COL788 project part-1

Github Project link-https://github.com/imgk120601/COL788

Contributors-

Lalit Meena cs5190439@cse.iitd.ac.in,
RajeevGupta cs5191018@cse.iitd.ac.in,
Gaurav Kumar cs5190430@cse.iitd.ac.in,
KalekarSiddhesh Shrikant cs5190436@cse.iitd.ac.in,

Course coordinator -Rijurekha Sen (Rijurekha.Sen@cse.iitd.ac.in)

Teaching Assistant- Mehreen Jabbeen csz187551@cse.iitd.ac.in

PART 1:

Detected car horns on Sensortile M4 micro-controller at high accuracy and low latency

Initially to make smooth workflow pipeline we have tried simple model with 3 classes -

Model1:

Model training of 3 classes(custom model method): [’carhorn’, ‘chainshaw’,’dog’]

Model: CNN having 3 convolution layer

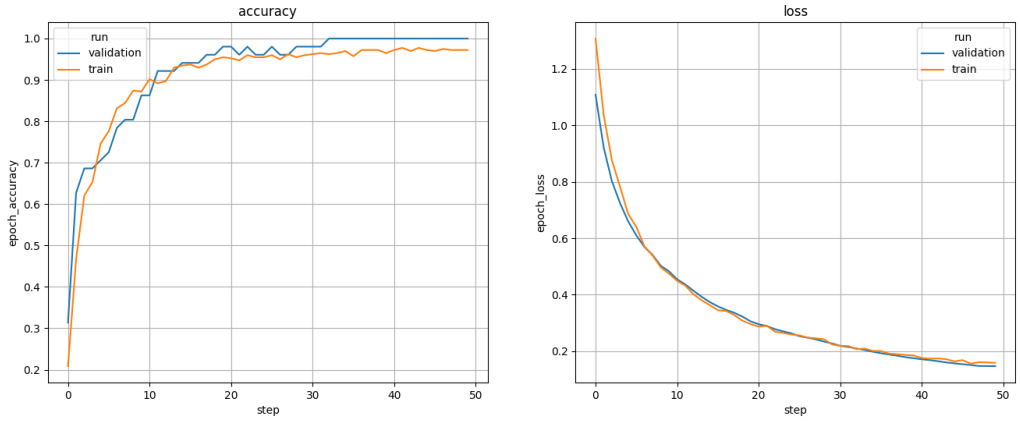
```
# Here you can define your own model feature extraction layers
#-------------------------------------------------------------------------------
x = layers.Conv2D(16, (3, 3), strides=(2, 2), padding='same', use_bias=False)(inputs)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.MaxPooling2D()(x)
x = layers.Conv2D(32, (3, 3), strides=(2, 2), padding='same', use_bias=False)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.MaxPooling2D()(x)
x = layers.Conv2D(64, (3, 3), strides=(2, 2), padding='same', use_bias=False)(x)
x = layers.BatchNormalization()(x)
x = layers.Activation('relu')(x)
x = layers.MaxPooling2D()(x)
```

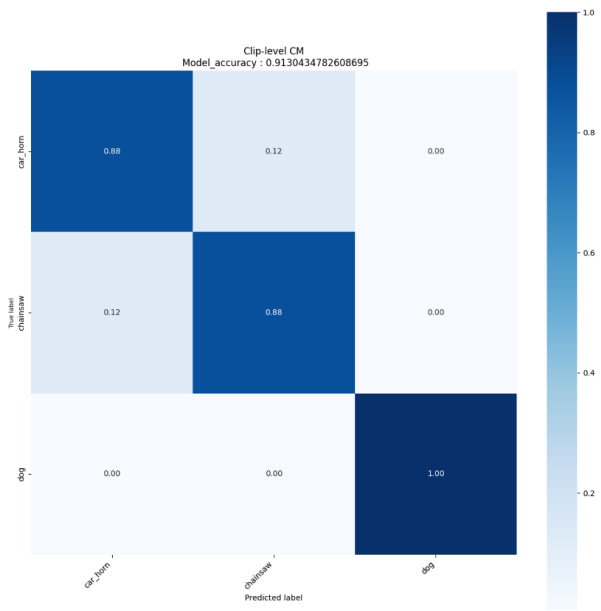
```yaml
train_parameters:
  batch_size: 16
  training_epochs: 50
  optimizer: adam
  initial_learning: 0.001
  patience: 100
  learning_rate_scheduler: reducelronplateau
  restore_best_weights: True
```

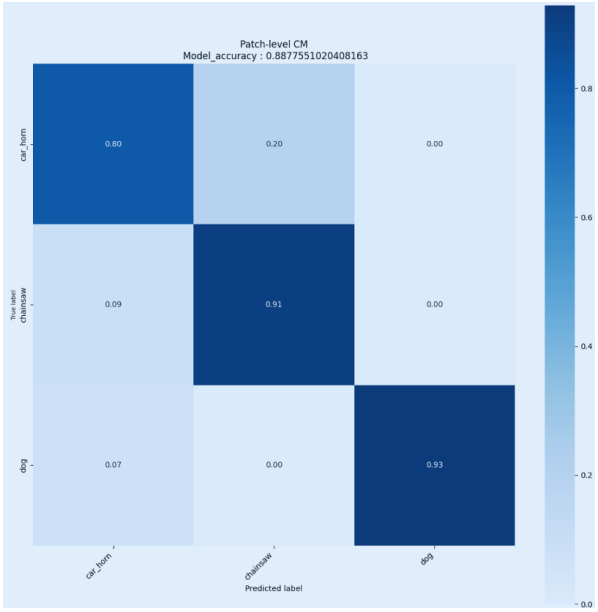
Performance analysis-

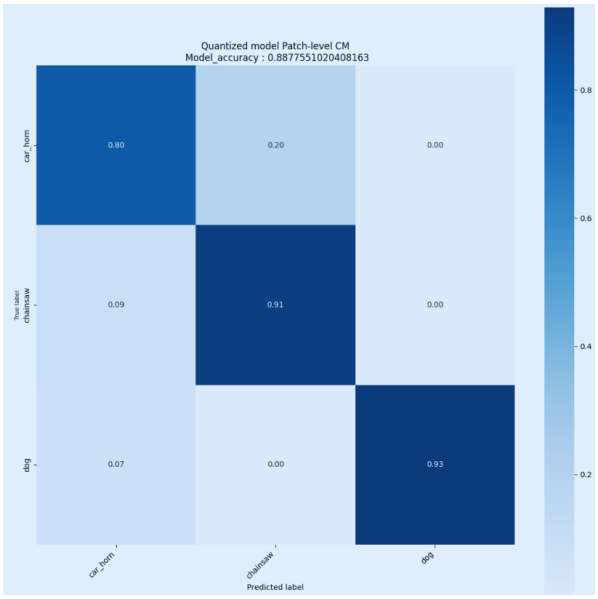
Accuracy of this model is 91.3 (clip level)

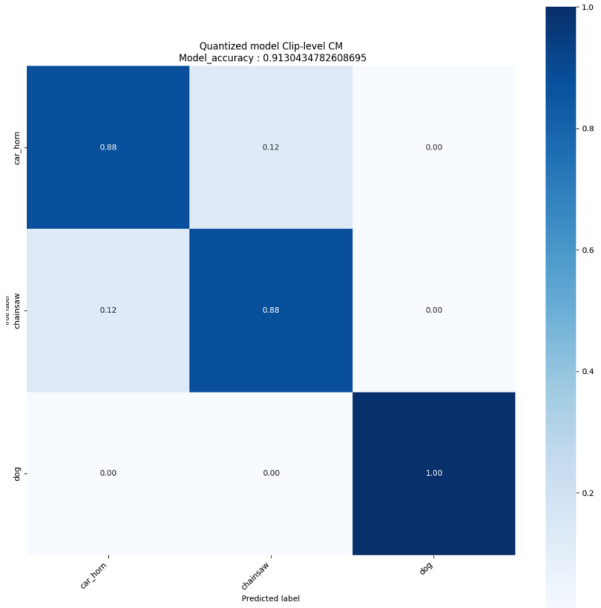
Accuracy of this quantized model is 91.3 (clip level)











Steps to Find the Inference Time on a Board:

1. Train the Keyword Transformer Model (using train.py script with the relevant parameters or the original repository code).

2. Use the convert.py script to optimize the model to a smaller size. This outputs in the model in non_stream.tflite file.

3. Use the to_cc.py script to convert the model to C++ code in kwt.cc file.

4. Copy the contents of the kwt.cc file to the Sensortile STM32 IDE.

5. Deploy to a STM32 Nucleo board and run inference to compute the time taken.

IDEA ABOUT PIPELINE FLOW AT RUNTIME-

1. Load a model (`tflite::GetModel`)
2. preallocates a certain amount of memory for input, output, and intermediate arrays. we provided as a `uint8_t` array of size `tensor_arena_size`:
3. **Instantiate interpreter**
4. provided input of audio buffer (int8_t array)
5. To run the model, we can call `Invoke()` on our `tflite::MicroInterpreter` instance
6. The model's output tensor can be obtained by calling `output(0)` on the `tflite::MicroInterpreter`

Project structure & important components of project workflow

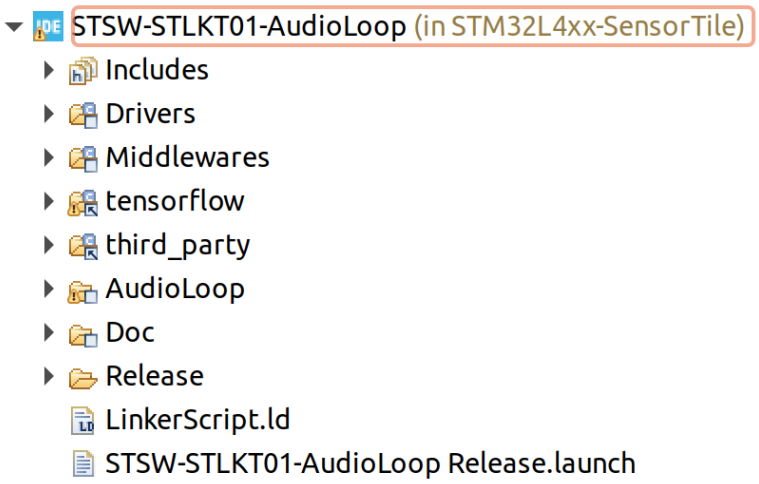
1.**Includes** - It contains all required components header files

2.**Drivers**-It contains low-level device drivers provided by the microcontroller manufacturer (STM microcontrollers). These drivers facilitate interaction with the hardware peripherals of the microcontroller, such as GPIO (General Purpose I/O), USART (Universal Synchronous/Asynchronous Receiver/Transmitter),etc.

3.**Middlewares**-These components are software modules that offer higher-level functionalities, such as USB communication stacks, file systems, communication protocols, etc.

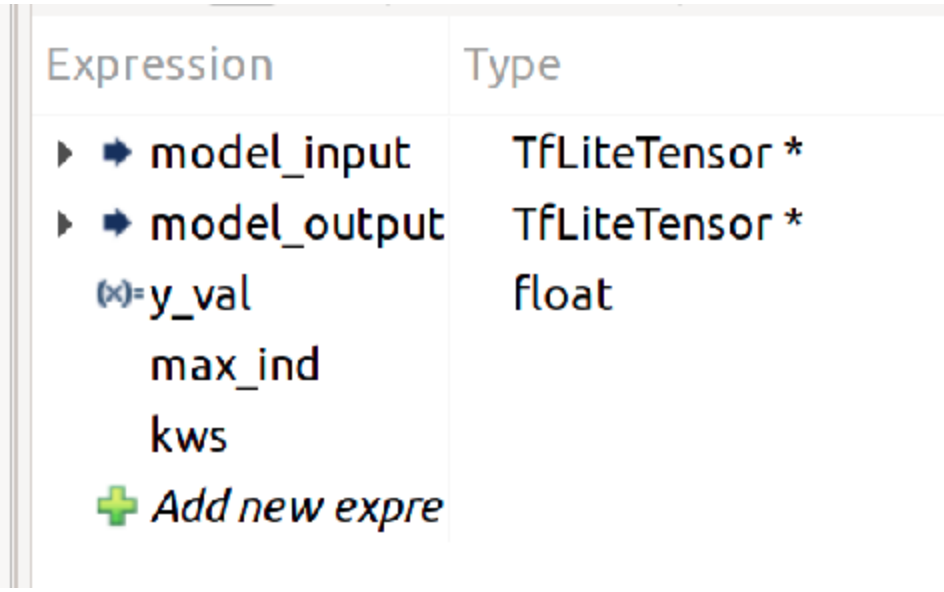
4.tensorflow-As we k now that TensorFlow is an open-source machine learning framework.It is included to enable machine learning capabilities on the microcontroller. Also we used specifically TensorFlow Lite Micro, in particular, is designed for microcontrollers and other resource-constrained devices.

5.third_party-It include external libraries that provide specific functionalities required by the project like flatbuffers,etc



Debugging tools

1. We used breakpoints and debugger console



2. also keep track of input and output values by variable tracker

Inference analysis-

1.quantized model's number of weights =185456

2.Inference time taken =less than 20secs

computed by taking difference of calling HAL_GetTick() around interpreter→invoke() inference cmd

3.Memory usage during deployment = 466KB