

Model Optimization For Edge Devices

Adolf Anthony Dcosta • 24th April 2020

Committee Members

Dr. Carlos E. Otero – Major Advisor

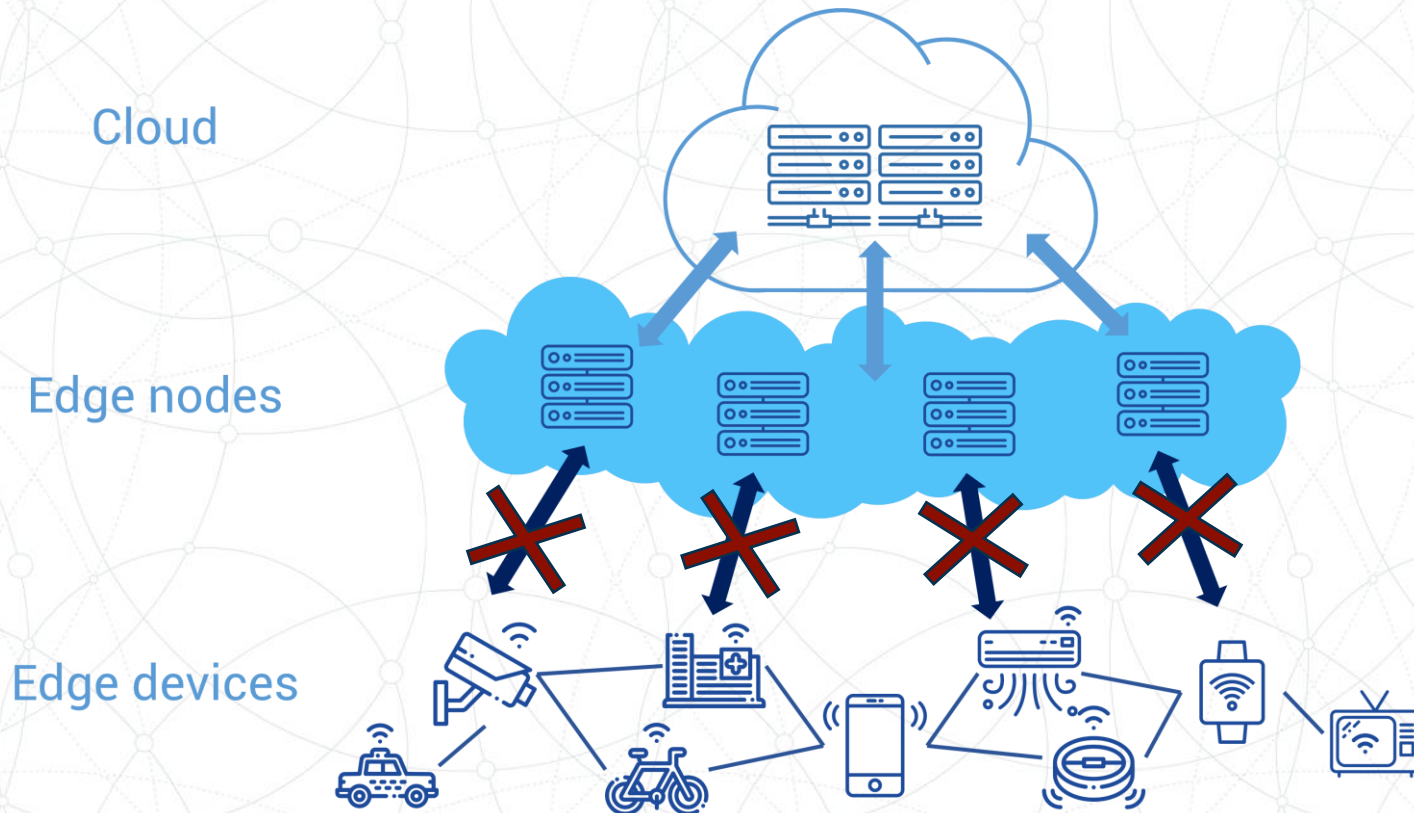
Dr. Veton Kepuska – Committee Member

Dr. William Allen – Committee Member

Thesis Agenda

- **What are edge devices ?**
- **Need of model optimization for edge devices ?**
- **Proposed Approach**
- **Pre-processing Technique**
- **Implementation**
- **Results**
- **Future Work**

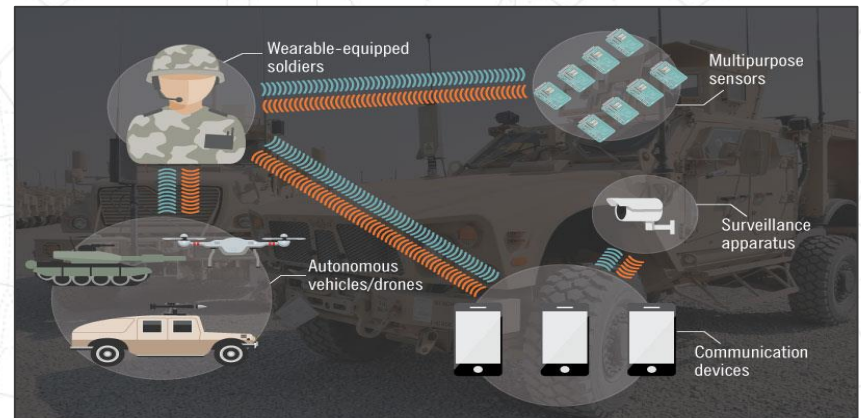
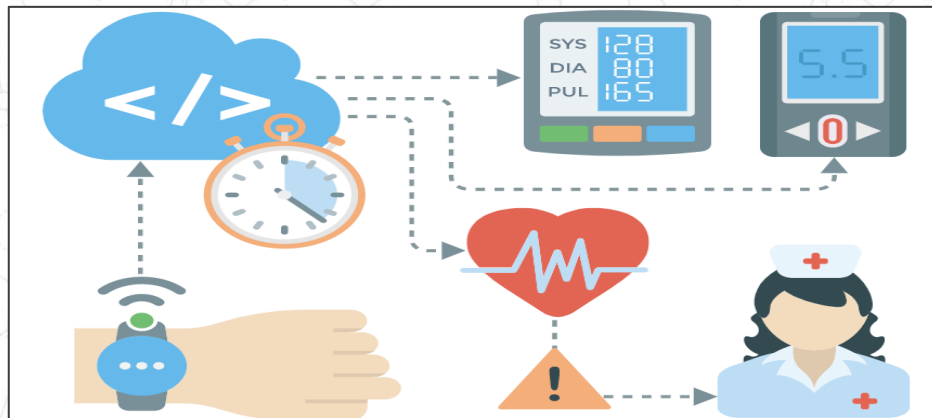
System Overview



Edge Computing – “Edge computing is a distributed computing paradigm which brings computation and data storage closer to the location where it is needed”.^[2]

Need for Optimization of Edge Devices

A study by IBM states Edge devices are now growing exponentially, expecting a staggering growth from 15 billion today to 150 billion by 2025^[1].



Images From
<https://www.redappletech.com/why-iot-smart-home-automation-is-in-demand>
<https://www.spec-india.com/blog/industrial-iiot-iiot>
<https://www.scnsoft.com/services/iiot/medical>
<https://www.theconvexlens.co/2017/06/12/internet-battlefield-things-iiot-war/>

Revolutionized Edge Devices



Google Coral



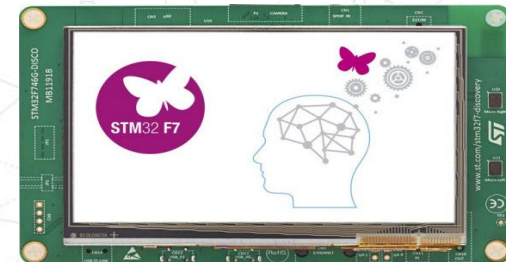
Nvidia Jetson Nano



Android Devices



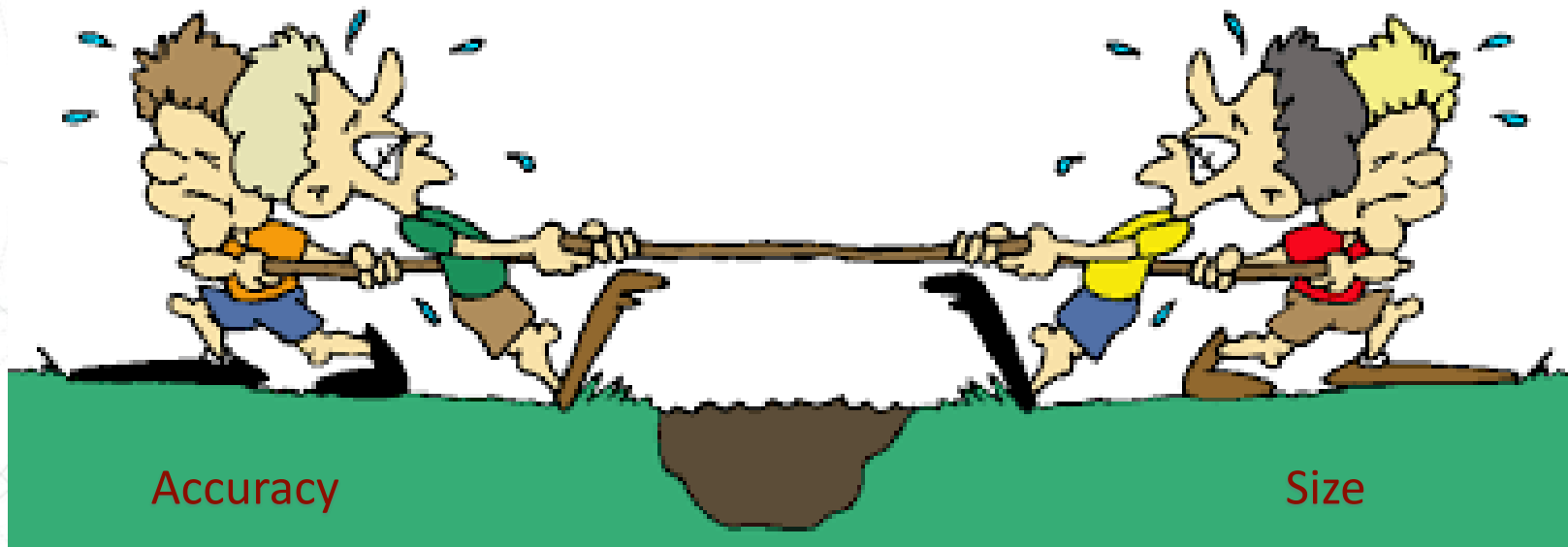
ARDUINO NANO 33 BLE SENSE



DiscoveryKit

Images From
<https://circuitdigest.com/microcontroller-projects/arduino-nano-33-ble-sense-board-review-and-getting-started-guide>
<https://www.hackster.io/news/say-hello-to-google-coral-cdbb49183864>
<https://www.t-mobile.com/cell-phone/google-pixel-4>
<https://www.pcworld.com/article/3368942/nvidia-99-jetson-nano-developer-kit.html>
<https://www.arrow.com/en/reference-designs/stm32f746g-disco-32f746gdiscovery-discovery-kit-with-stm32f746ng->

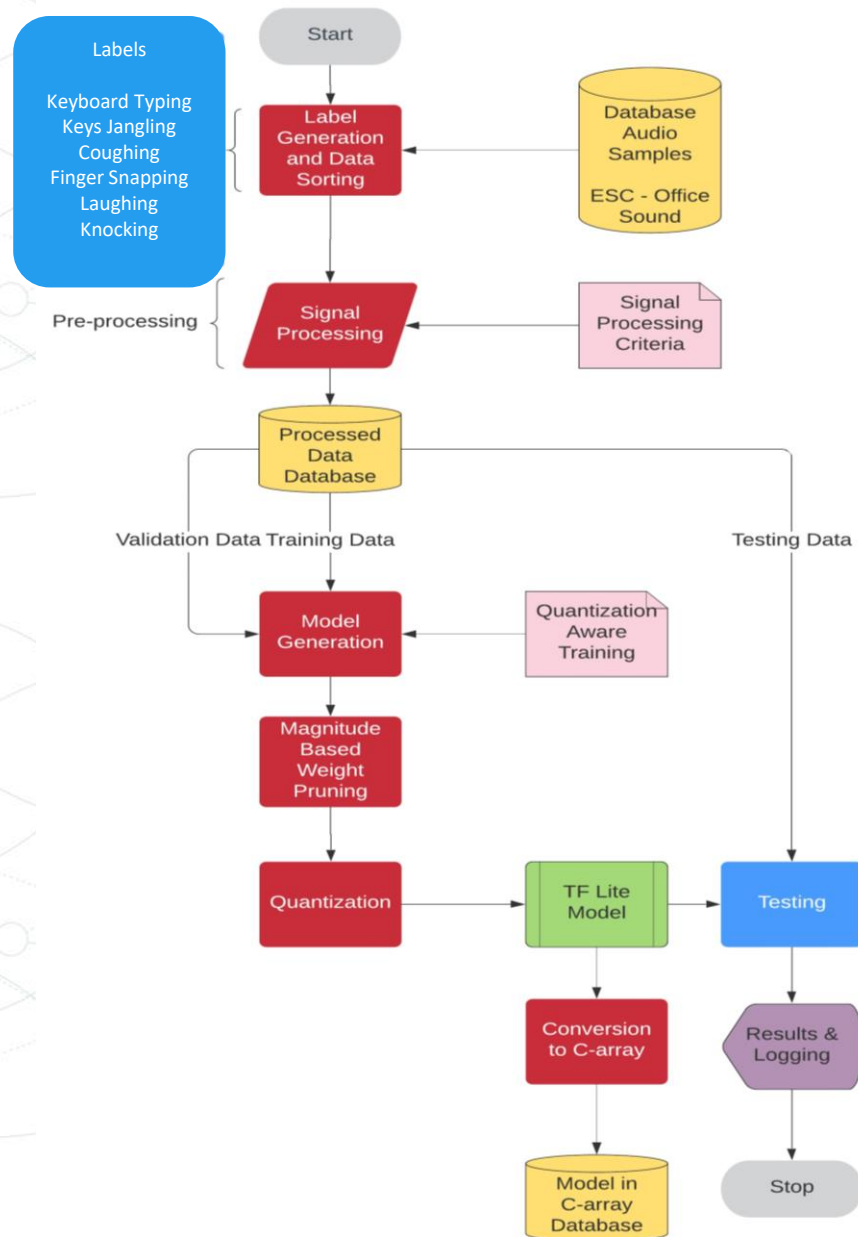
Goal of this Thesis



Trade-off between Accuracy and Size

Images From
<https://www.google.com/imgres?imgurl=https%3A%2F%2Fwikiclipart.com%2Fwp-content%2Fuploads%2F2017%2F03%2FTug-of-war-tug-war-free-download-clip-art-on-clipart.gif&imgrefurl=https%3A%2F%2Fwikiclipart.com%2Fbackupeverything.co.uk/technical-insight-into-data-compression/>

Proposed Approach



Data Set

Label	DCASE	ESC-50	Collected	Combined
Knocking	270	40	88	407
Laughing	290	40	68	398
Keyboard Typing	119	40	43	202
Coughing	243	40	3	286
Keys Jangling	99	40	7	146
Snap	77	40	52	169
Total	1107	240	261	1608

Images From
David Elliott, Evan Martino, Carlos E Otero, Anthony Smith, Adrian Peter, Benjamin Luchterhand, Eric Lam, and
Steven Leung. Cyber-physical analytics: Environmental sound classification at the edge.



Implementation

Pre-processing

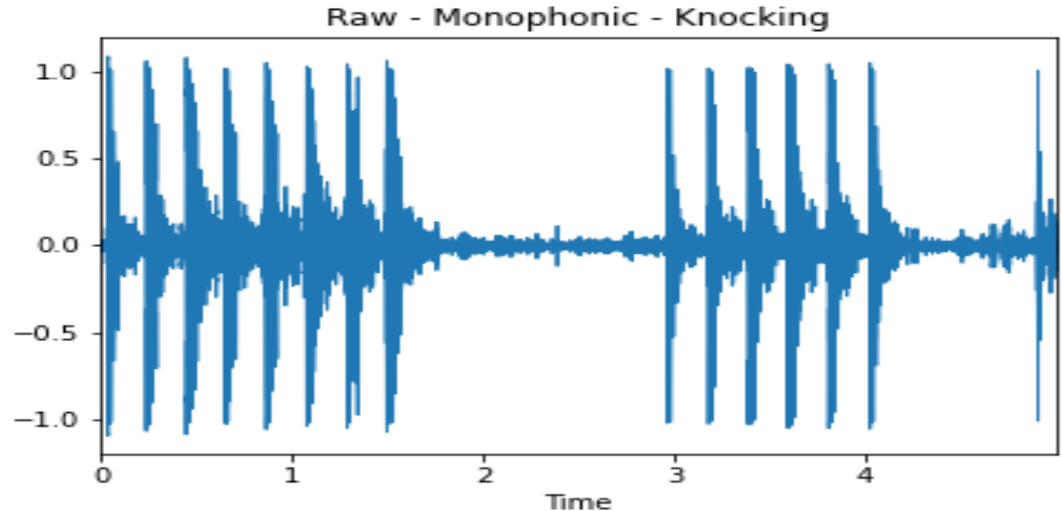
Example taken for visual representation - ESC-50 Knocking (1-81001-A-30.wav)



Visualization Parameters

- **Sample Rate - 16000**
- **Window Size - 3 Seconds**
- **Window Hop - 1.5 Seconds**
- **Channel - 1 (Mono)**

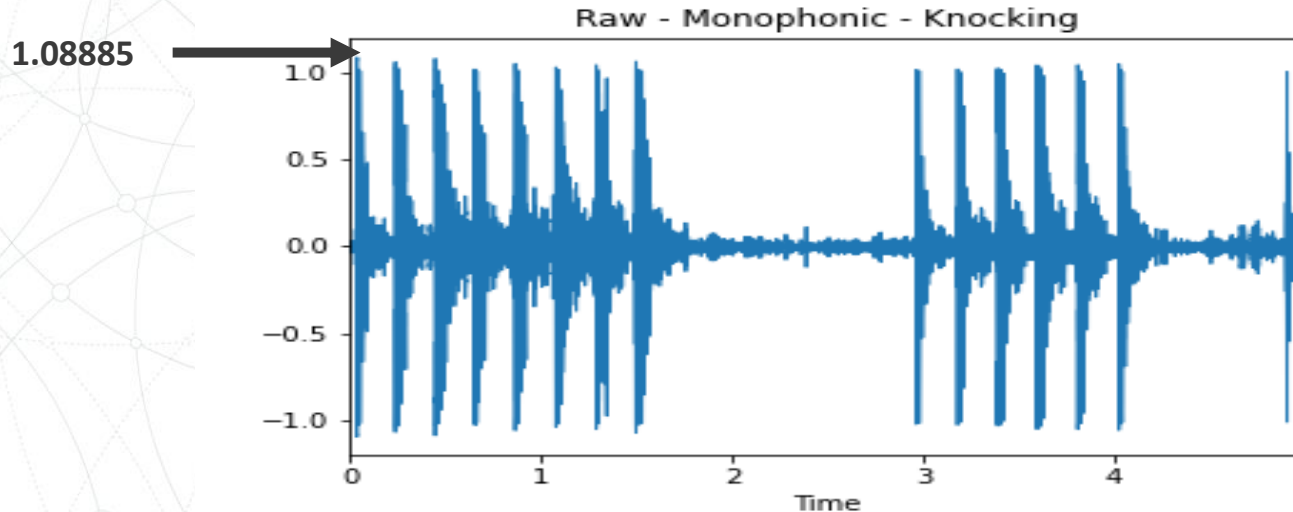
Visual Representation



Window Hop = (0.5 x Window Size) ^[3]

Implementation

Pre-processing



Mean calculated by the system (μ) = -0.0008

Standard Deviation calculated by the system (σ) = 0.2161

Raw Data (x) = 1.08885

Standard Scoring Normalization

$$z = \frac{x - \mu}{\sigma}$$

x = Raw data

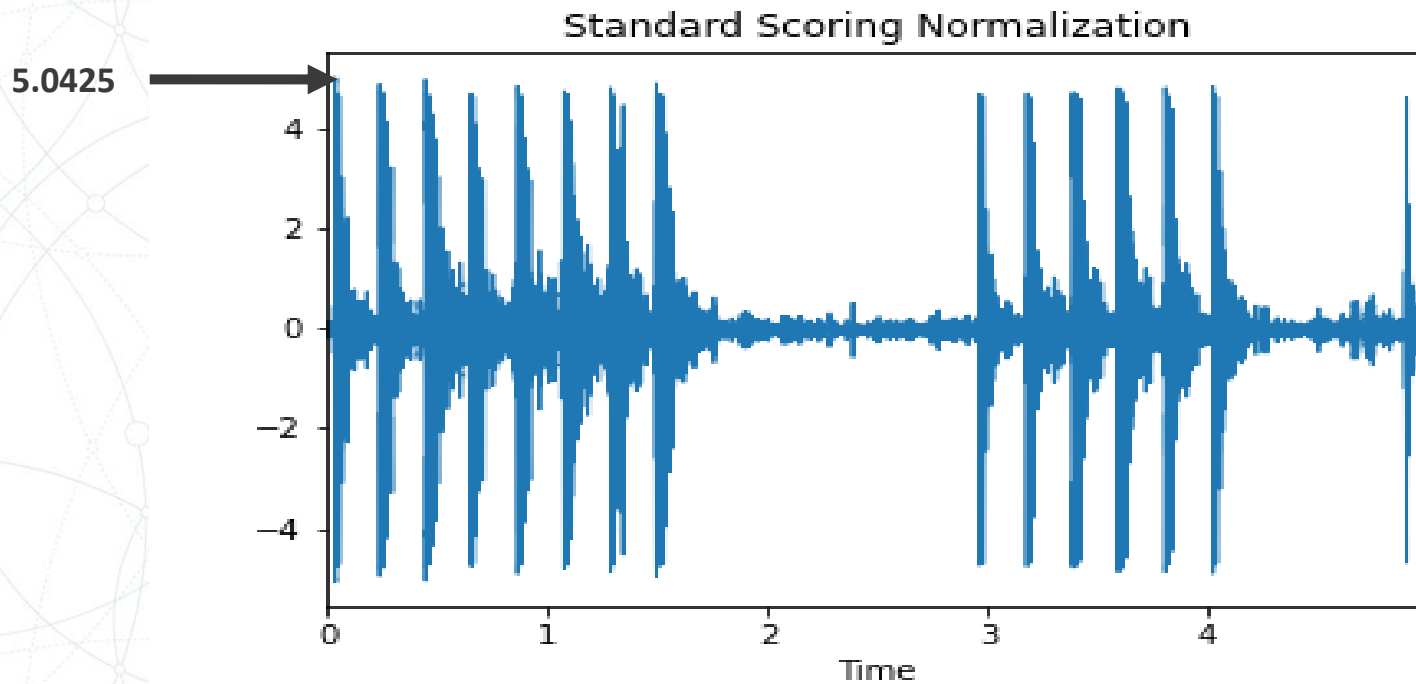
μ = Mean

σ = Standard Deviation

z = Standard Scoring Normalization

Implementation

Pre-processing

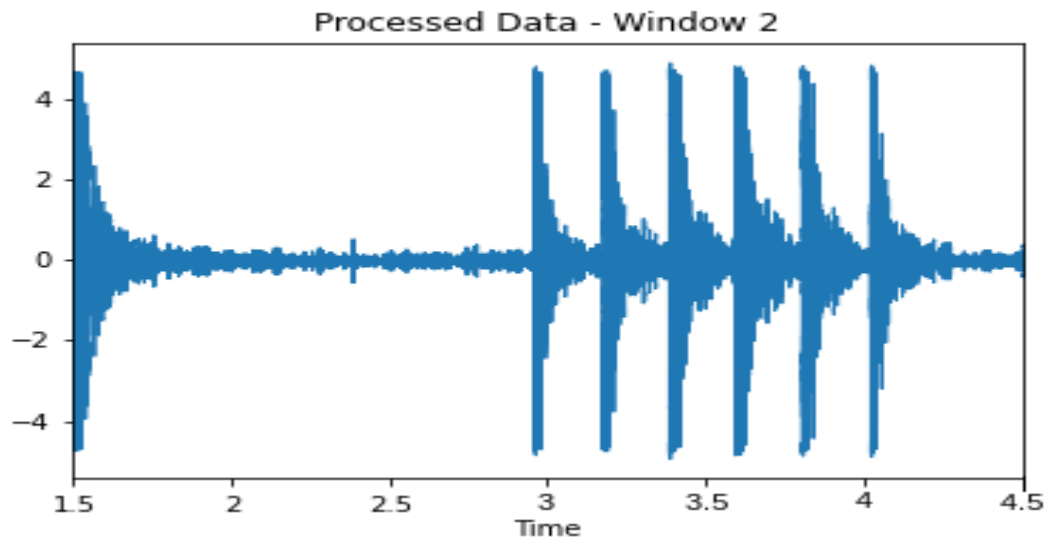
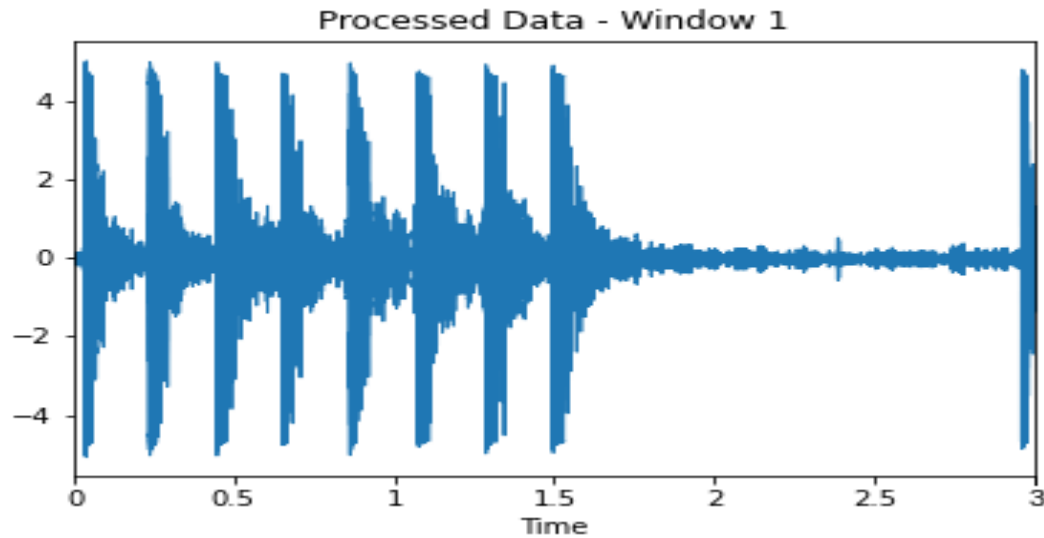


$$z = \frac{x - \mu}{\sigma} = \frac{1.08885 - (-0.0008)}{0.2161} = 5.04258$$

Implementation

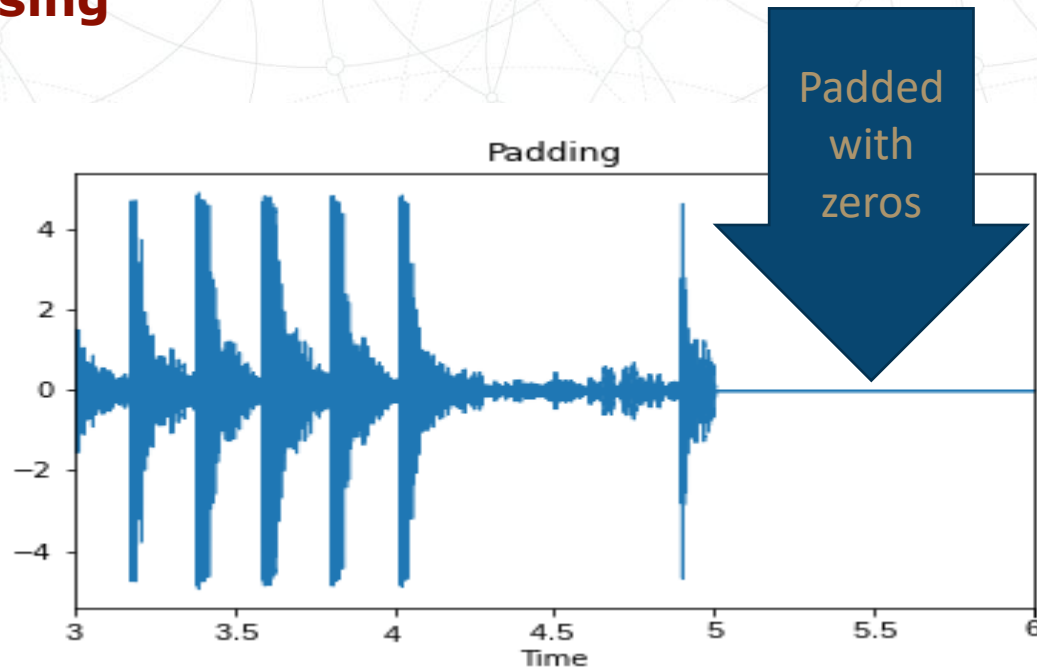
Pre-processing

- Window Size – 3 Seconds
- Window Hop – 1.5 Seconds



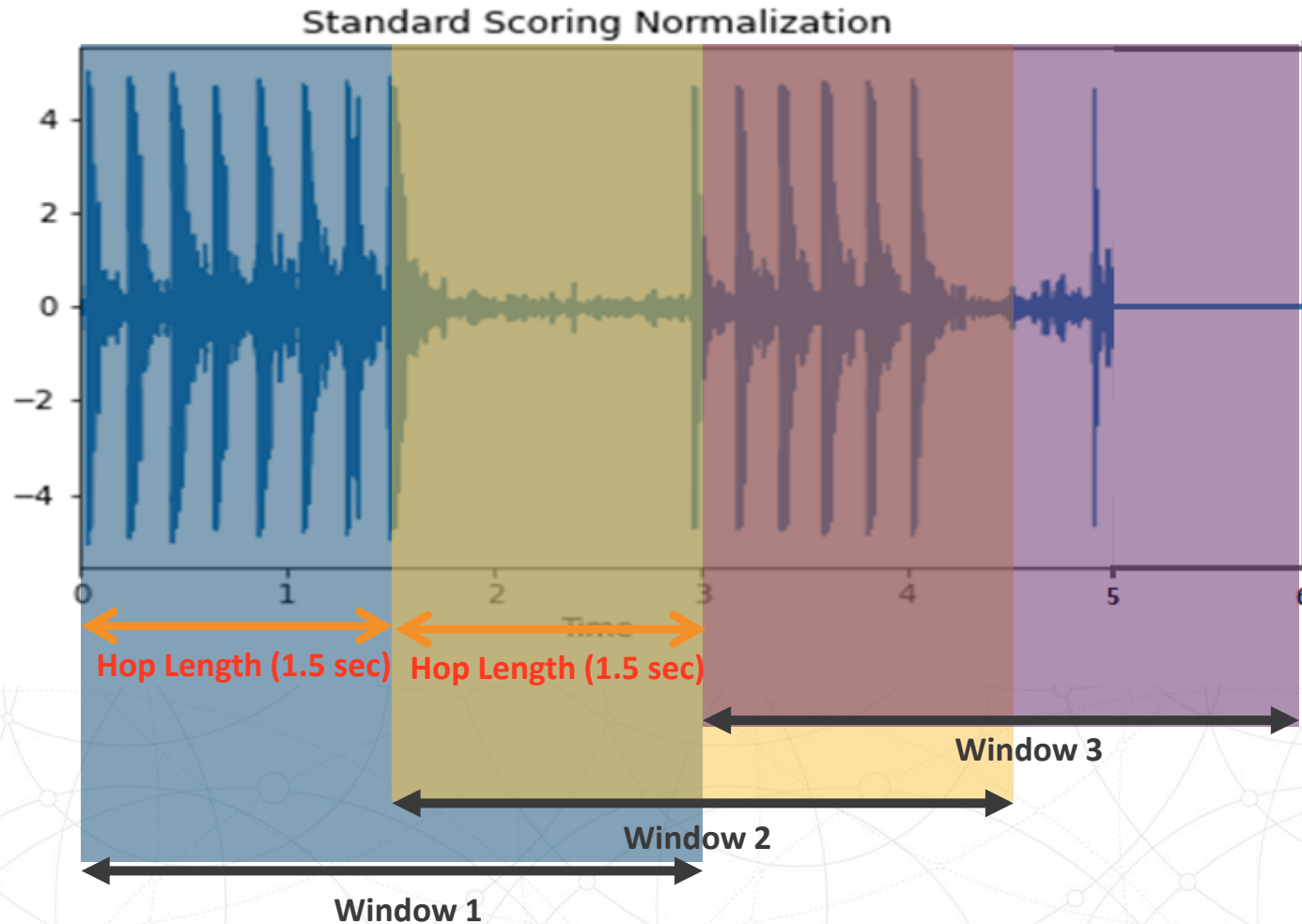
Implementation

Pre-processing



Padding Window: This is the final window. System pads zero to get fixed length.

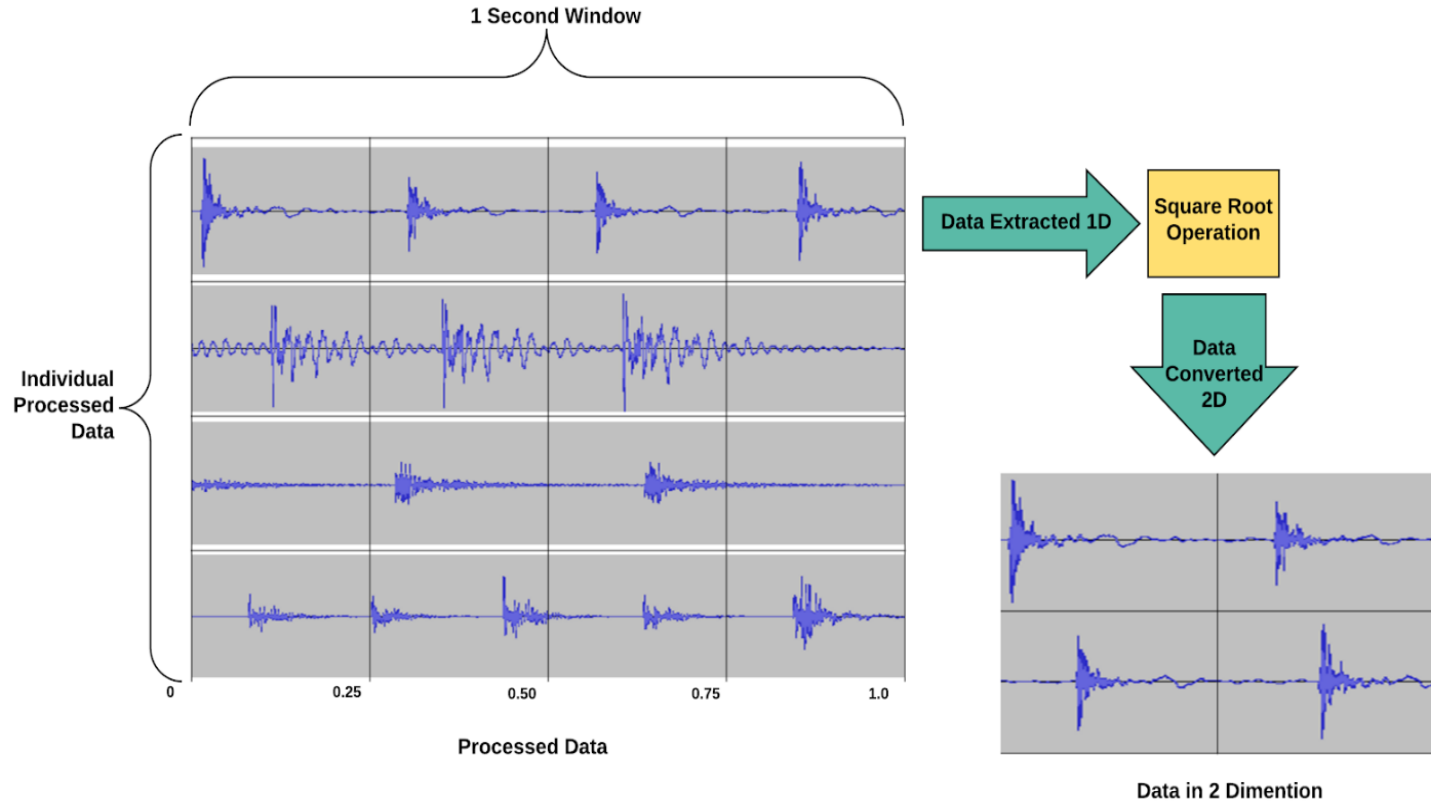
Implementation



LibROSA was python library used for sound processing.

Implementation

1D to 2D Conversion



Note: Square Root Operation is performed on NumPy array shape

Implementation

Actual Parameters Used in Our System for Model Design and Pre- processing

- **Sample Rate - 30000**
- **Window Size – 1 Seconds**
- **Window Hop – 0.5 Seconds**
- **Epoch – 100**
- **Pruning Epochs – 25**
- **Batch Size – 32 (No. of clips fed to model at once)**
- **Number of Classes – 6**
- **Channel – 1 (Mono)**

Number of Classes = 6

Keyboard Typing
Keys Jangling
Coughing
Finger Snapping
Knocking
Laughing

Implementation

Model Designing

Squeeze Net Strategies

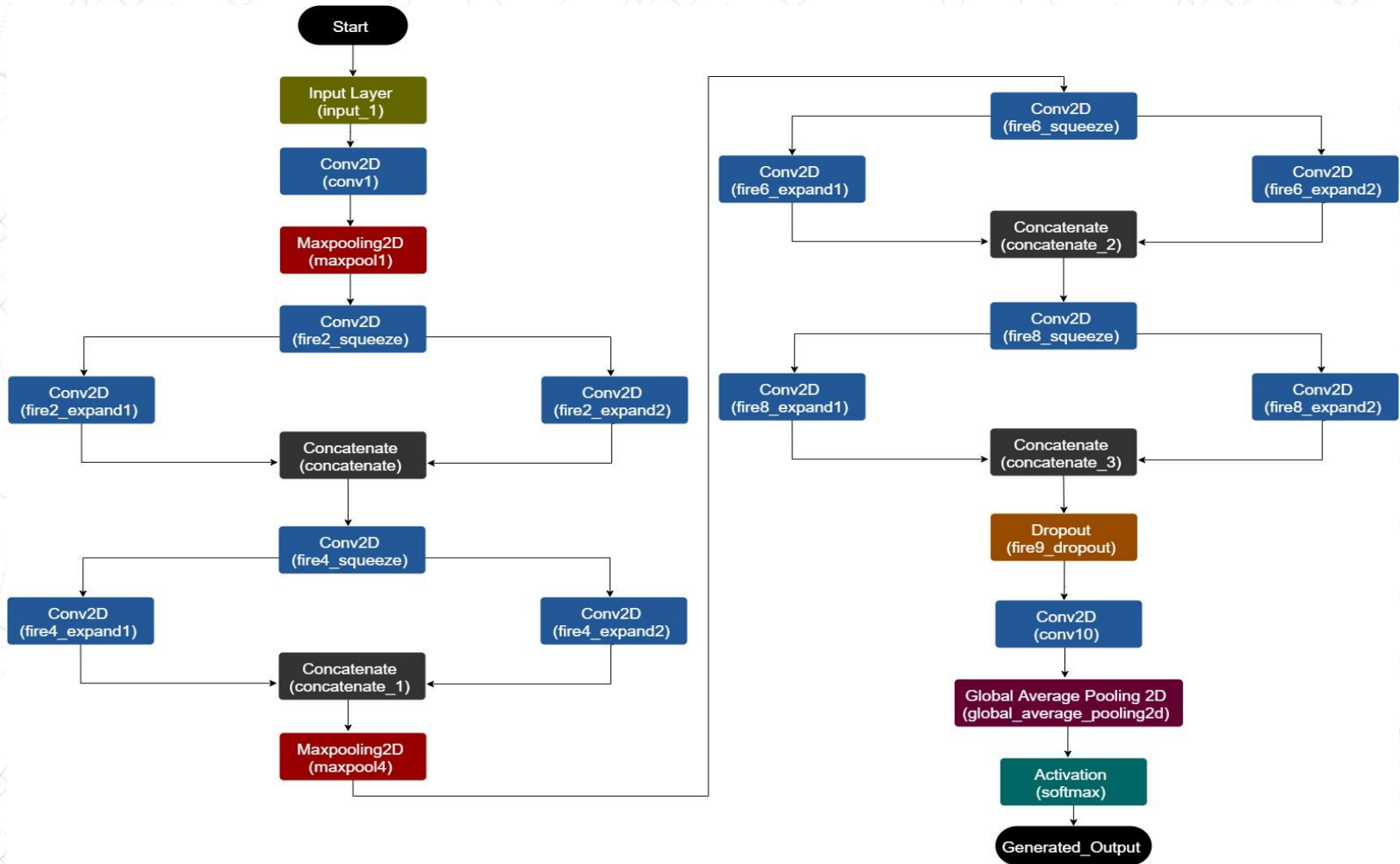
1. Replace 3X3 filters with 1x1 - Choose 1x1 convolution filters over 3x3 because of 9X fewer parameters.
2. Decrease the number of input channels to 3X3 filter - Design a layer that holds few parameters. Fewer parameters lead to smaller model size.

Total parameters = (Number of Input Channels) \times (Number of Filters) \times (Size of Filter)

3. Down sampling late into the network so that convolution layers have large activation map

Implementation

Model Architecture



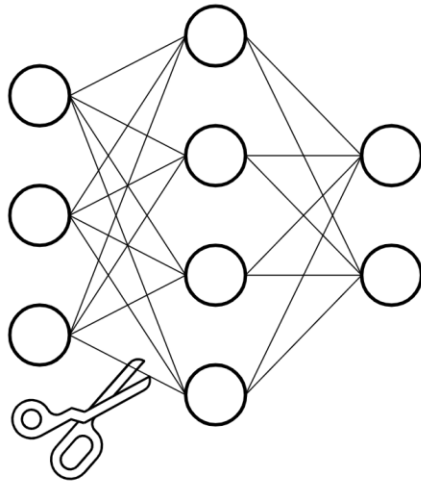
Implementation

Quantization Aware Training

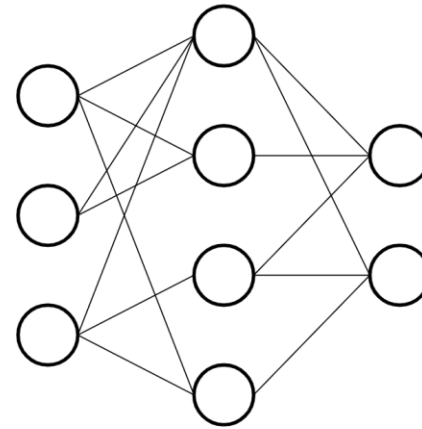
- **This process introduced fake nodes to the architecture during inference.**
- **Each floating-point number will be mapped to one low precision fake node.**
- **Values of the node is updated with high accuracy during the backward pass.**
- **This help to prevent the loss of accuracy during quantization.**

Implementation

Magnitude Based Weight Pruning



Before pruning



After pruning

- **Reduced the model size upto 5 times.**
- **Eliminates the unnecessary links in the weighted tensor**

Images From

<https://medium.com/tensorflow/tensorflow-model-optimization-toolkit-pruning-api-42cac9157a6a>

Implementation

Quantization

- **This method is applied using TensorFlow Lite**
- **It converts the 32-Bit Floating-Point Number to 8-Bit Integer**
- **It convert the entire model to a Flat Buffer**
- **Amount of accuracy lost at this step is less because of Quantization Aware Training Performed Initially.**
- **Reduced the Model size approximately 4 times.**

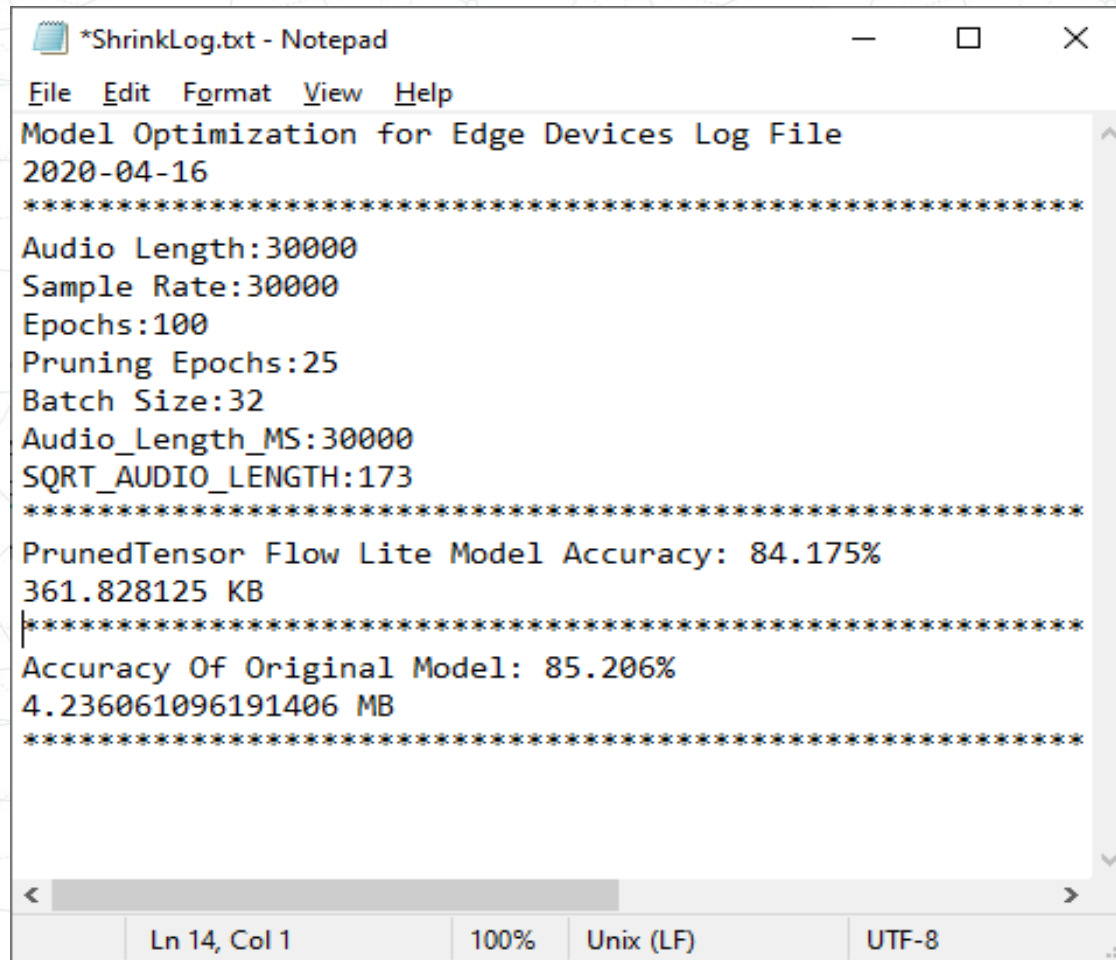
Implementation

Converting to C-Array

- **Converting from TensorFlow lite to C-array is essential, and the final step because most microcontrollers do not support native filesystem.**
- **The easiest way to use a model from your program is to include it as a C array and compile it into your program.**
- **For this we use a command in Linux “xxd -i ”**

Results

Log Generator Designed for the System



A screenshot of a Notepad window titled '*ShrinkLog.txt - Notepad'. The window displays a log file with the following content:

```
File Edit Format View Help
Model Optimization for Edge Devices Log File
2020-04-16
*****
Audio Length:30000
Sample Rate:30000
Epochs:100
Pruning Epochs:25
Batch Size:32
Audio_Length_MS:30000
SQRT_AUDIO_LENGTH:173
*****
PrunedTensor Flow Lite Model Accuracy: 84.175%
361.828125 KB
|*****
Accuracy Of Original Model: 85.206%
4.236061096191406 MB
*****
```

The status bar at the bottom of the Notepad window shows 'Ln 14, Col 1', '100%', 'Unix (LF)', and 'UTF-8'.

Results

Confusion Matrix

True	KT	403	6	2	7	6	3
	CO	9	217	2	2	13	49
	FS	3	7	80	2	4	1
	KJ	11	6	0	194	7	6
	KN	18	19	4	9	241	18
	LA	11	55	3	3	9	510
		KT	CO	FS	KJ	KN	LA
		Predicted					

TensorFlow Lite

KT : Keyboard Typing
CO : Coughing
FS : Finger Snapping
KJ : Keys Jangling
KN : Knocking
LA : Laughing

True	KT	405	5	3	7	5	2
	CO	9	218	2	2	11	50
	FS	2	7	80	2	5	1
	KJ	12	3	0	192	9	8
	KN	19	16	4	9	248	13
	LA	10	57	3	4	7	510
		KT	CO	FS	KJ	KN	LA
		Predicted					

Original



Results

Accuracy : Out of all the classes, how much we predicted correctly

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision: Out of all the positive classes we have predicted correctly, how many are actually positive.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Out of all the positive classes, how much we predicted correctly.

$$Recall = \frac{TP}{TP + FN}$$

F1 score: F-score helps to measure Recall and Precision at the same time.

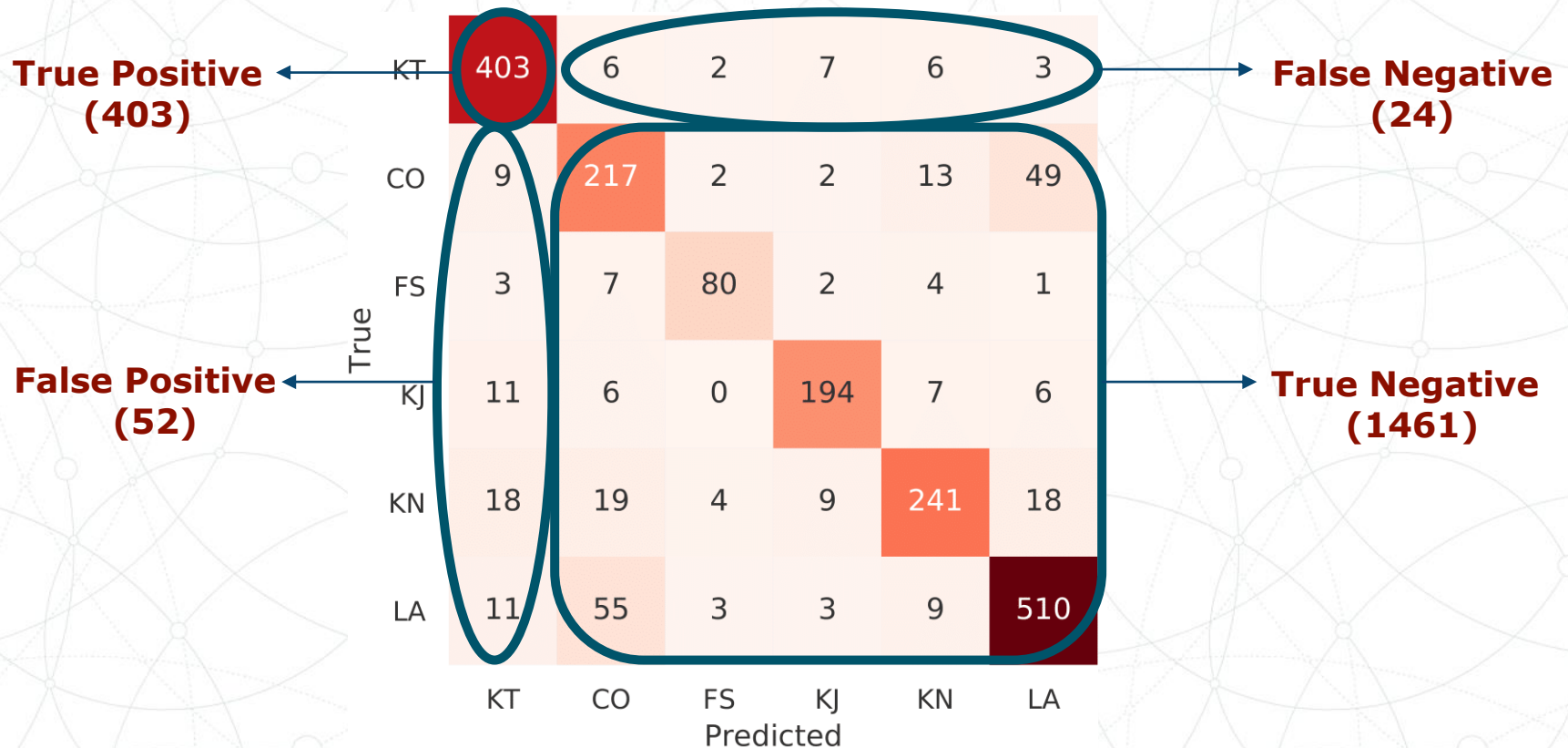
$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

Overall Accuracy

$$Overall Accuracy = \frac{\text{Correctly classified Values}}{\text{Total Number of Values}} \times 100$$

Results

Taking Example as KT: Keyboard Typing



KT : Keyboard Typing
CO : Coughing
FS : Finger Snapping
KJ : Keys Jangling
KN : Knocking
LA : Laughing



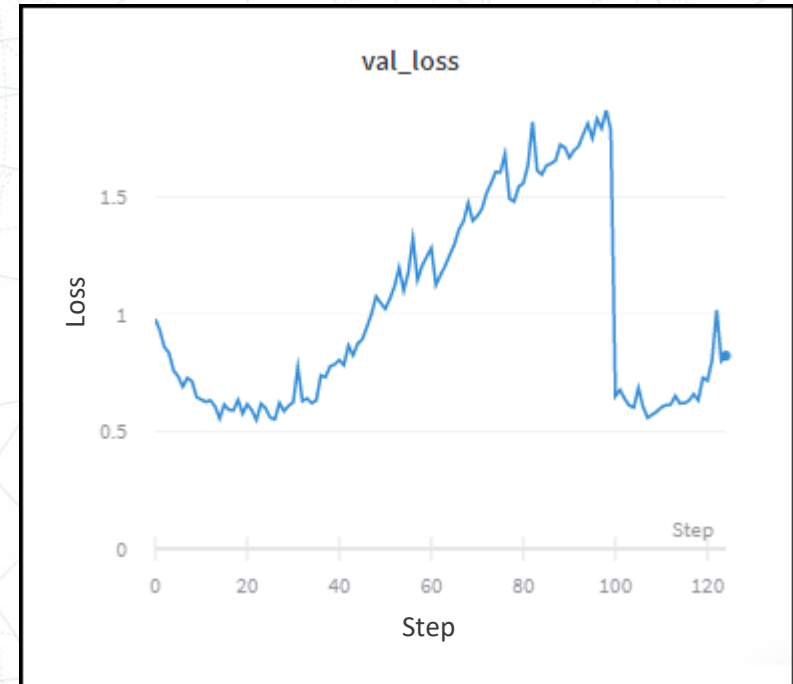
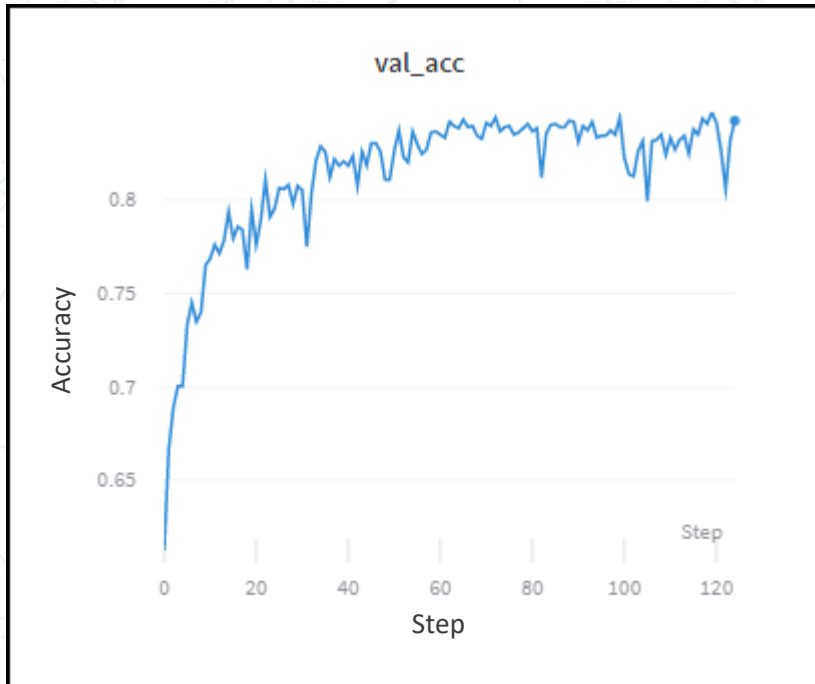
Results

Labels	Accuracy in %	Precision	Recall	F1 Score
Keyboard Typing	96.08	0.89	0.94	0.91
Coughing	91.34	0.70	0.74	0.72
Finger Snapping	98.56	0.88	0.82	0.85
Keys Jangling	97.27	0.89	0.87	0.88
Knocking	94.48	0.86	0.78	0.82
Laughing	91.8	0.87	0.86	0.87

Overall Model Accuracy = 84.79%

Results

Weight and Biases

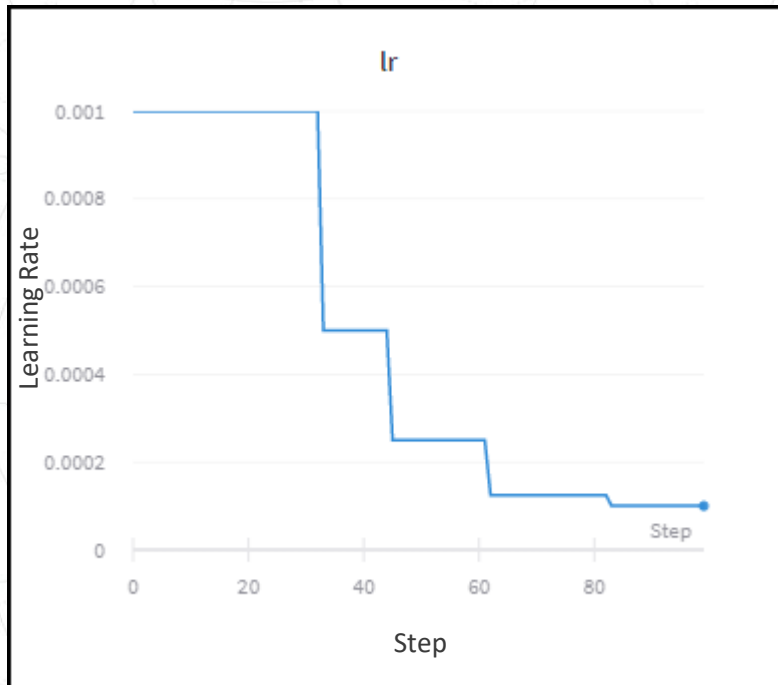


Validation Accuracy: Accuracy vs Number of Epochs. Max out at 85%

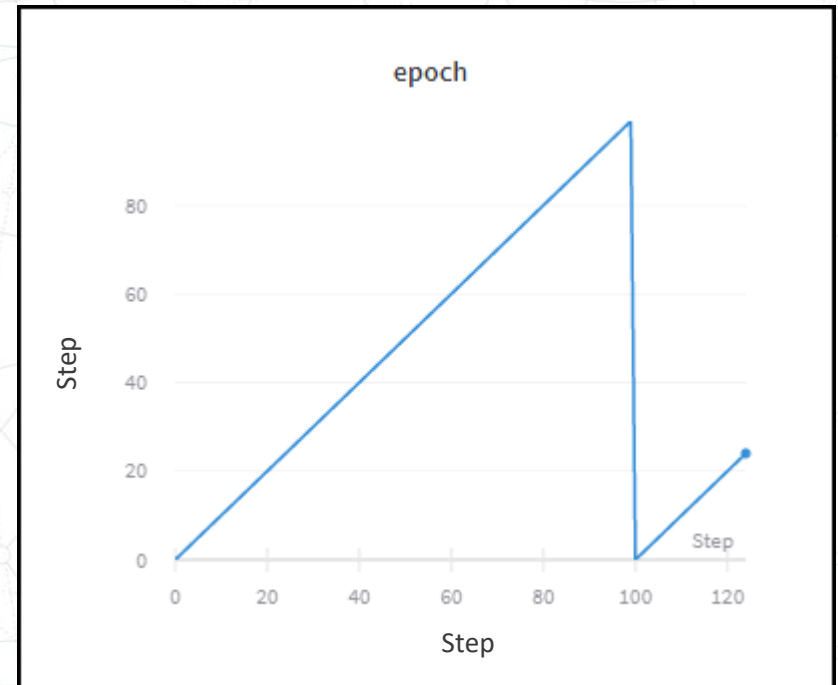
Validation Loss: Loss vs Number of Epochs.

Results

Weight and Biases



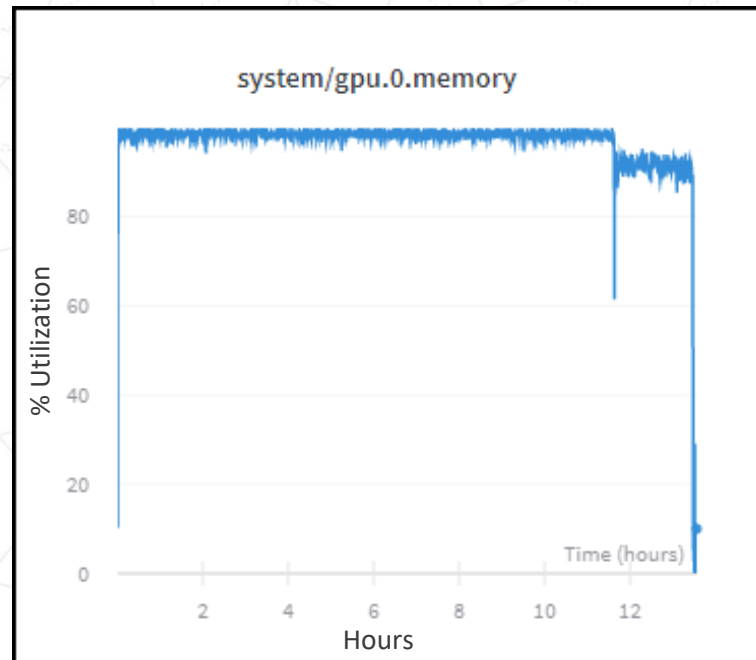
Learning rate: Rate of learning vs epochs



Epochs: Model training 100 epochs and Pruning 25 epochs

Results

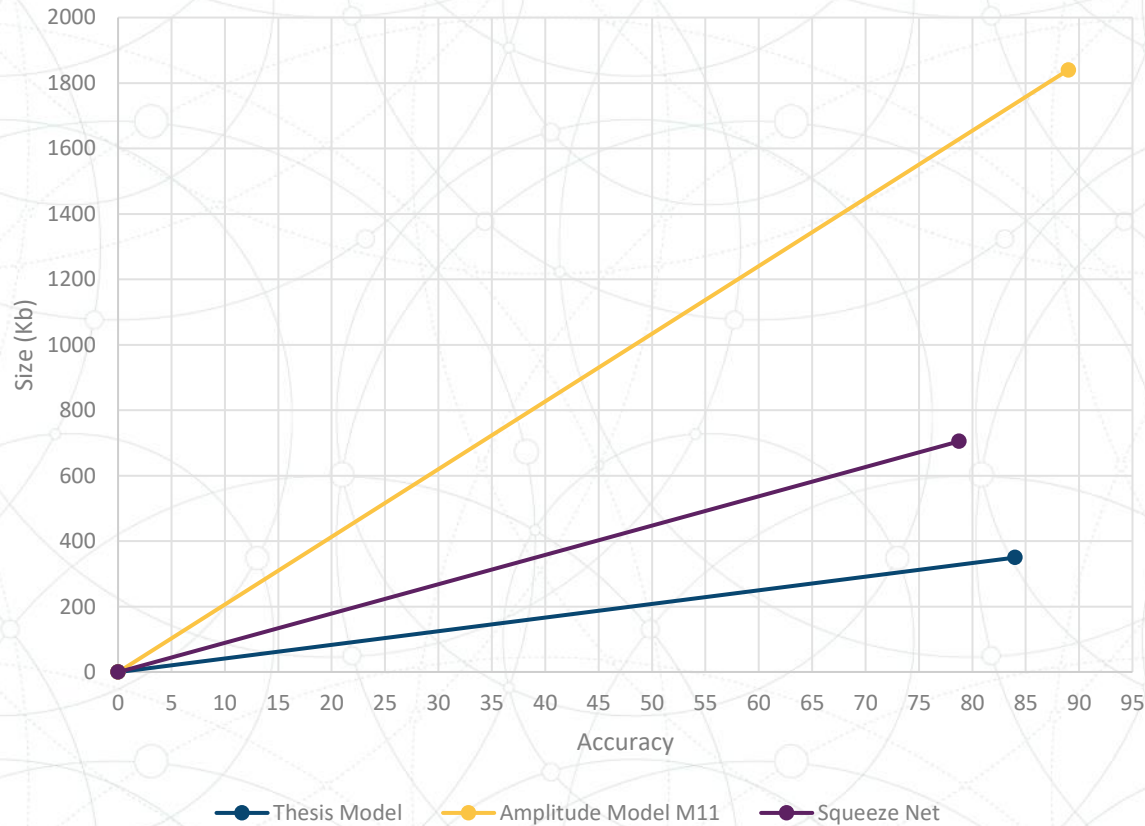
Weight and Biases



GPU Utilization : Model was trained for 13hrs graph represents memory utilization vs time

Results

Model Comparison



80% reduction in size with only 4% reduction in accuracy from amplitude model M11.

Future Work

- **Design a system that generates a model small enough for microcontrollers that can perform natural language processing.**
- **Improve on model size and accuracy.**
- **Reduce the latency of edge devices by inferring simpler preprocessing techniques.**

Links

- [Code and Results](#)
- [Weights and Biases](#)

Reference

- [1] <https://www.ibm.com/blogs/industries/rob-high-edge-computing>
- [2] https://en.wikipedia.org/wiki/Edge_computing
- [3] David Elliott, Evan Martino, Carlos E Otero, Anthony Smith, Adrian Peter, Benjamin Luchterhand, Eric Lam, and Steven Leung. Cyber-physical analytics: Environmental sound classification at the edge.

Thank you.

Adolf Anthony Dcosta • 24th April 2020