Deep Learning Model Results

Data size	Configuration	Training error (loss)	Validation error (loss)	Training accuracy	Validation accuracy	Time of execution (secs)
1000	1 hidden layer of 4 nodes	0.3445	0.3484	0.9112	0.94	3.05
10000	1 hidden layer of 4 nodes	0.0283	0.0308	0.996	0.9975	10.85
100000	1 hidden layer of 4 nodes	0.0067	0.0072	0.9986	0.998	90.05
1000	2 hidden layers of 4 nodes	0.3958	0.3835	0.7987	0.805	2.87
10000	2 hidden layers of 4 nodes	0.0133	0.0155	0.9975	0.998	11.37
100000	2 hidden layers of 4 nodes	0.0039	0.0065	0.9985	0.9977	94.11

1. Based on the results, which model do you consider as superior, among the deep learning models fit?

- I would select 2 hidden layers with 4 nodes with 10000 samples. It has the lowest losses and highest accuracies except for the models with 100000 samples but has considerably smaller amount of time for execution than models with 100000 samples. However, if I could have used GPU, I choose 100000 as it can make the process considerably faster.

2. Next, report the results (for the particular numbers of observations) from applying xgboost (week 11 – provide the relevant results here in a table). Comparing the results from XGBoost and deep learning models fit, which model would you say is superior to others? What is the basis for your judgment?

Accuracy:

- At 10,000 and 100,000 data sizes, deep learning achieves higher accuracy (99.75–99.80%) compared to all XGBoost models (max 99.1%)
- At 1,000, Python XGBoost slightly outperforms deep learning (95.1% vs. 94.0%).

Execution Time:

- For smaller sizes, XGBoost in Python is extremely fast (under 1 sec).
- Interms of Python comparison, Deep learning takes much longer, especially at larger sizes (90+ seconds), but still much faster than R.
- R-based caret is the slowest, particularly for large datasets.

Therefore, if I am looking for better accuracy, I will go for deep learning and if I am looking for time efficiency, I will go for XGBoost on Python.