A Review of Current Deep Learning Frameworks

Andrés Fernández Rodríguez

Software Engineering and Computer Vision

February 17, 2018

Table of contents

Section 1

Approach

Goal and Structure

The goal of this review is to **provide information about current DL frameworks** together with **criteria to allow their comparison**. For that,

1. It will focus on several factors that I considered relevant for any framework, and for DL frameworks in particular. This is discussed in the present section, **approach**
2. In the next section, several DL **frameworks** will be reviewed paying attention to the discussed factors
3. Finally, all the given information will be collapsed to **sum-up** and allow comparisons among frameworks
4. The information presented here comes **mostly from third-party sources**. Citations are marked in square brackets and attached in the **references** section

Scope

The scope of this review are **current software frameworks that integrate many DL-related tasks**. In this context,

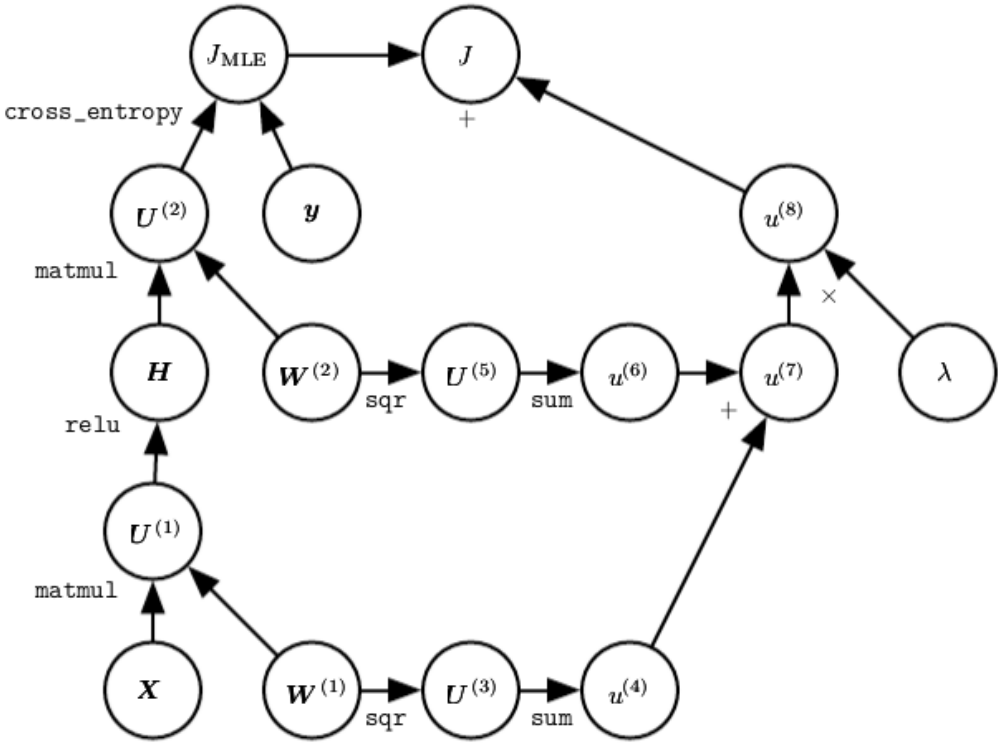
- A **framework** is a consistent piece of software that provides solutions for domain-specific tasks, usually exposed through an extensible API
- A **DL**-related task is concerned with the design, implementation, training... of programs that perform deep learning, as well as the management of the data and hardware required for that task
- DL is a rapidly changing landscape, so ***current*** doesn't necessarily mean *new*: some frameworks have been there for a while and are still alive and kicking

Relevant Factors of a Framework: A Trade-Off

- **Resources required**
  - Money (fees, new hardware/software requirements, etc.)
  - Time
    - Learning curve (structured API, long-lasting community)
    - Runtime (at training and production stages)
    - Development (verbosity, good documentation/help, IDE support)
    - Non-Development (testing, debugging, visualizing...)
    - Maintenance (stable API, large community)
- **Value provided**
  - Fiability (numerical stability, big data+long duration)
  - **API coverage and extendability (next slides)**
  - Licensing
  - Integration (saves time & money)
    - With the language(s) (via libraries, CLI...)
    - With communities (GitHub, model zoo, API stability...)
    - With other frameworks (f.e. serialization via ONNX...)
    - Further integration (O.S., PaaS, {C|G|T|I}PU...)

DL Frameworks: General Approach

Deep Learning is an approach to AI that can safely be regarded as the study of models that involve a great amount of **composition of learned funcs.**[10, p.8]



In DL Frameworks:

- Computations organized in graphs
- Nodes⇔operations
- Inputs⇔leaves
- Every node can be an output
- Dataflow as tensors across nodes
- Different approaches for backprop.

From [10, p. 220]: The computational graph corresponding to the following function composition:

$$J = \mathcal{L}(y, relu(x^T W^{(1)})^T W^{(2)}) + \lambda \sum_{w_i \in (W^{(1)}, W^{(2)})} w_i^2$$

DL Frameworks: API Coverage

The main actions to be done with this graphs are:

- Design and implement
- Train with backpropagation (+manage corresponding data and gradients)
- Deploy and run
- Test and debug
- Save, restore and serialize
- Visualize and/or communicate

For that, the APIs ideally provide:

- A comprehensive **Layer-API** with **differentiation** and batching
- A flexible **Model-API** supporting cyclic, dynamic graphs and param. sharing
- Facilities for **data** loading and management
- Facilities for testing, debugging and visualizing the graphs
- Support for (data/model) **parallelism** across different processing units
- Protocols for model **serialization** (architecture and parameters)
- Further desired features (DSP, PPLs...)

This can be achieved in different ways, as discussed in the following slides.

Graph: Declarative vs. Imperative Design (by [21, p.191])

Declarative

```
A = Variable('A')
B = Variable('B')
C = B*A
# compiles the function
f = compile(C)
c = f(A=np.ones(10), B=np.ones(10)*2)
```

- Static, *whole-graph* model
- Easier to optimize (e.g. distributed, batching, parallelization, black magic...) for devs
- More efficient (deploy, train, run)

Imperative

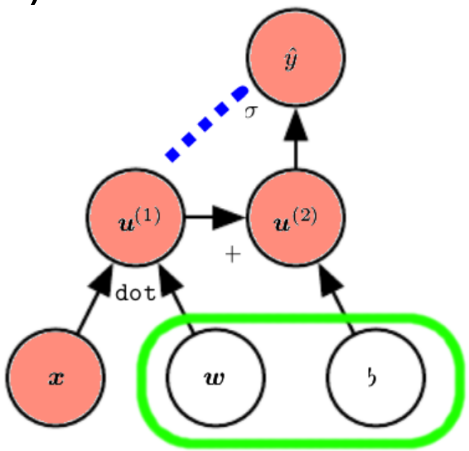
```
import numpy as np
a = np.ones(10)
b = np.ones(10)*2
c = b*a
d = c+1
```

- Dynamic, *layer-by-layer* model
- More intuitive, less verbose
- More flexible (data and graph are on same level, see * op)
- Complex behind the scenes, less efficient

Graph: Dynamic Elements

Using a computational graph can present two types of dynamism:

- **Shape of data input/throughput** changes:
  With exceptions (like flattening or pyramidal pooling) static graphs adapt poorly to this type of dynamism
- **Connections** and **weights** change:
  - Variable connections aren't usually problematic if the elements are pre-compiled, and the framework allows for conditional flow
  - Dynamic weight size is cumbersome even if the maximum can be allocated statically. Further solutions involve disabling the static graph checking (suboptimal)

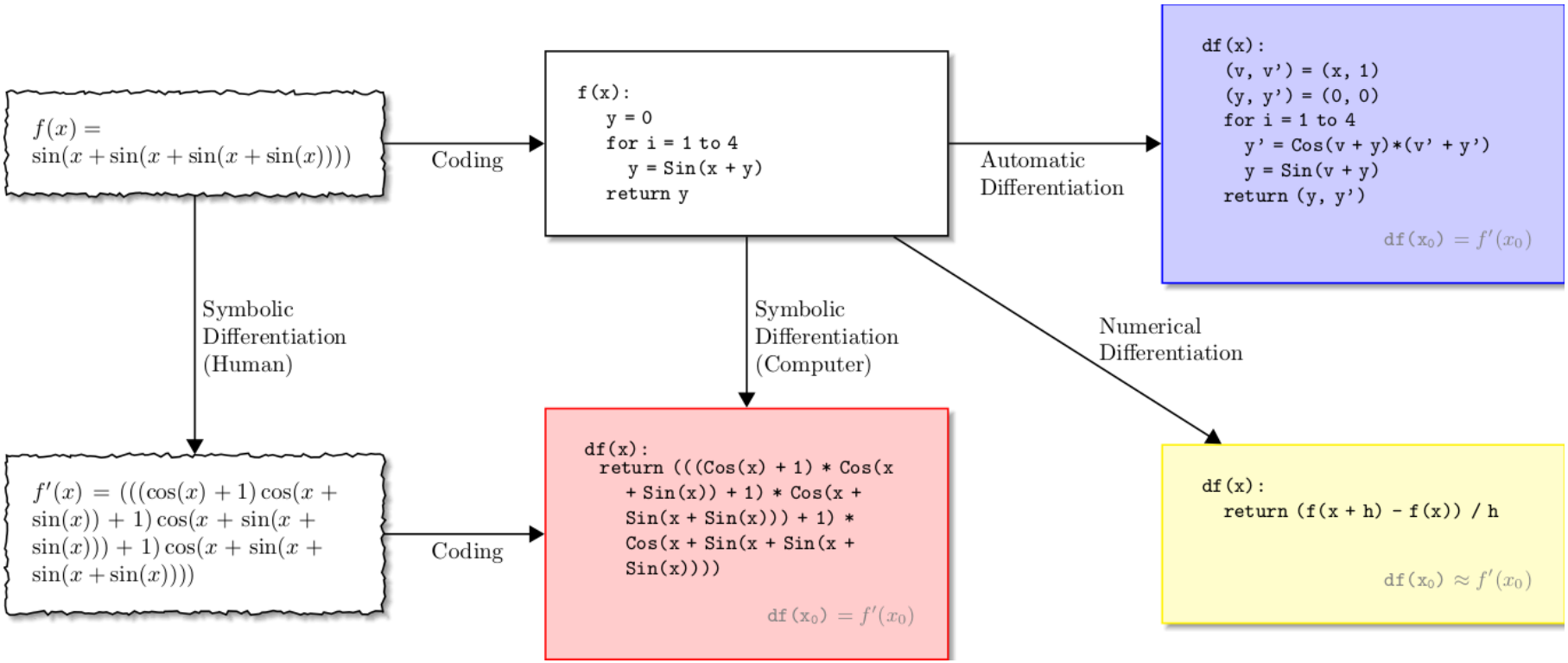


For $\hat{y} = \sigma(x^T w + b)$, basic example for dynamism in data (red), weights (green) and connections (blue). Adapted from [10, p.205].

Graph: Gradient Calculation

There are 3 popular methods to calculate the derivative, with diffent memory requirements, stability and performance features([2], [20], [15], [8]):

- **Numerical differentiation**: Imprecise, slow for high-dimensional data (used if f unknown)
- **Symbolic differentiation**: Expression for every architecture not always easy/stable/optimal
- **Automatic differentiation (tape-based)**: Numerical+symbolic+chain rule (compromise)



From [3]: Differentiation of mathematical expressions and code. Symbolic differentiation (lower center); numerical differentiation (lower right); automatic differentiation (upper right)

Comparing Frameworks: Problematic

It is difficult to convey all the relevant factors of a DL framework in a compact, precise and unbiased way.

For instance, a Framework with a quick data loader and dynamic graph model may perform better or worse in runtime/training depending on the particularities of the graph, data and hardware involved.

There are even cases in which the development time is more important than the runtime and a smoother learning curve or specific languages are prioritized.

This review tries to achieve a compromise by combining 3 formats: **tabular**, **pros-cons** bullet list and **kiviat** diagram (discussed in the following slides).

Comparing Frameworks: Tabular Format

Collapsed: info lost in specific/subjective goals

	Languages	Tutorials and training materials	CNN modeling capability	RNN modeling capability	Architecture: easy-to-use and modular front end	Speed	Multiple GPU support	Keras compatible
Theano	Python, C++	++	++	++	+	++	+	+
Tensor-Flow	Python	+++	+++	++	+++	++	++	+
Torch	Lua, Python (new)	+	+++	++	++	+++	++	
Caffe	C++	+	++		+	+	+	
MXNet	R, Python, Julia, Scala	++	++	+	++	++	+++	
Neon	Python	+	++	+	+	++	+	
CNTK	C++	+	+	+++	+	++	+	

From [19] (outdated): Has to be given along with the criteria to be interpretable

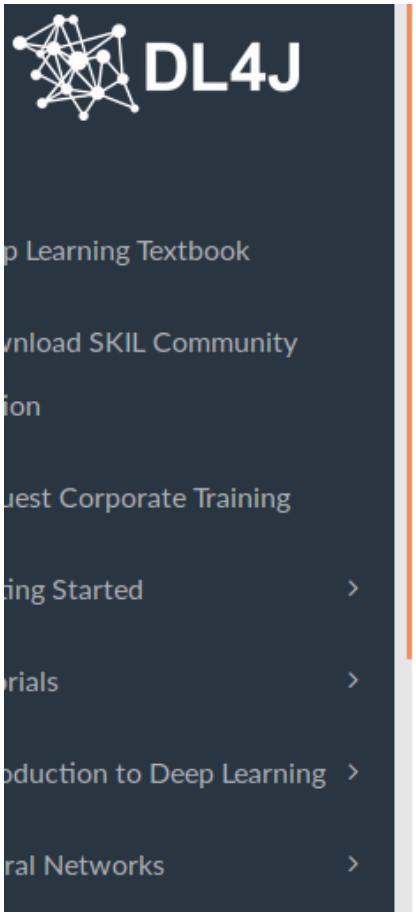
Extended: too sparse

		Chainer	PyTorch	TensorFlow	Theano-based	Caffe1 Caffe2	Torch7	MXNet	DyNet	PaddlePaddle	DL4J	CNTK	neon	Knet.jl	Darknet	Thinc
Basics	Language	Python	Python	Python	Python	Python/C++/ MATLAB	LuaJIT	Python/others	Python/C++	Python/C++	Java	BrainScript/ Python/C++	Python	Julia	C	Python
	Approach	define-by-run	define-by-run	symbolic autograd	symbolic autograd	static	static/ manual grads	symbolic autograd/ manual grads/ define-by-run [1]	define-by-run	symbolic autograd	static/ manual grads/ symbolic autograd [2]	static/ symbolic autograd	static/ symbolic autograd [3]	define-by-run	static	callback-based define-by-run
	CPU backend package	NumPy	TH	Eigen	NumPy		TH	mshadow	Eigen		ND4J		NumPy	Julia		NumPy
	GPU backend package	CuPy	THC	Eigen	libgpuarray		THC	mshadow	Eigen		ND4J		neon	KnetArrays		CuPy
	Primary sponsor	Preferred Networks	Facebook	Google	MILA	Facebook	Facebook	Amazon/Apache	CMU	Baidu	Skymind	Microsoft	Intel Nervana	Koc University	Joe Redmon	Explosion AI
NNs	CNNs	full	full	full	full	full	full	full	partial	full	full	full	full	partial	full	none
	RNNs	full	full	full	full	partial	full	full	full	full	full	full	partial	partial	partial	partial
	Reverse-mode autograd	Y	Y	Y	Y		torch-autograd	Y	Y	Y		Y	ngraph	Y		with closures
	Forward-mode autograd			tensorflow-forward-ad	Y											
	Higher-order grads		Y	Y	Y									Y		
	Variable-length loops	native	native	while_loop	scan	RNNs only	native	2017	native	RNNs only	none	dynamic axis	none	native	none	native
	Different architectures per batch	native	native	fold			torch-autograd	MinPy	native					native		native
Performance	cuDNN support	full	full	partial	partial	full	full	full	partial	full	partial	full	N/A [4]		partial	
	CPU/GPU generic backend	Y	Y				Y	Y	Y	Y	Y	Y	Y	Y		Y
	Multi-GPU data parallelism	Y	Y	Y	Y	Y	Y	Y		Y	Y	Y	Y	Y	Y	
	Multi-GPU model parallelism	Y	Y	Y	Y	Y	Y	Y		Y		Y	Y			
	Multiprocessing [5]	full	partial						full							
	Distributed training	ChainerMN	THD	Y		2017	torch-distlearn	Y		Y	Spark	Y	Y			
Misc	Runtime debugging	debug mode, typechecking, pdb	pdb	tfdbg				Monitor	pdb		Java debuggers	cntk.debugging		Gallium.jl	gdb	pdb
	Trainer abstraction	native	tnt		Blocks, Lasagne, Keras	native	torchnet			native	native	native	native			native
	Reporter abstraction	native	tnt	native			torchnet	native			native	native				
	Web interface			TensorBoard							DL4J-UI		Nervana Cloud			
	Graph compilation engine		2017	XLA		2017		NNVM					ngraph			

From [28]: This table demands a significant amount of time to be fully explored

Comparing Frameworks: Pros-Cons Bullet List

- \+ Qualitative and extended evaluation intuitively
- – Verbose, requires backing-up text
- – Less visual than extended tables and still sparse (inefficient for large number of factors)
- ? Non normalized, allows for more freedom but structure becomes more arbitrary



Pros and Cons

- (+) Python + Numpy
- (+) Computational graph abstraction, like Theano
- (+) Faster compile times than Theano
- (+) TensorBoard for visualization
- (+) Data and model parallelism
- (-) Slower than other frameworks
- (-) Much "fatter" than Torch; more magic
- (-) Not many pretrained models
- (-) Computational graph is pure Python, therefore slow
- (-) No commercial support
- (-) Drops out to Python to load each new training batch
- (-) Not very toolable
- (-) Dynamic typing is error-prone on large software projects

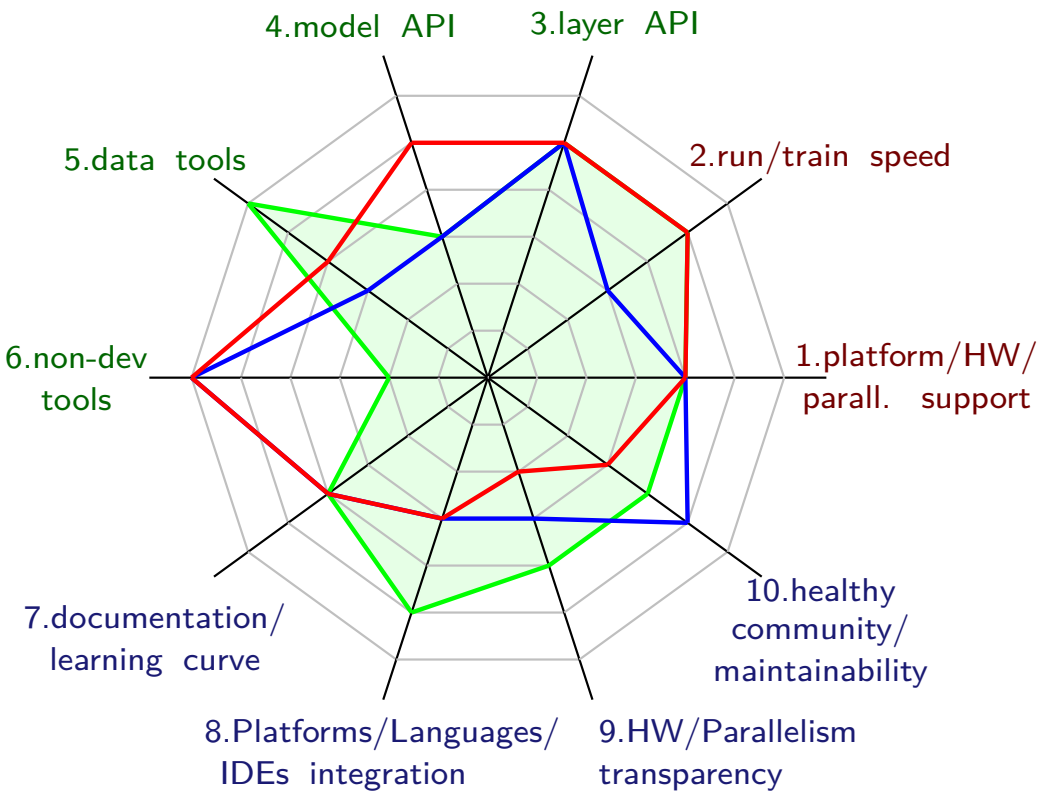
Caffe

Caffe is a well-known and widely used machine-vision library that ported Matlab's implem

From [29]: The DL4J webpage offers a review of several DL frameworks in this format

Comparing Frameworks: Kiviat Diagram - 1/2

+ Compromise between qualitative and extended evaluation
+ Very condensed: multiple elements ("rows") per diagram possible
– Bad handling of missing entries (zero-value? remove axis?)
– Saturates if no. of axes and elements increase (can become ambiguous)



- Scores: 0 (center)⇔unknown, 1-6⇔worst to best
- Relevant factors collapsed into 11 axes, grouped as well in 3 colors: performance, coverage, ecosystem
- Compared elements can also have a color code to help interpretation (see example on the left: green element would be ambiguous without the filling)

Comparing Frameworks: Kiviat Diagram - 2/2

The performance and coverage axes follow these criteria:

- **Score=5**: The framework provides a competitive solution that adapts to the user's needs, via built-in functionality or flawless integration (matplotlib, the Net class in PyTorch for NN design)
- **Score=4**: the framework provides a competitive but rigid/domain-specific solution (NN design in Caffe)
- **Score=3**: The framework provides convenient solution that is neither high-end nor easily customizable (TensorBoard)
- **Score=2**: the feature isn't provided by the framework, directly or via integration, and it has to (and *can*) be manually developed
- **Score=1**: Inexistent/unfeasible solution

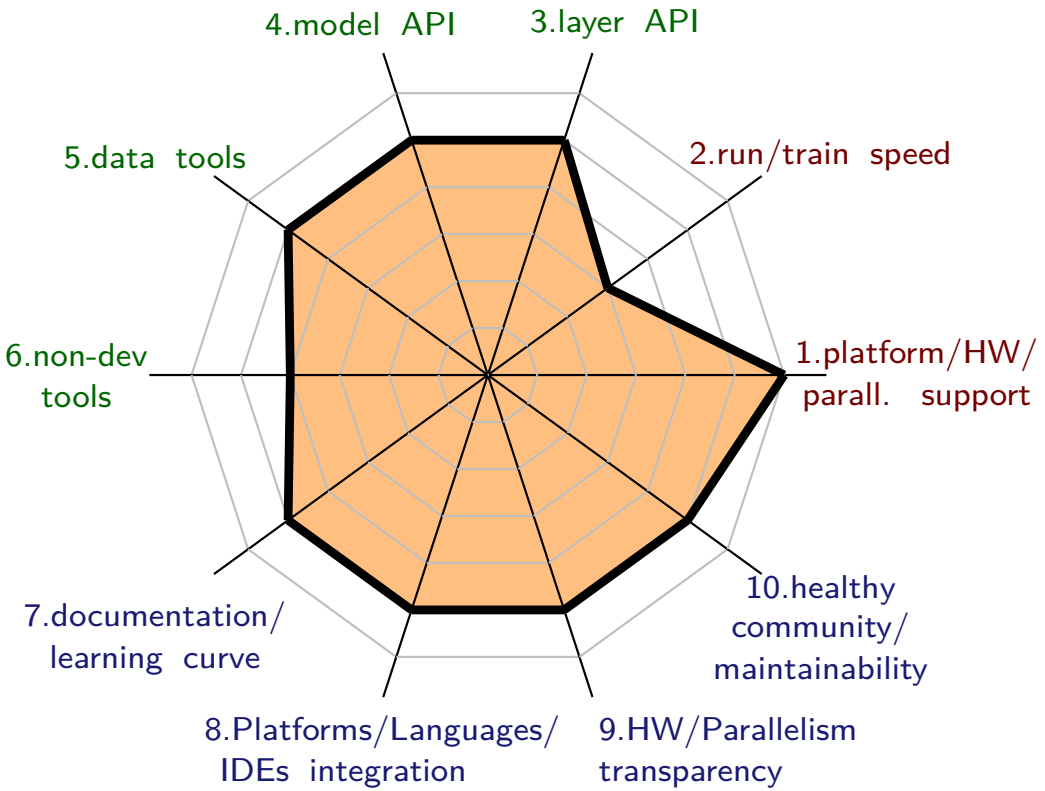
The ecosystem axes follow these criteria:

- **Score=5**: The framework is competitive with virtually no time overhead for the user
- **Score=4**: The user has to invest some time if no prior experience
- **Score=3**: The user has to invest time to get competitive
- **Score=2**: High time overhead, not competitive
- **Score=1**: Feature demands unfeasible amount of time

In all cases: **Score=0**: Not known, **Score=6** A speciality of the framework

Example Combining Table, List and Kiviat: TensorFlow

Host	Google, GitHub
Fees, License	No fees, Apache license since 2015[48])
Graph Build	Declarative
Backprop. Model	Automatic differentiation
Languages[47]	C++, Python, Java, Go, Ruby, R...[46]
Notes	Successor of Theano, general purpose framework



1. Runs on CPU, GPU, TPU, IPU, Android and embedded
2. **Dramatically** slower[22][25][1]
3. Very broad, 2^{nd}-order derivatives
4. Data/parameter dynamism can be achieved with Eager[5]
6. Provides all needed tools but require extra work (especially TensorBoard)
8. Great support for C++/Python. Poor APIs+docs for the rest
10. Broad and specialized community

Approach: Summary

So far, the following has been discussed:

- The goals, structure and scope of this review
- The desired features of a framework (general and DL-specific)
- Some basic aspects of DL frameworks
- Evaluation criteria and different representations to allow an effective comparison among frameworks

The next section will discuss several frameworks, following this schema.

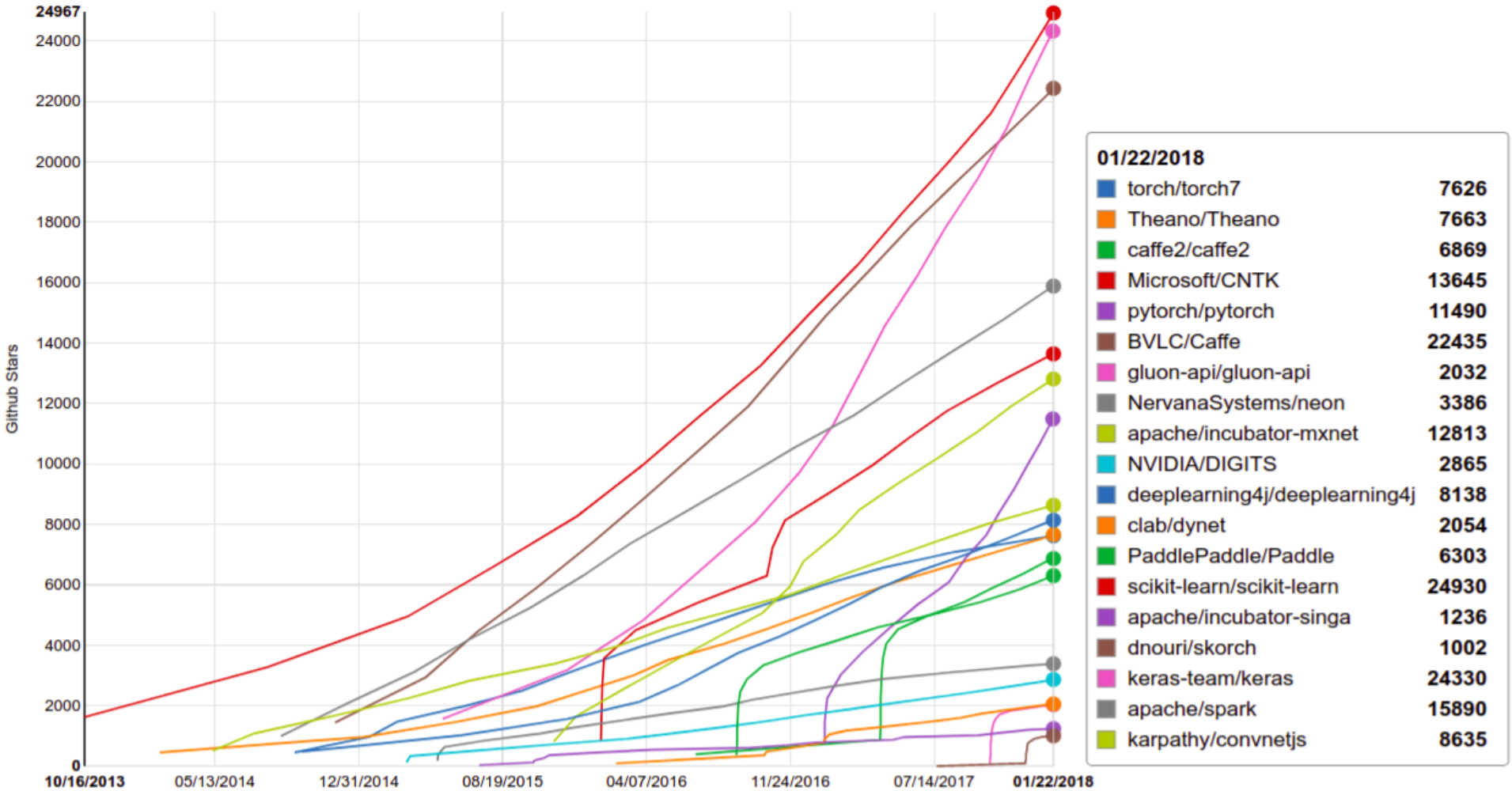
Section 2

Frameworks

Some of the DL Frameworks Currently Avaliable

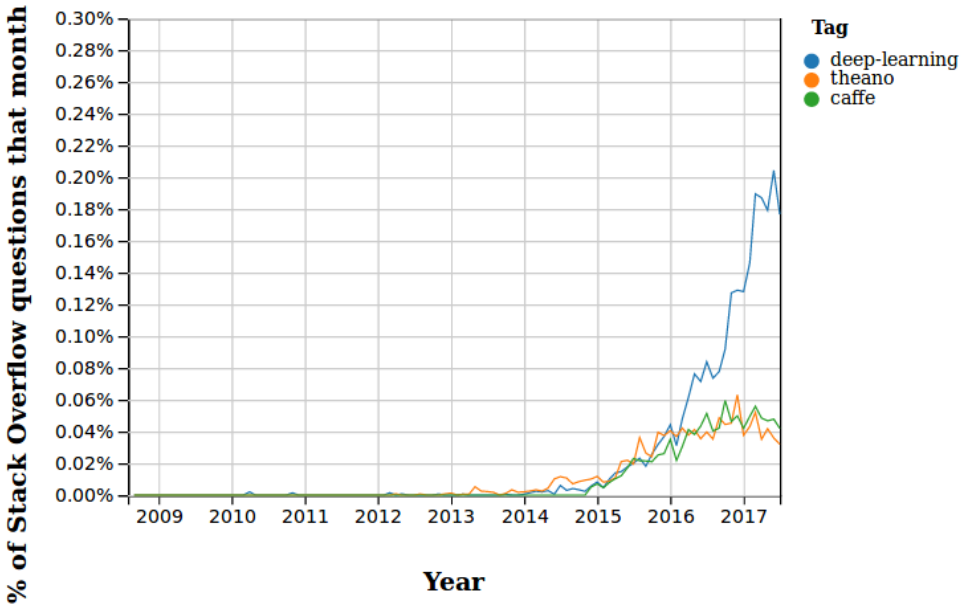
- **Open-Source(d)**:
  - PyTorch (Facebook) ← Torch (several)
  - Caffe2 (Facebook) ← Caffe (BVLC)
  - TensorFlow (Google) ← Theano (U. Montreal)
  - MXNet (Apache)
  - DL4J (Eclipse)
  - CNTK (Microsoft)
  - DyNet (Carnegie Mellon U.)
  - Chainer (U. Tokyo)
  - ConvNetJS (Karpathy)
  - POPLAR (GraphCore)
  - And more (not covered): DIGITS (NVIDIA), Paddle (Baidu), Scikit-learn, SINGA and MLlib (Apache), neon (Intel)
- **Closed-Source**:
  - Mathematica (Wolfram)
  - Michelangelo(Uber)
- **Higher-Order**:
  - Keras (on TF, Theano or CNTK)
  - Skorch (wraps PyTorch)
  - GLUON (on MXNet, CNTK)

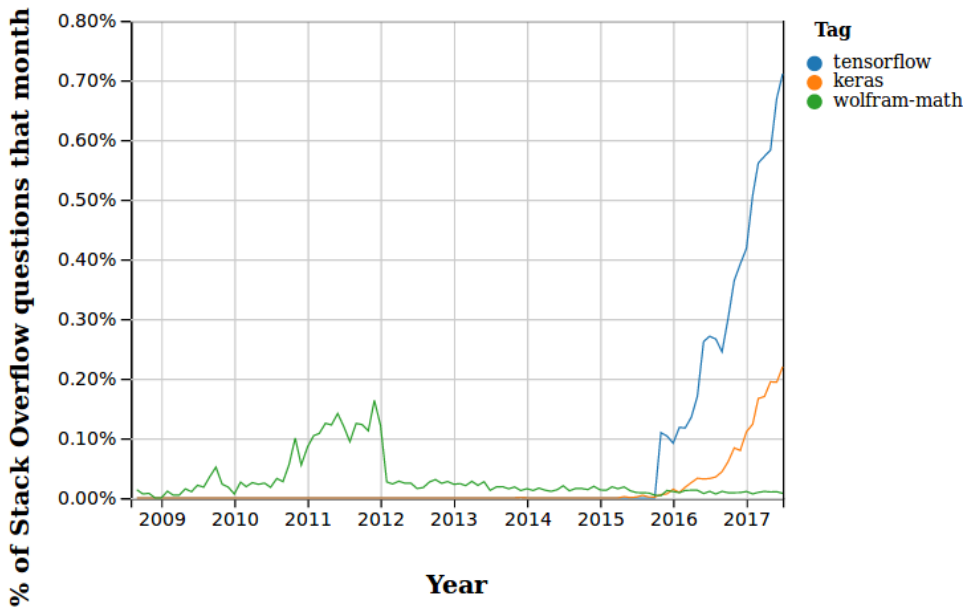
Popularity at GitHub



Repository stars on GitHub by date (generated with [24]). The TensorFlow repository, created at Nov.2015, has over 86k stars and was hidden for better visualization.

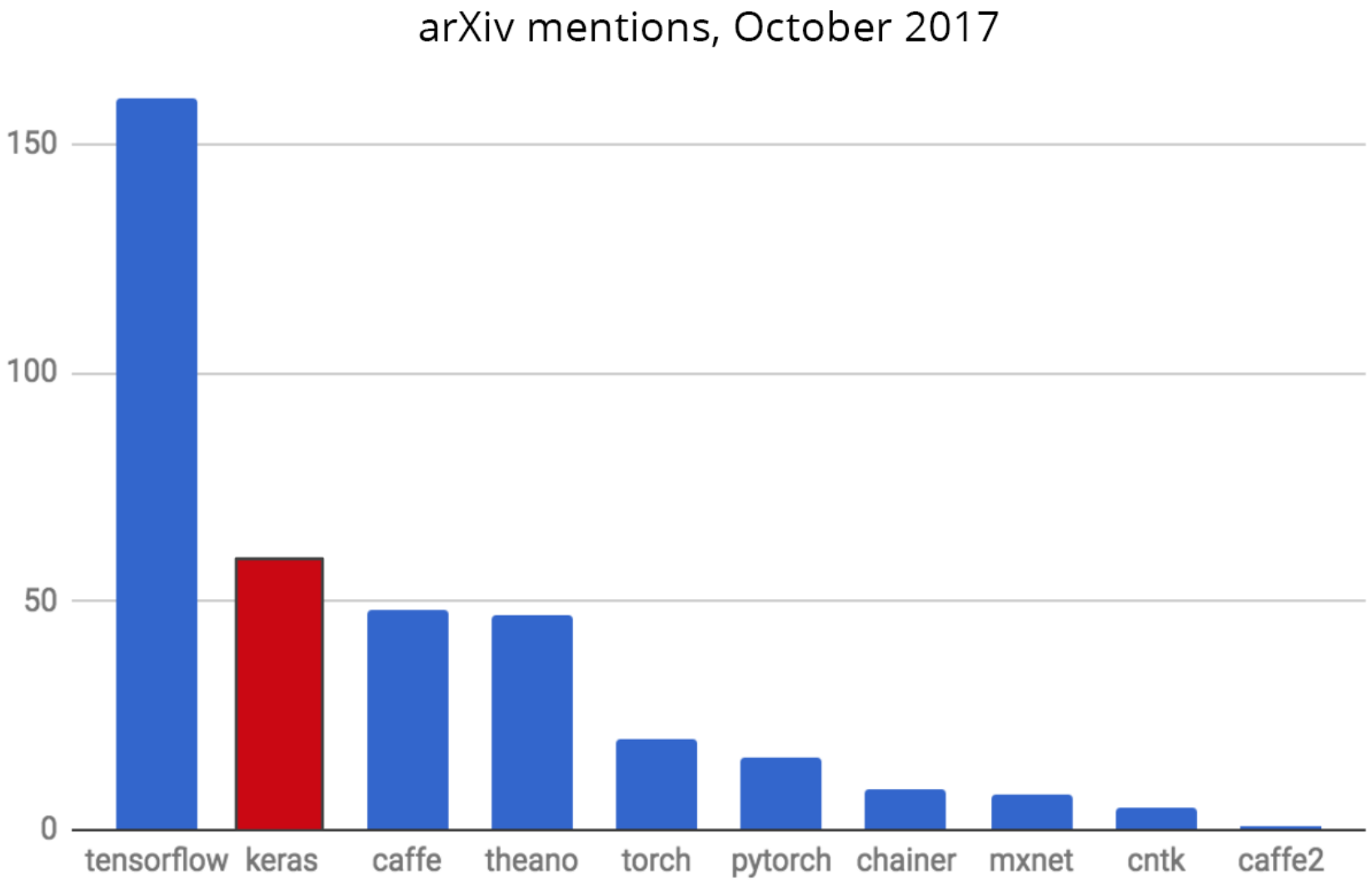
Popularity at StackOverflow





Popularity at StackOverflow for the existing tags[43]

Popularity at ArXiv

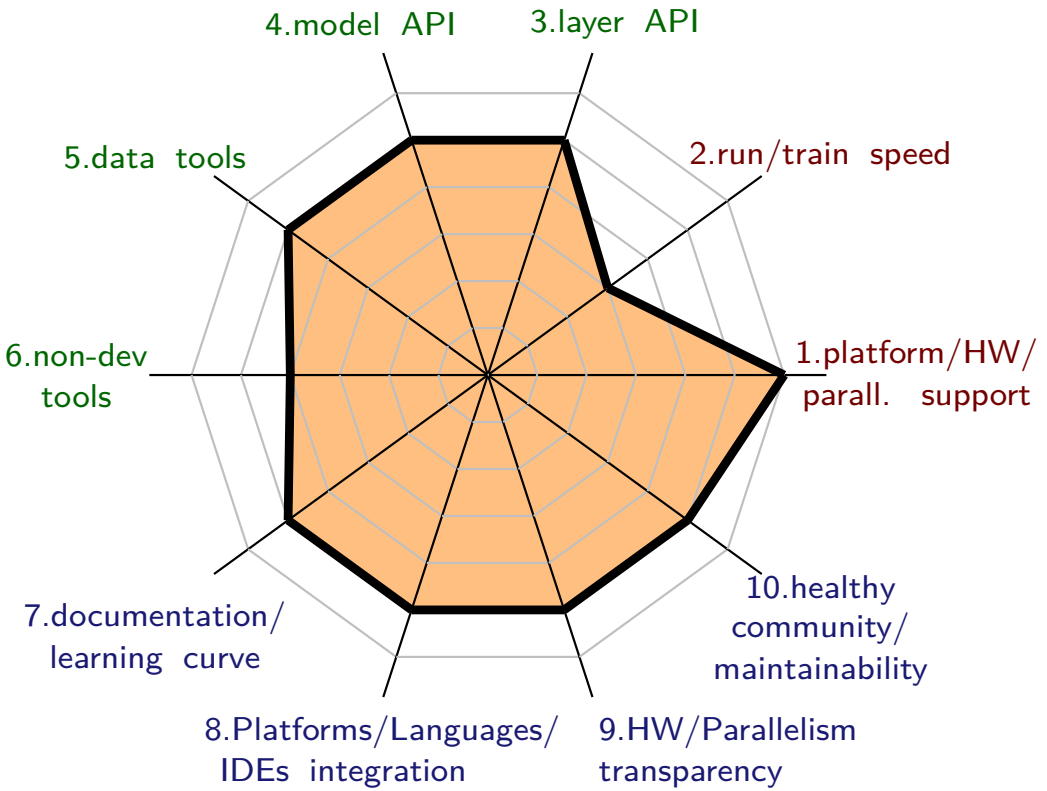


Mentions in scientific papers uploaded to the preprint ArXiv.org server (from [38])


TensorFlow

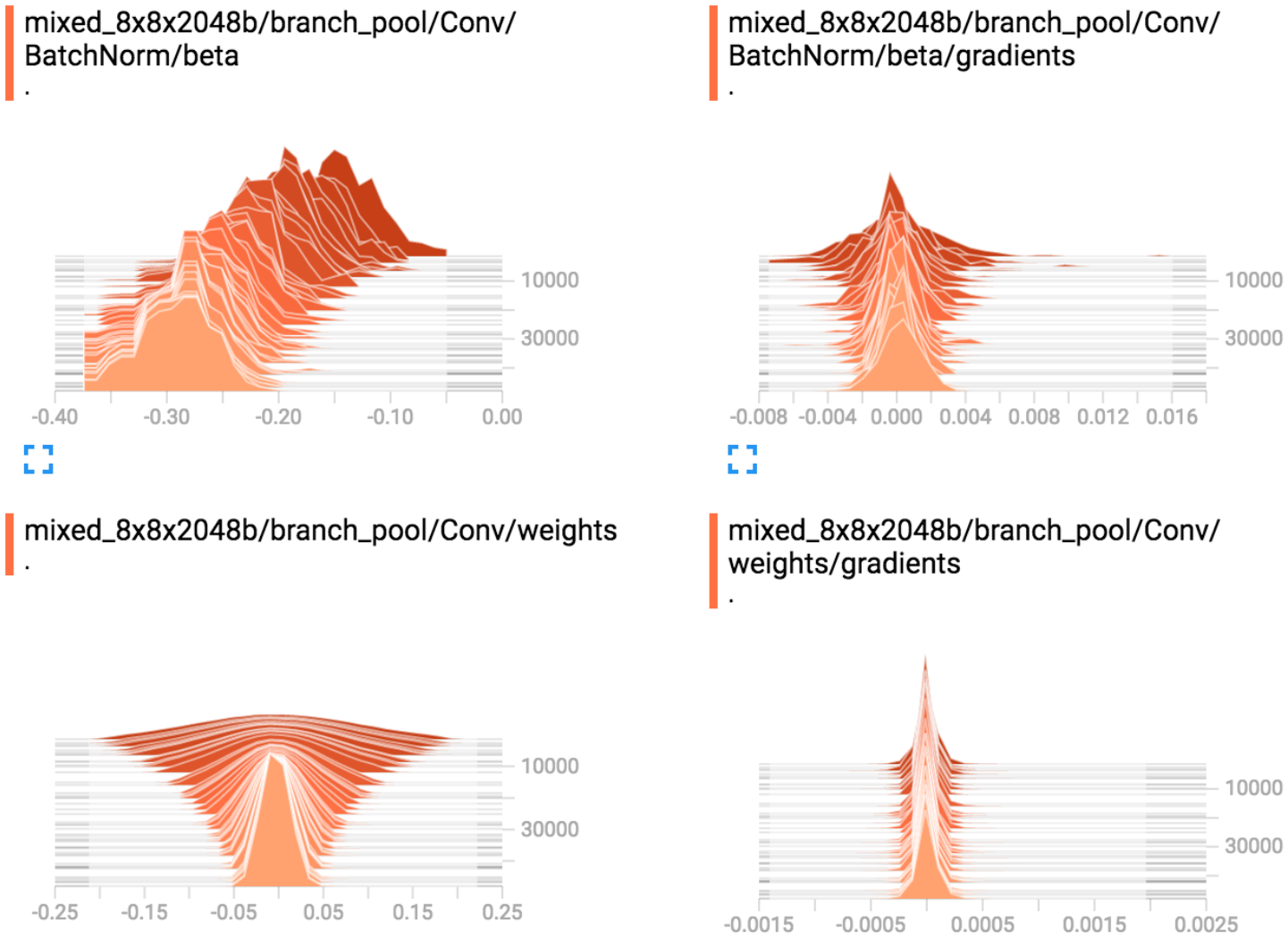
TensorFlow: Overview

Host	Google, GitHub
Fees, License	No fees, Apache license since 2015[48])
Graph Build	Declarative
Backprop. Model	Automatic differentiation
Languages[47]	C++, Python, Java, Go, Ruby, R...[46]
Notes	Successor of Theano, general purpose framework



1. Runs on CPU, GPU, TPU, IPU, Android and embedded
2. **Dramatically** slower[22][25][1]
3. Very broad, 2^{nd}-order derivatives
4. Data/parameter dynamism can be achieved with Eager[5]
6. Provides all needed tools but require extra work (especially TensorBoard)
8. Great support for C++/Python. Poor APIs+docs for the rest
10. Broad and specialized community

TensorBoard: Problematic

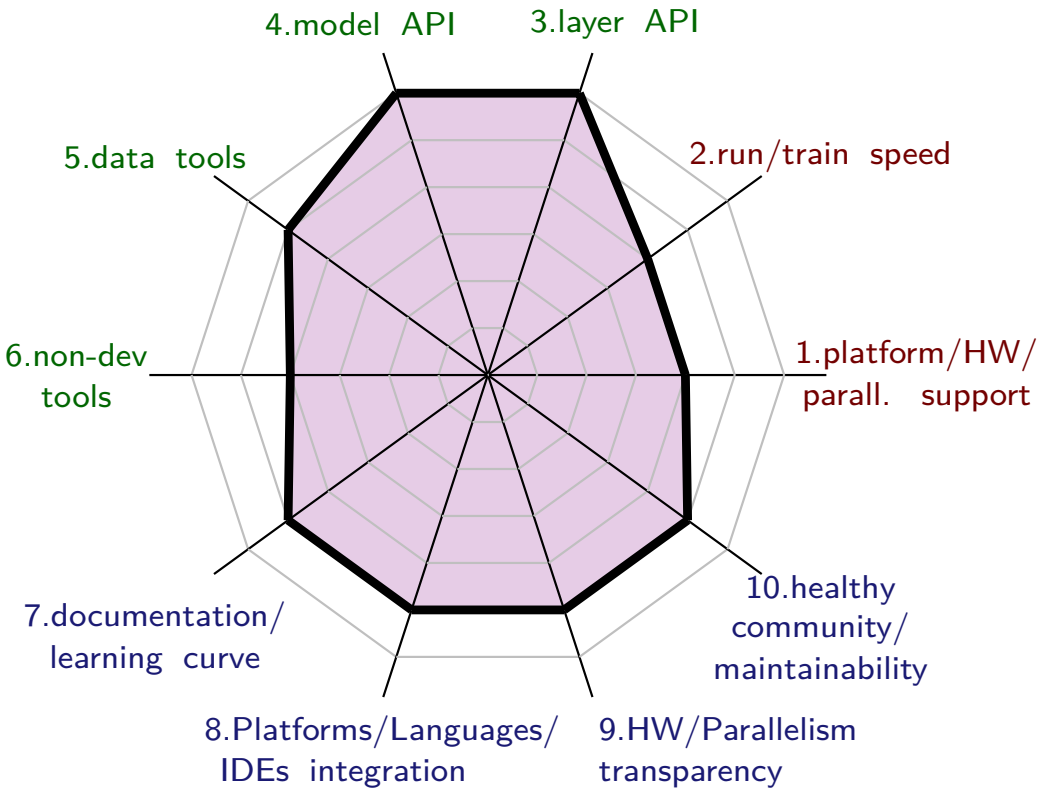


TensorBoard is a very competent and convenient tool, but has its drawbacks. For example, the histograms are hardcoded to 30 bins, and the process of extending it is involved (photo: [7]).


PYTORCH

PyTorch: Overview

Host	Facebook, GitHub
Fees, License	No fees, 3-Clause BSD since January, 2017[29]
Graph Build	Imperative (define-by-run)[28]
Backprop. Model	Automatic differentiation[45]
Languages[47]	Python
Notes	Spin-off of Torch

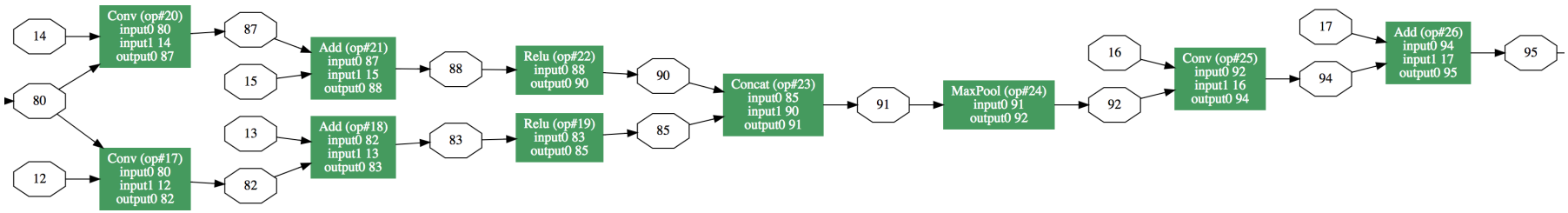


1. "essentially a GPU NumPy replacement"[11] + autograd
2. Extremely dynamic graph lifecycle may slow down some setups
4. PyTorch is especially capable for trying out new models
6. Relying highly in Python, requires some extra work for DL-specific (especially TensorBoard)
8. Arguably Python is all you need
9. Almost zero code overhead between CPU and GPU model
10. Fastest growing + ONNX

About Open Neural Network Exchange (ONNX)

Specification for serializing trained models (similar to Google's **protobuf**)

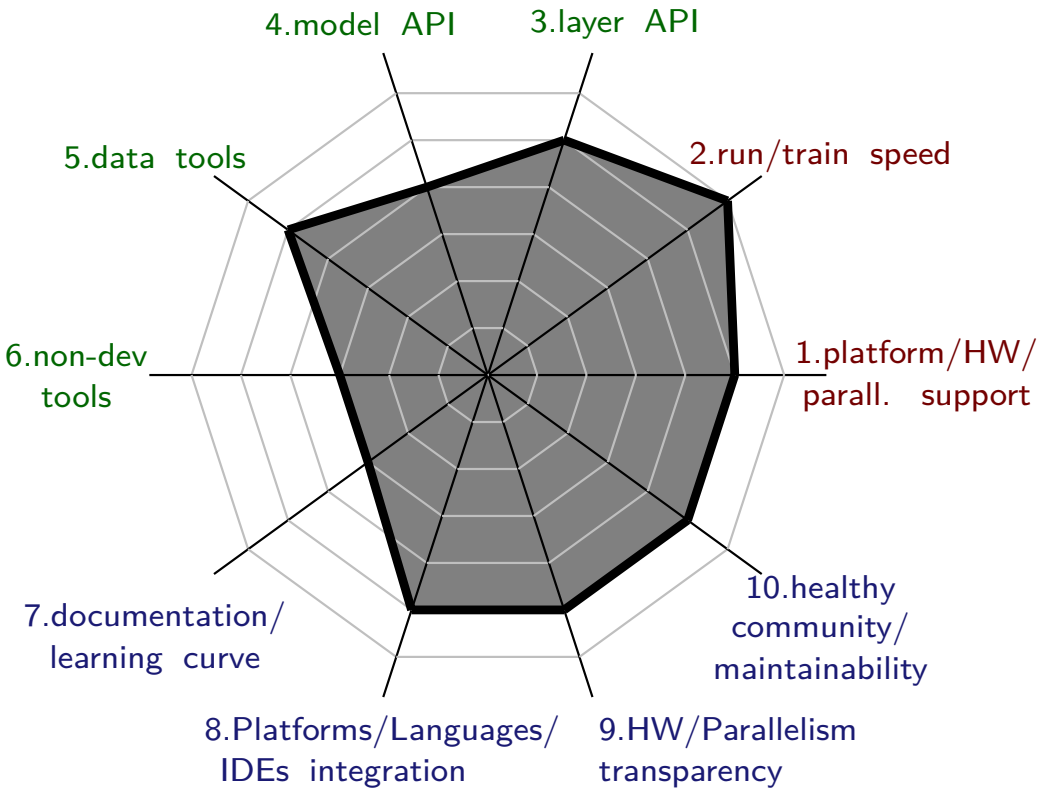
- Effort by Microsoft and Facebook (Amazon joined) to bridge the gap between model development and production. MIT License
- Released on September 2017, supported by CNTK, PyTorch, Caffe2, MXNet and Nvidia's TensorRT
- For static graph definitions, it only requires translation
- Branching only partially allowed (for shape operations etc): dynamic flow control not supported
- Graph visualization tools: NetDrawer (see picture from [42]) and Netron (interactive, supports also TF/Keras)




Caffe2

Caffe2: Overview

Host	Facebook, GitHub
Fees, License	No fees, Apache2.0 license since April 2017)
Graph Build	Static (plaintext schemas)[27]
Backprop. Model	Automatic differentiation[27]
Languages[47]	C++, Python
Notes	Successor of Caffe, focus on modularity and scalability[13]. Unlike Caffe, not only vision

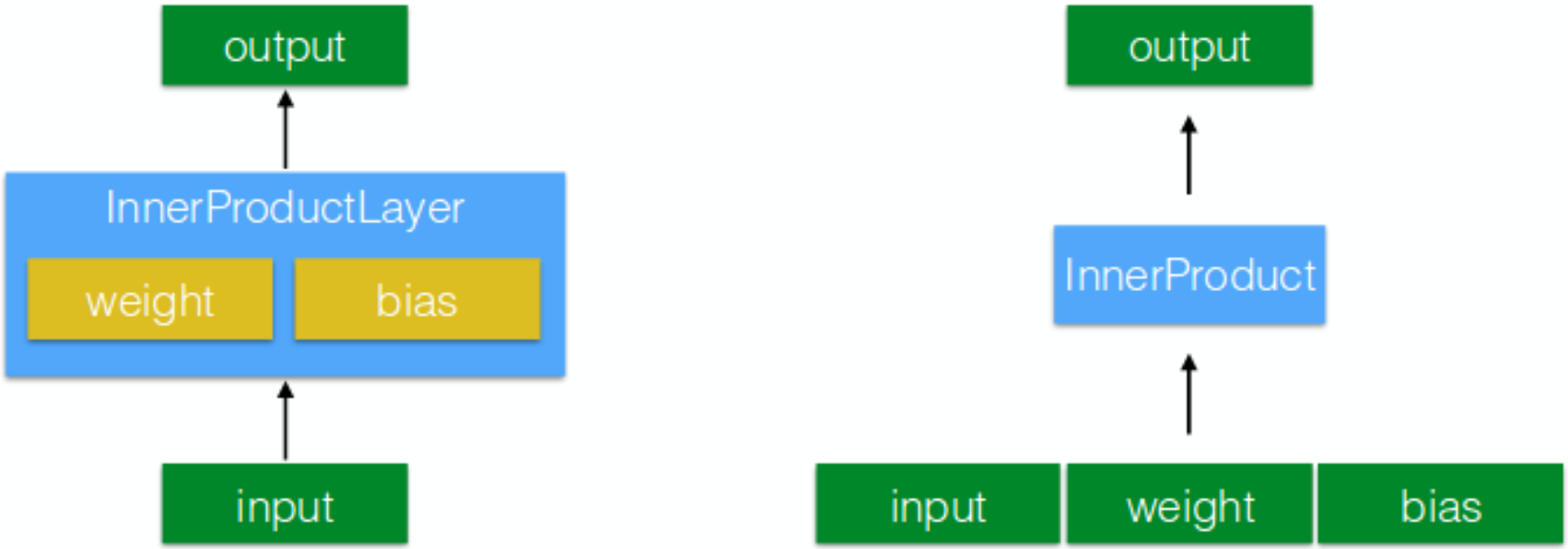


1. CPU, GPU, Android and iOS
3. Very rich and extendible DL API
4. Static model performs but prevents dynamism
6. Provides all needed tools but requires extra work (many dependencies)
7. well structured docs but lacking
8. Great support for C++/Python. Poor APIs+docs for the rest
9. Flawless transparency
10. As in Caffe, outstanding model zoo+incoming ONNX[6]

From Caffe to caffe2: Example

Purposes in the origins of caffe2[13]:

- Faster experimenting and prototyping (see picture)
- Simpler compilation (restructuring dependencies)
- More portability (CPU/CUDA, Android, further)

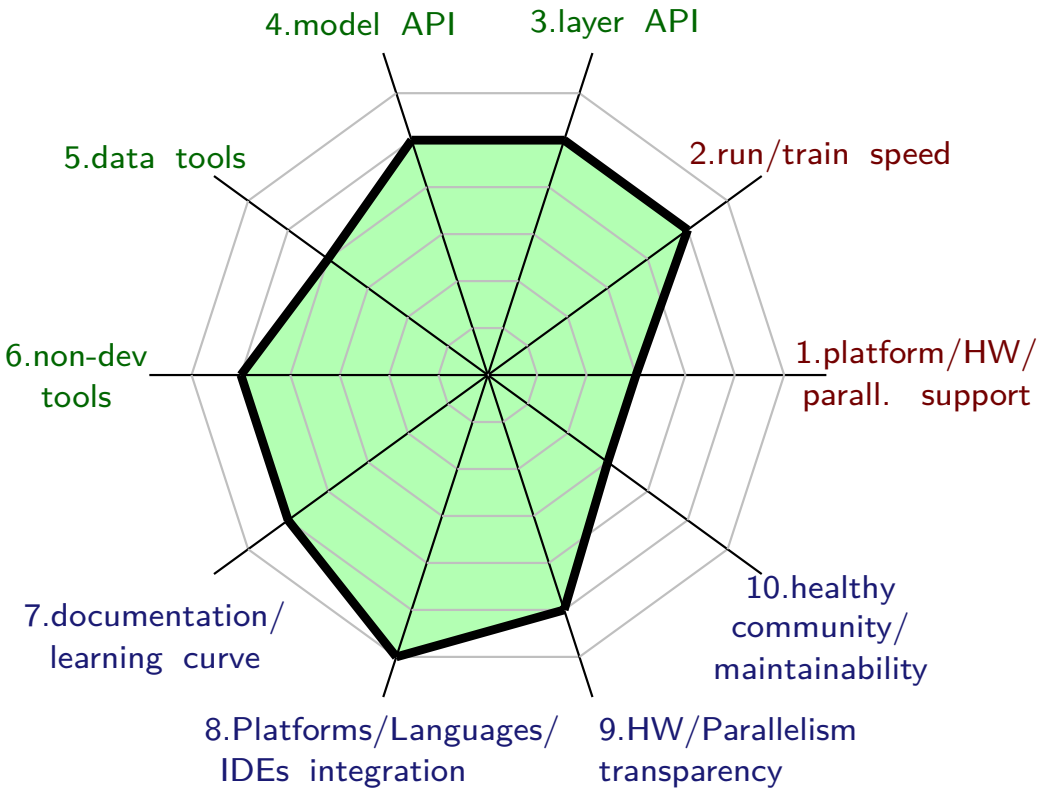


Difference in design: from Caffe's *Layers* (left) to Caffe2's *Operators* (right)[14]


Microsoft
CNTK

Cognitive Toolkit (CNTK): Overview

Host	Microsoft, GitHub
Fees, License	No fees, almost-OS license[29] since Jan 2016)
Graph Build	Declarative[49]
Backprop. Model	Automatic differentiation[49]
Languages[49]	C++, C#, Python, BrainScript, Java
Notes	Apparently what MS internally uses[51]

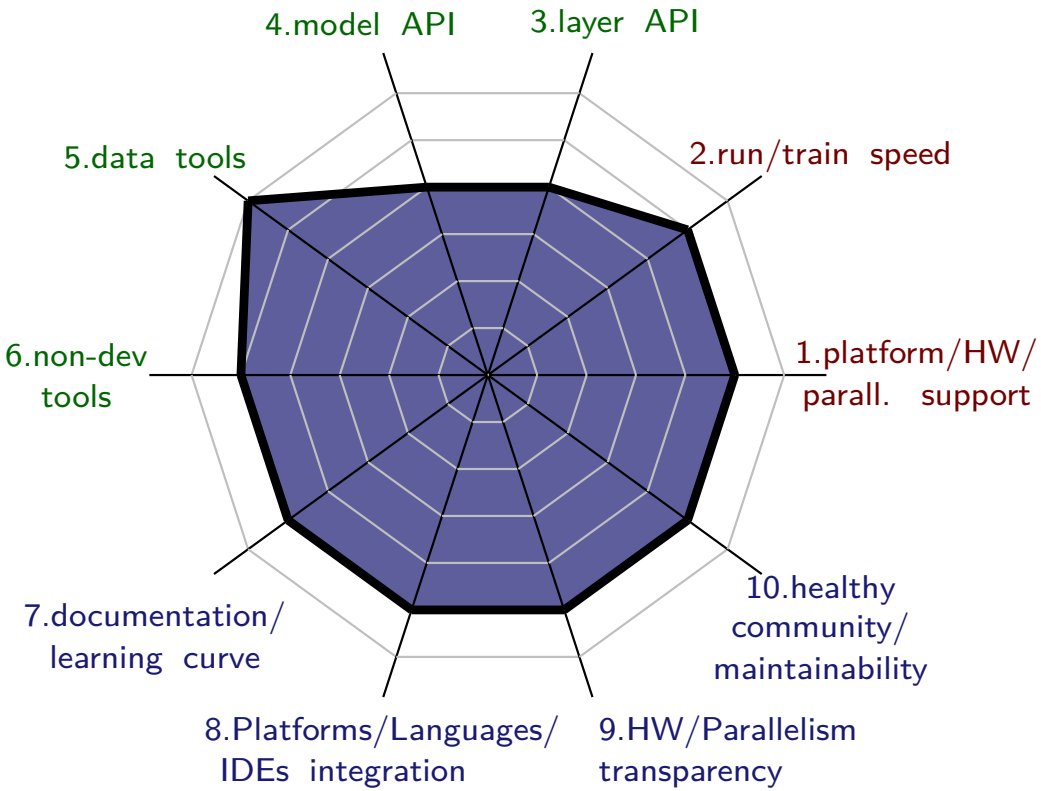


1. High-scale CPU/GPU, no traces of anything else
2. Much faster than TF ([1])
3. Claimed very broad[49]
4. Very comprehensive support (also through higher-order APIs)
6. Indirect support but very well documented
8. One of the greatest strengths
10. Small user base, restrictive licensing. But connections to Keras, Gluon, ONNX


DL4J

Deep Learning for Java (DL4J): Overview

Host	Eclipse, GitHub
Fees, License	No fees, Apache2.0 license[32], release 0.92 in Dec 2017
Graph Build	Declarative[33]
Backprop. Model	NOT AD (but unclear what, see [34])
Languages[32]	Java, Scala
Notes	CoC on the JVM, emphasis on scalability, UI and data management



1. JVM(CPU/GPU/Android) with Spark, Hadoop, Kafka
2. Speed=Caffe, $>$ TF/Torch (measuring data loading)[29]
3. No AD, cumbersome to extend
3. Lacking GANs, some RNNs...

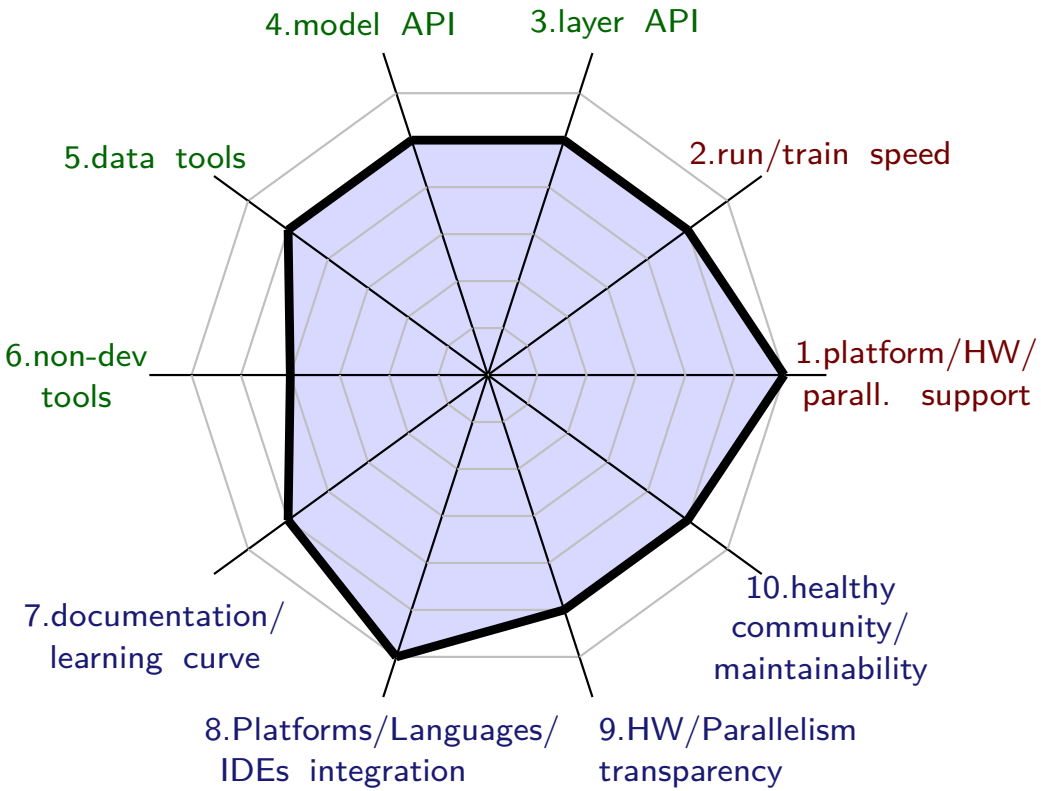
5.,6. Outstanding. The speciality of DL4J

8. Arguably Java is all you need
10. Less structured than other frameworks[34]. One of the few providing corporate support. Model Zoo + Keras frontend


mxnet

MXNet: Overview

Host	Apache, Amazon, GitHub
Fees, License	No fees, Release 1.0 in Jan 2017 under Apache2.0 license[39])
Graph Build	Both declarative and imperative allowed[40]
Backprop. Model	Automatic differentiation[26]
Languages	C++, Python, Scala, R, Julia, Perl[26]
Notes	In-house system at Amazon, supported by many companies and universities[41]



1. AWS, commodity, Android, IoT, IPU, containers...[41][37]
2. Competitive in small scale[1], scales well[41]

3., 4. Comprehensive, extendible. Dynamic with Gluon[29]

5. Includes sparsity
6. Logs into files, lacks further built-in support but has many SoTA APIs[18]

10. 6 OSS APIs + Keras + Gluon + ONNX + Model Zoo

Brief Reviews - 1/6

theano

- Google created TensorFlow to replace Theano (Univ. Montreal,3-clause BSD). The two libraries are in fact quite similar. Some of the creators of Theano, such as Ian Goodfellow, went on to create Tensorflow at Google before leaving for OpenAI[29]
- Still fairly popular, but Yoshua Bengio announced on Sept. 28, 2017, that development on Theano would cease[29]

Caffe

- Caffe (Berkeley Vision, 2-Clause BSD) is a very efficient DL-Framework for computer vision, as it is not intended for other deep-learning applications such as text, sound or time series data[29]. It has a huge user basis, but arguably no future apart from legacy (it provides an outstanding "model zoo"): caffe2 seems to overcome some of this limitations without giving up the perks, so it is regarded as the natural follower

Brief Reviews - 2/6

- With a focus on dynamic neural networks for NLP[21, p.200], DyNet (Carnegie Mellon, Apache2.0) doesn't seem to offer anything that PyTorch can't do, and has the drawback of a much smaller user community[29]

- [29] claims the same about chainer (Preferred Networks, MIT License), "the leading neural network framework for dynamic computation graphs" until the advent of DyNet. Chainer claims as well to be faster than Tensorflow, CNTK and MXNet (see benchmark[1])

- The least popular (but also the youngest) of the regarded frameworks, skorch is "a scikit-learn compatible neural network library that wraps PyTorch"[17]. Probably still too soon to consider using it but worth keeping track of.

Brief Reviews - 3/6

- Powerful language, comprehensive algorithms backed by a huge knowledge system. All closed-source, details inaccesible
- Low popularity at StackOverflow? maybe because of their own docs/support
- Yearly subscription in $: Education 1500, Industry 3500, enterprise 9500[50]

- The only advantage of Torch (IDIAP+NEC+NYU, 3-Clause BSD) over PyTorch is that the LuaJIT compiler is extremely fast and portable, and allows "super-simple design and the compactness of traversing from high-level easy-to-use API to bare-metal C/assembly"[23]
- Everything else can be found in the now more popular Pytorch (in fact one of the fastest growing Frameworks in GitHub)

Brief Reviews - 4/6

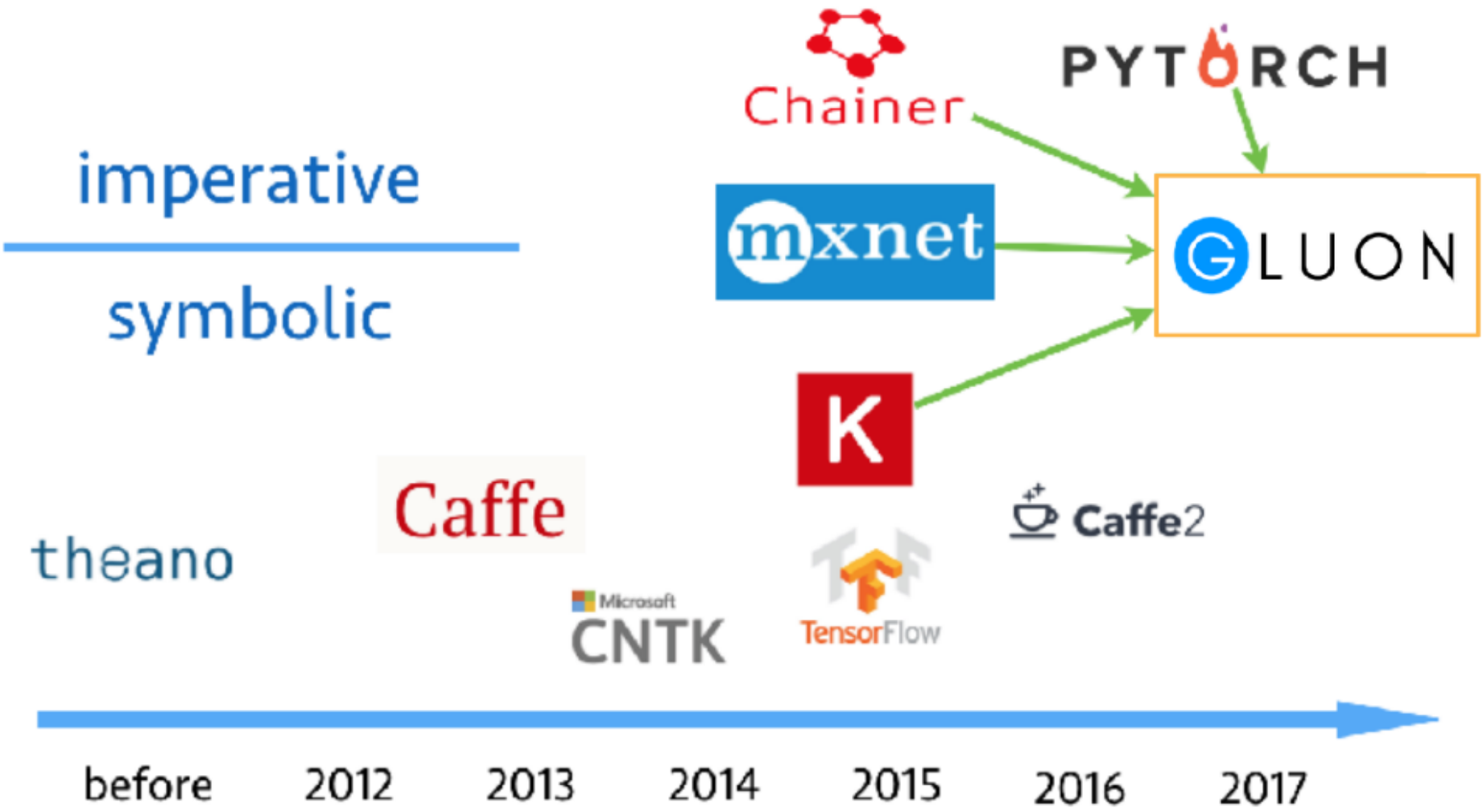
Keras

- Keras (Google, MIT License) is one of the most popular and established frameworks. It is designed as a high-level Python API that can run on the top of other frameworks (at the moment TensorFlow, CNTK, DL4J and Theano are supported and soon will MXNet too[38]).
- Keras claims to leverage the full potential of its backends seamlessly, being its features the union of the backends' features (plus the extra portability and the claimed "humanity" of the design). Many corporations and researchers seem to confirm this by using Keras actively

GLUON

- Gluon (Amazon&Microsoft, Apache2.0) was released in October, 2017. Similarly to Keras, intends to be on the top of many DL Frameworks for better&fast user experience
- It is distinguished from Keras by its imperative[26] style and dynamic computational graph, similar to Pytorch and Chainer[29]. Tutorial: [16]

Brief Reviews - 5/6



The strategy behind GLUON(from [16])

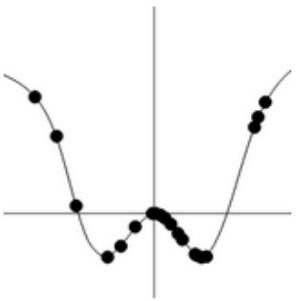
Brief Reviews - 6/6

Classify **MNIST** digits with a Convolutional Neural Network

Classify **CIFAR-10** with Convolutional Neural Network

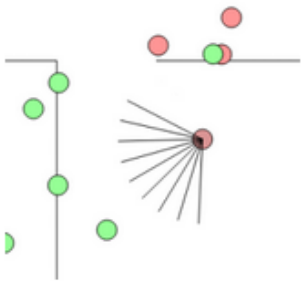
Interactively classify toy 2-D data with a Neural Network

Interactively regress toy 1-D data



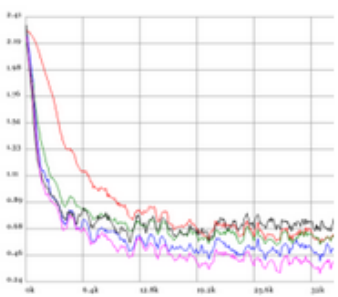
Train an MNIST digits Autoencoder

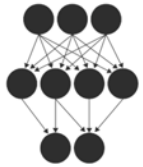
Reinforcement Learning with Deep Q Learning



Neural Network "paints" an image

Comparing SGD/Adagrad/Adadelta





ConvNetJS

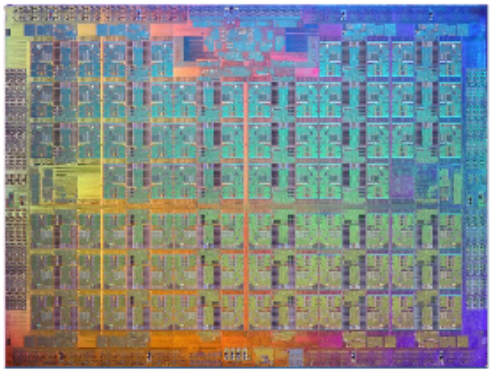
Deep Learning in your browser

- In-browser (or NodeJS server), single-file library by A. Karpathy
- Sequential (Keras-alike) building, saving trained models as JSON
- Low model coverage (no RNNs), but sufficient for many tasks
- Low performance, suitable for moderate inference or on-line learning
- Emphasis on interactivity (webapps) and communication (D3) of models

Browser Demos: CNN classification (MNIST and CIFAR), NN classification and regression, Autoencoder, reinf. learning with Deep Q Learning, color-regression (FCN) and comparing SGD/Adagrad/Adadelta (from [31]).


POPLAR®

The IPU

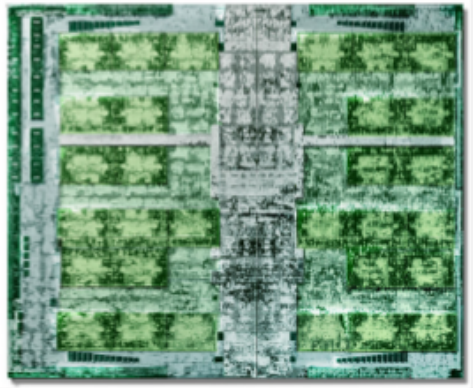


CPU

Scalar

Designed for office apps

Evolved for web servers

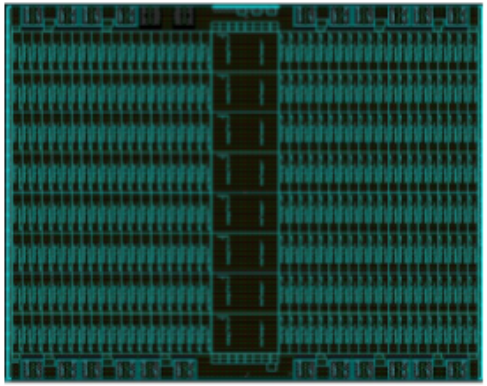


GPU

Vector

Designed for graphics

Evolved for HPC



IPU

Graph

Designed for intelligence

Intelligence requires a new architecture (from [36]).

- More bandwidth and memory for the models due to graph-based architecture
- Built-in suppor for algebra, low precision, noise generators, address-less communication
- Performance 2-3 orders of magnitude higher than best NVIDIA GPUs[35]

Poplar: Overview

From [37], [44]:

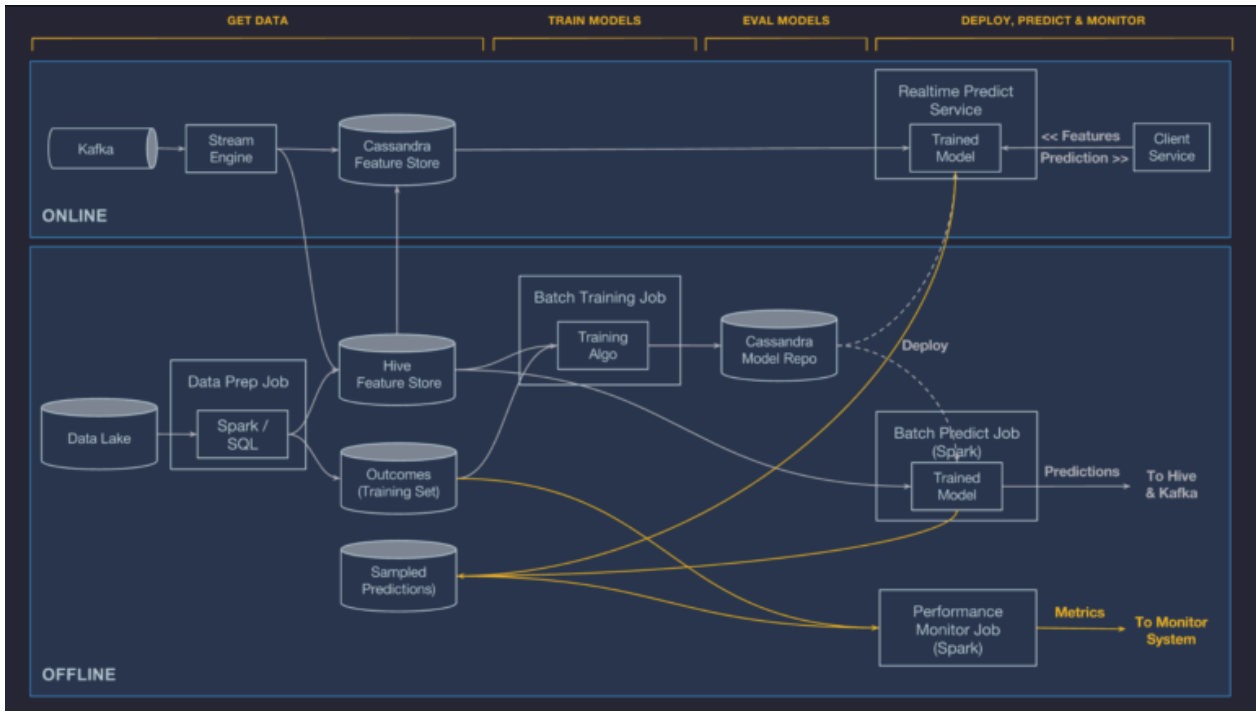
- Dependencies: IPU drivers, C++11 library with *poputil*, *popops*, *poplin*, *poprandom*, *popnn* (future: fft, robotics)
- Developed by GraphCore, library claimed to be OSS, but no public repos to the present date. Has Python API
- Deploys to IPU, like TF and MXNet (future: PyTorch, Caffe2, CNTK), but more optimized for it
- Translates into "codelets" that generate more vertices than TF graphs, usually millions (beyond tensor-centric)
- Deploys to IPU, like TF and MXNet, but more optimized for it
- Emphasis in non-dev tools for debugging and analysis

UBER

Michelangelo: Overview - 1/5

Like PyRo, developed by Uber. At the moment info only in [12].

Michelangelo allows **standardizing the workflows and tools** across teams through an end-to-end system that enables users across the company to easily **build and operate machine learning systems at scale.**

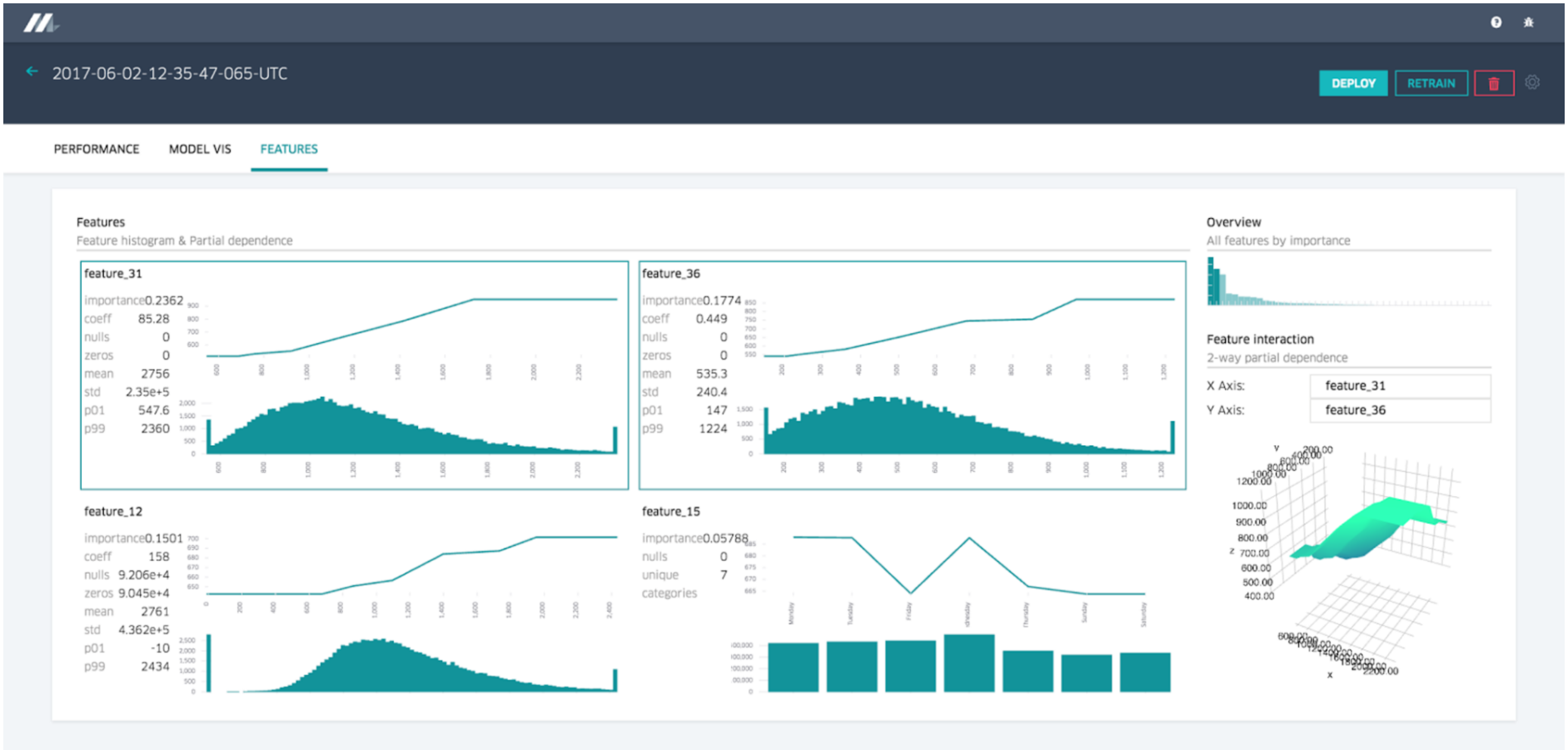


Every workflow involving Michelangelo covers 6 steps: **manage data** (1), **train** (2) **evaluate** (3) and **deploy** (4) models, **make** (5) and **monitor** (6) predictions.

Michelangelo: Overview - 2/5

1. **Manage data**: Scalable, performant access to the company's data lake and "feature store". Online and offline, aided by a DSL subset of Scala
2. **Train models**: Encapsulated models running on Apache infrastructures. Supports different models including DNNs, time-series, k-means... and is extendible
3. **Evaluate models**: After training, a "version object" is stored with metadata and trained parameters, and can be easily inspected via an integrated, model-specific web UI (picture in next slide)
4. **Deploy models**: Offline, Online or as Library. Re-training and re-deployment can be automatically scheduled
5. **Make predictions**: Highest traffic models around 250000preds/s.
6. **Monitor predictions**: A percentage of the predictions can be used to generate on-going metrics, which can trigger accions when a given threshold is reached

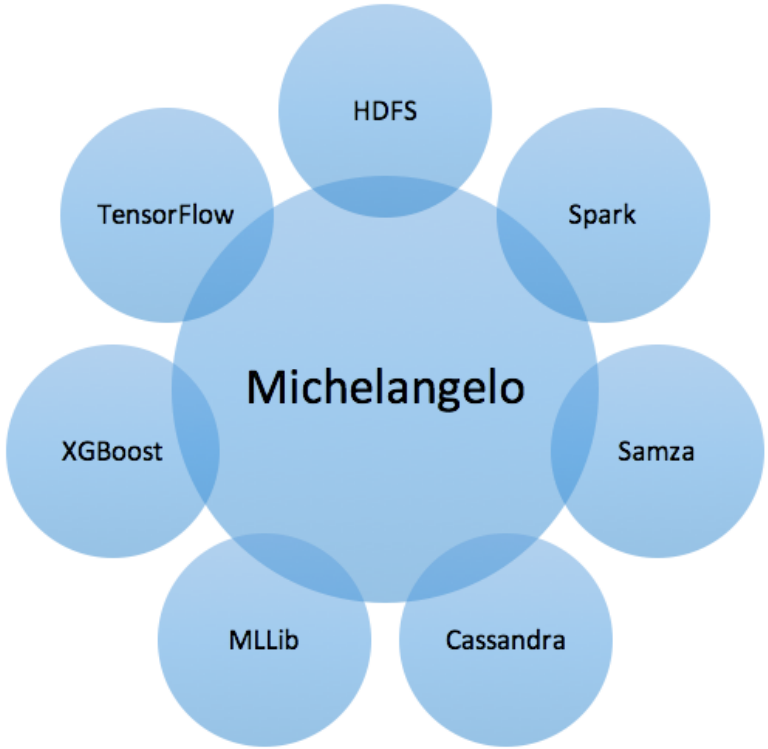
Michelangelo: Overview - 3/5



Screenshot of Michelangelo's web UI for model evaluation[12].

Michelangelo: Overview - 4/5

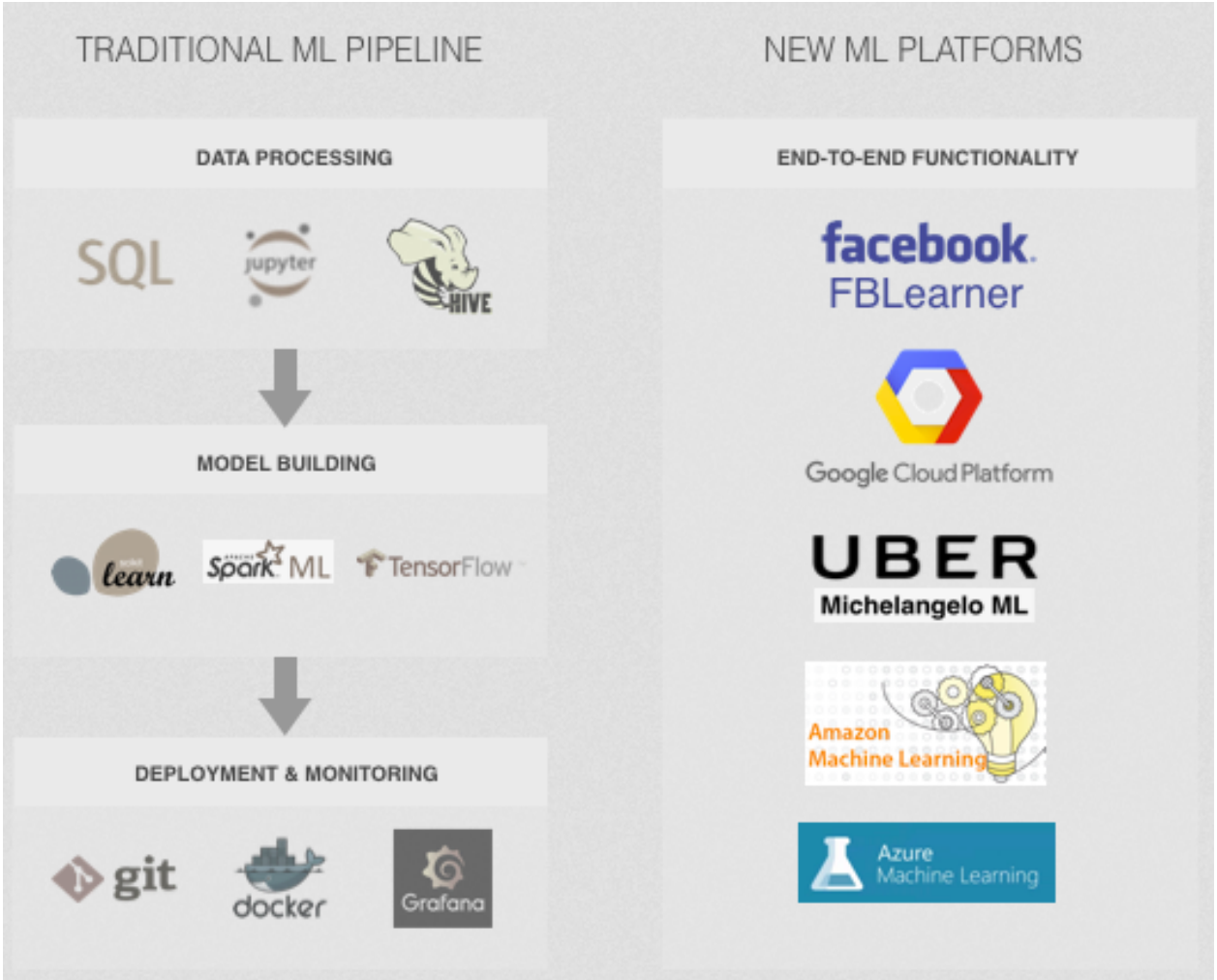
Uber began development in 2015. Unlike other well-known Uber products like Horovod (distributed DL framework) and PyRo (PPL), **it hasn't been open sourced**, but it uses HDFS, Spark, Samza (Kafka+Hadoop), Cassandra (DB), MLLib, XGBoost, and TensorFlow.



Challenges/Future:

- **AutoML**: Integrated metalearning, model search
- **Model Visualization**: Currently good support for tree-based models only
- **Online Learning**: Add further functionality to the existing one
- **Distributed DL**: "The user workflow of defining and iterating on deep learning models is sufficiently different from the standard workflow such that it needs unique platform support"

Michelangelo: Overview - 5/5



"Large tech companies have recently started to use their own centralized platforms for machine learning engineering, which more cleanly tie together the previously scattered workflows of data scientists and engineers"[9].

Section 3

Sum-Up

Interaction among Frameworks

Many vectors of interaction:

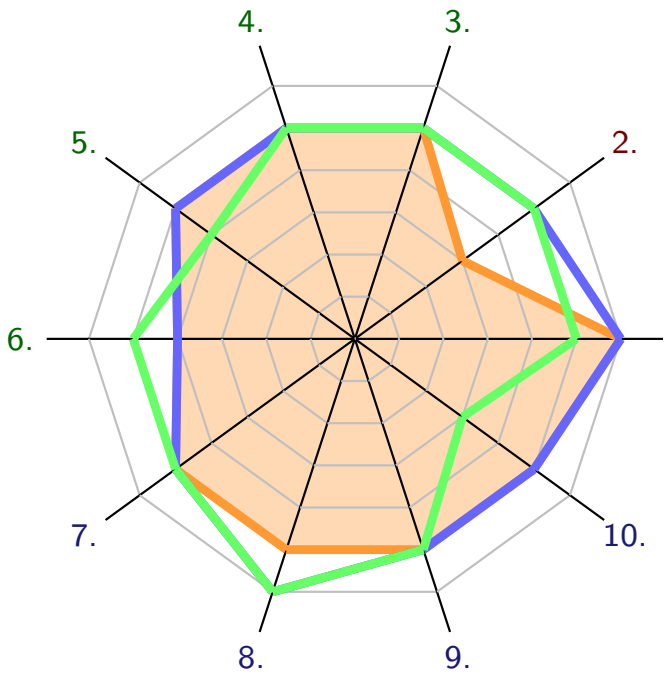
- Higher-order frameworks (e.g. Keras, Gluon)
- Cross-platform serialization: from scripts (caffe to caffe2 [30]) evolved to specifications (ONNX[4])
- Centralized platforms (e.g. Michelangelo, AML)
- Common languages (most frameworks have a Python API)

Different strategies:

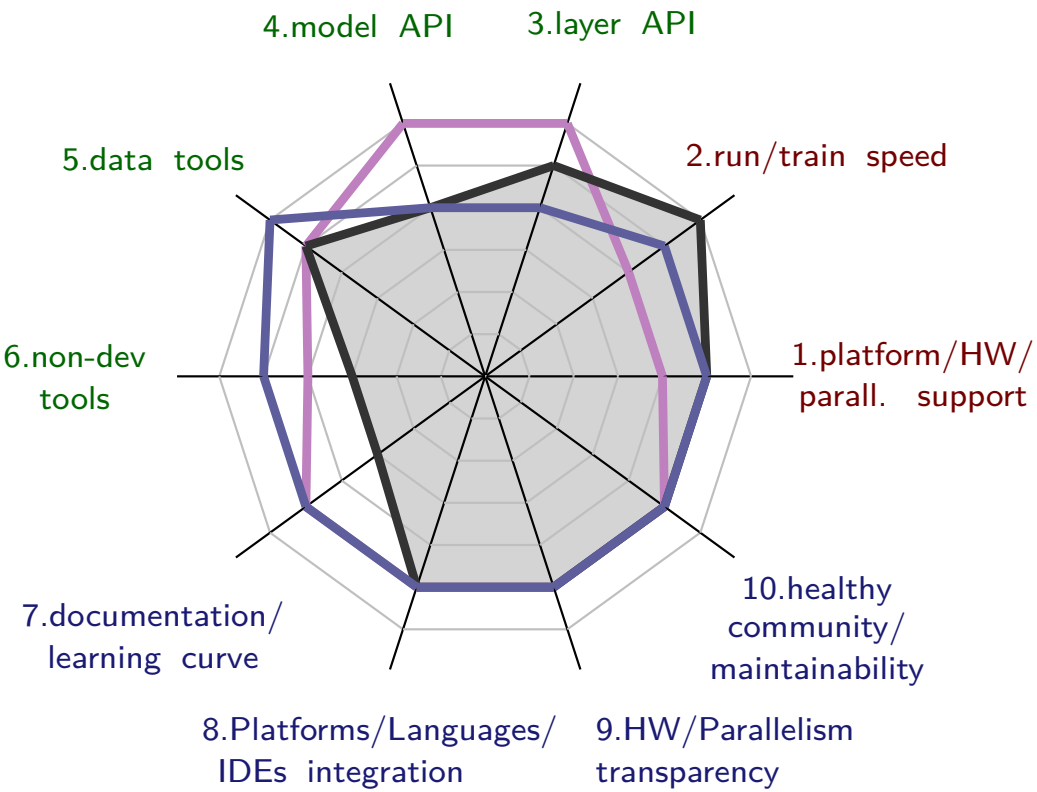
- Easy performance of mainstream models/HW (f.e. Caffe2)
- Popularity: broad coverage of platforms/languages (f.e. TF, CNTK)
- Integration into specific environments (f.e. DL4J, ConvNetJS)
- Emphasis on dynamic models (f.e. PyTorch, Chainer, DyNet)

Kiviat Summary

TF/MXNet/CNTK



PyTorch/Caffe2/DL4J



THANK YOU! Questions?

- **Open-Source(d)**:
  - PyTorch (Facebook) ← Torch (several)
  - Caffe2 (Facebook) ← Caffe (BVLC)
  - TensorFlow (Google) ← Theano (U. Montreal)
  - MXNet (Apache)
  - DL4J (Eclipse)
  - CNTK (Microsoft)
  - DyNet (Carnegie Mellon U.)
  - Chainer (U. Tokyo)
  - ConvNetJS (Karpathy)
  - POPLAR (GraphCore)
  - And more (not covered): DIGITS (NVIDIA), Paddle (Baidu), Scikit-learn, SINGA and MLlib (Apache), neon (Intel)
- **Closed-Source**:
  - Mathematica (Wolfram)
  - Michelangelo(Uber)
- **Higher-Order**:
  - Keras (on TF, Theano or CNTK)
  - Skorch (wraps PyTorch)
  - GLUON (on MXNET, CNTK)

References

[1] Takuya Akiba. *Performance of Distributed Deep Learning using ChainerMN*. February 2017. URL: https://chainer.org/general/2017/02/08/Performance-of-Distributed-Deep-Learning-Using-ChainerMN.html.

[2] GitHub user bartvm. *A comparison of deep learning frameworks*. October 2015. URL: https://gist.github.com/bartvm/69adf7aad100d58831b0.

[3] Atilim Gunes Baydin and Barak A. Pearlmutter. "Automatic Differentiation of Algorithms for Machine Learning". In: *CoRR* abs/1404.7456 (2014). URL: http://arxiv.org/abs/1404.7456.

[4] Sarah Bird and Dmytro Dzhulgakov. *ONNX V1 released*. December 6, 2017. URL: https://research.fb.com/onnx-v1-released/.

[5] Yaroslav Bulatov. *TensorFlow meets PyTorch with Eager execution*. Snapshot 19 Jan. 2018. URL: https://medium.com/@yaroslavvb/tensorflow-meets-pytorch-with-eager-mode-714cce161e6c.

[6] Joaquin Quinonero Candela. *Facebook and Microsoft introduce new open ecosystem for interchangeable AI frameworks*. September 2017. URL: https://research.fb.com/facebook-and-microsoft-introduce-new-open-ecosystem-for-interchangeable-ai-frameworks/.

[7] Vincent Chu. *We Need to Go Deeper: A Practical Guide to Tensorflow and Inception*. Snapshot 22 Jan. 2018. URL: https://medium.com/initialized-capital/we-need-to-go-deeper-a-practical-guide-to-tensorflow-and-inception-50e66281804f.

[8] Justin Domke. *A simple explanation of reverse-mode automatic differentiation*. March 2009. URL: https://justindomke.wordpress.com/2009/03/24/a-simple-explanation-of-reverse-mode-automatic-differentiation/.

[9] Catherine Dong. *The evolution of machine learning - Crunch Network*. August 8, 2017. URL: https://techcrunch.com/2017/08/08/the-evolution-of-machine-learning/.

[10] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[11] Awni Hannun. *Pytorch or TensorFlow?* 17 Aug. 2017. URL: https://awni.github.io/pytorch-tensorflow/.

[12] Mike Del Balso Jeremy Hermann. *Meet Michelangelo: Uber's Machine Learning Platform*. September 5, 2017. URL: https://eng.uber.com/michelangelo/.

[13] Yangqing Jia. *Comment and slides about birth of Caffe2*. 19 Apr. 2017. URL: https://github.com/caffe2/caffe2/issues/274#issuecomment-294999833.

[14] Yangqing Jia. *Improving Caffe: Some Refactoring (slides)*. 2015. URL: tutorial.caffe.berkeleyvision.org/caffe-cvpr15-improving.pdf.

[15] Dominique Luna. *Automatic Differentiation, A.K.A please save me from backprop! (blog entry)*. November 2015. URL: https://domluna.me/automatic-differentiation-a.k.a-please-save-me-from-backprop/.

[16] Zachary Lipton Mu Li. *Deep Learning in Apache MxNet Gluon*. September 2017. URL: https://github.com/zackchase/gluon-slides/blob/master/sept18-gluon.pdf.

[17] Daniel Nouri. *skorch (GitHub repository)*. Snapshot 19 Jan. 2018. URL: https://github.com/dnouri/skorch.

[18] Adrian Rosebrock. *How to plot accuracy and loss with mxnet*. December 25th 2017. URL: https://www.pyimagesearch.com/2017/12/25/plot-accuracy-loss-mxnet/.

[19] Matthew Rubashkin. *Getting Started with Deep Learning - A review of available tools*. February 15, 2017. URL: https://svds.com/getting-started-deep-learning/.

[20] StackOverflow user Salvador Dali. *Answer to: Is gradient in the tensorflow's graph calculated incorrectly?* June 2017. URL: https://stackoverflow.com/a/45133884.

[21] Maruan Al-Shedivat et al. *A Statistical Machine Learning Perspective of Deep Learning: Algorithm, Theory, Scalable Computing*. 2017. URL: http://www.petuum.com/pdf/DL-all-final2.pdf.

[22] Shaohuai Shi et al. "Benchmarking State-of-the-Art Deep Learning Software Tools". In: *CoRR* abs/1608.07249 (2016). arXiv: 1608.07249. URL: http://arxiv.org/abs/1608.07249.

[23] HackerNews user smhx. *Interview to one of Torch7's maintainers*. 2014. URL: https://news.ycombinator.com/item?id=7929216.

[24] GitHub user timqian. *GitHub Star Hystory (webapp)*. Snapshot 15 Jan. 2018. URL: http://www.timqian.com/star-history/#tensorflow/tensorflow&Microsoft/CNTK&BVLC/Caffe&caffe2/caffe2&torch/torch7&pytorch/pytorch&Theano/Theano&apache/incubator-mxnet&gluon-api/gluon-api&deeplearning4j/deeplearning4j&NervanaSystems/neon&NVIDIA/DIGITS&clab/dynet&PaddlePaddle/Paddle&scikit-learn/scikit-learn&dnouri/skorch&apache/incubator-singa&keras-team/keras&apache/spark.

[25] GitHub user u39kun. *Deep Learning Benchmark*. January 2018. URL: https://github.com/u39kun/deep-learning-benchmark.

[26] various/unknown. *Apache MXNet - A flexible and efficient library for deep learningMxNet*. Snapshot 22 Jan. 2018. URL: https://mxnet.apache.org/.

[27] various/unknown. *Caffe2 docs - Learn More*. Snapshot 19 Jan. 2018. URL: https://caffe2.ai/docs.

[28] various/unknown. *Chainer - Comparison with Other Frameworks*. July, 2017. URL: https://docs.chainer.org/en/v4.0.0b1/comparison.html.

[29] various/unknown. *Comparing Top Deep Learning Frameworks: Deeplearning4j, PyTorch, TensorFlow, Caffe, Keras, MxNet, Gluon CNTK*. 2017. URL: https://deeplearning4j.org/compare-dl4j-tensorflow-pytorch.

[30] various/unknown. *Converting Models from Caffe to Caffe2*. Snapshot 26 Jan. 2018. URL: https://caffe2.ai/docs/caffe-migration.html.

[31] various/unknown. *ConvNetJS - Deep Learning in your browser*. Snapshot 22 Jan. 2018. URL: http://cs.stanford.edu/people/karpathy/convnetjs.

[32] various/unknown. *deeplearning4j (GitHub repository)*. Snapshot 19 Jan. 2018. URL: https://github.com/deeplearning4j/deeplearning4j.

[33] various/unknown. *DL4J - Getting Started*. Snapshot 19 Jan. 2018. URL: https://deeplearning4j.org.

[34] various/unknown. *DL4J - Learn More*. Snapshot 19 Jan. 2018. URL: https://deeplearning4j.org/devguide#deeplearning4j-some-things-to-know.

[35] various/unknown. *GRAPHCORE - BENCHMARKS*. Snapshot 26 Jan. 2018. URL: https://cdn2.hubspot.net/hubfs/729091/NIPS2017/NIPS%2017%20-%20benchmarks%20final.pdf?t=1513603679766.

[36] various/unknown. *GRAPHCORE - INTELLIGENT PROCESSING UNIT*. Snapshot 26 Jan. 2018. URL: https://cdn2.hubspot.net/hubfs/729091/NIPS2017/NIPS%2017%20-%20IPU.pdf?t=1513603679766.

[37] various/unknown. *GraphCore: Poplar Overview*. Snapshot 26 Jan. 2018. URL: https://www.graphcore.ai/hubfs/assets/Poplar%C2%81%20technical%20overview%20NEW%20BRAND.pdf?t=1500587501625.

[38] various/unknown. *Keras Documentation*. Snapshot 19 Jan. 2018. URL: https://keras.io.

[39] various/unknown. *MxNet (GitHub repository)*. Snapshot 19 Jan. 2018. URL: https://github.com/apache/incubator-mxnet.

[40] various/unknown. *MxNet (Webpage)*. Snapshot 19 Jan. 2018. URL: https://mxnet.incubator.apache.org.

[41] various/unknown. *MxNet (Wikipedia article)*. Snapshot 16 Jan. 2018. URL: https://en.wikipedia.org/wiki/MXNet.

[42] various/unknown. *ONNX (GitHub repo)*. Snapshot 26 Jan. 2018. URL: https://github.com/onnx/onnx.

[43] various/unknown. *Per-tag Popularity at StackOverflow*. Snapshot Jan. 2018. URL: https://insights.stackoverflow.com/trends.

[44] various/unknown. *PROGRAMMING THE IPU - POPLAR*. Snapshot 26 Jan. 2018. URL: https://cdn2.hubspot.net/hubfs/729091/NIPS2017/NIPS%2017%20Poplar%20Final.pdf?t=1514539729071.

[45] various/unknown. *Pytorch - About*. Snapshot 19 Jan. 2018. URL: http://pytorch.org/about/.

[46] various/unknown. *R Interface to TensorFlow*. Snapshot 15 Jan. 2018. URL: https://tensorflow.rstudio.com.

[47] various/unknown. *TensorFlow in other languages*. Snapshot 15 Jan. 2018, API r1.4. URL: https://www.tensorflow.org/extend/language_bindings.

[48] various/unknown. *TensorFlow wikipedia article*. Snapshot 15 Jan. 2018. URL: https://en.wikipedia.org/wiki/TensorFlow.

[49] various/unknown. *The Microsoft Cognitive Toolkit*. July 2017. URL: https://docs.microsoft.com/en-us/cognitive-toolkit/.

[50] various/unknown. *Wolfram Mathematica (webpage)*. Snapshot 19 Jan. 2018. URL: https://www.wolfram.com/mathematica/.

[51] Pradeep Viswav. *8 reasons why you should switch from TensorFlow to Microsoft Cognitive Toolkit (CNTK)*. Juny 2017. URL: https://mspoweruser.com/microsoft-8-reasons-why-you-should-switch-from-tensorflow-to-cntk/.