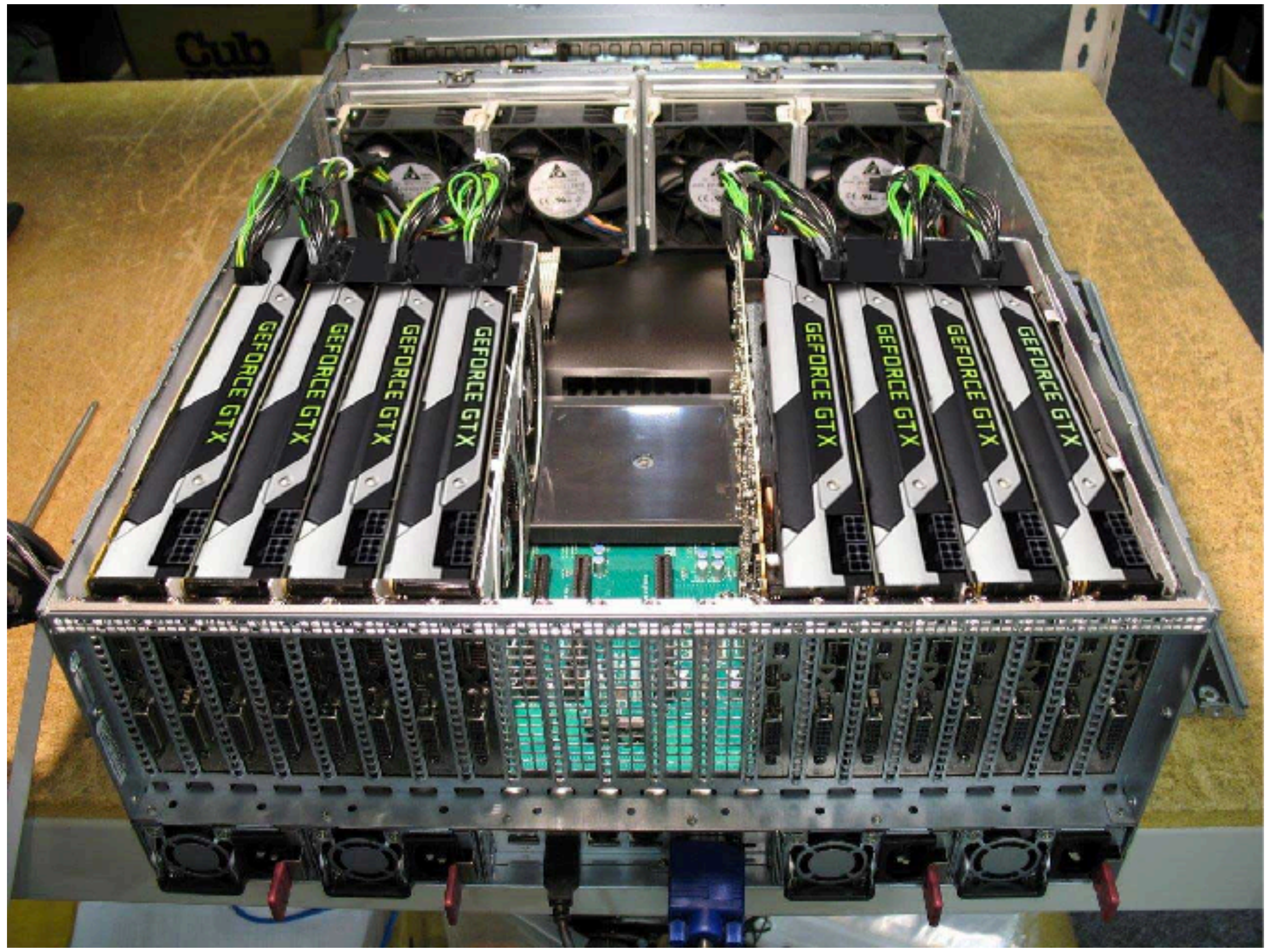
Challenges in Deep Learning for Mobile & Embedded

Carlo C. del Mundo
carlo@xnor.ai

Demo

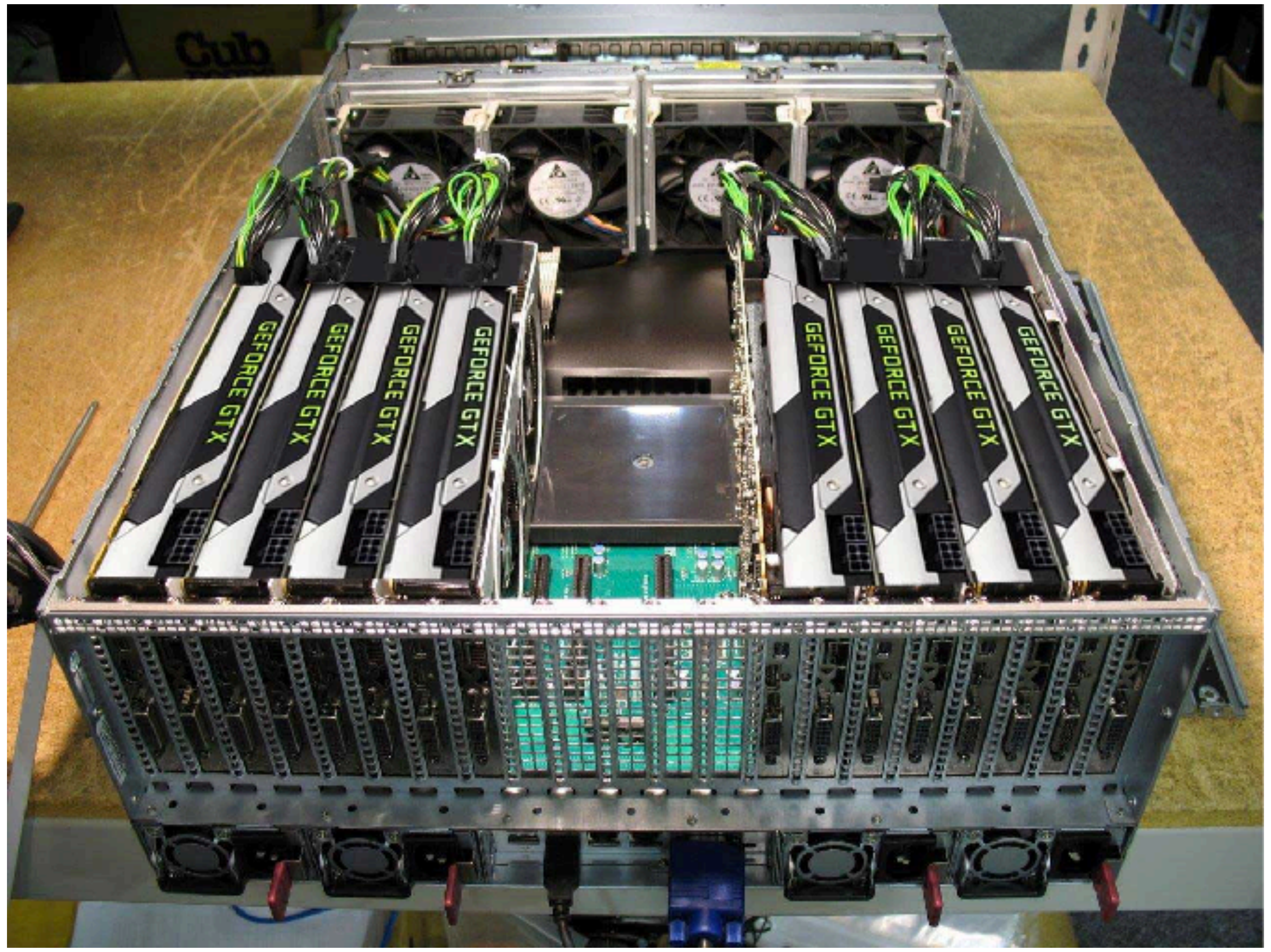
Where should I train?

- DNNs are typically trained on a high-powered, server machine with **one** or **many** GPUs.



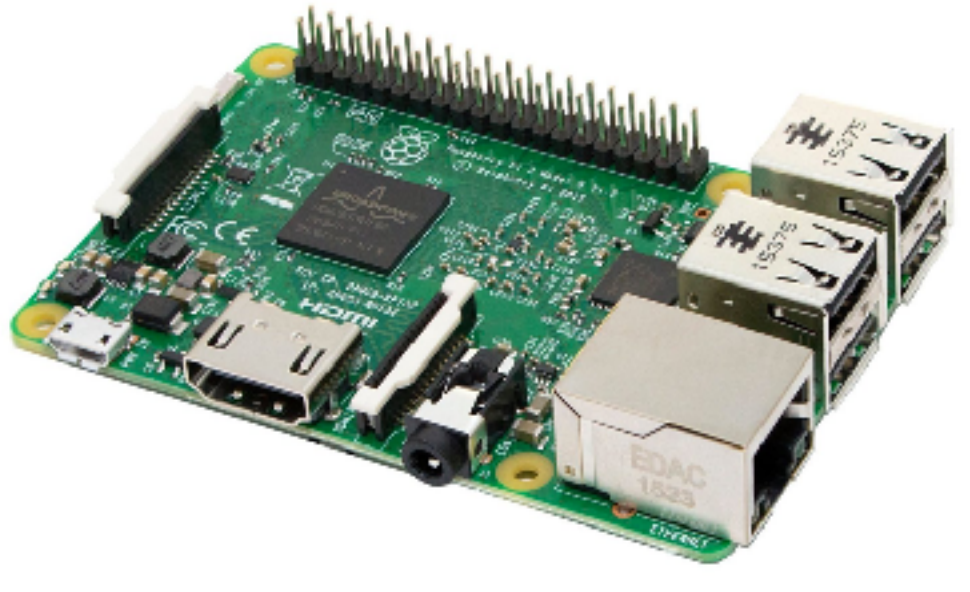
Where should I deploy?

- The machine that you train on is the machine you deploy on.



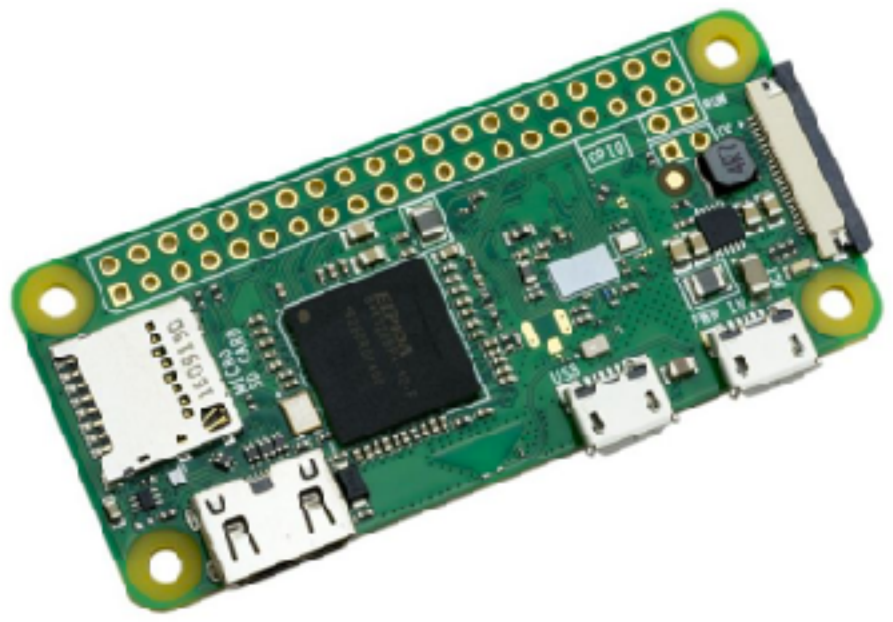
Where should I deploy?

- There’s significant interest in applying DNN-based solutions for a variety of devices.



Challenge #1

Which compute platform?



Machines for Deep Learning

- A popular platform for researchers and developers alike: **NVIDIA Jetson TX2** ($599 retail, $299 education).

	Jetson TX2
GPU	NVIDIA Pascal™, 256 CUDA cores
CPU	HMP Dual Denver 2/2 MB L2 + Quad ARM® A57/2 MB L2
Video	4K x 2K 60 Hz Encode (HEVC) 4K x 2K 60 Hz Decode (12-Bit Support)
Memory	8 GB 128 bit LPDDR4 59.7 GB/s
Display	2x DSI, 2x DP 1.2 / HDMI 2.0 / eDP 1.4
CSI	Up to 6 Cameras (2 Lane) CSI2 D-PHY 1.2 (2.5 Gbps/Lane)
PCIE	Gen 2 | 1x4 + 1x1 OR 2x1 + 1x2
Data Storage	32 GB eMMC, SDIO, SATA
Other	CAN, UART, SPI, I2C, I2S, GPIOs

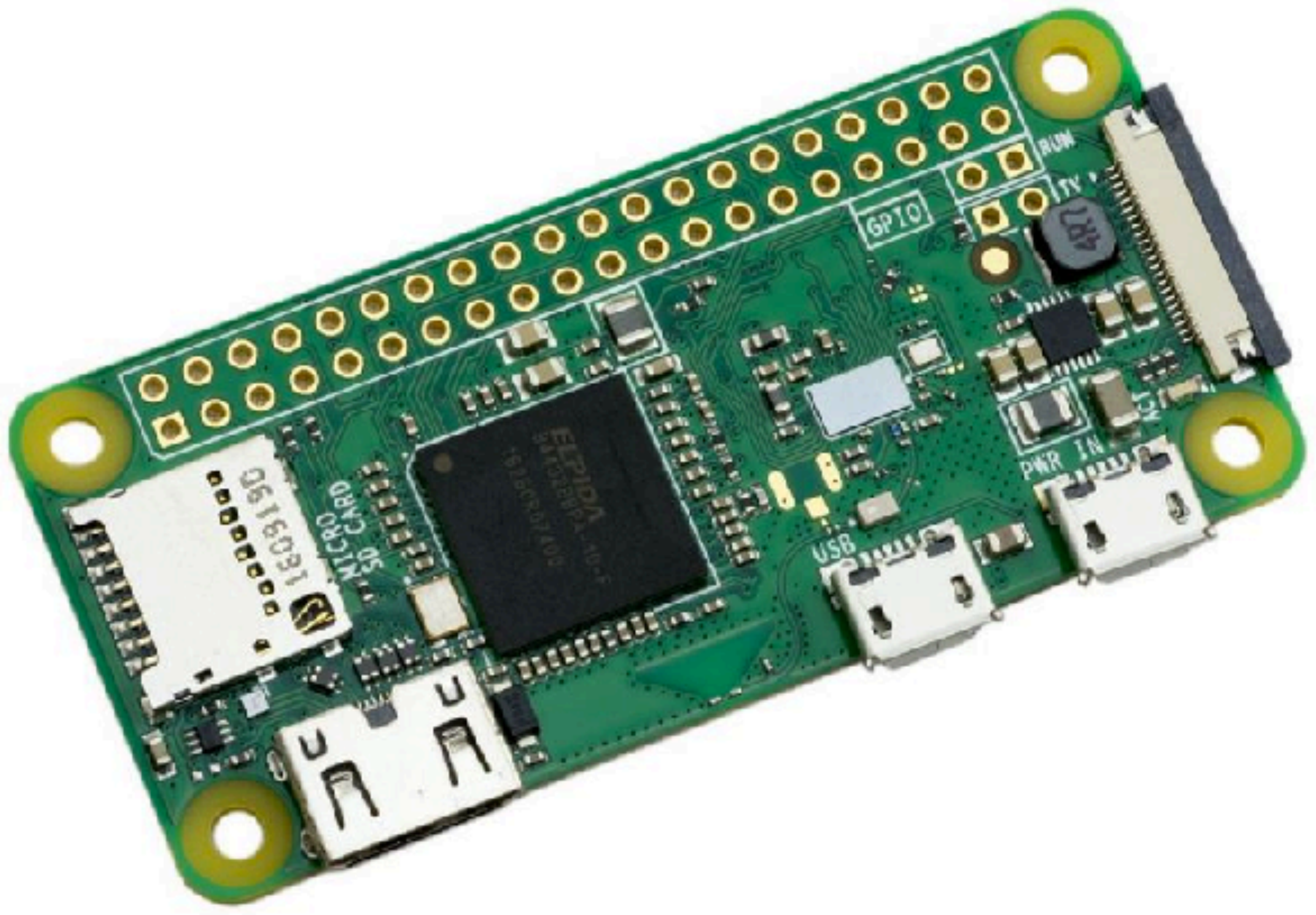
Machines for Deep Learning

- A ubiquitous mobile phone: **Apple iPhone 7+ 5.5”** ($769).

	Apple iPhone 7 and 7 Plus
Platform	iOS 10
Display	4.7" IPS LCD w/ 750 x 1334 pixels (for iPhone 7) 5.5" IPS LCD w/ 1080 x 1920 pixels (for iPhone 7 Plus)
SoC	Apple A10
RAM	2 GB
Storage	32 GB
Camera	12-megapixel iSight (iPhone 7) Dual rear camera (iPhone 7 Plus)
Battery	~ 1,715mAh for iPhone 7 ~ 2,750mAh for iPhone 7 Plus

Machines for Deep Learning

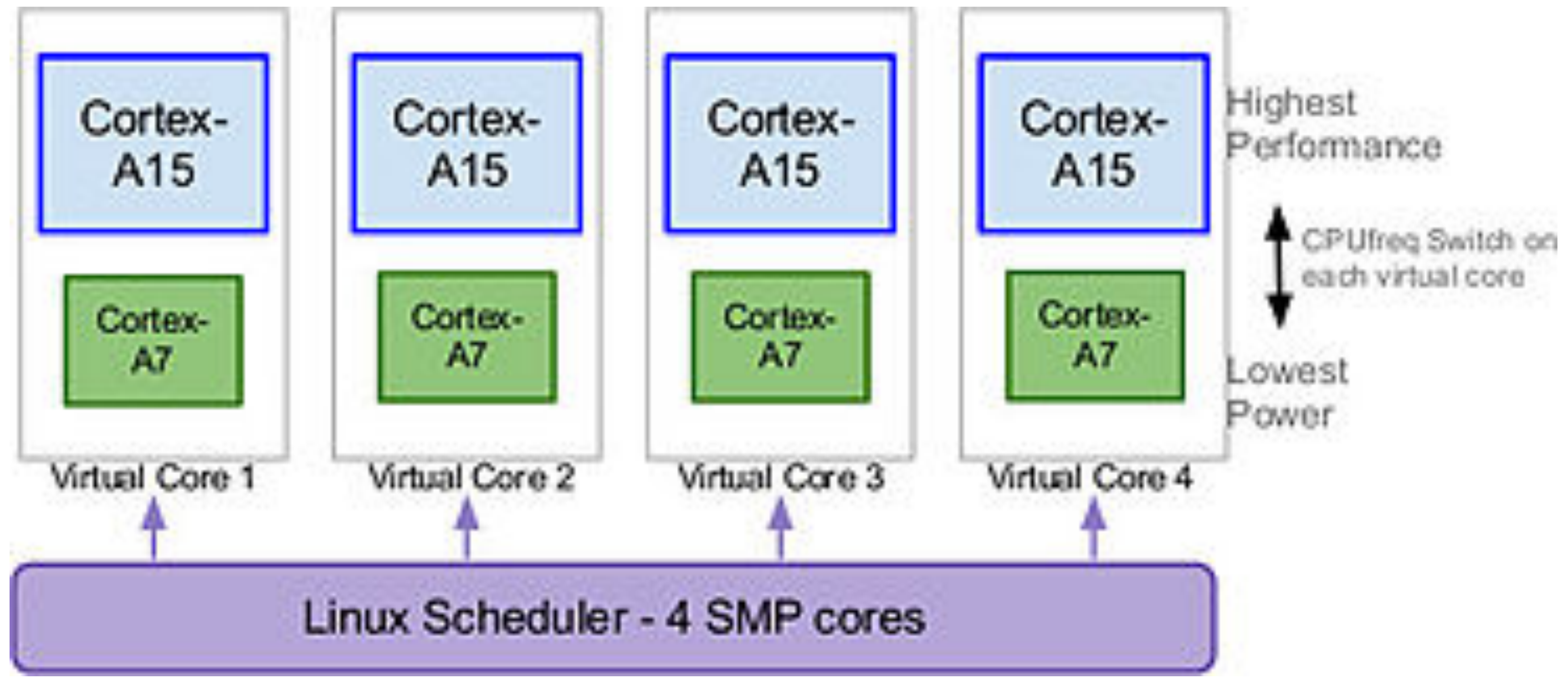
- A cheap, device: **Raspberry Pi Zero** ($5 board only).



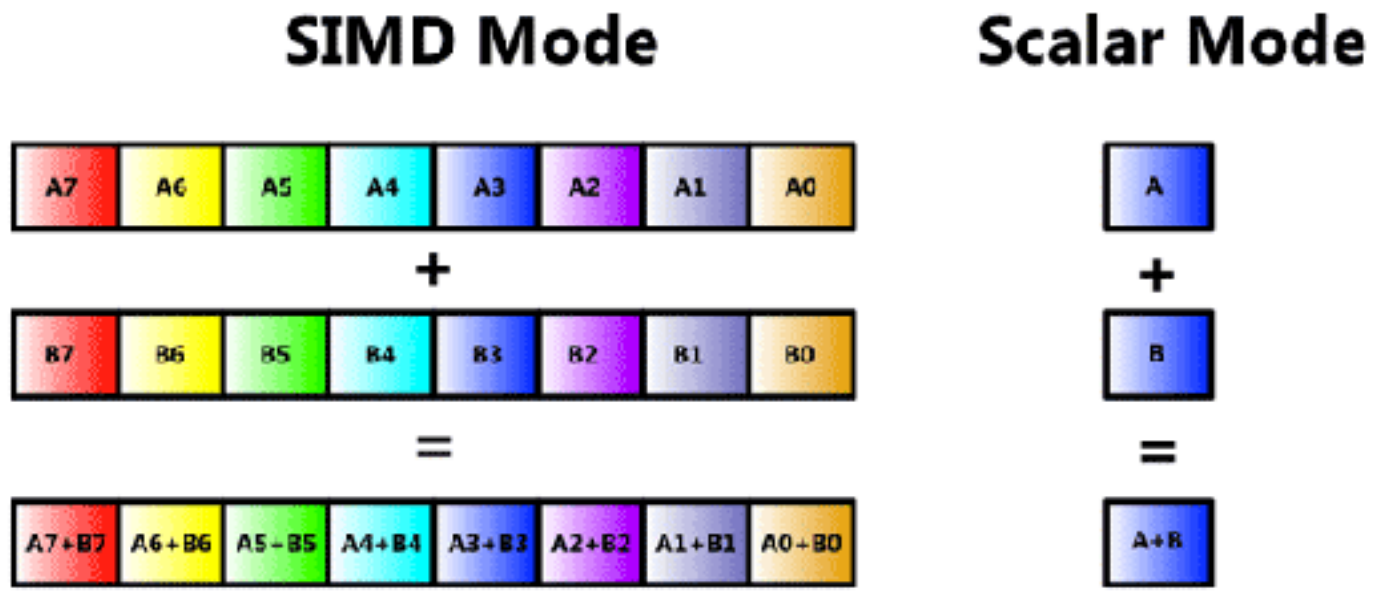
Raspberry Pi Zero
11/25/2015
BCM2835
ARM11 @ 1GHz
ARMv6
250MHz VideoCore IV
512 MB SDRAM
micro-SD
none
none
HDMI / Composite
HDMI
40
$5

Considerations in a Platform

- Does it have multiple cores? Deep learning is embarrassingly parallel, so more cores is better.



- Does it have vector units? Similar to above, most computation is vectorizable.



Considerations in a Platform

- Does it have specialized instructions useful for deep learning?

```
int _popcnt64 (__int64 a)                                   popcnt
__m256 _mm256_fmadd_ps (__m256 a, __m256 b, __m256 c)       vfmadd132ps, vfmadd213ps, vfmadd231ps
```

dp4a
Four-way byte dot product-accumulate.

8.7.4.1. Half Precision Floating Point Instructions: add

- Does it have a mobile GPU? The iPhone 7+ has a considerable mobile GPU rivaling that of desktop GPUs from a decade ago!.

Model	Date	Clusters	Die Size (mm²)	Config core[4]	SIMD lane	Fillrate			Bus width (bit)	HSA-features	API (version)						GFLOPS(@ 1 GHz) FP32/FP16
						MPolygons/s	(GP/s)	(GT/s)			Vulkan (API)	OpenGL ES	OpenGL	OpenVX	OpenCL	Direct3D	
GT7200 Plus	January 2016	2	?	2/4	64/128						1.0	3.2	3.3 (4.4 optional)	1.0.1	2.0	??	128 / 256
GT7400 Plus	January 2016	4	?	4/8	128/256												256 / 512
GT7600 Plus	June 2016	6	10 nm	6/12	192/384						1.0	3.2	4.4	1.0.1	2.0	12	384 / 768

Which compute platform?

Recommendations.

- NVIDIA Jetson TX2.
  - Pros: (1) relatively cheap, (2) fantastic compute with FP16 support, (3) has peripherals for USB cameras and other sensors.
  - Cons: (1) Not very portable.

- iPhone 7+.
  - Pros: (1) integration with camera and other sensors exposed via Apple libraries, (2) excellent battery life, (3) ubiquitous platform, (4) Accelerate framework.
  - Cons: (1) requires you to learn some XCode/Swift. (2) porting from other NN libraries is non-trivial.

- Raspberry Pi Zero.
  - Pros: (1) cheap, it’s $5, (2) can be easily battery-powered.
  - Cons: (1) severely resource-constrained, (2) no vendor libraries, (3) no vector units.

Challenge #2

Which deep learning framework?

PYTORCH

mxnet

TensorFlow ™

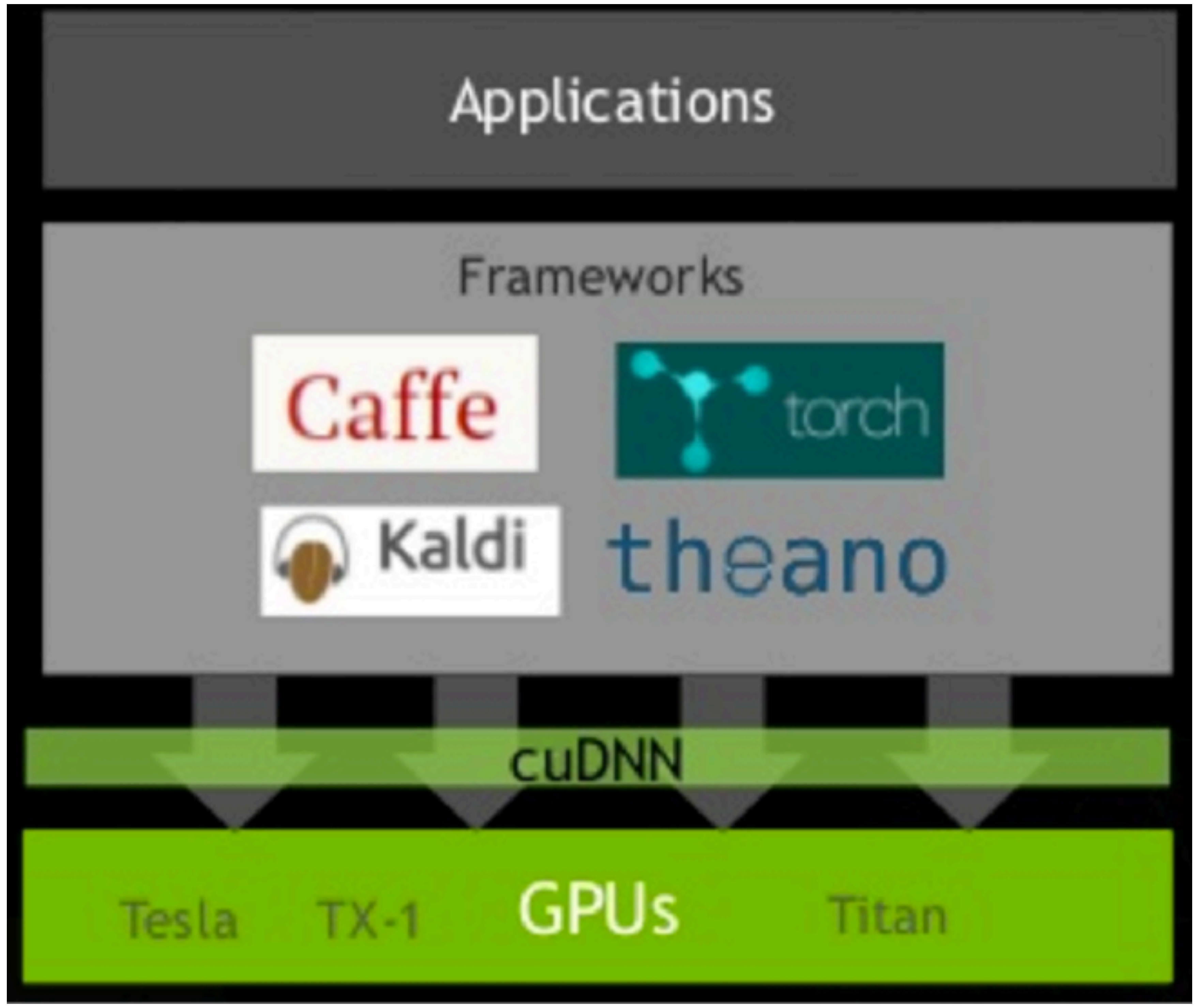
XNOR.AI

Caffe

Darknet

Frameworks use the same building blocks

- Frameworks (e.g, Torch) are built on libraries (e.g, cuDNN).
- Most frameworks use the same back-end library. Differences in implementations will play a minor role in the future.



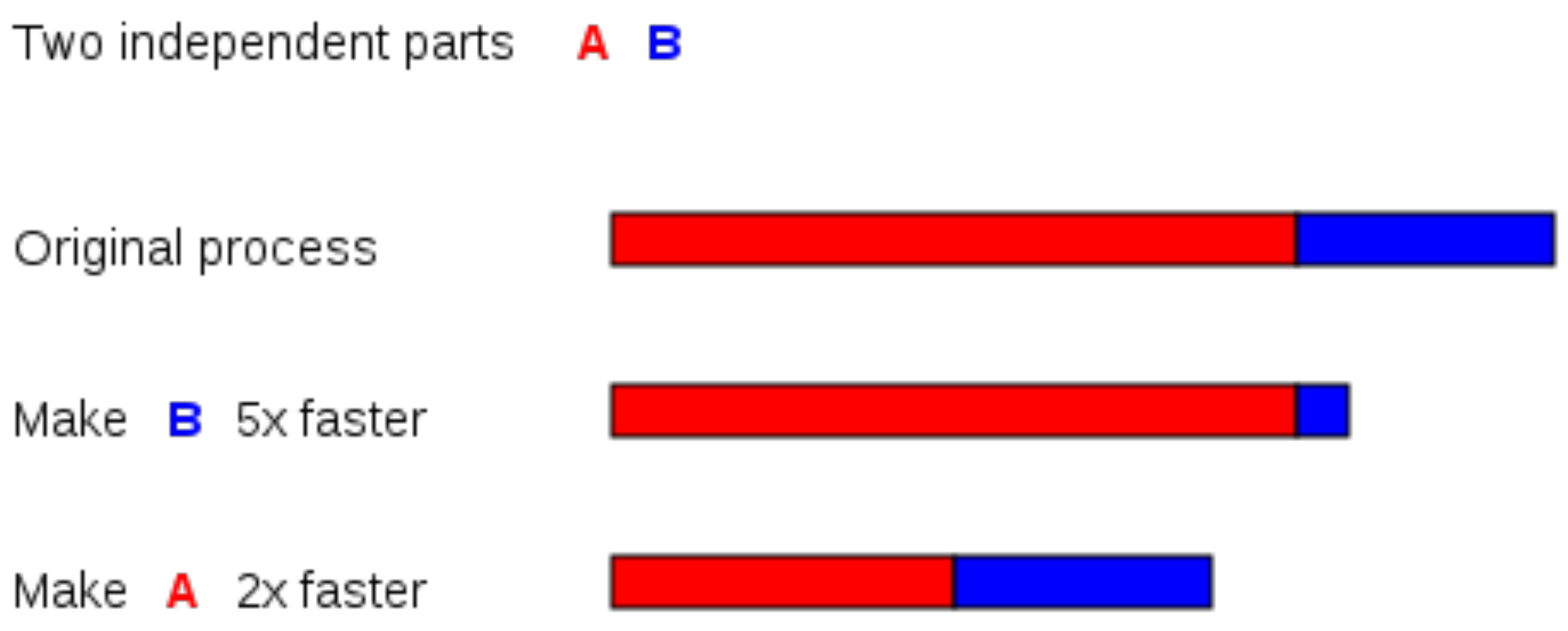
Deep Learning Building Blocks

- **Platform-specific libraries**
  - NVIDIA: cuDNN.
  - AMD: MIOpen.
  - Apple: Accelerate + CoreML.
  - Intel: Nervana Neon + Intel MKL.
  - ARM: ARM Compute Library.

- **Platform-agnostic libraries**:
  - Matrix multiplication: Eigen, OpenBLAS, Atlas, Gemmlowp.
  - FFT/Winograd Convolutions: NNPack.

Debunking Framework Performance

- **Amdahl's Law**: Overall speedup dependent on: (1) % of time the task consumes, (2) how much faster you sped up that task.



- Assume a DNN spends time: 70% convolutions, 30% everything else (pooling, activations, normalizations).
  - Accelerate convolutions by 2x: **1.53x** overall speedup.
  - Accelerate convolutions by 5x: **2.27x** overall speedup.
  - Accelerate convolutions by ∞x: **3.33x** overall speedup.

Challenge #3

Deployment from Training to Inference

Training & Deployment Strategies

- A common strategy is to train models using frameworks like PyTorch, TensorFlow, MXNet, etc.

- Once training is perfected, the network's weights are **frozen**, **extracted**, and **converted** to an inference-optimized pipeline (e.g., Facebook trains with PyTorch and deploys with Caffe2).

Separation of Inference vs. Training

- For high-performance, the deployment pipeline should be specialized for inference.

- An Inference Pipeline for object detection consists of:
  - **Image Acquisition:** acquire data from a sensor such as a camera.
  - **RGB to Float Conversion:** transform uint8_t (byte) values to float (4 bytes). Perform standardization, if required.
  - **Data Marshaling**: transpose data to data layout for back-end network.
  - **Inference**: run a forward pass on a newly acquired image.
  - **Decode**: make sense of the output in an application-specific manner.
  - **Draw**: for object detection, draw a video overlay with bounding box outputs.