ILLINOIS INSTITUTE OF TECHNOLOGY

MATH 554 – Big Data Technologies

Final Paper

Comparative Analysis of Machine Learning Libraries

Shahrukh Sohail
ssohail3@hawk.iit.edu

Prof. Jawahar Panchal

Dec 6, 2022

Table of Contents

1. Abstract

This paper performs a comparative analysis of Machine Learning Libraries. TensorFlow, PyTorch and Keras are one of the topmost adopted libraries today. Keras on TensorFlow and PyTorch are selected for comparative analysis. The first problem is a handwriting digit recognition problem, where both the libraries are trained, and their respected accuracies are observed for a given number of epochs. Secondly both these libraries are trained to solve a regression problem and their mean squared error is observed for given number of epochs.

The results of the comparison would be compiled in the form of graphs of training time and accuracy depending on the number of epochs and in tabular form. The results would be displayed in the following format:

Winner Model Table

Model	Classification	Regression
Accuracy/MSE	X	X
Training Time	X	X

The Model with better results is displayed for each category in Classification and Regression Problem.

2. Overview

Deep learning is a type of machine learning that imitates the way humans gain knowledge and learn new things. Deep learning algorithms are essentially neural networks with three or more layers. Deep learning neural networks attempt to mimic the human brain, they consider the input data, bias, and the weight of each epoch. They work together to accurately perform prediction, classification, clustering within the data. The term *"deep"* refers to the multiple layers along with hidden layers used in the network.

Deep neural networks are made up of several layers of interconnected nodes, these include nodes within input, hidden and output layer; each of which improves upon the accuracy/precision made by the one underneath it. Forward propagation is the calculations conducted in the network. Deep learning algorithms can run with only a single layer; however much higher accuracy is achieved by introducing multiple hidden layers. In a deep neural network input and output layers are visible layers. The deep learning model ingests the data for processing in the input layer, combined with computations in the hidden layers the final prediction or classification is performed in the output layer.

Backpropagation is another technique that uses gradient descent to calculate prediction errors, it then changes the weight and the bias of the function by iteratively going back through the layers to train the model. A neural network can generate predictions and make necessary corrections for any faults, thanks to the hidden layers and using propagation and backpropagation working together. The algorithm continuously improves in accuracy over time.

Deep learning can be used as a method to automate predictive analytics. Deep learning is different from most machine learning algorithms as ML models are linear whereas deep learning algorithms are of a hierarchal nature from arranged simple to complex abstractions. It solves a complex problem by creating a hierarchy where each level of abstraction is generated using information from the layer before it in the hierarchy. The ability of deep learning algorithm to solve complex abstractions using multiple hidden layer models with great accuracy, precision and speed have resulted in the recent trending of deep learning.

Many artificial intelligence applications are powered by deep learning, which enhances automation by carrying out intense computation without the need for human interaction. Deep learning is the driving technology that powers technologies like autonomous driving, voice-activated TV remote controls, digital assistants, and credit card fraud detection.

With each library from the overview above, we will build perceptron models. Each neural network will be trained, and training time and accuracy/MSE will be measured for a different number of training epochs. We will use these measurements for comparative analysis of libraries.

This paper applies neural network models using Keras on TF and PyTorch to solve a classification and regression problem, the performance of these libraries is then benchmarked and presented for comparative analysis.

2.1. TensorFlow

TensorFlow is an open-source library for numerical computation and large-scale machine learning. TensorFlow was developed by the Google Brain team and was made public in 2015. TensorFlow uses common programming metaphors, performs high level computing, and aggregates a variety of machine learning and deep learning models and methods to make them useful. The front-end API and applications are coded using Python or JavaScript, whereas the application are run in high-performance C++.

TensorFlow, which competes with frameworks like PyTorch, can train and run deep neural networks for image recognition, word embeddings, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation)-based simulations. It can also train and run recurrent neural networks. The best part is that TensorFlow uses the same models that were used for training to provide production prediction at scale.

You can leverage TensorFlow's extensive library of pre-trained models in your own applications. For training your own models, you may also utilize code from the TensorFlow Model Garden as an example of best practices.

2.2. Keras

Keras was created to be user friendly, modular, easy to extend, and to work with Python. The API was “designed for human beings, not machines,” and “follows best practices for reducing cognitive load.”

Neural layers, cost functions, optimizers, initialization schemes, activation functions, and regularization schemes are all standalone modules that you can combine to create new models. New modules are simple to add, as new classes and functions. Models are defined in Python code, not separate model configuration files.

Given that the TensorFlow project has adopted Keras as the high-level API for the upcoming TensorFlow 2.0 release, we will be using Keras on Tensor for comparative analysis.

The biggest reasons to use Keras stem from its guiding principles, primarily the one about being user friendly. Beyond ease of learning and ease of model building, Keras offers the advantages of broad adoption, support for a wide range of production deployment options, integration with at least five back-end engines (TensorFlow, CNTK, Theano, MXNet, and PlaidML), and strong support for multiple GPUs and distributed training. Plus, Keras is backed by Google, Microsoft, Amazon, Apple, Nvidia, Uber, and others.

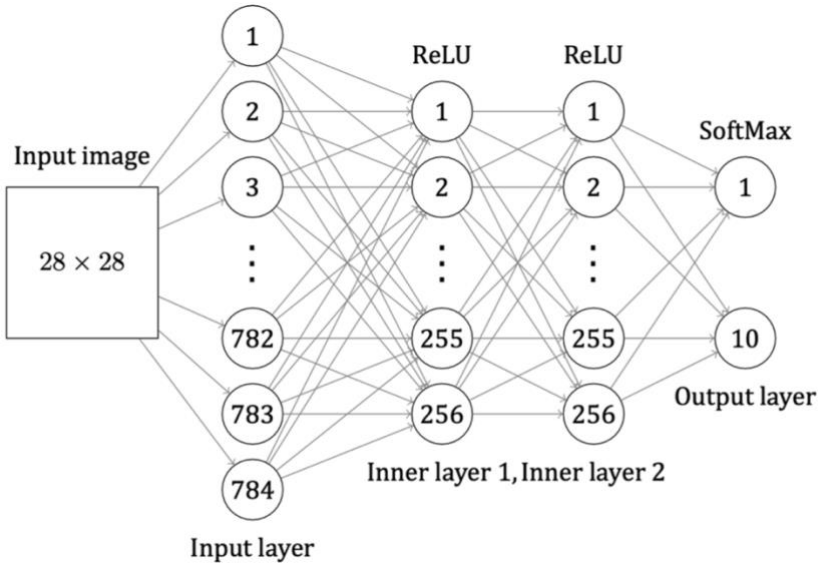
2.3. PyTorch

PyTorch is a fully featured framework for building deep learning models, which is a type of machine learning that's commonly used in applications like image recognition and language processing. Written in Python, it's relatively easy for most machine learning developers to learn and use. PyTorch is distinctive for its excellent support for GPUs and its use of reverse-mode auto-differentiation, which enables computation graphs to be modified on the fly. This makes it a popular choice for fast experimentation and prototyping.

PyTorch is the work of developers at Facebook AI Research and several other labs. The framework combines the efficient and flexible GPU-accelerated backend libraries from Torch with an intuitive Python frontend that focuses on rapid prototyping, readable code, and support for the widest possible variety of deep learning models. PyTorch lets developers use the familiar imperative programming approach, but still output to graphs. It was released to open source in 2017, and its Python roots have made it a favorite with machine learning developers.

2.4. Experiment Outline

With each library from the overview above, we will build perceptron models. Each neural network will be trained, and training time and accuracy will be measured for a different number of training epochs. The goal is to compare the training time and accuracy/MSE of the two models. Keras on TF and PyTorch. We will use these measurements for comparative analysis of libraries. We will have 2 hidden layers, 1 input and output layers to train the neural network. The neural network architecture for the regression problem is also the same as below.



3. Data Sources

The first step of this paper was to determine the right data sets. Many online resources exist with access to a plethora of classification and regression datasets. Since this paper aims to solve a classification and regression problem. For this purpose, the two datasets that were chosen to benchmark the performance of machine learning libraries are as follows:

3.1. Classification Dataset

The problem of handwritten digit recognition by neural network from the MNIST database is chosen to get a comparative analysis of the machine learning libraries. Link for the dataset is given in the Datasets section.

The MNIST database is contained in a CSV file, the digits are written in comma-separated format. The initial value in a CSV file serves as a marker for the associated digit. The following value represents the size of the digit picture in pixels, which has 784 values and a 28x28 square dimension.

There are 60,000 copies of the training file and 10,000 copies of the test file. We decide to use the MLP (multilayer perceptron) architecture to solve the problem. Perceptron was one of the first models of neural networks, which was supposed to simulate the neural processes in human mind. This model was proposed by Frank Rosenblatt in 1957 and first implemented in 1960. A multilayer perceptron according to Rosenblatt differs from a single layer in that it contains additional hidden layers.

The test dataset contains 784 columns, each column representing a pixel of the image. The training dataset contains 785 columns, in addition to the pixels of the image there is a *label* column that classifies an image based on its pixel. This is also the target variable.

Table 1: Data types of columns

Column	Data Type	Range
label	Integer	0-9
pixel0	Integer	0-255
pixel1	Integer	0-255
…	…	…
pixel783	Integer	0-255

The goal is to predict the ***label*** variable and classify an image. The prediction accuracy and neural network training time is measured for a given number of epochs for comparative analysis of PyTorch and Keras on TF.

3.2. Regression Dataset

Several constraints were placed on the selection of these instances from a larger database data analysis is used to analyze and summarize the main characteristics of the data. The data analysis shows us various aspects and distributions of our data along with the analysis of different features within our data.

The neural networks are trained on this dataset for regression, results are benchmarked in terms to mean squared error and algorithm runtime against number of epochs of the two machine learning libraries.

1. *Petrol_tax*: the buying price of the car
2. *Average_income*: the maintenance cost
3. *Paved_Highways*: number of doors
4. *Population_Driver_licence(%)*: the seating capacity
5. *Petrol_Consumption*: the luggage capacity

The neural network by Keras on TF and PyTorch are trained on this dataset to predict the *Petrol_Consumption*. 5 models were trained after pre-processing of the dataset which was obtained from Kaggle uploaded by UCI Machine Learning organization. The dataset contains 768 records against 9 attributes i.e. *Petrol_tax, Average_income, Paved_Highways, Population_Driver_licence, and Petrol_Consumption*. 80% of the dataset was used for training and the remaining 20% was used for testing the trained model.

Table 2: Data types of columns

Column	Type	Data Type
Petrol_tax	Float	float64
Average_income	Integer	int64
Paved_Highways	Integer	int64
Population_Driver_licence	Float	float64
Petrol_Consumption	Integer	int64

The goal is to predict the ***Petrol_Consumption*** target variable to . The main technical challenge it poses to predicting Outcome is the high frequency of null values in data as it's medical data, so those null values are crucial for our analysis. The goal of this is to train the neural network to obtain the optimal MSE.

4. Experiment

The goal is to compare the training time and accuracy/MSE of the two models. Keras on TF and PyTorch

4.1. System Setup

The results were compiled by performing computation on a local machine combined with a GPU. The system setup and information used for all computations is as follows:

- Python 3 on Google Colab
- MacBook M1
- 16GB RAM
- Python 3 Google Compute Engine backend (GPU)

4.2. Benchmarking

4.2.1. Digit Classification Problem

- Epochs vs Model Training Time
- Epochs vs Training Accuracy
- Epochs vs Validation Accuracy

4.2.2. Linear Regression Problem

- Epochs vs Model Training Time
- Epochs vs Training MSE
- Epochs vs Validation MSE

4.3. Simulations

The results were compiled by calculating and neural network training runtime iteratively for epochs of size 1 to 20. The accuracies are the average accuracies for a given epoch.

For the Handwriting Digit Recognition Problem results were plotted for the number of epochs in comparison to model training time, training accuracy and validation accuracy. The simulation results for Keras on TF were as follows:

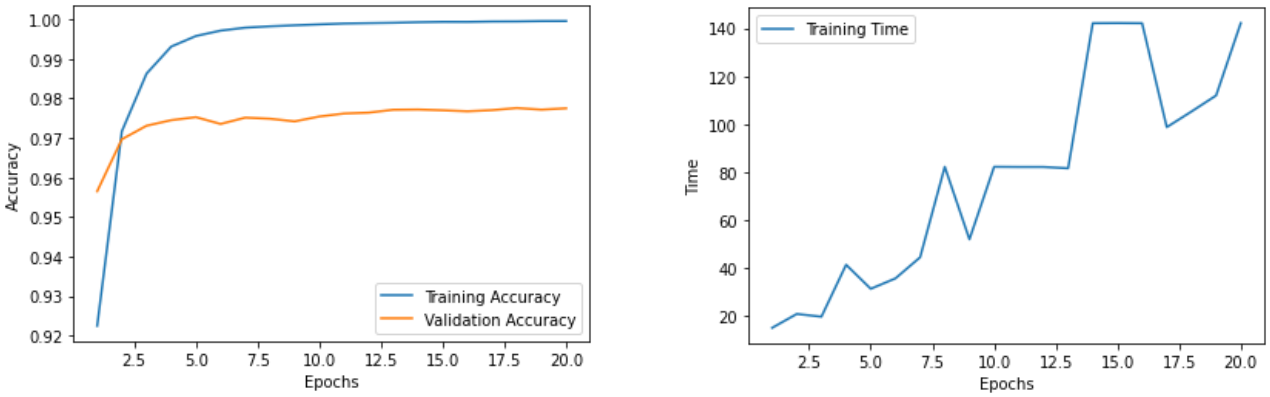


Figure 1: Keras on TF Digit Classification Results

The simulation results for Handwriting Digit Recognition Problem by PyTorch were as follows:

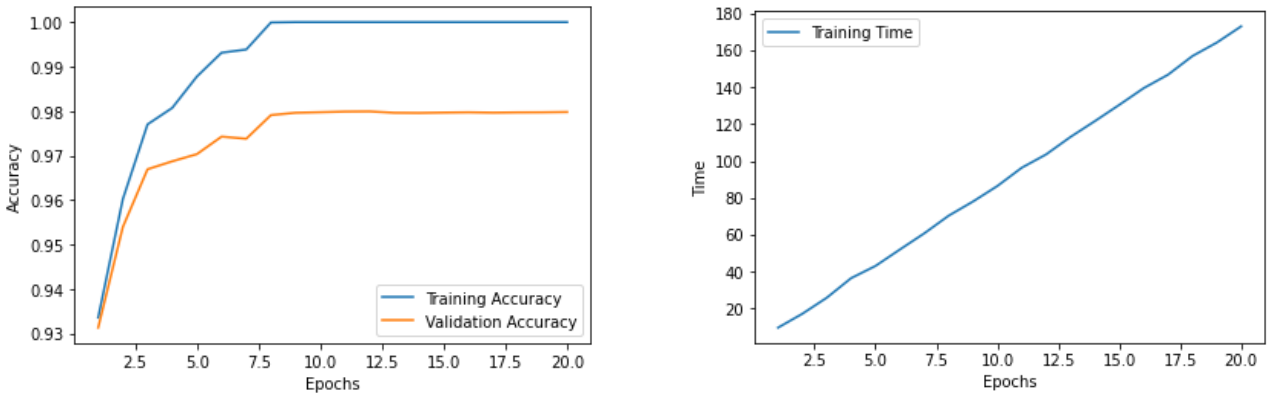


Figure 2: PyTorch Digit Classification Results

The results were compiled by calculating and neural network training runtime iteratively for epochs of size 1 to 30. The accuracies are the average accuracies for a given epoch.

For the Linear Problem results were plotted for the number of epochs in comparison to model training time, training MSE and validation MSE. The simulation results for Keras on TF were as follows:

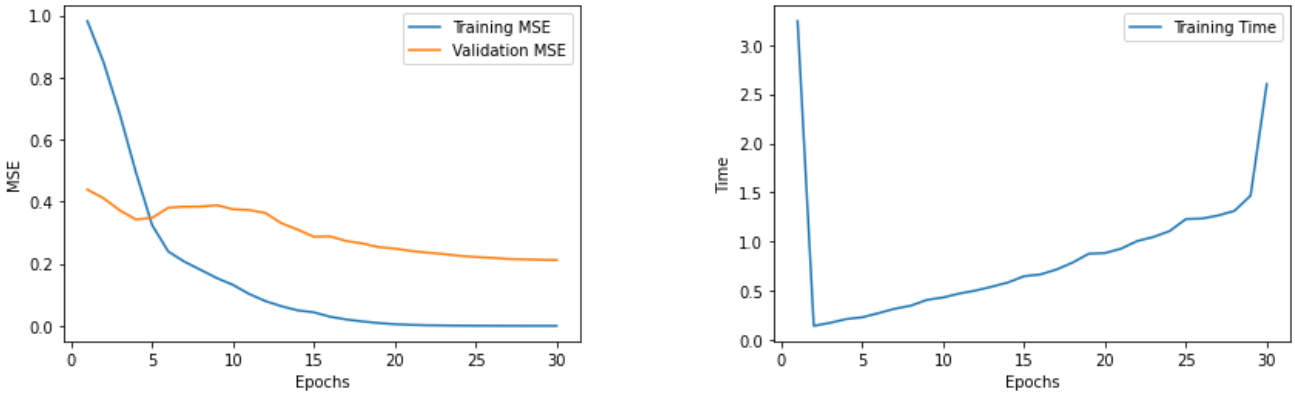


Figure 3: Keras on TF Linear Regression Results

The simulation results for Linear Regression Problem by PyTorch were as follows:

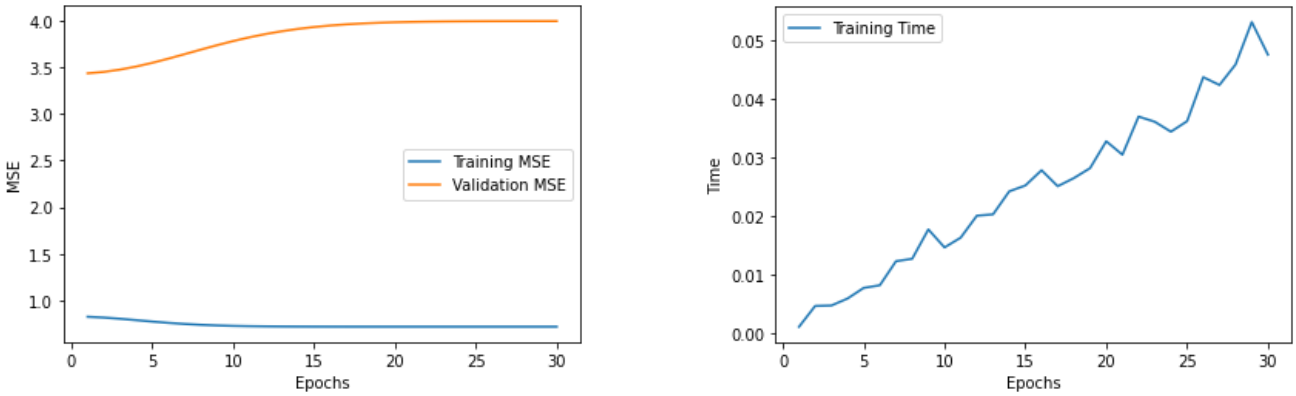


Figure 4: PyTorch Linear Regression Results

5. Results

5.1. Benchmark Results

Table 1: Digit Classification Results

Model	Training Accuracy	Validation Accuracy	Time(msec)
Keras on TF	0.993	0.976	140
PyTorch	0.999	0.980	180

Keras on TF has lower neural network training time however the training and validation accuracy of PyTorch is better as compared to the accuracy of Keras on TF model.

Table 2: Linear Regression Results

Model	Training MSE	Validation MSE	Time(msec)
Keras on TF	0.0000237	0.212	2.6
PyTorch	0.8148	2.795	0.05

Keras on TF has much lower training and validation MSE. Whereas training time for the PyTorch model is much less, but the training and validation MSE are much higher than expected.

Table 3: Winner Table

Model	Classification	Regression
Accuracy/MSE	PyTorch	Keras on TF
Training Time	Keras on TF	PyTorch

The Model with better results is displayed for each category in Classification and Regression Problem.

6. Conclusions

6.1. Conclusions

For the digit classification problem Keras on TF has lower neural network training time however the training and validation accuracy of PyTorch is better as compared to the accuracy of Keras on TF model.

For the Linear Regression problem Keras on TF has much lower training and validation MSE. Whereas the neural network training time for the PyTorch model is much less, but the training and validation MSE are much higher than expected.

6.2. Failure

The training and validation MSE of the PyTorch model are much higher than expected. This might be because of some wrong calculation of the mean or incorrect definitions of the neural network layers. Despite the best efforts to debug and rectify this issue I was unable to improve the MSE for linear regression using PyTorch neural network.

6.3. Future Work

For the future work the MSE of PyTorch model can be reduced within the acceptable range to get a better comparison between the models. We can add clustering and dimension reduction problems to further compare the two models. We should also perform this computation in a dedicated local or preferably a cloud environment for much accurate results.

7. Source Code

https://github.com/srkcheema/ComparativeAnalysis-ML-Libraries

8. Datasets

[1] http://yann.lecun.com/exdb/mnist

[2] https://www.kaggle.com/datasets/harinir/petrol-consumption

9. Bibliography and Credits

9.1. Research Papers

[1] Gevorkyan, Migran N., Anastasia V. Demidova, Tatiana S. Demidova, and Anton A. Sobolev. "Review and Comparative Analysis of Machine Learning Libraries for Machine Learning." Discrete and Continuous Models and Applied Computational Science 27, no. 4 (December 15, 2019): 305–15. https://doi.org/10.22363/2658-4670- 2019-27-4-305-315

[2] Ulker, Berk, Sander Stuijk, Henk Corporaal, and Rob Wijnhoven. "Reviewing Inference Performance of State-of-the-Art Deep Learning Frameworks." In Proceedings of the 23th International Workshop on Software and Compilers for Embedded Systems, 48– 53. St. Goar Germany: ACM, 2020. https://doi.org/10.1145/3378678.3391882

[3] Tizpaz-Niari, Saeid, Pavol Černý, and Ashutosh Trivedi. "Detecting and Understanding Real-World Differential Performance Bugs in Machine Learning Libraries." In Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis, 189–99. Virtual Event USA: ACM, 2020. https://doi.org/10.1145/3395363.3404540

[4] Dinghofer, Kai, and Frank Hartung. "Analysis of Criteria for the Selection of Machine Learning Frameworks." In 2020 International Conference on Computing, Networking and Communications (ICNC), 373–77. Big Island, HI, USA: IEEE, 2020. https://doi.org/10.1109/ICNC47757.2020.9049650

[5] Munjal, Rohan, Sohaib Arif, Frank Wendler, and Olfa Kanoun. "Comparative Study of Machine-Learning Frameworks for the Elaboration of Feed-Forward Neural Networks by Varying the Complexity of Impedimetric Datasets Synthesized Using Eddy Current Sensors for the Characterization of Bi-Metallic Coins." Sensors 22, no. 4 (February 9, 2022): 1312. https://doi.org/10.3390/s22041312.