A COMPARATIVE STUDY OF MACHINE LEARNING FRAMEWORK TENSOR FLOW (TF) AND PYTORCH

S. Singh[1] and E.B. Khedkar[2]
Dr D Y Patil School of MCA, Pune MS, India
Dr. D.Y. Patil School of Management, Pune MS, India
Sapan23k@gmail.com[1], ebkhedil.comkar@gmail.com[2]

ABSTRACT

As a part of defining a solutioning approach, we were required to select the best Deep Learning framework to suit a particular requirement. We considered the following factors; ease of implementation, shorter implementation time, ease of understanding, larger developer community, support, advanced feature list. Researcher has done competitive study of PyTorch with TF V 1.0 in terms of Model Build, Session and Variable Scoping, Symbolic and Derivative links, Debugging, Data Pipeline, Distributed Computing. Once the frameworks were identified, comparison of the framework (point-to-point, feature to feature) across versions was the next step. Future road map plans for both the frameworks were taken into account before taking a decision. Apart from the above comparison between frameworks, we also considered doing a comparison between multiple versions of the same framework to ensure the right selection of the best framework.

Keywords: *Machine Learning framework, Tensor Flow (TF) and Pytorch.*

Introduction

Deep Learning (DL) remains the hottest technology in data science. It has achieved exceptional momentum compared to any other technology in Data Science. Technologies that are based on Deep Learning has secured the place quite up in the ladder. It plays an important role in achieving the AI dreams, for firms, generating results that are superior to the state of the art in tricky and critical areas such as NLP and image (Computer Vision) processing.

Each Deep Learning framework has its unique characteristics, which is implemented to cater different purposes. They differ in the algorithms, quality and support in the implementation. Top players in this space includes PyTorch,Tensor Flow (TF), Microsoft Cognitive Toolkit/CNTKCaffe,Chainer,Keras, MXNet, and DeepLearning. These frameworks have evolved over a period with its unique capabilities.

Our initial approach for the study was to carry a high-level assessment of the available frameworks and shortlist the top two frameworks, giving consideration to the main areas that are listed below:

- Availability of pre-trained models
- Licensing model
- Connected to a research university or academia
- Benchmarks: Speed of inference, Speed of training
- Known large-scale deployments by notable companies
- Availability of the dedicated cloud optimized for a framework
- Engineering productivity
- Availability of debugging tools
- Compatibility (supported languages to write applications)
- Learning : Quality of the official documentation
- Open-source
- Supported Deep Learning algorithmic families and models
- Supported operating systems and platforms
- Computation Availability of CPU version optimized by Intel, Support for multiple CPUs, Horizontal scalability

The frameworks that were identified through the first level assessment were: Tensor Flow (TF) and Pytorch.

Research Methodology

Researcher has used experimental method. Comparison was done in multiple steps/stages. To make sure that our study is as comprehensive as possible, we did go through

multiple experiments (Multiple Approaches) using multiple datasets with enough variance and volume from different areas of Deep Learning (Computer Vision, NLP, etc.) and measure the performance of the frameworks. The same has been recorded in detail as part of this research Paper for reference.

- Feature wise comparison in detail, Comments & Observations.
- TF1.X to TF2.0 Conversion Experiment with Observations.

Feature wise comparison in detail: Comments and Observations The following areas were considered to compare. TF 1.0, TF 2.0 & PyTorch.

- Model Build
- Session and Variable Scoping
- Symbolic and Derivative links
- Debugging
- Data Pipeline
- Distributed Computing
- Model deployment

Comparison: Tenser Flow and Pytorch

The final qualitative and quantitative study and comparison: TF 1.0, TF 2.0 and PyTorch.

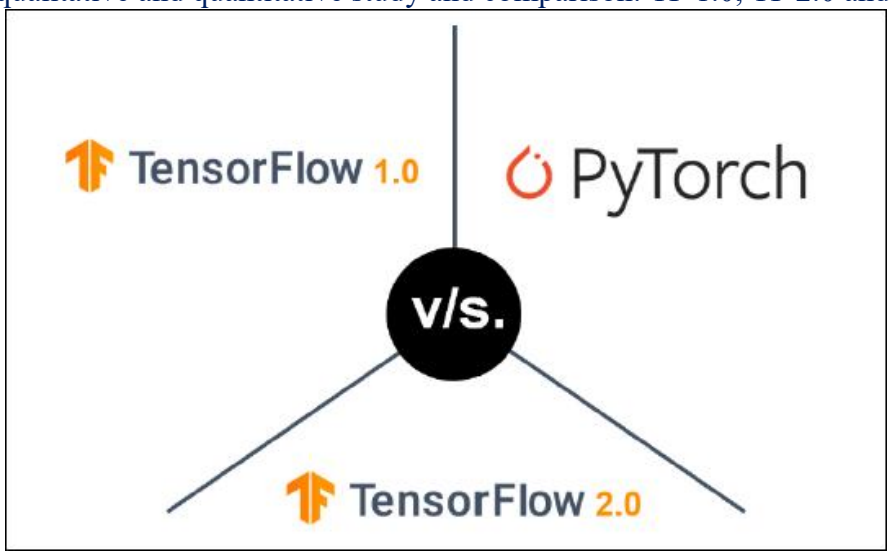

Figure 1: Qualitative and quantitative study and comparison: TF 1.0, TF 2.0 and PyTorch

Overall observation on the comparison summary – TF1.13, TF2.0, Pytorch

- Keras - TF 2.0, Deep Learning framework has an upper hand over a simple TF i.e. TF 1.0 and Pytorch. Being a high level implementation framework, it provides the following advantages:
  - Rapid prototyping
  - Speed of execution
  - Easy debugging
  - Multiple Back-end support.
- TF 2.0 carries the advantage of having both TF 1.0's low-level implementation and Kera's high-level implementation. This factor clearly makes Tensor flow 2.0 to be in the advantageous position.
- Compared to its nearest rival, this version reduces the gap with an improved user experience and features
- TF2.0 provides multiple levels of abstraction, which can suit any type of developer. For example: Like a researcher who requires a very low level API or a standard ML practitioner who expects a high level API to build and experiment on models as quickly as possible.

TF1.X to TF2.0 Conversion Experiment

As a first step towards understanding the complexity while migrating from the older version of Tensorflow i.e. 1.13 to the latest version 2.0, we identified solutions which were implemented using TF 1.0 (Computer Vison & NLP based) and efforts were made to migrate

that to TF 2.0. This experiment not only played a significant role in helping us in understanding the new set of features in TF 2.0 but it also helped us in analyzing the process, effort and complexities involved in migrating from one version to another. This experiment provided us clear insight on the added enhancement in TF 2.0. To achieve this, we followed the steps outlined in the TF2.0 conversion documents.

Overall observation from the experiment – TF1.X to TF2.0 Conversion

- Though upgrade script is easy to execute, the script makes only high-level changes to the old version of code. The remaining functional changes like replacing tf.Session.run calls, changing low-level variable etc. need to be performed manually.
- TF2 documentation gives out details at a very granular level. Most technical users understand only high-level information on supporting packages. This would make it difficult to rectify the issues faced when executing upgraded code.
- Though information about the code changes are provided, the exact module of code changes required in supporting packages used are not provided.
- For very old versions of tensorflow code, as per documentation, at least two upgrade steps are required. It cannot be directly converted to TF2.0
- TF2.0 is better than TF1.x when creating a new module since it uses less number of packages; the new packages used are also more efficient compared to old ones. TF2.0 also reduces major chunk of codes to abstract versions of it. However, conversion from TF1.x.

TF2.0 vs Pytorch Comparison

Overall observation from the experiment – TF2.0 – Pytorch Comparison

- Speed of execution: TF 1.x requires a computational graph to be built followed by creation of a Tensorflow session and finally running the session. This improves TF's speed of execution since the computational graph makes it possible for TF 1.x to execute extremely efficient through an interpreted set of instructions (if using Python). Pytorch, on the other hand, interprets instructions as it goes along, which has cost in terms of execution speed but is more flexible if one needs to modify the NN algorithm during execution, Whereas TF 1.x requires the entire computational graph to be recreated and a new session instantiated and run which makes it programmatically inefficient and complicated. TF 2.0 combines the best of both – the ability to create the computational graph for improved speed if needed, and the new eager execution mode allows instructions to be executed as they are encountered for better runtime flexibility.
- Ease of programming: Earlier Pytorch 1.0 had an ease-of-programming advantage over Tensorflow 1.x. it executed instructions right after they were encountered, which was intuitive for developers to understand. Tensorflow 2.0's Eager Execution mode has made a huge improvement in allowing instructions to be executed instantly without the requirement of creating a full computational graph first, and makes TF 2.0 superior to TF 1.x in this regard.
- Automatic utilization of all GPUs: Pytorch has a capability called Data Parallelism that allows any AI model to automatically run on the available GPUs in the machine. In Tensorflow 1.x, scaling the model across multiple GPUs requires a procedure to be followed, which may end up in mis-configuration if not handled carefully. . In TF 2.0, it is easier to scale the model to multiple GPUs automatically.
- Flexibility of API: Both TF 1.x and 2.0 both offer a level of flexibility in implementation that is not matched by Pytorch. TF 1.x as well as 2.0 have a rich API set, providing programmers with various choices for creating sophisticated neural networks.
- Learning Curve of API: The high flexibility of Tensorflow comes at a cost. Having worked with both Pytorch as well as TF 1.x and 2.0 alpha, Pytorch is still ahead of TF in terms of intuitive understanding and ease of use. The rich API of TF 1.x as well as

2.0 gives programmers various choices for accomplishing the same objective, which makes it harder for the programmer to decide on the best approach to go with. With Pytorch the library and API calls are fewer and simpler to understand. In TF's API (whether 1.x or 2.0), it is rather easy to get stuck, debugging an invalid parameter that was set, or to use the wrong API function, whereas with Pytorch there are fewer parameters in the function calls and the function names are more intuitive to understand.

- Debugging: To add to the above comment, Pytorch still appears easier to debug in Jupyter Notebook (or Pycharm, VS Code, etc.) than Tensorflow 1.x since one can process one statement (instruction) at a time and observe how the variables advance. However, with TF 2.0's eager execution, debugging it in Jupyter Notebook is now easier and more intuitive.

Conclusion

The overall summary of the above study concludes that, though PyTorch had been leading the race in comparison with TF V 1.0 in terms of Model Build, Session and Variable Scoping, Symbolic and Derivative links, Debugging, Data Pipeline, Distributed Computing, TF 2.0 (Alpha Version) is clearly ahead with Keras incorporation. TF 2.0 is more flexible and user friendly reducing the complexity and consumption of time and efforts. The outcome of the study recommends Tensor Flow framework for Deep learning.

References

1. Bansal A, Harit G, Roy S D (2014). Table Extraction from Document Images using Fixed Point Model. http://dx.doi.org/10.1145/2683483.268355
2. Gao, S., Wang, Z., Chia, L.-T., Tsang, I. W.-H. (2010). Automatic image tagging via category label and web data Proceedings of the International Conference on Multimedia -MM '10
3. Park J and Lee G (2008).A Robust Algorithm for Text region Detection In Natural scene Images. can.j. elect. comput. eng., vol.33, no. ¾, summer/fall 2008
4. Pise, A., & Ruikar, S. D. (2014). Text Detection and Recognition in Natural Scene Images. In Communications and Signal Processing (ICCSP), 2014 International Conference on (pp. 1068-1072). IEEE.
5. Ruiz M.E., Srinivasan P. (2008), Automatic Text Categorization Using Neural Networks. School of Library and Information Science, The University of Iowa. Advances in Classification Research. VIII. 59-66.
6. Verma R and Ali J. (2012). "A-Survey of Feature Extraction and Classification techniques in OCR Systems." Proceeding of the international journal of Computer Application and Information Technology, Volume 1, Issue 3, November 2012.
7. Wang, G., Hoiem, D., Forsyth, D. (2009). Building text features for object image classification. 2009 IEEE Conference on Computer Vision and Pattern Recognition.
8. Yue A (2018). Automated Receipt Image, Identification, Cropping and Parsing. Princeton University.0.