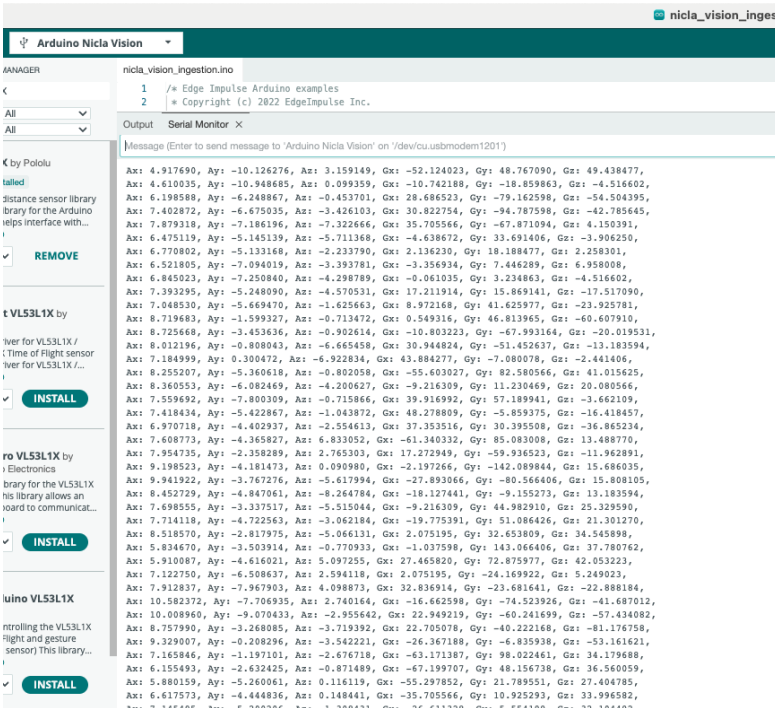
Divya Amirtharaj

CS249r Assignment 3

Section 0

1. Insert an image of your serial monitor IMU readings.



2. Specify if you used the dataset we provided you with or you collected your own. If you did collect your own dataset, please submit the .json file and describe the classes in your dataset.

I used the provided dataset

Section 1

1. What is the total accuracy for this baseline CNN model on the test dataset?

The total accuracy of this baseline CNN model on the test dataset is **89.55%**.

2. How large are your Tensorflow, TensorFlow Lite, and Tensorflow Lite Quantized models for this baseline CNN? How many times smaller is the quantized model from the original tensorflow model?

Size comparison of the models:

TensorFlow	107431 bytes	
TensorFlow Lite	3212 bytes	(reduced by 104219 bytes)
TensorFlow Lite Quantized	2992 bytes	(reduced by 220 bytes)

The TensorFlow Lite Quantized model is approximately 35 times smaller than the original TensorFlow model.

Section 2

1. What is the total accuracy for the CNN model that you built on the test dataset?

The total accuracy of my CNN model build on the test dataset is **95.45%**.

2. Given the code snippet provided, which converts a TensorFlow model to TensorFlow Lite (TFLite) format both in its regular and quantized forms, could you explain the quantization method applied? Specifically, which parts of the model undergo quantization? To what values or types are they being quantized. Additionally, how does the representative_dataset function contribute to this process?

The quantization method applied is called post-training integer quantization. This allows the model to run on integer only devices and improves CPU and hardware accelerator latency while mostly preserving accuracy. As we see on the Tensorflow website, this allows the model to be compatible with edge devices.

- Model weights are quantized by default: converter.optimizations = [tf.lite.Optimize.DEFAULT]

- All the operations are quantized to use integer computations. This means that the activations are quantized in addition to the weights: converter.target_spec.supported_ops = [tf.lite.OpsSet.TFLITE_BUILTINS_INT8]
- Inputs and outputs to the model are also quantized: converter.inference_input_type = tf.int8, converter.inference_output_type = tf.int8

The weights, activations, inputs, and outputs are all quantized to 8-bit integers. The representative_dataset function creates a generator that produces input data for the model from the dataset that reflects the distribution of the actual inference inputs. This real input data helps the converter calibrate the model, since the data is used to estimate the range of possible values for inputs and activations which then is used to determine the quantization parameters used.

3. Please discuss the trade-offs between Float16 Quantization and the quantization method you applied in the previous cell.

We will look at the tradeoffs between accuracy, size, and hardware compatibility of Float16 quantization and full integer quantization:

- **Accuracy:** Integer models theoretically are less accurate than Float16 quantized models, especially if the model has not been carefully calibrated with a representative dataset. This is because integer quantization introduces more quantization error than floating-point quantization.
- **Model size:** Integer quantization requires less memory bandwidth and size of the model overall is smaller by roughly half.
- **Compatibility & Performance:** As covered in the TensorFlow documentation, full integer quantization is more compatible with a larger range of hardware meaning it is more effective for TinyML/ on-device machine learning.

We also see that there is no representative_dataset function in the Float16 quantization. This is because in Float16 we don't need to quantize activations to integer values. Overall Int8 provides a smaller model size and faster inference speed on a larger range of hardware, but has a tradeoff potentially of accuracy if real-world data is not properly used for estimation.

4. How large are your Tensorflow, TensorFlow Lite, and Tensorflow Lite Quantized models for this CNN? How many times smaller is your quantized model from the original tensorflow model?

Size comparison of the models:

TensorFlow	639685 bytes	
TensorFlow Lite	98792 bytes	(reduced by 540893 bytes)
TensorFlow Lite Float16 Quantized	53124 bytes	(reduced by 586561 bytes)
TensorFlow Lite Quantized	31112 bytes	(reduced by 608573 bytes)

The TensorFlow Lite Quantized model is approximately 20 times smaller than the original TensorFlow model.

5. How many times smaller is your Tensorflow Lite Float16 Quantized model from the Tensorflow Lite model? Why is this the case?

The Tensorflow Lite Float16 quantized model is approximately 1.86 times smaller than the TensorFlow Lite model. This is because the quantization converts the 32 bit weights to 16 bit weights during quantization which is why the model size is roughly half. It's not exactly half since only the weights are halved, and not all the other parts of the model.

Section 3

1. What is the total accuracy for the pruned CNN model that you built on the test dataset?

The total accuracy of the pruned CNN model that I built on the test dataset is **94.09%**.

2. Please examine the summary table after pruning and describe what you learned about the effect of the pruning process on the model structure. (Hint: Compare it to the previous summary table)

Looking at the summary table of the model before and after pruning, we see that the architecture of the model is the same– the total number of parameters, layer structure, and output shapes are all the same and have a similar division of trainable and non-trainable parameters. Instead, the sparsity is increased which it does by adding a mask to the weights and zeroing them out. This means that once the model is saved after training, the pruning library can take advantage of these zeroed out weights and reduce the model's size when saving. Overall the structure is the same, but the increased sparsity provides benefits for deploying the model in resource-constrained environments since the zeroed out weights can be skipped in computations.

3. Out of all the models you trained, what model would you choose? Please justify your answer.

I would choose the pruned model (Section 3) since it is the most efficient model for deployment (in terms of size, and latency) without sacrificing a significant amount of accuracy. The model with the highest accuracy was 95.45%, and pruning the model only slightly sacrificed accuracy with a total accuracy of 94.09%. However, pruning the model reduced its size to 4096 bytes which even compared to the quantized models is roughly 8 times smaller.

Section 4

1. Download and open up your `magic_wand_model_data.cc`. Please explain what this file contains. (Hint: What does the array of hexadecimal values represent?)

The array of hexadecimal values represents the binary data of the model. It is in this format so that it can be compiled directly into the application's binary which then allows the trained machine learning model to be deployed into embedded devices/ devices that don't have a file storage system. This allows us to build this model in a human readable environment like Python, and then still be able to convert it to TFLite and deploy it for use on devices.