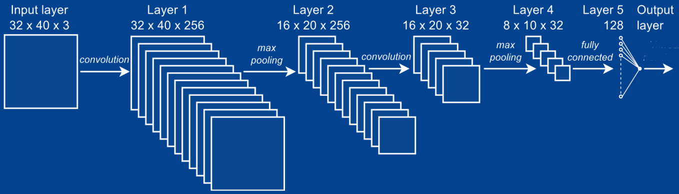
AARHUS UNIVERSITY
SCHOOL OF ENGINEERING

LESSON 09: Deep Learning + CNN's + Hardware

CARSTEN EIE FRIGAARD

SPRING 2021



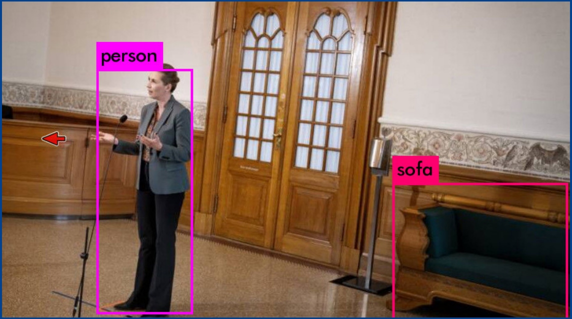
"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E." — Mitchell (1997).

L09: DL + CNN's + HW, Agenda

- Spørge-minutter..
- Admin:
  - O3 opgavesæt afsluttet, Deadline 14/4, 18:00:
    - Opgave fra L07: `capacity_under_overfitting.ipynb`
    - Opgave fra L07: `generalization_error.ipynb`
    - Opgave fra L08: `regulizers.ipynb`
    - Opgave fra L08: `gridseach.ipynb`
    - **'Cheat'-review** af O3: før aflevering, se beskrivelse i O1+3 'cheat'-review. Resume skal denne gang inkluderes i O3 afleveringen.
  - Ændringer i "Almindelige formelle journal krav":
    - NY: Ingen direkte genbrug af overskrifter ...
    - NY: Skriv kun på eet sprog igennem hele journalen..

- Convolutional Neural Networks (CNN's),
  - and Deep-learning (DL).
- Hardware for Machine Learning,
  - (CPUs), GPUs, TPUs, Exotic Hardware.

CONVOLUTIONAL NEURAL NETWORKS



Deep-learning (DL)

Artificial Intelligence:

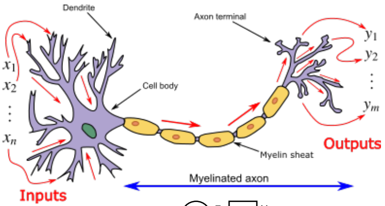
Mimicking the intelligence or behavioural pattern of humans or any other living entity.

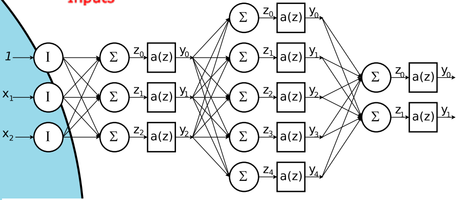
Machine Learning:

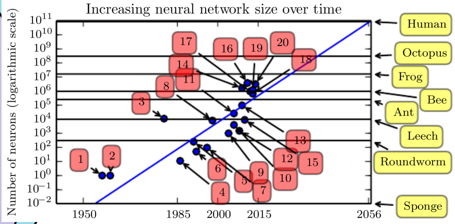
A technique by which a computer can "learn" from data, without using a complex set of different rules. This approach is mainly based on training a model from datasets.

Deep Learning:

A technique to perform machine learning inspired by our brain's own network of neurons.





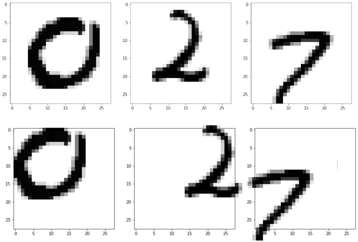


Definition of DL=??

[Deep Learning, Ian Goodfellow et. al., The MIT Press, 2016], [Prof. Loc Vu-Quoc, commons.wikimedia.org/w/index.php?curid=72816083], [Tukijaaliwa, commons.wikimedia.org/w/index.php?curid=9013135]

Preprocessing/Feature-extraction + Machine Learning

Can your ML model handle simple image translation (rotation and scaling)?



..no problem for Your visual cortex, right?

Introducing the Convolutional Neural Networks: trade of

- preprocessing/feature-extraction + classification

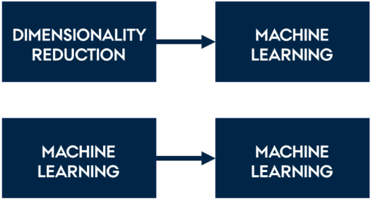
with:

- feature learning (CNN kernels) + classification (fully connected NN)

Convolutional Neural Networks

Feature Extraction vs. Feature Learning

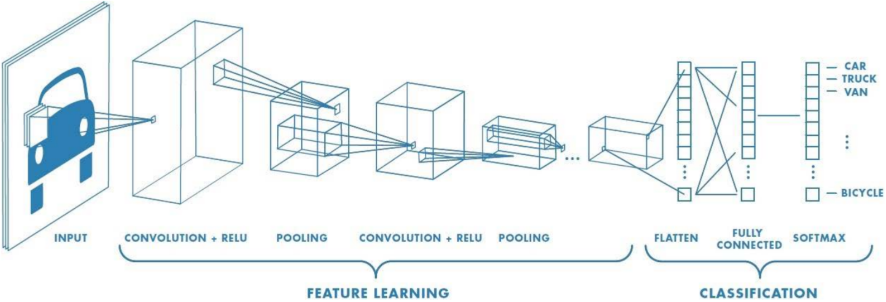
- Smart filtering:
  the distriction bewteen dimensionality reduction and machine learning blurs..



- Fundamental problem of filtering:
  What is noise, what is signal?

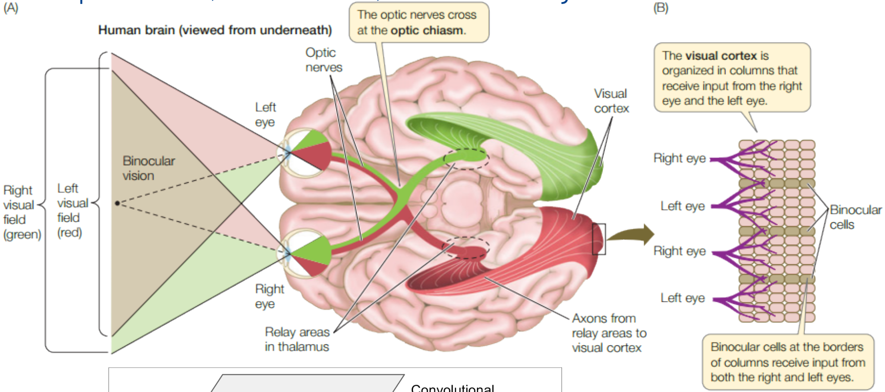
Convolutional Neural Networks

Feature Extraction vs. Feature Learning



Convolutional Neural Networks

Human Eye and Brain Image Processing: Receptive Fields, Visual Cortex, and Neuro Cognition



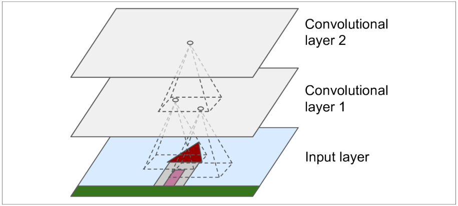


Figure 14-2. CNN layers with rectangular local receptive fields

Feature Learning + Machine Learning

CNN principle

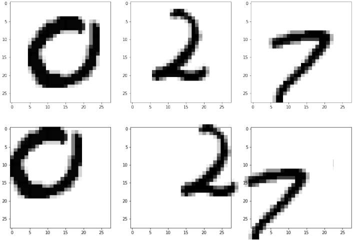
Translation, rotation, and scaling invariant

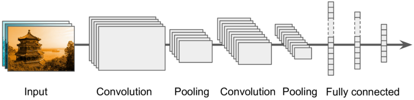
- automatic image feature extraction

that is feature learning via

- 'convolutional', 'pooling' CNN layers,

then a final fully connected NN





..no problem for Your CNN ML model, right?

Feature Learning + Machine Learning

CNN principle: Convolutional Layer (via CNN kernels)

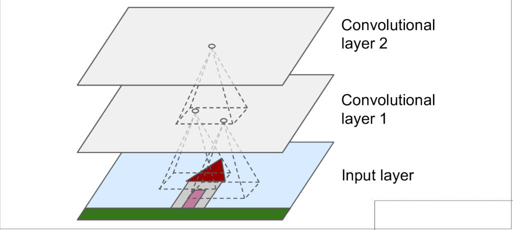


Figure 14-2. CNN layers with rectangular local receptive fields

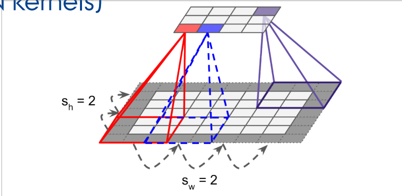


Figure 14-4. Reducing dimensionality using a stride of 2

Feature map 1 => with kernel 1

Feature map 2 => with kernel 2

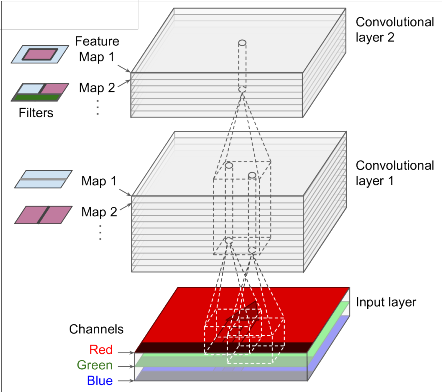


Figure 14-6. Convolution layers with multiple feature maps, and images with three color

CNN Kernels

2D Convolution with a Kernel: Principle

0	0	0	0	0	0	0
0	60	113	56	139	85	0
0	73	121	54	84	128	0
0	131	99	70	129	127	0
0	80	57	115	69	134	0
0	104	126	123	95	130	0
0	0	0	0	0	0	0

Kernel

0	-1	0
-1	5	-1
0	-1	0

114				

$$5*60 + (-1)*113 + (-1)*73 = 114$$

CNN Kernels

2D Convolution with a Kernel: Different Kernels

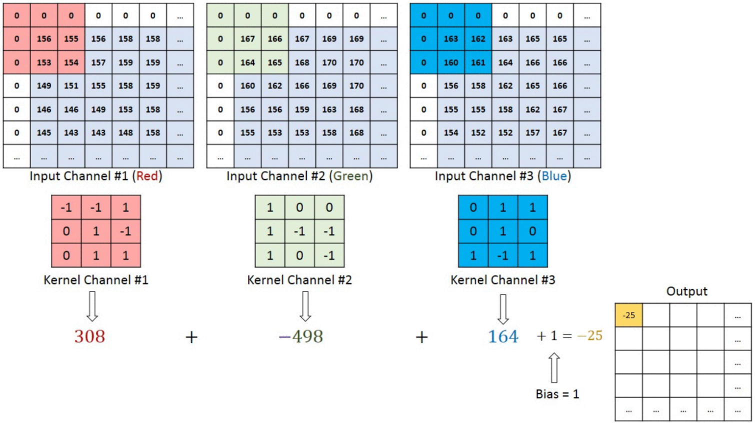
-0.2	0.0	0.5
1.0	0.3	-0.6
0.0	0.0	0.8

0.0	0.0	0.0
0.8	-0.5	0.8
0.0	-0.2	0.0

0.4	0.2	-0.2
-0.8	0.0	0.8
0.0	-0.5	0.2

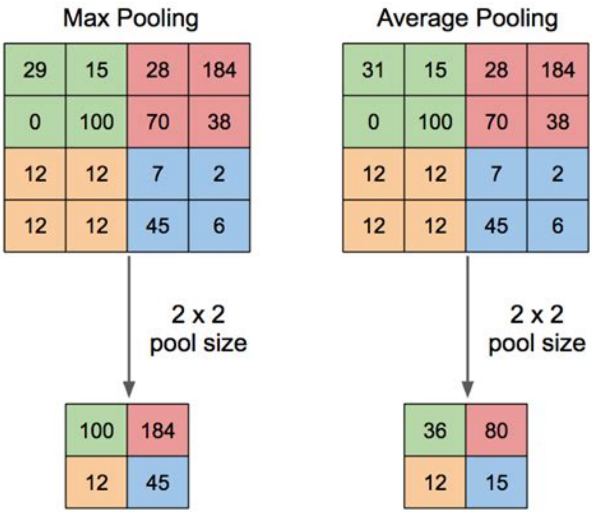
CNN Kernels in 3D

3D Convolution with a Kernel: Principle



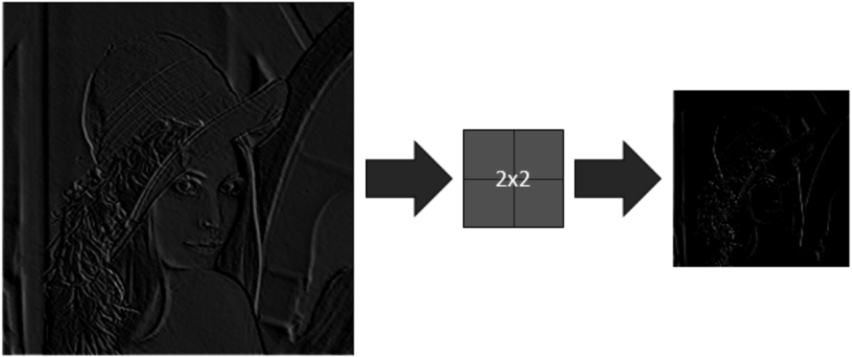
CNN Pooling

The Principle



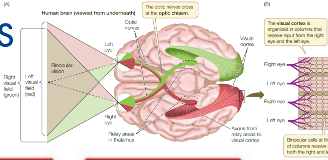
CNN Pooling

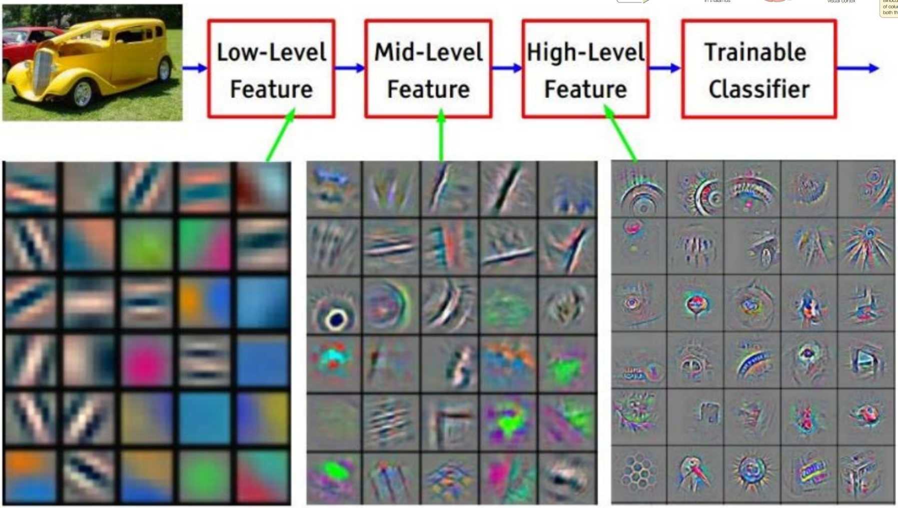
On real data (effectivly subsampling)



Convolutional Neural Networks

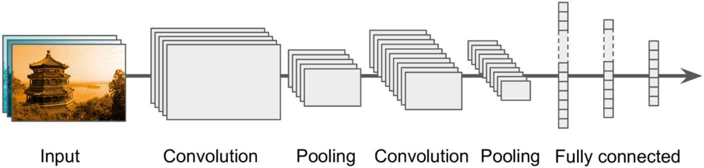
Low-, Mid-, and High-Level Feature Extraction

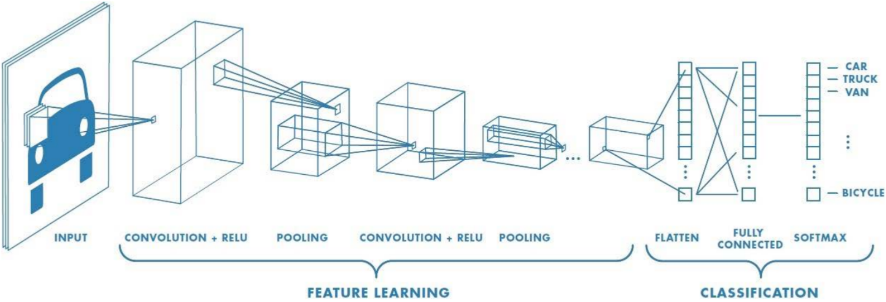




Convolutional Neural Networks

Stacking It All Up..





CNN's in Practice

Varroa mite detector—with a CNN somewhere in the pipeline

A computer vision system to monitor the infestation level of *Varroa destructor* in a honeybee colony

Kim Bjerge[a,*], Carsten Eie Frigaard[a], Peter Hogh Mikkelsen[a], Thomas Holm Nielsen[a], Michael Misbih[a], Per Kryger[b]

[a]School of Engineering, Aarhus University, Finlandsgade 22, 8200 Aarhus N, Denmark
[b]Department of Agroecology - Entomology and Plant Pathology, Aarhus University, Flakkebjerg, Forsøgsvej 1, 4200 Slagelse, Denmark

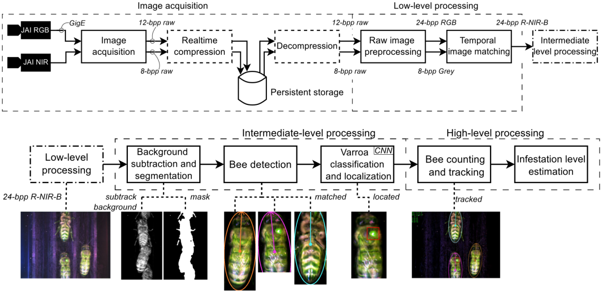
Abstract

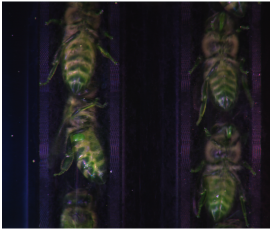
This paper presents a portable computer vision system, that is able to monitor the infestation level of the *Varroa destructor* mite in a beehive by recording a video sequence of live honeybees *Apis mellifera* for 5-20 minutes. A video monitoring unit with multispectral illumination and camera was designed to be placed in front of the beehive, where bees from a selected frame were shaken off. Subsequently, a computer vision algorithm (denoted as the *Infestation Level Estimator*) based on deep learning analysis of the video stream counted the number of honeybees and found the position of identified varroa mites. In this paper, the design and the algorithm that were used to determine the number of bees and mites are presented. Based on a video sequence with 1,775 bees and 98 visual mites, the algorithm measured the infestation level to 5.80% compared to a ground truth of 5.52% The algorithm had a high F_1-score accuracy for counting bees (0.97), while the F_1-score for detecting varroa mites was lower (0.91). The latter was due to mispredictions, which can be resolved by improving both the trained varroa classifier and the mechanical setup. Overall, the proposed computer vision system and algorithm showed a promising results in nondestructive and automatic monitoring of infestation levels in honeybee colonies and should be considered as an alternative to traditional methods, which require the killing of bees.

Keywords: Computer Vision, Deep Learning, *Varroa destructor*, *Apis mellifera*, Multi-spectral Illumination
2018 MSC: 00-01, 99-00

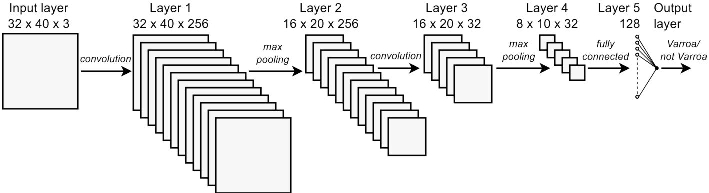
1. Introduction

Image processing pipeline





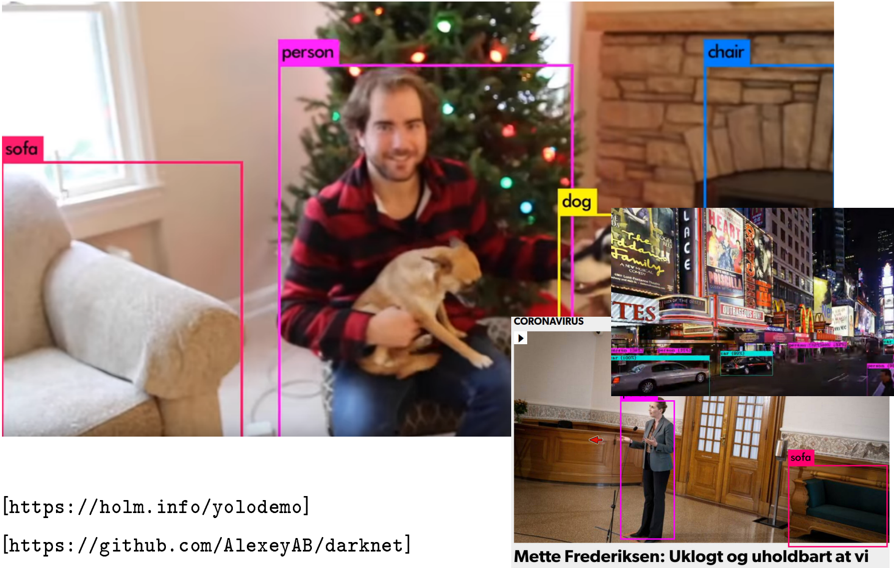
CNN architecture



CNN's in Practice

YOLOv2+3+4 (You Only Look Once)

Real-time object detection, demo..



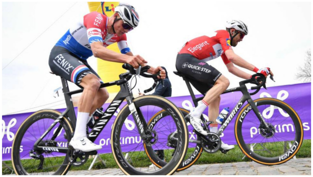
[https://holm.info/yolodemo]

[https://github.com/AlexeyAB/darknet]

CNN's in Practice

YOLOv2+3+4 (You Only Look Once)

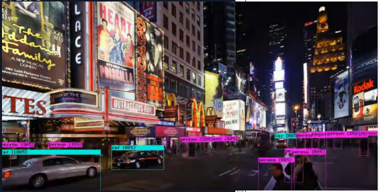
```
o obj/art.o obj/tag.o obj/cifar.o obj/go.o obj/rnn.o obj/segmenter.o obj/regressor.o obj/classifier.o obj/coco.o
 obj/yolo.o obj/detector.o obj/nightmare.o obj/instance-segmenter.o obj/darknet.o libdarknet.a -o darknet -lm -p
thread  -L/opt/opencv/opencv4/lib -lopencv_core -lopencv_imgproc -lopencv_imgcodecs -lopencv_videoio -lopencv_hi
ghgui  -ltiff -lstdc++ libdarknet.a
layer     filters    size              input                output
    0 conv     32  3 x 3 / 1   256 x 256 x   3   ->   256 x 256 x  32  0.113
    1 max          2 x 2 / 2   256 x 256 x  32   ->   128 x 128 x  32
    2 conv     64  3 x 3 / 1   128 x 128 x  32   ->   128 x 128 x  64  0.604
    3 max          2 x 2 / 2   128 x 128 x  64   ->    64 x  64 x  64
    4 conv    128  3 x 3 / 1    64 x  64 x  64   ->    64 x  64 x 128  0.604
    5 conv     64  1 x 1 / 1    64 x  64 x 128   ->    64 x  64 x  64  0.067
    6 conv    128  3 x 3 / 1    64 x  64 x  64   ->    64 x  64 x 128  0.604
    7 max          2 x 2 / 2    64 x  64 x 128   ->    32 x  32 x 128
    8 conv    256  3 x 3 / 1    32 x  32 x 128   ->    32 x  32 x 256  0.604
    9 conv    128  1 x 1 / 1    32 x  32 x 256   ->    32 x  32 x 128  0.067
   10 conv    256  3 x 3 / 1    32 x  32 x 128   ->    32 x  32 x 256  0.604
   11 max          2 x 2 / 2    32 x  32 x 256   ->    16 x  16 x 256
   12 conv    512  3 x 3 / 1    16 x  16 x 256   ->    16 x  16 x 512  0.604
   13 conv    256  1 x 1 / 1    16 x  16 x 512   ->    16 x  16 x 256  0.067
   14 conv    512  3 x 3 / 1    16 x  16 x 256   ->    16 x  16 x 512  0.604
   15 conv    256  1 x 1 / 1    16 x  16 x 512   ->    16 x  16 x 256  0.067 BFLOPs
   16 conv    512  3 x 3 / 1    16 x  16 x 256   ->    16 x  16 x 512  0.604 BFLOPs
   17 max          2 x 2 / 2    16 x  16 x 512   ->     8 x   8 x 512
   18 conv   1024  3 x 3 / 1     8 x   8 x 512   ->     8 x   8 x1024  0.604
   19 conv    512  1 x 1 / 1     8 x   8 x1024   ->     8 x   8 x 512  0.067
   20 conv   1024  3 x 3 / 1     8 x   8 x 512   ->     8 x   8 x1024  0.604
   21 conv    512  1 x 1 / 1     8 x   8 x1024   ->     8 x   8 x 512  0.067
   22 conv   1024  3 x 3 / 1     8 x   8 x 512   ->     8 x   8 x1024  0.604
   23 conv   1000  1 x 1 / 1     8 x   8 x1024   ->     8 x   8 x1000  0.131
   24 avg                        8 x   8 x1000   ->  1000
   25 softmax                                        1000
Loading weights from darknet19.weights...Done!
test.jpg: Predicted in 1.965434 seconds.
64.68%: mountain bike
16.00%: web site
12.87%: bicycle-built-for-two
 1.05%: crash helmet
 0.86%: alp
cef@leno:~/texitmal/darknet$
```


De danske cykelryttere har aldrig været bedre, siger cykelekspert Brian Nygaard efter Kasper Asgreens sejr i Flandern Rundt.

Kasper Asgreen slog Mathieu van der Poel i årets Flandern Rundt og sikrede den sjette danske sejr på årets World Tour. (Foto: DAVID Stockman © Scanpix)

AF MATHIAS HAUGE JACOBSEN

[?] FORKLAR ORD [+A] STØRRE TEKST [o] LÆS OP



CNN's in Practice

The LeNET-5 Architecture

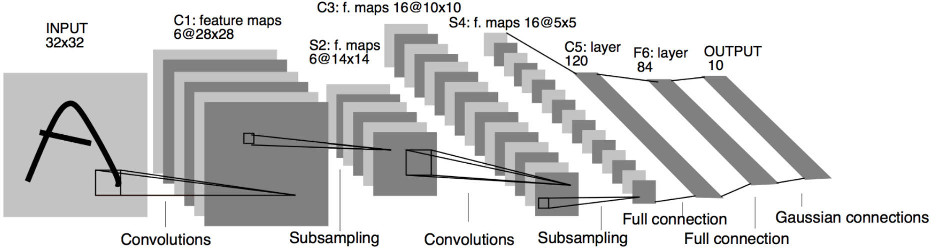


Table 14-1. LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

Other famours CNN-architectures:

- AlexNet,
- GoogLeNet (inception),
- ResNet (152 layers, skip-connections),
- VGGNet,
- Inception-v4 (GoogLeNet + ResNet),
- ...

CNN's in Practice

A LeNET-5 'Like' Architecture

Table 14-1. LeNet-5 architecture

Layer	Type	Maps	Size	Kernel size	Stride	Activation
Out	Fully Connected	–	10	–	–	RBF
F6	Fully Connected	–	84	–	–	tanh
C5	Convolution	120	1×1	5×5	1	tanh
S4	Avg Pooling	16	5×5	2×2	2	tanh
C3	Convolution	16	10×10	5×5	1	tanh
S2	Avg Pooling	6	14×14	2×2	2	tanh
C1	Convolution	6	28×28	5×5	1	tanh
In	Input	1	32×32	–	–	–

In [9]:

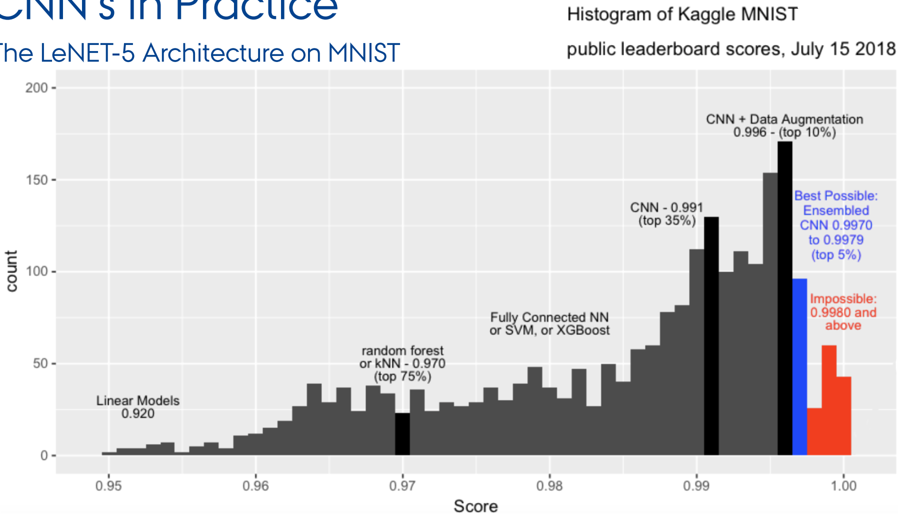
```python
import keras
from keras import layers

model = keras.Sequential()

model.add(layers.Conv2D(filters=6, kernel_size=(3, 3),
                        activation='relu', input_shape=(32,32,1)))
model.add(layers.AveragePooling2D())

model.add(layers.Conv2D(filters=16, kernel_size=(3, 3), activation='relu'))
model.add(layers.AveragePooling2D())

model.add(layers.Flatten())

model.add(layers.Dense(units=120, activation='relu'))
model.add(layers.Dense(units=84,  activation='relu'))
model.add(layers.Dense(units=10,  activation='softmax'))

model.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_5 (Conv2D)            (None, 30, 30, 6)         60
_________________________________________________________________
average_pooling2d_5 (Average (None, 15, 15, 6)         0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 13, 13, 16)        880
_________________________________________________________________
average_pooling2d_6 (Average (None, 6, 6, 16)          0
_________________________________________________________________
flatten_3 (Flatten)          (None, 576)               0
_________________________________________________________________
dense_7 (Dense)              (None, 120)               69240
_________________________________________________________________
dense_8 (Dense)              (None, 84)                10164
_________________________________________________________________
dense_9 (Dense)              (None, 10)                850
=================================================================
Total params: 81,194
Trainable params: 81,194
Non-trainable params: 0
```

[https://colab.research.google.com/drive/1CVm50PGE4vhtB5I_a_yc4h5F-itK0VL9]

CNN's in Practice

The LeNET-5 Architecture on MNIST



- using pre-traindeded models => Transfered Learning,
- object detection,
- semantic segmentation,
- time series => RNN's, ...

Qd MNIST Search Quest II

Fra tidligere semester:

Highest score = 0.97369,

$\longrightarrow$ ITMALGpr28:

```
model=KNeighborsClassifier(
  algorithm='ball_tree',
  n_neighbors=3,
  p=4,
  weights='distance')
```

ITMAL MNIST Search Quest II Results (1. præmie: kage til gruppen)

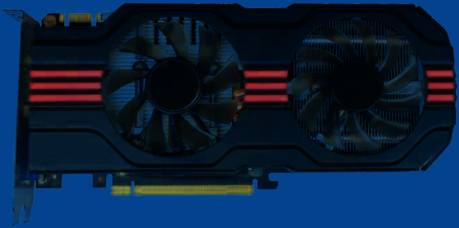
Send message...

```
12/10 11:13> "ITMALGrp17: best: dat=mnist, score=0.97206,
model=KNeighborsClassifier(leaf_size=40,p=3)"
08/10 22:56> "ITMALGrp09: dat=mnist, score=0.96973,
model=KNeighborsClassifier(leaf_size=10,n_neighbors=3,weights='distance')"
08/10 22:04> "ITMALGrp09: dat=mnist, score=0.93550,
model=KNeighborsClassifier(leaf_size=10,n_neighbors=3,weights='distance')"
08/10 10:02> "08/10 10:01 Is from 'ITMALGrp28'"
08/10 10:01> "best: dat=mnist, score=0.97369,
model=KNeighborsClassifier(algorithm='ball_tree',n_neighbors=3,p=4,weights='distance')"
08/10 08:53> "ITMALGrp17: dat=mnist, score=0.47090, model=KNeighborsClassifier(algorithm='auto',
n_neighbors=3, weights='distance', p=2, metric='minkowski', metric_params=None, n_jobs=-1),
score=0.97206"
07/10 21:43> "ITMALGrp25: KNeighborsClassifier(algorithm='ball_tree', leaf_size=30,
metric='minkowski', metric_params=None, n_jobs=None, n_neighbors=5, p=2, weights='uniform'),
score=0.9677142857142857"
07/10 18:53> "ITMALGrp21: dat=mnist, score=0.96878,
model=KNeighborsClassifier(algorithm='ball_tree',n_neighbors=3)"
06/10 22:54> "ITMALGrp08: dat=mnist, score=0.96973,
model=KNeighborsClassifier(algorithm='ball_tree',leaf_size=10,n_neighbors=3,weights='distance')"
29/09 13:40> "ITMALGrp05: dat=mnist , score=0.97257,
model=KNeighborsClassifier(n_neighbors=3,p=2,weights='distance')"
24/09 13:55> "ITMALGrp02: dat=mnist, score=0.96810, model=RandomForestClassifier(n_estimators=1000)"
```

Training time?

```
12/10 11:13> "ITMALGrp17: score=0.97206, model=KNeighborsClassifier...
08/10 22:56> "ITMALGrp09: score=0.96973, model=KNeighborsClassifier...
08/10 22:04> "ITMALGrp09: score=0.93550, model=KNeighborsClassifier...
08/10 10:02> "08/10 10:01 Is from 'ITMALGrp28'"
08/10 10:01> " score=0.97369, model=KNeighborsClassifier...
08/10 08:53> "ITMALGrp17: score=0.47090, model=KNeighborsClassifier...
07/10 21:43> "ITMALGrp25: score=0.9677, model=KNeighborsClassifier...
07/10 18:53> "ITMALGrp21: score=0.96878, model=KNeighborsClassifier...
06/10 22:54> "ITMALGrp08: score=0.96973, model=KNeighborsClassifier...
29/09 13:40> "ITMALGrp05: score=0.97257, model=KNeighborsClassifier...
24/09 13:55> "ITMALGrp02: score=0.96810, model=RandomForestClassifier...
```

HARDWARE FOR MACHINE LEARNING



Hardware for Machine Learning

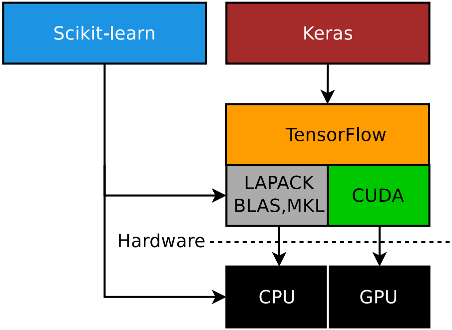
Methods and Terminology (SKIP most except μ, η)

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.

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,

CPUs

Build Tensorflow from source (SKIP)

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

TensorFlow

```
> 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..
```

docker

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

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

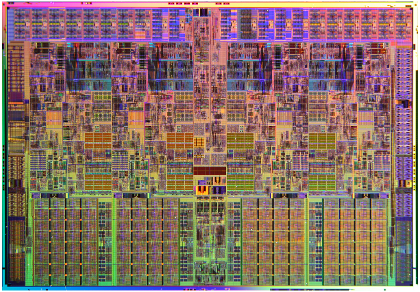
(lots of scripting and pain..)

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

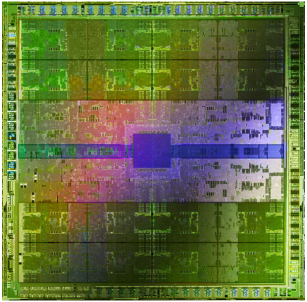
Howtos:

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 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: $\sim 3 \cdot 10^9$

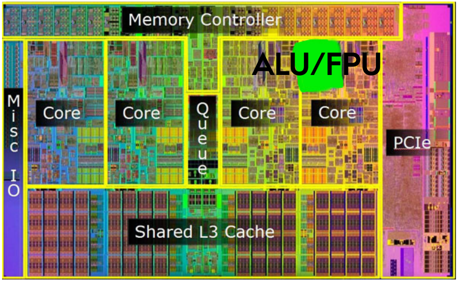
Nvidia arch. transistor counts	
Pascal	$\sim 15 \cdot 10^9$
Turing	$\sim 19 \cdot 10^9$
Volta	$\sim 21 \cdot 10^9$
Ampere	$\sim 28 \cdot 10^9$ (GPU 3090, 8nm)

CPUs vs GPUs

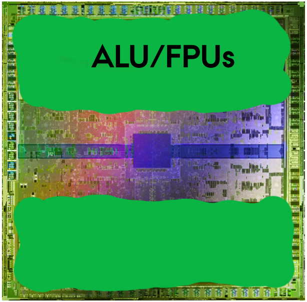
So many transistors, but how many for the ALUs/FPUs?

What makes GPUs such excellent HW for Machine Learning?

ALU: Arithmetic logic unit
FPU: Floating-point unit
Memory Controller: six controllers on GPU,
CPU: lots of speculative execution; waste of transistors.



CPU (type?)
with one ALU/FPU marked



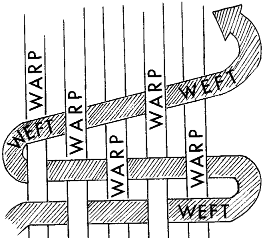
Fermi GPU
ALUs/FPUs all over

GPUs

Fundamental Problems with the GPUs Hardware (SKIP most, except WARP)

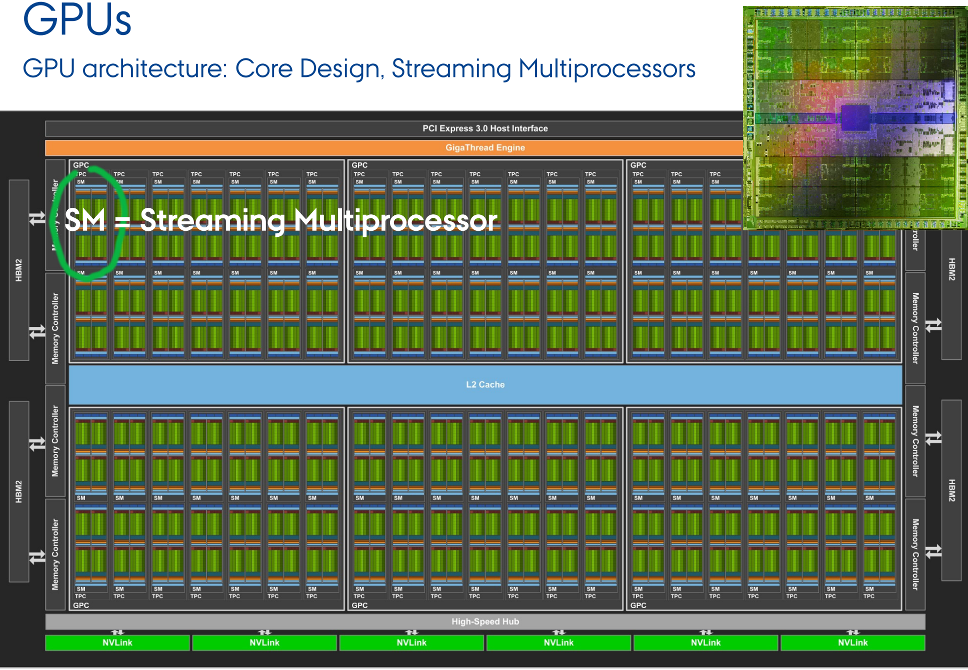
GPUs had several *Achilles heels* related to its hardware, many of them addressed in the latest Vola V100 architecture:

- coding problems graphically, now CUDA,
- no STACK, now added,
- no CACHE (or Texture only), now both L_1 and L_2 cache,
- distinct GPU memory, 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: ?

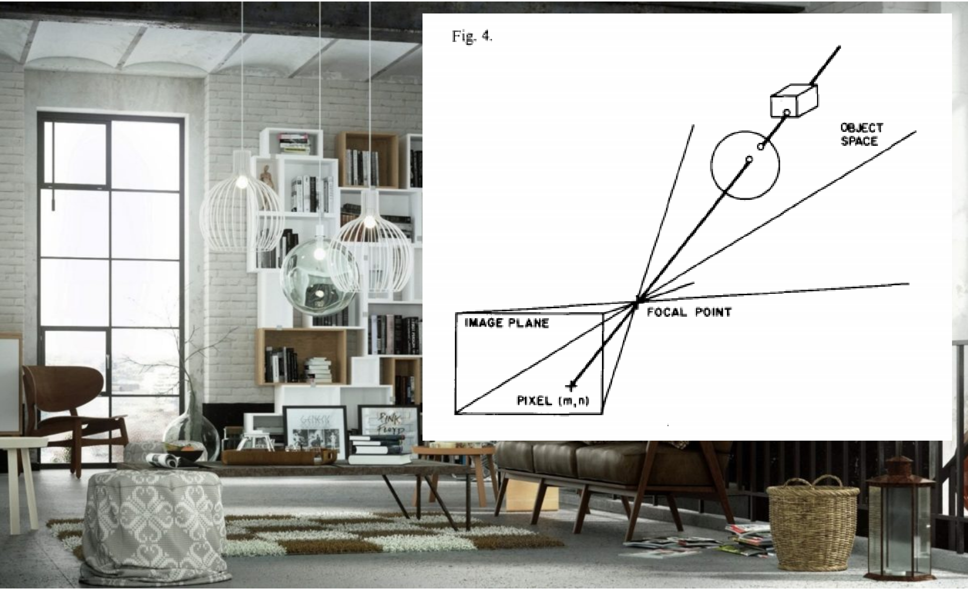
Ampere, RTX, 3090:
Raytracingkerner: ?

GEFORCE RTX 3090

NVIDIA CUDA® kerner	10496	Raytracingkerner	2. generation	Standard hukommelseskonfiguration	24 GB GDDR6X
Høj CPU-hastighed (GHz)	1.70	Tensor Cores	3. generation	Hukommelsesgrænsefladens bredde	384-bit
Normal CPU-hastighed (GHz)	1.40				

GPUs

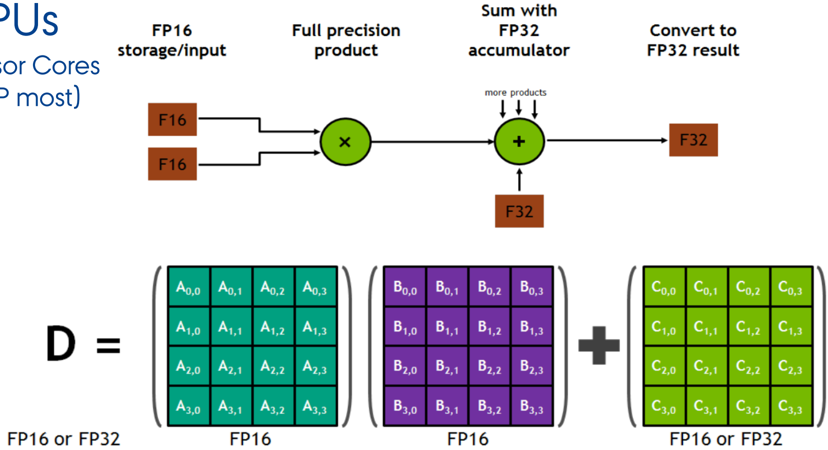
Tensor cores: Raytracing vs Rasterization on GPUs



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

GPUs

Tensor Cores (SKIP most)



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

[volta-architecture-whitepaper.pdf]

Analog: DSD: `half-adder`, `full-adder`, `ripple-carry-adder` with `n-bit multiplier` based on adders (scales `n-bit`^{2})..

HPC Top500

Best High-Performance Computer

[https://www.top500.org/list/2007/06/]
[https://en.wikipedia.org/wiki/List_of_Nvidia_graphics_processing_units]

Model	Launch	Transistors (billion)	Die size (mm^2)	Processing power (TFLOPS)			
				Single precision	Double precision	Half precision	Tensor compute (FP16) (2:1 sparse)
GeForce RTX 3070[120]	October 29, 2020[121]	17.4	392.5	17.66 20.37	0.276 0.318	17.66 20.37	141.31 162.98
GeForce RTX 3080[123]	September 17, 2020	28.3	628.4	25.06 29.76	0.392 0.465	25.06 29.76	200.54 238.14
GeForce RTX 3090[124]	September 24, 2020	28.3		29.38 35.68	0.459 0.558	29.38 35.68	235.08 285.48

TOP500
The List.

HOME LISTS STATISTICS RESOURC

Home »Lists »TOP500 »June 2007

JUNE 2007

Rank	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	BlueGene/L - eServer Blue Gene Solution, IBM DOE/NNSA/LLNL United States	131,072	280.6	367.0	1,433
2	Jaguar - Cray XT4/XT3, Cray/HPE DOE/SC/Oak Ridge National Laboratory United States	23,016	101.7	119.3	
3	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core, Cray/HPE NNSA/Sandia National Laboratories United States	26,544	101.4	127.4	

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!

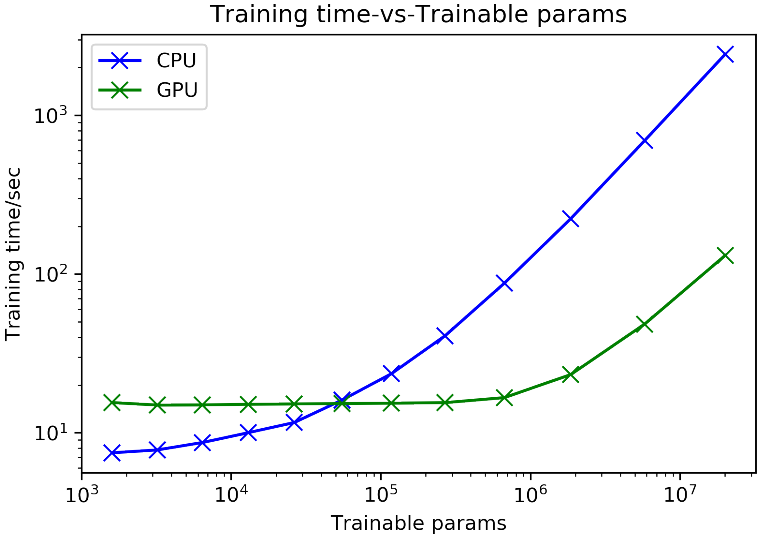
`model.summary()`:

```
# 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

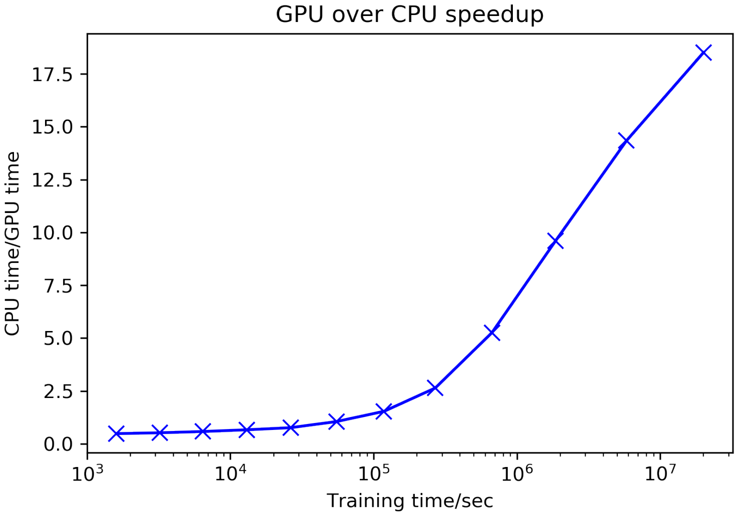
CPU vs GPU on MNIST for a three layer NN with dropout...



GPUs

Actual test on the GPU-server

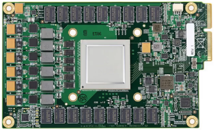
CPU vs GPU on MNIST for a three layer NN with dropout...



TPUs

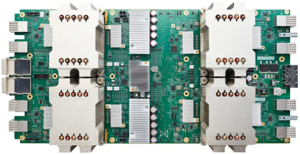
Tensor Processing Units

Custom ASICs by Google



TPU v1

Launched in 2015
Inference only

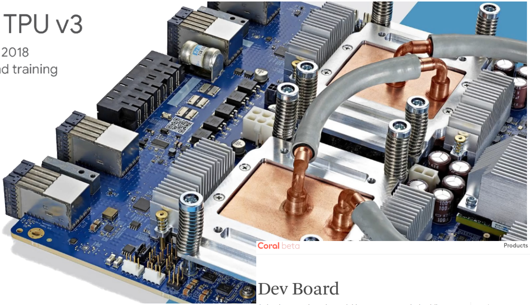


TPU v2

Launched in 2017
Inference and training

Cloud TPU v3

Launched in 2018
Inference and training



Coral beta   Products   Docs

Dev Board

A development board to quickly prototype on-device ML products. Scale from prototype to production with a removable system-on-module (SOM).

→ Datasheet
→ Get started guide

$149.99

Buy

Dev board with Google Edge TPU ML accelerator coprocessor

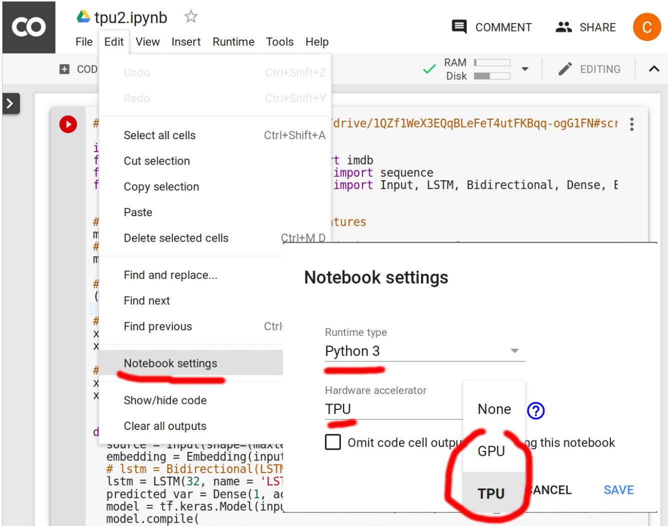
[https://coral.withgoogle.com/products/dev-board/]

TPUs

Access to TPUs (SKIP)

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? (SKIP most)

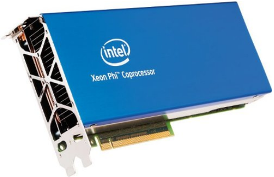
- Huge advantage for TPU performance-per-watt,
- Colab performance:
  inconclusive (TPU part does not work yet),
- TPU only for inference?

	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

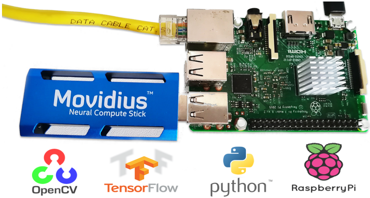
[https://www.extremetech.com/computing/247403-nvidia-claims-pascal-gpus-challenge-googles-tensorflow-tpu-updated-benchmarks]

Exotic Hardware

- Intel Phi multicore CPU, 64 i386 cores:



- Raspberry PIs + Intel Movideus stick



- FPGAs