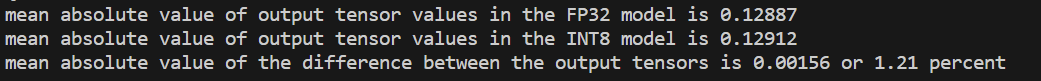
Dynamic Quantization:

**Tutorial Code Outcome**

Model comparison  
A black background with white text

Description automatically generated

Latency comparison  


Accuracy   
  
We can see that the INT8 model has a similar mean to the output values produced by the original floating point 32 numbers, indicating that the INT8 model does not stray away that much from the range we get from the full FP32 model numbers.

**LeNet Code Outcome**

**Accuracy**

After testing with LeNet using the normal approach and the dynamically quantized approach, we get the same accuracy for both, proving the above results where the output tensor values were almost the same.

**Model Size**

FP32: 672.554 KB

Int8: 58.026 KB

**Time Consumption:**

FP32: 0.001 seconds

Int8: 0.002 seconds

Even though at first glance the quantized model should perform faster due to smaller computations, the increase in time taken can be explained by the overhead of changing the values to Int8 and FP32 constantly, which adds additional time to the inference.

Static Quantization:

**Tutorial Code Outcome**

FP32  
A screen shot of a computer code

Description automatically generated  
Int8

A black background with white text

Description automatically generated

For the Static and the QAT approaches, we cannot implement those on the LeNet models like we did with the dynamic approach because the tutorials use the models (AlexNet for example) from the torch.quantization module provided by pytorch, which have the hyper functions necessary to apply the quantization techniques. But since our model is not within this module, we cannot use those hyper functions, and so, we have to implement them ourselves if we have to approach this problem.