A comparison of open-source real-time online learning frameworks for neural networks

Maurits van der Vijgh
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
m.vandervijgh@student.utwente.nl

ABSTRACT

With the increasing amount of data and the requirement to derive insights from this data, it is important to have the ability to update the models that are used with new data in real-time. This paper provides an analysis of existing solutions for real-time online machine learning using neural networks. Six different machine learning frameworks are examined. Quantitative analysis is performed on the following metrics: samples per second, predictions per second, the time between new data and inclusion in the model. Performance evaluation is carried out using a benchmarking framework that is created for this purpose and released as open-source software. In the qualitative comparison the following aspects are evaluated: preprocessing, normalization, hardware support, drift countermeasures, ensemble learning support and license.

Keywords

machine learning, online learning, real-time, neural networks

1. INTRODUCTION

Large data streams are generated by users, sensors and an increasing amount of connected devices, according to a prediction from the International Data Corporation it will grow to 175 trillion gigabytes by 2025 [12]. Neural networks can be used to make predictions based on this data to generate recommendations, predict when a device will fail, etc.. Currently training these models used for these purposes happens mostly offline with over 60% of companies questioned by Forrester Research stating that they want to implement machine learning for their real-time streaming data [9].

There are two approaches to training a model in machine learning, there is offline learning in which a model is trained using the complete dataset at once and there is online learning where a model can be incrementally trained when a new data-point arrives. Between offline and online learning there is a middle ground in which mini-batches are used, here incremental updates of the model are made by training on multiple data-points at the same time. Because of the incremental nature of learning using mini-batches they are considered to be online learning for this paper.

32th Twente Student Conference on IT Jan. 31nd, 2019, Enschede, The Netherlands.

Training a model offline is possible if the dataset fits in memory completely and is advantageous for performance reasons. In contexts in which there is no new data continuously arriving to update the model with or if there is little value in having an updated model at all times, offline learning is the preferred choice.

Real-time online machine learning can be employed to improve predictions quickly which can help reduce costs and increase the value gained from these predictions. In financial networks such as the ones run by large credit card companies the value of early fraud detection is readily apparent. With online learning, the model used for fraud detection can be updated in real-time when a fraudulent transaction is confirmed to improve the model to flag possible similar fraudulent transactions. At thousands of transactions per second, the prediction performance of the model is also relevant as it needs to keep up with the influx of transactions.

For computer vision applications however, a dataset can quickly exceed the memory of the computer the model is trained on. With the uprise of self-driving cars, the need arises to train computer vision models on vast amounts of data for recognizing and identifying roads, road signs, cars, cyclists, people, and other objects. For this purpose, the real-time aspect of online learning is not interesting but rather the decreased memory requirements of it.

In the existing literature, as outlined in 3.2, there is no comprehensive qualitative and quantitative analysis of open-source machine learning frameworks that can be used to select the right tool for the purpose of training neural networks in an online manner. Therefore in this paper, the current usability, performance, and features of open-source frameworks are analyzed and presented to serve as decision support for companies, researchers and developers looking to incorporate (real-time) online learning with neural networks.

Six machine learning frameworks are analyzed, these selection criteria for these frameworks are as follows:

1. Popularity, typically the more popular software is the more related online resources are available which makes it easier to use.

2. Open-source, this allows the software to be used in almost any context at no cost.

3. Supporting neural networks with various hidden layer configurations

4. Ease of integration in the benchmark suite

In an experiment devised to rank deep learning frameworks according to their popularity performed at the end of 2018 [1] it was found that the following frameworks are popular (in order or descending popularity): Tensorflow, Keras, PyTorch, Caffe, Theano, MXNET, CNTK, etc.. These are all open-source frameworks and they support neural networks with different hidden layer configurations. Keras is not itself a framework to execute machine learning, it is merely a way to program machine learning models for execution on different backends. Keras supports the following backends: Tensorflow, Theano, MXNET, and CNTK. For ease of integration into the benchmarking suite, programming the model once in Keras and executing it on all supported backends saves time and should ensure model consistency between the different backends. For these reasons all possible Keras backends are selected. PyTorch being the third most popular on the aforementioned list is selected as well as scikit-learn which as of January 2020 has more "stars" (similar to "likes") on GitHub than PyTorch, arguing for its popularity. The selected frameworks that can be found in table 1 on page 3.

2. RESEARCH QUESTIONS

In RQ1 we want to evaluate the supported software features in the selected open-source frameworks to train neural networks in an online and/or real-time fashion. This answer gives an overview of the currently available solutions and helps the reader narrow down the framework or frameworks they might use for their purposes.

To this end of interest is the general pre-processing and algorithm support to show versatility and range of supported features of the framework. Possible input formats and external system support will be investigated such that an informed decision for a framework can be made based on existing software and data it needs to integrate with. For the online learning specific behavior, the support of drift counter-measures and ensemble learning will be added to the table answering RQ1.

RQ1 What software features are supported and what characteristics do these frameworks have?

RQ1.1 What preprocessing, normalization features, concept drift detection algorithms, and ensemble learning methods do the selected frameworks support?

RQ1.2 How extensible are the selected frameworks? Can the features that are mentioned in RQ1.1 be added to a framework if currently lacks support.

If a framework lacks support for features mentioned in RQ1.1a, how extensible are the selected frameworks for adding missing features?

The second research question aims to give the reader a clear indication of training, prediction and real-time performance of the selected frameworks in a real life setup.

RQ2 How do the selected frameworks perform on the following metrics: model training time on the selected datasets, model prediction performance and latency in a system testing approach?

3. RELATED WORK

There is extensive existing research in the domain of online learning in which approaches, problems, possible drawbacks, and countermeasures are discussed. Additionally, benchmarks of frameworks for machine learning have been performed in the past, an overview of existing benchmarks is given. The possible countermeasures are used to inform the analysis of the features of the frameworks while the existing benchmarks are used to show what differentiates the benchmarks in this paper.

[1] `https://www.kaggle.com/discdiver/deep-learning-framework-power-scores-2018`

3.1 Online learning theory

Online learning is machine learning without training on the complete dataset at once, an online learning algorithm can train on a single datapoint, update its internal parameters and then discard the datapoint ready to process the next one. There exist surveys explaining the basic concepts and algorithms of online learning dating back to at least 1998 [7] [13].

A recent survey [11] specifically about online learning combined with artificial neural networks describes the problems associated with online learning and discusses possible solutions. The main problem with online learning is the phenomenon called concept drift, which happens when training on a dataset in which the patterns and relations shift over time. This can mean that the model becomes obsolete and strategies need to be used to deal with this problem. The survey mentions three main strategies: using sliding windows for selecting the data points on which to train the model, weighting the data points such that old data points are still influencing the model but not as much as newer data points and using ensembles of classifiers to make the final prediction.

The availability of features to detect concept drift and counteract its effects by means of ensemble learning are therefore evaluated in this paper.

3.2 Neural network frameworks comparisons and benchmarks

Benchmarks of artificial neural network frameworks have not been found in surveys or papers in the last 2 years as of 2019. However, papers discussing or introducing specific algorithms or frameworks with some benchmarks have been encountered.

Multiple benchmarking projects do exist on GitHub which are either outdated, non-comprehensive and/or do not evaluate online learning performance. In table 2 an overview is given. Another problem with most of these projects is reproducibility if the specific versions of the software are mentioned it might still be difficult to set up the environment in a consistent way to reproduce the results.

In this table outdated means that the benchmark is over 3 years old, a framework can improve to realize huge performance improvments in this time. Not comprehensive means that less than 5 popular frameworks were evaluated.

On Wikipedia a page with a comparison table of deep learning software exists, while it is quite extensive it does not contain information about features concerning preprocessing, normalization, online learning, extensibility and performance [1]. Therefore an enhanced version of that table is given which includes more practical information to help in picking a framework.

4. METHOD

To answer RQ1 the websites, documentation, and code of the selected frameworks are examined and basic examples are set up in each framework to evaluate and judge the ease of use and extensibility.

For answering RQ2 a neural network benchmarking frame-

Table 1. Evaluated machine learning frameworks

Name	License	Version	Rel. date	Written in	Language bindings
scikit-learn	BSD	0.22.1	2020-01-02	Python	Python
PyTorch	BSD	1.4.0	2020-01-15	Python, C++	Python, C++, Java
TensorFlow	Apache 2.0	2.1.0	2020-01-08	Python, C++	Python, C, C++, Java, Go, JS, R, Julia, Swift
CNTK	MIT	2.7	2019-04-19	C++	Python, C++
MXNet	Apache 2.0	1.5.1	2019-10-01	C++, Python	C++, Python, Julia, Matlab, JS, Go, R, Scala, Perl, Clojure
Theano	BSD	1.0.4	2019-01-15	Python	Python

Table 2. Existing benchmarks

Name	ref	comparison between	comment
Benchmark analysis of representative deep neural network architectures	[8]	NN architectures	no comparison between frameworks
guolinke deep-learning-benchmarks	[2]	NN frameworks	outdated (2017)
mlpack benchmarks	[3]	ML frameworks	not comprehensive, not online
sdpython _benchmarks	[4]	ML/NN implementations	not comprehensive (1 framework)
szilard benchm-ml	[5]	ML frameworks	outdated (2016)
u39kun deep-learning-benchmark	[6]	NN frameworks	not comprehensive (3 frameworks)

work is made. A benchmarking wrapper class abstracts the details of each specific framework exposing an initialization, train and predict method which can be called to initialize the model, update the model by training on a subset of data and predicting the class of a sample using the current model. In the wrapper a model of a deep neural network is implemented containing 1 hidden layer of 100 artificial neurons. Rectified linear unit (ReLU) activation is used for the artificial neurons and the Adam optimization algorithm is selected as the optimizer. In the output layer of the model, softmax activation is configured to make the model output its prediction. Keras is used to program the model for TensorFlow, Theano, MXNet and CNTK and the models for PyTorch and scikit are configured their respective APIs.

The dataset that is selected for benchmarking is the MNIST dataset [10], which is a very well known dataset consisting of 60000 handwritten digits to be used for training and 10000 images for testing the model. This is not a very large or challenging dataset, nonetheless it can be used to show the training speed of the different frameworks as training on this data is not trivial.

Three benchmarks are performed for speed evaluation:

1. Training speed: 1 data-point per model update benchmark, 1 epoch (iteration of the complete dataset)
2. Training speed: 8, 16, 32, 64, 128, 256, 512,1024, 2048, 4096, 8192 samples per model update, 3 epochs each. The internal (mini)-batch size of the model itself was kept the same at 200.
3. Prediction speed: Training the model for 3 epochs with 128 data-points per model update, then predicting the 10000 test images one-by-one. Training time is not measured.

In benchmark 1 we measure the training speed of a framework for the purpose of real-time online learning, the model update function is called for 1 data-point at a time. For use cases where the model needs to adapt to new data as soon as possible this should show the framework with low overhead on updating the model with new data. The framework that is the fastest on this benchmark will be the one that can deal with new data the quickest.

The outcome of benchmark 2 should yield the framework to be used for online learning when a dataset is too large to hold in memory, as it is assumed that by training on multiple data-points at a time the training speed will be higher than in benchmark 1.

Finally, benchmark 3 provides a value for the prediction speed of a framework for a setup in which the model can still be trained. There are ways of optimizing prediction speed of a static trained model, however for the purpose of real-time online learning this is irrelevant as the model must stay available to be updated with new data at any time.

To evaluate the real-time behavior of the frameworks the training on a sample or batch of samples is measured by timing before and after calling the model update function on a subset of the training data. This is training latency from which we derive the total training time. From the total training time, batch size and the number of epochs the throughput is derived in samples per second. The prediction speed in predictions per second is calculated in a similar way. The benchmarking setup uses Docker[2] to execute all the different components, this combined with strict version pinning allows reproducibility of the experiment.

For the purposes of the benchmarks, a virtual private server was rented at DigitalOcean of the following type: "CPU-Optimized / 8 GB / 4 vCPUs". The server is explicitly created for this purpose and the benchmarks are therefore performed without interference from other processes.

5. RESULTS

5.1 Framework feature support

An overview of the framework feature support can be found in 3 on page 4.

It can be seen that most frameworks do not support handling problems relating to online learning out of the box.

[2] `https://www.docker.com/`

In general, you need to use different libraries to address the concept drift problem in conjunction with the machine learning framework in use. Because detecting concept drift and applying ensemble learning can happen independently of actually training the model(s) its easy to integrate a solution external to the machine learning framework.

A common solution to lacking preprocessing and normalization support is using the functionality that scikit provides for these purposes and after those steps feed the data to the framework you want to use to train your model with. For ensemble learning and drift detection, TensorFlow has indirect support through the use of TensorFlow Extended. For the other frameworks other external libraries can be used, e.g. scikit-multiflow [3] and alibi-detect [4].

Table 3. Machine learning framework feature comparison

Name	Preprocessing	Normalization	Ensemble learning	Drift detection	GPU support
scikit-learn	++	++	Y	N	N
PyTorch	-	+	N	N	Y
TensorFlow	o	+	N	N	Y
CNTK	o	+	N	N	Y
MXNet	+	++	N	N	Y
Theano	-	+	N	N	Y

++ extensive support (3+ prebuilt options)
+ basic support (1-3 prebuilt options)
o building blocks provided
- abstract classes
– the concept does not exist
Y Yes
N No

5.2 Benchmarks

5.2.1 Training

The results of the first benchmark shown in figure 1 on page 4 for which the data was supplied to the model one data point at a time have two clear winners on both latency and training speed which is expected as the throughput is has a direct inverse linear relationship with the latency in this case. The winners are MXNet and TensorFlow. MXNet processed 278 samples/second with an average latency of just under 3.6ms. Scikit does not deal with this workload well at 47 samples per second and a latency of 21ms.

Figure 3 on page 4 shows the data of training runs at different batch sizes. It can be seen that increasing the batch size can massively improve the training speed. Tensorflow outperforms all other frameworks in this benchmark on raw throughput and latency across all batch sizes, the highest throughput it achieved was with a batch size of 8192 samples per model update. Theano seems to perform worse in this benchmark when given more data points at a time, at about 4600 samples/second and a batch size of 128 it hits a wall were the throughput just barely increases with increased batch sizes.

[3] `https://scikit-multiflow.github.io`
[4] `https://github.com/SeldonIO/alibi-detect`

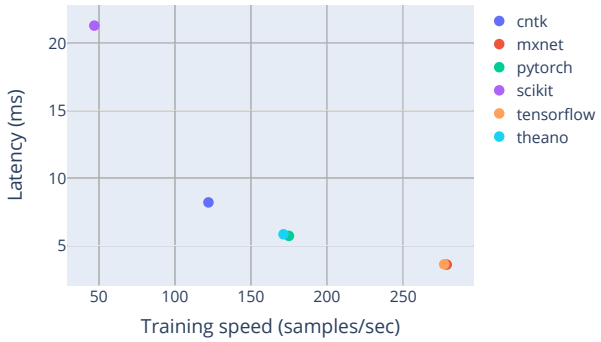

Figure 1. Training latency vs throughput, single datapoints

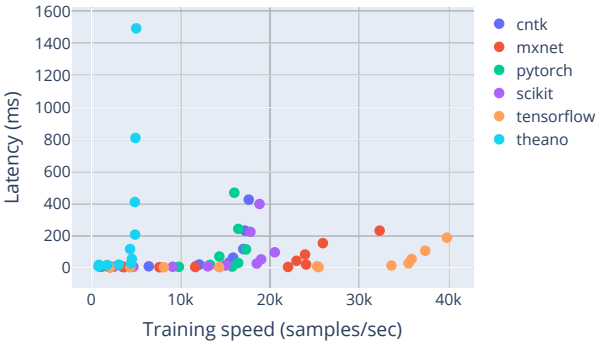

Figure 2. Training latency vs throughput, various batch sizes

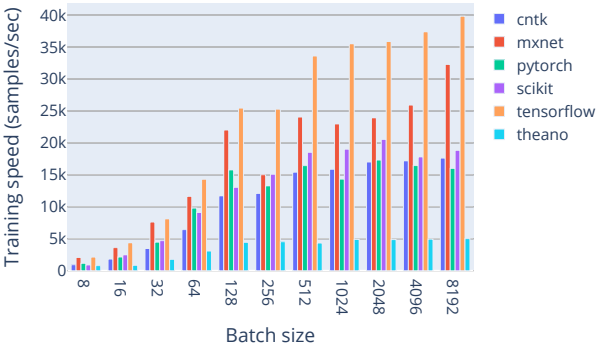

Figure 3. Throughput vs batch size (# samples)

Interestingly, the accuracy of the Keras defined model shows varies depending on the backend that executes the model. PyTorch and scikit consistently perform above 0.9 accuracy (accuracy being the proportion of predictions the model predicted correctly). One of the Keras backends, Tensorflow, also scored above 0.9 on every run, while

the other Keras backends (MXNET, CNTK, Theano) performed much worse with accuracies between 0.1 and 0.74. In figure 4 on page 5 the accuracies achieved at different batch sizes can be seen. Training the frameworks that achieve low accuracy with more epochs of the same data did improve the accuracy of the models, yet these frameworks do not seem significantly fast enough in training to achieve similar model accuracy by training more epochs in the same time to PyTorch, scikit or Tensorflow. No explanation has been found for the large variation between the Keras backends.

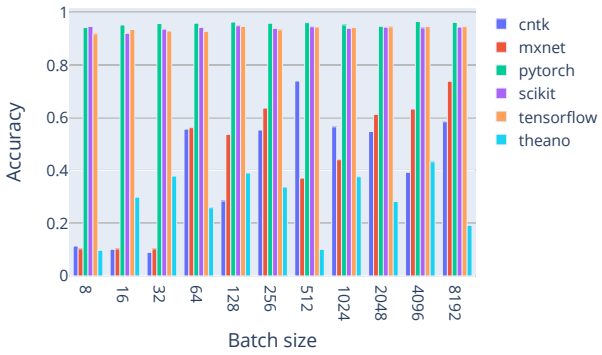

Figure 4. Accuracy vs batch size (# samples)

5.2.2 Prediction

A quick overview of the prediction performance in table 4 on page 5 in which the prediction speed, average latency and total time is displayed, shows that scikit performs particularly well here while Tensorflow, MXNet and especially CNTK are on the slow side. Batching the predictions could improve the throughput but in production systems the latency and performance on single predictions are important.

Table 4. Prediction performance

Name	Speed (predictions/s)	Latency (ms)	Total time (s)
scikit	2853	0.35	3.51
PyTorch	2494	0.40	4.00
TensorFlow	1295	0.77	7.72
CNTK	1013	0.99	9.87
MXNet	1345	0.74	7.44
Theano	2265	0.44	4.42

6. CONTRIBUTIONS

There are three contributions in this paper, firstly a table outlining the features and hardware support of the online neural network frameworks under investigation. This table can serve as decision support for companies, researchers and developers looking to incorporate real-time online learning using neural networks.

Secondly, to evaluate the performance of existing neural network libraries for online learning a benchmarking suite was developed and released as open-source software. Configuration to launch the benchmarks as a docker image which includes all necessary dependencies are provided. This allows the results to be verifiable as well as getting benchmark results for specific use cases and hardware. Included is a README file with instructions on how to run the benchmarks yourself or implement your specific use case to evaluate the performance of the different frameworks. New datasets, for example the ones found on the OpenML website [5] can be integrated into benchmarking suite with little work by adding the code to download and load the data. The benchmarking can be found on GitHub. [6]

Lastly the results of running the benchmark suite on various neural network frameworks as graphs accompanied by an analysis of the results.

7. LIMITATIONS

The main limitations to consider are related to the benchmarks performed. Ideally, the feature comparison part of the research could have had more time devoted to it to provide a more in-depth analysis of the frameworks.

7.1 CPU vs GPU

For deep learning purposes, GPU's can vastly outperform CPU's, this benchmarking study is performed on CPU's only. It is not the case that these workloads only are executed on GPU's in real life, however, real-time use cases or the higher expense for GPU's might mean that CPU's are used for these workloads. The benchmarking code does not prohibit running on GPU's however so if GPUs are the intended hardware for your model to be executed upon then it is advisable to adapt the benchmark to run on GPUs to get representative results.

7.2 Model diversity

In this benchmark, only one type of Artificial Neural Net was evaluated, specifically a feed-forward artificial neural network. However quick tests do show that the trends in the results generalize to running the model with different configurations of the hyperparameters and the configuration of the hidden layer(s). To improve diversity, benchmark models could be implemented for convolutional neural nets and recurrent neural nets.

7.3 Dataset diversity

In these benchmarks only one dataset is considered which consists of image data. Different types of data (audio or text for instance) should be considered to evaluate performance of the frameworks when training models with different input dimensions.

7.4 Keras and model equivalency

Four out of the six investigated frameworks have their models implemented using Keras as the frontend, this theoretically ensures that the model is similar for each of them, only this can impact performance as compared to programming the models directly in the API of the respective frameworks. The similarity of the models, in general, is a problem as it turned out to be difficult to program the same model and achieve similar model performance for each of the six frameworks.

8. CONCLUSION

There are good open-source options for picking a machine learning framework for training neural networks using an online learning approach. TensorFlow performed the best in the training benchmarks in throughput and accuracy

[5] `https://www.openml.org/`
[6] `https://github.com/mauritsvdvijgh/neural-net-benchmark`

while scikit performed the best in the prediction benchmark. Scikit has the best overall feature support but might not be optimal for use-cases where training performance is critical as it lacks GPU support. CNTK, Theano and MXNet were shown not to perform well in terms of accuracy out of the box. These benchmarks serve as an indication but more representative results can be obtained by benchmarking a specific use case using the benchmarking framework introduced in this paper. Future work could include making the benchmarking suite more comprehensive in model diversity and frameworks support.

9. REFERENCES

[1] Comparison of deep-learning software. `https://en.wikipedia.org/wiki/Comparison_of_deep-learning_software`. Accessed: 2019-11-28.

[2] guolinke/deep-learning-benchmarks. `https://github.com/guolinke/deep-learning-benchmarks`. Accessed: 2019-11-28.

[3] mlpack/benchmarks: Machine Learning Benchmark Scripts. `https://github.com/mlpack/benchmarks`. Accessed: 2019-11-28.

[4] sdpython/_benchmarks: Benchmarks on machine learning experiments. `https://github.com/sdpython/_benchmarks`. Accessed: 2019-11-28.

[5] szilard/benchm-ml: A minimal benchmark for scalability, speed and accuracy of commonly used open source implementations. `https://github.com/szilard/benchm-ml`. Accessed: 2019-11-28.

[6] u39kun/deep-learning-benchmark: Deep Learning Benchmark for comparing the performance of DL frameworks, GPUs, and single vs half precision. `https://github.com/u39kun/deep-learning-benchmark`. Accessed: 2019-11-28.

[7] R. Bank. Online learning and stochastic approximations. *On-line learning in neural networks*, 1998.

[8] S. Bianco, R. Cadene, L. Celona, and P. Napoletano. Benchmark analysis of representative deep neural network architectures. *IEEE Access*, 6:64270–64277, 2018.

[9] Forrester Research. Don't Get Caught Waiting On Fast Data. Technical report, 2018.

[10] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[11] B. Pérez-Sánchez, O. Fontenla-Romero, and B. Guijarro-Berdiñas. A review of adaptive online learning for artificial neural networks. *Artificial Intelligence Review*, 49(2):281–299, 2018.

[12] D. Reinsel, J. Gantz, and J. Rydning. Data Age 2025: The Digitization of the World From Edge to Core. Technical Report November, 2018.

[13] S. Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends in Machine Learning*, 4(2):107–194, 2011.