Analysis of Criteria for the Selection of Machine Learning Frameworks

Kai Dinghofer, Frank Hartung
FH Aachen - University of Applied Sciences
Aachen, Germany
kai.dinghofer@alumni.fh-aachen.de, f.hartung@fh-aachen.de

***Abstract*—With the many achievements of Machine Learning in the past years, it is likely that the sub-area of Deep Learning will continue to deliver major technological breakthroughs [1]. In order to achieve best results, it is important to know the various different Deep Learning frameworks and their respective properties. This paper provides a comparative overview of some of the most popular frameworks. First, the comparison methods and criteria are introduced and described with a focus on computer vision applications: *Features and Uses* are examined by evaluating papers and articles, *Adoption and Popularity* is determined by analyzing a data science study. Then, the frameworks TensorFlow, Keras, PyTorch and Caffe are compared based on the previously described criteria to highlight properties and differences. Advantages and disadvantages are compared, enabling researchers and developers to choose a framework according to their specific needs.**

***Index Terms*—Machine Learning, Deep Learning, Computer Vision, Comparison, TensorFlow, Keras, PyTorch, Caffe**

I. Introduction

A. Motivation

Machine Learning (ML) is not only a buzzword – it is a big milestone in computer science. The sub-area *Deep Learning* (DL) has been constantly at the peak of the *Gartner Hype Cycle* graph starting from 2015 until today which indicates that the technology is future-proof and not short-lived [2]. It has already been adopted and actively used in many areas. One of them is *computer vision* via image analysis and recognition, which can be applied to many real-life scenarios.

From a technical perspective, a developer has to choose between various different frameworks. This paper aims at providing help to find the best suited one with a focus on computer vision applications by using *Convolutional Neural Networks* (CNN). Data scientists as well as multimedia scientists and developers can subsequently make a decision based on the results of this conference paper by balancing the pros and cons of the different frameworks.

B. State of the Art

Complex ML models enabled ML-based technologies to find solutions e.g. for "perceptual problems" that have not been possible for machines before [3]. Regarding Deep Learning there have been many advances since 2010. One of the most important steps for DL and CNNs was the *AlexNet* introduced by Krizhevsky et al. in 2012. It is an architecture working on Graphics Processing Units (GPU) for convolutional operations consisting a total of eight layers. This CNN competed in the 2012 *ImageNet Large Scale Recognition Challenge* (ILSVRC) which is a benchmark for image classification that started in 2010 [4].

In 2015, *DeepMind* introduced an agent that is able to play better than average human players in 49 *Atari* games by receiving only two types of inputs; the pixel and the current score of the game, i.e. not more inputs than the human counterpart [5].

At the beginning of 2017, Bansal et al. showed *PixelNet* which is a CNN concept (composed of multiple design principles) that uses pixels as representations for pixel prediction problems [6].

In the recent two years, DL is quite common in everyday life. While often working in the background as a passive component (e.g. better ad targeting or improved user-specific search results) human users also actively decide to interact with AI and thereby also with DL by using digital assistants (for instance *Amazon Alexa*) [1]. In the specific field of image classification, DL is e.g. used in machine translation which is improved by combining it with image recognition (e.g. *Google Translate* [7]) or to enable autonomous driving.

II. Methodology of Comparison

A. Features and Uses

To effectively compare the features and the functionality of the frameworks, it has to be differentiated between the advertised features that have not yet been implemented and functionality that is already available in the latest released version. Therefore, articles and papers that follow a practical approach are preferred as a source because they expose the potential most realistically. First, each framework will be generally introduced. Then, the implemented and exclusive features will be mentioned which can often be mapped to a specific field of application. Some frameworks offer specialized features that aim to significantly ease development. Where applicable, training performance for CNN models will be compared too.

B. Adoption and Popularity

It is usually a good sign when a technology is widely adopted and used. For instance, an active user base gives developers the ability to openly discuss issues and get help by exchanging solutions. Additionally, example code or libraries

can be used for learning purposes to implement software in less time (also resulting in lower development costs). For this aspect, data science is used; J. Hale has developed a method to find useful adoption indicators in his 2018 study [8] – *usage*, *interest* and *popularity* which were summarized to the term "Power Score".

III. Comparison of the Frameworks

All of the investigated frameworks support the needed standard routines for CNNs which are a crucial element for image classification to perform *Computer Vision* (CV) related tasks and applications.
Because videos are simply consecutive frames (single images), all of the following compared frameworks can be used for video processing as long as they are suited for the more generic use case of image recognition when effectively combining them with computer vision library such as *OpenCV*. In general, GPUs (Graphics Processing Units) are strongly preferred over CPUs (Central Processing Units) for CNNs and computer vision applications due to the performance gain [1] [9]. Concerning performance improvements via hardware acceleration, all of the four compared frameworks support GPU-acceleration via the *NVIDIA cuDNN* (*CUDA* Deep Neural Network) library which has been released in 2014 [9] and provides highly optimized access to the standard routines used in Deep Neural Networks. This is a relevant aspect for the development since shorter training times lead to shorter testing times likely resulting in better overall CNN models. *CUDA* dominates *OpenCL* (its Open-Source non-proprietary variant) [9] which can be linked to *NVIDIA's* extensive research and investments into DL [1]. All of the investigated frameworks support the popular Python programming language.

A. Features and Uses

1) TensorFlow: TensorFlow is a machine learning framework publicly introduced by a deep learning research team called the *Google Brain* in 2015 [9] that is intended to be used in large scale system environments using *dataflow graphs*. This is accomplished by enabling the developer to use parallelization schemes and sharing the state across multiple machines and devices. It can run on multicore CPUs, GPUs and also specialized hardware units for Machine Learning known as TPUs (Tensor Processing Units) [3].

TensorFlow offers an extensive documentation and many tutorials (often with a focus to industrial and research applications) [10]. For visualization, *TensorBoard* can be used making it possible to display the TensorFlow graph including images which is helpful to compare between different training runs of a model and which can be utilized as a powerful debugging tool [10]. With extended features such as distributed training and specialized deployment options for the models via *TensorFlow Serving*, TensorFlow can be considered as very flexible for developers who want to use the full potential of their DL models and also deploy on mobile devices [10].

Because of TensorFlow's origin, a key task for the framework is image classification. From the beginning, it has been engineered with computer vision applications and performance in mind [3].

Figure 1. Step-by-step Google translation using letter recognition [7].

This core focus is also demonstrated in the *Google Translate* application which is equipped with a trained CNN for letters. As seen in figure 1, the app is able to find letters, filter out and recognize them in images for translating purposes. The CNN has been made "smaller" and more efficient so that it can also be executed – even locally – on mobile devices running on limited resources (e.g. smartphones) [7].

2) Keras: In direct comparison to TensorFlow, Keras is a higher-level library released in early 2015 [1] and sits on top of a backend that handles the low-level actions. Keras is used as a model-level library for a specified backend (that can be TensorFlow itself) – comparable to a "frontend" made for ML developers. The backend can be exchanged at any time without changing the Keras model implementation, thus even enabling the developer to test and use multiple backends which can also be useful to compare performance among the different backends (additionally to TensorFlow, the *Microsoft Cognitive Toolkit* and *Theano* are supported) [1]. When using TensorFlow as a backend, Keras can be used to run on multiple processing units seamlessly (such as GPU and CPU) and is therefore qualified to use CNNs for computer vision applications [1].

Kaggle.com is a popular website for ML contests for various problem formulations (two basic types; shallow learning problems and perceptual problems) which offers high-valued prizes to the winners. This site can be used to examine the current ML landscape and to estimate what is practically (im)possible with ML technology. Keras is one of the most used library by entrants specialized in the field of DL (to solve perceptual problems like image classification) on *Kaggle.com* [1].

After all, Keras' purpose is to make DL easier and more usable for everyone by providing a user-friendly and high-level library that abstracts the often more complicated code constructs of its backends [1].

3) PyTorch: PyTorch has been initially released in 2016 [9] by the *Facebook* development team and is the Python-descendant of the *Torch* Lua-framework [10].

The framework is highly Python-specific (i.e. "pythonic" language features are used) while the core logic of PyTorch itself is written in C++ for better performance and less overhead [10].

The main differences to TensorFlow: PyTorch features dynamic computational graph definitions which makes it easier to interact with external data – in TensorFlow, special

placeholders have to be used which will be replaced by the actual data at runtime; the graph itself has to be defined statically before. Because the computational graph is dynamic in PyTorch, it is also by default possible to use dynamic inputs for a model, whereas in TensorFlow an additional library called *TensorFlow Fold* has to be used. For the same reason, debugging is possible with regular Python tools, where a specialized debugging tool like *tfdbg* is needed in TensorFlow to view the internal states of the computational graph [10].

PyTorch is extensively used by the transport network company *Uber*. Its *AI Labs* team has built *Pyro* based on PyTorch to improve the transportation experience by predictions and route optimizations [11].

4) Caffe: Caffe (stands for Convolutional Architecture for Fast Feature Embedding) has been originated from an academic background and developed by Yangqing Jia while he was a student at the University of California, Berkeley starting from 2014. It is a C++ library utilizing *CUDA* with well supported bindings to *Python* and *MATLAB* available. Caffe attempts to follow software engineering best practices to offer a good code quality which is made possible by complete unit tests (code coverage) and by clean coding e.g. through *modularization*, *separation of concerns* and *abstraction* [12].

It targets multimedia scientists, researchers and the industry. In its first release in 2014, it had a focus on computer vision – one of Caffe's key applications still is object classification. Later on, it has been strongly extended by community contributions to also (better) support uses such as robotics and speech recognition [12] and the members of Berkeley also cooperated with major industry corporations like *Adobe* or *Facebook*.

A consequential architectural difference to the other mentioned frameworks is the separation of the model's representation and the implementation. Hence, model definitions are stored in separate configuration files that are written in the *Protocol Buffer Language* (and in contrary to the other frameworks not actual code). As a result, Caffe has a *static* computational graph structure (like TensorFlow) and is not dynamic like Keras [12].

A popular architecture that uses Caffe for the purpose of person re-identification has been introduced by Ahmed et al. in 2015 which uses a CNN to detect if two images show the same person more efficiently by comparing local relationships of images using a thoughtfully adapted nearest neighbor layer and another one that outlines differences between the images [13].

5) Performance comparison: When comparing the training time duration of TensorFlow, Keras and PyTorch using popular CNN models on a *NVIDIA GTX 1080* on the "Dog Breed Identification" dataset from *Kaggle.com*, the following can be said: PyTorch and TensorFlow are on a nearly equal speed level when training the models *ResNet-50*, *VGG-16* and *VGG-19*. Keras is substantially slower than both other frameworks; it performs nearly twice as slow compared to TensorFlow in training the *Inception-ResNet-V2* and is approximately 80 percent slower than TensorFlow and PyTorch for *ResNet-50*. To be fair, these differences can potentially be linked to the abstractions that Keras uses for simplicity [14].

B. Adoption and Popularity

The data scientist J. Hale compared 11 Open-Source frameworks TensorFlow, Keras, PyTorch, Caffe and others that are not part of this paper (namely: Theano, Apache MXNET, Microsoft CNTK, Deeplearning4j, Caffe2, Chainer, FastAI). The majority of the compared frameworks use Python as the primary programming language.

To calculate the "Power Score", he used multiple data sources and weighted them – the higher the weight, the more is the score affected [8]. The following table shows how the data sources have been categorized to offer an overview:

Table I
DATA SOURCES - DEEP LEARNING FRAMEWORK POPULARITY

Category	Data source(s)	Weight
Jobs and Business	LinkedIn, Indeed, Simply Hired, Monster, Angel List	30%
Scientific Papers	ArXiv	10%
Articles	Medium	10%
Books	Amazon.com Books	10%
Usage Survey	KDnuggets	20%
Development	GitHub Activity	10%
Web Search Volume	Google Trends	10%

It can be argued that deep learning is very important for the industry (e.g. CNNs that are used for image recognition tasks for self-driving cars) which explains the relatively high chosen weight of 30 percent for the "Jobs and Business" category. Here, the most relevant job market platforms have been queried using seach term `machine learning <framework_name>`.

The categories "Scientific papers", "Articles" and "Books" can be summarized to "Science and Research". Publications on the platforms *ArXiv* (one of the biggest e-print services for scientific papers), *Amazon Books* (in the category "Computers & Technology") and *Medium* (a platform where many data science articles are published among others) have been counted. This category is especially relevant for educational purposes and therewith also for this paper itself. With a total of 30 percent for "Science and Research", it is weighted appropriately with the same relevance as "Jobs and Business".

To measure usage, Hale examined the results of a survey on *KDnuggets*, an international website for researchers in the machine learning field. In the poll of 2018, the following question was asked to data scientists: "What Analytics, Big Data, Data Science, Machine Learning software you used in the past 12 months for a real project?" [8].

In the "Development" category, only *GitHub* has been taken into consideration with a weight of 10 percent on the Power Score. Because of its popularity, *GitHub* is a suitable platform to measure development activity, but the category could gain more meaningfulness by using more data sources – for instance the competitor *GitLab* and also *StackOverflow*

specifically to measure the questions asked about each framework which can be seen as another activity indicator. What also has to be viewed critically is the small importance of this category given by the weight of only 10 percent.

Next, in the "Search Activity" category which can be viewed as an indicator for popularity *Google Trends* has been used. The same argument as already used for the "Development" category applies; more variety would increase representativity, although *Google* is the most used web search engine. The results of the analysis are shown in the following diagram:

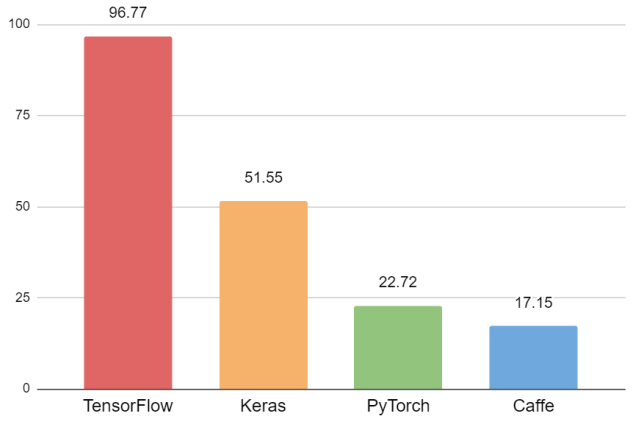

Figure 2. Power Scores of Deep Learning Frameworks [8].

Figure 2 shows the frameworks on the x-axis and the respective power score on the y-axis where a maximum value of 100 percent means that the framework has the first place in every category (as seen in table I).

The Google-backed TensorFlow has the highest score in every category of table I except for *Medium* articles. The most significant weighted relative difference to other frameworks is in the "Jobs and Business" category with an advance of the factor 3.2 to the next candidate Caffe.

From a science and research perspective, a vast majority of scientific papers and articles on *ArXiv* were about TensorFlow (3700), followed by PyTorch (1560) and Caffe and Keras having a nearly equal amount (about 1375). There were more than twice as much TensorFlow books (202) on *Amazon.com* compared to the next candidate Keras (79). However, when comparing Medium articles, Keras has nearly 3000 more articles than TensorFlow which could be because the former technology is generally easier to handle and to describe in tutorials, guides and articles because of the simplifying abstraction layer.

In *KDnuggets'* usage survey results, Keras has an interesting 22 percent of the votes compared to the leading 30 percent of TensorFlow, with only 6.4 percent PyTorch users and 1.5 percent of the respondents voted for Caffe.

For both categories "Development" and "Web Search Volume" the clear winner is again TensorFlow. From a future perspective, it is relevant that the Google Trends search shows that there is growth for PyTorch and Keras since September 2016 while the popularity growth for TensorFlow and Caffe has been stagnating since the middle of 2017 [8].

IV. Framwork Properties Summary

A. TensorFlow

As shown in section III-A1, TensorFlow has been developed from the beginning with a focus on computer vision applications. It features good performance (see section III-A5) and is flexible because of its toolset for deployment (*TensorFlow Serving*), debugging possibilities (*tfdbg*) and the various configuration options. On one hand, this enables professionals to fine-tune their ML models – on the other, it might be more difficult for beginners to start with TensorFlow because of the manual work needed. Eventually, it is also additional work to master tools like *tfdbg*, since debugging is not easily possible with regular tools or the ones provided by a standard IDE (Integrated Development Environment) as a result of the static graph. Though this can be relativized by the extensive documentation and high quality tutorials available online which has also been demonstrated in section III-B. TensorFlow is the most popular ML framework; data scientists around the world primarily choose TensorFlow as the *KDnuggets* survey suggests and TensorFlow is in the greatest demand on the job market. On top of that, there is much related (GitHub) development activity and it is the most mentioned framework in scientific papers.

All in all, TensorFlow seems to be an extremely popular and good choice for professional data scientists who might focus on optimizing computer vision applications (also because of the *TensorBoard* which eases the visualization and debugging of TensorFlow programs to optimize CNNs via statistical information etc.). Still, because of the many books, tutorials and comprehensive documentation available, it is also suitable for motivated beginners aiming to work on a lower level or people who want to learn TensorFlow to work in an enterprise that offers ML-related jobs.

B. Keras

The main argument for Keras (see section III-A2) is its abstraction layer which enables simpler, user-friendly and faster software development – it effectively lowers the learning curve and also enables fast prototyping of DL models. Because of its functionality to act as a frontend, developers can seamlessly test out a choice of backend frameworks besides TensorFlow. The high presence on *Kaggle.com* makes it easy to quickly get a big picture [1] and also participate in contests to learn in a practice-oriented way so that computer vision applications are made easier for beginners. This is also stated in section III-B, indicating that Keras seems to be very helpful for machine learning beginners because of its tutorial and *Medium* articles availability.

Concluding, Keras is a trending framework and also very popular among data scientists as the 22 percent result in the *KDnuggets* survey shows (probably also related to its fast prototyping potential). It is particular beginner-friendly, but might be slower than TensorFlow or PyTorch as outlined

in section III-A5 because of the less fine-grained operations potentially downgrading it for huge enterprise solutions that require an outstanding performance.

C. PyTorch

PyTorch (see section III-A2) is geared to performance and less overhead while still making DL models debugging-friendly (i.e. easier to debug than in the other compared frameworks) by using the dynamic computational graph approach; there are no special tools needed to debug. Section III-B points out that many researchers use PyTorch or write about it, although it is only used by 6.4 percent of the people on *KDnuggets* and the few books available. Nevertheless, *Google Trends* indicates that it is a trending technology which is used the second most (after TensorFlow) in many scientific works. It offers a performance on an equal level as TensorFlow (see section III-A5) when training data.

Finally, it might feel very natural for (experienced) Python developers because of the good language integration. The dynamic properties make it a good choice for beginners, since a debugging session can be started out of any standard Python IDE eliminating the need for a template mechanism to fill models with actual data.

D. Caffe

The case for Caffe (see section III-A4) is that it specifically targets computer vision applications (e.g. demonstrated by the Caffe-based architecture for person re-identification). As it aims to offer high code quality, it makes it easier to be openly extended by contributors. Model definitions are in configuration files to support the *separation of concerns* – this may seem to help persons to concentrate on the important parts of a model but could also confuse others because it is realized differently than in other frameworks.

In conclusion, because Caffe is in relatively high demand on the job market (after TensorFlow, as seen in section III-B), it has a good relation to industrial applications which is also made clear by the various cooperations with major corporations that could also ensure a long support lifecycle. Thus, Caffe is a suitable choice for multimedia scientists, although it should be noted that there has been a usage fall; both *KDnuggets* and *Google Trends* indicate less activity (this could be linked to Caffe's successor *Caffe2* which has recently been merged into PyTorch) [8].

V. Conclusions and Outlook

Among the different DL frameworks that are suitable for CV applications, there is no clear outstanding winner. The previous section summarized the strengths of the different frameworks. In the late 2019, TensorFlow is still the dominating framework, but the other candidates have specific advantages in terms of beginner friendliness, collaboration features, and speed. Besides TensorFlow, PyTorch is currently growing fastest [15].

References

[1] Francois Chollet. *Deep Learning with Python*. 1st. Greenwich, CT, USA: Manning Publications Co., 2017. ISBN: 9781617294433.

[2] James Kotecki. "Deep Learning's Permanent Peak On Gartner's Hype Cycle". In: *Medium - Machine Learning in Practice* (2018). URL: https://medium.com/p/-96157a1736e.

[3] Martin Abadi et al. "Tensorflow: a system for large-scale machine learning." In: *OSDI*. Vol. 16. 2016, pp. 265–283.

[4] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *CoRR* abs/1409.0575 (2014). arXiv: 1409.0575.

[5] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: (2015). URL: https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf.

[6] Aayush Bansal et al. "PixelNet: Representation of the pixels, by the pixels, and for the pixels". In: *CoRR* abs/1702.06506 (2017). arXiv: 1702.06506.

[7] Otavio Good. "How Google Translate squeezes deep learning onto a phone". In: *Google AI Blog* (July 29, 2015).

[8] Jeff Hale. "Deep Learning Framework Power Scores 2018". In: *Medium - Towards Data Science* (Sept. 20, 2018).

[9] Sam Redmond and Christopher Sauer. "Exploring Hardware Parallelism's Influence on Programming Through Convolution in CUDA and PyTorch". In: (2017).

[10] Kirill Dubovikov. "PyTorch vs TensorFlow - spotting the difference". In: *Medium - Towards Data Science* (June 20, 2017). URL: https://towardsdatascience.com/-25c75777377b.

[11] Noah Goodman. "Uber AI Labs Open Sources Pyro, a Deep Probabilistic Programming Language". In: (Nov. 3, 2017). URL: https://eng.uber.com/pyro/.

[12] Yangqing Jia et al. "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 675–678.

[13] Ejaz Ahmed, Michael Jones, and Tim K Marks. "An improved deep learning architecture for person re-identification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3908–3916.

[14] Wojciech Rosinski. "Deep Learning Frameworks Speed Comparison". In: *Deeply Thought* (Nov. 22, 2017). URL: https://wrosinski.github.io/deep-learning-frameworks/.

[15] Jeff Hale. "Which Deep Learning Framework is Growing Fastest?" In: *Medium - Towards Data Science* (Apr. 1, 2019).