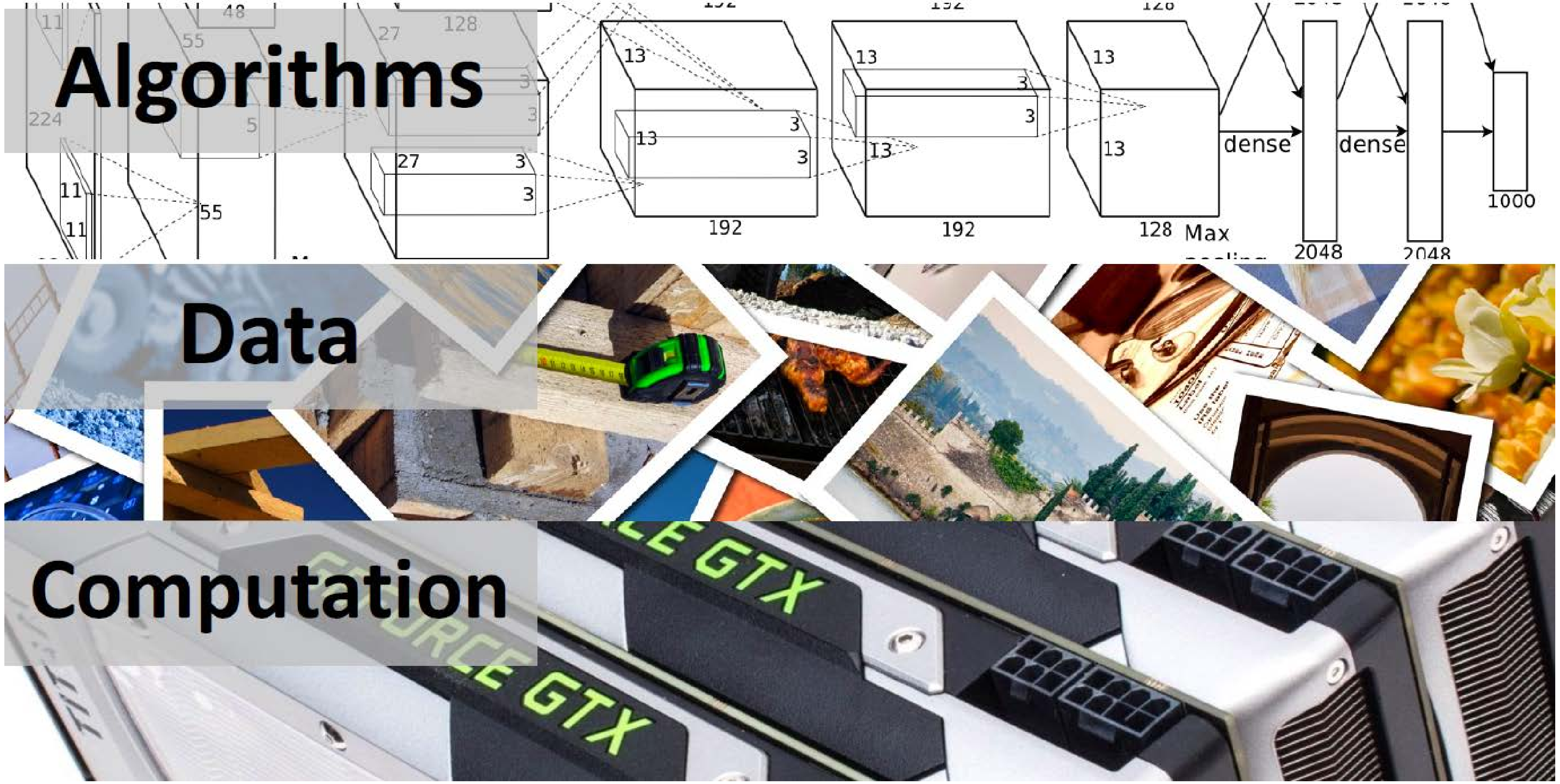
DNN Accelerators

Outlines

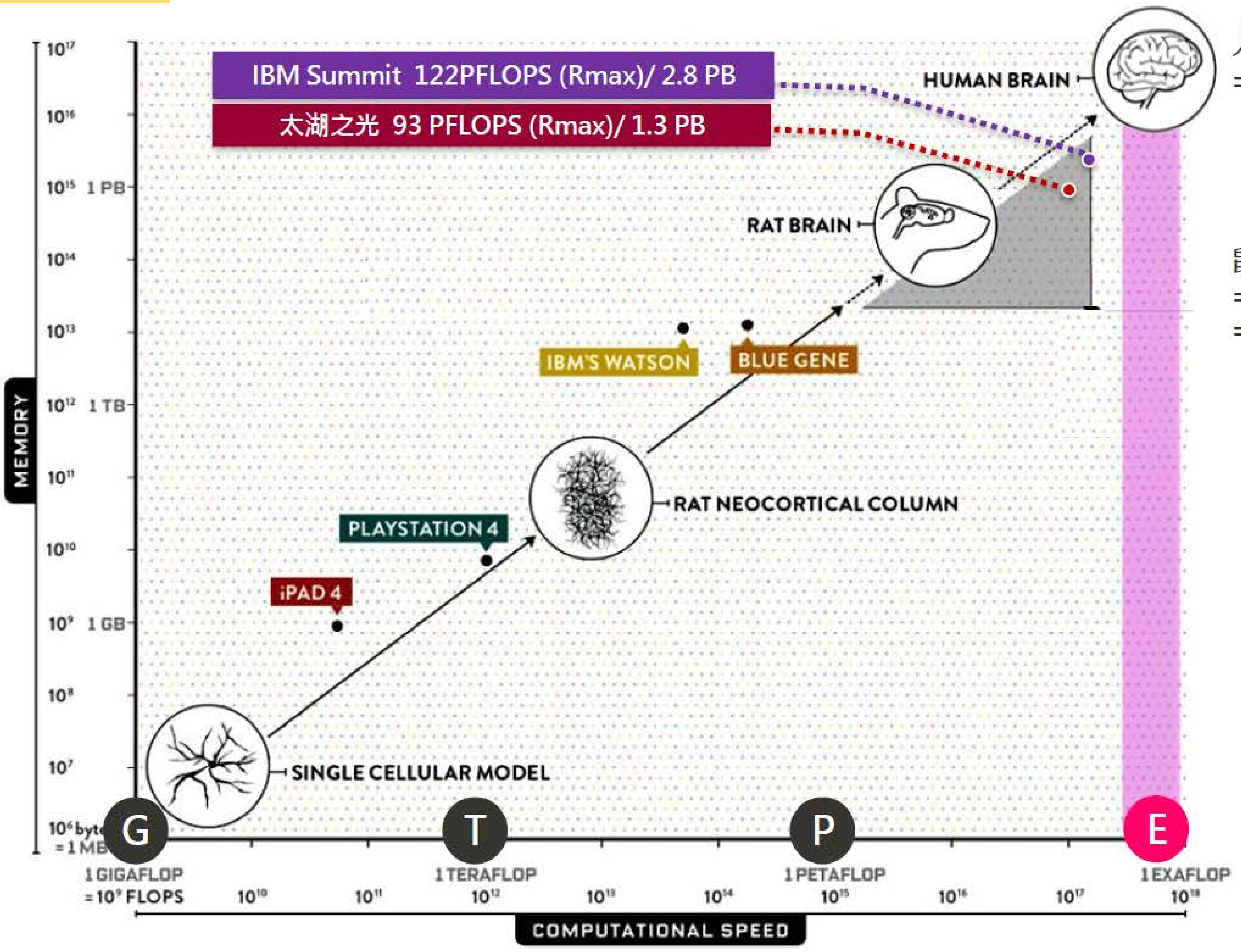
- GPU cudnn
- ASIC
- Xilinx xfdnn

Ingredients in Deep Learning

- Algorithm (Model) + Training Data + Computation
  - CNN models + Big Data + Accelerators



Computation Complexity



人類大腦 ~1 Exaflops
= ~1000 鼠類大腦

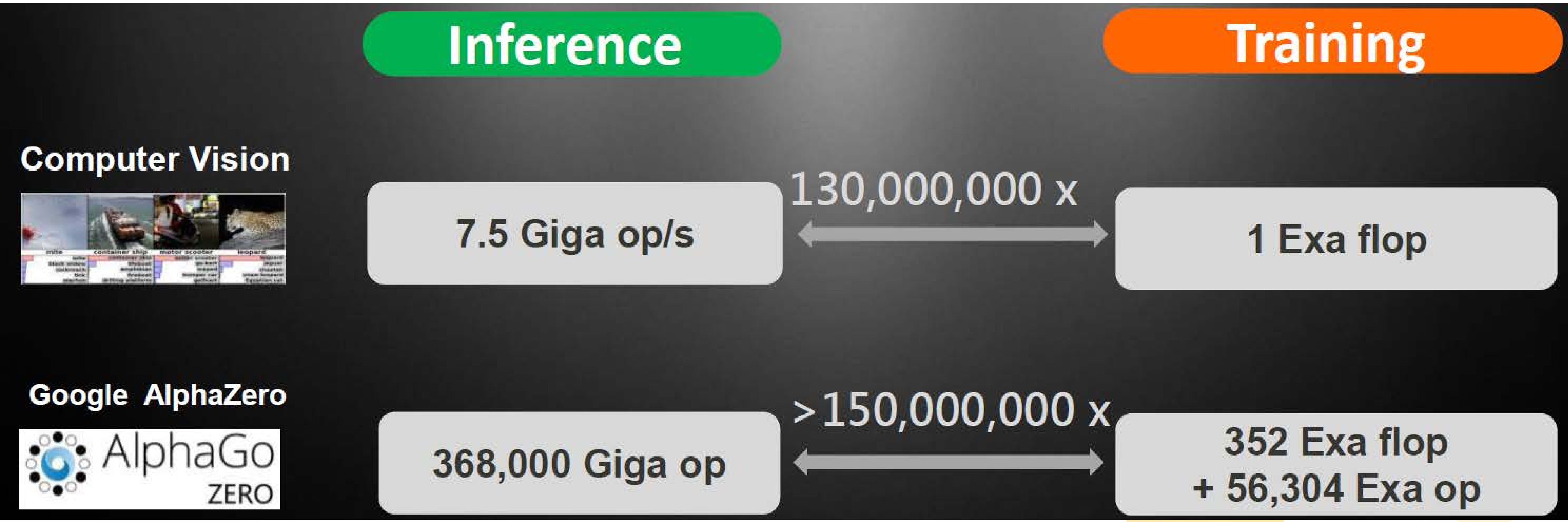
鼠類大腦
= ~100 mesocircuits 皮質中型迴路
= ~10000 neocortical columns 新皮質柱

鼠類 新皮質柱
=~10000 neurons

Source:
(1) Wired, 2013
(2) "Computer modelling: Brain in a box", Nature, 482, 456–458 , 23 Feb 2012
(3) Top500.org
Illistrated/Modified by Bor-Sung Liang, 2018.10

Inference vs. Training

- Training takes much more time than inference



	Inference		Training
Computer Vision	7.5 Giga op/s	130,000,000 x	1 Exa flop
Google AlphaZero	368,000 Giga op	>150,000,000 x	352 Exa flop + 56,304 Exa op

AlphaGo ZERO

Training/Inference for Computer Vision

- training dataset: ImageNet
- ResNet-50

Inference		Training		
Smartphone AP	**Hardware**	M40 GPU	nVidia DGX	nVidia DGX x 32
		M40 GPU x 1	P100 GPU x 8	P100 GPU x 256
7.5 G ops x 30 frame = 210 G op/s	**Performance**	7 T flop/s	170 T flop/s	5,440 T flop/s
1/30 sec	**Time**	**336 hr (14 Days)**	**21 hr**	**1 hr**
$500	**Hardware Cost**	$3,000	$129,000	$ 4.1M

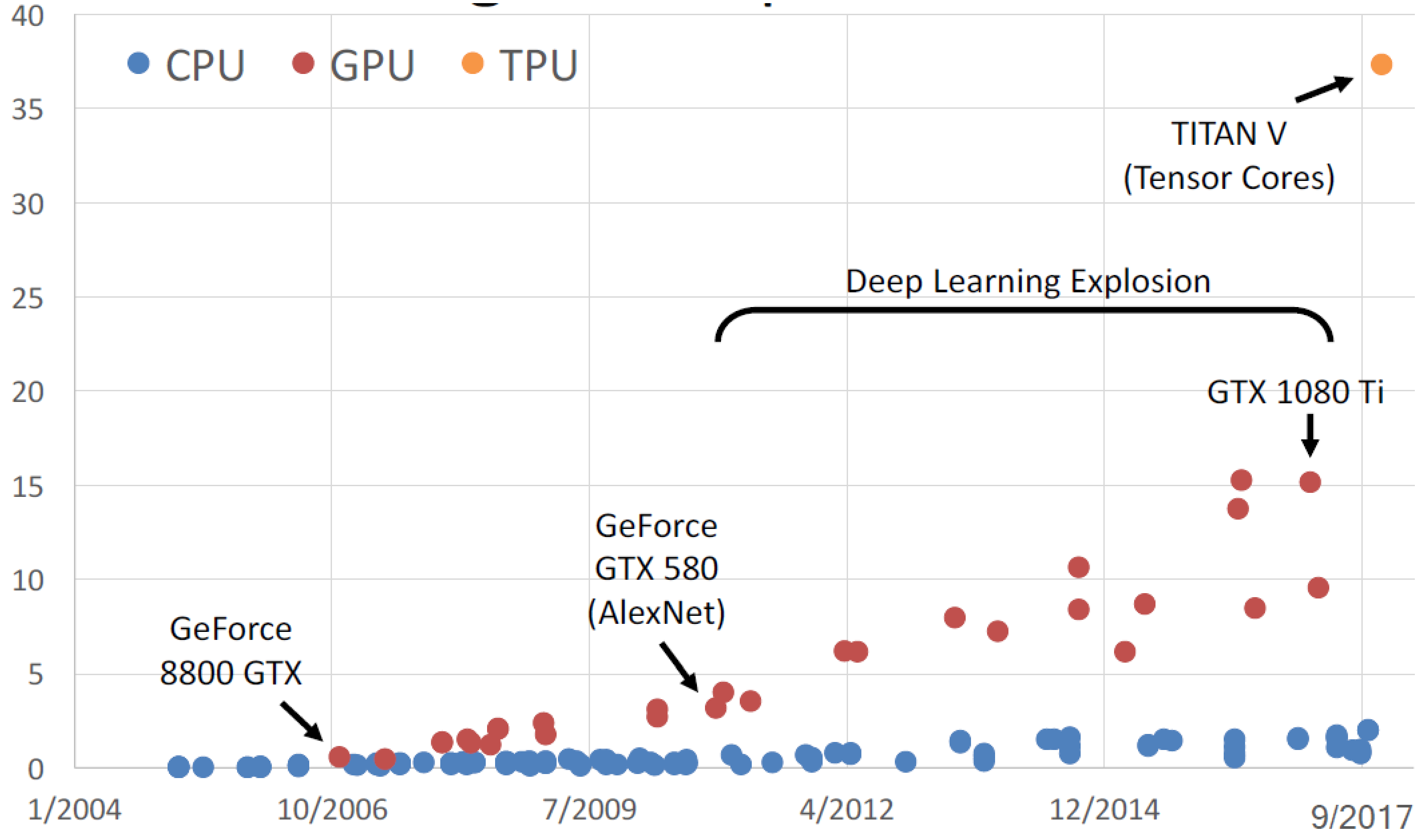
Training/Inference for GO Gaming

- DeepMind AlphaZero

Inference		Training	
Inference for Go		**Self-play games**	**NN Training**
TPU1	Config.	**TPU1**	**TPU2**
92 Top/s		92 Top/s	45TFlop/s
4	Comp. Resource *(estimated)*	**5000**	**64**
368 Top/s		**460,000 Top/s**	**2,880 Tflop/s**
0.2 sec	Time *(estimated)*	**34 hr**=122,400 sec **700,000 steps**	
268 T op	Computation *(estimated)*	56,304,000,000 Tops =56,304 Exa op	352,512,000 Tflops = 352 Exa flop

GFLOP/Dollars in CPU, GPU, TPU

- CPU: multi-core (~8) with 128-bit floating-point operations(FLOPs)/core
- GPU: many-core (~5,000) with 32-bit FLOPs / core
- TPU: ultra-many-core (~100,000) with 8/16-bit fixed-point operations / core



CPU vs. GPU vs. TPU

	Cores	Clock Speed	Speed (TFLOPs)	Power
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.2 GHz	~540 FP32	91 W
GPU (NVIDIA GTX 1080 Ti)	3584 CUDA	1.6 GHz	~11.4 FP32	250 W
GPU (NVIDIA GTX 2080 Ti)	4352 CUDA, 544 Tensor	1.55 GHz	~12 FP32	250 W
GPU (NVIDIA TITAN V)	5120 CUDA, 640 Tensor	1.5 GHz	~14 FP32 ~112 FP16	250 W
GPU (NVIDIA V100)	5120 CUDA, 640 Tensor	1.53 GHz	~16 FP32 ~125 for ML	300 W
TPU (Google Cloud TPU v3)	8 Tensor cores (8 x 128 x 128 x 2 = 262,144 FP16 MAC)	0.7 GHz	~91.8 for FP8 ~180 for ML	75 W

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

GPU: More cores, but each core is much slower and "dumber"; great for parallel tasks

TPU: Specialized hardware for deep learning

fastest supercomputers: 1. IBM Summit: 125,000 TFLOPs(10^{12}), 15,000,000 W !!!

2. 神威·太湖之光: 92,000 TFLOPs, 15,371,000 W

TensorFlow: Tensor Processing Units

Google Cloud TPU
= 180 TFLOPs!

Power: 40 W

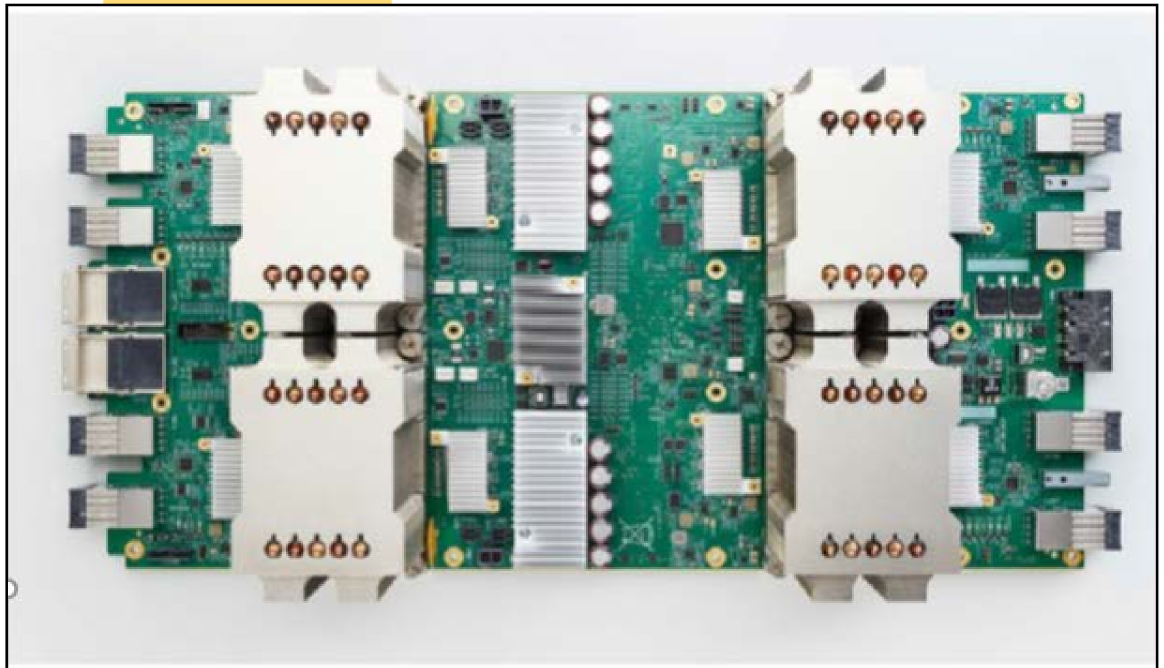
NVIDIA Tesla V100
= 125 TFLOPs

Power: 300 W

NVIDIA Tesla P100 = 11 TFLOPs
NVIDIA GTX 580 = 0.2 TFLOPs

But Power in mobile devices: <1 W !!!

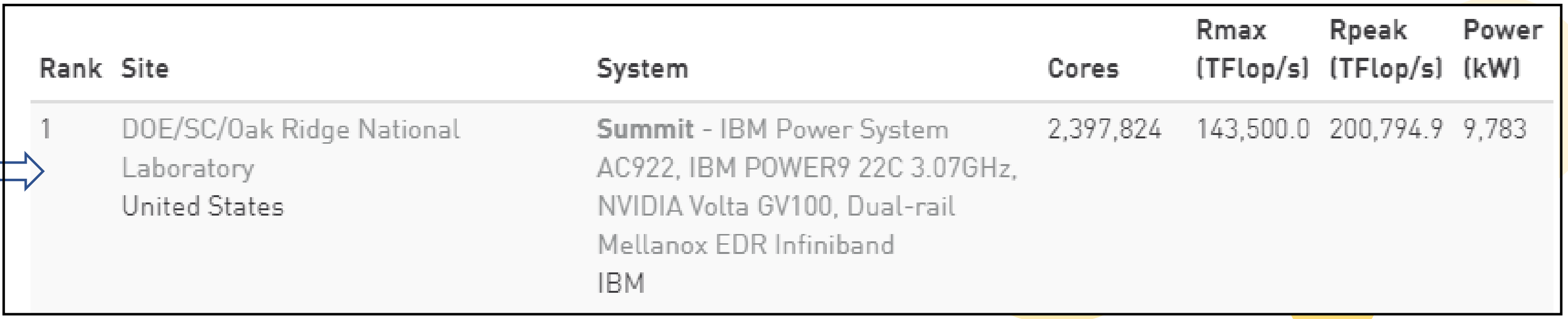
TPU in Google Cloud



Google Cloud TPU
= 180 TFLOPs of compute!

Google Cloud TPU Pod
= 64 Cloud TPUs
= 11.5 PFLOPs of compute!

the fastest supercomputer of the world in 2018/11

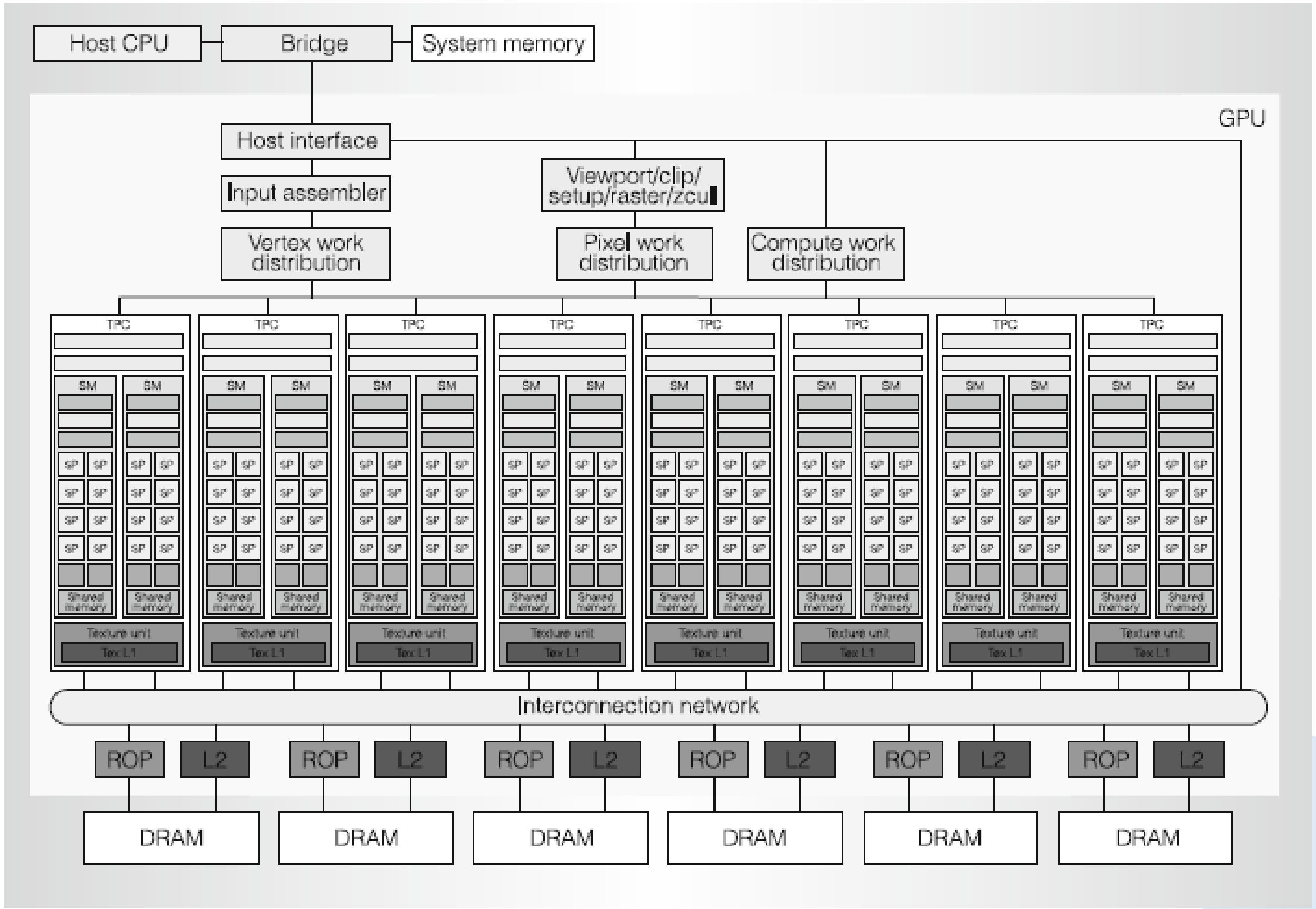


Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	DOE/SC/Oak Ridge National Laboratory United States	**Summit** - IBM Power System AC922, IBM POWER9 22C 3.07GHz, NVIDIA Volta GV100, Dual-rail Mellanox EDR Infiniband IBM	2,397,824	143,500.0	200,794.9	9,783

https://www.tensorflow.org/versions/master/programmers_guide/using_tpu

Nvidia Tesla GPU

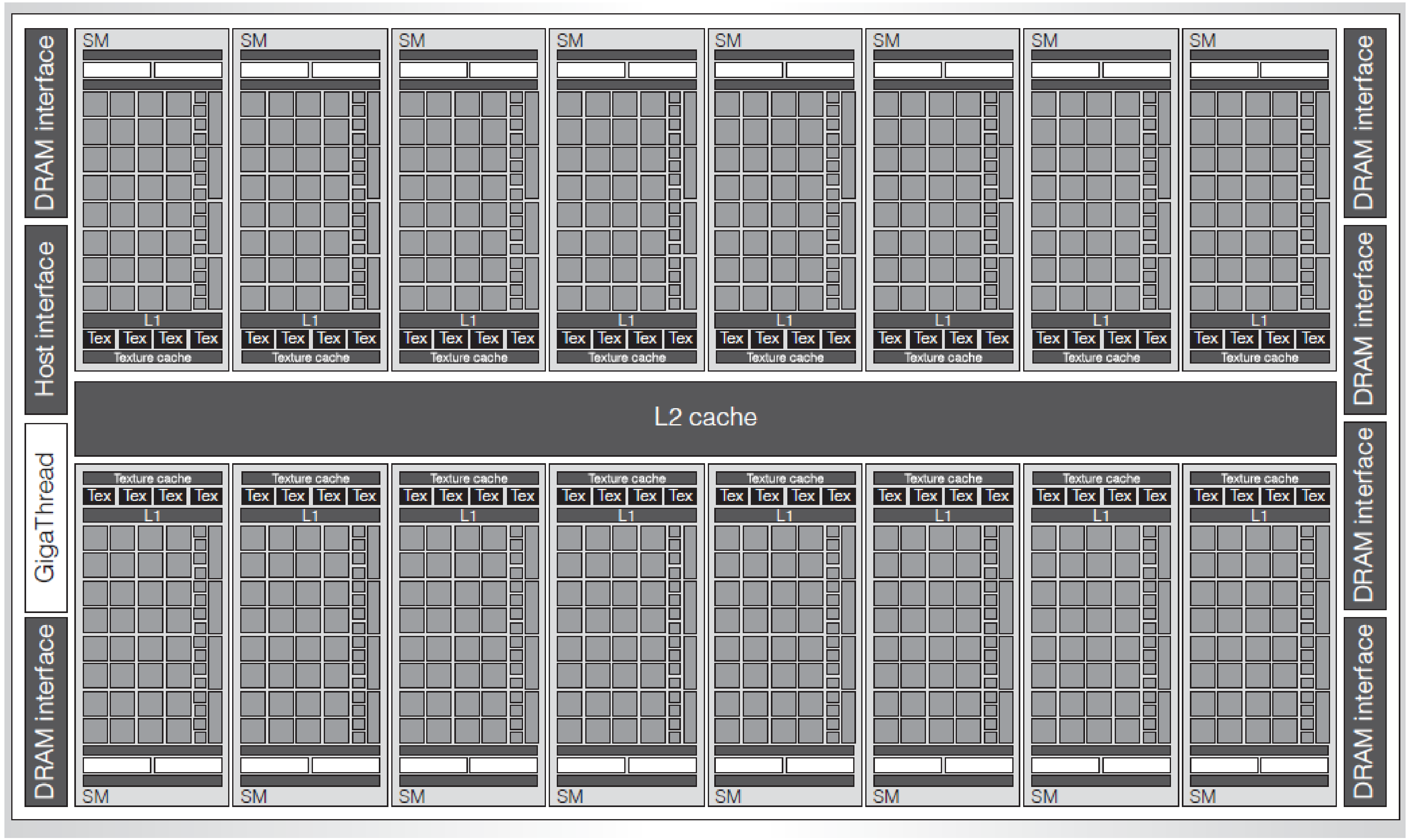
- Scalable unified architecture based on GeForce 8-series



NVIDIA Telsa, A Unified Graphics and Computing Architecture, *IEEE Micro*, Mar./Apr. 2008.
Patterson and Hennessy, *Computer Organization and Design, The Hardware/Software Interface*, 4th ed., Appendix A, 2009.

NVIDIA Fermi GPU

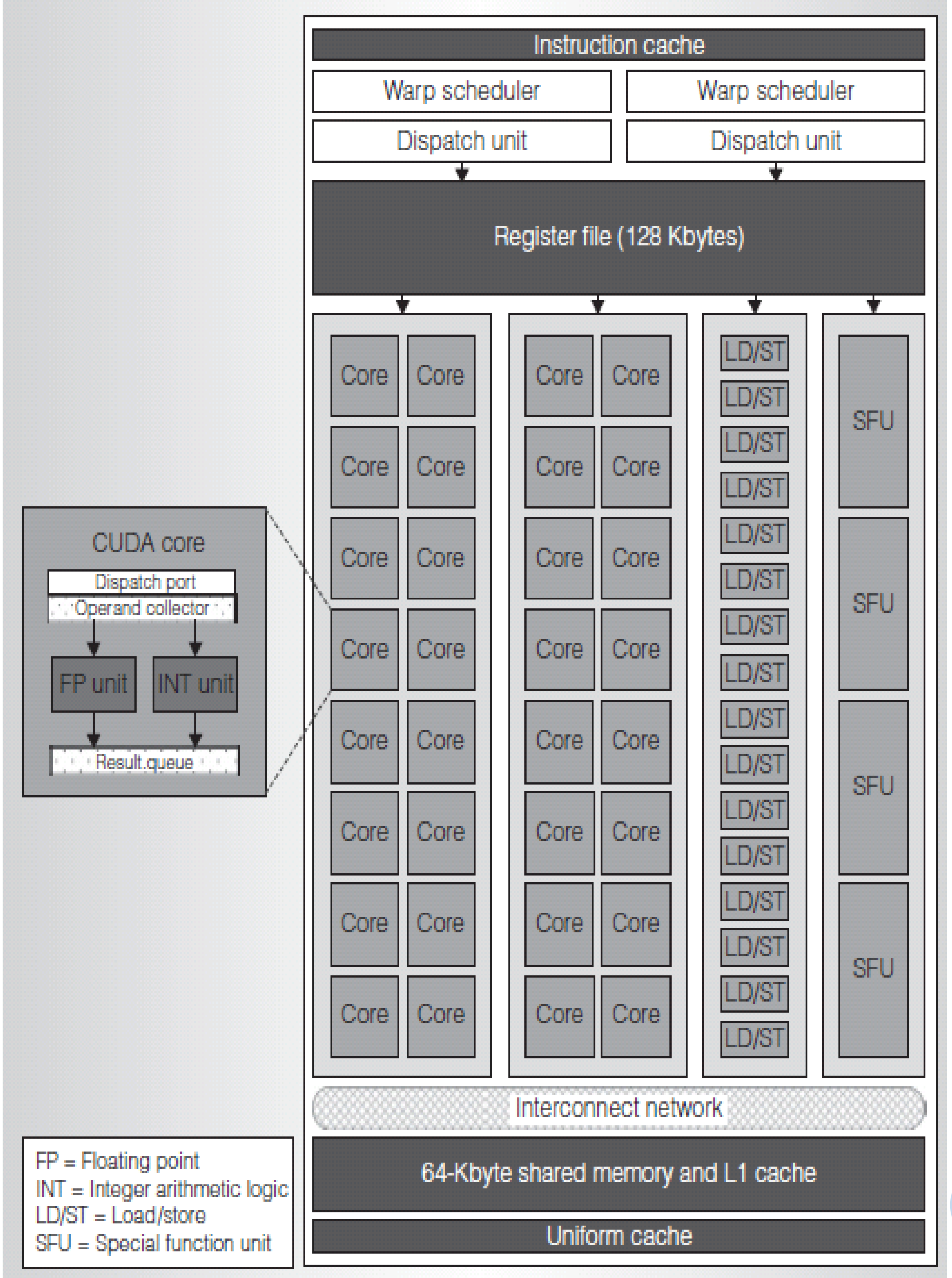
- 16 Streaming Multiprocessors (SM)
- 32 CUDA cores in each SM



The GPU Computing Era, *IEEE Micro*, Mar.-Apr. 2010.

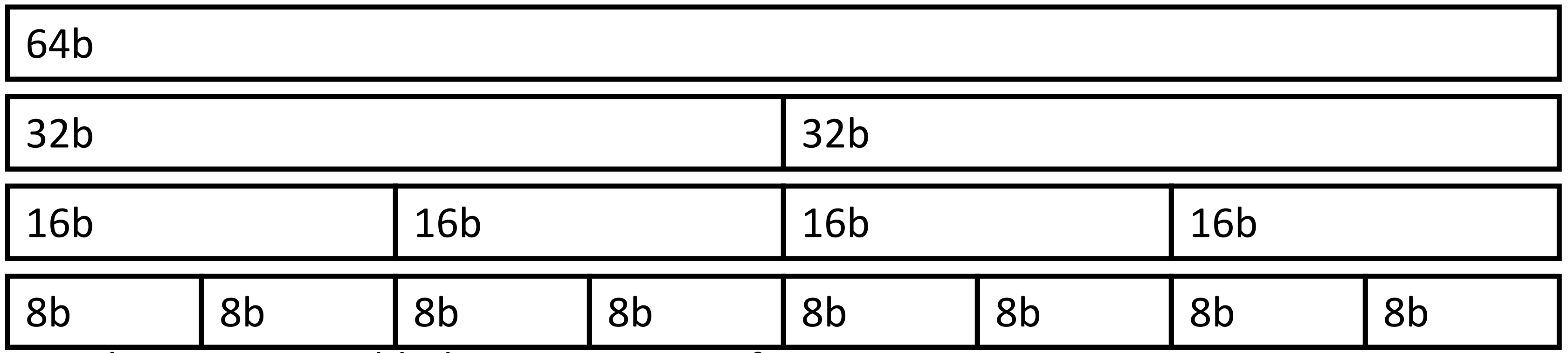
Fermi Streaming Multiprocessor

- 32 CUDA processor cores
- 16 load/store units
- 4 special function units
- 64KB shared memory/L1 cache
- 128KB register file
- Up to 1536 concurrent threads

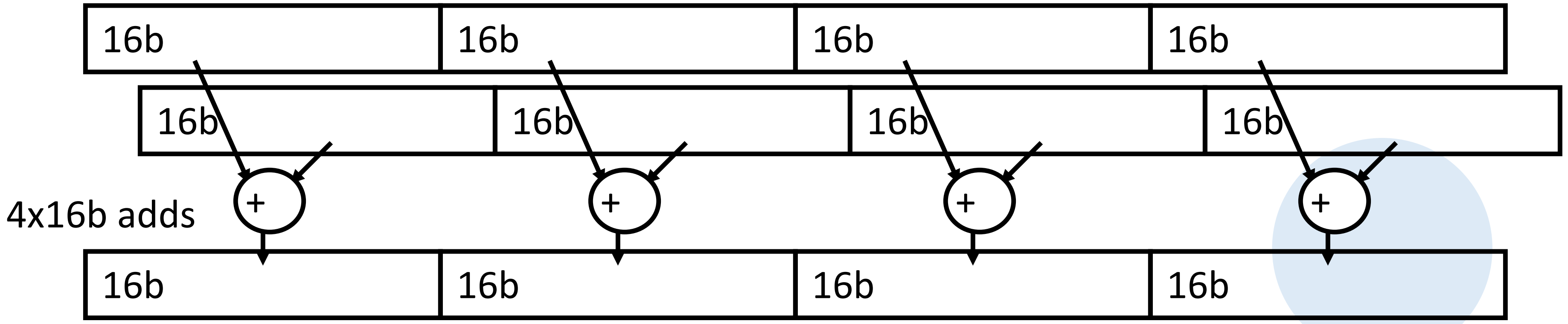


SIMD (Single Instruction Multiple Data)

Nvidia
Cudnn
INT8

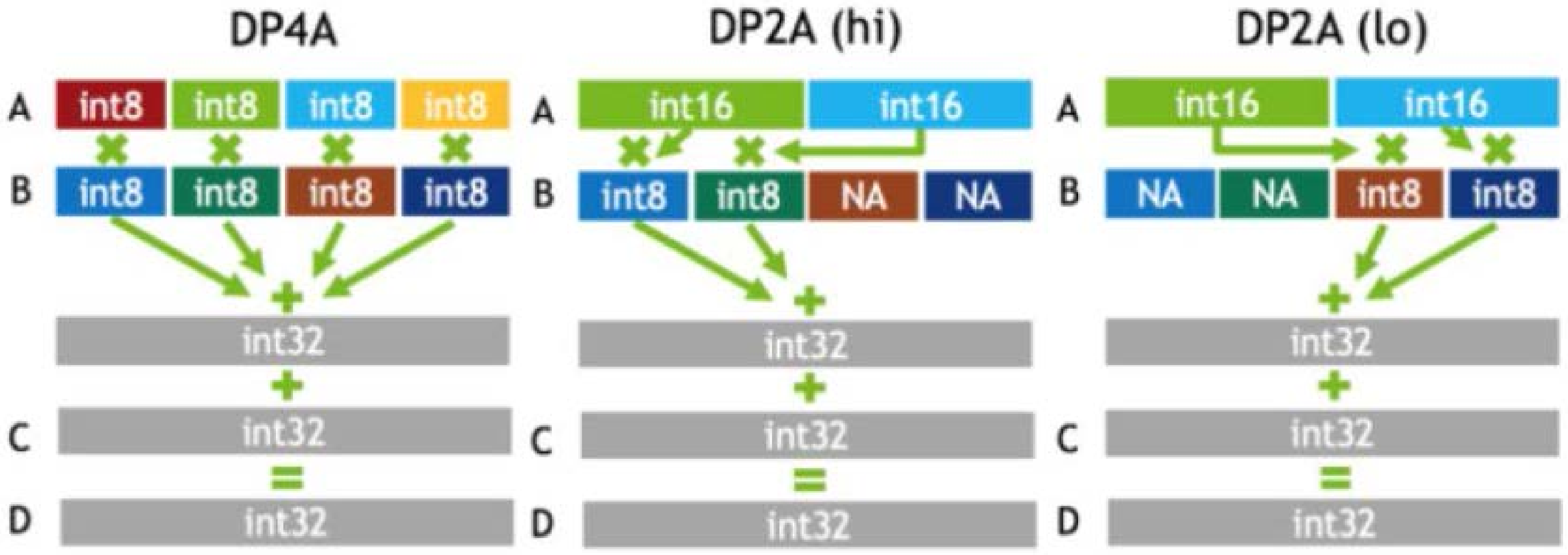


- Very short vectors added to existing ISAs for microprocessors
- Use existing 64-bit registers split into 2x32-b or 4x16-b or 8x8-b
  - Lincoln Labs TX-2 from 1957 had 36b datapath split into 2x18b or 4x9b
  - Newer designs have wider registers
    - 128b for PowerPC Altivec, Intel SSE2/3/4
    - 256b for Intel AVX
- Single instruction operates on all elements within register



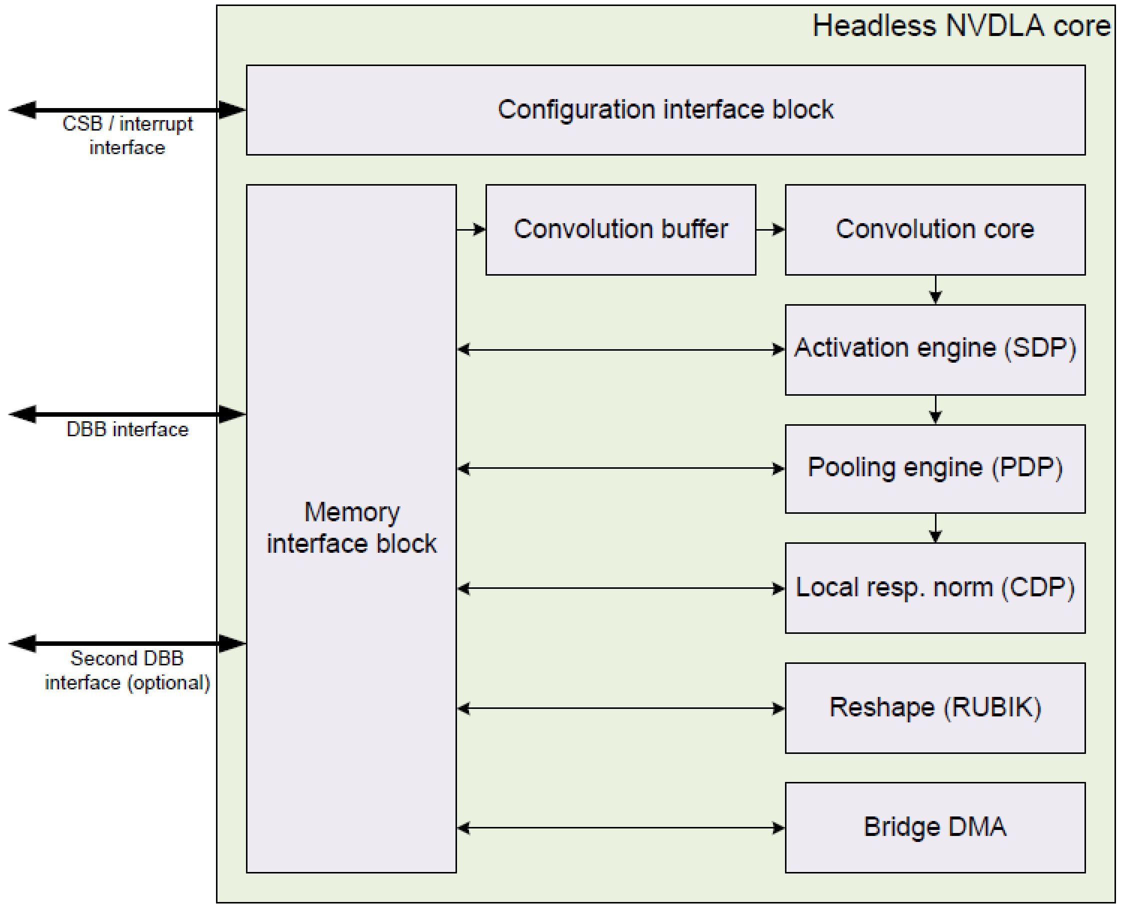
Nvidia Int8

- Low-bit accuracy (<16-b) in most deep learning applications
  - sometimes, even binary (1-b) is enough, e.g., BWN
- one Nvidia CUDA core has hardware of 32-b x 32-b MAC hardware
  - one 32-b x 32-b MAC (Multiply-ACcumulate)
  - two 16-b x 16-b MAC
  - four 8-b x 8-b MAC



NVidia Deep Learning Architecture (NVDLA)

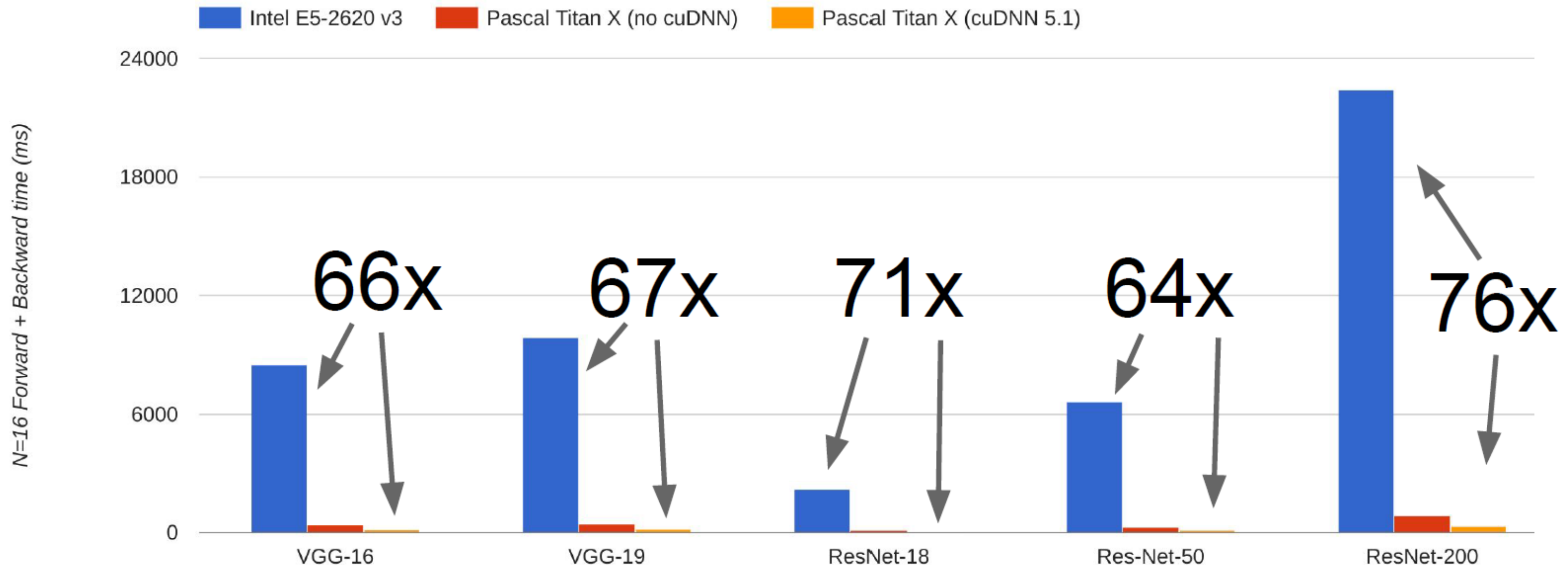
- **configurable fixed function inference HW as a co-processor for the management processor**
  - convolution, deconvolution, fully connected, activation, pooling, LRN
- **every block has a double buffer for its configuration register**
- **two operations modes**
  - independent mode
    - memory-to-memory operations
  - fused mode
    - some blocks assembled as a pipeline
    - bypassing round-trip through memory
    - block communication via smaller FIFOs



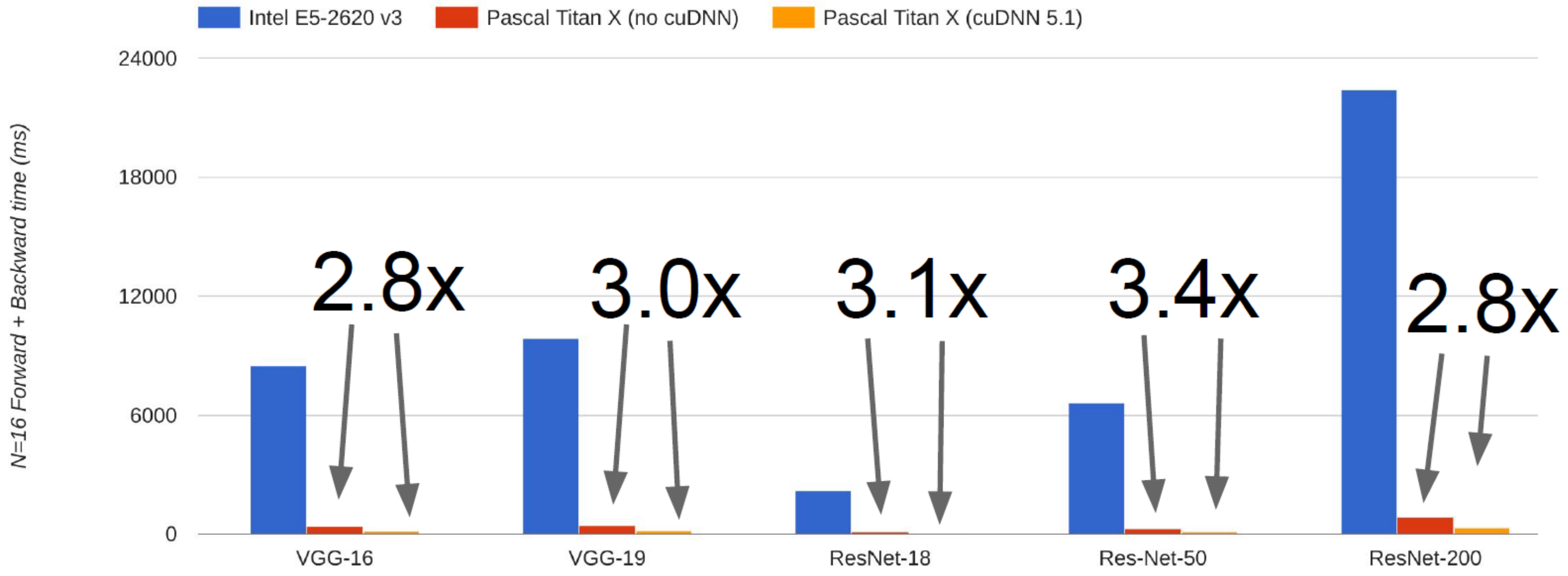
SDP: Single Data Point Processor (activation)
PDP: Planar Data Processor (pooling)
CDP: Cross-channel Data Processor (LRN), multi-plane operations
Rubik （俄羅斯方塊）: splitting or slicing, merging, contraction, reshape-transpose in deconvolution

CPU vs. GPU in practice

(CPU performance not well-optimized, a little unfair)



cuDNN much faster than "unoptimized" CUDA



Software Frameworks

- quick to develop and test new ideas
- automatically compute gradients
- efficiently run on GPU (wrap cuDNN, cuBLAS, etc.)

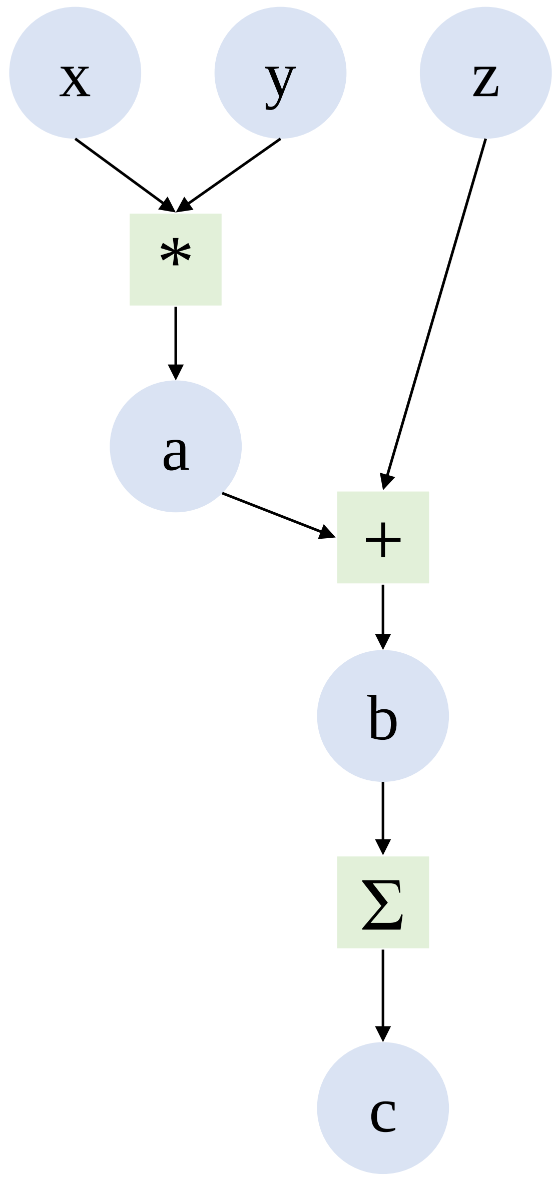
Caffe (UC Berkeley)	→	**Caffe2** (Facebook)	**PaddlePaddle** (Baidu)
			Chainer (Preferred Networks)
Torch (NYU / Facebook)	→	**PyTorch** (Facebook)	**MXNet** (Amazon) Developed by U Washinton, CMU, MIT, Hong Kong U, etc but main framework of choice at AWS
			CNTK (Microsoft)
Theano (U Montreal)	→	**TensorFlow** (Google)	**Deeplearning4j**

And others…

Computational Graphs in PyTorch

Numpy

```
import numpy as np
np.random.seed(0)

N, D = 3, 4

x = np.random.randn(N, D)
y = np.random.randn(N, D)
z = np.random.randn(N, D)

a = x * y
b = a + z
c = np.sum(b)

grad_c = 1.0
grad_b = grad_c * np.ones((N, D))
grad_a = grad_b.copy()
grad_z = grad_b.copy()
grad_x = grad_a * y
grad_y = grad_a * x
```

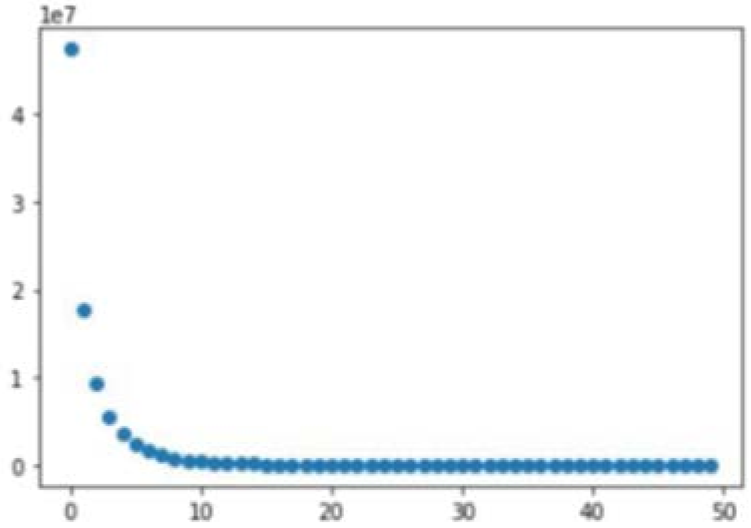


PyTorch

```
import torch

device = 'cuda:0'
N, D = 3, 4
x = torch.randn(N, D, requires_grad=True,
device=device)

y = torch.randn(N, D, device=device)
z = torch.randn(N, D, device=device)

a = x * y
b = a + z
c = torch.sum(b)

c.backward()
print(x.grad)
```

Trivial to run on GPU – just construct arrays on a different device!

TensorFlow Neural Net



Train the network:
Run the graph over and over,
use gradient to update weights

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))
w1 = tf.placeholder(tf.float32, shape=(N, H))
w2 = tf.placeholder(tf.float32, shape=(H, D))

h = tf.maximum(tf.matmul(x, w1), 0)
y_pred = tf.matmul(h, w2)
diff = y_pred - y
loss = tf.reduce_mean(tf.reduce_sum(diff ** 2, axis=1))
grad_w1, grad_w2 = tf.gradients(loss, [w1, w2])

with tf.Session() as sess:
        values = {x: np.random.randn(N, D),
                wl: np.random.randn(D, H),
                w2: np.random.randn(H, D),
                y: np.random.randn(N, D),}
        learning rate = le-5
        for t in range(50):
                out = sess.run([loss, grad_wl, grad_w2], feed dict=values)
                loss val, grad_wl_val, grad_w2_val = out
                values[wl] -= learning_rate * grad_wl_val
                values[w2] -= learning_rate * grad_w2_val
```

Keras: High-Level Wrapper

Keras is a layer on top of TensorFlow, makes common things easy to do .

(Used to be third-party, now merged into TensorFlow)

```
N, D, H = 64, 1000, 100
x = tf.placeholder(tf.float32, shape=(N, D))
y = tf.placeholder(tf.float32, shape=(N, D))

model = tf.keras.Sequential()
model.add(tf.keras.layers.Dense(H, input_shape=(D,), activation=tf.nn.relu))

model.add(tf.keras.layers.Dense(D))
y_pred = model(x)
loss = tf.losses.mean_squared_error(y_pred, y)

optimizer = tf.train.GradientDescentOptimizer(le0)
updates = optimizer.minimize(loss)
with tf.Session() as sess:
        sess.run(tf.global_variables_initializer())
        values = {x: np.random.randn(N, D),
                y: np.random.randn(N, D)}
        for t in range(50):
                loss_val, _ = sess.run([loss, updates], feed dict=values)
```

ASIC (Application Specific IC) DNN HW Accelerators

- Acceleration methods
  - GPU
  - DSP
  - ASIC
- Evaluation metrics
  - speed performance (GOPs, TOPs), power (mW), power efficiency (GOPs/W=GOP/J)
- Deep learning hardware accelerator for “edge” devices
  - power < 1W
- Several benchmarks
  - DianNao series (CAS中國科學院, 2014~2016)
  - Angel-Eye (Tsinghua北京清大, 2016~2018)
  - DNA, GNA, RNA, Thinker (Tsinghua北京清大, 2017~2019)
  - Eyeriss v1, v2 (MIT 麻省理工學院, 2017~2019)
  - EIE, ESE (Stanford史丹福大學, 2016~2017)
  - TPU (Google谷歌, 2017~2018)
  - DNPU, UNPU (KAIST南韓科大, 2017~2019)
  - …

Outlines

- Evaluation Metrics
- **DianNao, 2014 ASPLOS, 2015 ACM TOCS, China CAS**
- **DaDianNao, 2014 MICRO, China CAS**
- **ShiDianNao, 2015 ISCA, China CAS**
- **Cambricon-X, 2016 MICRO, China CAS**
- AngleEye, 2016 FPGA, 2018/1 TCAD, China Tsing-Hua Univ.
- EIE (Efficient Inference Engine), 2016 ISCA, Stanford Univ.
- ESE (Efficient Speech Recognition Engine), 2017 FPGA, Stanford Univ., DeePhi, Tsing-Hua Univ.
- Eyeriss, 2016 ISCA, 2017 JSSC, MIT
- Precision-Scalable (PS) ConvNet, 2017 JSSC, KU Leuven, Belgium
- ENVISION, 2017 ISSCC, KU Leuven, Belgium
- Origami, 2017 TCSVT, ETH Zurich, Switzerland
- Tensor Processing Unit (TPU), 2017 ISCA, Google
- ZeNA, 2018 IEEE Design & Test, Seoul National Univ., Korea
- DNPU (Deep Neural Processing Unit), 2017 ISSCC, 2017 ASSCC, H.-J. Yoo, KAIST
- UNPU (Unified Neural Processing Unit), 2018 ISSCC, H.-J. Yoo, KAIST
- DSIP (Deep learning Specific Instruction-set Processor), 2018/2, I.-C. Park, KAIST
- Binary CNN (BCNN), 2018/2 TVLSI, Zhongfeng Wang, Nanjing Univ., China
- Efficient Hardware Architecture for Deep CNN, 2018 TCAS-I, Zhongfeng Wang. Nanjing Univ., China
- DNA (Seep Neural Architecture), 2017/8 TVLSI, Shaojun Wei, Tsing-Hua Univ.

Comparison of DNN ASIC (1/2)

	Tech. (nm)	bits	f(MHz)	Power(mW)	speed (GOP/s)	pwr.eff. (GOP/s/W)	SRAM (KB)	Multi.	layer types	Parallelism Types(註一)	features
DianNao (2014)	65	16	1,000	0.485	452	932	44	256	CNN	OCP (no data reuse)	MAT
DaDianNao (2014)	28	16	606	16	5,580	349	36,000	4,096	CNN	OCP (no data reuse)	NoC, MAT
ShiDianNao (2015)	65	16	1,000	0.32	606	1,893	288	-	CNN	OCP (input data transfer inter PE)	2D, MAT
CambriconX (2016)	65	16	1,000	0.95	544	573	56	-	CNN FC	OCP (no data reuse)	1D, MAT
EIE (2016)	45	16	800	0.6	102	170	10,368	64	FC	OCP	1D, MAC
Eyeriss (2016, 2017)	65	16	200	0.28	60	23.1	83	336	CNN FC	WP 其餘平行不明	2D, MAC
Origami (2017)	65	12	500	0.5	196	437	43	196	CNN	WP、OCP	1D, MAT
TPU (2017)	28	8	700	40,000	92,000	2,300	28,000	65,536	CNN FC LSTM	平行不明	2D, MAC
DNPU (2017)	65	4~16	50~200	0.063	300@16-b	4,200	280	768@16-b	CNN FC LSTM	ICP、WP、OCP	2D, MAT
PS-ConvNet (2017)	40	4,8,12,16	204	0.287@16-b	74	270@16-b	148	256	CNN	WP、OCP	2D, MAC
DNA (2017)	65	16	200	0.48	194	406	280	1,024	CNN FC	ICP、WP、OCP	2D, MAC

Comparison of DNN ASIC (2/2)

	Tech. (nm)	bits	f(MHz)	Power(mW)	speed (GOP/s)	pwr.eff. (GOP/s/W)	SRAM (KB)	Multi.	layer types	Parallelism Types(註一)	features
FlexFlow (2017)	65	16	1,000	-	420	-	64	256	CNN	ICP、WP、OCP	2D, MAC, MAT
DSIP (2018)	65	16	250	0.153	16	105	140	64	CNN	ICP、WP、OCP	2D, MAC
UNPU (2018)	65	1~16	200	297	345.6@16-b	3,080	256	1,152	CNN FC LSTM	ICP、WP、OCP	2D, MAC, NoC
Think1 (2018)	65	8, 16	10~200	4~386	410	1,060~5,090	348	512	CNN FC LSTM	ICP、WP、OCP	2D, MAC
Think2 (2018)	28	1, 2, 4, 8, 16	20~400	3.4~20.8	410@(16, 1)-b	95,800@(16, 1)-b	224	32	CNN	ICP、OCP	MAT
GNA (2018)	28	8, 16	200	142	409.6	2,880	404	256	CNN DeCNN	ICP、WP、OCP、cross layer	2D, MAT
GPU Titan X	28	32f	1,075	210	5,991	29					
GPU K40	28	32f	560	250	1,783	7					
mGPU Tegra K1	28	32f	852	9	68	8					
CPU Intel Xeon	22	32f	2,900	130	97	1					
FPGA XC7Z045		16	150	9.6	137	14					
FPGA XC7Z020		8	214	3.5	84.3	24					

註一： OCP(output channel parallel) 、
ICP(input channel parallel)、
WP (window parallel)、
BP(batch parallel)

List of Some DNN ASIC

- Evaluation Metrics
- **DianNao, 2014 ASPLOS, 2015 ACM TOCS, China CAS**
- **DaDianNao, 2014 MICRO, China CAS**
- **ShiDianNao, 2015 ISCA, China CAS**
- **Cambricon-X, 2016 MICRO, China CAS**
- AngleEye, 2016 FPGA, 2018/1 TCAD, China Tsing-Hua Univ.
- EIE (Efficient Inference Engine), 2016 ISCA, Stanford Univ.
- ESE (Efficient Speech Recognition Engine), 2017 FPGA, Stanford Univ., DeePhi, Tsing-Hua Univ.
- Eyeriss, 2016 ISCA, 2017 JSSC, MIT
- Precision-Scalable (PS) ConvNet, 2017 JSSC, KU Leuven, Belgium
- ENVISION, 2017 ISSCC, KU Leuven, Belgium
- Origami, 2017 TCSVT, ETH Zurich, Switzerland
- Tensor Processing Unit (TPU), 2017 ISCA, Google
- ZeNA, 2018 IEEE Design & Test, Seoul National Univ., Korea
- DNPU (Deep Neural Processing Unit), 2017 ISSCC, 2017 ASSCC, H.-J. Yoo, KAIST
- UNPU (Unified Neural Processing Unit), 2018 ISSCC, H.-J. Yoo, KAIST
- DSIP (Deep learning Specific Instruction-set Processor), 2018/2, I.-C. Park, KAIST
- Binary CNN (BCNN), 2018/2 TVLSI, Zhongfeng Wang, Nanjing Univ., China
- Efficient Hardware Architecture for Deep CNN, 2018 TCAS-I, Zhongfeng Wang. Nanjing Univ., China
- DNA (Seep Neural Architecture), 2017/8 TVLSI, Shaojun Wei, Tsing-Hua Univ.