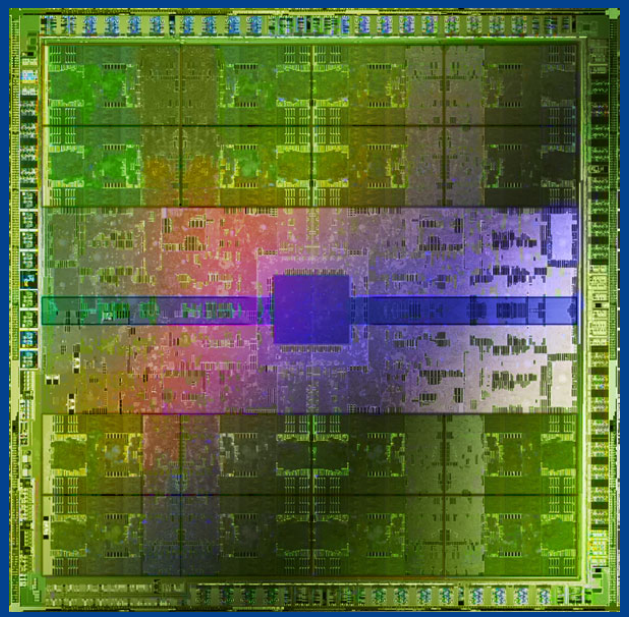
AARHUS
UNIVERSITY
SCHOOL OF ENGINEERING

MACHINE LEARNING

LESSON 14: Hardware for Machine Learning

CARSTEN EIE FRIGAARD

SPRING 2019



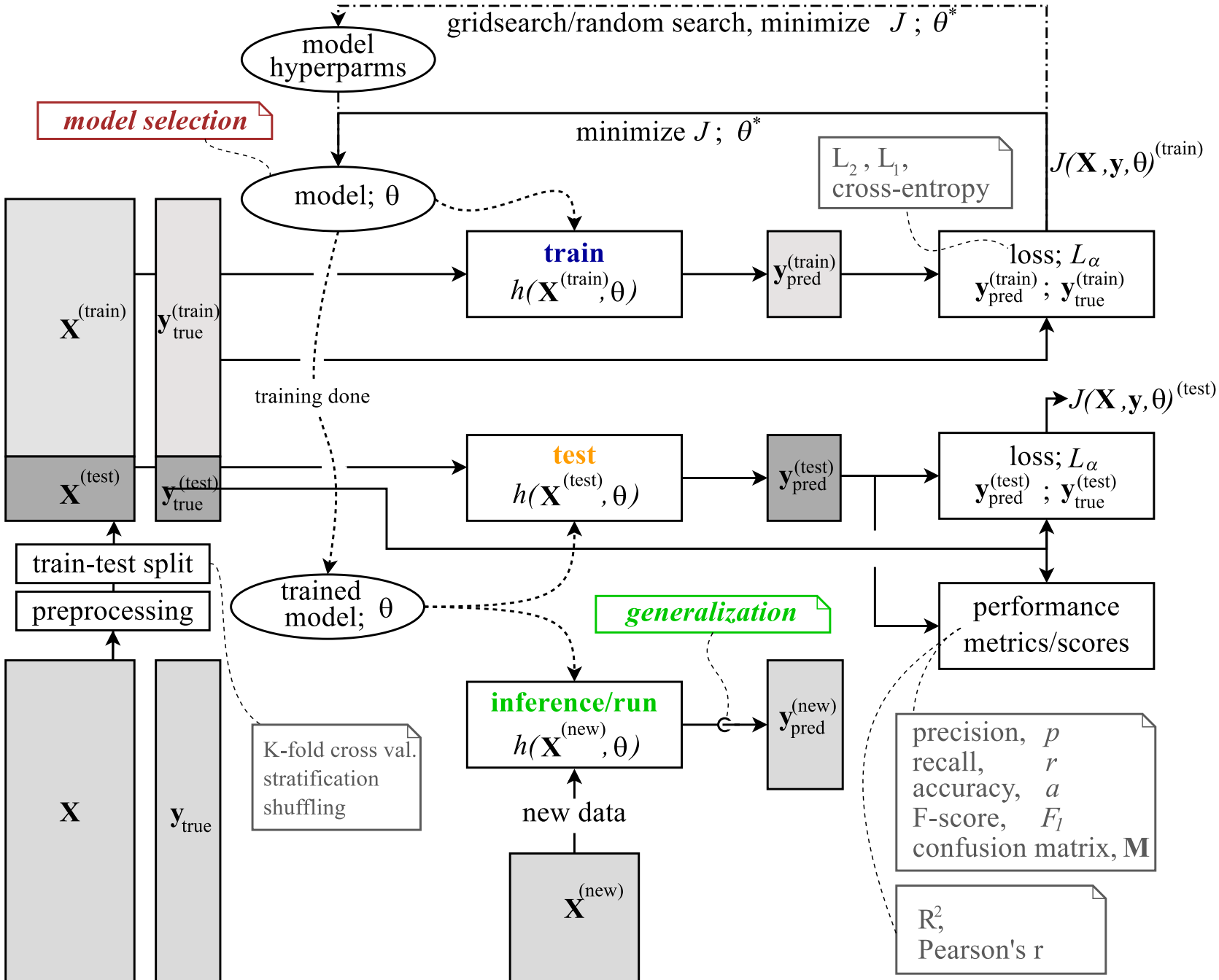
L14: Hardware for ML: Agenda

- J3: Criteria and Poster notes.
- Please all come to the Poster-session: 28-05-19
- Course Evaluation.

- RESUMÉ: end-to-end Machine Learning

- Hardware for Machine Learning
  - CPUs
  - GPUs
  - TPUs
  - Exotic Hardware

RESUMÉ: ML block diagram



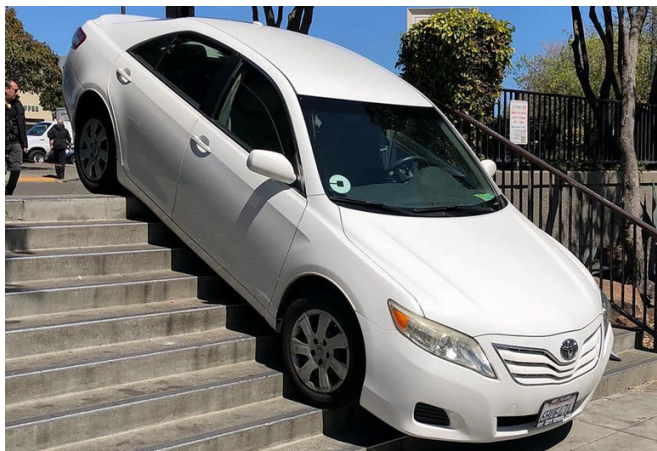
Hardware for Machine Learning

Methods and Terminology

Objective:

Why optimize using 'application specific' hardware?

- Effectiveness:
  - cost of purchasing/operating systems,
    μ=FLOPS/$
    η=FLOPS/Watt
  - cut-down developer waiting time,
  - make modelling iterations fast (say minutes).
- Big-data:
  - enable training on x-large data.
- Real-time constraints:
  - inference (on visual data)
    in real-time,
  - low-power constraints.

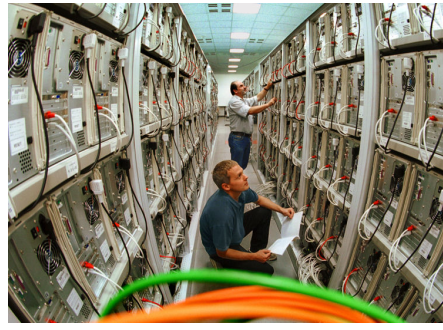


Hardware for Machine Learning

Methods and Terminology

Heterogen computing: systems that use more than one kind of processor or cores, say CPU + GPU.

Cluster computing: (loosely) or tightly connected computers that work together; can be viewed as a single system.



HPC: High-performance computing; a 'super-computer' with high level of performance (vs. general-purpose computer).

Flynn's taxonomy:

SISD: single instruction, single data,

SIMD/SIMT: single instruction, multiple data/threads (data parallel); this one is our objective,

(MISD: multiple instructions, single data (fault tolerance)),

(MIMD: multiple instructions, multiple data (distributed)).

Methods and Terminology

Amdahl's law

What speedup can we expect when optimizing/ parallelizing a program?

$$S = \frac{1}{(1-p) + p/s}$$

where

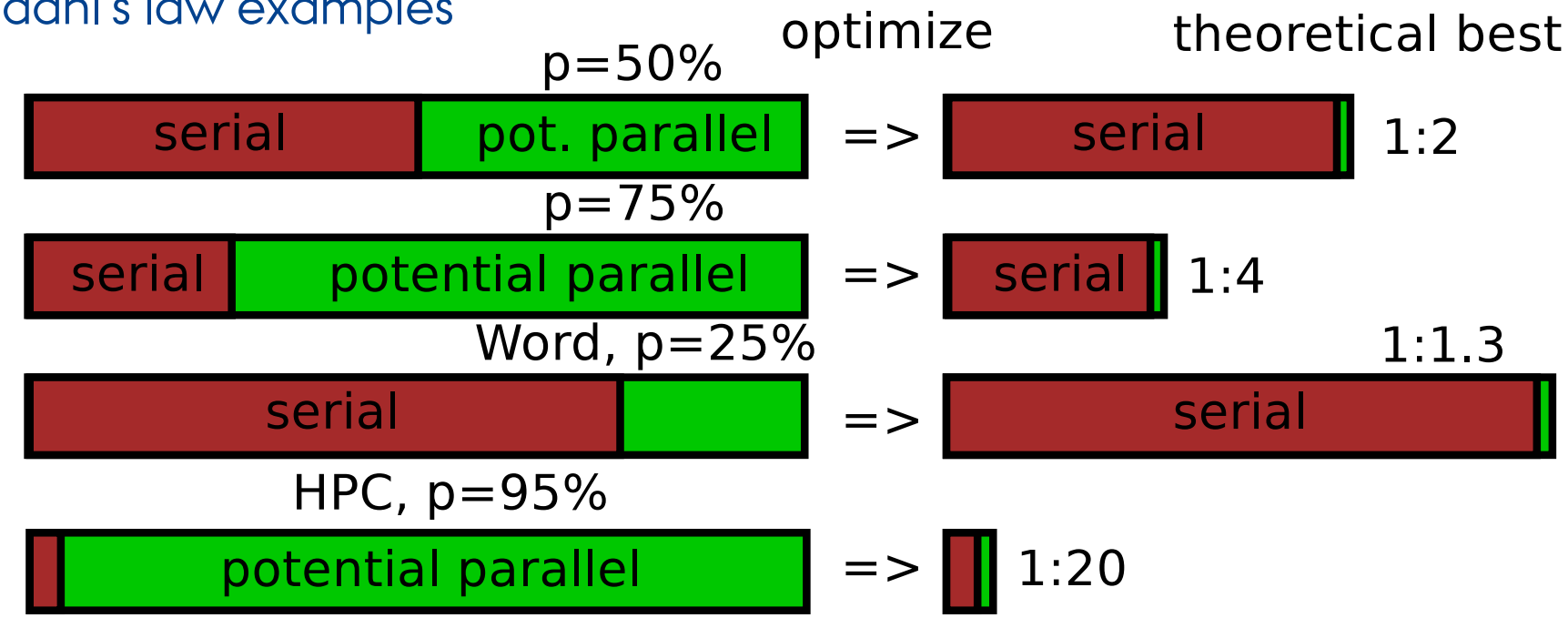
- S: total program speedup,
- p: fraction of the program that can be run in parallel,
- s: parallel fraction speedup factor.

If we take $s = \infty$ the theoretical max. total speedup will be

$$S_{\text{max}} = \frac{1}{(1-p) + p/\infty} = \frac{1}{1-p}$$

Methods and Terminology

Amdahl's law examples

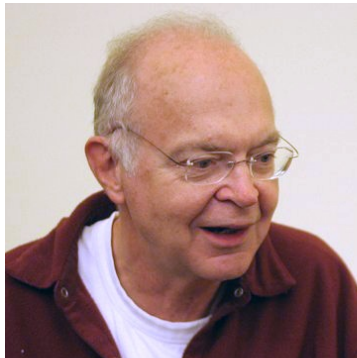


And there is seldom need to optimize unless $S_{max} \gg 2$.

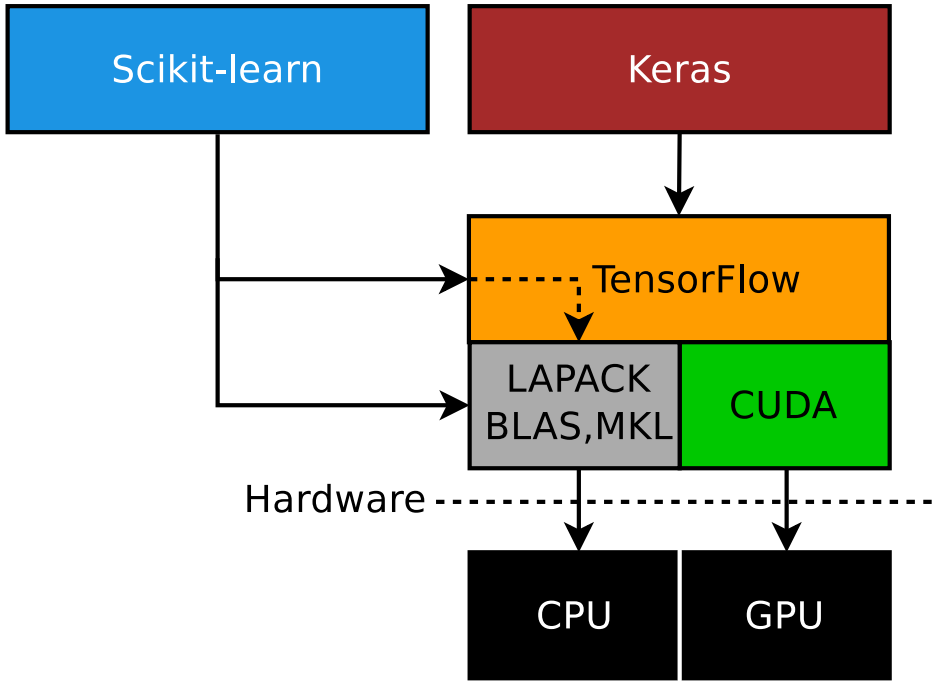
A 1:5 speedup is a good, ad-hoc compromise btw. a fast program and a costly optimization dev. phase.

What does Donald E. Knuth say about premature optimization?

"Root of all evil"!



RESUMÉ: Keras and Tensorflow



GP-GPU: General-Purpose Graphics Processing Unit...or just **GPU**.

CUDA: Compute Unified Device Architecture, API for SIMD/SIMT on GPU,

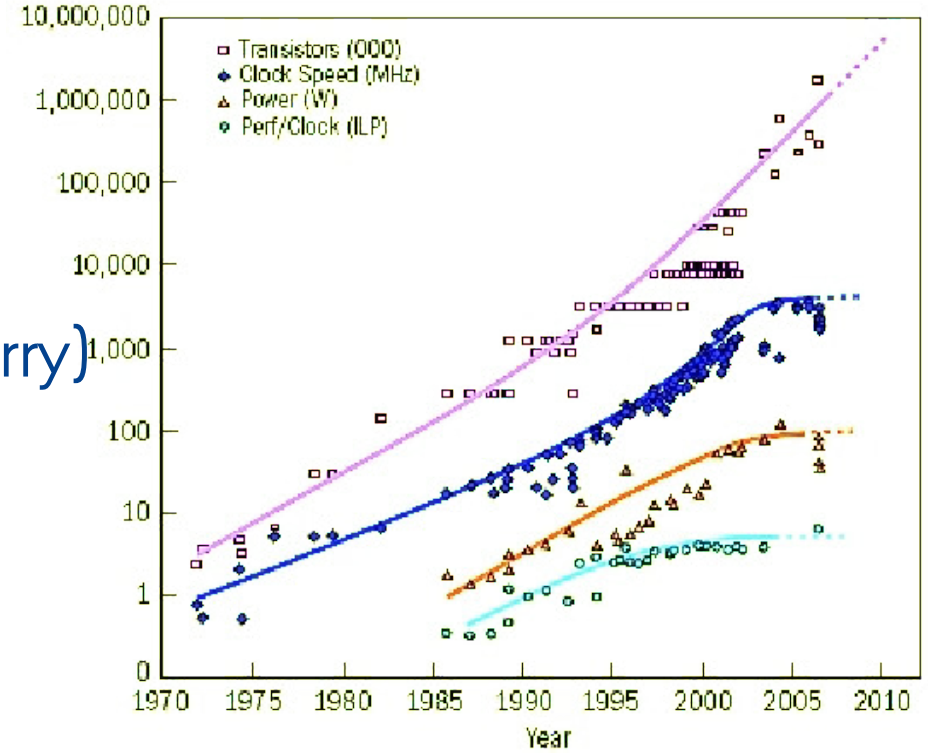
LAPACK: Linear Algebra Package, numerical linear algebra lib, with roots in FORTRAN 77,

BLAS: Basic Linear Algebra Subprograms; vector/matrix lib,

MKL: Math Kernel Library; fast Intel-arch optimized math lib.

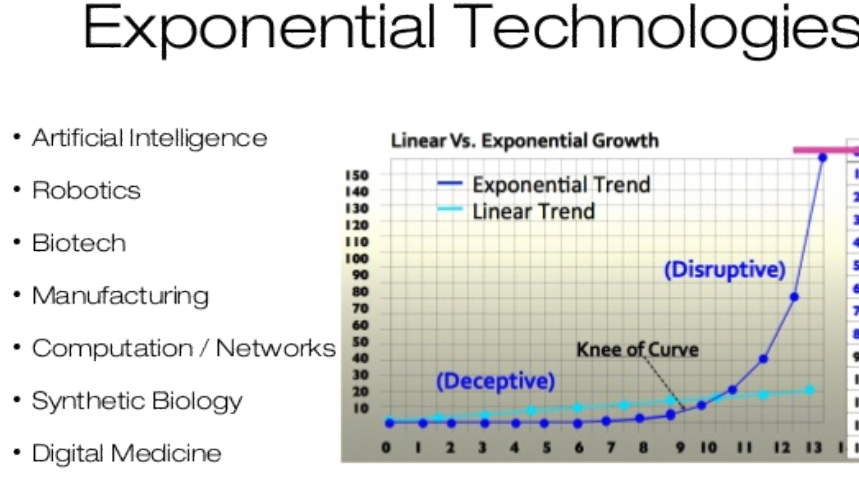
CPUs

SIMD/Data parallel compute using CPU



CPU architectures:

- i386/AMD64,ARM (Raspberry)
- all CPU types: lots of cores; end of Moore's Law, 'Singularity University', problem w. exp. growth?



Scikit-learn problem with SIMD:

- no ML-algo GPU hardware enabled,
- many ML-algo not even multicore aware!

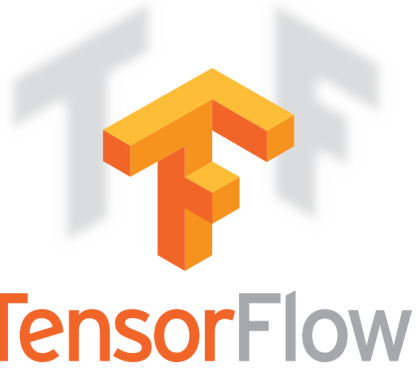
Keras/TensorFlow:

- CPU optimization of TF possible,
- (GPU enabled TF possible, using CUDA + cudNN).

CPUs

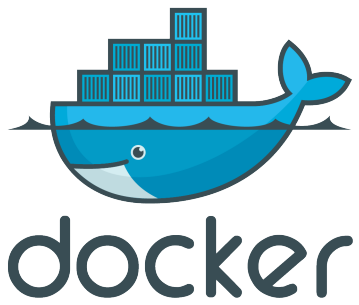
Build Tensorflow from source

- for specific architecture, say ARM,
- or for HPC optimization for all CPU feature



> `lscpu`

```
Architecture:  x86_64
Model name:    Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz
Flags:         fpu vme de pse tsc msr pae mce cx8 apic sep mtrr
               pge dts acpi mmx fxsr fma..sse sse2..sse4_1sse4_2..
```



Using Docker and pulling TF from GIT + a lot of scripting!

```
> git clone https://github.com/tensorflow/tensorflow
> git checkout 1.12
```

(lots of scripting and pain..)

```
> bazel build -copt=-mfma -copt=-msse4.2 tensorflow
```

Howtos:

https://www.tensorflow.org/install/source
https://www.pugetsystems.com/labs/hpc/Build-TensorFlow-CPU-with-MKL-and-Anaconda-Python-3-6-using-a-Docker-Container-1133
https://www.pugetsystems.com/labs/hpc/Build-TensorFlow-GPU-with-CUDA-9-1-MKL-and-Anaconda-Python-3-6-using-a-Docker-Container-1134

CPUs

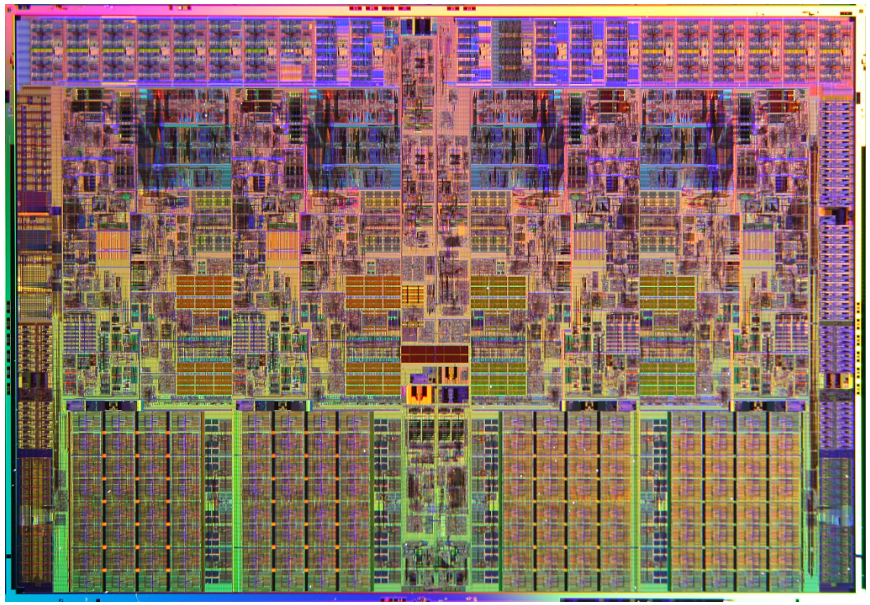
Build Tensorflow from source

Demo...

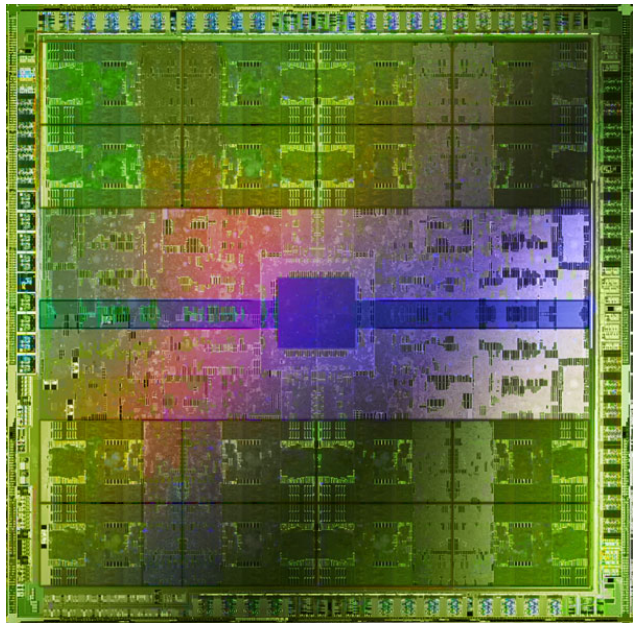
```
    Compiling external/boringssl/src/ssl/ssl_buffer.cc [for host]; 0s local
[2,531 / 6,664] 3 actions running
[2,533 / 6,677] 3 actions running
[2,539 / 6,689] 4 actions running
    Compiling external/boringssl/src/ssl/ssl_lib.cc [for host]; 1s local
[2,541 / 6,692] 4 actions running
    Compiling external/boringssl/src/ssl/ssl_lib.cc [for host]; 1s local
[2,548 / 6,718] 3 actions running
    Compiling external/boringssl/src/ssl/ssl_lib.cc [for host]; 1s local
[2,549 / 6,722] 3 actions running
    Compiling external/boringssl/src/ssl/ssl_lib.cc [for host]; 1s local
[2,550 / 6,723] 3 actions running
[2,552 / 6,733] 3 actions running
    Compiling external/grpc/src/core/tsi/ssl/session_cache/ssl_session_boringssl.cc
[2,555 / 6,749] 3 actions running
[2,559 / 6,766] 4 actions running
    Compiling external/icu/icu4c/source/common/uloc.cpp [for host]; 0s local
    Compiling external/boringssl/src/ssl/tls13_enc.cc [for host]; 0s local
    Compiling external/grpc/src/core/ext/filters/client_channel/client_channel.cc [f
or host]; 0s local
    Compiling external/grpc/src/core/lib/gpr/string.cc [for host]; 0s local
```

- lots of Makefile, Dockerfile and TF build scripting,
- did succeed with TF/CPU, but not TF/GPU,
- not tried with TF 2.X,
- specific builds for Raspberry Pi/ARM also possible.

CPUs vs GPUs



Nehalem CPU
die size: $\sim$700 mm^2
transistors: $\sim 2.3 \cdot 10^9$



Fermi GPU
die size: 520 mm^2
transistors: $\sim 3 \cdot 10^9$

Nvidia architectures
transitor counts:

Pascal	$\sim 15 \cdot 10^9$
Turing	$\sim 19 \cdot 10^9$
Volta	$\sim 21 \cdot 10^9$

NOTE: NVIDIA naming scheme

Tesla P100=Pascal microarch.,
Tesla V100=Volta microarch.,
and Maxwell, Kepler microarc.

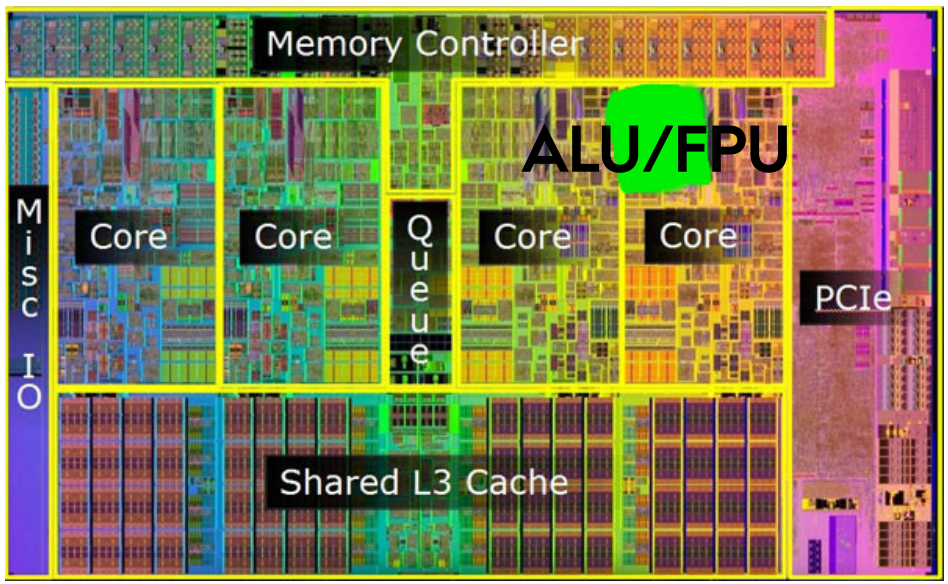
Tesla=HPC product + microarcitecture
GeForce, Quadro=product lines

CPUs vs GPUs

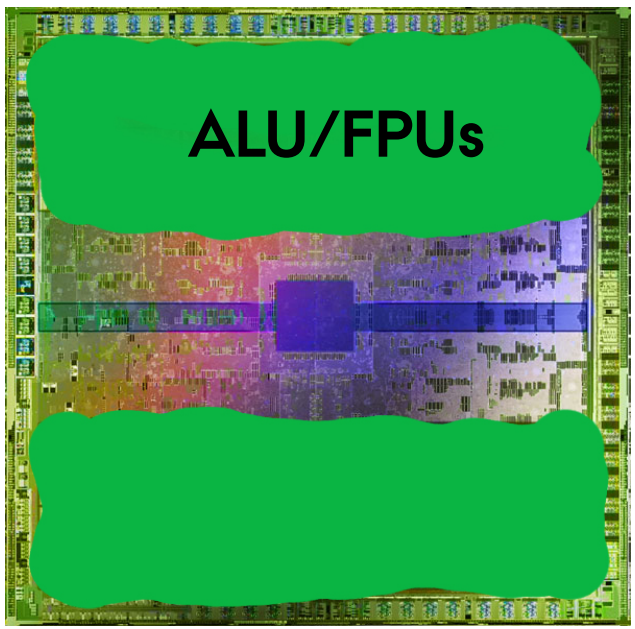
So many transistors, but how many for the ALUs/FPUs?

What makes GPUs such excellent HW for Machine Learning?

ALU: Arithmetic logic unit
FPU: Floating-point unit
Memory Controller: six controllers on GPU.



CPU (type?)
with one ALU/FPU marked



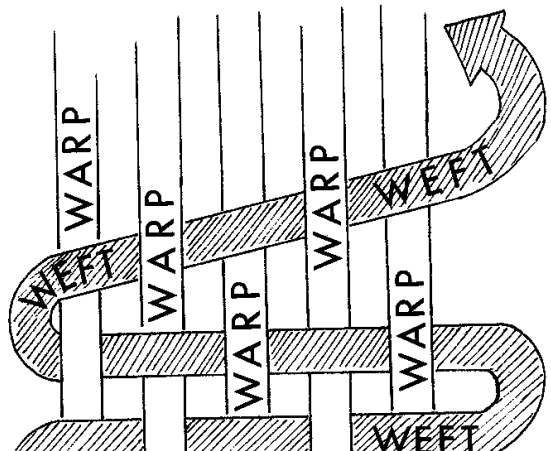
Fermi GPU
ALUs/FPUs all over

GPUs

Fundamental Problems with the GPUs Hardware

GPUs had several *Achilles heels* related to its hardware, many of them addressed in the latest Vola V100 architecture:

- coding problems graphically, now CUDA,
- no STACK, now added,
- no CACHE (or Texture only), now both L_1 and L_2 cache,
- distinct GPU med, now UNIFIED memory,
- SIMT WARP-bunch of 32-threads, now true SIMD,



GPUs

GPU architecture: Core Design, Streaming Multiprocessors

GPUs

Core Design for a SM (Streaming Multiprocessor)

Volta SM design (new gen. GPU):

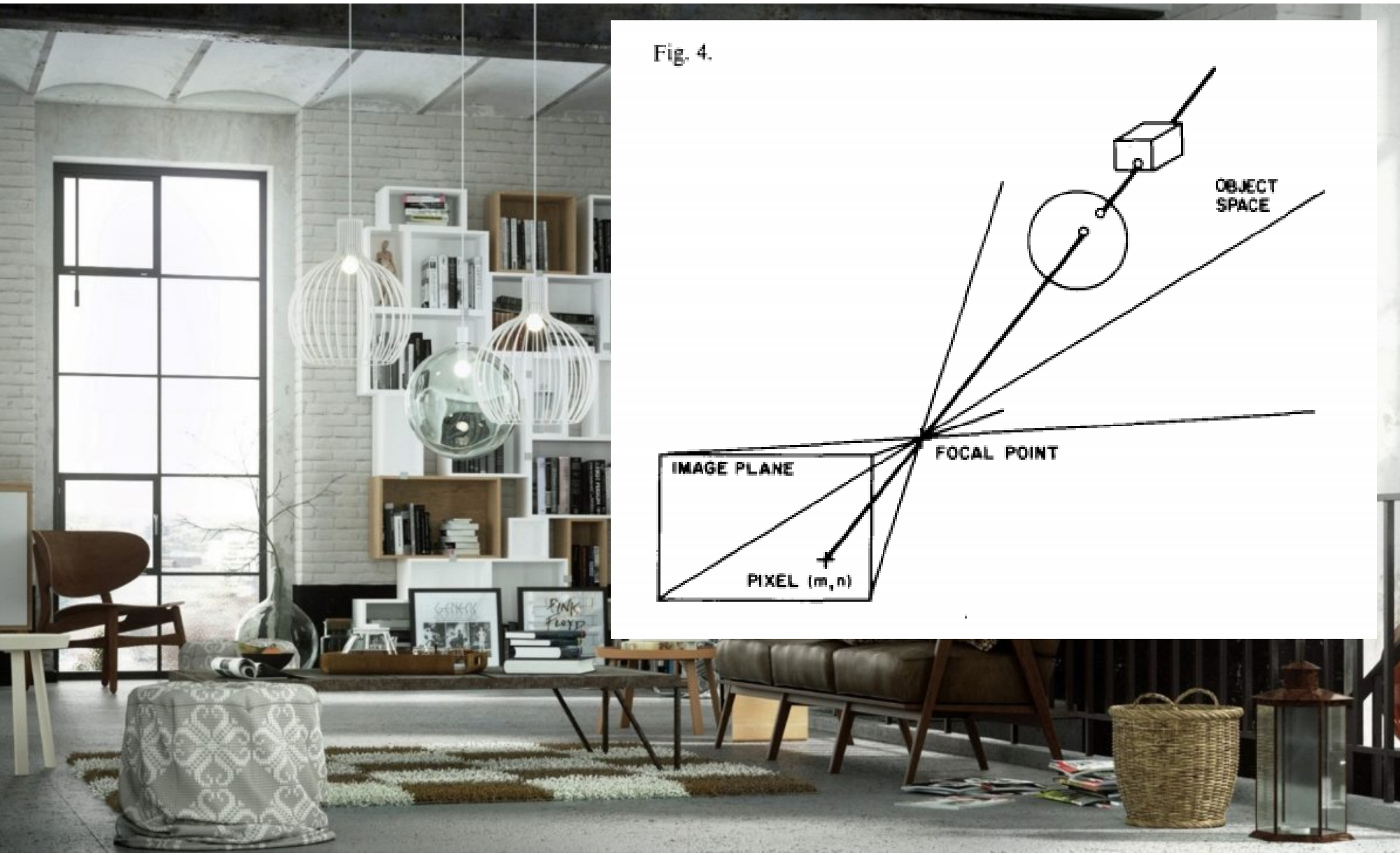
FP64/32:FPUs

INT: ALUs

TENSOR CORES: ?

GPUs

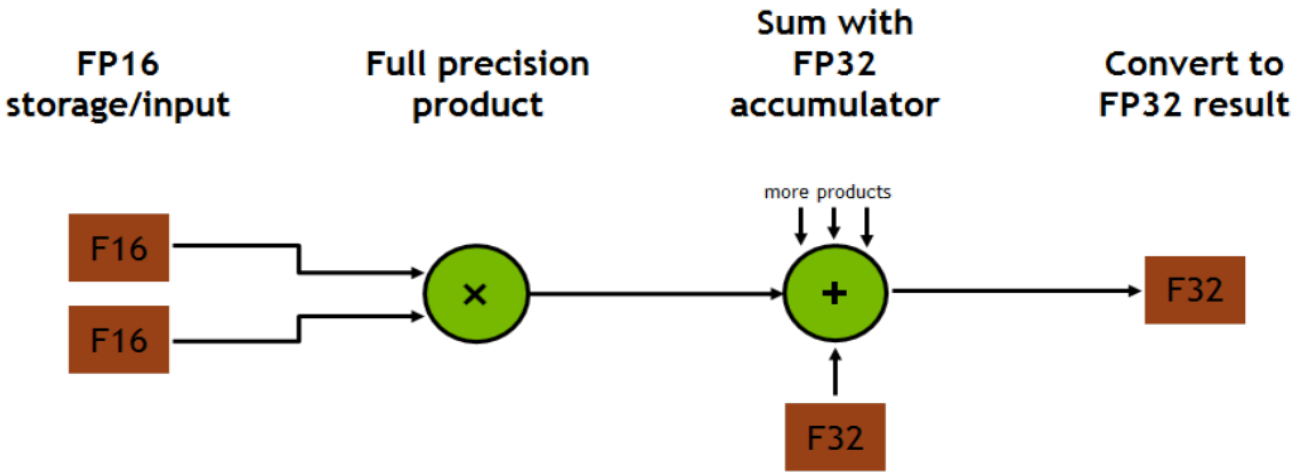
Tensor cores: Raytracing vs Rasterization on GPUs

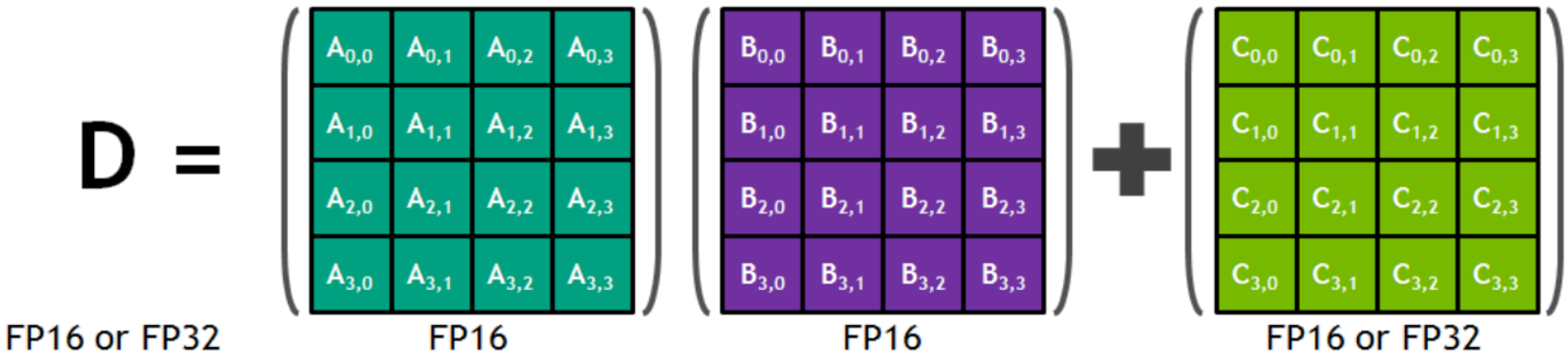


[https://blogs.nvidia.com/blog/2018/03/19/whats-difference-between-ray-tracing-rasterization]

GPUs

Tensor Cores





Tesla V100's Tensor Cores deliver up to 125 Tensor TFLOPS for training and inference applications.

[volta-architecture-whitepaper.pdf]

Engineering Department: matrix mul-and-add cores!

Marketing Department: name 'em Tensor Cores!

HPC Top500

Best High-Performance Computer in 2005

[https://www.top500.org/list/2005/06/]

Search

HOME LISTS STATISTICS RESOURCES ABOUT MEDIA KIT

Home / Lists / June 2005 / List

TOP500 List - June 2005

R_{max} and **R_{peak}** values are in TFlops. For more details about other fields, check the TOP500 description.

R_{peak} values are calculated using the advertised clock rate of the CPU. For the efficiency of the systems you should take into account the Turbo CPU clock rate where it applies.

previous 1 2 3 4 5 next

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/NNSA/LLNL United States	**BlueGene/L** - eServer Blue Gene Solution IBM	65,536	136.8	183.5	716
2	IBM Thomas J. Watson	**BGW** - eServer Blue Gene	40,960	91.3	114.7	448

GPUs

When is the GPU faster than the CPU for NN?

GPU slower for CPU for a three-layer NN + MNIST, why?

- GPU needs a reasonable about of trainable paramters + data to beat the CPU!

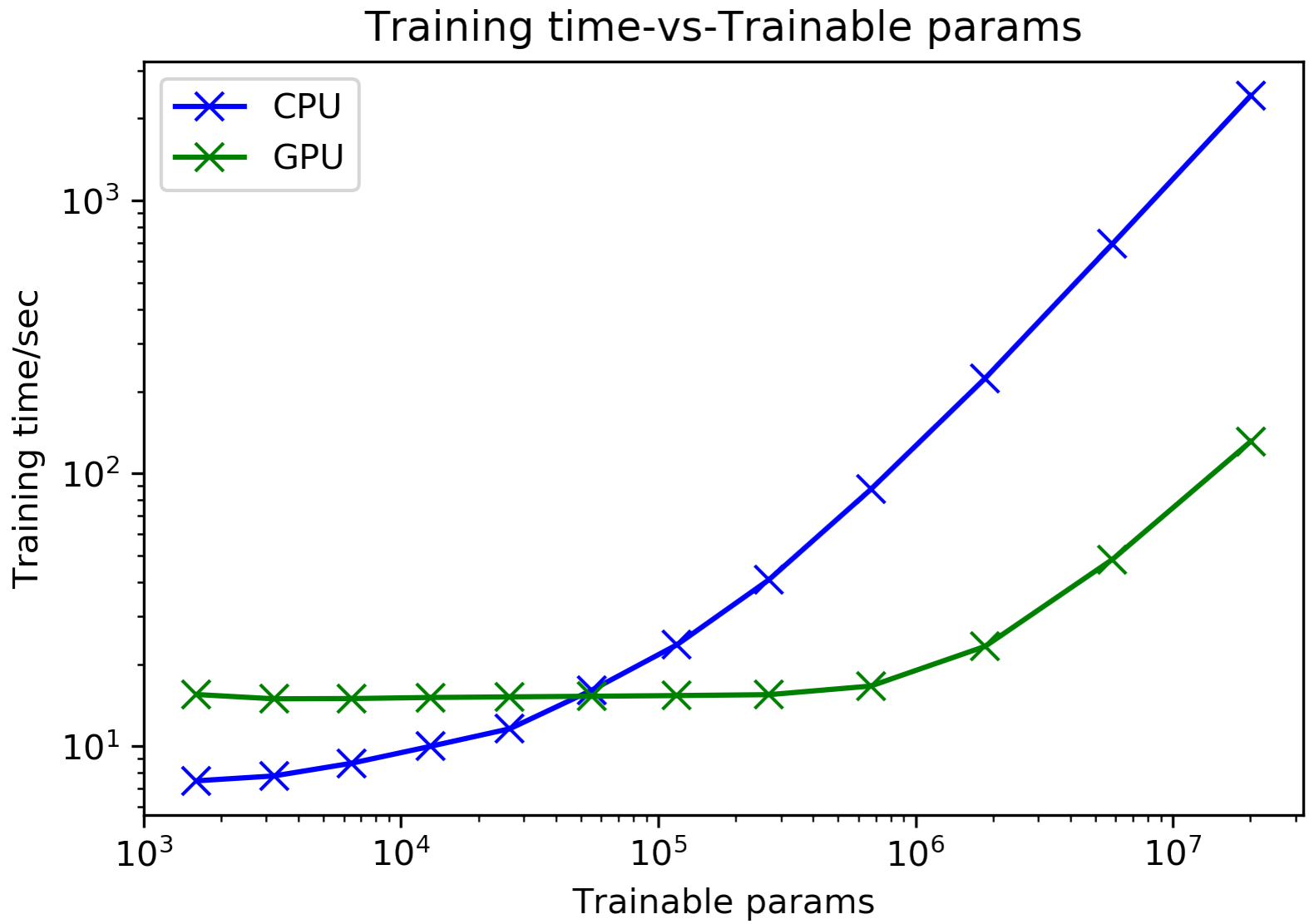
`model.summary()`:

```
#  i=12, n=4096
#  _________________________________________________________________
#  Layer (type)                 Output Shape              Param #
#  =================================================================
#  dense_46 (Dense)             (None, 4096)              3215360
#  _________________________________________________________________
#  dropout_31 (Dropout)         (None, 4096)              0
#  _________________________________________________________________
#  dense_47 (Dense)             (None, 4096)              16781312
#  _________________________________________________________________
#  dropout_32 (Dropout)         (None, 4096)              0
#  _________________________________________________________________
#  dense_48 (Dense)             (None, 10)                40970
#  =================================================================
#  Total params: 20,037,642
#  Trainable params: 20,037,642
#  Non-trainable params: 0
```

GPUs

Actual test on the GPU-server

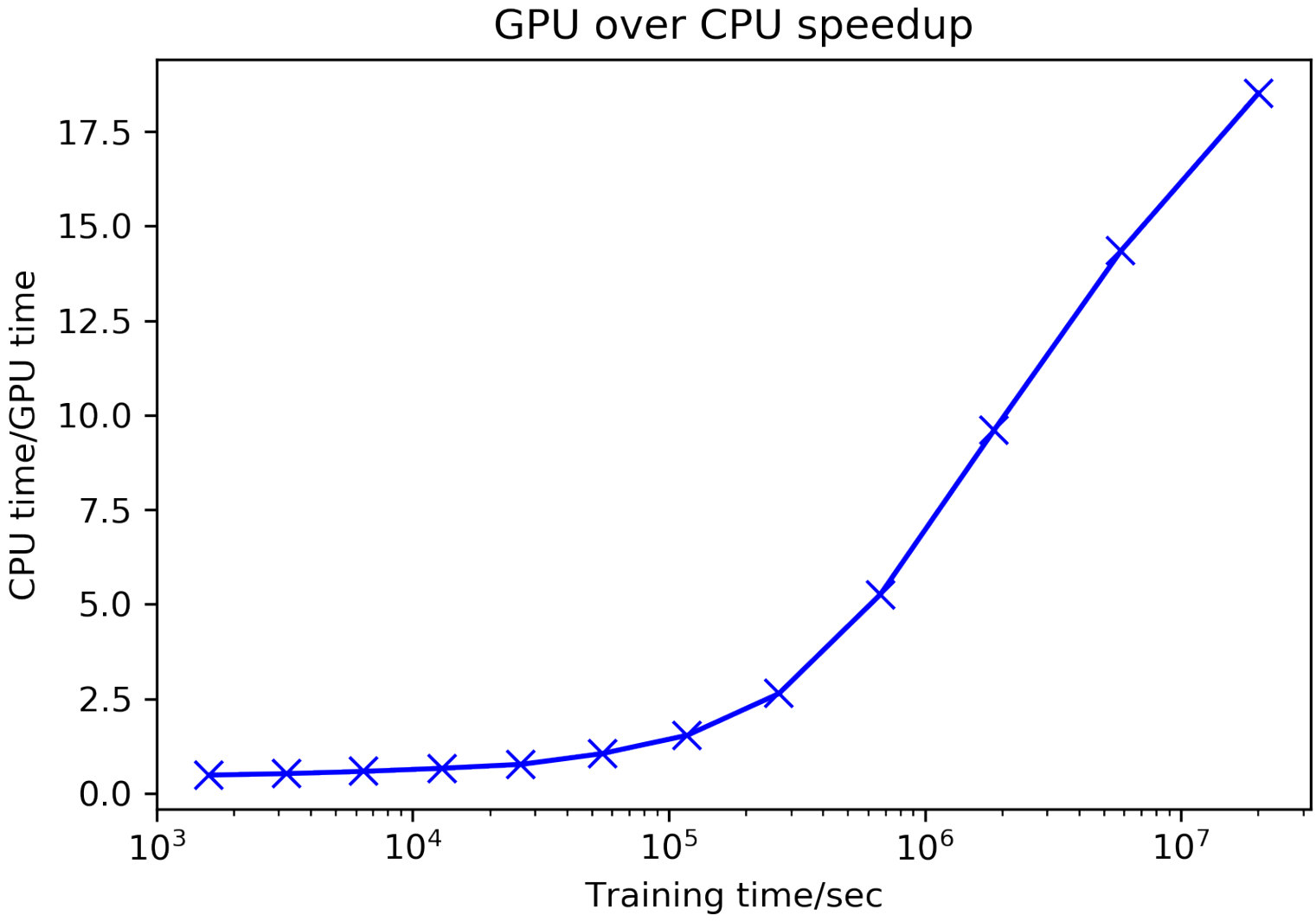
CPU vs GPU on MNIST for a three layer NN with dropout...



GPUs

Actual test on the GPU-server

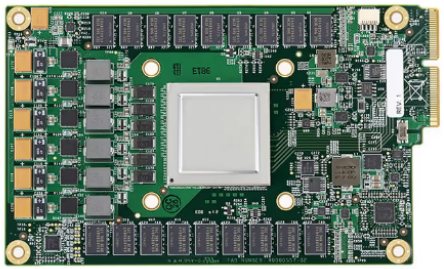
CPU vs GPU on MNIST for a three layer NN with dropout...



TPUs

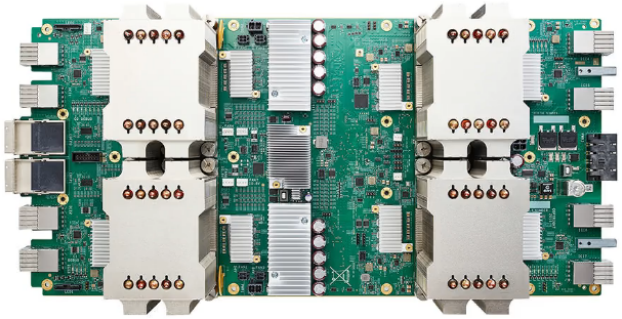
Tensor Processing Units

Custom ASICs by Google



TPU v1

Launched in 2015
Inference only

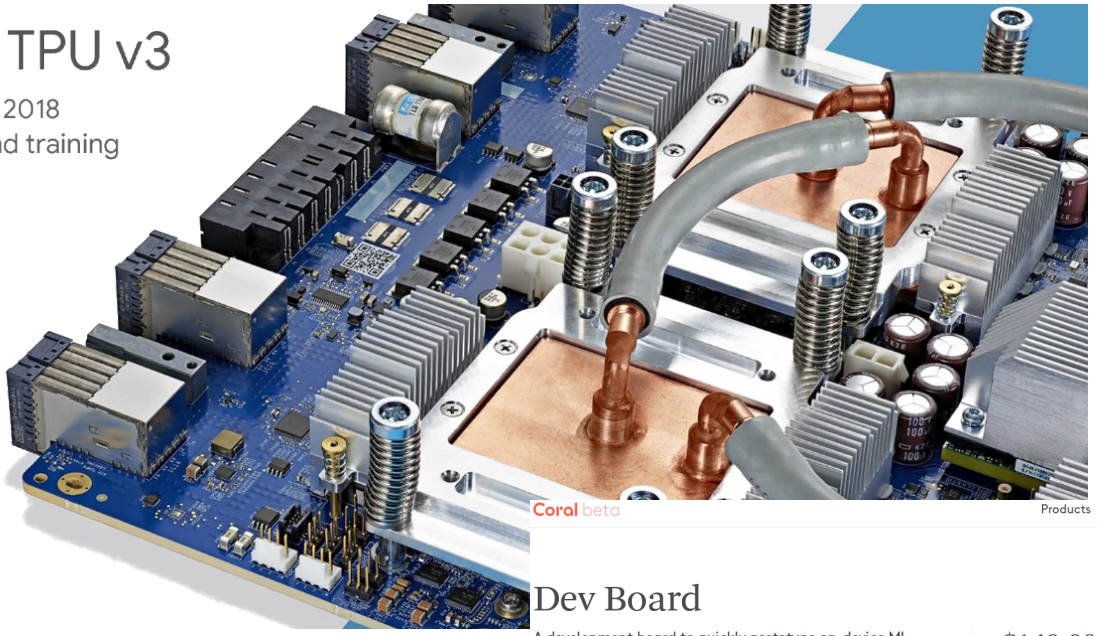


TPU v2

Launched in 2017
Inference and training

Cloud TPU v3

Launched in 2018
Inference and training



Dev board with Google Edge TPU ML accelerator coprocessor

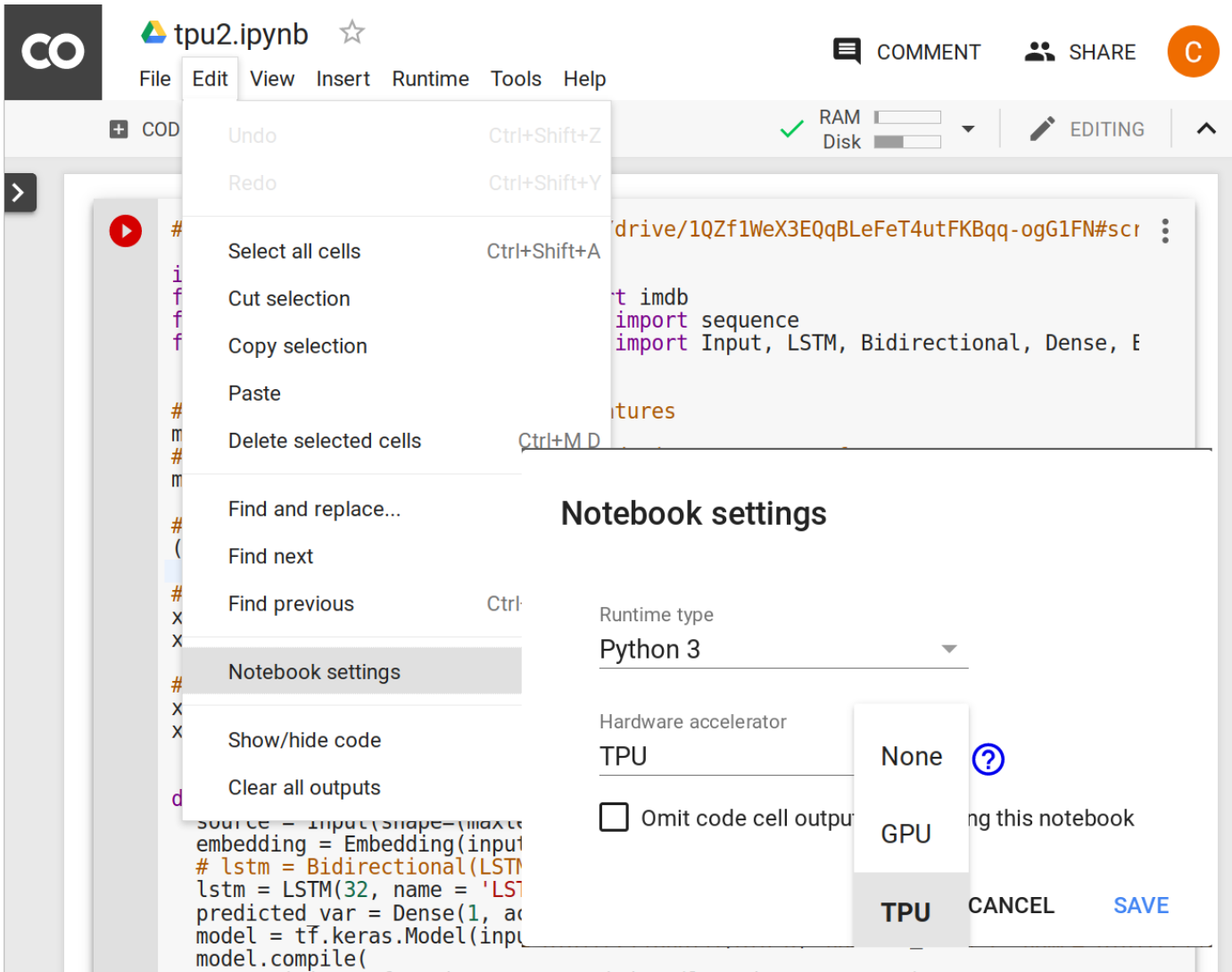
[https://coral.withgoogle.com/products/dev-board/]

TPUs

Access to TPUs

Free Jupyter Notebook environment with access to TPUs:

`https://colab.research.google.com`



TPUs

TPU Cloud

[https://storage.googleapis.com/nexttpu/index.html]

TPUs vs GPUs

Performance, TPUs vs GPUs, who wins?

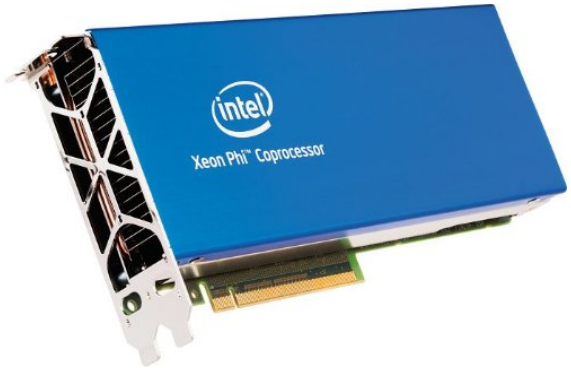
- Huge advantage for TPU performance-per-watt,
- Colab performance:
  inconclusive (TPU part does not work),
- TPU only for inference?

	K80 2012	TPU 2015	P40 2016
Inferences/Sec <10ms latency	1/13 TH	1X	2X
Training TOPS	6 FP32	NA	12 FP32
Inference TOPS	6 FP32	90 INT8	48 INT8
On-chip Memory	16MB	24 MB	11 MB
Power	300W	75W	250W
Bandwidth	320 GB/S	34 GB/S	350 GB/S

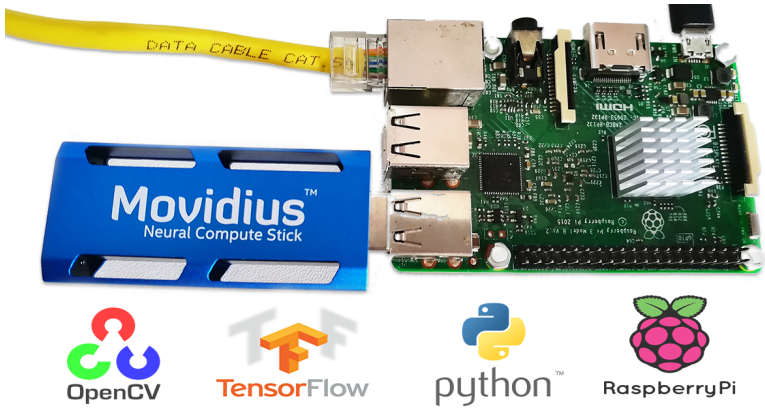
[https://www.extremetech.com/computing/247403-nvidia-claims-pascal-gpus-challenge-googles-tensorflow-tpu-updated-benchmarks]

Exotic Hardware

- Intel Phi multicore CPU, 64 i386 cores:



- Raspberry Pls + Intel Movideus stick



- FPGAs

Extra Slide(s)

ASICs vs GPUs

Bitcoin-mining

AntMiner S7: based on ASICs.

Some specs:

- hash rate: 4.8 THash/s
- chips per unit: 162 x BM1385
- power consumption: **1210 W**
- power efficiency: 0.25 W/GHash
- price: $479.95 ~ 2880,- DKK
- production: **0.16 bitcoin/month**