A Comparative Study of Deep Learning Frameworks Based on Short-term Power Load Forecasting Experiments

Jingyu Zhang*, Yongjun Zhang and Yanping Lu
Beijing Institute of Economics and Management, China
*Corresponding author’s e-mail: jingyuzhang@biem.edu.cn

Abstract. This article introduces the results of comparative research on mainstream deep learning frameworks, including PyTorch, Keras, Tensorflow, and MXNet. Analyzed their similarities and differences, and compared their efficiency through short-term power load prediction experiments.

1. Introduction
In recent years, the enthusiasm for deep learning research and application has continued to rise, and various open source deep learning frameworks have emerged one after another, including TensorFlow, Keras, MXNet, PyTorch, CNTK, Theano, Caffe, DeepLearning4j, Lasagne, Neon [1]. Google, Microsoft and other business giants have joined this deep learning framework war. The most mainstream frameworks are TensorFlow, Keras, MXNet, and PyTorch. I will briefly compare these four mainstream deep learning frameworks from several different aspects. Then I will use LSTM-based short-term power load forecasting experiments to compare the efficiency of the framework.

2. Features
TensorFlow is the second-generation artificial intelligence learning system developed by Google Brain based on DistBelief. Its name comes from its own operating principle [2]. It was released under the Apache 2.0 open source license on November 9, 2015.

Keras is an open source neural network library written in Python, which can run on TensorFlow, CNTK, Theano or MXNet. Aims to achieve rapid experiments of deep neural networks, it focuses on user-friendliness, modularity and scalability. Its main author and maintainer is Google engineer François Chollet.

MXNet is an open source, lightweight, portable and flexible deep learning library developed by DMLC (Distributed Machine Learning Community), which allows users to mix symbolic programming mode and instructional programming mode to maximize efficiency and flexibility. It is currently the deep learning framework officially recommended by AWS. Many of the authors of MXNet are Chinese, and its biggest contribution organization is Baidu.

PyTorch is an open source deep learning library for python released by Facebook on January 18, 2017, based on Torch. It Supports dynamic calculation graphs and provides great flexibility.

The comparison of some basic features of the four frameworks is shown in Table 1:
### 3. Popularity
The four deep learning frameworks are all open source. We can see their popularity in the industry through their data on Github. As of June 28, 2021, the data on Github is shown in Fig 1.

![Fig 1. Popularity of Frameworks](image)

### 4. Flexibility
TensorFlow mainly supports the form of static calculation graph. The structure of the calculation graph is relatively intuitive, but it is very complicated and troublesome in the debugging process, and some errors are more difficult to find. Although the dynamic graph mechanism Eager Execution was released at the end of 2017, adding support for dynamic calculation graphs, the original static calculation graph form is still used [3]. Moreover, TensorFlow has the TensorBoard application, which can monitor the running process and visualize the calculation graph.

Keras is a high-level API based on multiple different frameworks. It can quickly design and build models. It supports both sequential and functional design model methods. It can quickly turn ideas into results. However, due to the high degree of encapsulation, Modifications to existing models may not be so flexible.

MXNet supports both imperative and declarative programming methods, which is, it supports static calculation graphs and dynamic calculation graphs at the same time, and has packaged training functions, which integrates flexibility and efficiency. And it has launched an advanced interface Gluon with MXNet as the backend, just like Keras do.

PyTorch is a typical representative of dynamic calculation graphs. It is easy to debug and highly modular. It is very convenient to build models. At the same time, it has excellent GPU support, and the migration of data parameters between CPU and GPU is very flexible.

### 5. Difficulty of learning
The difficulty of learning and ease of use of the framework are still more important. I think it should be based on the language design of the framework itself, the level of detail of the documentation, and the scale of the technology community. As for the language design of the framework itself,
TensorFlow is relatively unfriendly, and has a large gap with languages such as Python. It is a bit like redefining a programming language based on a language, and it is more complicated when debugging. Every time the version is updated, the various interfaces of TensorFlow often have very large changes, which also greatly increases the learning time for it.

Keras is a high-level API, based on a variety of deep learning frameworks, pursuing simplicity, fast model building, easy to use, and can quickly realize what you want, it’s very suitable for entry or rapid implementation. However, excessive encapsulation leads to insufficient learning of deep learning knowledge and the rewriting of existing neural network layers is very complicated.

MXNet supports imperative programming and declarative programming at the same time. It’s very flexible, simple and convenient, and supports multiple languages at the same time, which can reduce the time to learn a new main language. The upper interface Gluon is also extremely easy to use.

PyTorch supports dynamic calculation graphs, pursues as little packaging as possible, the code is concise and easy to read, and the application is very flexible. The interface continues to use Torch, which has strong ease of use and can make good use of the various advantages of the main language Python.

Regarding the level of detail of the document, TensorFlow has very detailed official documents, which are very convenient to find, while maintaining a fast update speed, but the organization is not very clear and there are too many tutorials [4].

For the community, a huge community can promote the development of technology and facilitate the solution of problems. TensorFlow developed and maintained by Google has the largest community and a large group of application personnel; Keras has attracted a large number of researchers because of the simple implementation of the problem; MXNet is supported by giants such as Amazon, Baidu, and has attracted a large number of users with its perfect memory and video memory optimization; PyTorch is supported by Facebook and will soon be seamlessly connected with Caffe2. With its flexible, concise, and easy-to-use features, it has attracted a large number of developers and researchers within a year of launch. The popularity is still rising, and the community is also growing.

6. Performance
In order to compare the performance of the four frameworks (mainly operating speed), I used a short-term power load forecasting experiment based on LSTM, using real power load records in a certain area from 2012 to 2015 as a data set, and conducted speed tests on different frameworks.

Data set address:
https://github.com/hesoyam001/Electric-power-forecast/blob/main/data.txt

The following experiment time is the total training time, excluding the time required to read the data into the memory.

Model introduction: Long short-term memory (LSTM) is a special RNN, mainly to solve the problem of gradient disappearance and gradient explosion during long sequence training [5]. Simply put, compared to ordinary RNN, LSTM can perform better in longer sequences.
First, load the data set. Then all data will be normalized, and then the data set will be transformed into a supervised learning problem.

Track the training and test failures during the training process by setting the validation_data parameter in the fit function. At the end of the run, plot the training and test losses:

Forecast the electrical load in the last month:
Perform training for 50 epochs, count the time, and the experimental results are shown in Table2:

| Framework    | CPU(s)  | GPU(s)  |
|--------------|---------|---------|
| TensorFlow   | 15.8230 | 17.0182 |
| Keras(tf)    | 24.8000 | 21.0712 |
| MXNet        | 6.6413  | 4.8743  |
| PyTorch      | 150.4422| 25.3309 |

7. Conclusions

Through the above experiments, we can find out that different deep learning frameworks have certain differences in the optimization of computing speed and resource utilization: Keras is a high-level API based on other deep learning frameworks, which is highly encapsulated, has a slower computing speed and uses resources; in the case of complex models, large data sets, and large numbers of parameters, MXNet and PyTorch are excellent in optimizing computing speed and resource utilization on GPU, and MXNet optimization processing is even better in terms of speed; in contrast TensorFlow is slightly inferior, but TensorFlow performs better for computing acceleration on the CPU.

References

[1] Bastien F, Lamblin P, Pascanu R, et al. Theano: new features and speed improvements[J]. arXiv preprint arXiv:1211.5590, 2012.
[2] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, S. Ghemawat. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv preprint arXiv:1603.04467, 2016.
[3] Sanjay Krishnan, Jiannan Wang, Eugene Wu, Michael J. Franklin,and Ken Goldberg. 2016. ActiveClean: Interactive Data CleaningFor Statistical Modeling. PVLDB 9, 12 2016, 948–959.
[4] Krishnan S, Wang J, Wu E, et al. Activeclean: Interactive data cleaning for statistical modeling[J]. Proceedings of the VLDB Endowment, 2016, 9(12): 948-959.
[5] Apache Beam: An Advanced Unified Programming Model. https://beam.apache.org/, accessed 2017-06-05