VARIABLE PRECISION DEEP LEARNING (FLOAT16 & FLOAT32)

- **CS-597: Reading and Special Problems**
- **Team Members**
  - Nilesh Jorwar (A20405042)
  - Ashish Ryot (A20405230)

PROBLEM STATEMENT

- Performance of CPU with low precision data types, along with data representation techniques (floating point) and the variation of results with adjustments to various parameters of the model needs to be experimented.
- Improvisation of the previous project to get better accuracy and lower training time using various techniques:
  - Implementation of AVX instructions for speeding up runtime (float32)
  - Multithreading implementation for float32 and float16 data types

MOTIVATION

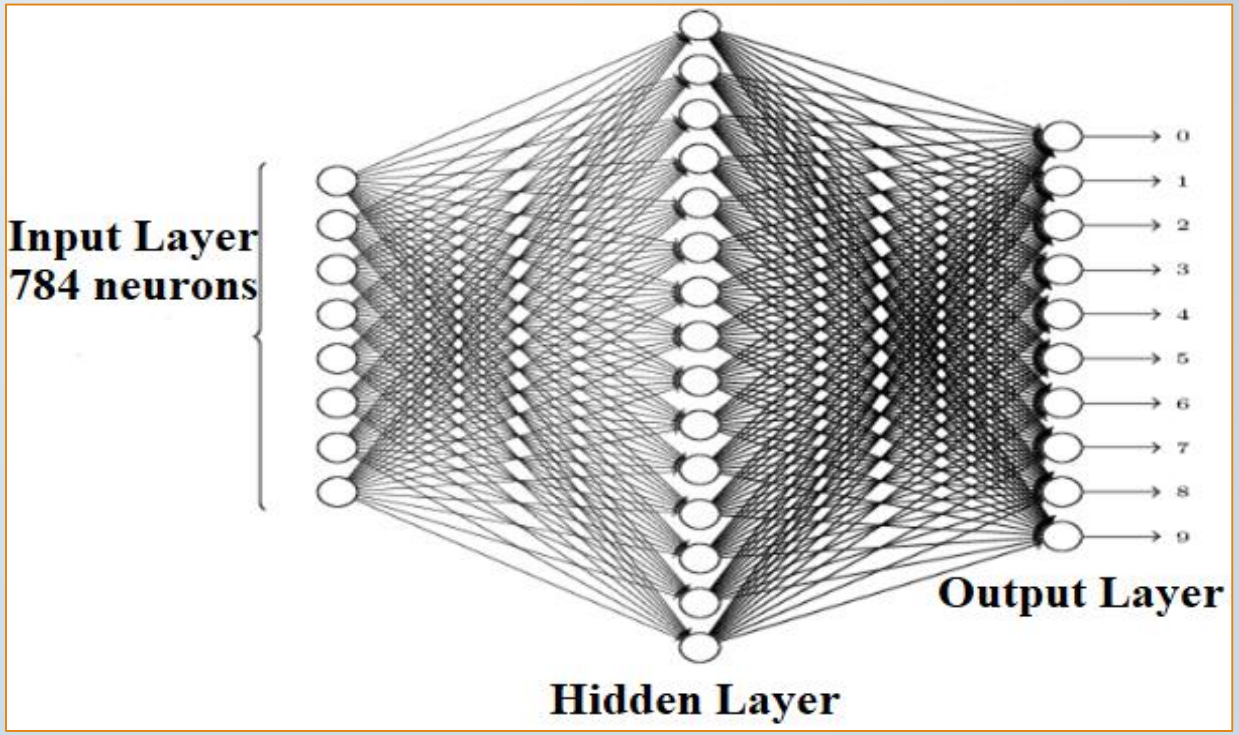
- Need for speedup and performance on larger datasets using low variable precision data types without loss of accuracy, precision and recall.
- Deep learning on problems like image recognition is very compute intensive task.

IMPROVEMENT OVER PAST PROJECT & CURRENT PROGRESS

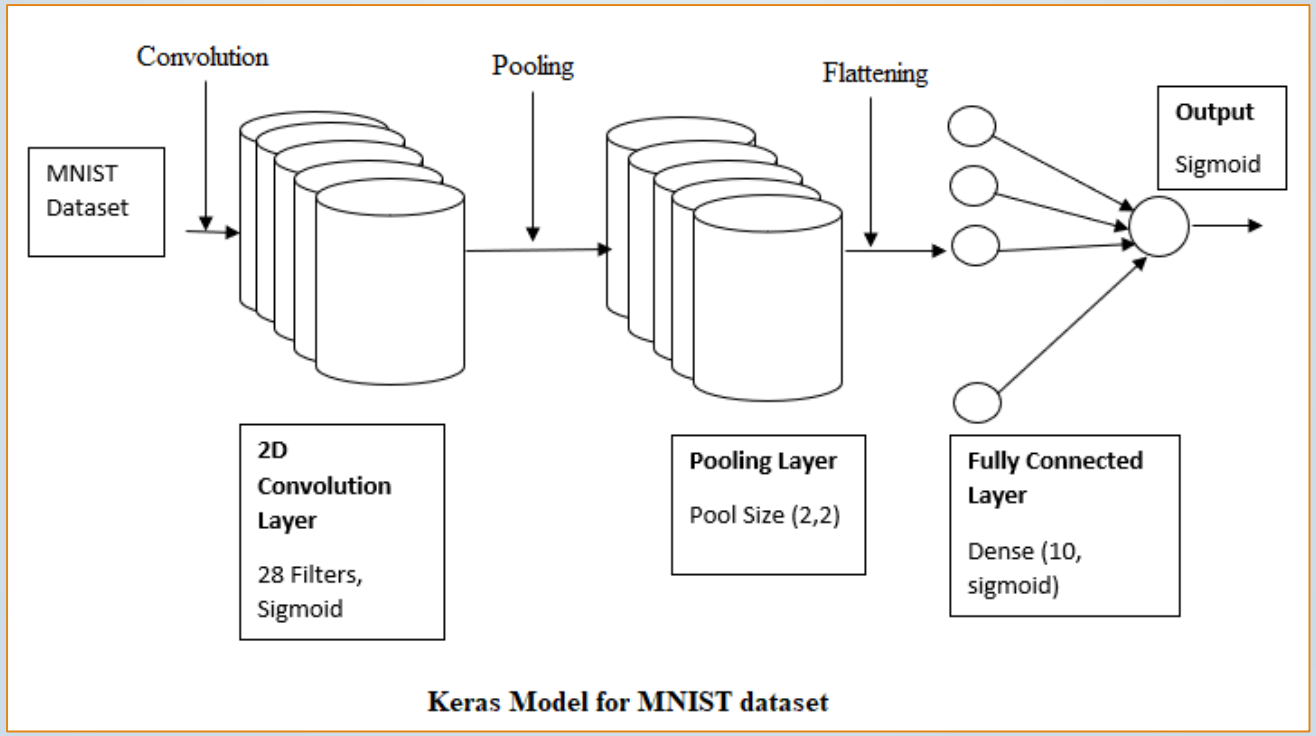
- AVX instructions for float32 data types
- Parallelization of neural network using thread library for float16 data types
- Testing model on more complex datasets
- Comparison of C++ model with TensorFlow and Keras in Python
- Testing of Keras model with int data type

ARCHITECTURES

- cpp Model – Input layer with 784 neurons, hidden layer (variable neurons), and output layer (10 neurons)
  - Activation Functions: sigmoid
- TensorFlow Python Model
- Keras Python Model

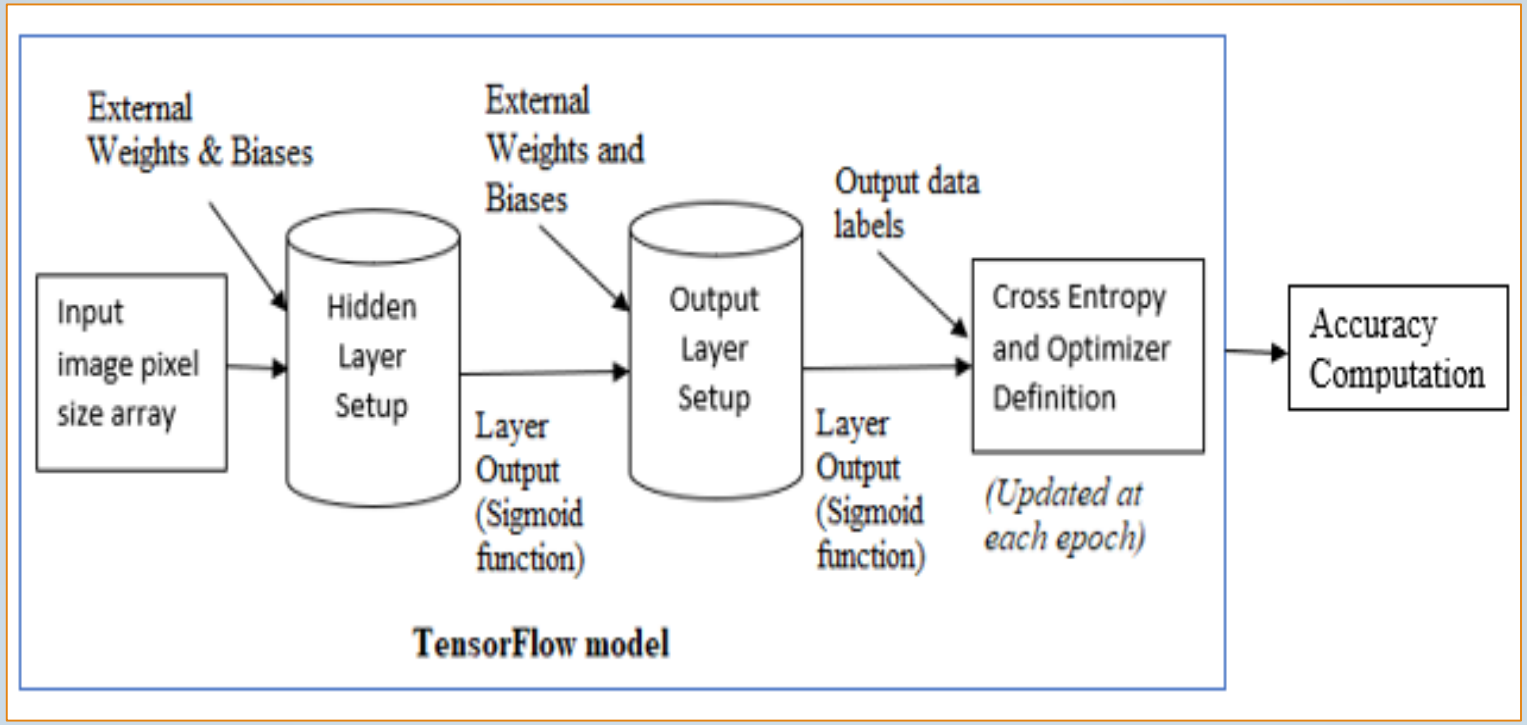


ARCHITECTURE – Keras Model



Keras Model for MNIST dataset

ARCHITECTURE – TensorFlow Model



TESTBED REQUIREMENTS

- Intel(R) Xeon(R) CPU e5-2667 v3 @ 3.20ghz
- CPU 2
- Cores 16
- Logical Processors 32

ACHIEVEMENTS

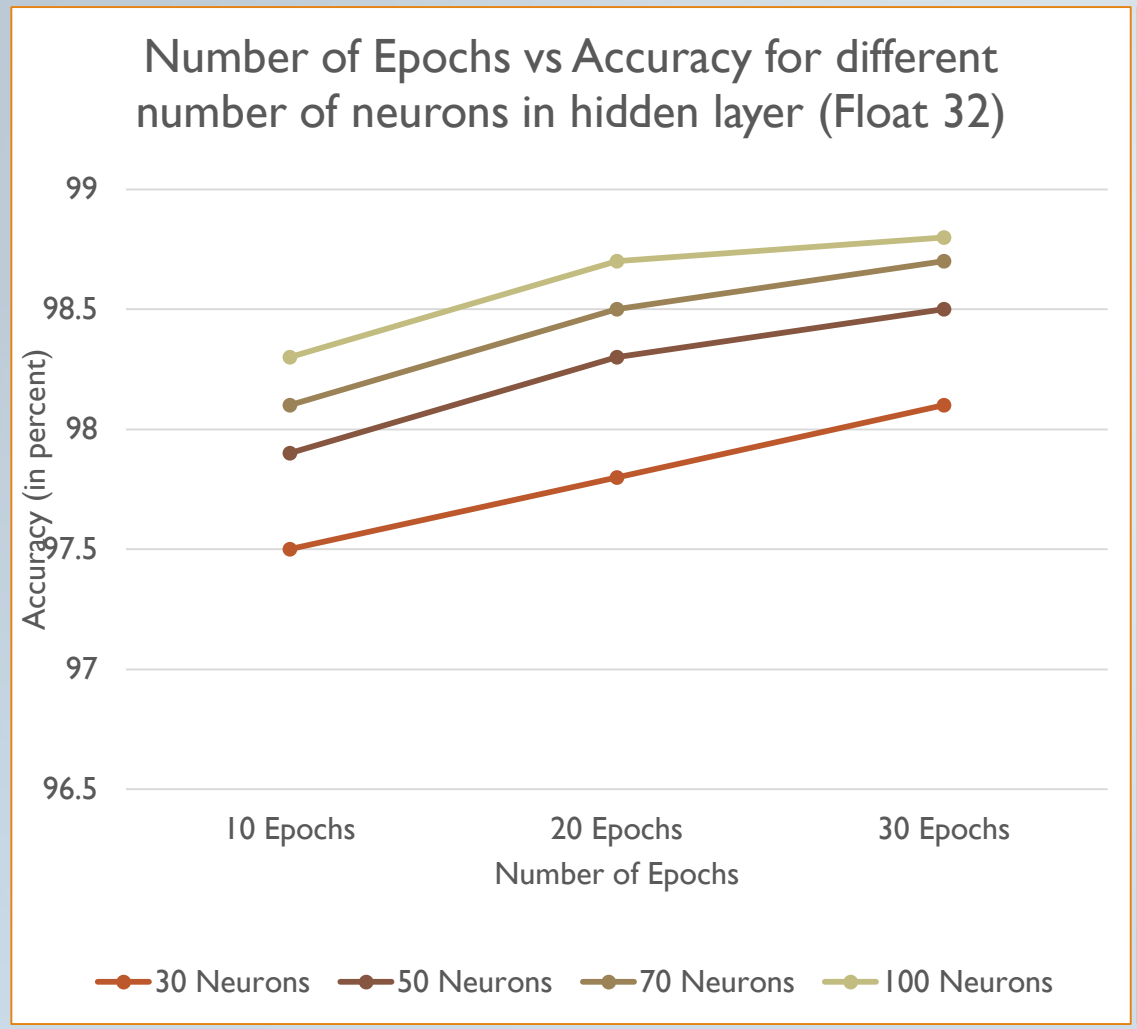
- Successful implementation of multithreading on neural network model
- AVX instructions implementation
- Improved accuracy, precision and recall
- Reduced runtime.
- Experiments over different datasets
- Evaluation over different platforms and models like Keras, TensorFlow, and CPP

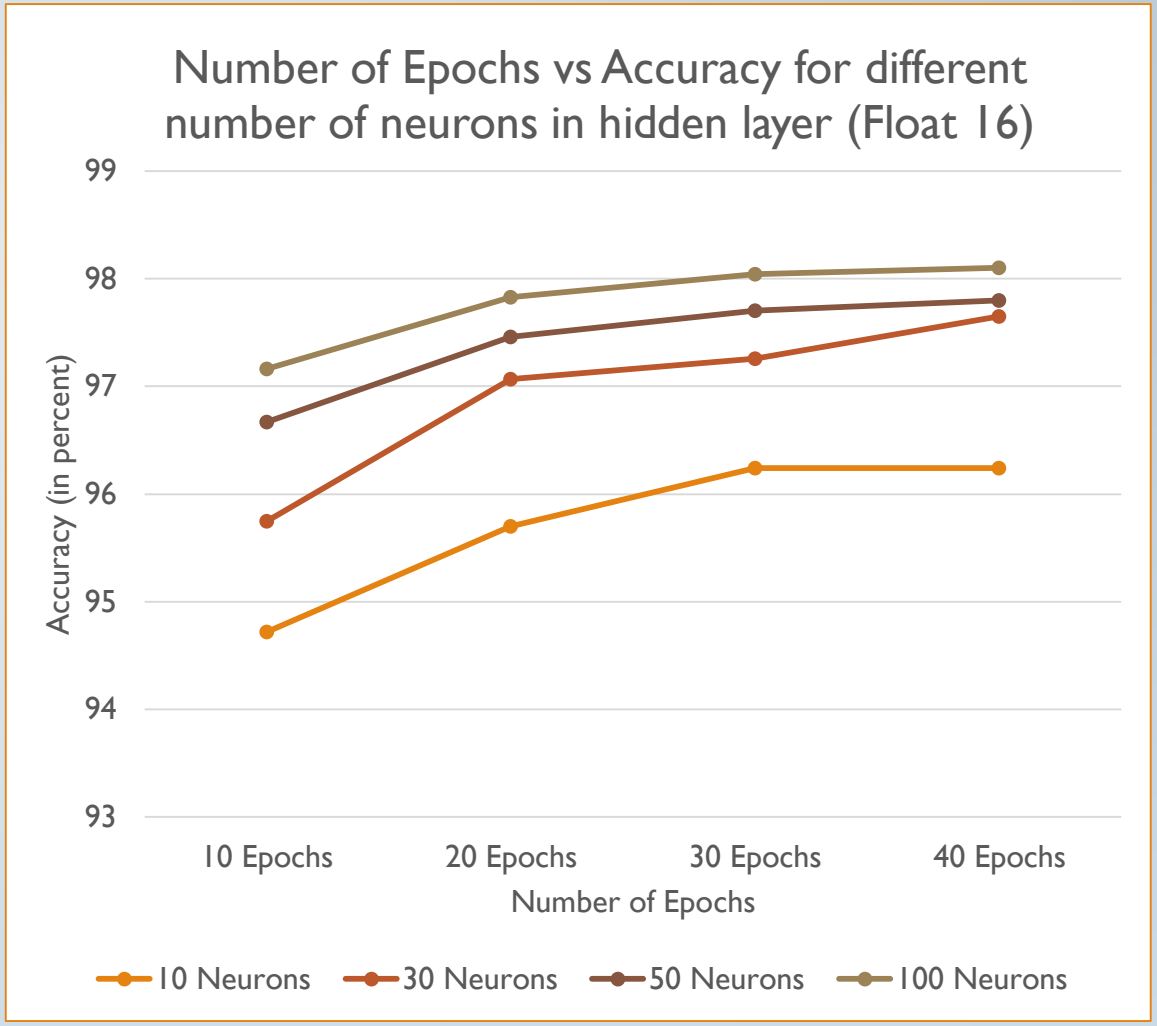
CHALLENGES FACED

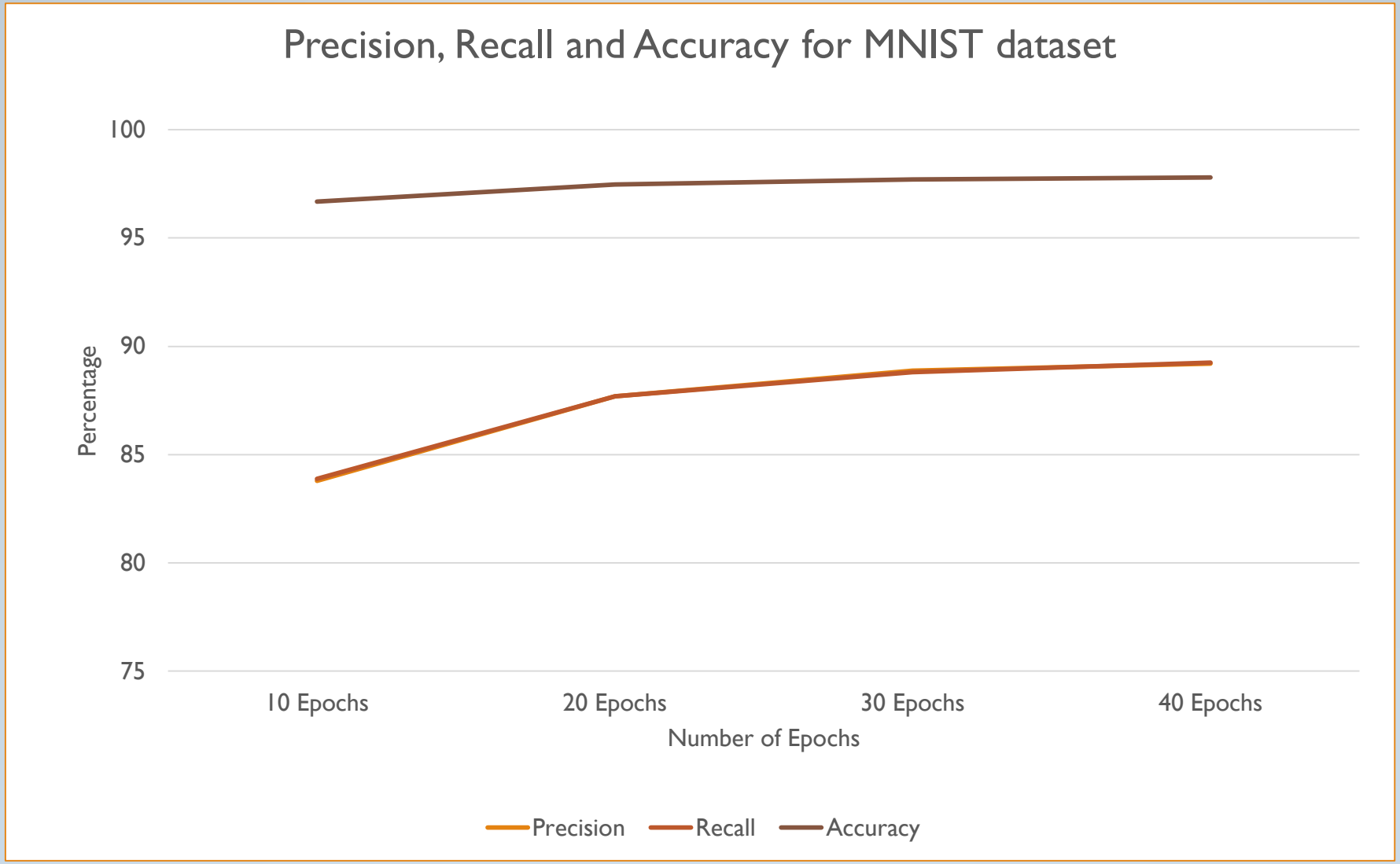
- Implementation of AVX512 instructions for float16 data types.
- Implementation using int data types.
- Implementation using low precision data types (CLN) [4].
  - The external library used, rounds the original data into fixed number of real bits , which results in loss of precision.
  - The precision decreases and number of epochs required to get good accuracy increases.

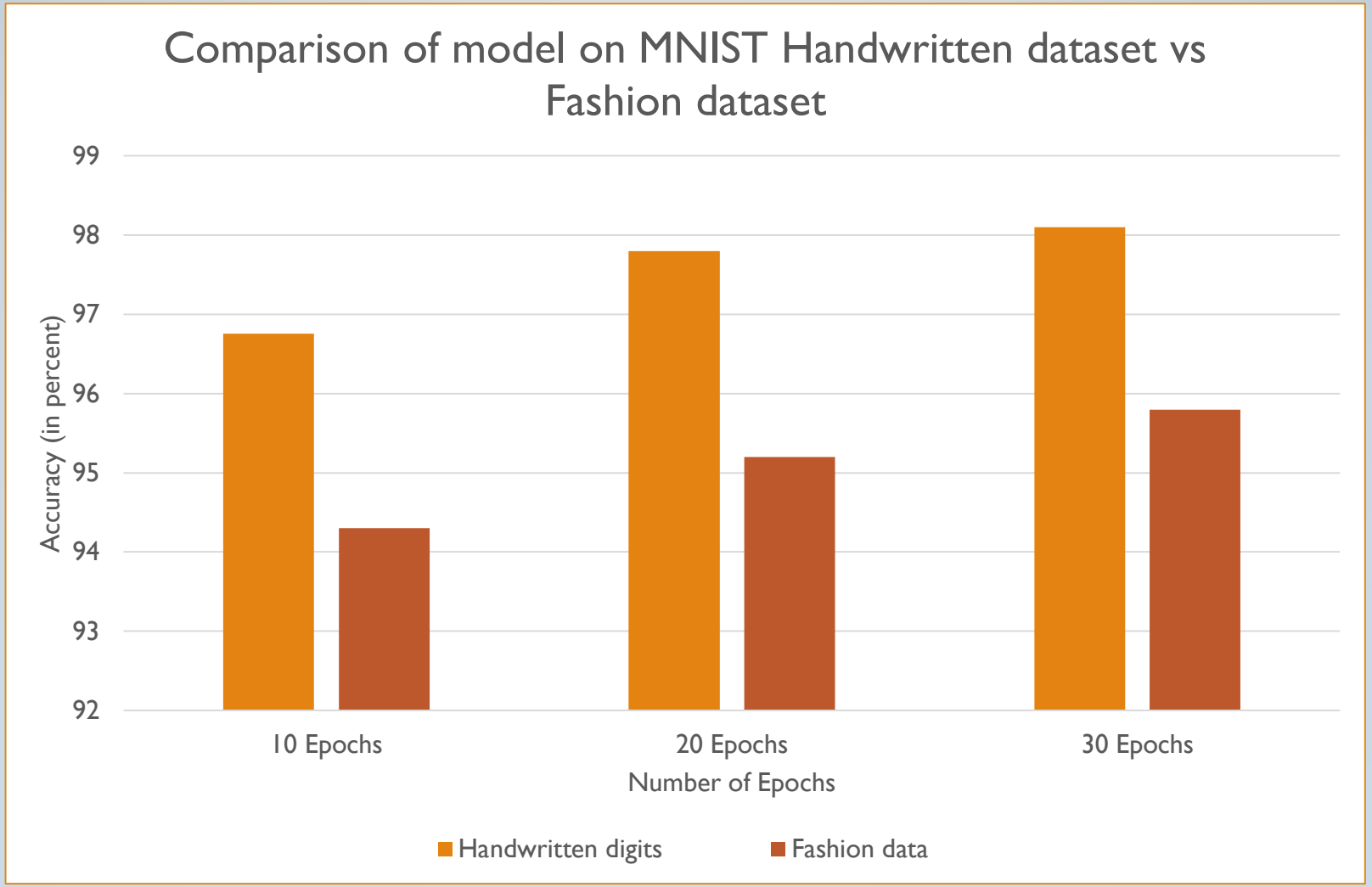
PERFORMANCE EVALUATION

- MNIST and Fashion dataset
  - Float32 and float16 data types
- Evaluation using varying number of nodes like 10, 20, 30, 50, 70, 100, and 400 in hidden layer
- Accuracy computations for different number of epochs like 10, 20, 30, and 35.


Number of Epochs vs Accuracy for different number of neurons in hidden layer (Float 32)
99
98.5
98
97.5
97
96.5
Accuracy (in percent)
10 Epochs
20 Epochs
30 Epochs
Number of Epochs
30 Neurons
50 Neurons
70 Neurons
100 Neurons


Number of Epochs vs Accuracy for different number of neurons in hidden layer (Float 16)
99
98
97
96
95
94
93
Accuracy (in percent)
10 Epochs
20 Epochs
30 Epochs
40 Epochs
Number of Epochs
10 Neurons
30 Neurons
50 Neurons
100 Neurons

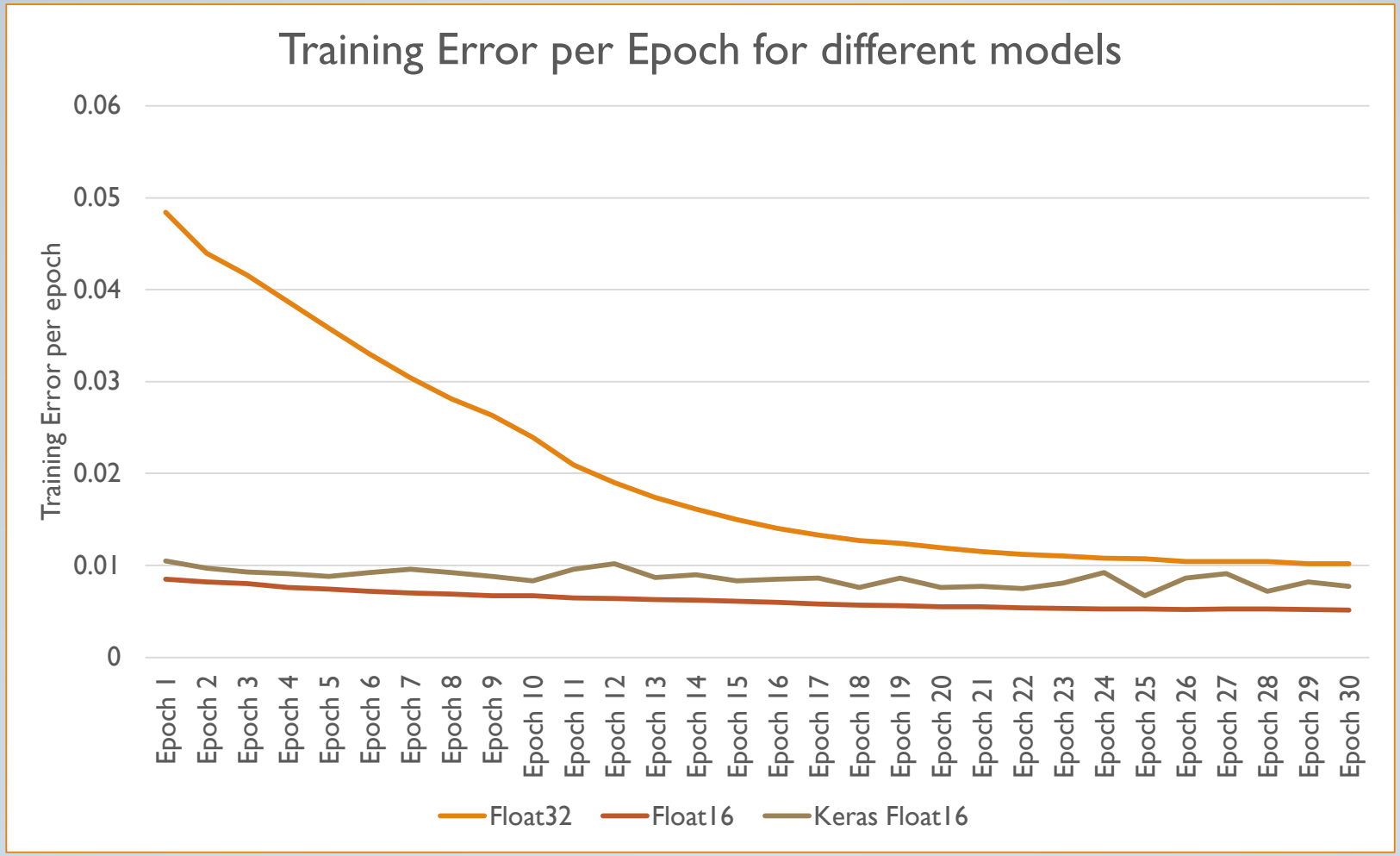

Precision, Recall and Accuracy for MNIST dataset
100
95
90
85
80
75
Percentage
10 Epochs
20 Epochs
30 Epochs
40 Epochs
Number of Epochs
Precision
Recall
Accuracy


Comparison of model on MNIST Handwritten dataset vs Fashion dataset
99
98
97
96
95
94
93
92
Accuracy (in percent)
10 Epochs
20 Epochs
30 Epochs
Number of Epochs
Handwritten digits
Fashion data

Accuracy and Runtime over different epochs for different models

		cpp Model Float32 (avx)	cpp Model Float16 (multi-threaded)	TensorFlow Model (Keras) Float16
10 Epochs	Accuracy (%)	96.8	95.72	98.32
	Runtime (secs)	40.51	654	761
20 Epochs	Accuracy (%)	97.8	97.07	98.43
	Runtime (secs)	133.50	1593	1530
30 Epochs	Accuracy (%)	98.1	97.26	98.55
	Runtime (secs)	205.47	2425	2276

		cpp Model Float32	TensorFlow Model Float32
10 Epochs	Accuracy (%)	96.2	95.69
	Runtime (secs)	51.33	30
20 Epochs	Accuracy (%)	97.8	95.65
	Runtime (secs)	152.52	60.928
30 Epochs	Accuracy (%)	97.9	95.72
	Runtime (secs)	231.37	90.88


Training Error per Epoch for different models
Training Error per epoch
0.06
0.05
0.04
0.03
0.02
0.01
0
Epoch 1
Epoch 2
Epoch 3
Epoch 4
Epoch 5
Epoch 6
Epoch 7
Epoch 8
Epoch 9
Epoch 10
Epoch 11
Epoch 12
Epoch 13
Epoch 14
Epoch 15
Epoch 16
Epoch 17
Epoch 18
Epoch 19
Epoch 20
Epoch 21
Epoch 22
Epoch 23
Epoch 24
Epoch 25
Epoch 26
Epoch 27
Epoch 28
Epoch 29
Epoch 30
Float32
Float16
Keras Float16

FUTURE SCOPE

- Implementation of VNNI instructions on float16 data types
- Conversion of CPP model to work on GPU platform using CUDA
- Testing the CPP model on ImageNet

REFERENCES

- [1] https://thinkingandcomputing.com/posts/using-avx-instructions-in-matrix-multiplication.html
- [2] https://adventuresinmachinelearning.com/python-tensorflow-tutorial/
- [3] https://cognitivedemons.wordpress.com/2017/07/06/a-neural-network-in-10-lines-of-c-code/
- [4] https://ginac.de/CLN/cln.html