Evolution of Tensorflow

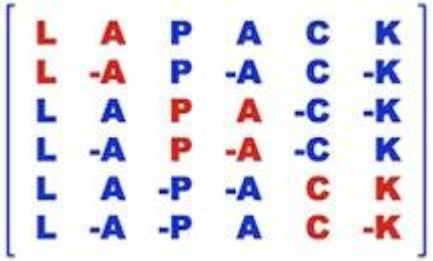
Michael Simbirsky
Software Engineer at Google Research

AI Ukraine
Kharkiv, Ukraine, Sep. 2017

Plan

- Predecessors and the birth of ML at Google
- Three pillars: algorithm, hardware and data
- Flow graphs in TF and beyond.
- TF and Google Cloud
- What’s next for TF? XLA, AutoML and others

LAPACK -- the mother of them all

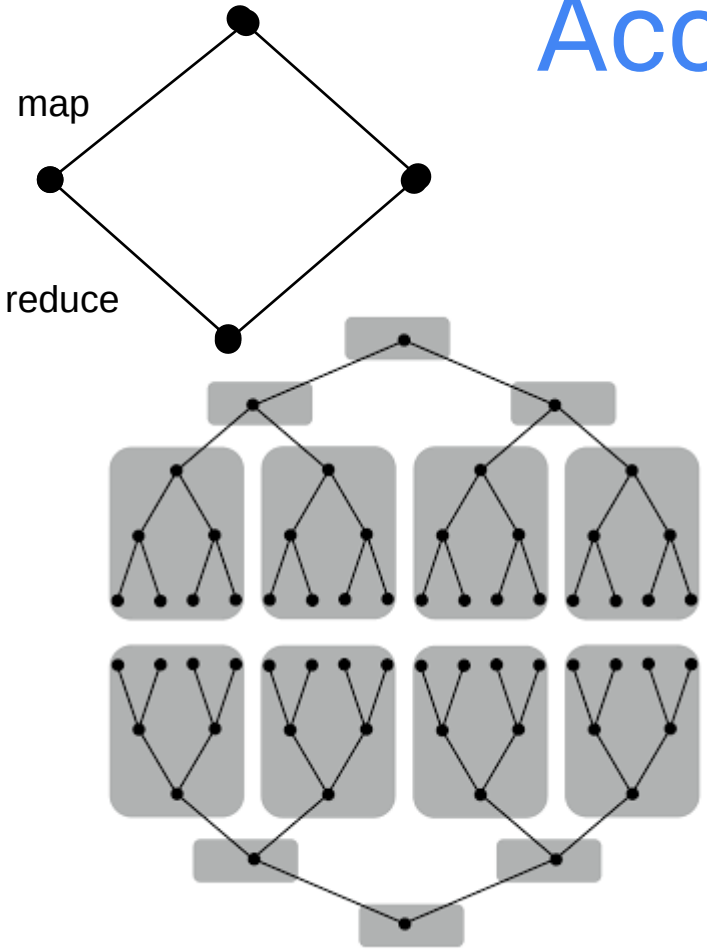


- Written in Fortran 77 and 90 in 1992 with support from US government
- LAPACK uses BLAS(1,2,3):
  - BLAS1: scalar on vector and dot-product
  - BLAS2: matrix times vector
  - BLAS3: matrix times matrix
- LAPACK wrappers and extensions: R, Matlab, SciPy, Numpy, Math Kernel Lib (Intel MKL), etc.

Lapack is the ultimate answer to many questions:

- Will TF/ML work on this chip? --- the answer is: provided Lapack works
- Why tensors are immutable? --- the answer is: *because of Lapack*
- etc.

Accelerating LAPACK



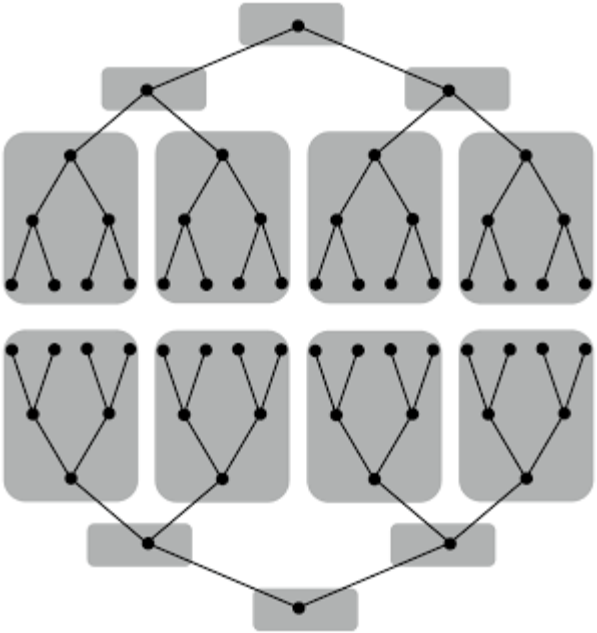
Hardware acceleration and parallelization for vector-vector operations:

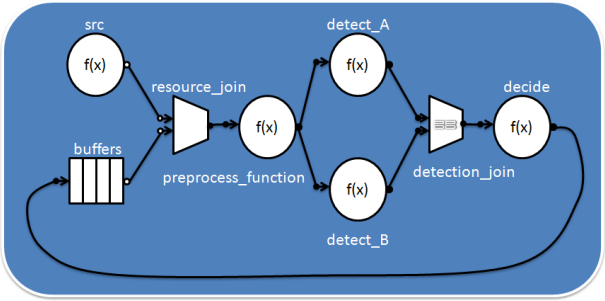
- MMX → CUDA → TPU

Software parallelization:

- Almost all algorithms in LAPACK are recursive: split the matrix, perform op, merge. The paradigm known in algorithmic lingo as “Map-Reduce”
- IMHO, LAPACK parallel implementation by Intel (actually, by Cilk) led to development of TBB (c.2005)

Workflow graph





- One of the central ideas of TBB is “flow” graph which is conceptually quite similar to TF Graph.
  - Central idea: describe *what* you need, let the system decide *when and how*
  - More on this later

- While we are here, perfect ref on parallel “Design Patterns”, now part of TBB docs.

Google starts using ML

- Google has very specific infrastructure with thousands of CPU and GPU available for parallelism.
- Existing ML platforms did not scale well enough, so **Jeff Dean** et al. started DistBelief (2011, NIPS 2012)
- **Geoff Hunton** applied Restricted Boltzmann machines to create good initial state for “deep” NN (~2010)
- Transition from “wide” to “deep” NNs made possible by advances in hardware, algorithms and data -- circa 2010-2011.

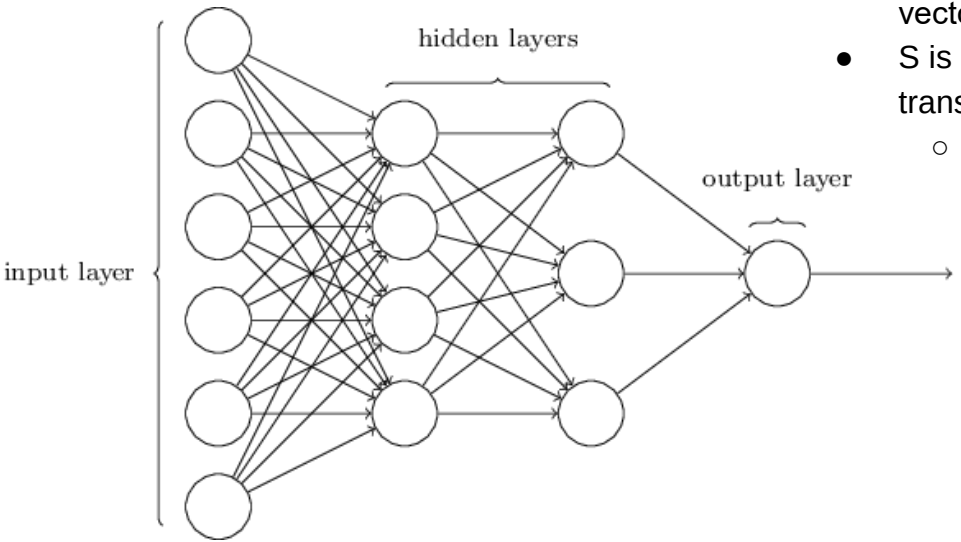
Jeff Dean at Google

- Author or co-author of MapReduce, BigTable, Google Brain, DistBelief, TensorFlow.

Some of Jeff Dean Facts

- Google: it's basically a Jeff Dean's side project.
- Jeff Dean's PIN is the last 4 digits of pi
- Jeff Dean got promoted to level 11 in a system where max level is 10. (actually True.)
- When Jeff gives a seminar at Stanford, it's so crowded Don Knuth has to sit on the floor. (True)
- Compilers don't warn Jeff Dean. Jeff Dean warns compilers
- Jeff Dean can instantiate abstract classes.
- gcc -O4 sends your code to Jeff Dean for a complete rewrite.
- Jeff Dean doesn't exist, he's actually an advanced AI created by Jeff Dean.

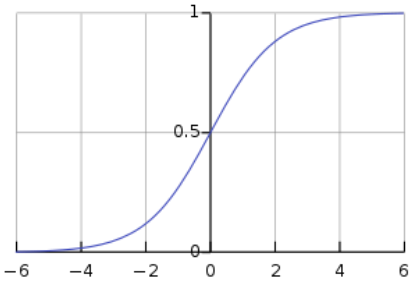
Central element of the flow: TF Graph

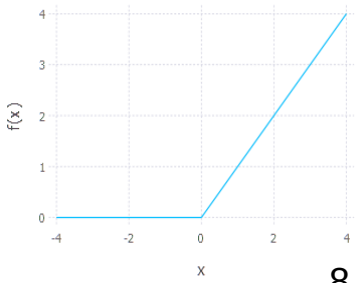


- Mathematically, this Neural Network (NN) represents a function

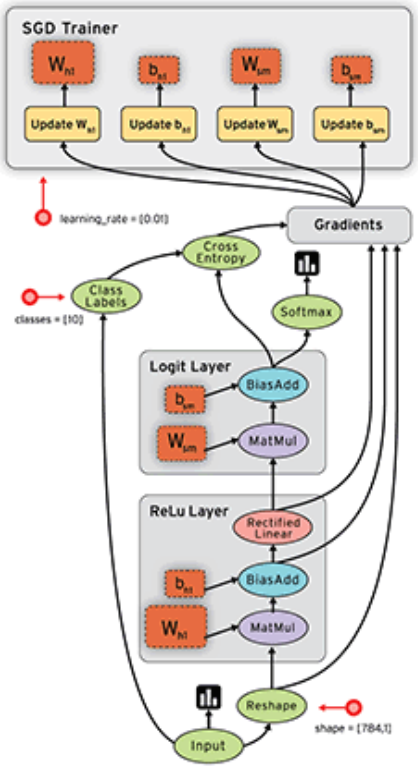
  $$L_{n+1} = S (W_n * L_n + B_n)$$

- where L_n is a vector, W_n is a matrix of “weights”, B_n is a vector of “biases”.
- S is so called activation function, usually element-wise transform like *logistic function*, *tanh*, *relu*, dropout, etc.
  - (modern NNs usually use relu as much faster alternative)





Important features of the TF Graph



- **Extendable** (one can add “input” nodes for forward feed, “loss” node for training or eval, “gradient” nodes for back-propagation, etc.)
- **Distributed**: Nodes and subgraphs can be placed onto specific device
  - with tf.device(name):
- **Persistent**: you can save the graph or part of it and load on a different machine
- Graph topology is modifiable -- UNTIL YOU LAUNCH THE SESSION
  - E.g. SGD optimizer can modify the graph by adding extra nodes, etc.

Most important slide about the Graph

- Once you call session.Run(), Graph becomes non modifiable (except weights and biases)
- Once again: IT IS NOT MODIFIABLE
- It means, TF can inspect, analyze, optimize, split, clone, transform the graph any way it sees fit.

- Two stages of TF Graph:
  - Construction time (essentially, “at compile time”)
  - Session time (aka “runtime” or “training time”)

- Any “compile time” and JIT self-reflections and optimizations are beneficial:
  - Typical training time at Google: 1-2 weeks on 100-1000 machines
  - with 10-1000 Tb of data
- YOU DO NOT WRITE IN PYTHON, YOU WRITE IN TENSORFLOW!
- Your compiler is [inside] Tensorflow Session object

Tensorflow “compilation” elements

- All modern compilers do “truth propagation” aka “constant folding”
- One class of important constants in TF are tensor shapes
  - my_tensor.shape -- construction-time information about shape, e.g. [?,3,5].
  - tf.shape(my_tensor) -- run-time tensor op

- ==> Session.run() verifies that shapes are consistent with operations.

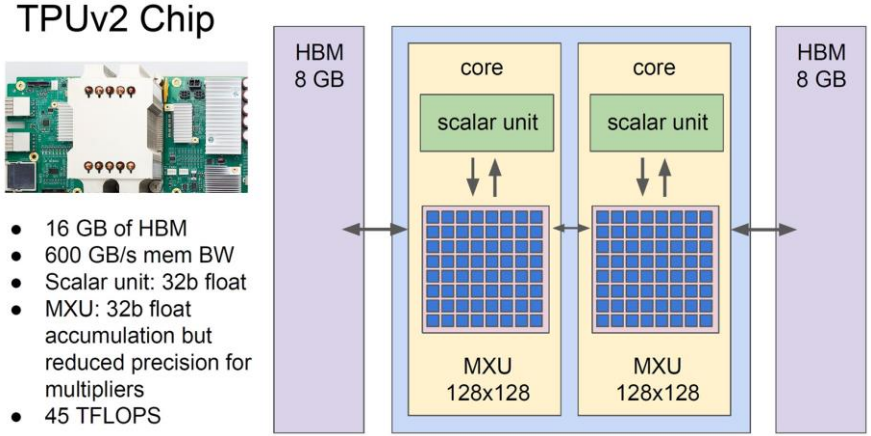
- Graph is a graph ==> all kind of graph algorithms (min-cut, clustering, etc.) can produce valuable insights.

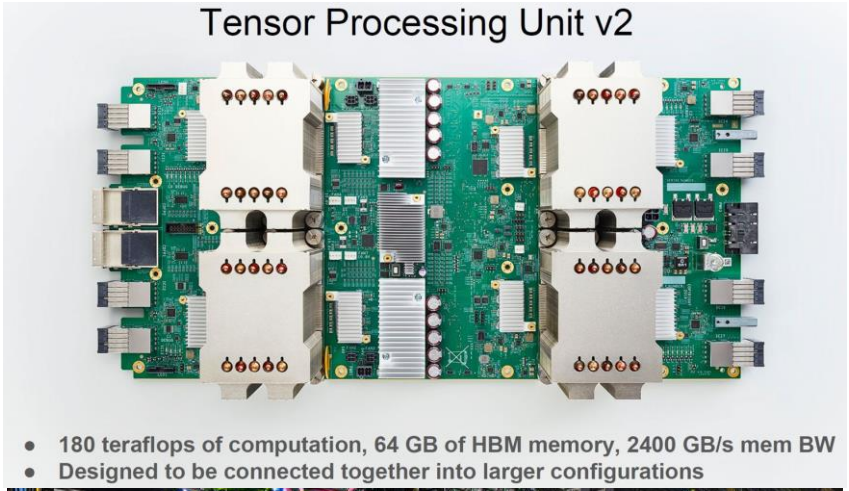
- COMPILATION TIME IS NEGLIGIBLE in comparison with TRAINING or INFER TIME

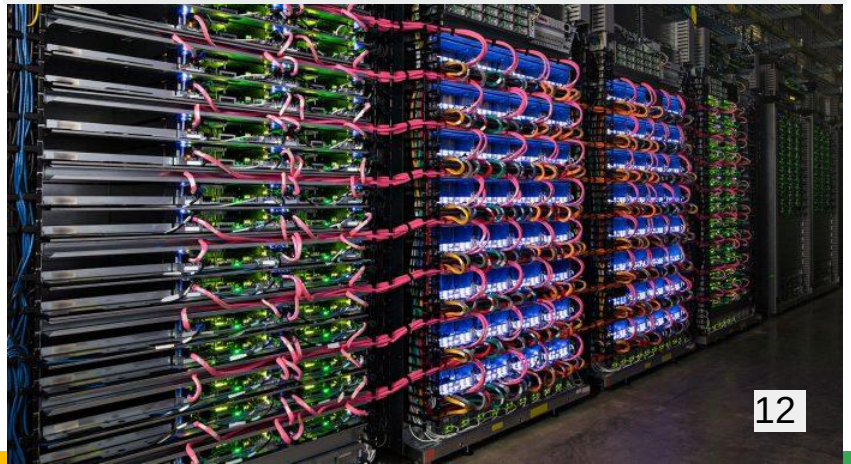
- *Bad news: error messages are cryptic and buried within Python stack trace*

Another twist in optimizations: TPUs

- SGD is so noisy, no sense to compute AxB with high precision
- ⇒ instead of 32- and 64-bit FLOPs one can use 16-bit or even 8-bit FLOPs
- ⇒ one gets much faster processing







Source: https://www.servethehome.com/case-study-google-tpu-gddr5-hot-chips-29/

What made ML possible?

Three developments made current explosion of Machine Learning possible:

- Better hardware: large amount of CPU, GPU and later TPU
- Better algorithms: DNNs, CNNs, RNNs (LSTM, GRU, Attention), GANs, RL, etc.
- Huge, really enormous amounts of data

Monetization

- Data is not for sale!
- Hardware is not for sale!
- It turns out, of these three, the algorithms is the least valuable part

==> Google's decision: open source Tensorflow

Where is the money (for Google)?

-- in AI transformation

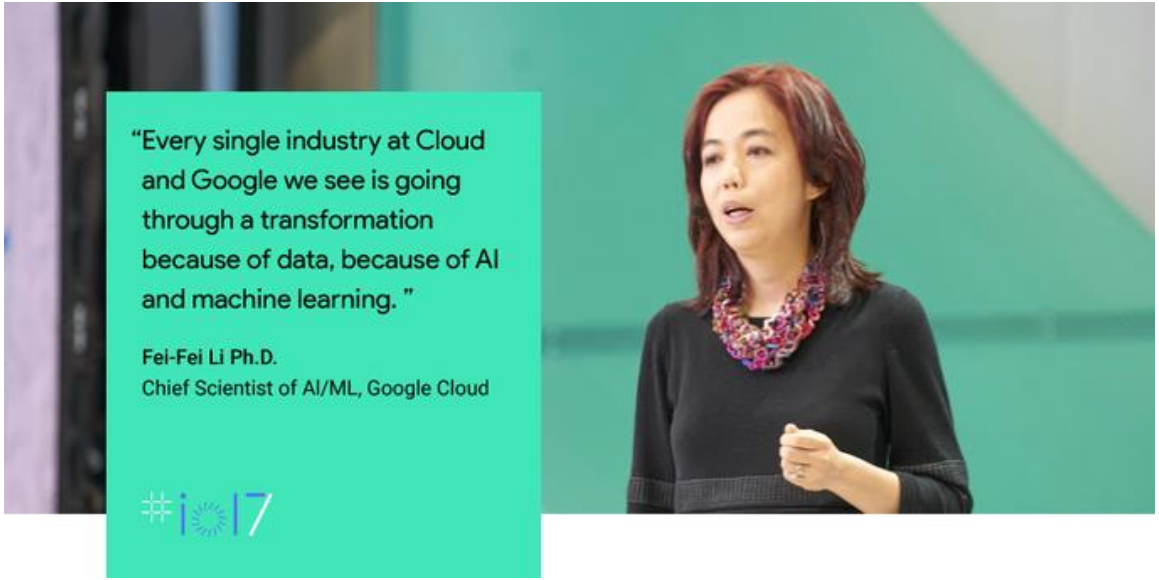
- CEO Sundar Pichai said that all of the company and its products are being revamped to be “AI-first”

-- in the Cloud

- Google Cloud AI:
  - Large Scale Machine Learning Service
  - Image and Video analysis API
  - Speech Recognition API
  - Text analysis API
  - Translation API

People behind Google Cloud

- Diana Greene -- founder and CEO of VMWare (till 2008)
- now -- SVP for Google Cloud



- Fei-Fei Li -- Stanford professor, director of Stanford AI lab, creator of the ImageNet project and competition
- Chief Scientist at Google Cloud

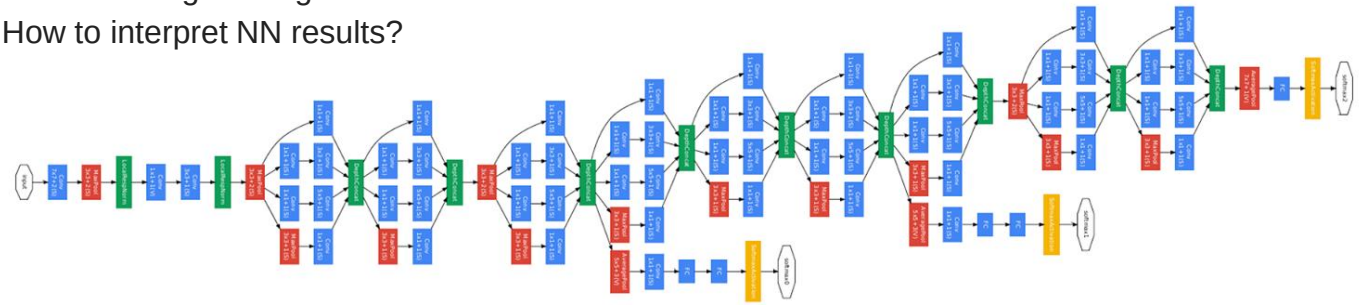
Where does it take all of us?

- Google employs at least 30,000 R&D engineers
- With “AI-first” principle many of them become users of Tensorflow
- They experiment with enormous amounts of data collected by Google (Search, Ads, Youtube, Maps, Android, etc.)
- They build complicated and huge NNs

- They face the same challenges you do:
  - What NN will work the best?
  - How to debug training?
  - How to interpret NN results?

- **AutoML** (Auto Machine Learning) ... has the ability to change its own architecture”

- **Tensorboard**
  - started as learning visualization tool
  - With open plugins it will convert to ML IDE



Thank you!