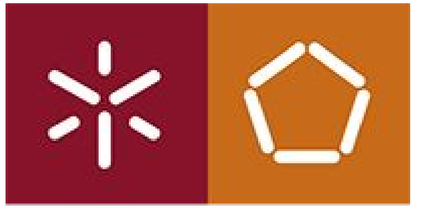


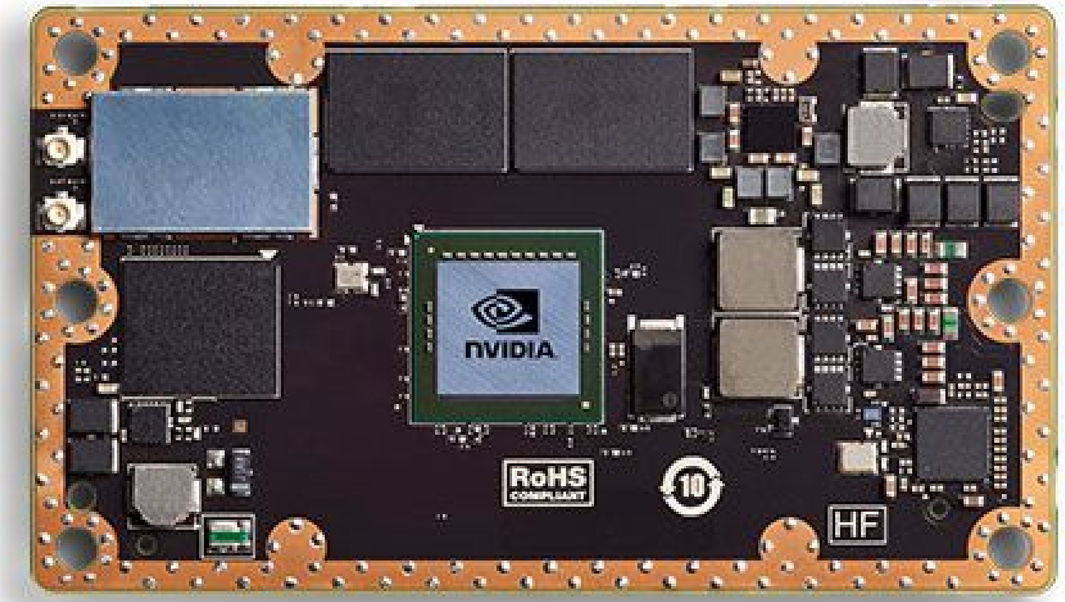
Nvidia Jetson TX2 and its Software Toolset

João Fernandes
2017/2018

In this presentation

- Nvidia Jetson TX2: Hardware
- Nvidia Jetson TX2: Software
- Machine Learning: Neural Networks
  - Convolutional Neural Networks
- Tensorflow
- A case study

NVIDIA Jetson TX2



NVIDIA Jetson TX2
Developer Kit

NVIDIA Jetson TX2

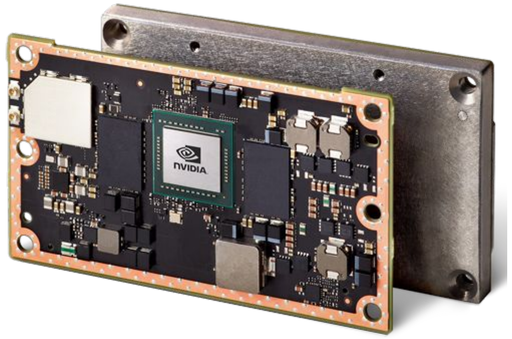
- “Jetson is the world’s leading low-power embedded platform, enabling server-class AI compute performance for edge devices everywhere”
- “Edge computing is an emerging paradigm which uses local computing to enable analytics at the source of the data”
- “Jetson TX2 accelerates cutting-edge deep neural network (DNN) architectures using the NVIDIA cuDNN and TensorRT libraries”

Image Recognition
Classification

Object Detection
Localization

Segmentation
Free Space

Hardware Specifications

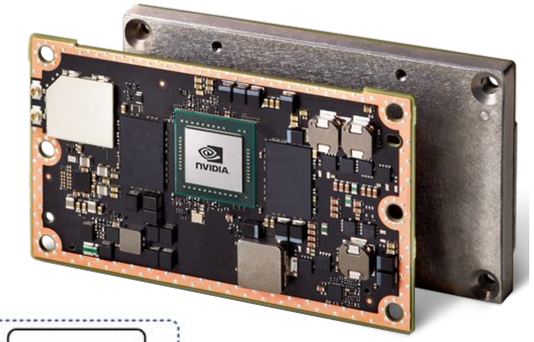


	NVIDIA Jetson TX2
CPU	ARM Cortex-A57 (quad-core) @ 2GHz + NVIDIA Denver2 (dual-core) @ 2GHz
GPU	256-core Pascal @ 1300MHz (2 SM's)
Memory	8GB 128-bit LPDDR4 @ 1866Mhz | 59.7 GB/s
Storage	32GB eMMC 5.1
I/O	UART, SPI, I2C, I2S, GPIOs, HDMI 2.0, USB, Ethernet, CAN

Heterogeneous Multi-Processing: Cortex-A57 vs Denver2

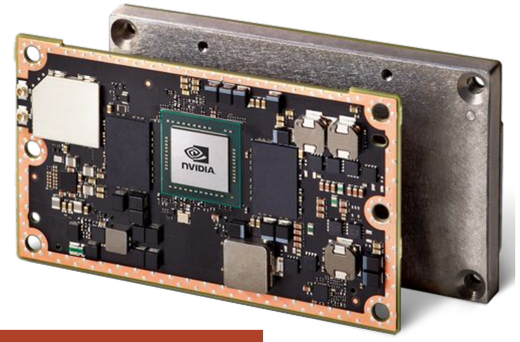
	4xCortex - A57	2xDenver2
ARM ISA	ARMv8 (32/64 bits)	ARMv8 (32/64 bits)
Frequency	Up to 2 GHz	Up to 2 GHz
L1 Cache	48 KB (Inst) + 32 KB (Data)	128KB (Inst) + 64 KB (Data)
L2 Cache	2MB (shared between the 2 Chips)	2MB (shared between the 2 Chips)
GFLOPS(FP32)	750	437
Full coherency guaranteed across the 2 CPU Clusters		

Hardware Architecture



MCPU Power Rail
BCPU Power Rail
Denver 2 CPU Cluster
Cortex-A57 CPU Cluster
GIC
CPU switch fabric (**Coherent**)
CoreSight
BPMP, APE, SPE, SCE
APB Bridges
Clock & Reset
Timers, Mailboxes, Semaphore, Config Reg, Sys Config
GPIO Controller
AXI
APB Bridge
CAN, PMC, I2C, SPI, DMIC, GPIO
UART, I2C, DTV, SPI, Fuses, ThrmSnsr, PWFM
APB Bridges
SysRAM TZRAM iRAM
Cortex-R5 BPMP
Boot, **Power Mgmt.**
DMA
System fabric
General purpose DMA
PCIE
Cortex-A9 APE Audio Processor
Audio IO
XUSB **eAVB**, UFS, SD/MMC, SATA
GPU Power Rail
GPU **Pascal**
128 CUDA Cores
128 CUDA Cores
compress decompress
Host Command buffer and Synchonization
CSI camera input
camera ISP
Video encode
Video decode
JPEG
Security TSECs & SE
2D graphics
decompress
Display **3x** heads
decompress
Memory Controller w/**ECC**
System MMU
Memory fabric and arbitration
32/64/128b LPDDR4

Performance Modes

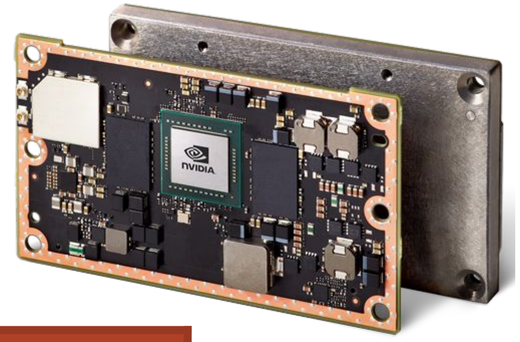


Mode	Mode Name	Denver2	ARM A57	GPU Freq.
0	Max-N	2 @ 2.0 GHz	4 @ 2.0 GHz	1.30 GHz
1	Max-Q	0	4 @ 1.2 GHz	0.85 GHz
2	Max-P Core All	2 @ 1.4 GHz	4 @ 1.4 GHz	1.12 GHz
3	Max-P ARM	0	4 @ 2.0 GHz	1.12 GHz
4	Max-P Denver	2 @ 2.0 GHz	0	1.12 GHz

Can be changed at run time using:

```
sudo nvpmodel -m <desired mode>
```

Max-Q vs Max-P



		Max-N	Max-P	Max-Q
GoogLeNet	Perf	290 FPS	253 FPS	196 FPS
	Power Consumption	12.8 W	8.9 W	5.9 W
	Efficiency (FPS/W)	22.7	28.5	33.2
AlexNet	Perf	692 FPS	601 FPS	463 FPS
	Power Consumption	12.4 W	8.6 W	5.6 W
	Efficiency	55.8	69.9	82.7

The Software: NVIDIA JetPack

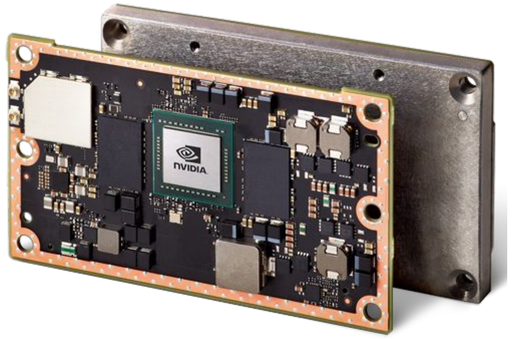
- A set of software tools to ease the development for the Jetson platform:
  - Flash Jetson Developer Kit with the latest OS image
  - Install developer tools for both host PC and Developer Kit
  - Install libraries and APIs
  - Samples and documentation
- Key Software:
  - OS: L4T (Based on Ubuntu)
  - CUDA 8
  - TensorRT 2.1
  - cuDNN 6.0

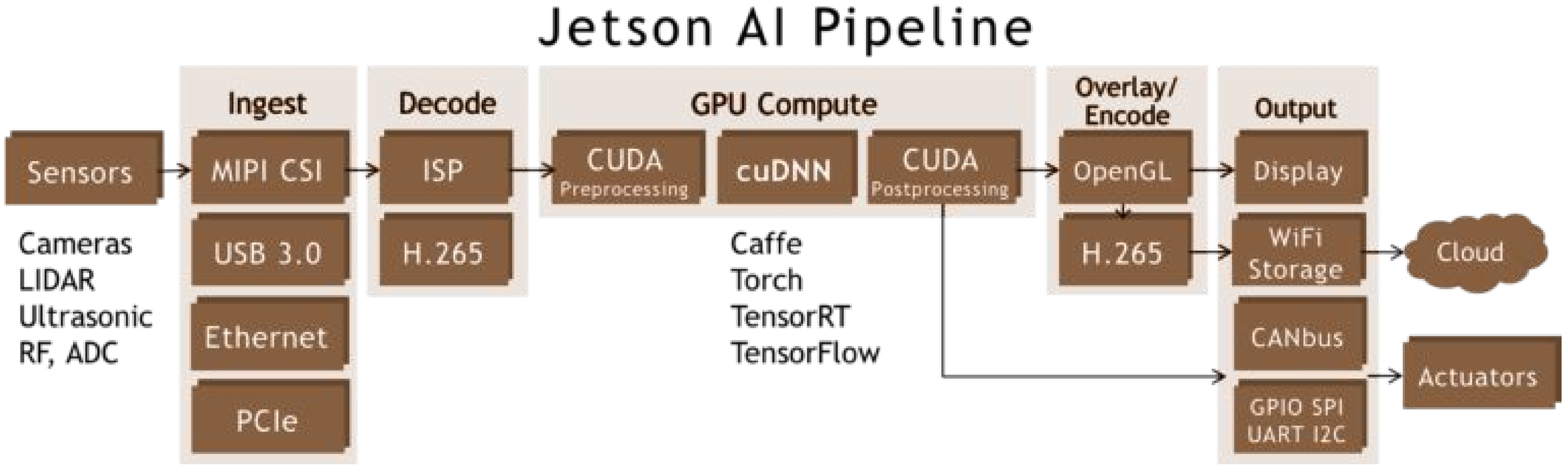
Requires Ubuntu 14 on the Host!

The Software

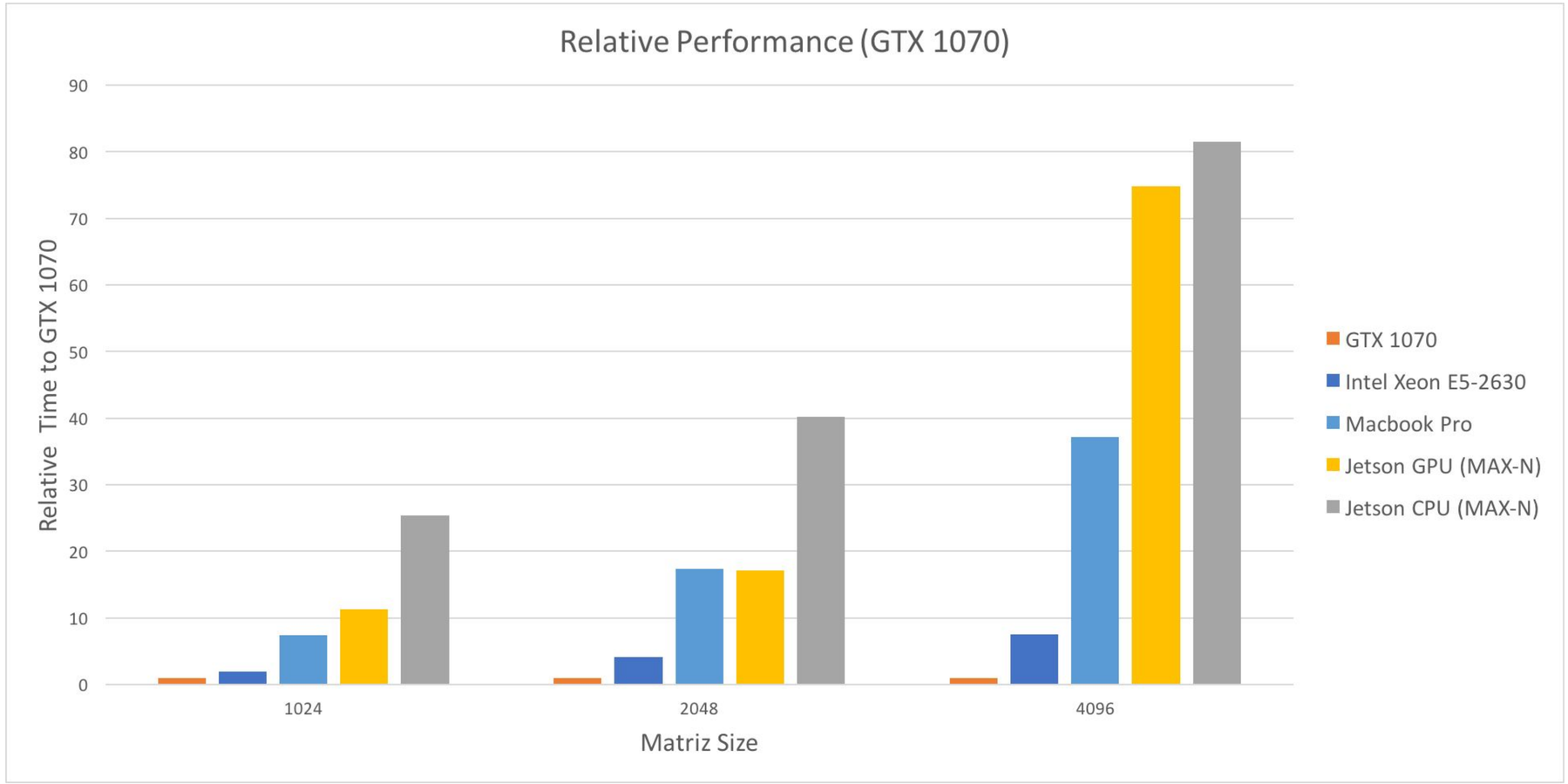
- CUDA
  - CUDA Toolkit provides a comprehensive development environment for C and C++ developers building GPU-accelerated applications. It includes a compiler for NVIDIA GPUs, math libraries, and tools for debugging and optimizing the performance of your applications
- cuDNN
  - CUDA Deep Neural Network library provides high-performance primitives for all deep learning frameworks. It includes support for convolutions, activation functions and tensor transformations.
- TensorRT 2.1
  - TensorRT is a high performance deep learning inference runtime for image classification, segmentation, and object detection neural networks. It speeds up deep learning inference as well as reducing the runtime memory footprint for convolutional and deconv neural networks.
- Multimedia API
  - The Jetson Multimedia API package provides low level APIs for image acquisition (via camers). It enables video decode, encode, format conversion and scaling functionality

Real World Utilization





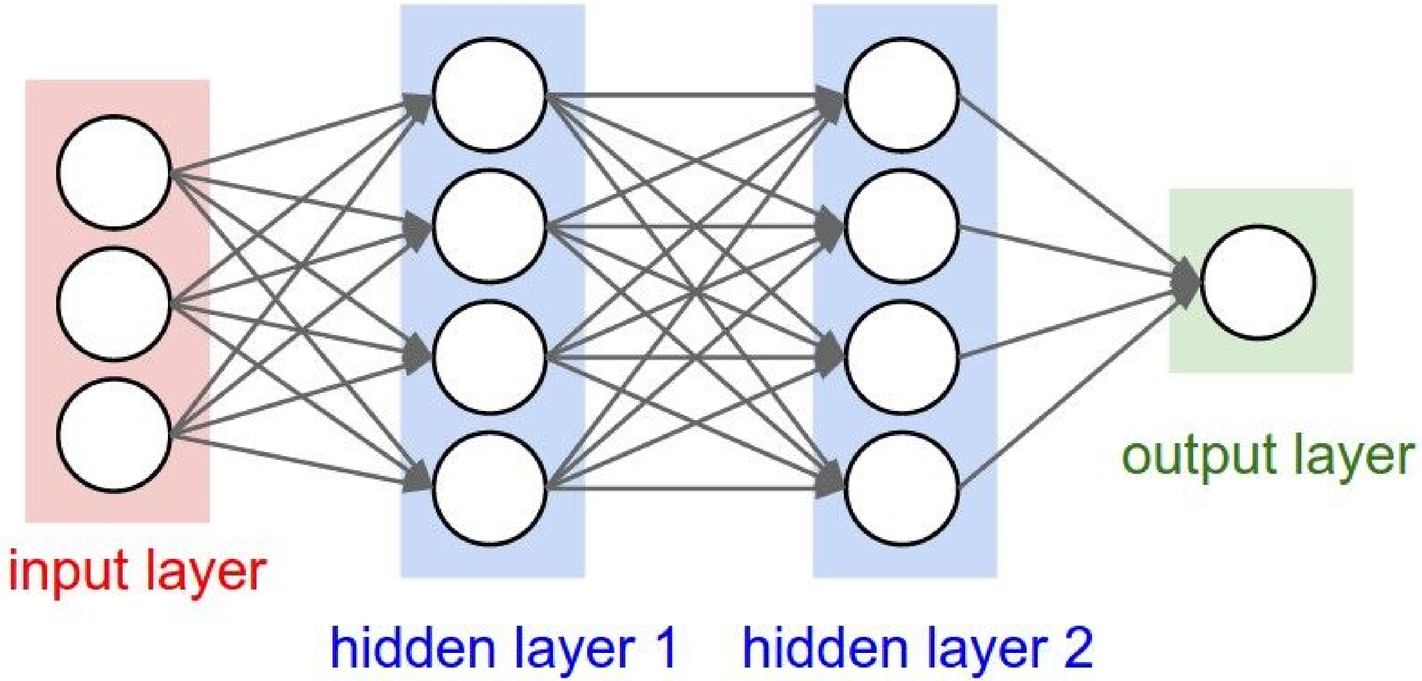
Relative Performance (GTX 1070) Matrix-Mul FP32



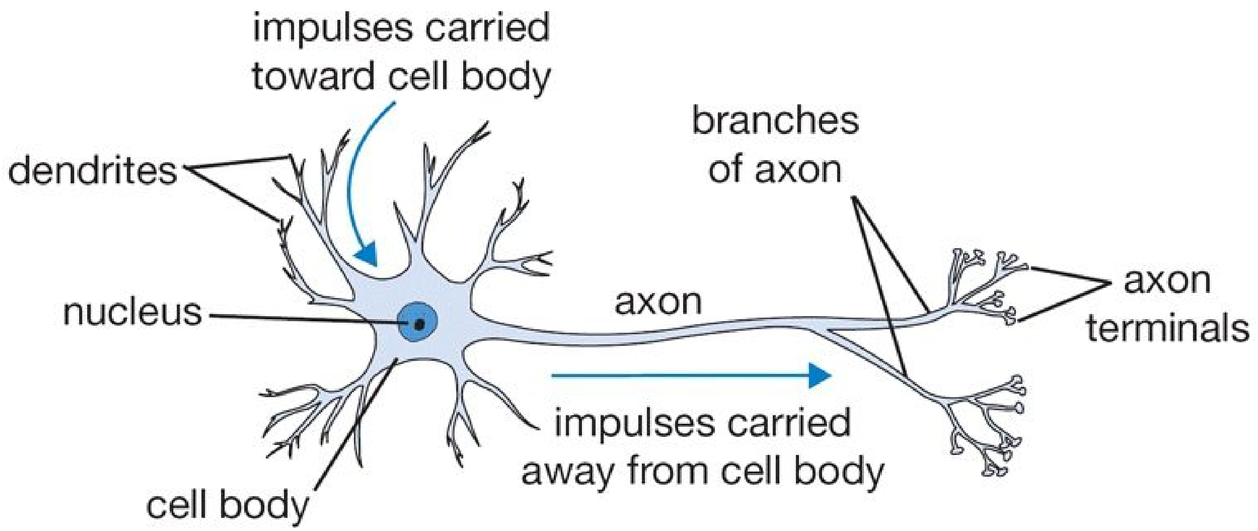
Machine Learning

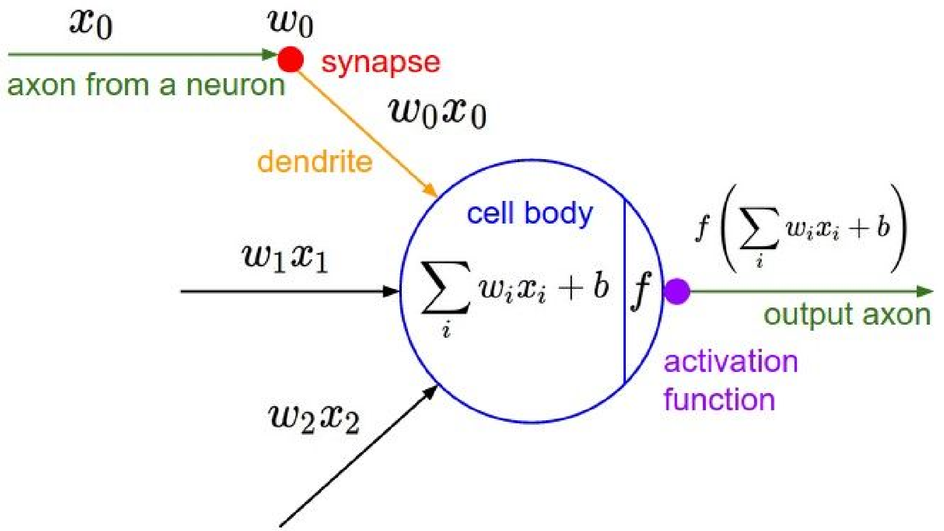
- Machine learning is the science of getting the computer to perform a certain task without explicitly programming it. Normally, this requires a sample dataset from which the algorithm can infer the model parameters (commonly known as the training phase).
  - Decision tree learning
  - Clustering
  - **Neural Networks**

Neural Networks

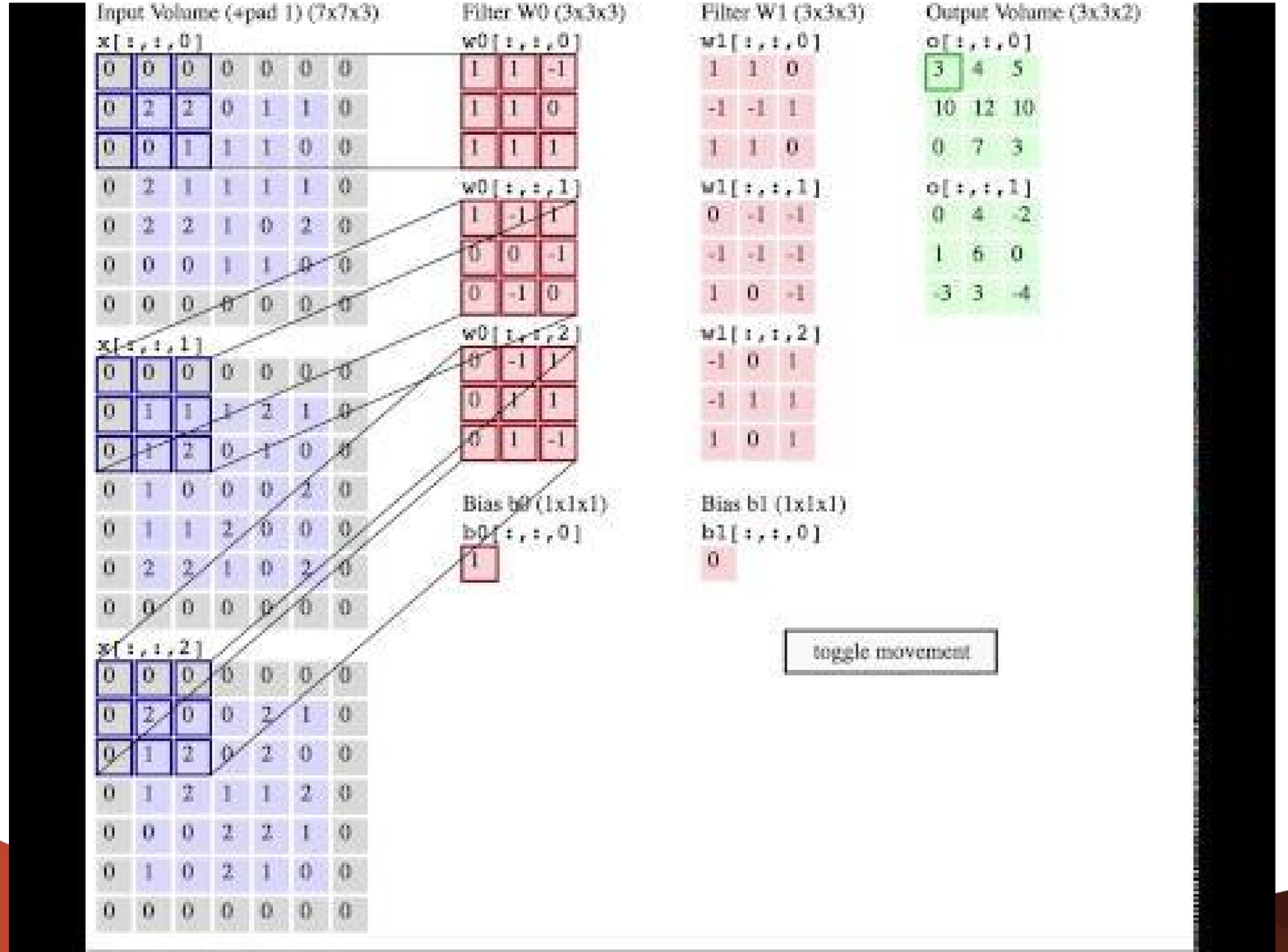


Neural Networks: The neuron

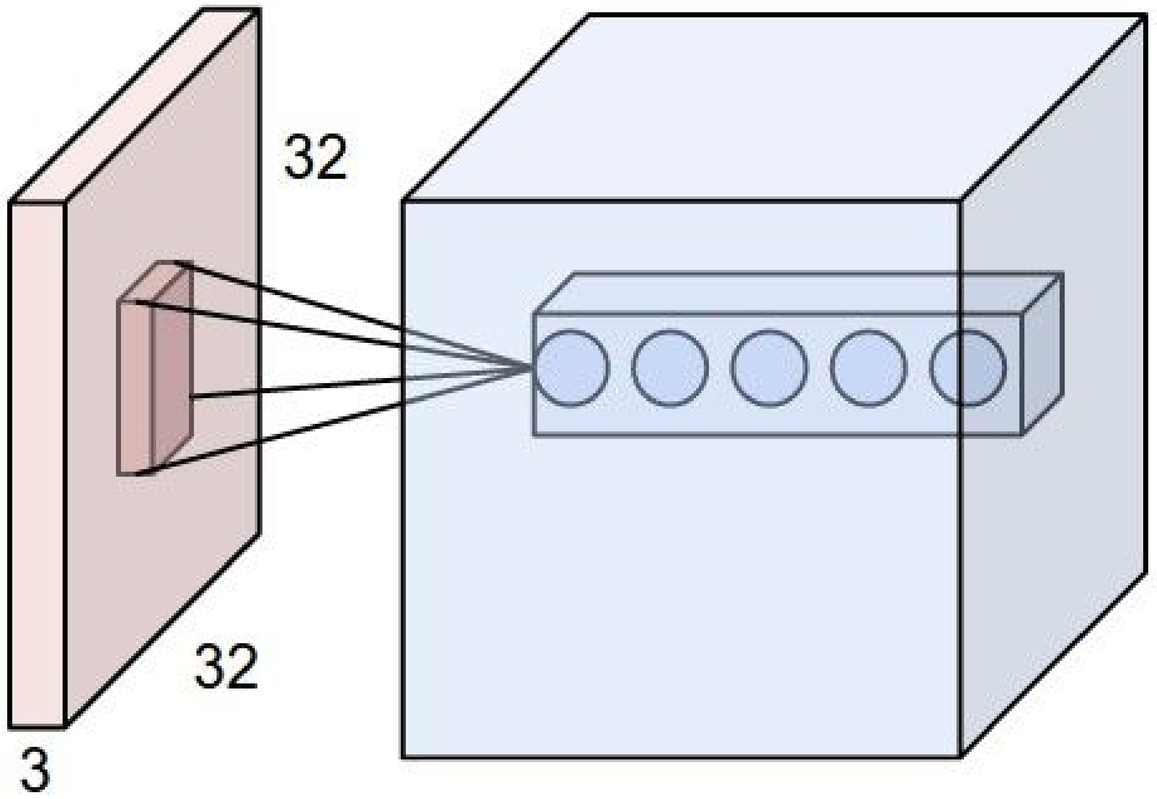




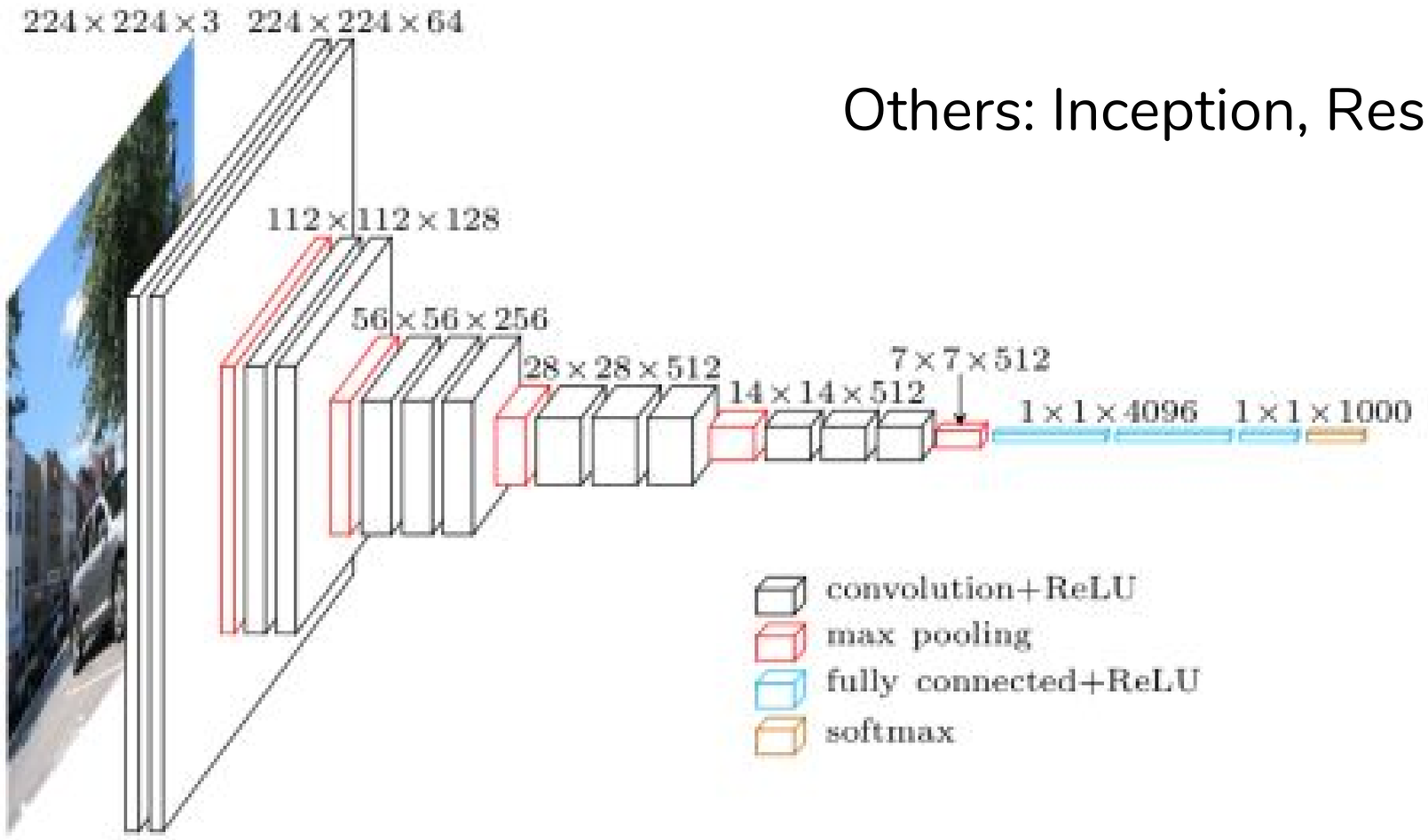
Convolutional Neural Networks



Convolutional Neural Networks



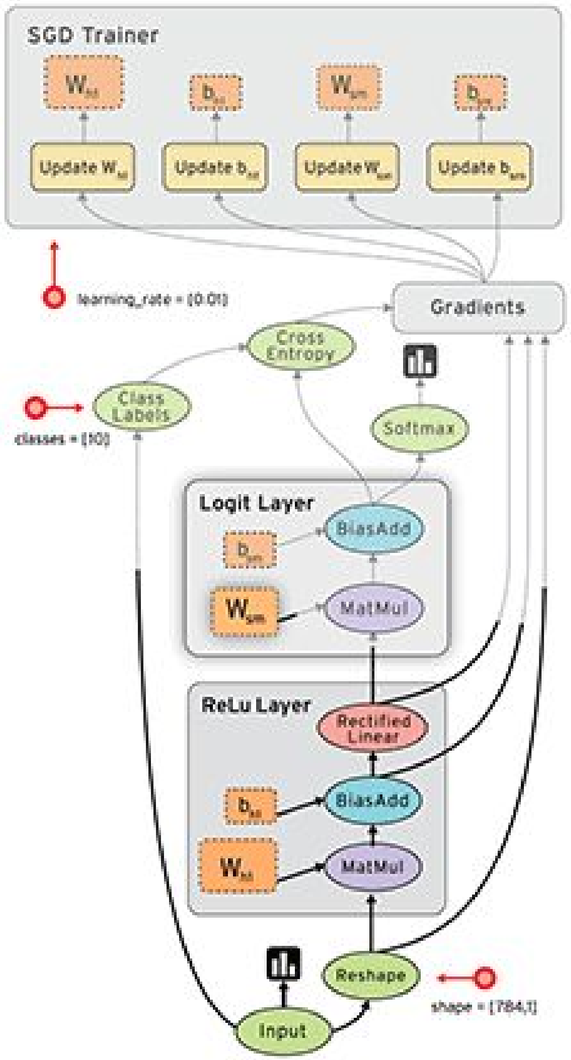
State of the art: Object Classification (VGG16)



Others: Inception, ResNet

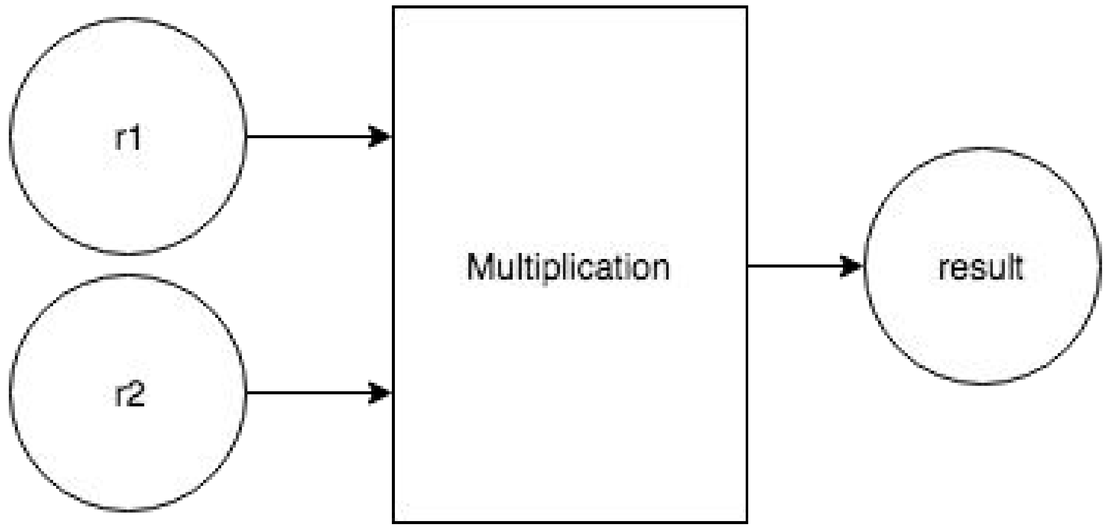
Tensorflow

TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them.

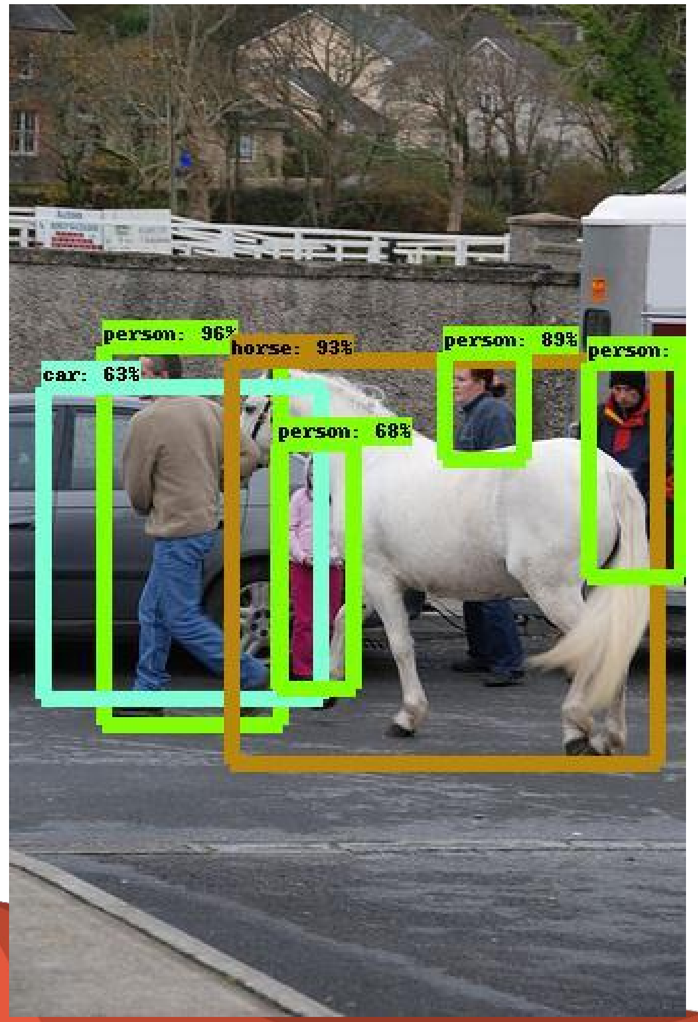


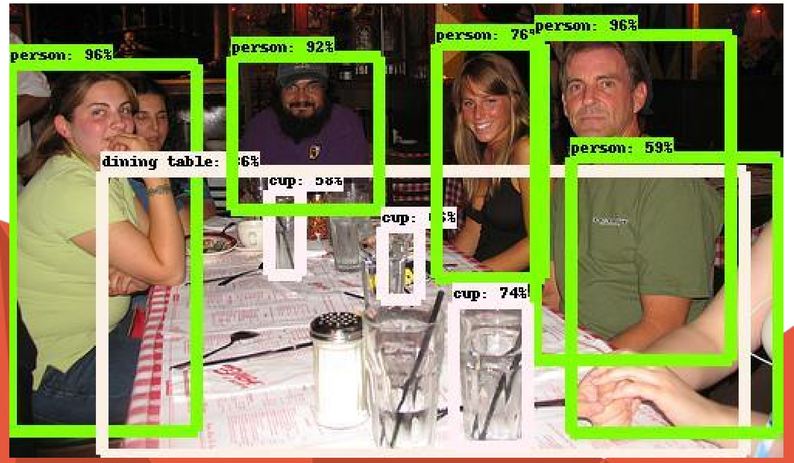
Computational Graph

```
with tf.Session() as session:
        r1 = tf.random_uniform(shape=shape, minval=0, maxval=1, dtype=data_type)
        r2 = tf.random_uniform(shape=shape, minval=0, maxval=1, dtype=data_type)
        dot_operation = tf.matmul(r2, r1)
        start_time = time.time()
        result = session.run(dot_operation)
```



The Case Study



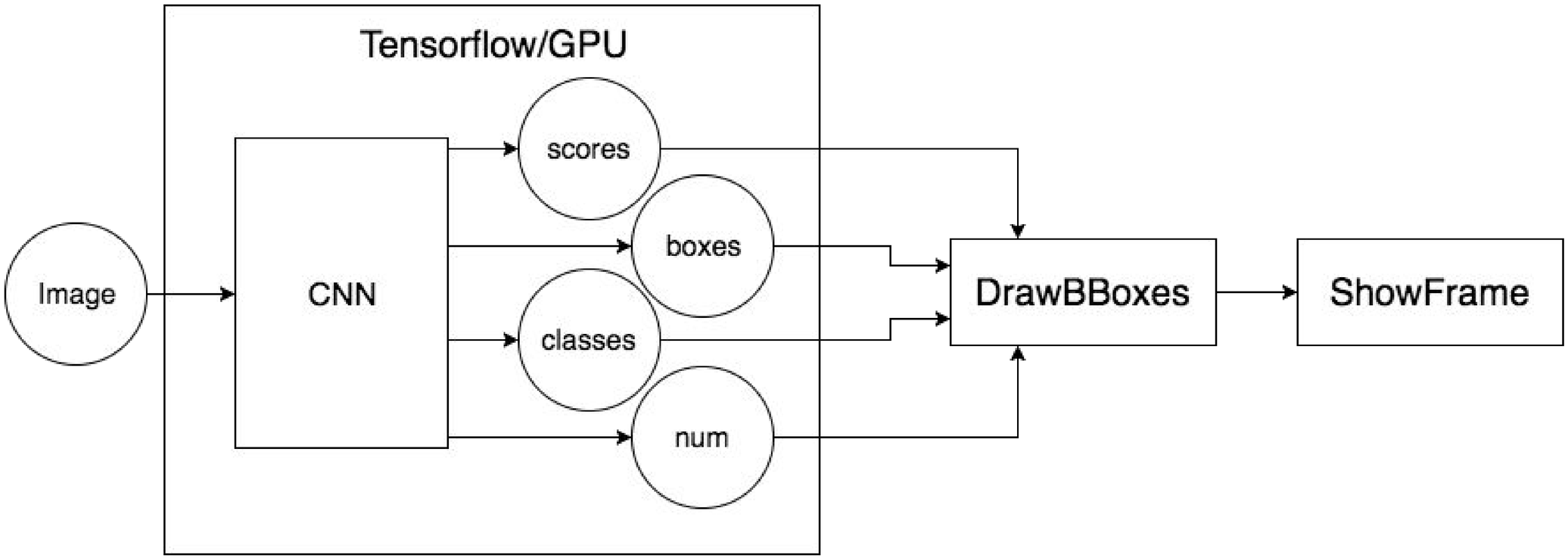


Naive Implementation

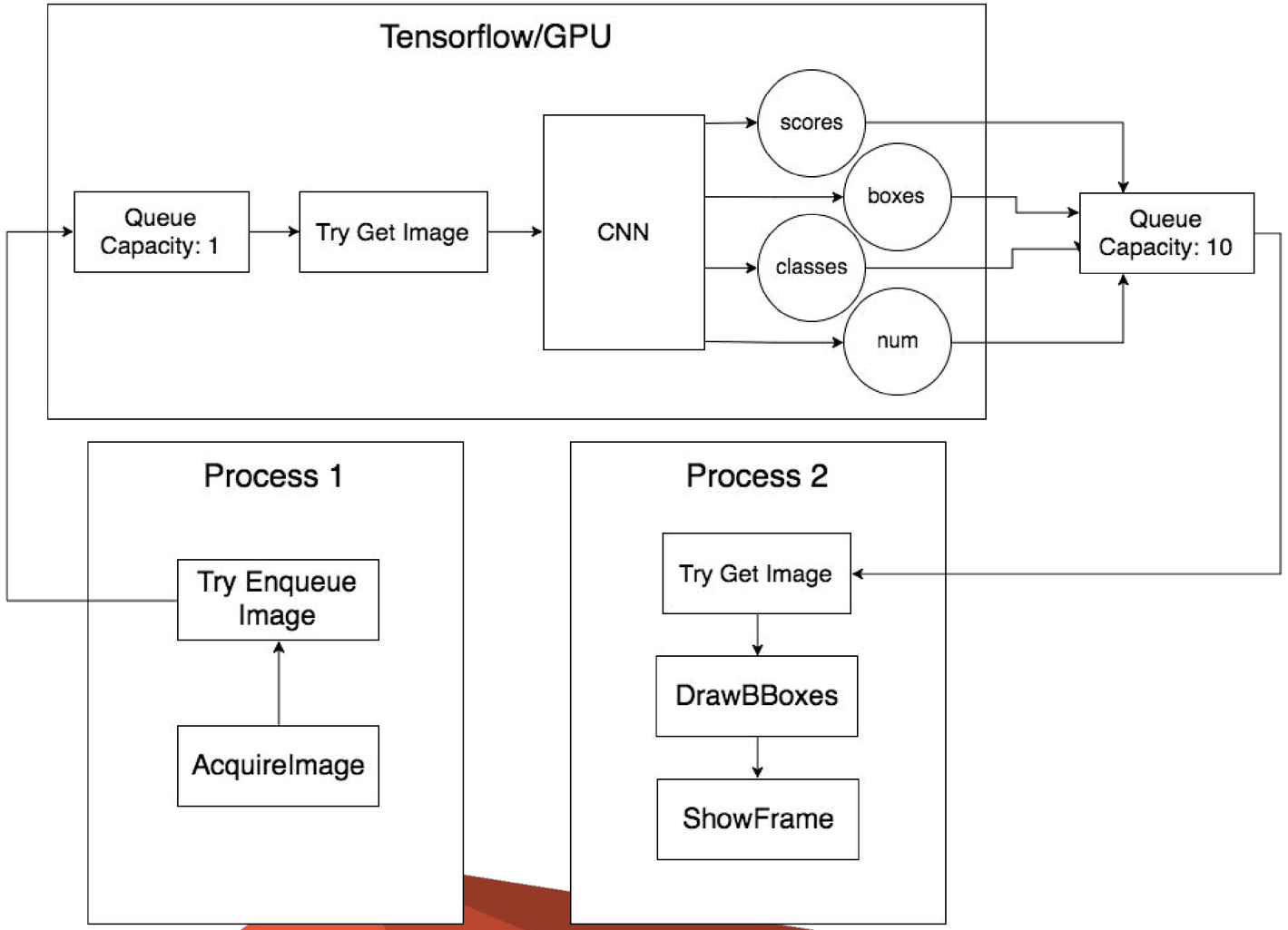
```
while(True):
    image = CaptureFrame()
    (boxes, scores, classes, num) = sess.run(
        [detection_boxes, detection_scores, detection_classes, num_detections],
        feed_dict={image_tensor: image_np_expanded})

    DrawBBoxes(image, boxes, scores, ...)
    ShowFrame(image)
```

Naive Implementation: Computational Graph



Implementation: Computational Graph



Conclusion

- Performance Engineering can look like it’s changing… But the basics are always the same:
  - Measure/Profile
  - Identify and Understand the Bottleneck
  - Start the iterative tuning process
  - Validade
  - Be scientific