
Tech
Mahindra
Connected World. Connected Experiences.

Deep Learning continues to be the hottest technology

Hareesh Kumar Puthanmadom Seetharaman

Principal Solution Architect, COE Head - NLP

Date: 31 – March - 2020

Table of contents

Contents

Section 1

A study to understand how Tensor Flow (TF) got the edge over its competitor PyTorch after the release of its latest version TF 2.0 (TF 1.0 + Keras).

“Deep Learning (DL) continues to be the hottest technology in data science. It is gaining exceptional momentum compared to any other technology under Data Science. Deep Learning (based technologies has secured the place up in the ladder, as it plays a significant role in achieving the AI dreams, for organizations, producing results superior to the state of the art in critical and tricky areas such as image (Computer Vison) processing and NLP.

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

Background

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.

Our initial approach was to conduct a high-level assessment of available frameworks and shortlist the top two frameworks from the list, considering the main areas which are listed below the frameworks which were identified through the first level assessment: Tensor Flow (TF) and Pytorch.

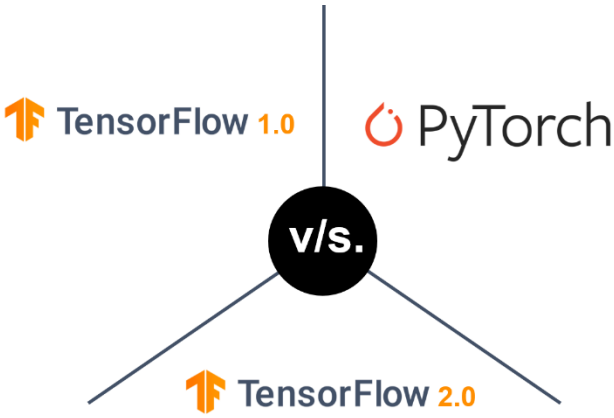
- Availability of pre-trained models	- Licensing model
- Connected to a research university or academia	- Benchmarks : Speed of inference, Speed of training
- Known large-scale deployments by notable companies	- Availability of the dedicated cloud optimized for a framework
- Engineering productivity	- 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

Section 2

Comparison: Tenser Flow and Pytorch

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

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



Comparison was done in multiple steps/stages. The aim was to help guide / help end users to make an informed decision about the best Deep Learning frameworks (Tensor Flow (TF) & PyTorch) which suits their needs and resources. 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 white 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

Areas	TF1.13<	TF2.0	Comments
Model Build	• tf.Layers was the primary package that was used for building the neural network. • The developer was required to develop the NN at a very low-level abstraction, which was rather time consuming and complex. • Multiple lines of code was required in order to create a simple NN architecture. • Rapid prototyping was difficult and time consuming • To build tensor graphs for complex calculations, a static graph had to be envisioned and built from scratch.	• With TF2.0 tf.keras an equivalent package is introduced to build NN. • The developer can build a NN with a very high level of abstraction and avoid unnecessary complexity. • A NN architecture can be built in a few lines of code. • Rapid prototyping is quick and very efficient. • With tf.function, the developer can concentrate on building the logic leaving the graph creation to TF to handle and build. • A hybrid approach is possible with TF2.0 where for complex customized calculations the lower level APIs of TF can be utilized and for a quick prototyping & build; the high level Keras API can be used	• Pytorch is user friendly, easy to debug and follows a well-defined structure. • TF2.0 reduces the gap. Keras provides a clean and rapid prototyping interface that can be used to build NN quickly, debug as it is developed and make NN architecture up and running just in a few lines of code. • TF2.0, in addition provides both means of completely building a customized graph when needed, as well as using Keras to build models rapidly. This feature provides the right level of abstraction as needed compared to Pytorch.
Session and Variable Scoping	• This method had relatively little abstraction for the developer. • The developer was required to build the NN, maintain the variables in	• The developer can concentrate on only building the model. • Any corresponding sessions, variables etc.,	• TF helps in removing sessions and unnecessary complexity of maintenance and variable scoping,

Areas	TF1.13<	TF2.0	Comments
	the form of placeholders and think in terms of sessions and variable scoping. • These aspects were really of no particular importance to a developer whose ultimate aim was to have a model built. • Unnecessary complexity was introduced into the whole build process.	are abstracted away by the high level API – Keras. • Eager execution provides a better way to play around and experiment with the API. • With a simple "import tensorflow as tf", one can very quickly build a NN, with minimum effort.	enabling undivided focus on the model built. • As a developer, all one needs to concentrate on is the Python implementation of one's logic and leave the rest to TF.
Symbolic and Derivative links	• Only symbolic links in the form of sequential and functional programing were possible.	• Both symbolic and derivative programing is possible, through sequential, functional and sub-classing. • The Model class can be sub classed to create a NN layered architecture. This provides an object oriented feel to the programing in TF2.0. • This provides the end user to choose the right level of abstraction needed as per requirement.	
Data Pipeline	• Since the graphs created were dynamic graphs, debugging and monitoring of the graphs were difficult and separate sessions were created to run the graphs	• With eager executions, the graphs are static in nature and can be debugged as they are being built.	• Easy debugging allows a developer to detect any errors as early as possible. • In addition to the features, TF provides a robust debugging tool compared to Pytorch.
Data Pipeline	• Variable management and data inputs were complex and needed to be in pre-defined feed_Dict formats.	• Data is treated as an ETL process that provides all the necessary tools to prepare, clean and normalize the data.	• Pytorch treats data inputs using simple methods and does not consider that the data has to be generally processed.

Areas	TF1.13<	TF2.0	Comments
		• In addition, the complex feed_Dicts are removed to be replaced with simple numpy or tf.data.Datasets objects.	• TF2.0 treats data as a pipeline, which allows in building a production grade code that can pre-process the data efficiently.
Distributed Computing	• TF1.X system was built to run and train a single node system.	• With 2.0 we can make use of cloud based computing to train on multi-node clusters having different execution modes such as CPU or GPU or TPU.	• Distributed computing can make a big difference in terms of utilizing cloud hardware's to reduce time and cost.
Model deployment	• TF1.X provided the means to save a model for either TF lite or JS depending on the requirement. • Developers needed to convert from one form to another in order to make it compatible as per the requirement.	• With TF2.0 standardizing the model saving format, developers can use the same model in various formats with minimum effort.	• Pytorch does not have an option to use models through mobile or client devices. • TF2.0 enables optimized models for each environment to be built with minimum effort. Developing production ready models becomes easy and efficient.

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.
- In addition, the TF2.0 library is far cleaner in structure compared to previous version. Multiple libraries that performed the same functionality have been removed (De- Duplication was done) making it simpler for the developer to understand and use.
- 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.

Section 3

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 to TF2.0 requires huge amount of manual work; if the upgrade script could handle a bit more complexity, it would have been more user friendly.

Section 4

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

https://software.intel.com/en-us/articles/hands-on-ai-part-5-select-a-deep-learning-framework

Disclaimer

Tech Mahindra Limited, herein referred to as TechM provide a wide array of presentations and reports, with the contributions of various professionals. These presentations and reports are for information purposes and private circulation only and do not constitute an offer to buy or sell any services mentioned therein. They do not purport to be a complete description of the market conditions or developments referred to in the material. While utmost care has been taken in preparing the above, we claim no responsibility for their accuracy. We shall not be liable for any direct or indirect losses arising from the use thereof and the viewers are requested to use the information contained herein at their own risk. These presentations and reports should not be reproduced, re-circulated, published in any media, website or otherwise, in any form or manner, in part or as a whole, without the express consent in writing of TechM or its subsidiaries. Any unauthorized use, disclosure or public dissemination of information contained herein is prohibited. Individual situations and local practices and standards may vary, so viewers and others utilizing information contained within a presentation are free to adopt differing standards and approaches as they see fit. You may not repackage or sell the presentation. Products and names mentioned in materials or presentations are the property of their respective owners and the mention of them does not constitute an endorsement by TechM. Information contained in a presentation hosted or promoted by TechM is provided “as is” without warranty of any kind, either expressed or implied, including any warranty of merchantability or fitness for a particular purpose. TechM assumes no liability or responsibility for the contents of a presentation or the opinions expressed by the presenters. All expressions of opinion are subject to change without notice.

Thank You

Visit us at techmahindra.com