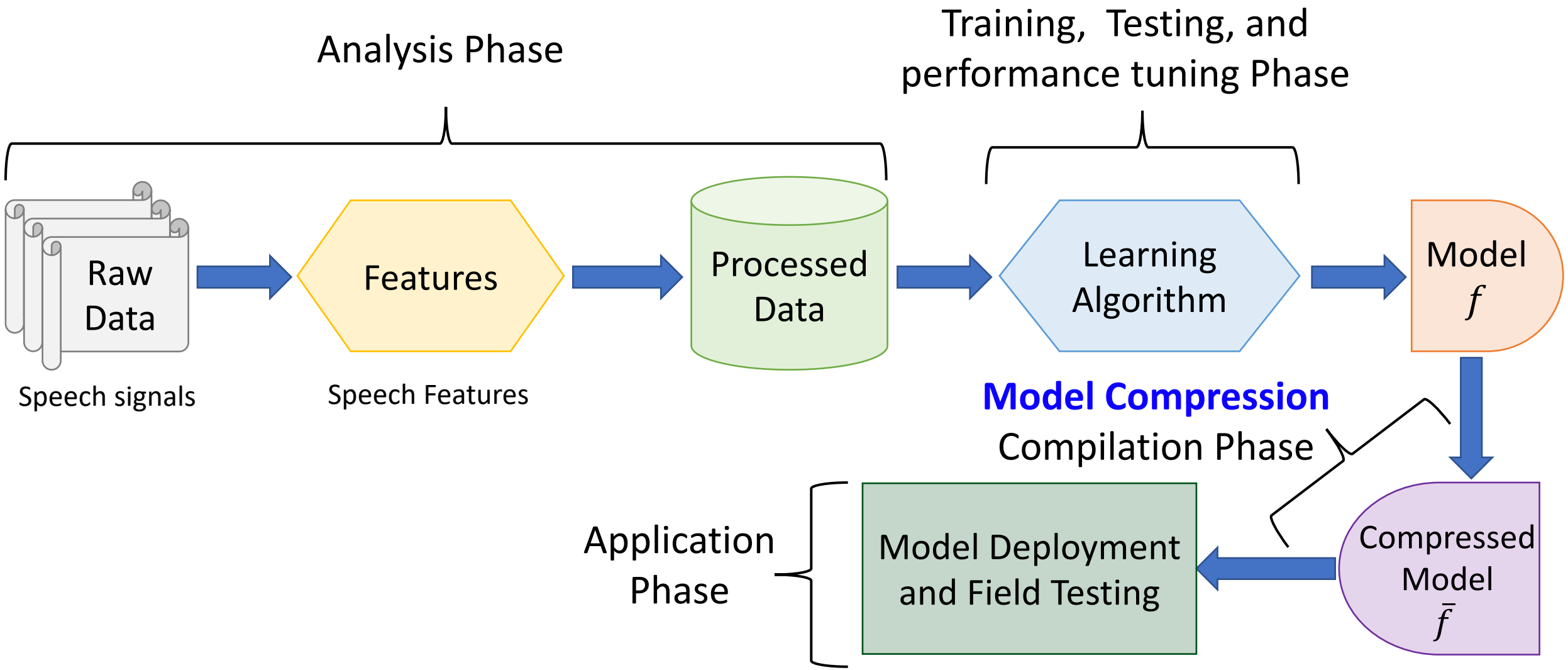


3 Send the data to the cloud when triggered

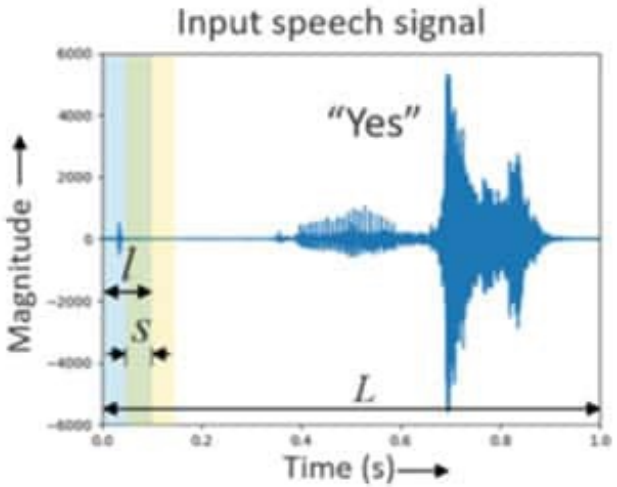
4 Process the full speech data with a large model in the cloud

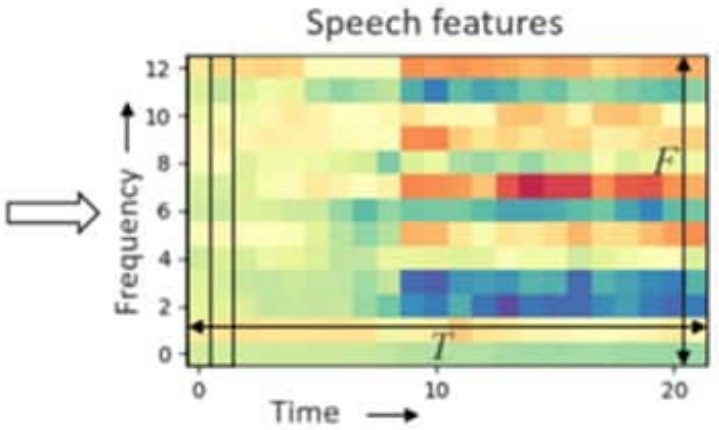


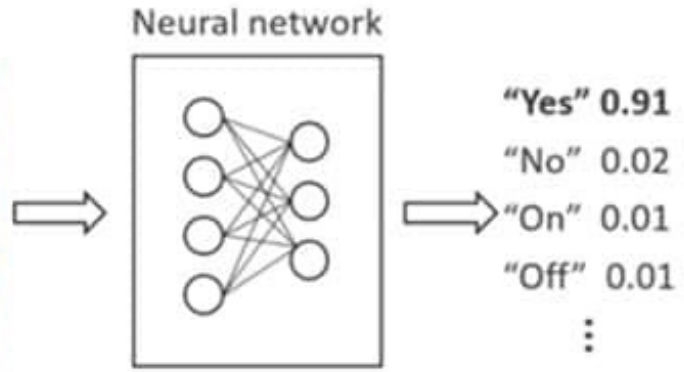
A Schematic View of TinyML and Its Phases



Breaking down TinyML phases







What specific features can we extract from speech signals?

How can we represent and process raw audio signals?

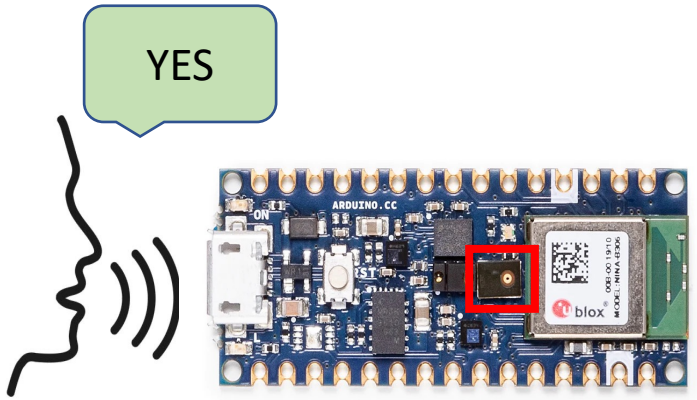
The spectrogram is a time series that is plotted against the frequency of the input signal.

How to capture a specific keyword using Machine Learning?

The spectrogram can be fed into a neural network (NN) to train the TinyML algorithm to recognize specific words.

KWS TinyML Implementation - Architecture

Overview of the KWS architecture



- In-built mic (MP34DT05) captures audio input from user
- Audio is converted into 16-bit PCM data

Data Pre-Processing (Feature Extractor)

- Raw audio signal is converted into **spectrograms** for feature extraction

TensorFlow Lite Model

- Model is optimized using quantization to run on the tiny device

Data Post Processing

- Command recognizer uses inference to determine whether action should be taken
- Command responder performs the action

Types of Audio Input

Single Shot Audio Input

"**Alexa**"

Listening..

"**Turn on the lights**"

"Turning on the lights"

Streaming Audio Input

"I am not able to see clearly, hey **Alexa turn on the lights**"

"Turning on the lights"

KWS Datasets

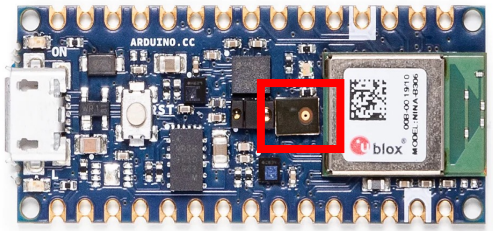
- TensorFlow provides speech_commands audio dataset of spoken words designed to help train and evaluate keyword spotting systems.
- The dataset is derived from paper: Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition

Split	Examples
'test'	4,890
'train'	85,511
'validation'	10,102

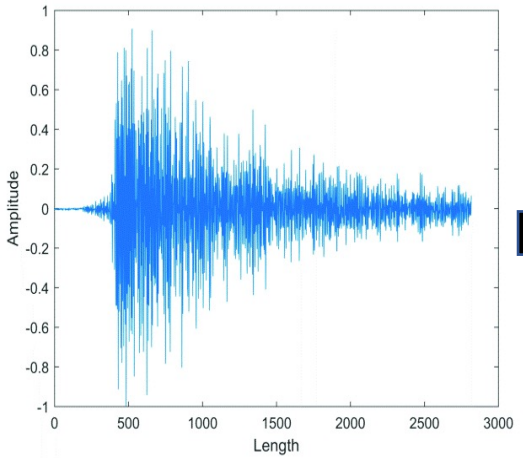
Data Pre-Processing for KWS

Data pre-processing for KWS – Audio signal processing to:

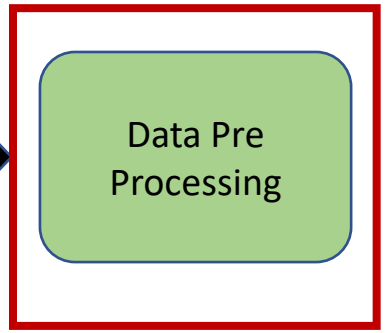
- Identify the **start of signal** for wake word recognition
- **Extracting important features** for feeding into the KWS model

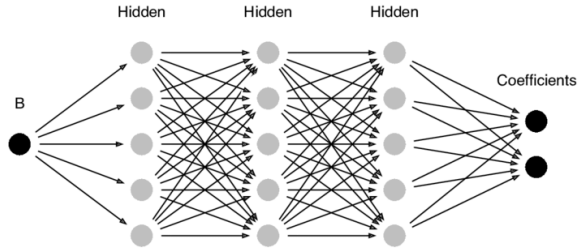


Input audio signal to microphone



Audio to Electric Signal (in Time)



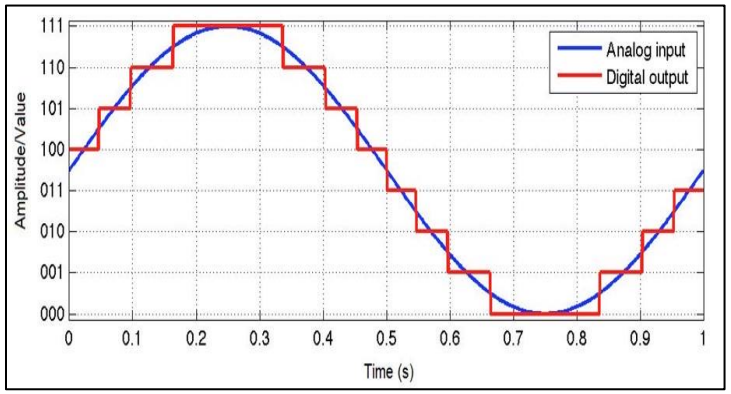


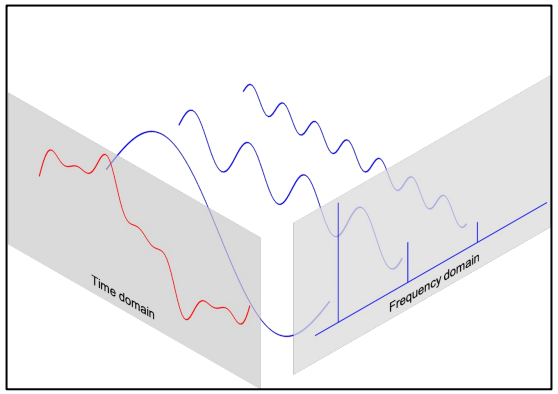
Keyword Spotting Model (NN)

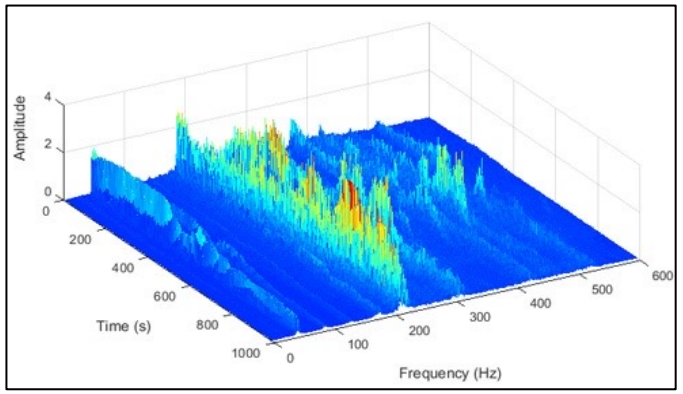
Data Pre-Processing for KWS

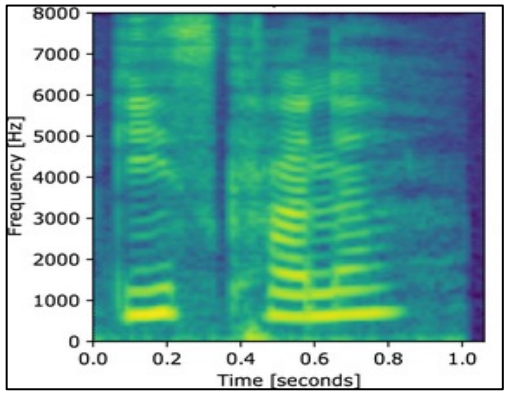
Data pre-processing steps:

Convert raw Analog Audio Signal to Digital Signal → **Fast Fourier Transform (FFT)** → **Convert FFT to Spectrogram – Sliding Window** → **Generate Mel Frequency Cepstral Coefficients (MFCC)**





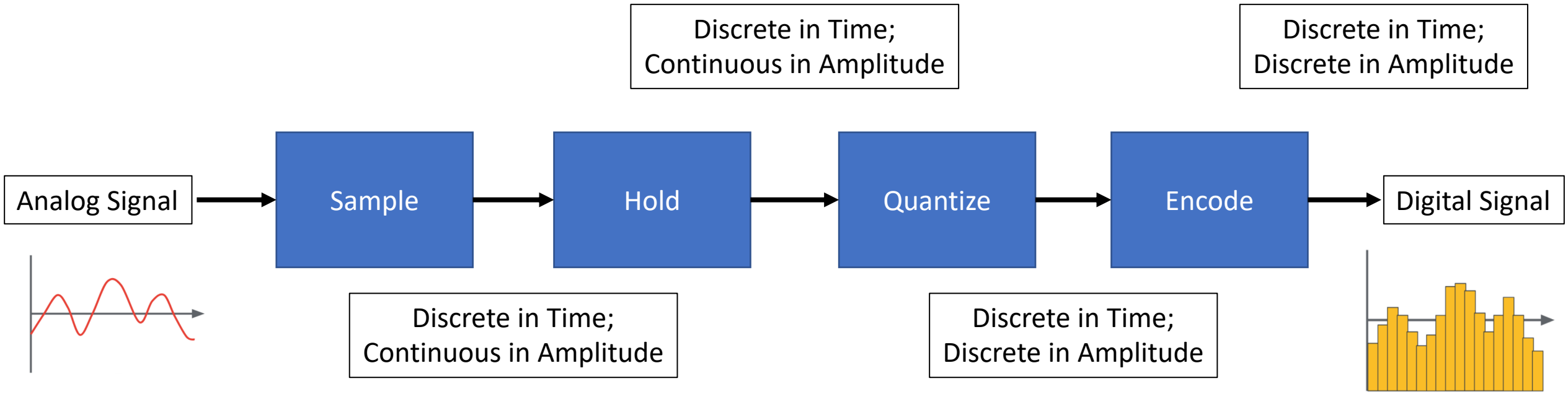




Source: https://heartbeat.comet.ml/neural-network-pruning-research-review-2020-bc21a77f0295

Audio - Analog to Digital Conversion

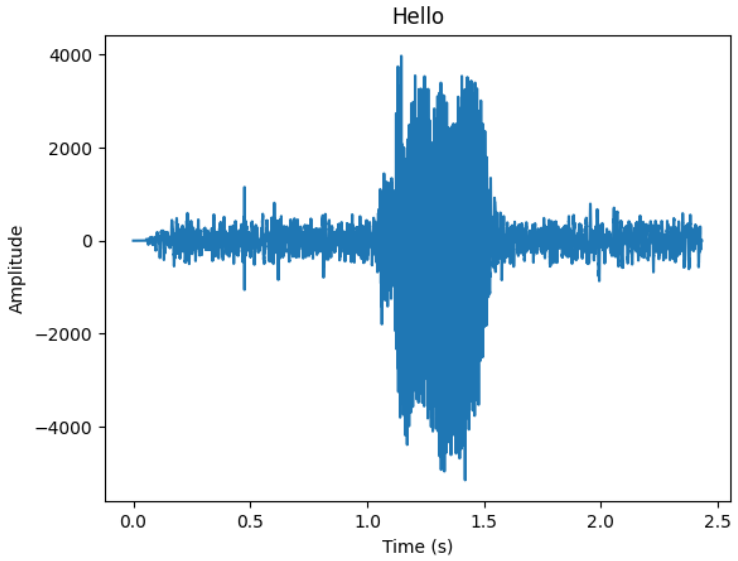
- Raw analog audio signal is first transformed into a digital signal.
- Human frequency is between 20 Hz – 20 KHz and the sampling rate must be determined accordingly.



Fast Fourier Transform (FFT)

- The digital signal in the time domain (Time v. Amplitude) is transformed into the frequency domain (Frequency v. Amplitude) using Discrete Fourier transform.
- The Fast Fourier Transform (FFT) is an efficient algorithm to calculate the DFT of a sequence.

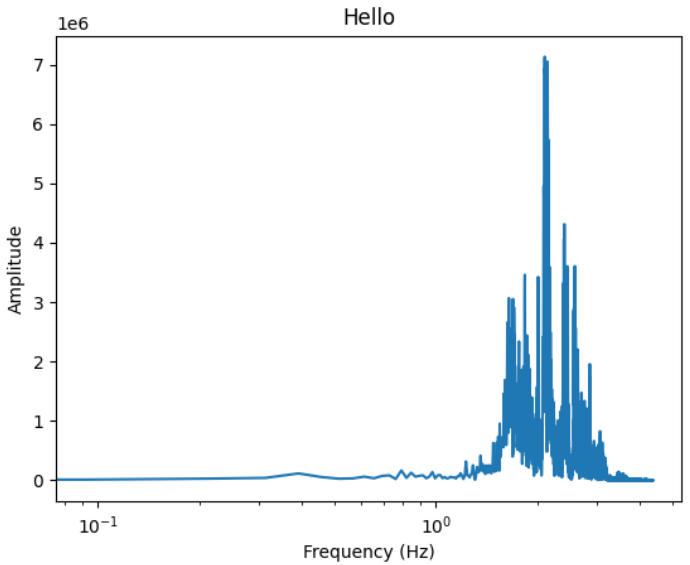
Word: Hello



Speech Signal in Time Domain

Fast Fourier Transform

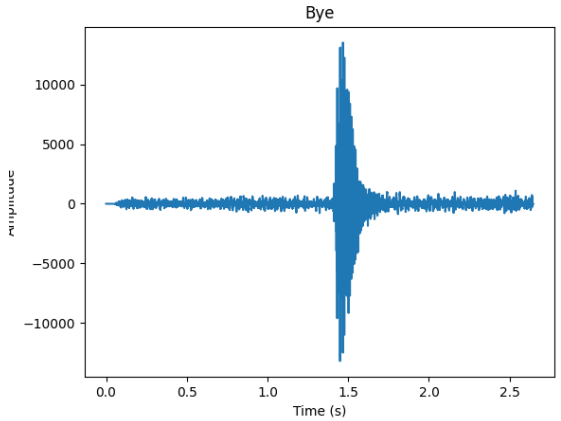
Word: Hello

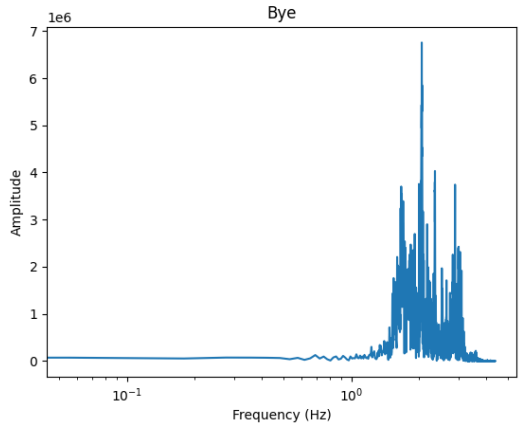


Speech Signal in Frequency Domain

Fast Fourier Transform (FFT)

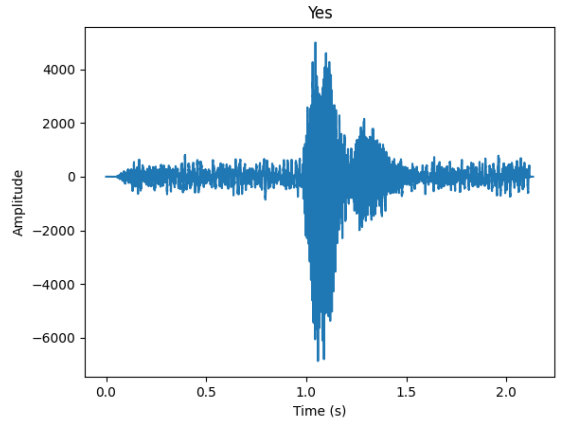
Word: Bye

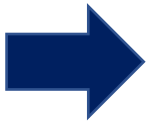


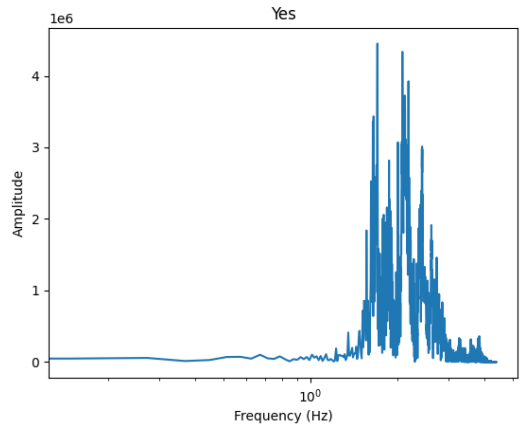


Fast Fourier Transform

Word: Yes



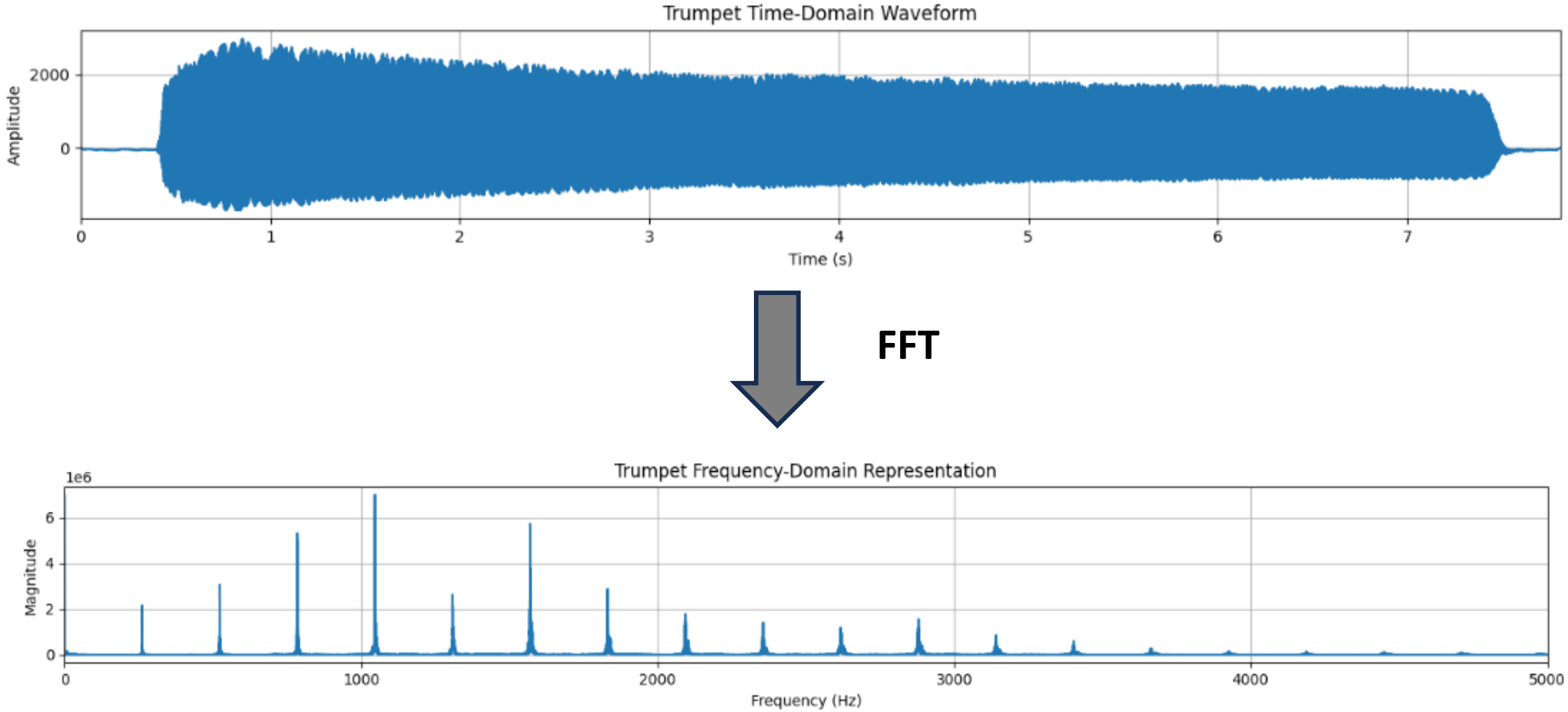




Speech Signal in Time Domain

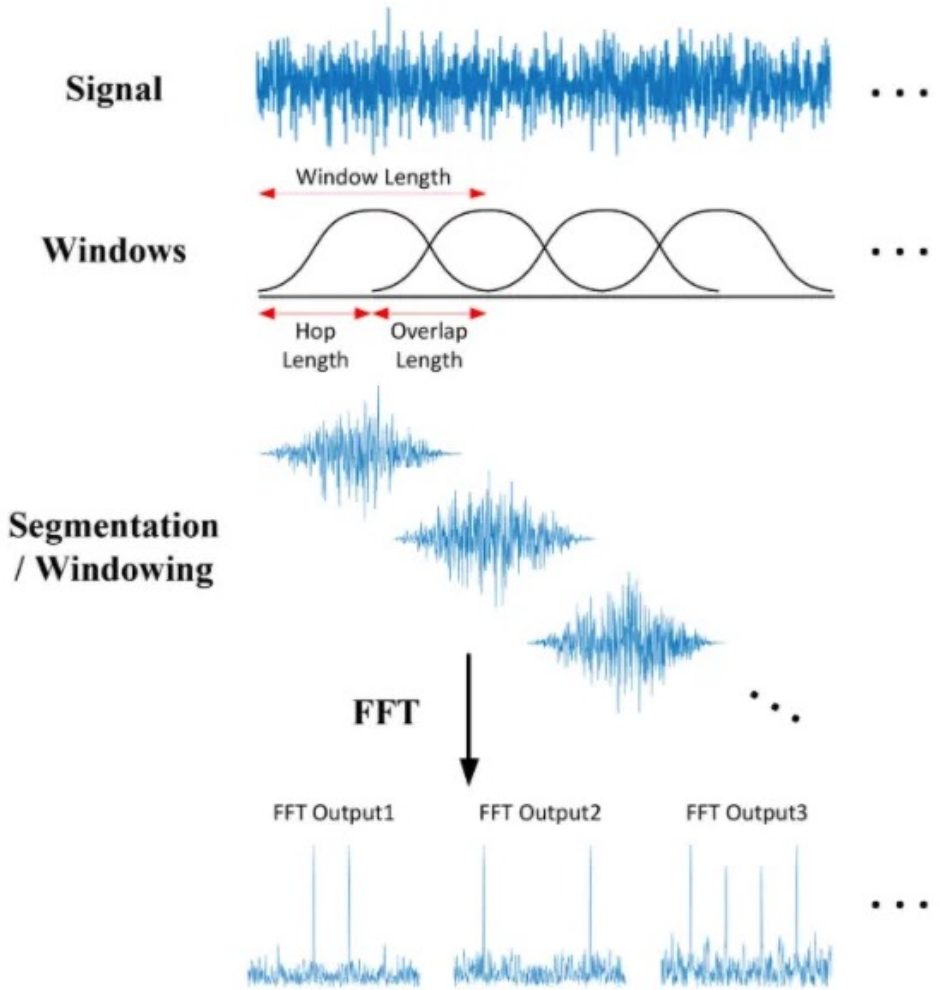
Speech Signal in Frequency Domain

Fast Fourier Transform (FFT)



FFT to Spectrogram

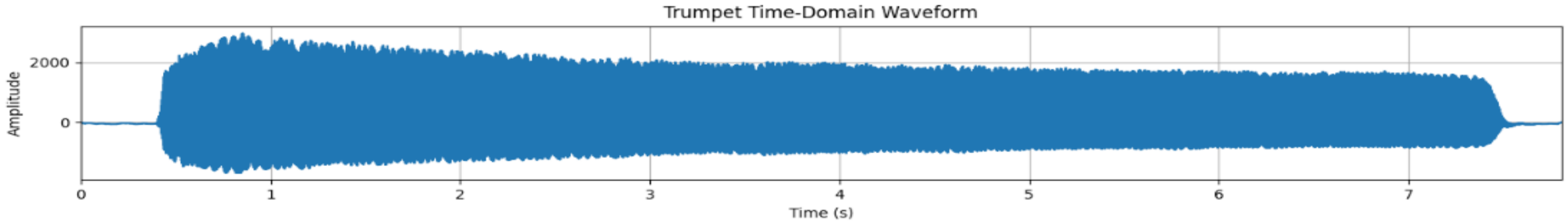
Apply the FFT through a sliding window



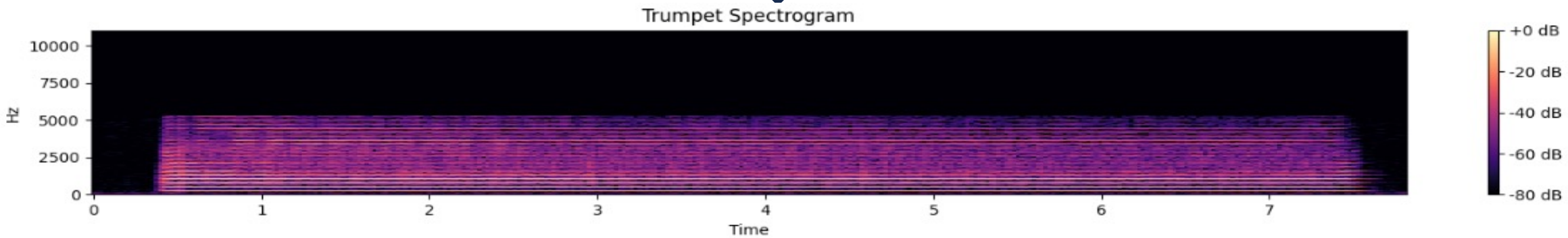
https://towardsdatascience.com/all-you-need-to-know-to-start-speech-processing-with-deep-learning-102c916edf62

FFT to Spectrogram

- Spectrogram contains varying color intensity for each extracted frequency over time.
- Since this extraction is done at the pre-processing state of the TinyML pipeline, this improves latency and decreases computational requirements of DNN.

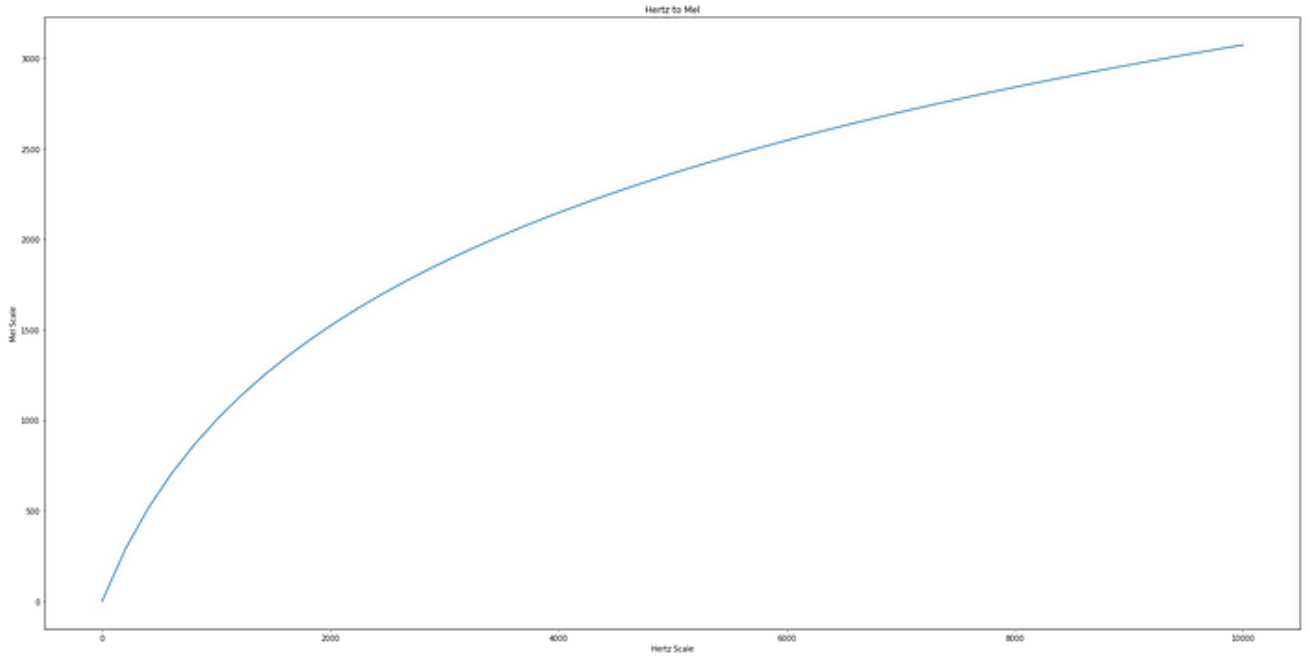


Sliding Window FFT



Mel Scale

- Perceptual scale of pitches that is designed to better represent the way humans hear sounds.
- Pitch is how high or low a sound feels to a person.
- When an object vibrates quickly, it makes a high-pitched sound (e.g., chirping of birds), and when it vibrates slowly, it makes a low-pitched sound (e.g., sound of distant thunder).



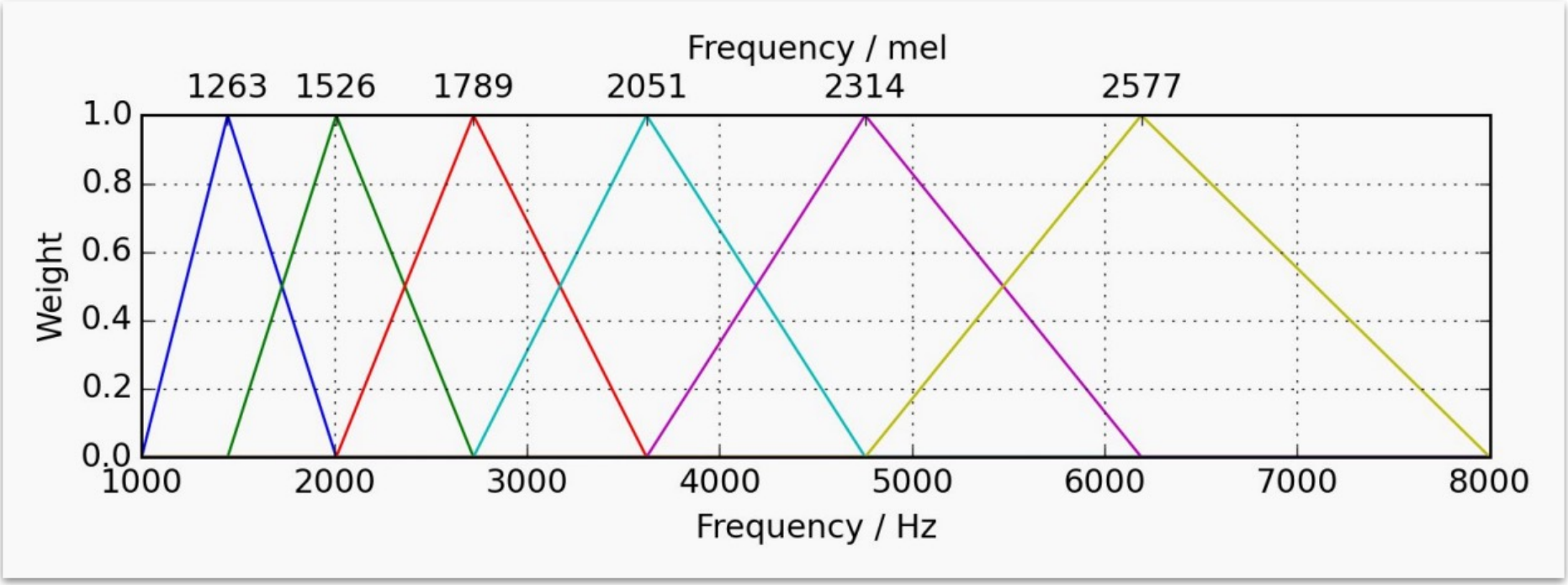
Hertz Scale

$$m = 1127 \cdot \log\left(1 + \frac{f}{700}\right)$$

- Frequencies that are lower in Hz have a larger distance between them Mels.
- Frequencies that are higher in Hz have a smaller distance between them in Mels.

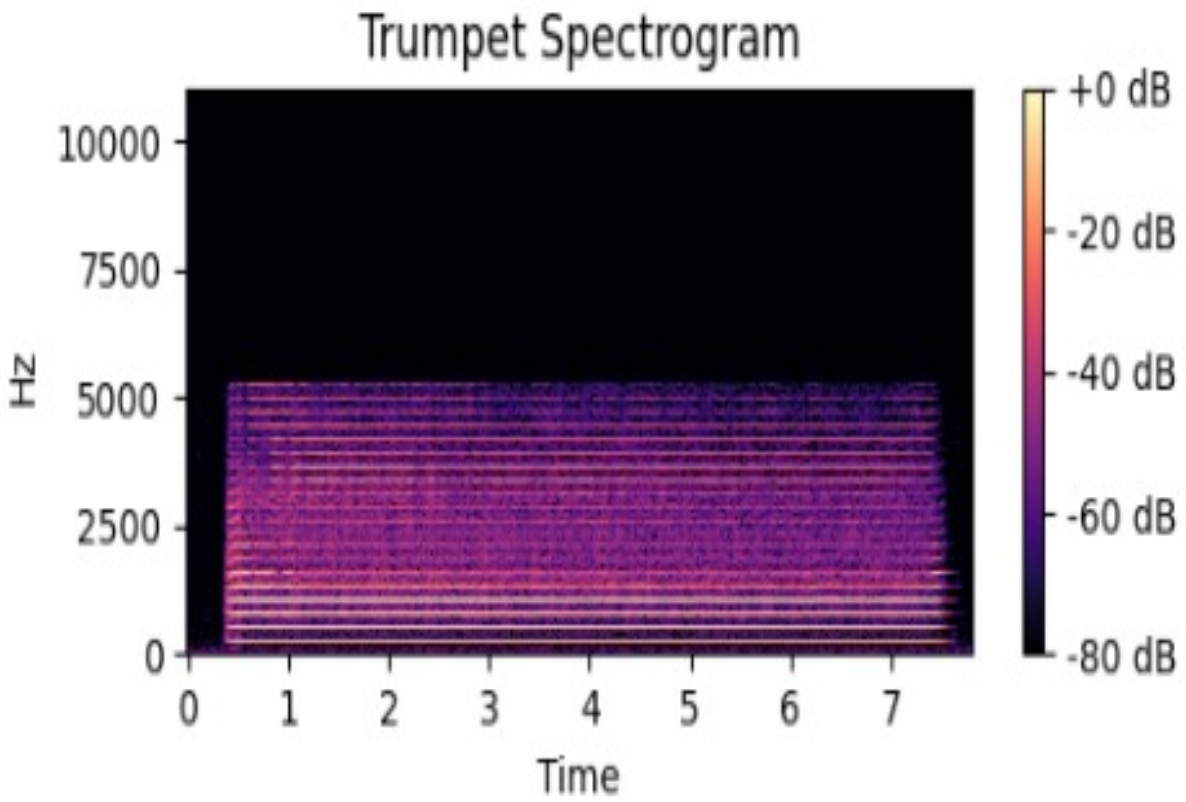
Spectrogram to Mel-Spectrogram

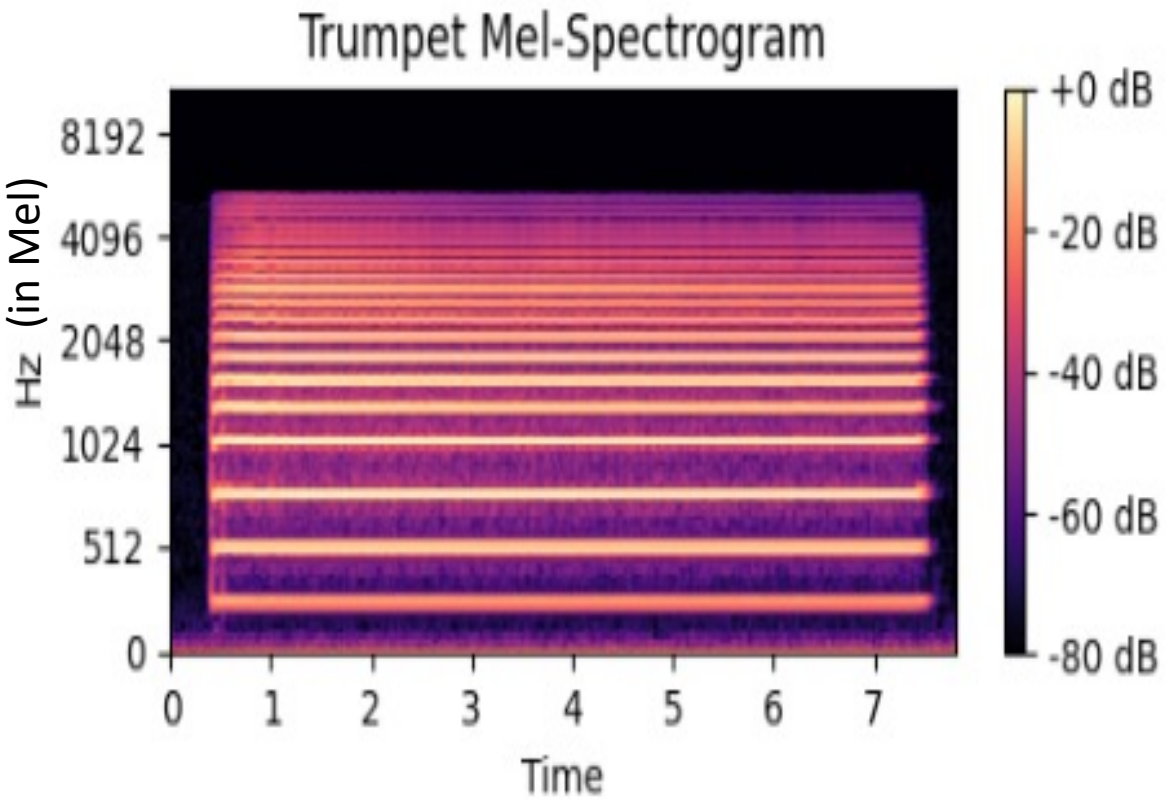
- Mel scale is a non-linear transformation of the frequency scale to match the human auditory system's response more closely
- In practice, the Mel filter bank is applied to map the frequencies onto the Mel scale



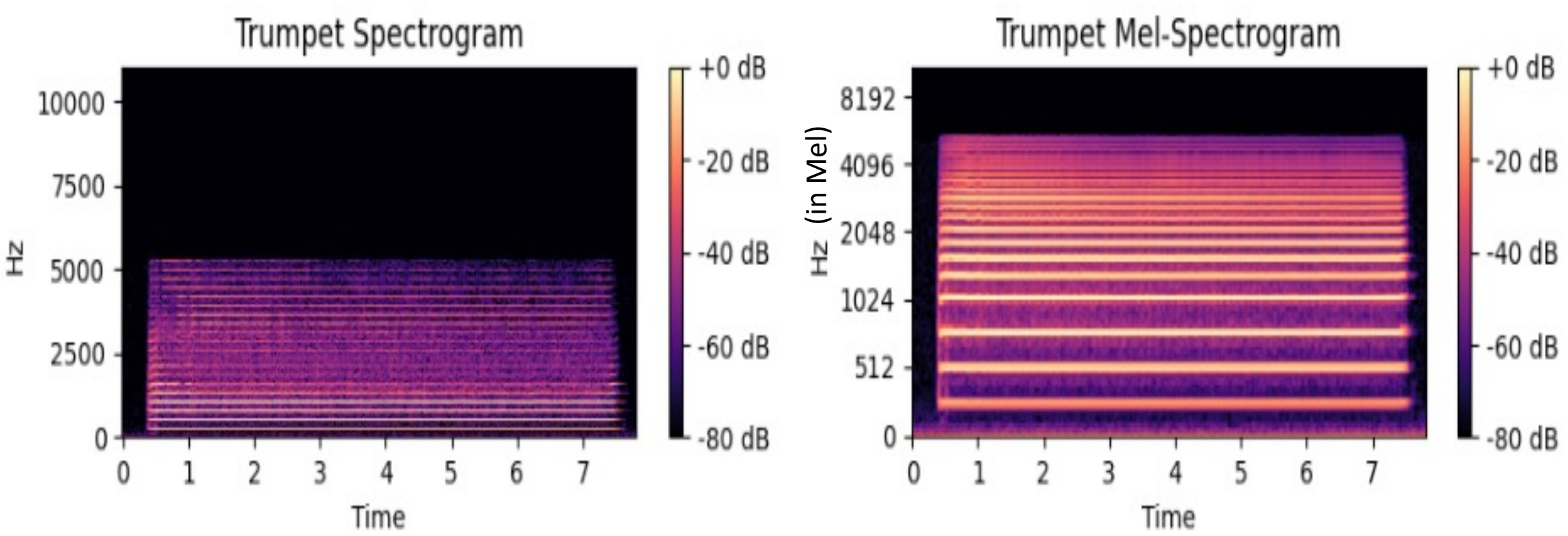
Spectrogram to Mel-Spectrogram

- In Mel-spectrogram y-axis represents frequencies in the mel-scale
- In Mel scale, equal distances correspond to equal perceptual differences in pitch





Spectrogram to Mel-Spectrogram



- Mel spectrogram is still noisy – not all components may be needed for analyzing the sounds.
- How can we convert this into a reliable image while the amount of information retained is minimized?

Discrete Cosine Transform (DCT)

- Signal processing technique particularly used in the context of data compression, such as in JPEG image compression.
- Converts a finite sequence of equally-spaced samples of a function (like a signal or image) into a sum of cosine functions oscillating at different frequencies and amplitudes.
- The two-dimensional DCT for an $M \times N$ matrix of image pixel values x_{mn} is defined as:

$$X_{kl} = \alpha(k)\alpha(l) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{mn} \cos\left[\frac{\pi}{M}\left(m+\frac{1}{2}\right)k\right] \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)l\right]$$

Represents the amplitude of a certain frequency component in the image data

Discrete Cosine Transform (DCT)

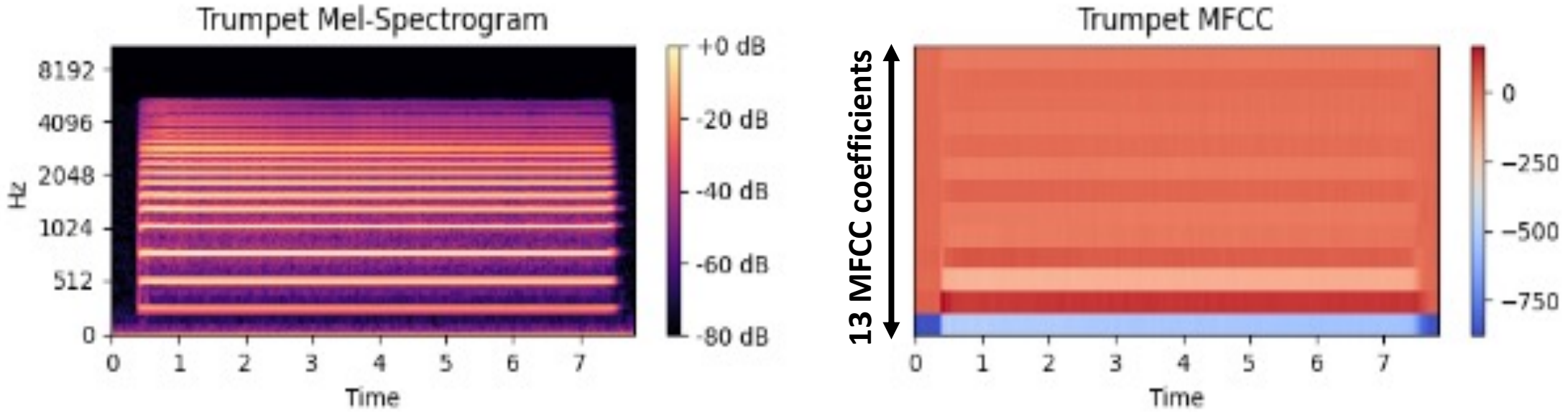
$$X_{kl} = \alpha(k)\alpha(l) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{mn} \cos\left[\frac{\pi}{M}\left(m+\frac{1}{2}\right)k\right] \cos\left[\frac{\pi}{N}\left(n+\frac{1}{2}\right)l\right]$$

where:

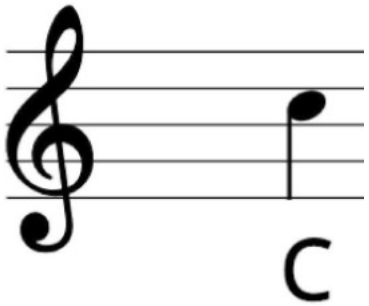
- X_{kl} is the DCT coefficient at indices k and l,
- m and n are the row and column indices of the input matrix,
- k and l are the row and column indices for the DCT coefficients,
- M and N are the dimensions of the input matrix,
- $\alpha(k)$ and $\alpha(l)$ are normalization factors, often chosen to ensure the transform is orthonormal. Typically, $\alpha(0) = \sqrt{\frac{1}{N}}$ and $\alpha(k) = \sqrt{\frac{2}{N}}$ for $k, l > 0$.

Mel Frequency Cepstral Coefficients (MFCC)

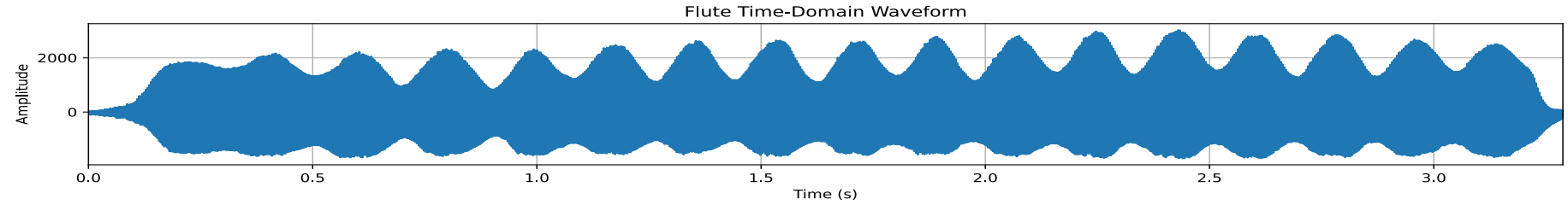
- The Discrete Cosine Transform (DCT) is applied to the logarithm of the Mel spectrogram to decorrelate the Mel frequency bands.
- The result of the DCT is a set of coefficients that compactly represent the sound's spectral shape.
- The first few of these coefficients (excluding the first one, which represents the average power) are the Mel-frequency Cepstral Coefficients (MFCCs).

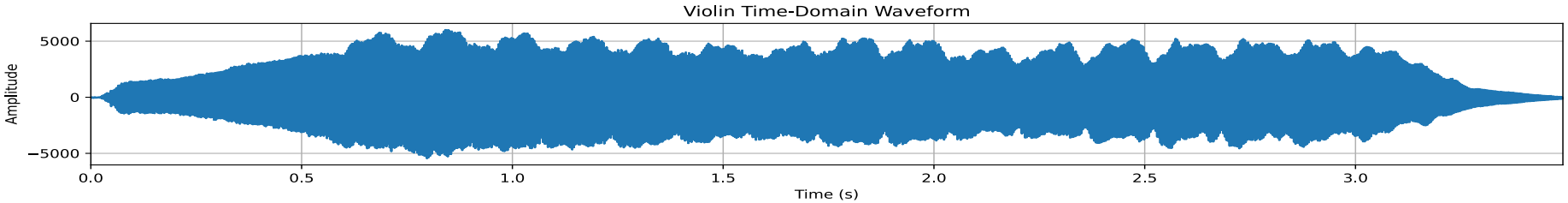


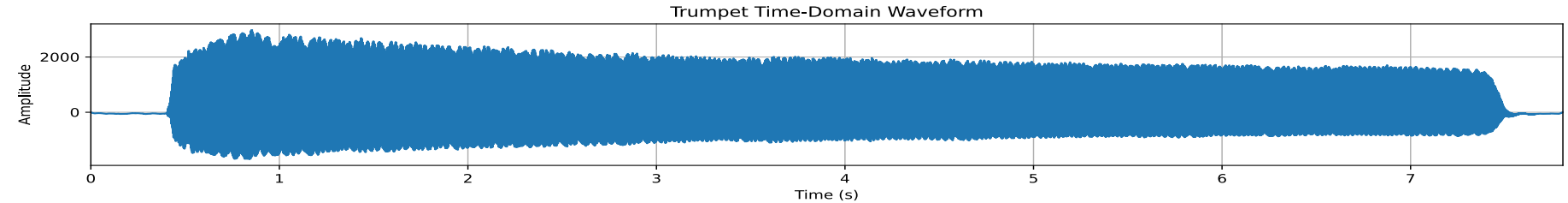
Example: Different Musical Instruments Playing Middle C

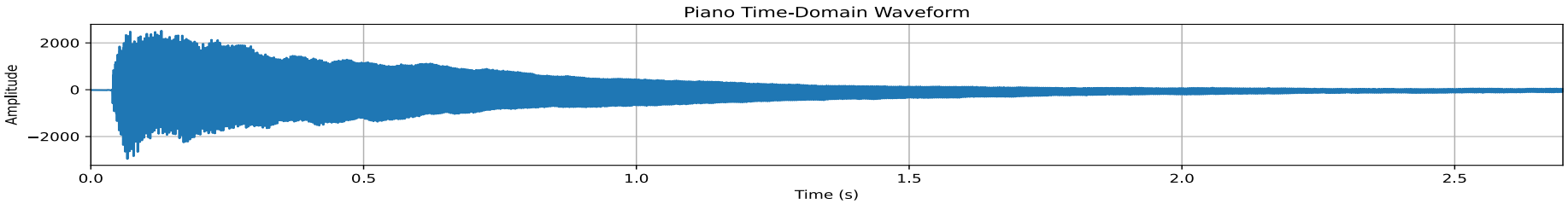


Middle C (C4)
261.63 Hz

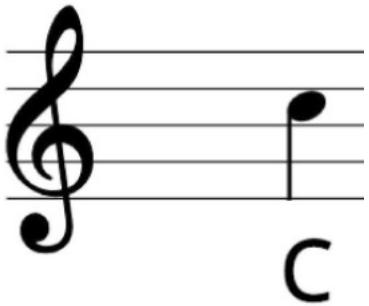




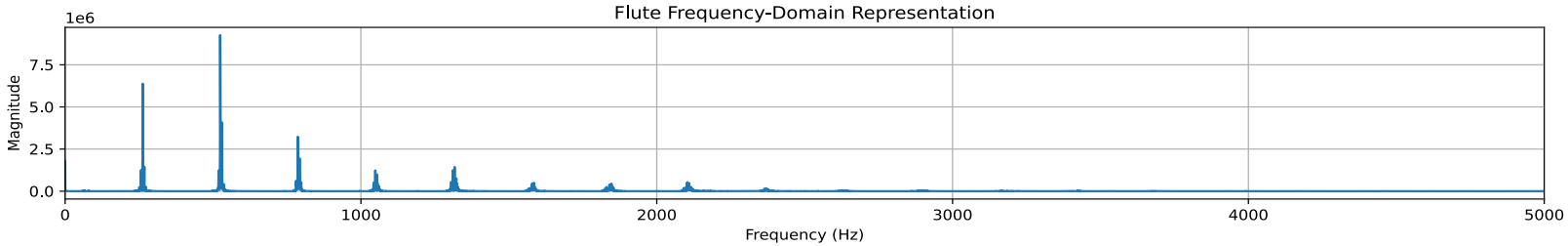


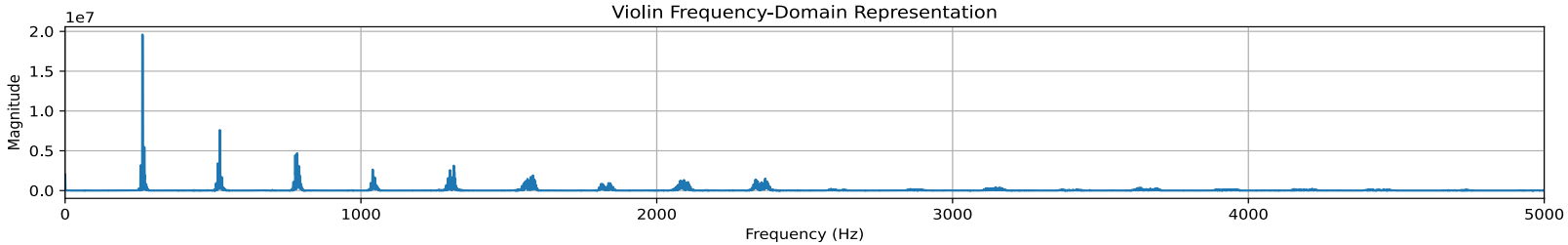


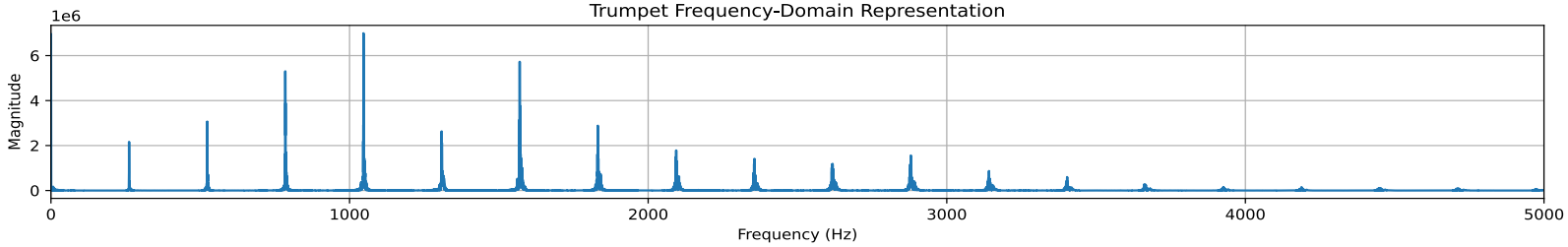
FFT Results

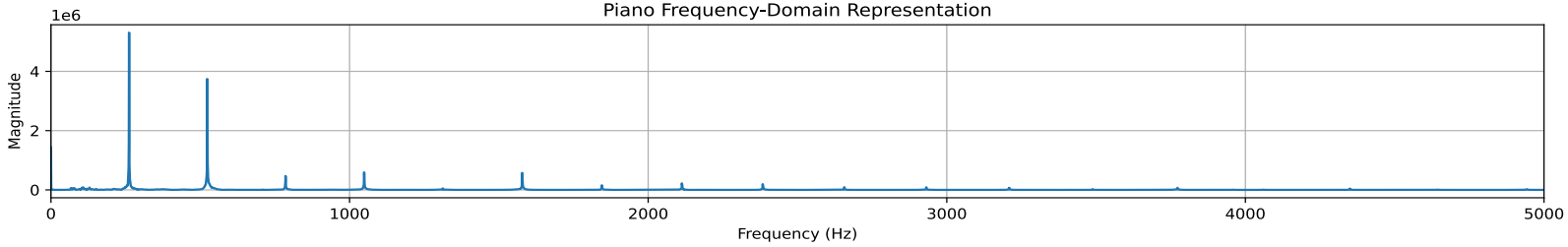


Middle C (C4)
261.63 Hz

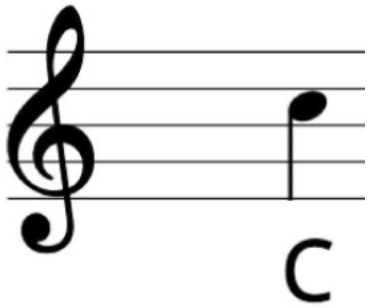




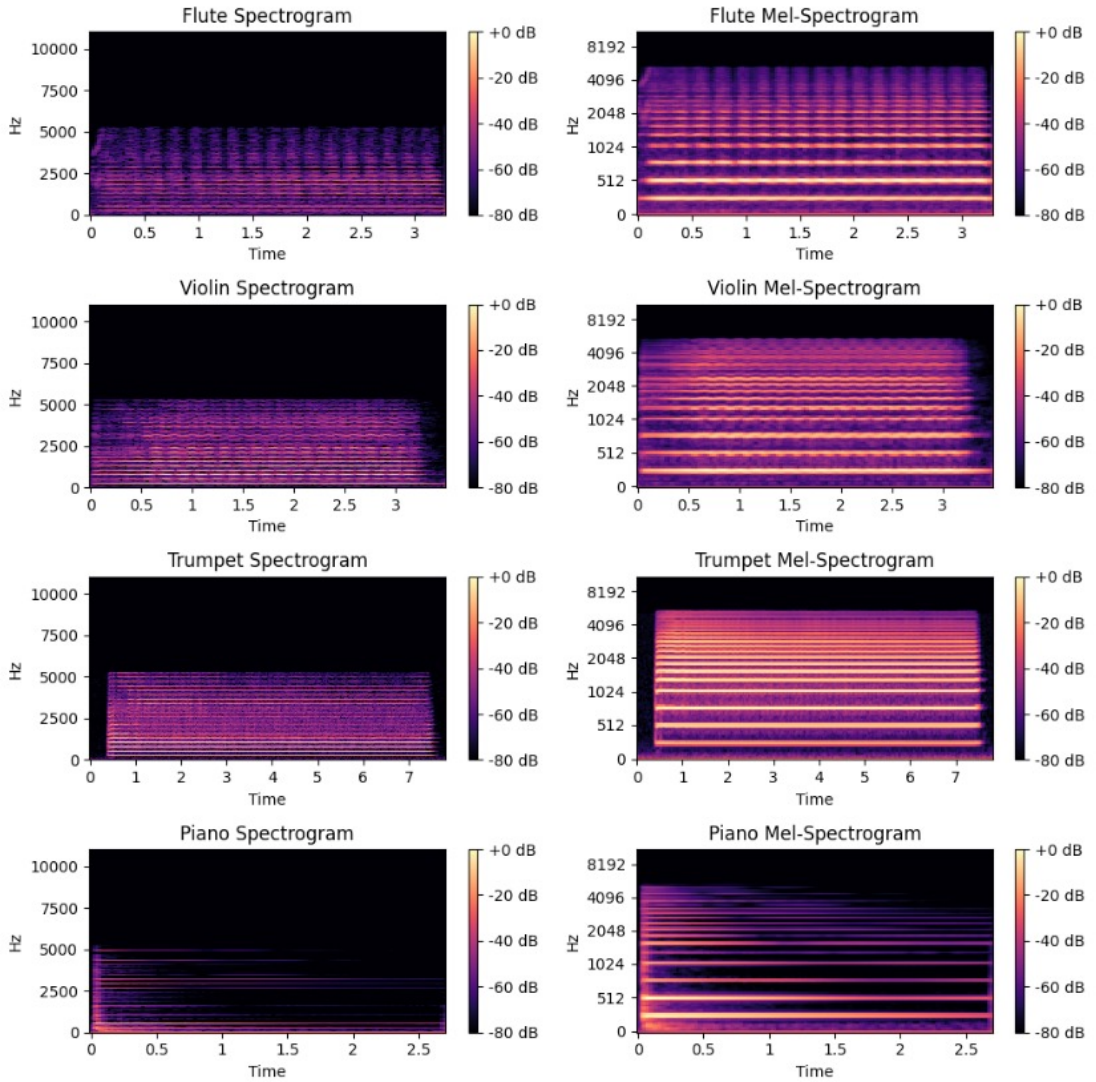




Spectrogram Vs. Mel Spectrogram Vs. MFCC

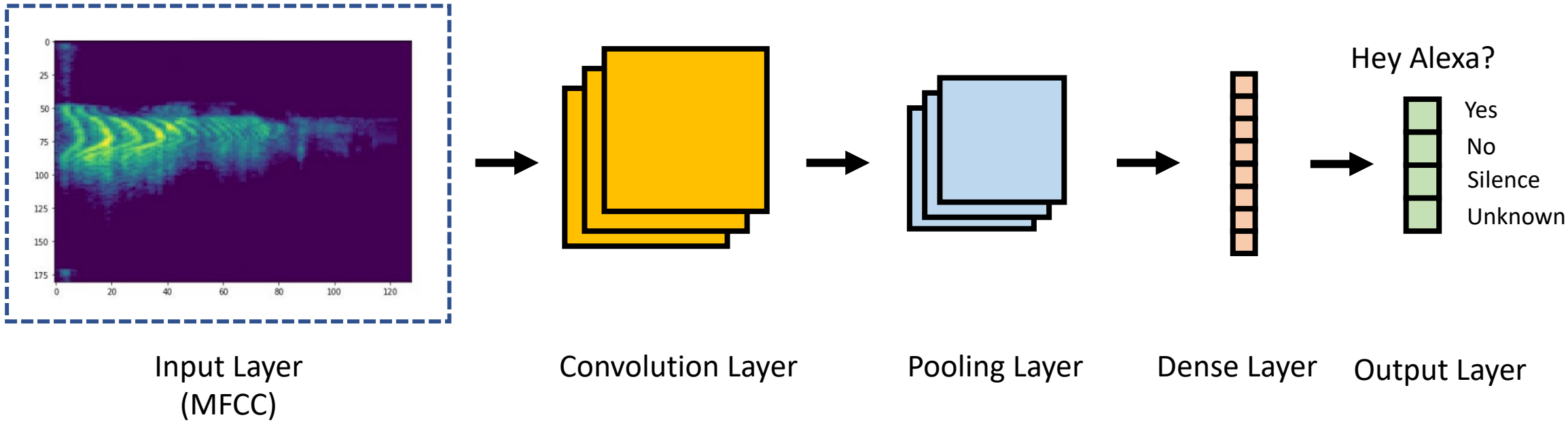


Middle C (C4)
261.63 Hz



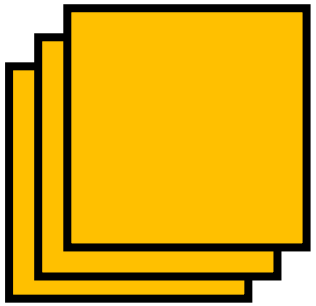
ML Model for KWS

Introduction to Convolutional Neural Networks (CNNs)



Convolutional Neural Networks

Convolution Layer

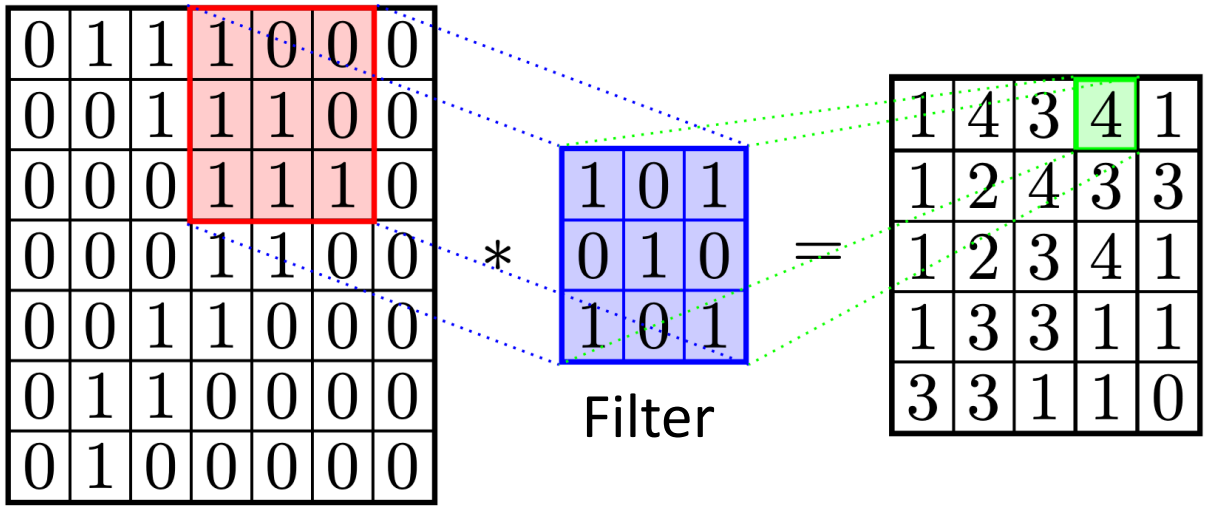


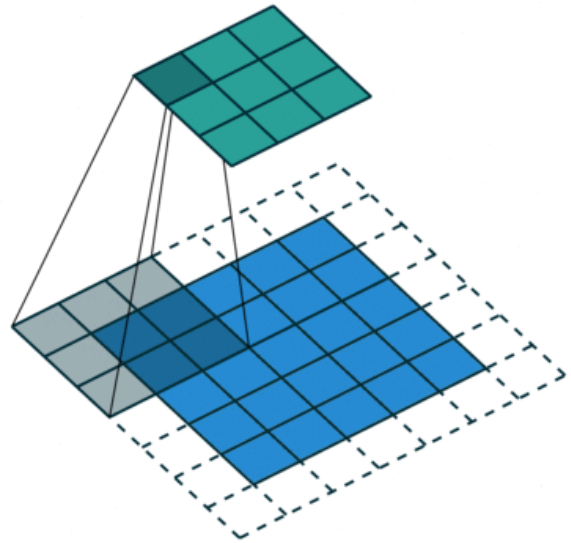
Recall Convolution:

One-dimensional discrete-time signal processing, linear time-invariant systems, convolve with FIR (finite impulse response) filter

$$y(t) = (h * x)(t) = \sum_{\tau} h(t-\tau)x(\tau) = \sum_{\tau} h(\tau)x(t-\tau)$$

Convolution can also be done in 2-D.

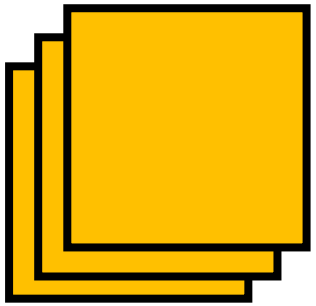




https://i.stack.imgur.com/i3wEL.png

Convolutional Neural Networks

Convolution Layer



The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image.

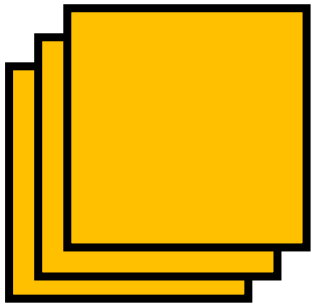
What are filters/kernels?

- measure for **how close** a patch or a region of the input resembles a feature
- CNN **learns multiple features** in parallel for a given input.
- For example, it is common for a convolutional layer to learn from 32 to 512 filters in parallel for a given input.

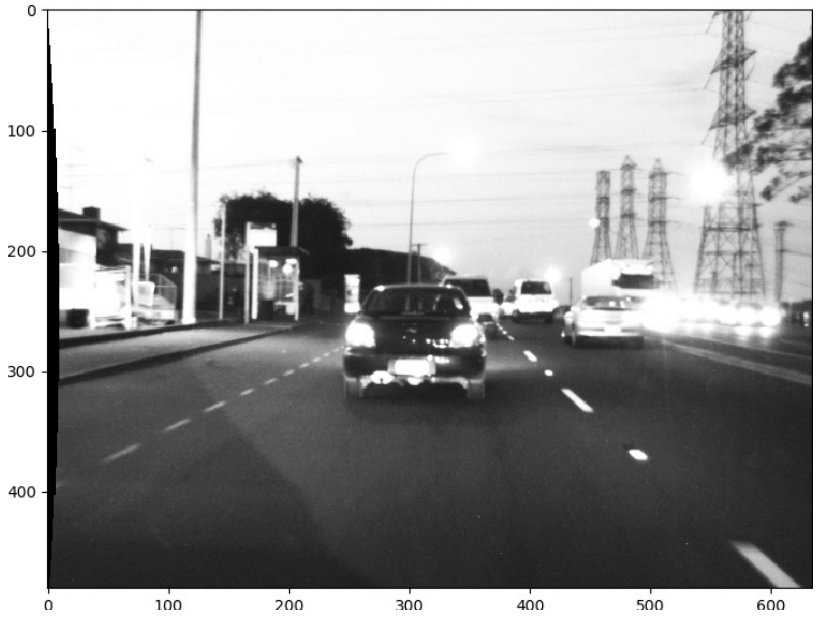
We will give examples of different types of filters.

Convolutional Neural Networks

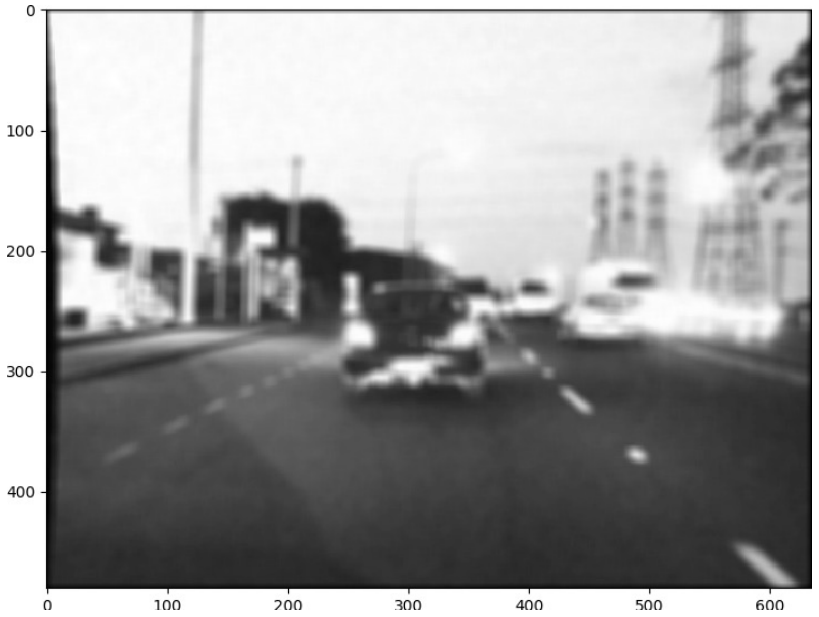
Convolution Layer **1. Box, mean or average filter**



$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



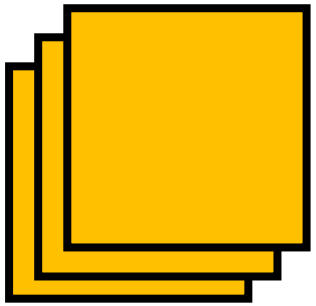
Original



Filtered

Convolutional Neural Networks

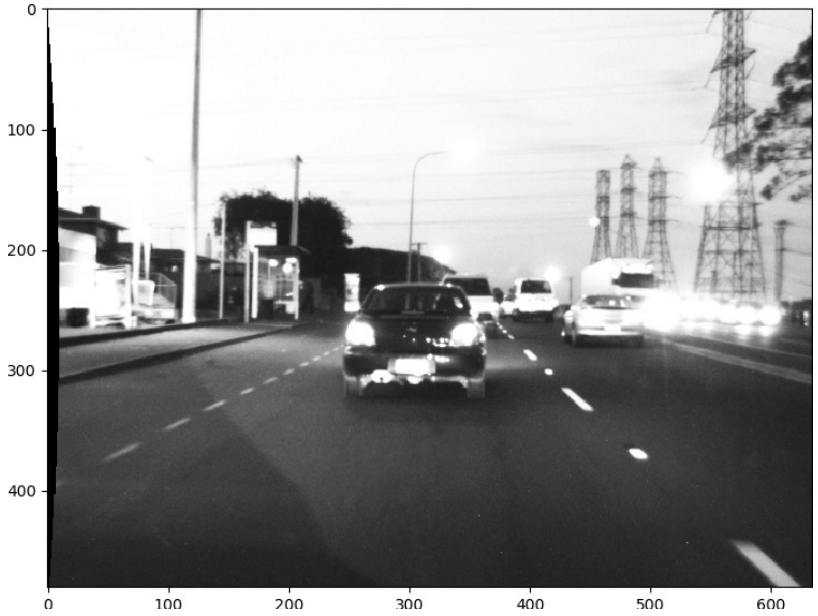
Convolution Layer



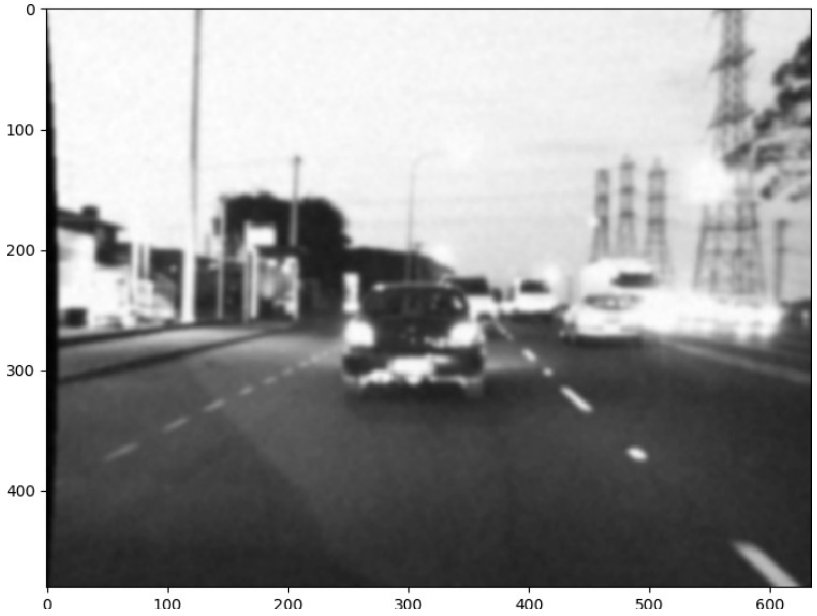
2. Gaussian Filter

Derived from 2D gaussian function $h(u,v) = \frac{1}{2\pi\sigma^2} e^{\left(-\frac{u^2+v^2}{\sigma^2}\right)}$

Example: $\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$



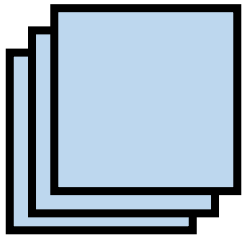
Original



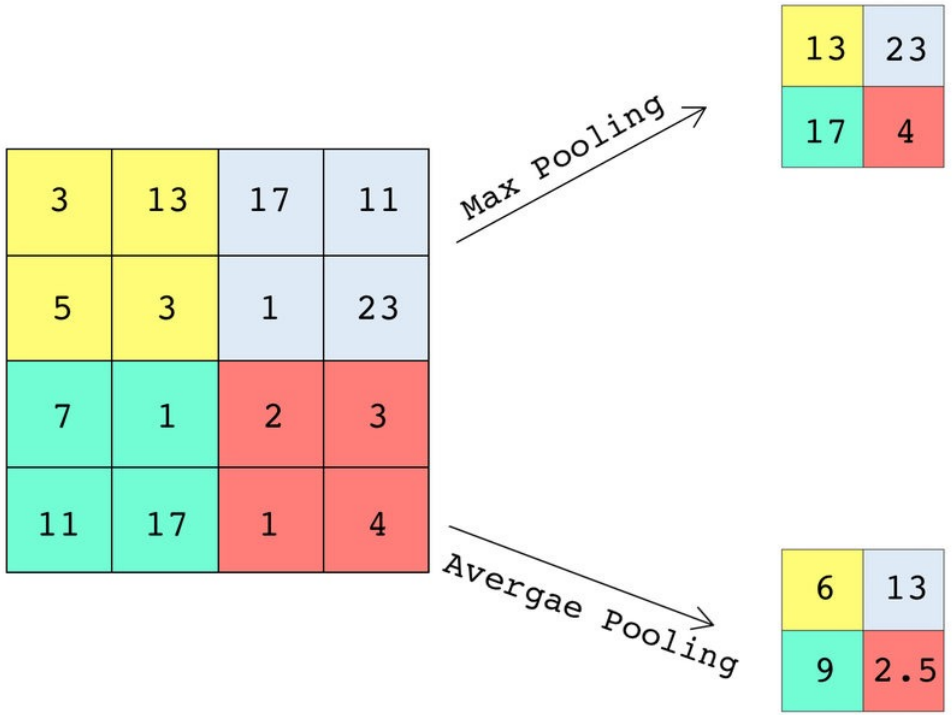
Filtered

Convolutional Neural Networks

Pooling Layer

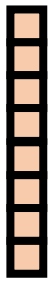


- Decrease the computational power required to process the data through **dimensionality reduction**.
- **Extracting dominant features** which are rotational and positional invariant

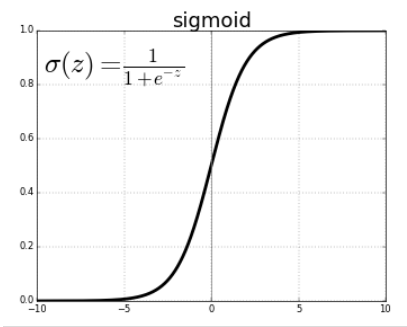


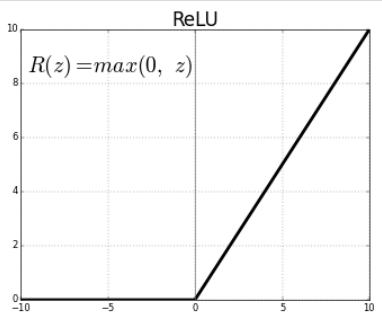
Convolutional Neural Networks

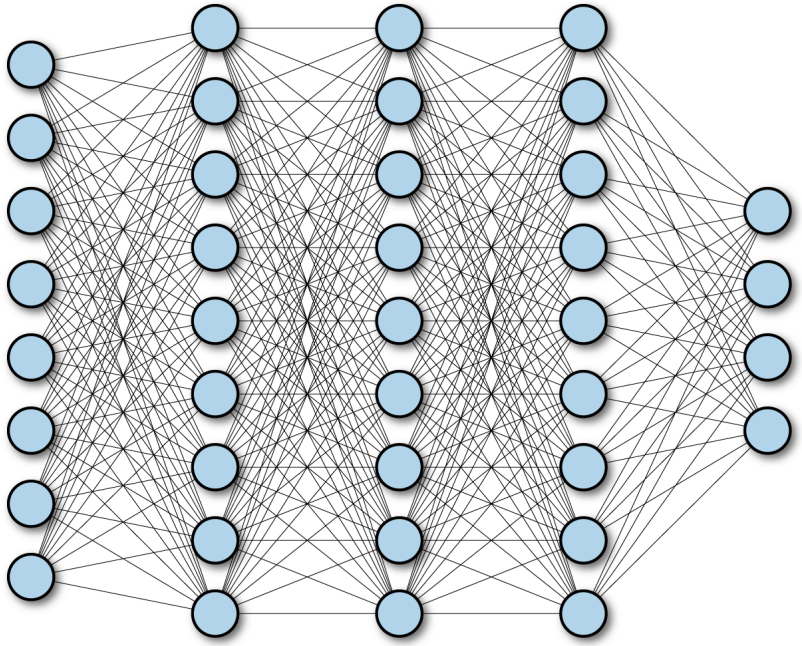
Dense Layer (Fully-Connected Layer)



- **Neuron connections**: Each layer receives input from all neurons in the previous layer and sends its output to all neurons in the next layer.
- **Weighted sum**: The input to each neuron is a weighted sum of the outputs from the previous layer
- **Activation function**: An activation function, such as ReLU or sigmoid, is applied to the weighted sum to introduce **non-linearity** into the model.
- **Universal approximator**: Fully connected layers can theoretically approximate any continuous function, given a sufficiently large number of neurons and layers.

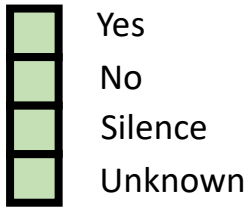






Convolutional Neural Networks

Output Layer

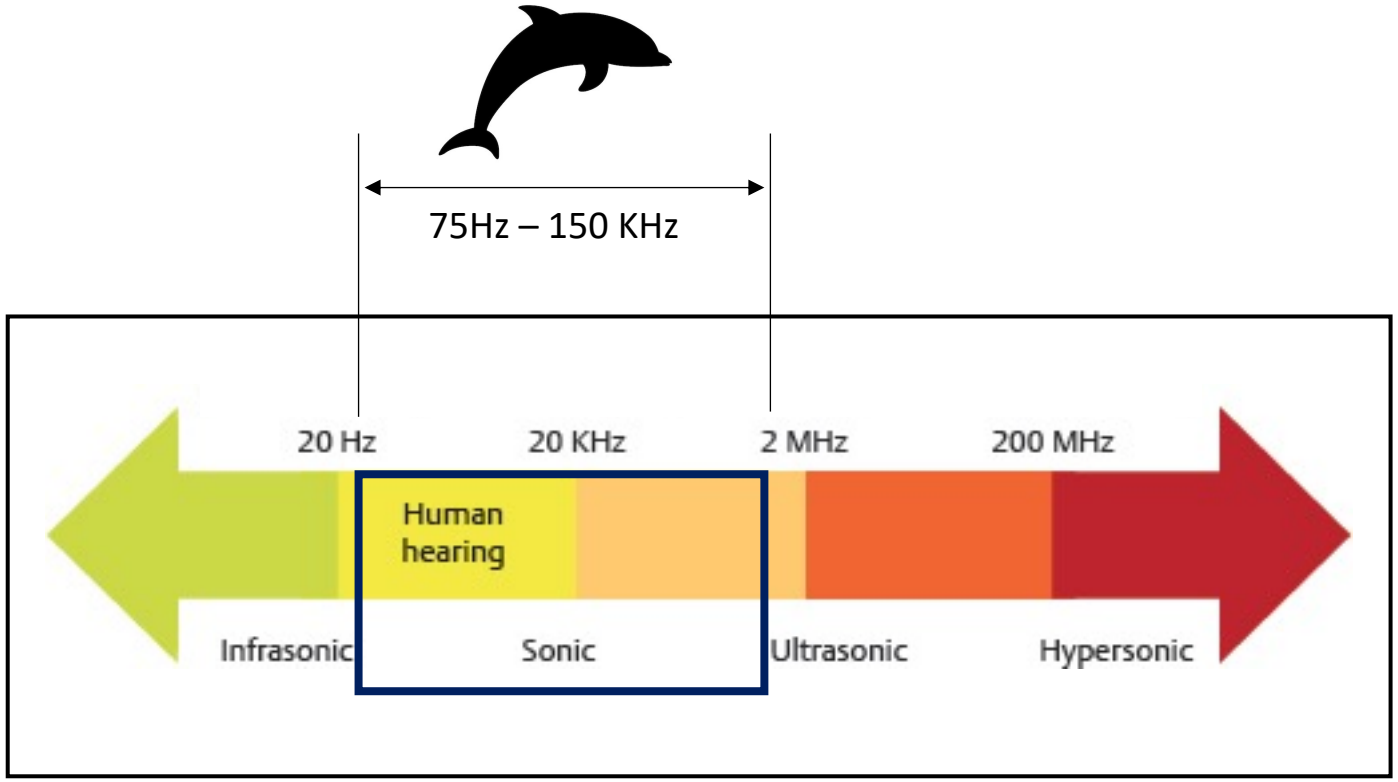


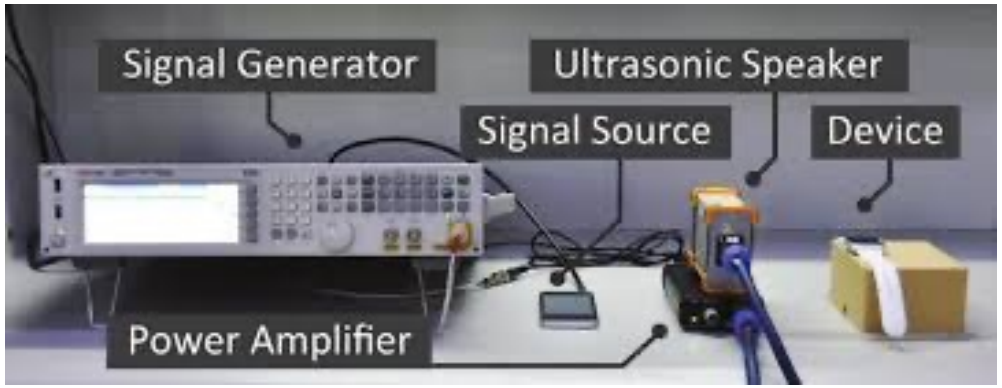
- It is also a fully connected layer, represents the final output classifications of the network
- Usually, softmax is used. WHY?
- The term "softmax" derives from the amplifying effects of the exponential on any maxima in the input vector.
- The standard (unit) softmax function σ: $\mathbb{R}^K \to (0,1)^K$ is defined when $K \geq 1$ by the formula

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \ldots, K \text{ and } \boldsymbol{z} = (z_1, \ldots, z_k) \in \mathbb{R}^K$$

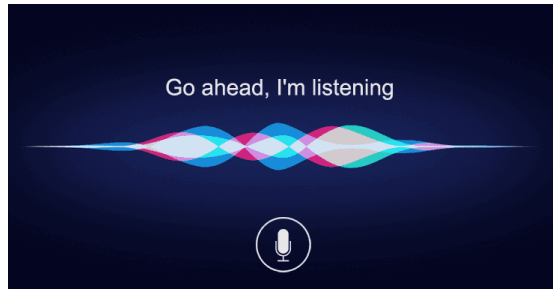
Dolphin Attack: High Frequency Voice Commands

Dolphin Attack: A attack that modulates voice commands on ultrasonic carriers (f > 20 kHz) which is out of range of human hearing but triggers speech recognition systems.





Hardware setup



Source: https://arxiv.org/pdf/1708.09537.pdf

Dolphin Attack - Steps

- The attacker listens to Jim's normal conversation and record the audio.
- Jim says: "I met my friend yesterday. He was in the city carrying a cake".
- The attacker extracts syllables and jumbles them.

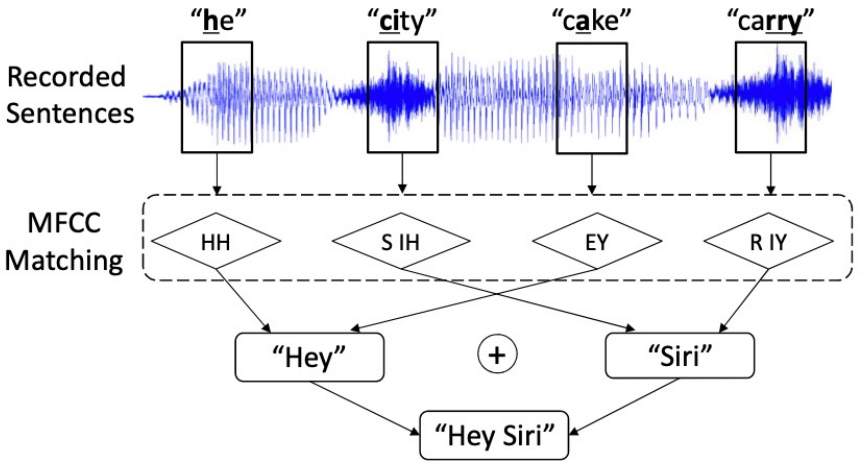
I met my friend yesterday. He was in the city carrying a cake.

Ha cirry

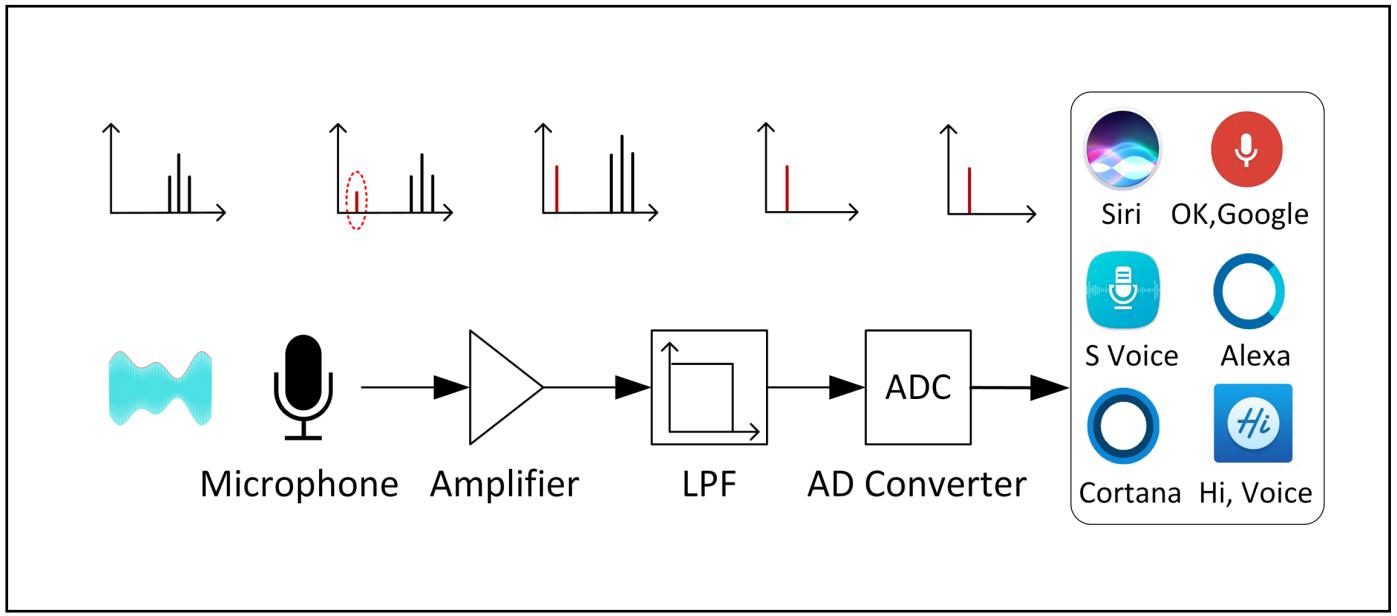
"Hey Siri"

Dolphin Attack - Steps

Voice activation command generation: The MFCC feature for each segment in a recorded sentence is calculated and compared and the matched voice segments are shuffled and concatenated in a right order.



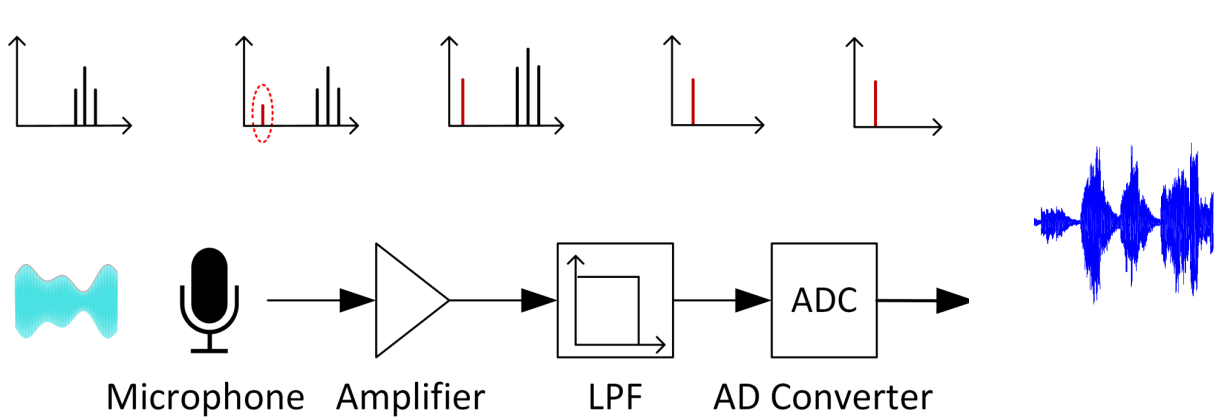
Attack: Voice command generation

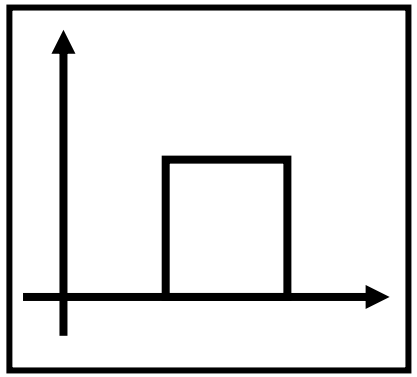


Attack: High frequency signal generation

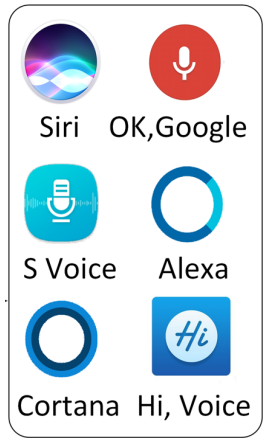
Source: https://arxiv.org/pdf/1708.09537.pdf

Thwarting the Dolphin Attack



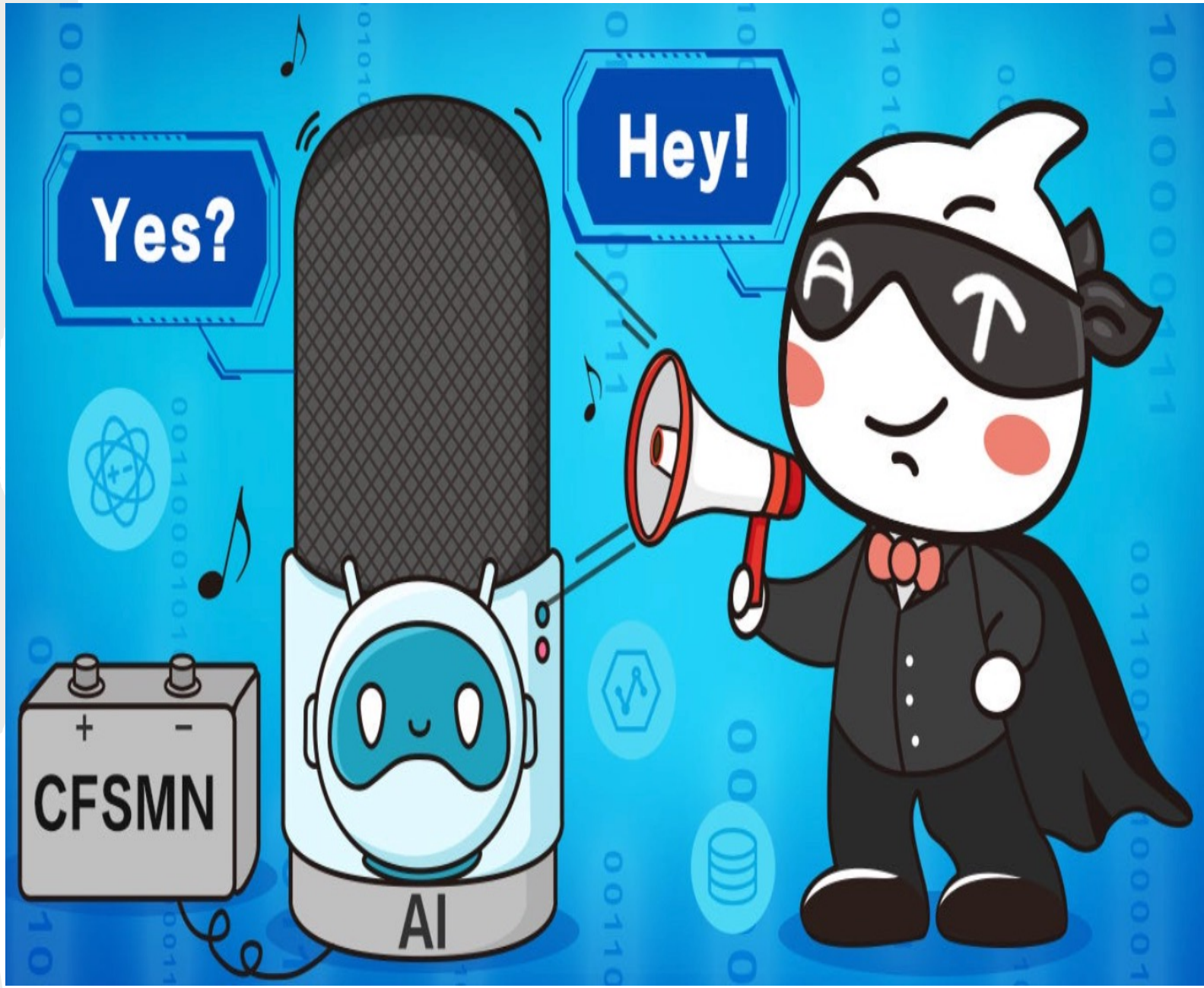


Band Pass Filter



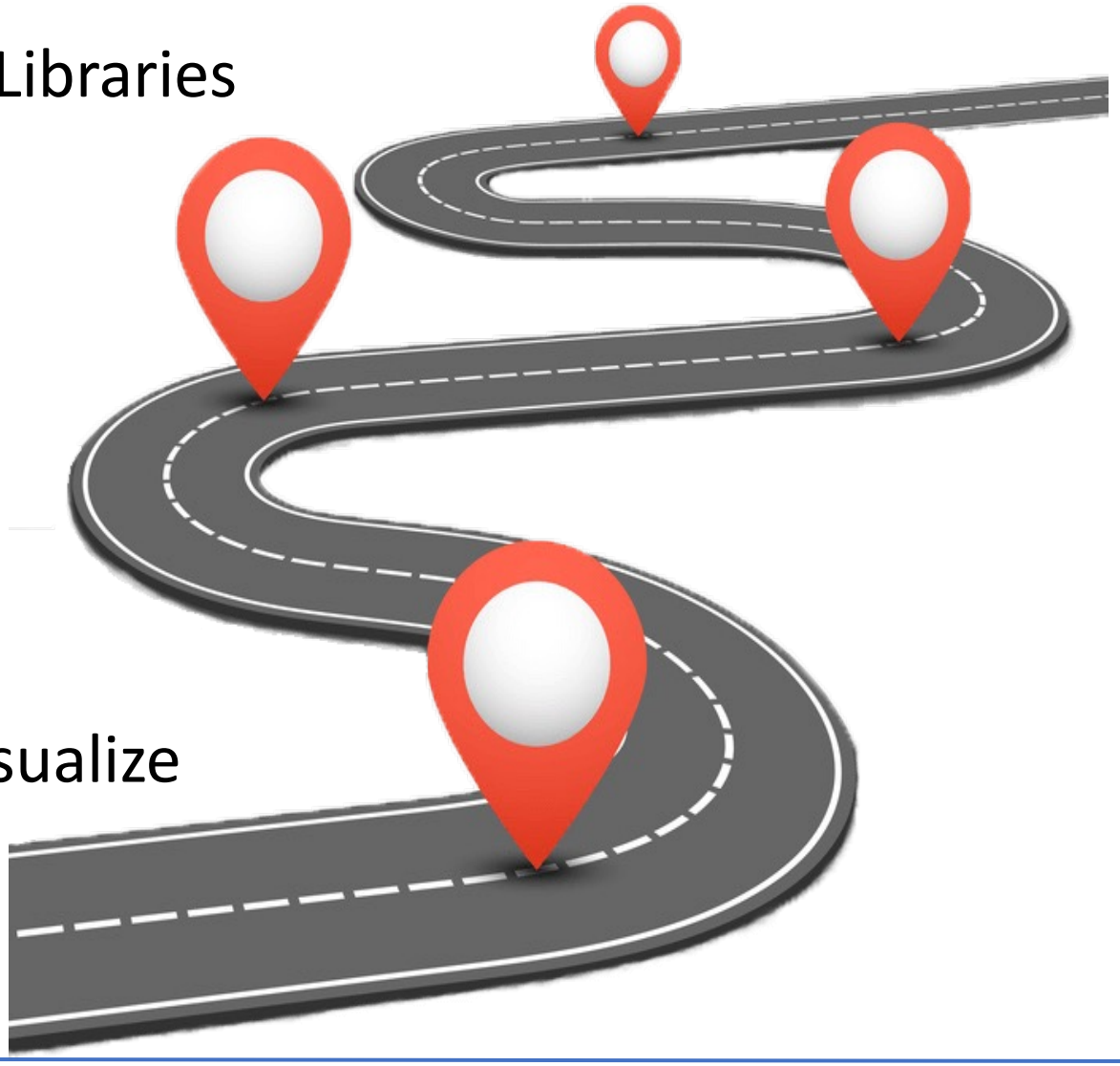
Today's Lab

- Both Software and Hardware Lab!
- Loading, Inspecting, and Visualizing Audio Data
- Computing FFT, Spectrogram, and MFCC of Audio Samples
- ML Models for KWS
- Implement the Model with TinyML Kits



Lab 3 - Software - Road Map

1. Installing and Configuring Needed Python Libraries
2. Define Python Audio Importing Function
3. Load in the Audio Samples
4. Visualize the Audio Samples as Signals
5. Compute the FFT of Audio Samples and Visualize
6. Convert FFT to Spectrogram and Visualize



Lab 3 - Software - Road Map

7. Convert Spectrogram to MFCC and and Visualize

8. Import and Configure Tensorflow Libraries

9. Building/Loading ML model for KWS

10. Training ML model for KWS

11. Evaluation of ML Model Performance

Lab 3 - Hardware - Road Map

1. Train a TensorFlow Lite model
2. Convert the TensorFlow Lite model to the TinyML model
3. Upload the TinyML model to boards

Lab 3 - Analyzing Audio Signals

- Using your laptop's audio recorder, record a few audio samples
- Run the Python code to convert the recorded signal to the frequency domain.

Here are some words you can try:

1. Hello
2. Hello Hello
3. Hello Hello Hello
4. Yes
5. Alexa
6. Alexa Alexa Alexa
7. Ok Google

Lab 3 – Software

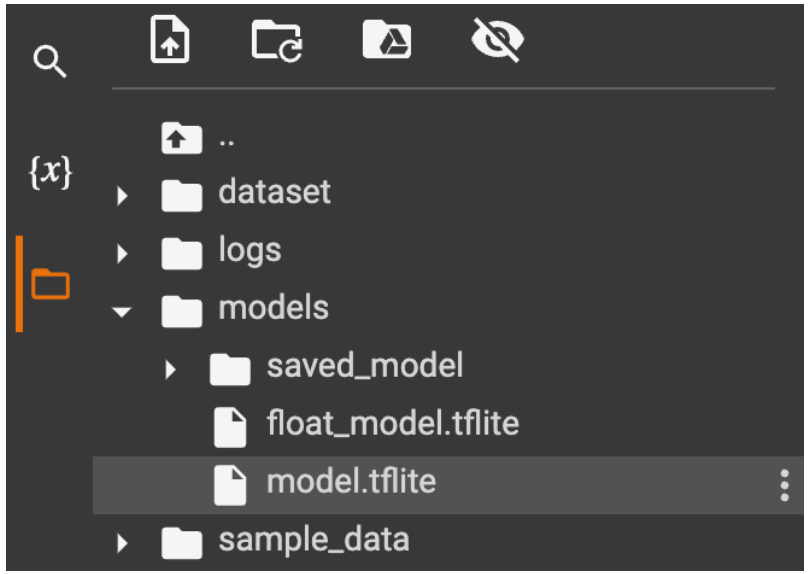
We will use Google Colab for this Lab.
Website: colab.research.google.com
(You can login in using your own Google account or UW's Google account)

Simply open the file Lab3_KeywordSpotting.ipynb in Colab, and run the cells!

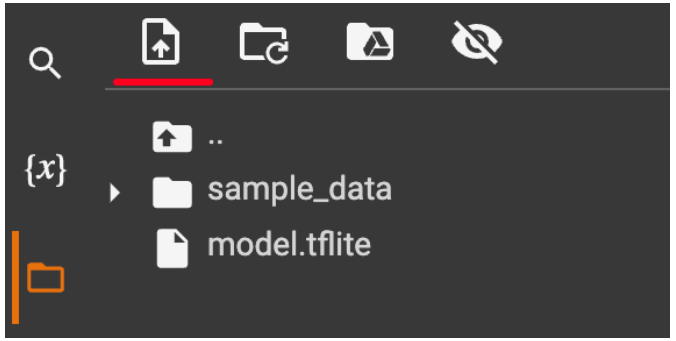
Lab 3 – Hardware

1. Download model.tflite (quantized model) from Google Colab.

2. Open EEP595-TinyML-Lab3 (Hardware).ipynb and upload model.tflite



EEP595-TinyML-Lab3 (Software)



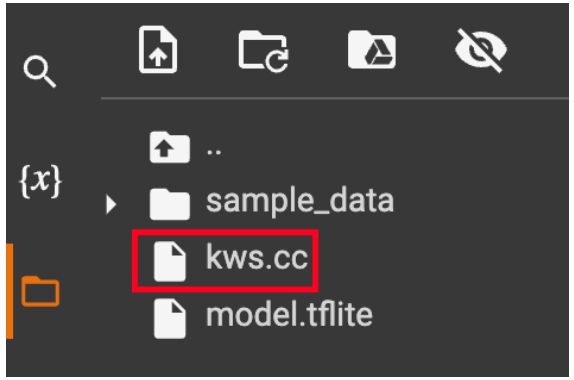
EEP595-TinyML-Lab3 (Hardware)

Lab 3 – Hardware

3. Now run this cell

```
MODEL_TFLITE = 'model.tflite' #enter the name of your TFlite file uploaded to the folders section
MODEL_TFLITE_MICRO = 'kws.cc' #update the name of your .cc file (This can be anything)
!xxd -i {MODEL_TFLITE} > {MODEL_TFLITE_MICRO}
REPLACE_TEXT = MODEL_TFLITE.replace('/', '_').replace('.', '_')
!sed -i 's/'{REPLACE_TEXT}'/g_model/g' {MODEL_TFLITE_MICRO}
```

After running, you can see kws.cc in the sidebar. This is the supported model type for TinyML.



Lab 3 – Hardware

4. Now run this cell

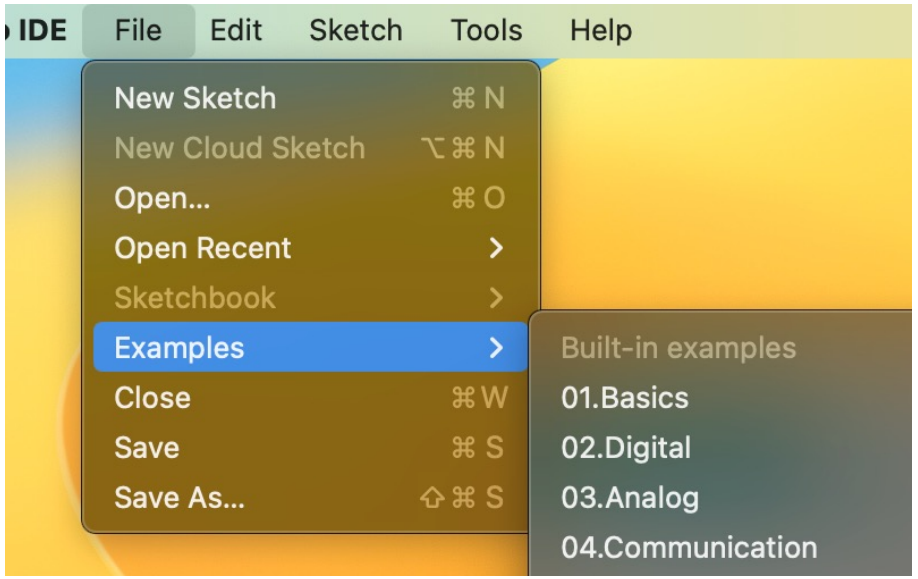
```
!cat {MODEL_TFLITE_MICRO}
```

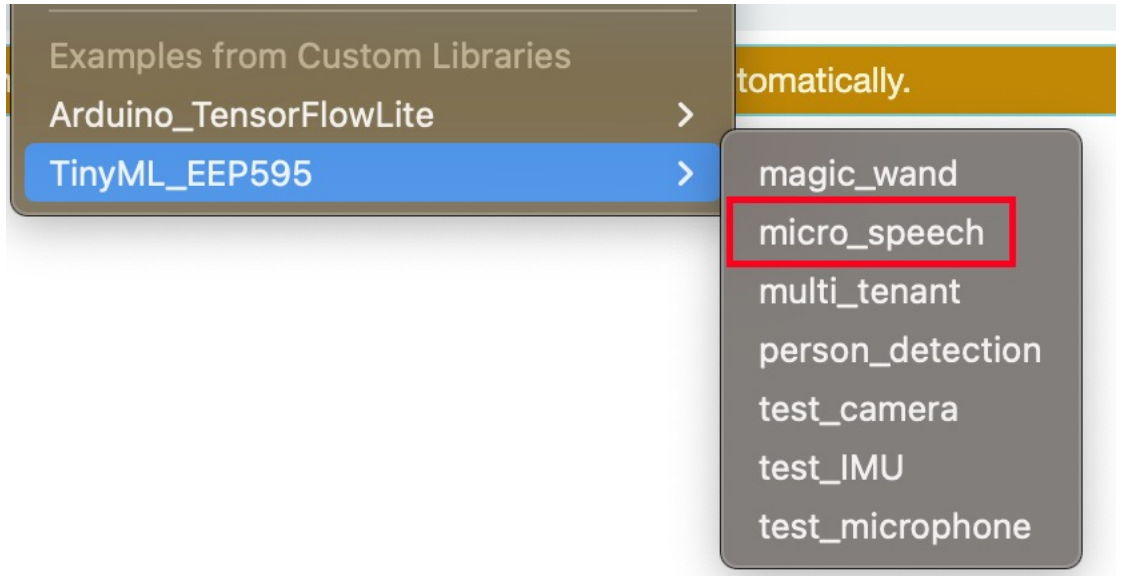
```
0xe7, 0xd6, 0x13, 0xe3, 0x30, 0x09, 0x00, 0xf5, 0xe0, 0xf3, 0x11, 0xe2,
0x38, 0x0d, 0xf6, 0x05, 0xec, 0x05, 0x00, 0xe5, 0x24, 0xef, 0xfe, 0xf8,
0x00, 0xd8, 0x18, 0xf1, 0x26, 0x0b, 0xf2, 0xfc, 0xe0, 0xe4, 0x06, 0x0b,
0x1a, 0x05, 0xc6, 0xf6, 0xe8, 0xde, 0xfe, 0x0c, 0x03, 0x09, 0xfe, 0xe2,
0x18, 0x1b, 0xfb, 0xf7, 0x06, 0xf1, 0xfe, 0xf6, 0xef, 0x1b, 0x07, 0x0d,
0x01, 0x0a, 0xed, 0xf0, 0xad, 0x1a, 0x17, 0xd6, 0x37, 0xfd, 0xd8, 0xec,
0xca, 0xf1, 0x15, 0xc4, 0x33, 0xf1, 0xed, 0xf0, 0xe9, 0x15, 0x0d, 0xf2,
```

You will covert .cc models to Hex values. We will use it in Arduino IDE.

Lab 3 – Hardware

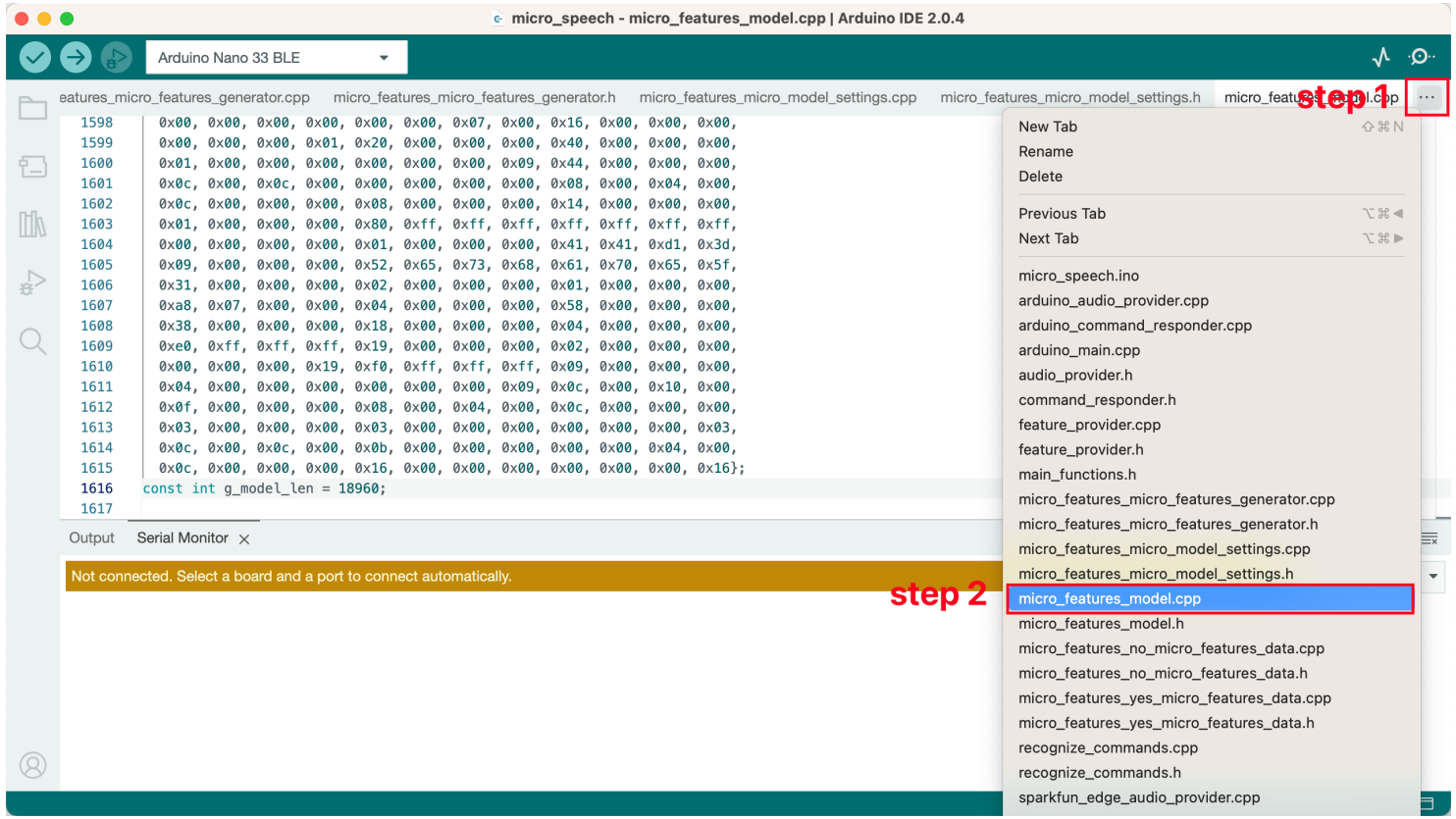
5. Open Arduino IDE and open TinyML_EEP595 library -> micro_speech example.





Lab 3 – Hardware

6. Click … on the top bar and open micro_features_model.cpp



Lab 3 – Hardware

7. Replace Hex in your file with your generated Hex from Step 4.

Don’t forget to replace g_model_len with what you get in Step 4.

```
  0x0c, 0x00, 0x00, 0x00, 0x16, 0x0
};
unsigned int g_model_len = 18960;
```

```
  0x0c, 0x00, 0x0c, 0x00, 0x0b, 0
  0x0c, 0x00, 0x00, 0x00, 0x16, 0
const int g_model_len = 18960;
```

Lab 3 – Hardware

8. Save files (You may save it to another place as examples are read-only).

And, upload it to Arduino Nano 33 BLE.

After pressing reset, you can now open serial monitor to see the KWS result !