c_izefak: TensorFlow VS PyTorch: A Deeper Dive

Dyuti De, Anusha Godavarthi, Yash Katariya, Sri Sai Bhargav Tiruveedhula
Deptartment of Computer Science and Department of Electrical and Computer Engineering
North Carolina State University
Raleigh, NC, USA
{dde, agodava, yskatari, stiruve}@ncsu.edu

***Abstract*—In this project we compare two leading machine learning frameworks; TensorFlow and PyTorch. We use various metrics like performance, code quality, documentation health and cross platform deployment to compare them. According to our findings, TensorFlow outperforms PyTorch in almost every metric and thus we recommend TensorFlow as the go-to ML framework.**

I. Overview

TensorFlow and Pytorch are both open source software libraries for machine learning which have been created by two different wizards - Tensorflow is based on Theano and has been developed by Google, whereas PyTorch is based on Torch and has been developed by Facebook.

TensorFlow and PyTorch both use Python libraries and expose convenient application programming interfaces but there are significant differences that are to be explored over the duration of this project when it comes down to their adoption on platforms, community health, performance, and documentation.

To capture a well-rounded result, we apply various tools and target different metrics to determine which framework works the best depending on different use-cases a common user might have.

A. Tools Applied

To answer our research questions - which are explained in the following section, we take help from certain widely available tools.
Tqdm - Tqdm [3] predicts the time remaining to train a model and skip unnecessary iteration displays. This tool would be used to compare the model training times on TensorFlow and PyTorch.

FOSS Heartbeat - We have used an open source software [4] and codes to understand the community health and code performance of TensorFlow and PyTorch models. It helped us create interactive dashboards that you would tune to understand the results better.

Docker - We would use Docker to create an image of the trained model and deploy it [6] on various devices and test if the trained model works on various devices or not. For example, mobile, raspberry pi, websites, etc.

Survey of documentation - The survey would be done using Google forms as a tool to survey a sample of about fifty students from different backgrounds (e.g networking, machine learning, non-technical, security, non-native speakers) to measure the effectiveness of Pytorch and Tensorflow documentation.

B. Metrics Targeted

- Time taken to train a model on a GPU and CPU
  Machine learning or deep learning tasks can become unfeasible for lack of computing power. This measure would identify how much time is taken by both libraries to train a model on a CPU and GPU.
- Statistics for contributors
  TensorFlow and PyTorch benefits from contributions and interactions across the community since they are open source. Through this data, we plan to derive the developer community statistics based on commit counts, trending repositories, the number of identifiable male or female developers actively contributing to projects, etc. We could possibly identify any implicit biases pervading the community using this measure.
- Cross-Platform Deployment
  This measure is to assess the usability of TensorFlow and PyTorch trained models on different devices like mobiles.
- Documentation
  This measure lays the foundation for quality, traceability, and history for TensorFlow and Pytorch and it is important that the documentation is well arranged, easy to read and adequate for the users. Hence, it is vital to community growth that there is proper documentation in place and thus, it is an important measure for the success and popularity of both TensorFlow and PyTorch in the global developer community

All the homegrown scripts along with the tool outputs can be found in our Github repository. The folders are organized as per the tools used and they contain 'requirement.txt' file or 'README.md' for the ease of setup. [1]

II. Approach

Our approach was focused towards a way through which we could help different groups of user choose between the 2 frameworks. Our work surrounds the following research questions.
For the developers using Pytorch and Tensorflow:
R1: How helpful is the documentation to perform meaningful

[1]https://github.com/reCursedd/SE-18

tasks?
R2: How well does the models created by these frameworks deploy?
R3: What is the time taken to train a model?
For the present and future contributors:
R4: What are the general pattern in code contribution?
R5: What is the rate of code contributed?

To understand **R1** we had rolled out a survey for evaluating the efficiency of documentation of both the open source machine learning frameworks- Tensorflow and Pytorch. We have obtained around 35 responses from various backgrounds (Computer Science, Computer Networking, Electrical Engineering). [1]. We ask 6 survey questions for an all-round result.

For **R2** For **iOS** The tfcoreml tool allows developers to convert Tensorflow models to CoreML models and for PyTorch, the ONNX (open neural network exchange) format is used as an intermediary to convert to CoreML. We have considered the compatible Inception v1 model for Tensorflow to CoreML conversion which contains a .pb file which is the model and .txt file that has the class labels. It was critical that correct operators needed to be passed and images needed to be pre processed. To achieve that, a text summary of the model was created and the operators that corresponded to the output feature name was selected. A converter python script was created and executed. A CoreML model was successfully generated. It opens in an Xcode IDE upon being clicked and it can be imported into the iOS project setup that creates an app to predict the image labels and runs it on an Iphone simulator.

For Pytorch to ONNX conversion, we ran a script to generate the pytorch model along with a blank image of the resolution we want. The blank image is used to export the PyTorch models (.pth files) which generate ONNX format(.onnx) model . After applying the neural-style.py script , we get the .onnx files. After this, we had to execute convert-onnx-to-coreml script which had an error upon execution and the conversion failed. There was no proper documentation to debug the segmentation error.

We used docker for cross platform deployment of pytorch and tensorflow models. We have tested for **MacOs** and **Ubuntu**. Both the frameworks have been successfully deployed.

For answering **R3**, we compare Pytorch with 2 variants of TensorFlow on the tool tqdm. First one is TensorFlow Eager which creates a dynamic graph similar to Pytorch. The second one is using Defun with TensorFlow. Defun returns a callable graph which can then be executed. Finally we note down the time taken for data ingestion and to train two different models.

R4 and **R5** were answered by analyzing the Github communities of each project. We look at various interesting measures like 'interactions of newcomers' and 'activity of the contributors'. We wanted to look for code-smells in the projects, but it is still a very vague and subjective notion to decide what exactly makes a code smell. [1] and even extensive machine learning techniques do not perform entirely accurately [2] and are very costly. Thus we look at the code decay or amount of re-factoring that occurs in the respective code bases since the start of projects to understand the quality of past code and analyze the rate of growth of code as well using an interesting visualization. It shows us the efforts made by the communities to keep the code bug-free and add new features.

III. Conclusion

For **R1**, from the results obtained from our survey form, we find that Tensorflow out-preforms Pytorch on all following questions with margins shown in Appendix:
(1) Which framework had a better accelerator specific documentation? IV-A[2]
(2) Which framework had a better documentation of APIs? IV-A[3]
(3) Which Framework had a better explanation of its debugging techniques? IV-A[4]
(4) Which framework had better tutorials to learn and use ML? IV-A[5]
(5) Which framework had a better documentation of it's ease of deployment on real-world models?

For deployment purposes **R2**, both PyTorch and TensorFlow have been successfully deployed as docker images on Linux and Mac IV-A[7]. We have obtained iOS deployment results for Tensorflow IV-A[6] but we are not very sure for Pytorch since ONNX to CORE ML was faulty. We conclude that for cross-platform deployment, both frameworks are good choices but it was easier to do it with TensorFlow.

For **R3**, as we can see from the observations IV-A, Tensorflow outperforms Pytorch in terms of performance on a CPU and a GPU. We havent tried it on a TPU, since PyTorch doesnt support TPU yet, so thats a win for TensorFlow. In terms of data ingestion i.e. time taken to loop over the entire dataset, TensorFlow outperforms Pytorch there again.

For **R4**, we have created a working dashboard to roll out statistics from both the Github projects. We can draw in several insights from the visualizations. We have compared measures five measures to help a new-comer choose a community. Our findings are: Pytorch is more suited for a new-comer because it takes lesser time for a new-comer to get used to the community and merge a pull request. IV-A[7] and new-comer feel more confident to open an issue in Pytorch. Also when we look at the contributor activity, it looks like Pytorch triages its bugs and issues faster that Tensorflow IV-A[9].

For **R5**, we look at the code decay to understand how fast the existing code is getting refactored and new features are being added. We find that both are almost as good but Pytorch community is performing slightly faster IV-A[10] we also look at the code addition stack to understand the growth of rate of growth - where we find that Tensorflow is growing with a more stable curveIV-A[11]

A. Limitations and Future Work

It would be interesting to perform bigger surveys on usability and documentation. We can perform research on metrics like static code analysis, sentiment analysis on comments and performance of different models like Neural Networks or Graphs

IV. Results

A. Figures and Tables

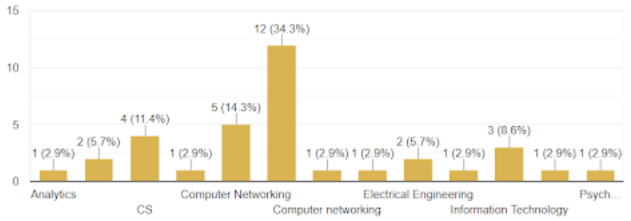


Fig. 1. Types of surveyors

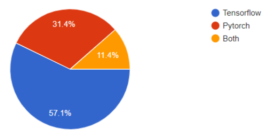


Fig 2: Better Accelerator

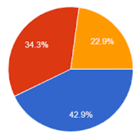


Fig 3: Better API

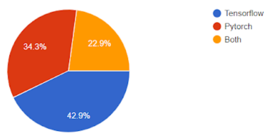


Fig 4: Debugging Techniques

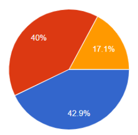


Fig 5: Better tutorials to learn and use ML:

Aspect	Metric	Result	Explanation
Training time - CPU	Performance	TensorFlow Eager=86 sec TensorFlow Defun=79 sec Pytorch = 113 sec	On the CIFAR10 dataset: Conv(32) -> Maxpool -> Conv(64) -> Maxpool -> Dense(128) -> Dense(10)
Training time - GPU	Performance	TensorFlow Eager=15 sec TensorFlow Defun=15 sec Pytorch=15 sec	On the CIFAR10 dataset: Conv(32) -> Maxpool -> Conv(64) -> Maxpool -> Dense(128) -> Dense(10)

Data Ingestion Rate	Performance	TensorFlow = 7.15 sec
		Pytorch = 8.9 sec
			Time to loop over the entire MNIST dataset

Training time - CPU	Performance	TensorFlow Eager=70 sec TensorFlow Defun=66 sec Pytorch = 110 sec	On the MNIST dataset: Conv(32) -> Maxpool -> Conv(64) -> Maxpool -> Dense(128) -> Dense(10)
Training time - GPU	Performance	TensorFlow Eager=21 sec TensorFlow Defun=17 sec Pytorch=22 sec	On the MNIST dataset: Conv(32) -> Maxpool -> Conv(64) -> Maxpool -> Dense(128) -> Dense(10)

Table 1: Performance Table

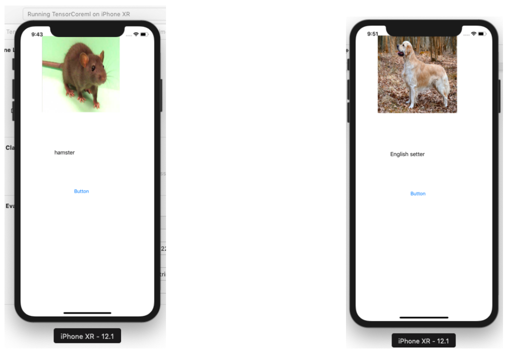
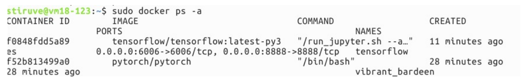


Fig 6: iOS deployment of models using Tensorflow



```
stiruve@vm18-123:~$ sudo docker ps -a
CONTAINER ID        IMAGE                              COMMAND                  CREATED
                  PORTS                                            NAMES
f0848fdd5a89        tensorflow/tensorflow:latest-py3   "/run_jupyter.sh --a…"   11 minutes ago
es                0.0.0.0:6006->6006/tcp, 0.0.0.0:8888->8888/tcp   tensorflow
f52b813499a0        pytorch/pytorch                    "/bin/bash"              28 minutes ago
28 minutes ago                                                     vibrant_bardeen
```

Fig: 7: Command line output for successfully docker deployment

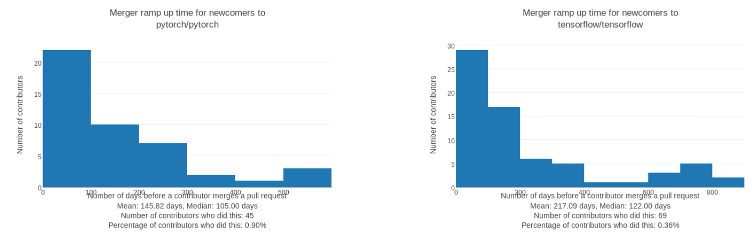


Fig 8: For newcomers

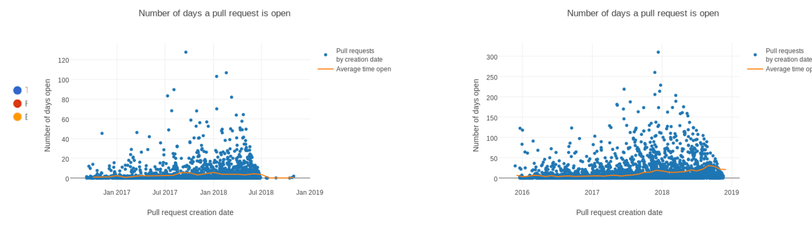


Fig 9: Issues getting triaged faster in Pytorch

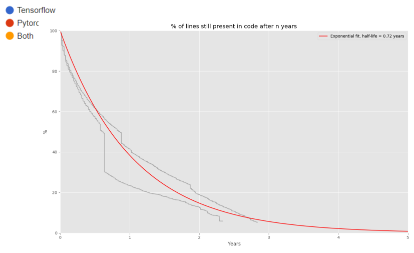
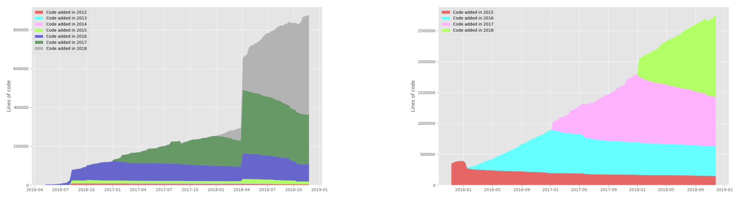


Fig: 10:Decay of existing code



References

[1] Bowes D, Randall D, Hall T (2013) The inconsistent measurement of message chains. Proceedings of the 4th International Workshop on Emerging Trends in Software Metrics (WETSoM 2013). IEEE, San Francisco, CA, USA, 6268

[2] F. Arcelli Fontana, M. V. Mntyl, M. Zanoni, and A. Marino, Comparing and experimenting machine learning techniques for code smell detection, Empirical Software Engineering, vol. 21, no. 3, pp. 11431191, 2016.

[3] tqdm4.28.1 [https://pypi.org/project/tqdm/]

[4] Mohammad Azam *Integrating TensorFlow Model in an iOS App*[https://hackernoon.com/integrating-tensorflow-model-in-an-ios-app-cecf30b9068d], Medium 2012

[5] Sage Sharp *Foss-Heartbeat* [https://sagesharp.github.io/foss-heartbeat/]

[6] Mohammad Azam*Integrating TensorFlow Model in an iOS App*[https://hackernoon.com/integrating-tensorflow-model-in-an-ios-app-cecf30b9068d], Medium 2012]