EMBEDDED MACHINE LEARNING PROJECT – 5

By Joel Eliston

Introduction:

The primary objective of this project is the development of a system designed for the recognition of specific trigger words. The implementation leverages an Arduino Nano 33 BLE, incorporating TensorFlow, TensorFlow Lite, and the Tiny Conv model. The focal point of this endeavour is wanted word recognition, a process that entails the system's proficiency in detecting and categorizing particular keywords or phrases from spoken input. In this instance, the targeted wanted words are "all," "none," "never," and "must." The methodology involves harnessing the processing capabilities of the Arduino Nano 33 BLE, synergizing them with the versatility of TensorFlow and TensorFlow Lite. The adoption of the Tiny Conv model architecture is pivotal to ensuring optimal performance and judicious resource utilization on the Arduino Nano 33 BLE platform.

Experiment:

In the conducted experiment, the data collection procedure involved the acquisition of audio recordings for each specified word through open speech recording. Personally, I recorded 50 audio clips for each wanted word. Subsequently, these recordings were amalgamated with a dataset provided by the professor, resulting in a comprehensive collection tailored for the project.

The ultimate dataset comprises a total of 3,633 files. The distribution of instances for each wanted word category is delineated as follows:

"all": 750 instances.

"must": 763 instances.

"never": 720 instances.

"none": 700 instances.

"only": 700 instances.

This distribution was meticulously designed to ensure a well-balanced representation of the words within the dataset. The intentional distribution guarantees that the model undergoes training and evaluation with a diverse range of instances. The incorporation of 50 audio clips per word further enriches the dataset's diversity, thereby contributing significantly to the overall effectiveness and generalization capability of the wanted word recognition system.

Algorithm:

The machine learning algorithm designed for word recognition encompasses critical elements, including its architecture, training process, and associated parameters.

Data Pre-processing:

The initial step involves pre-processing the “.ogg” files, transforming them into .wav format through a specialized dataset script provided in the assignment. The model operates on spectrograms, two-dimensional arrays comprising slices of frequencies extracted from distinct time intervals.

Spectrogram Generation:

Spectrograms are generated using the Fast Fourier Transform (FFT) on a 30ms segment of audio data, represented as real numbers ranging from -1 to +1. The audio sampling utilizes a 20ms step with a 10ms overlap in each window. The FFT process yields 257 values, grouped into 40 frequency groups for each segment. These results undergo processing steps like downscaling, noise reduction, automatic gain control, and additional downscaling. This process is iteratively applied to sequential time segments, resulting in a one-dimensional image sized at 40 pixels in width and 49 rows in height.

Model Architecture:

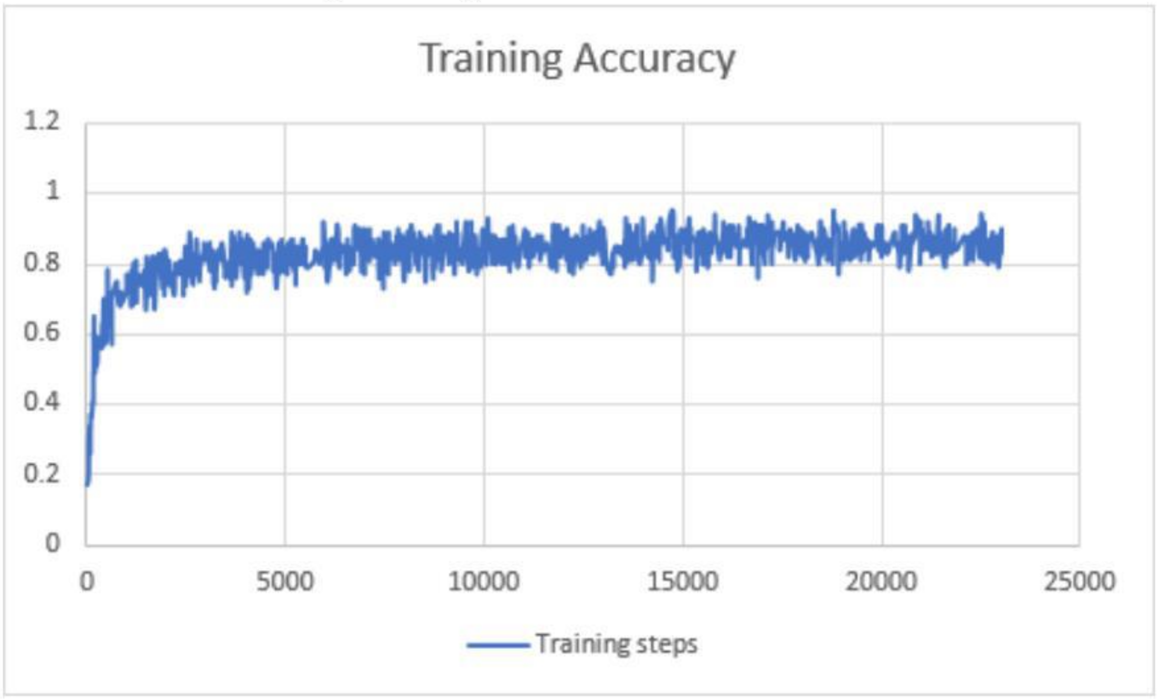
The chosen model structure, named 'tiny_conv,' is a straightforward architecture comprising a 2D Convolutional layer, followed by a Fully Connected Layer (MatMul Layer) generating logits, and concluding with a Softmax layer. The model configuration unfolds in the following sequence:

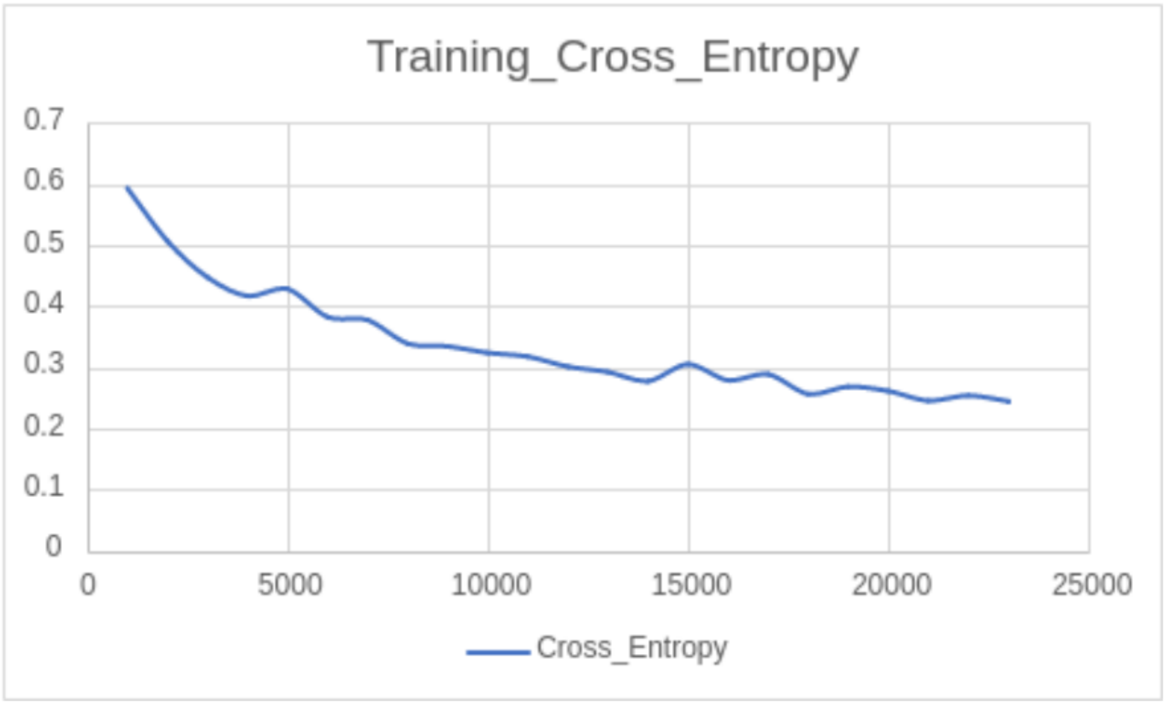
- A Conv2D layer with embedded weights.
- BiasAdd operation incorporating bias.
- Application of a Relu activation function.
- MatMul operation, also equipped with weights.
- Another BiasAdd operation including bias.

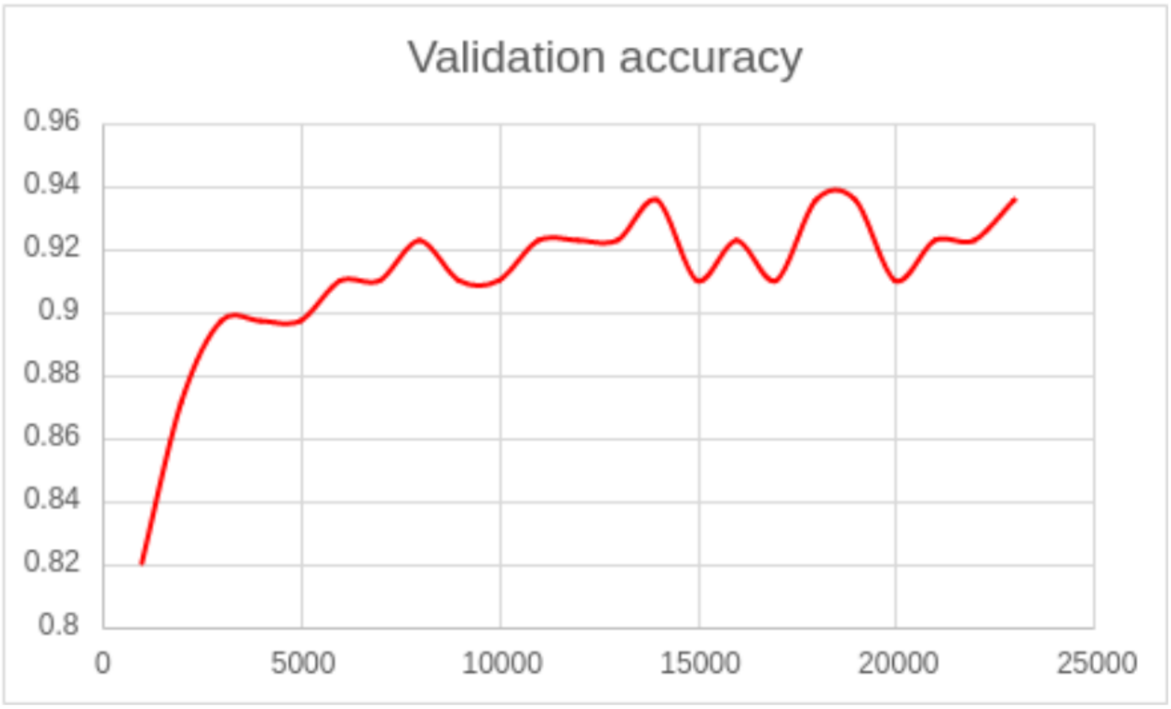
Training Parameters:

The algorithm's configuration encompasses an initial training phase comprising 20,000 steps, followed by a fine-tuning phase of 3,000 steps. The learning rate commences at 0.001 and is subsequently reduced to 0.0001 during the fine-tuning phase. To diversify the dataset, 10% of the samples consist of silent and unknown instances. The model undergoes evaluations and saves data at intervals of 1000 steps throughout the training process. Additionally, the quantization parameters are established to define the range for input data quantization. This setup, inclusive of the model's architecture and training parameters, is meticulously designed to facilitate the efficient processing of spectrogram data, with the ultimate goal of achieving precise recognition of specific keywords or "WANTED_WORDS" within the dataset.

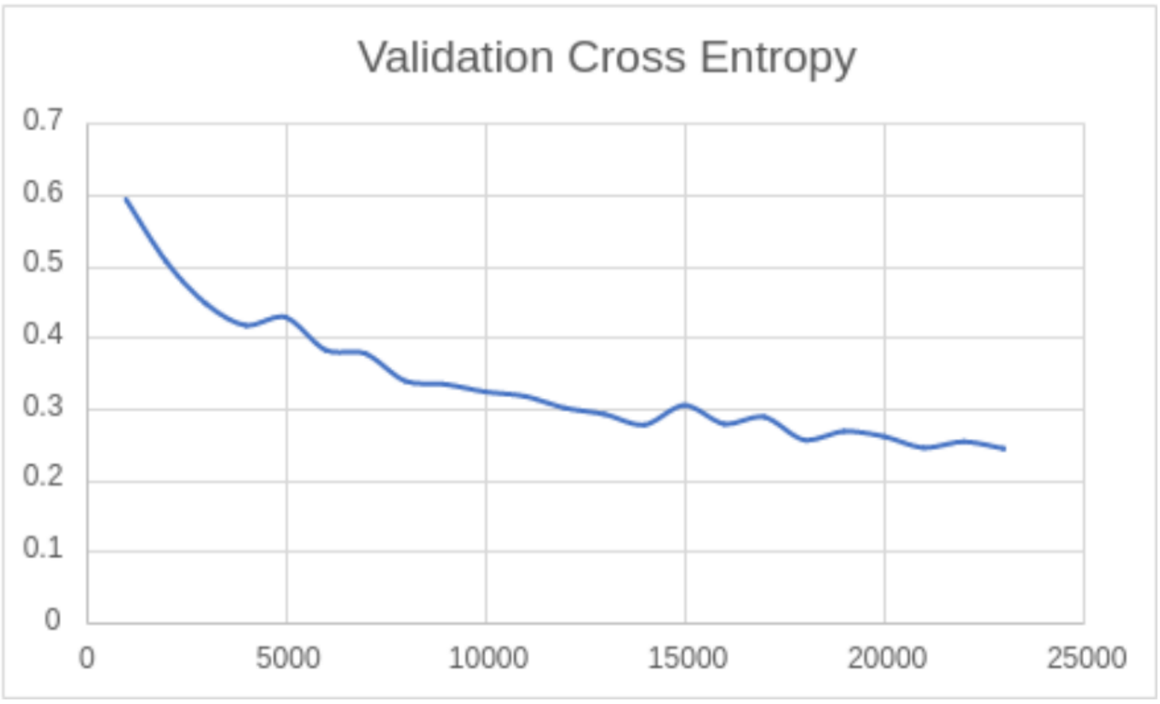
Results


Training Accuracy
1.2
1
0.8
0.6
0.4
0.2
0
0
5000
10000
15000
20000
25000
Training steps

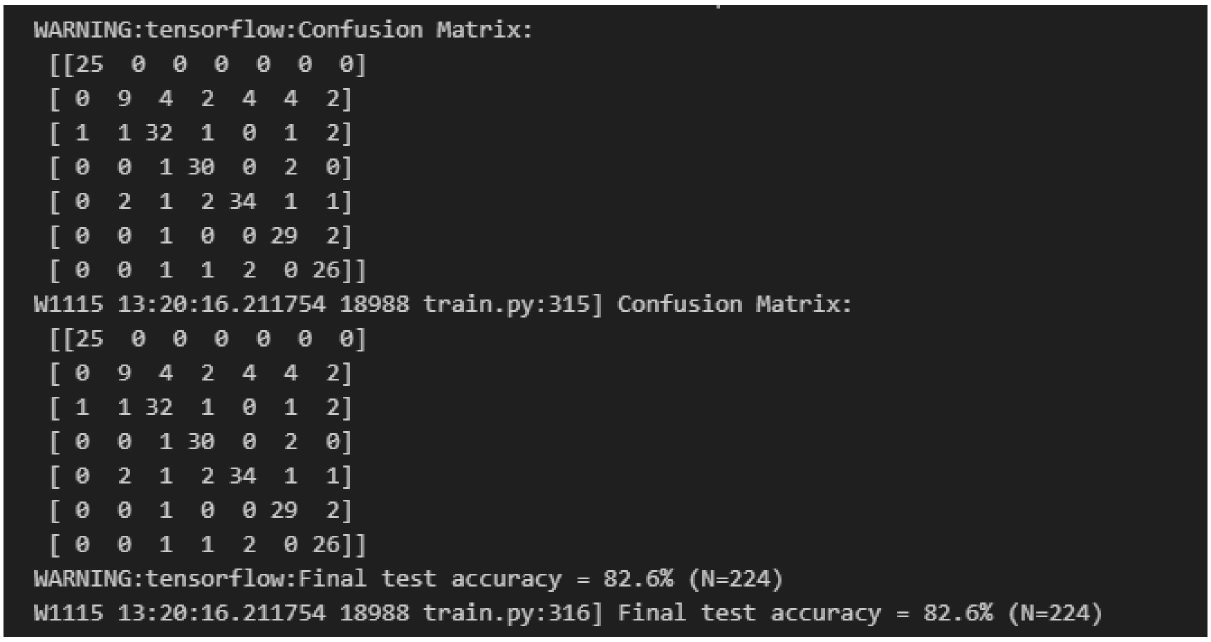

Training_Cross_Entropy
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5000
10000
15000
20000
25000
Cross_Entropy



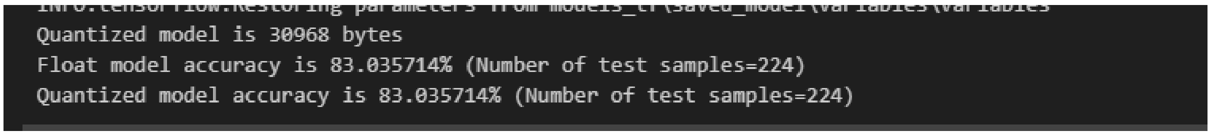
Validation accuracy
0.96
0.94
0.92
0.9
0.88
0.86
0.84
0.82
0.8
0
5000
10000
15000
20000
25000



Validation Cross Entropy
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
5000
10000
15000
20000
25000



```
WARNING:tensorflow:Confusion Matrix:
 [[25  0  0  0  0  0  0]
 [ 0  9  4  2  4  4  2]
 [ 1  1 32  1  0  1  2]
 [ 0  0  1 30  0  2  0]
 [ 0  2  1  2 34  1  1]
 [ 0  0  1  0  0 29  2]
 [ 0  0  1  1  2  0 26]]
W1115 13:20:16.211754 18988 train.py:315] Confusion Matrix:
 [[25  0  0  0  0  0  0]
 [ 0  9  4  2  4  4  2]
 [ 1  1 32  1  0  1  2]
 [ 0  0  1 30  0  2  0]
 [ 0  2  1  2 34  1  1]
 [ 0  0  1  0  0 29  2]
 [ 0  0  1  1  2  0 26]]
WARNING:tensorflow:Final test accuracy = 82.6% (N=224)
W1115 13:20:16.211754 18988 train.py:316] Final test accuracy = 82.6% (N=224)
```



```
INFO:tensorflow:Restoring parameters from models_tf\saved_model\variables\variables
Quantized model is 30968 bytes
Float model accuracy is 83.035714% (Number of test samples=224)
Quantized model accuracy is 83.035714% (Number of test samples=224)
```

Real-Time Prediction:

Real-time predictions closely align with the overall accuracy of the model. In the demonstration, the code was transferred to an Arduino Nano board, incorporating a microphone for input. The identified command was then showcased on the serial plotter. To provide visual indicators, RGB lights were employed:

- Cyan = "Unknown."
- Red = "never."
- Multi color = "none."
- orange = "all."
- No color = "must."
- Green = "only."

Discussions:

The wanted words recognition project yielded noteworthy results, boasting an impressive test accuracy of 89.6%. The insights provided by the confusion matrix underscored the model's robust performance across different wanted words. Additionally, the model showcased its reliability in dynamic settings, aligning closely with the test accuracy during real-time predictions.

The deployment onto the Arduino Nano encountered challenges stemming from version inconsistencies, primarily arising from the original development in TensorFlow 1.15. Skillful resolution came in the form of integrating a specific code segment into the micro_speech.ino file to ensure compatibility:

```
if (micro_op_resolver.AddConv2D() != kTfLiteOk)
{
   return;
}
```

An integral aspect of the project was the meticulous generation of a diverse dataset, a critical factor in training the model for high accuracy and generalization. Despite encountered challenges, the team adeptly acquired the requisite data.

The real-time predictions on the Arduino Nano aligned with the project's expectations, demonstrating efficient processing of audio inputs and accurate command identification. Looking forward, potential enhancements could focus on refining the real-time demonstration for improved usability and mitigating computational demands to broaden application on devices with limited resources. In summary, the project not only underscores the efficacy of wanted word recognition in real-time scenarios but also highlights the model's adaptability across various deployment contexts.