Efficient AI Model

Chia-Chi Tsai (蔡家齊)
cctsai@gs.ncku.edu.tw
AI System Lab
Department of Electrical Engineering
National Cheng Kung University

Outline

- Demands for Efficient Model
- Efficiency Metrics

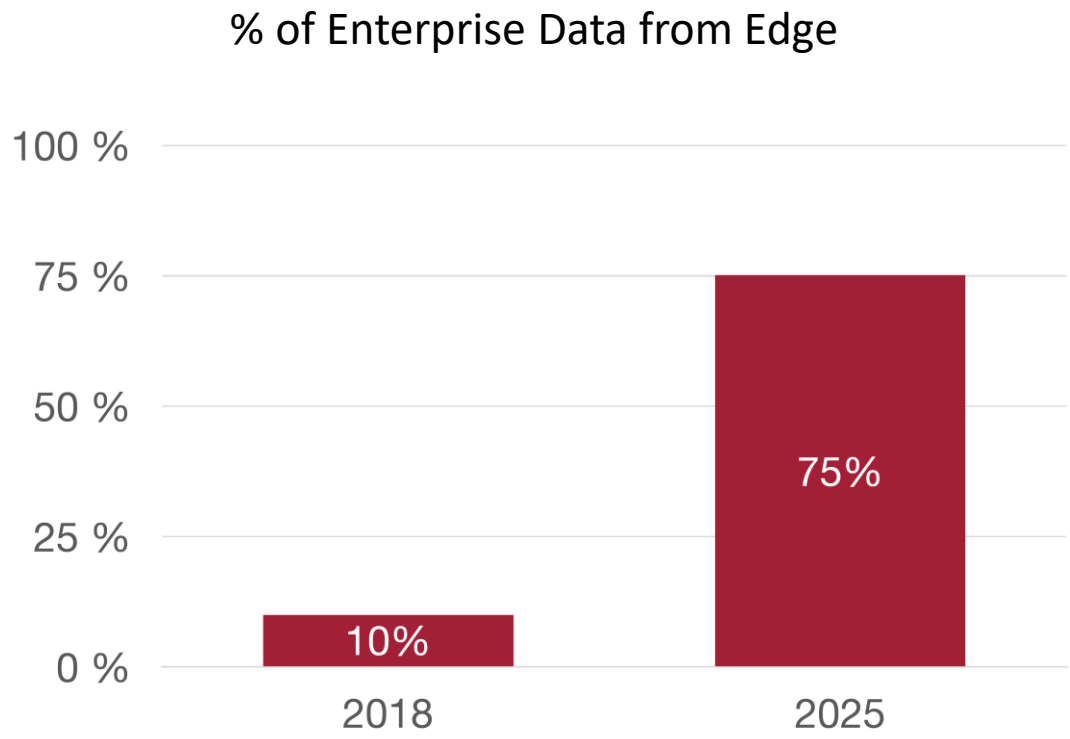
Outline

- Demands for Efficient Model
- Efficiency Metrics

AI is Quickly Coming to the Edge

- Privacy, latency and cost

- Number of Edge Devices 2021
  - 15 Billion mobile phones
  - 1.4 Billion cars
  - 770 Million security cameras
  - 15 Million robots



AI is Quickly Coming to the Edge

- Edge AI devices are different from cloud AI processors

	Cloud AI	Mobile AI	Tiny AI
Memory(Activation)	32G	4GB	320KB
Storage(Weights)	~TB/PB	256GB	1MB

Deep Learning for Image Classification

- DNNs achieve super-human classification accuracy on ImageNet

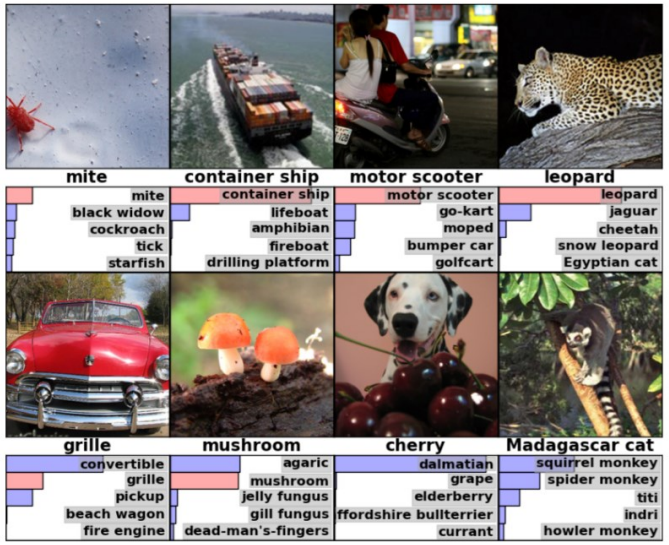
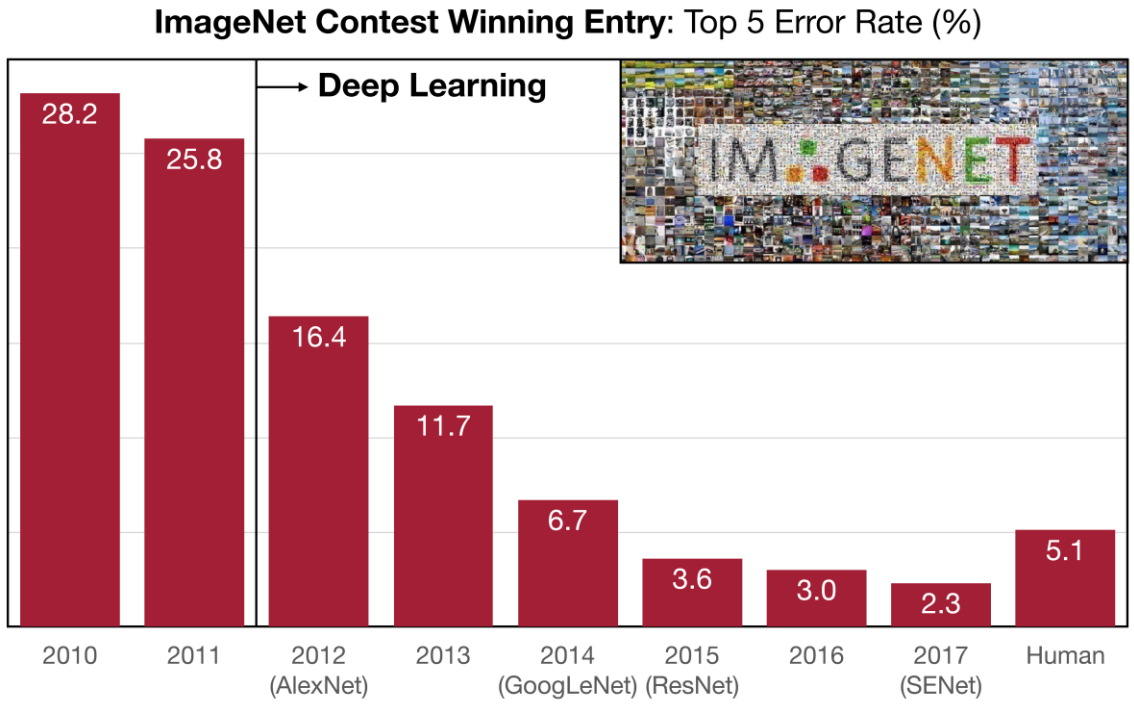
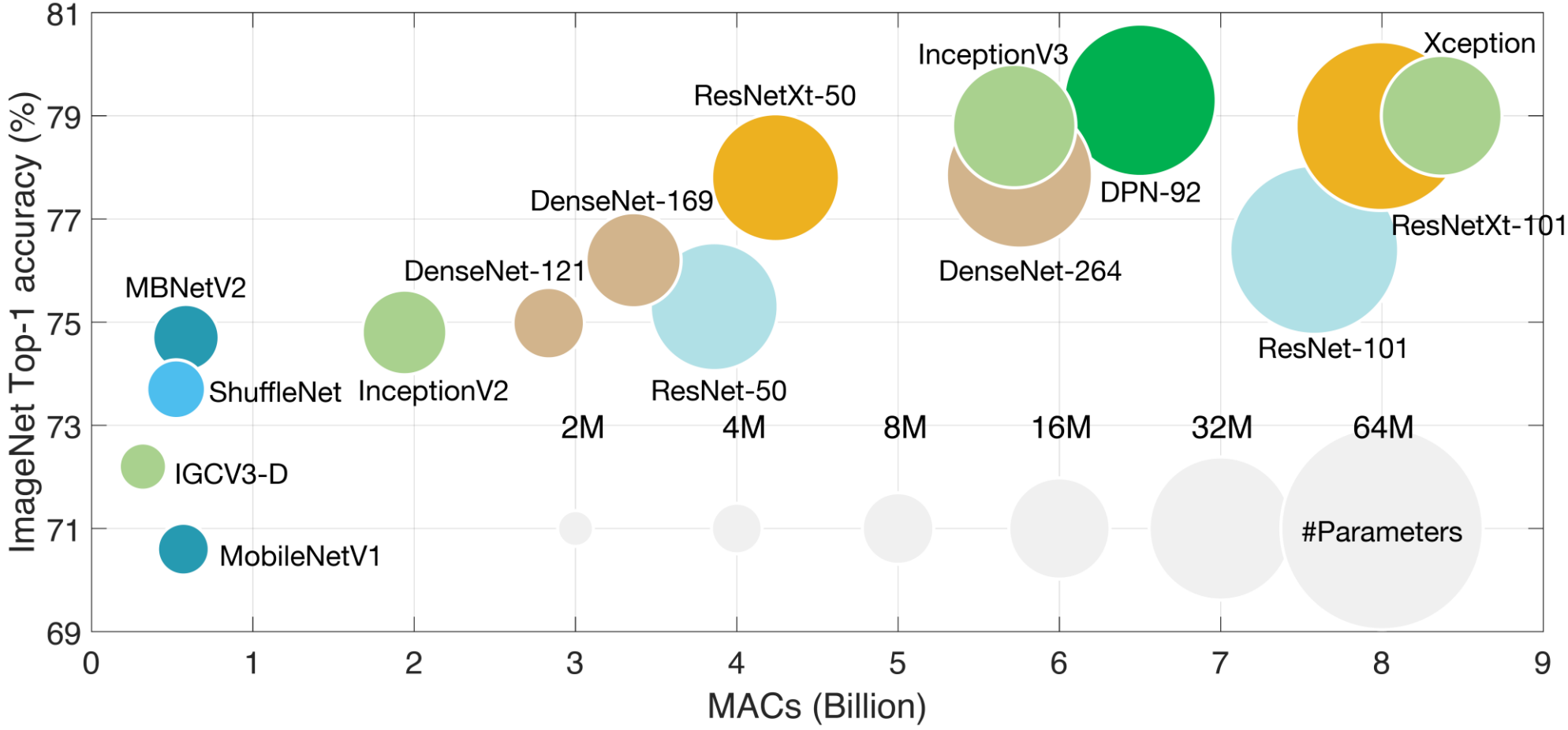


Image Classification



Deep Learning for Image Classification

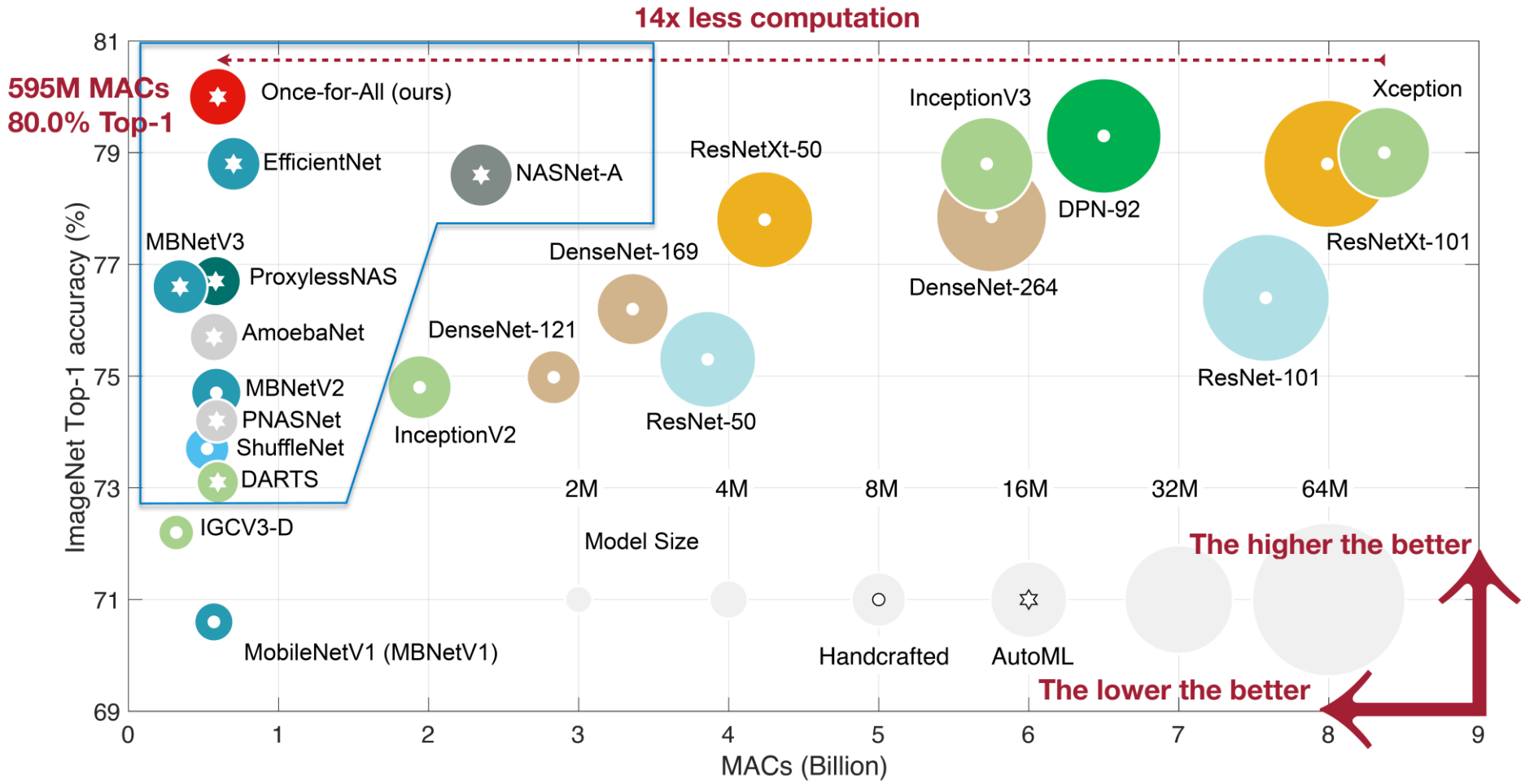
- Computational cost of DNNs is growing



L. Deng, G. Li, S. Han, L. Shi and Y. Xie, "Model Compression and Hardware Acceleration for Neural Networks: A Comprehensive Survey," in *Proceedings of the IEEE*, vol. 108, no. 4, pp. 485-532, April 2020, doi: 10.1109/JPROC.2020.2976475.

Efficient Deep Learning for Image Classification

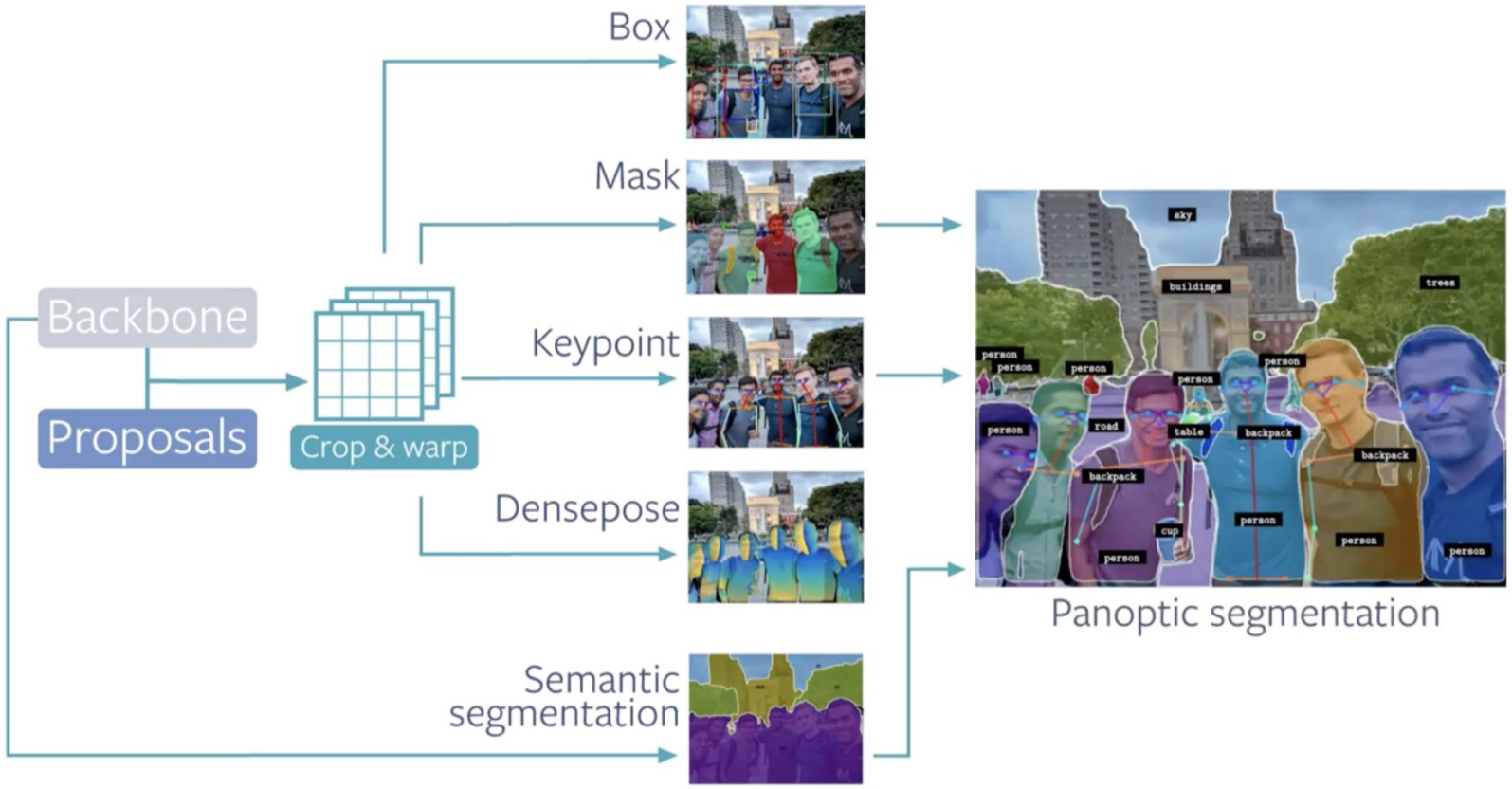
- Neural architecture search reduces the computational cost



H. Cai, C. Gan, T. Wang, Z. Zhang, and S. Han, 'Once for All: Train One Network and Specialize it for Efficient Deployment', in *International Conference on Learning Representations*, 2020.

Deep Learning for Image Recognition

- Detectron2: detection, segmentation, keypoint and pose estimation



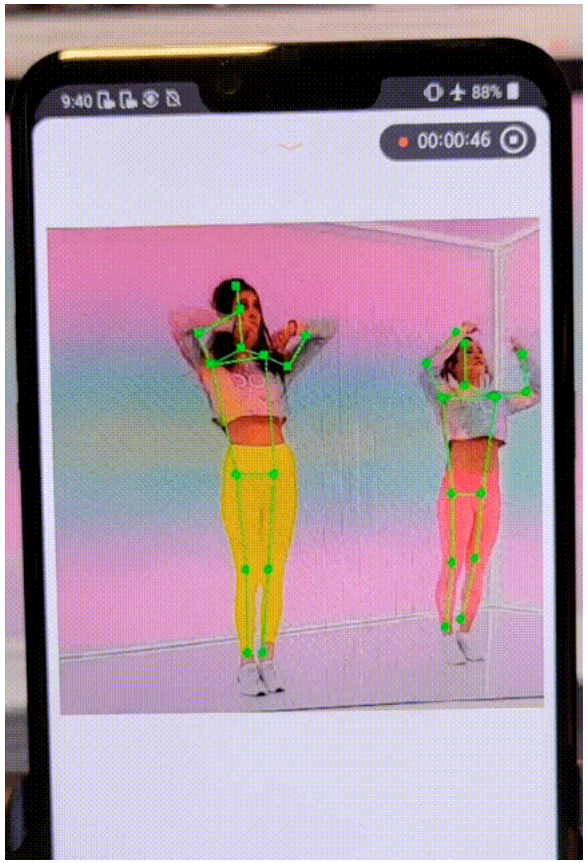
https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/

Efficient Deep Learning for Image Recognition

- Efficient deep learning enables daily life application on mobile phones

iPhone People Recognition

https://machinelearning.apple.com/research/recognizing-people-photos



On-Device Pose Estimation

Wang, Y., Li, M., Cai, H., Chen, W. M., & Han, S. (2022). Lite pose: Efficient architecture design for 2d human pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 13126-13136).

Deep Learning for Image Generation

- DALL·E 2 creates realistic images/art from a natural language description

Teddy bears, mixing sparkling chemicals as mad scientists, as a 1990s Saturday morning cartoon

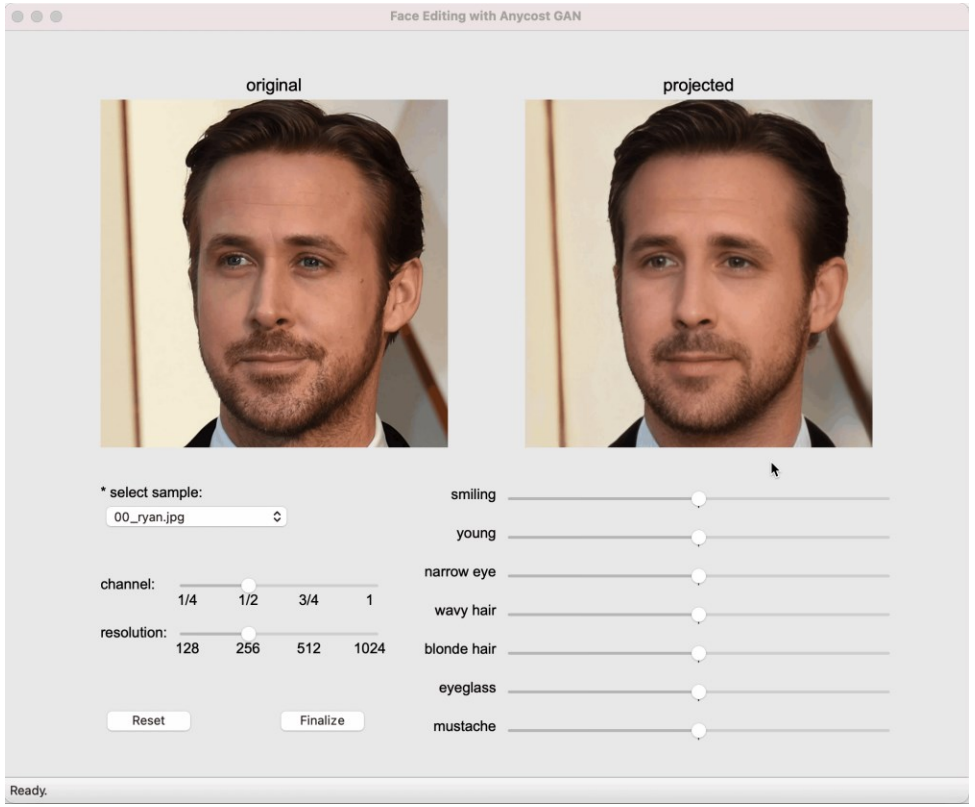
A bowl of soup, as a planet in the universe, as a 1960s poster

Compute: 256 GPUs for 2 weeks (>$100k)

https://openai.com/dall-e-2/

Efficient Deep Learning for Image Generation

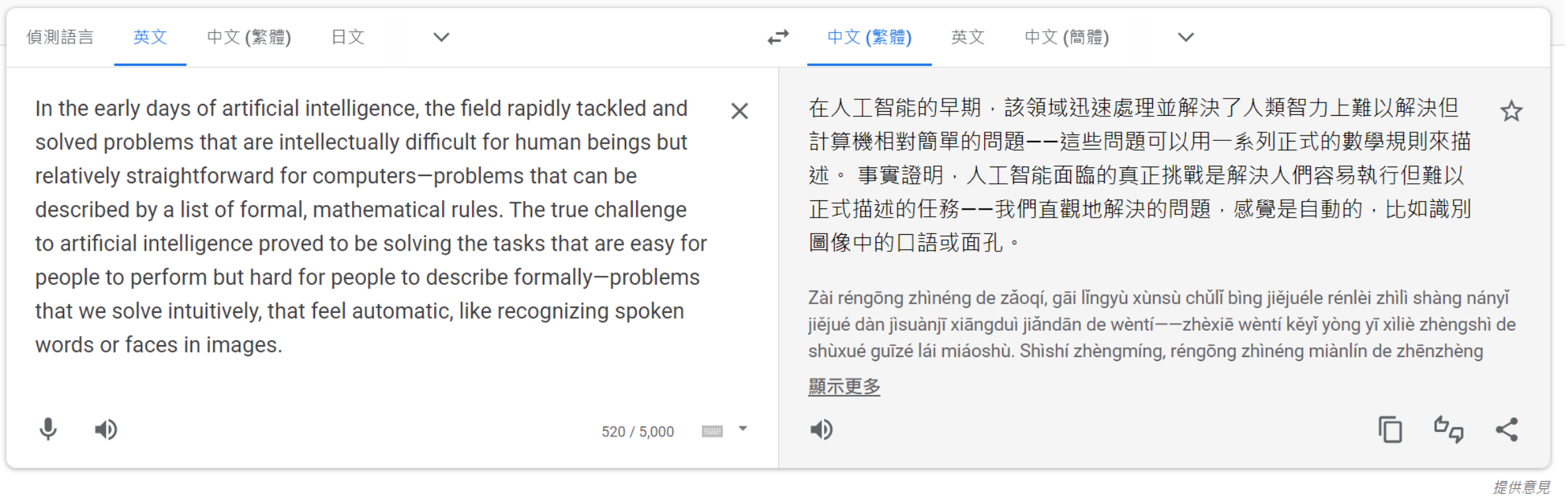
- AnycostGAN: interactive image synthesis and editing on a laptop



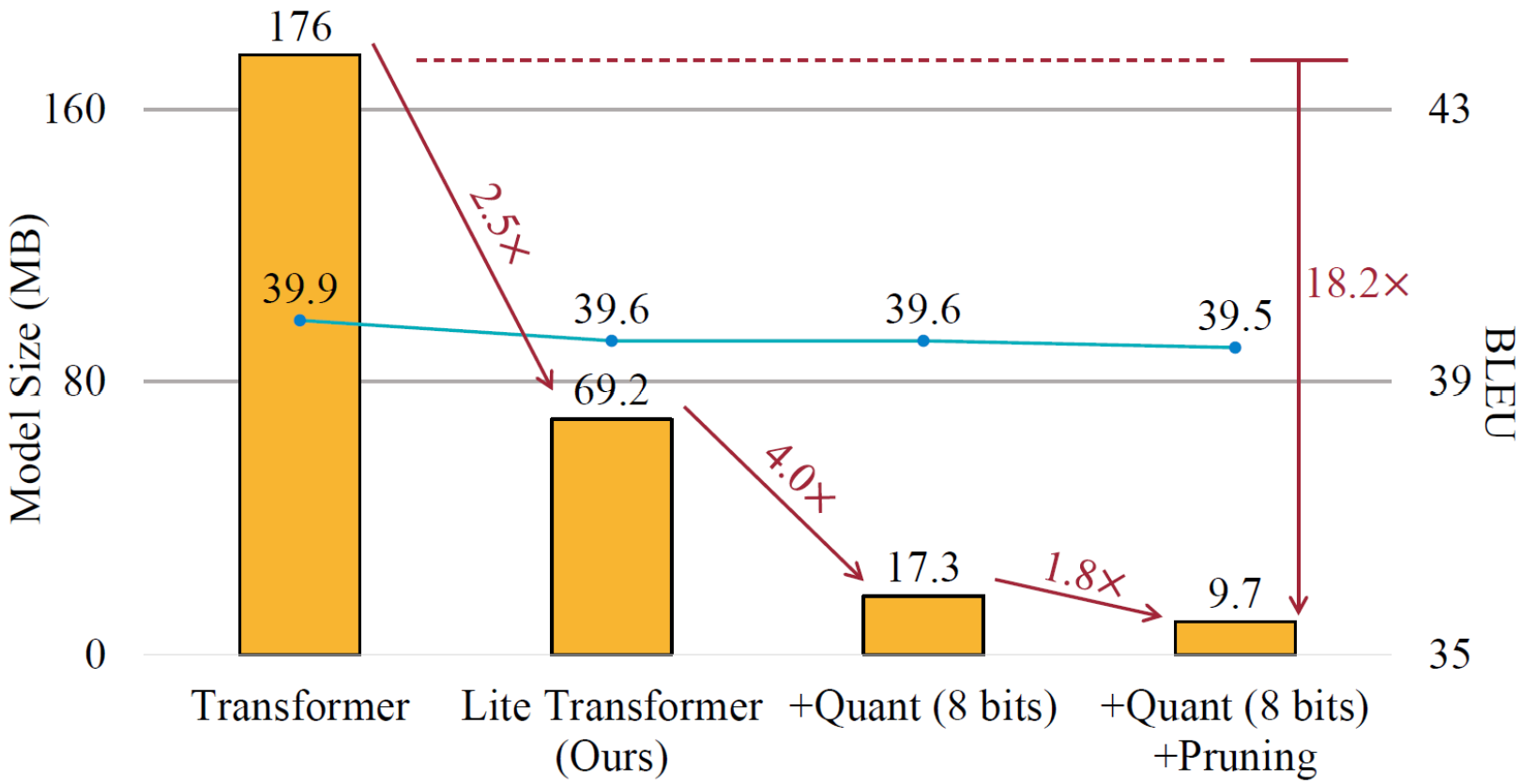
Lin, J., Zhang, R., Ganz, F., Han, S., & Zhu, J. Y. (2021). Anycost gans for interactive image synthesis and editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 14986-14996).

Deep Learning for Text Translation

- Neural machine translation bridges the language barrier



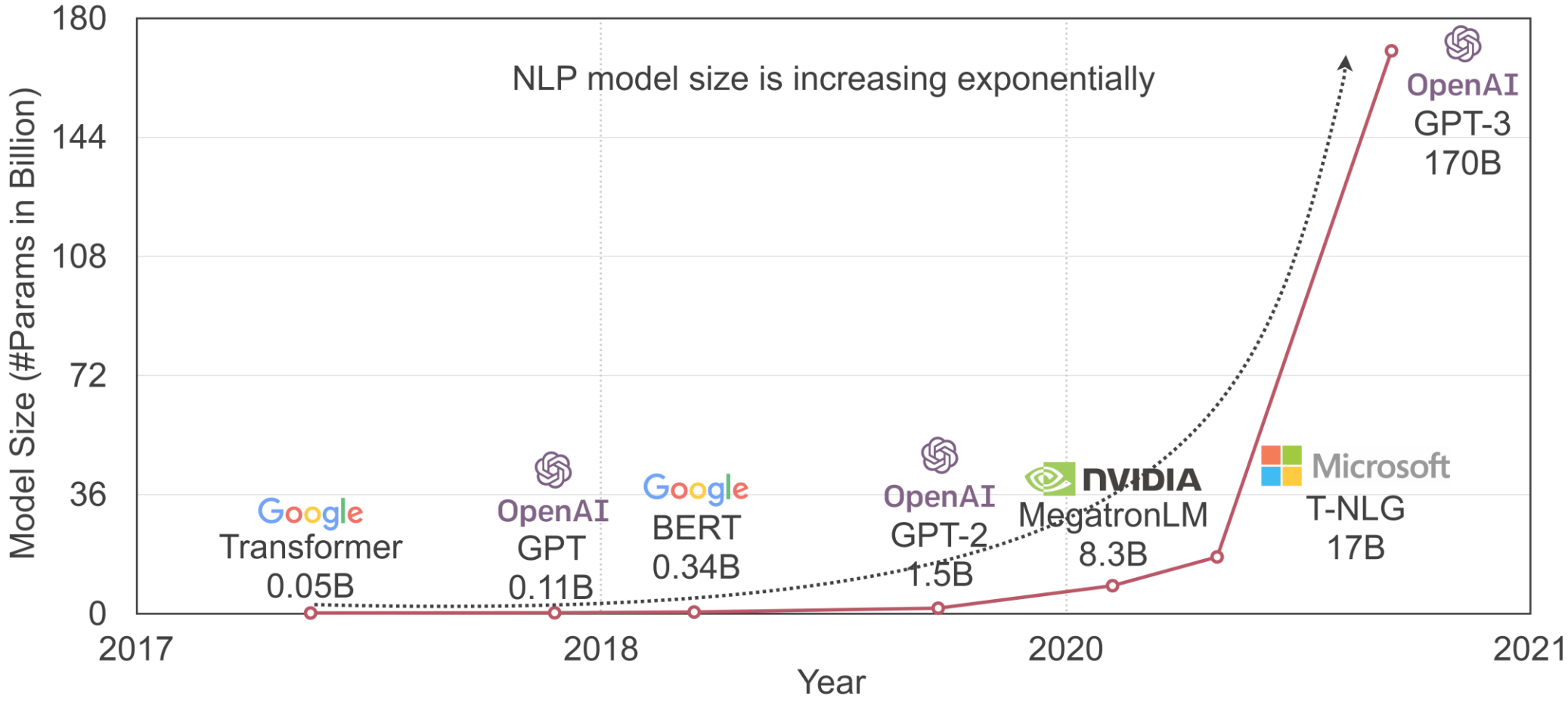
Efficient Deep Learning for Text Translation



Z. Wu*, Z. Liu*, J. Lin, Y. Lin, and S. Han, 'Lite Transformer with Long-Short Range Attention', in *International Conference on Learning Representations (ICLR)*, 2020.

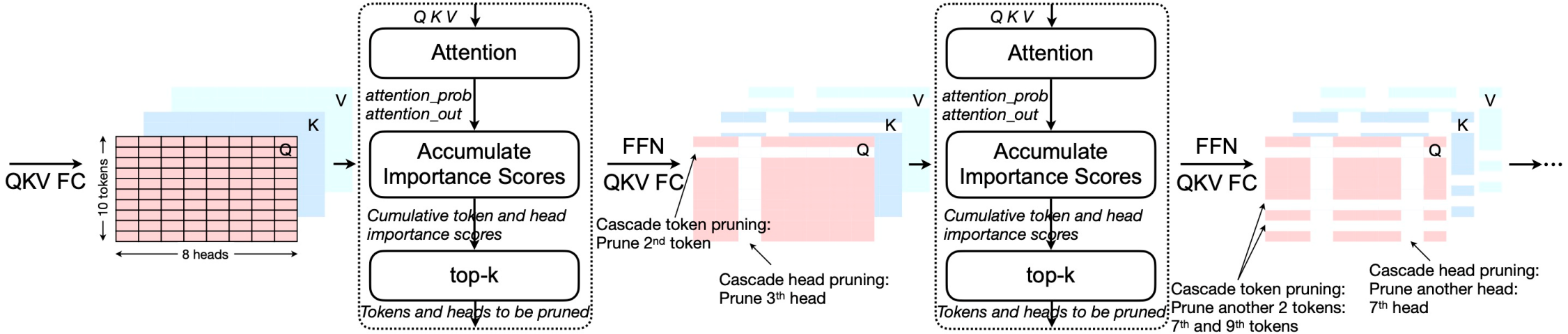
Deep Learning for Language Modeling

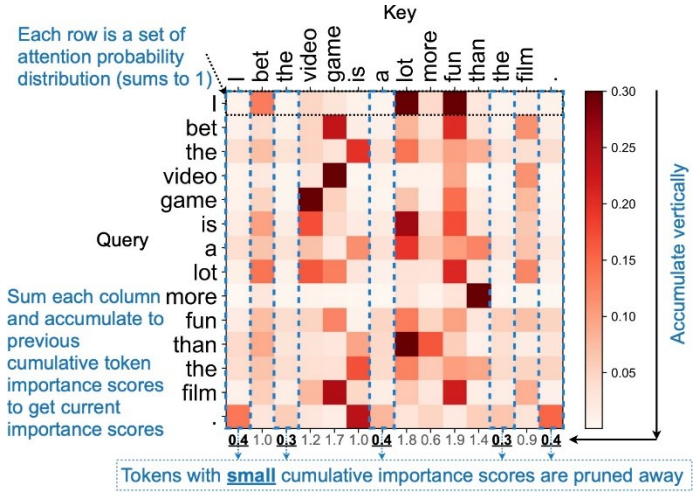
- Model size of language models is growing exponentially

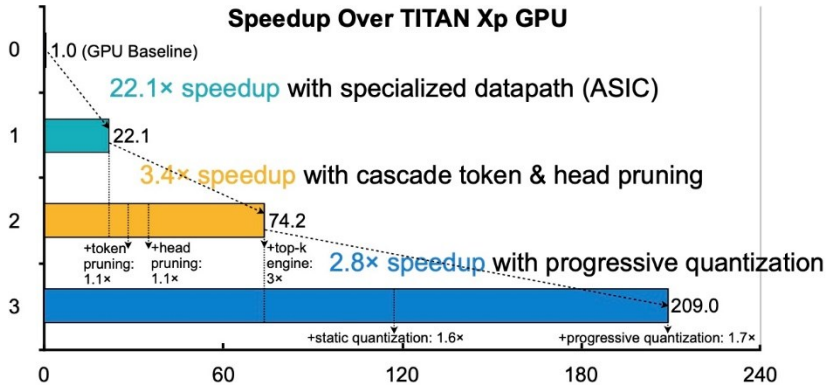


Efficient Deep Learning for Language Modeling

- SpAtten accelerates language models by pruning redundant tokens



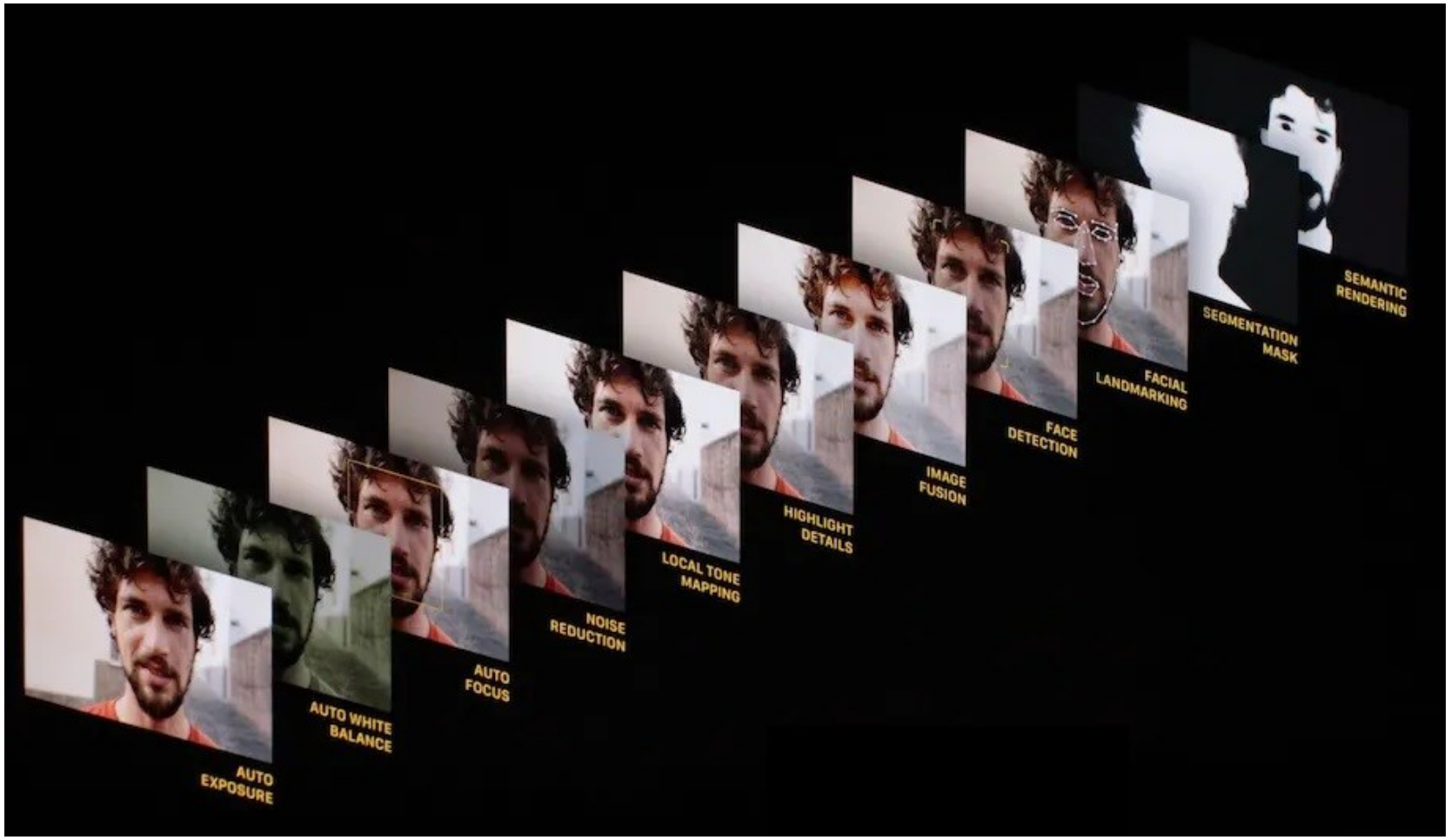




Wang, H., Zhang, Z., & Han, S. (2021, February). Spatten: Efficient sparse attention architecture with cascade token and head pruning. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 97-110). IEEE.

Deep Learning for Computational Photography

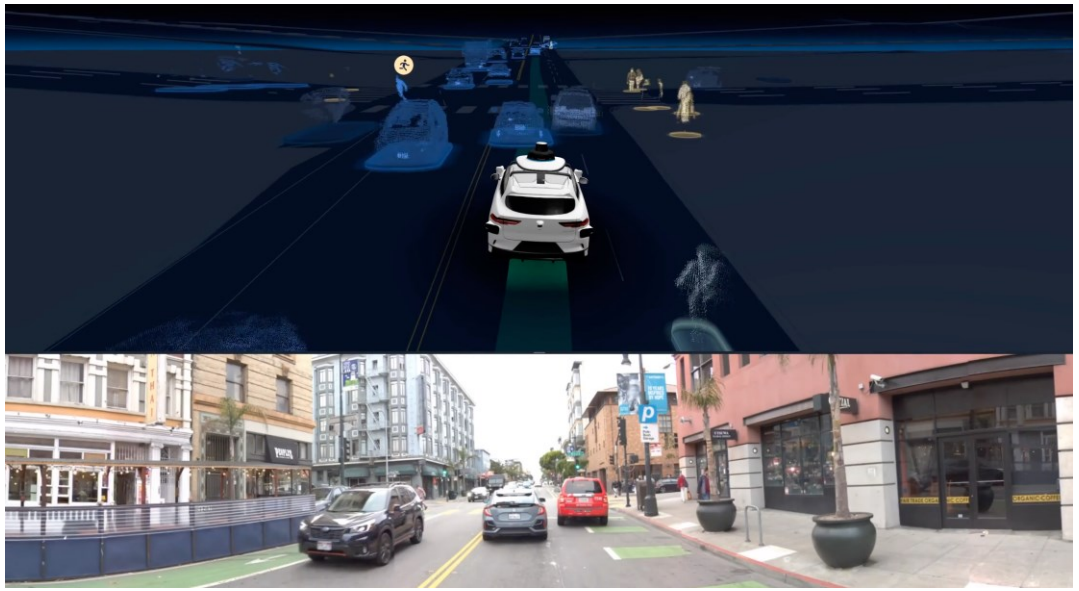
- Computational photography helps capture more sensational shots



https://www.theverge.com/2019/9/10/20850694/apple-iphone-11-pro-max-camera-specs-features-triple-three-slofies

Deep Learning for Autonomous Driving

- Deep learning helps machine perceive the surrounding environment



Waymo Driver

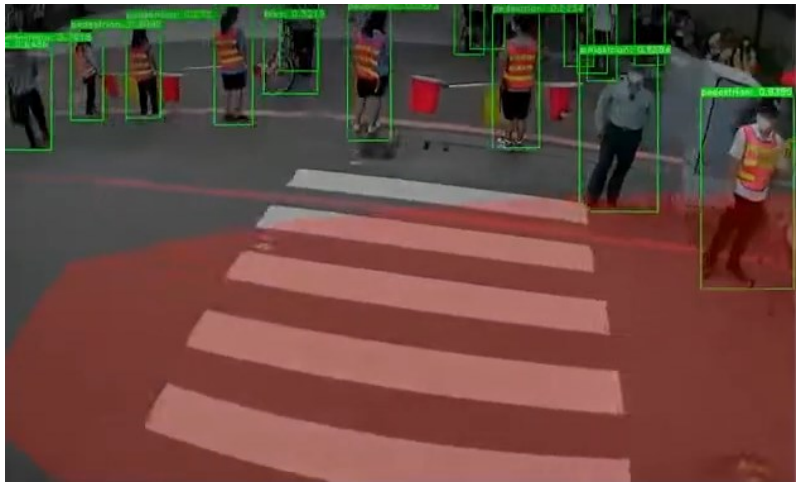
A whole trunk of workstation

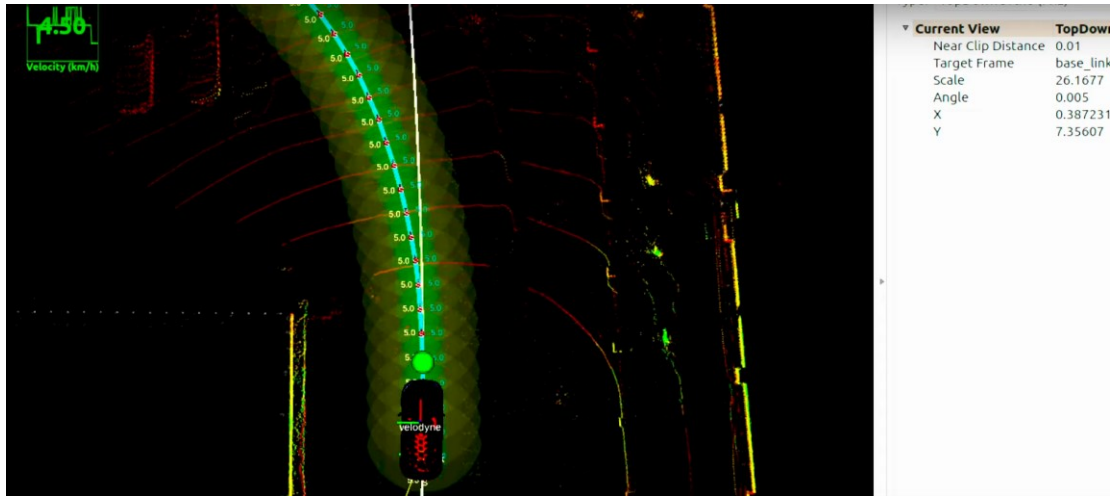
Efficient Deep Learning for Autonomous Driving

- Efficient Lidar/Camera based perception/localization/control

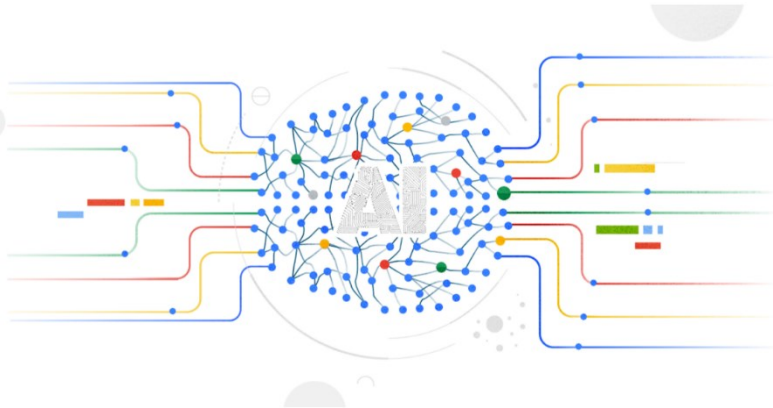
iVSLab

Efficient Deep Learning for Autonomous Driving



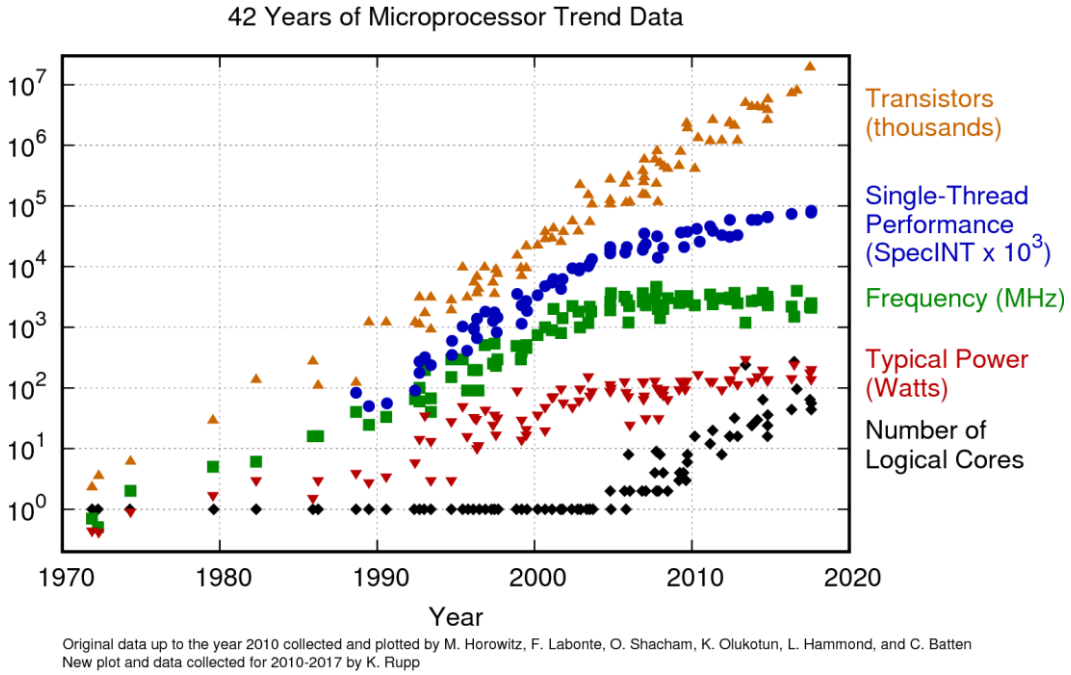


Three Pillars of Deep Learning

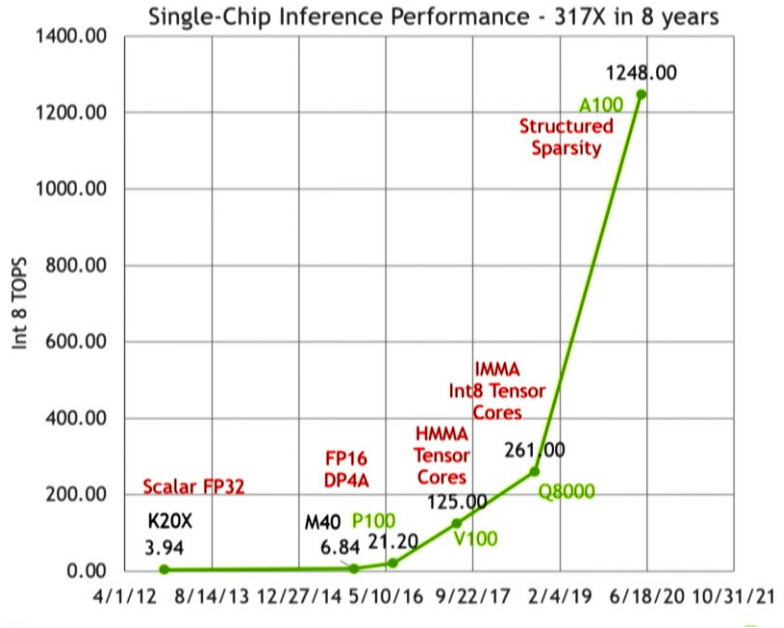


Architectural Support

- Architectural support for quantization/pruning brings tremendous improvement
- Quantization: $FP32 \rightarrow FP16 \rightarrow INT8 \rightarrow INT4 \rightarrow Binary$
- Pruning: $dense \rightarrow sparse$



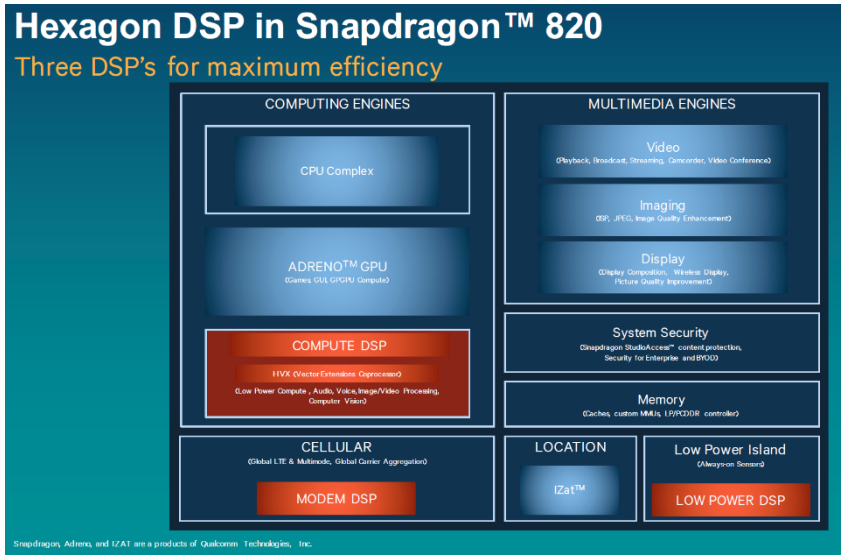
source: M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond, C. Batten, K. Rupp

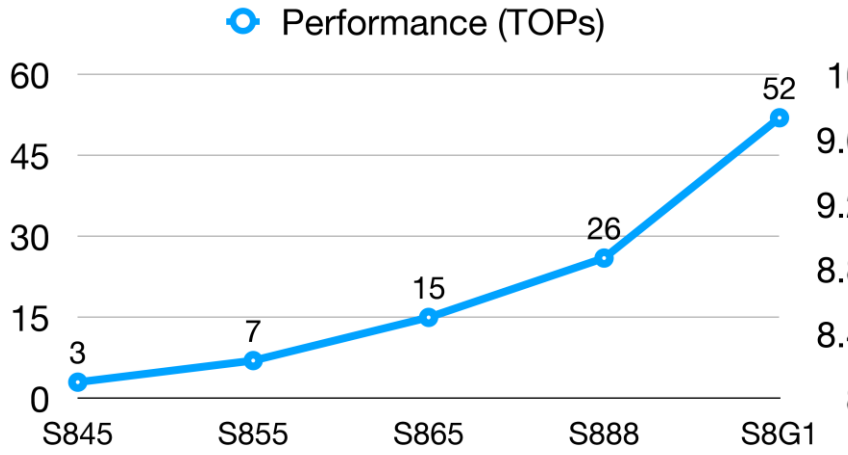


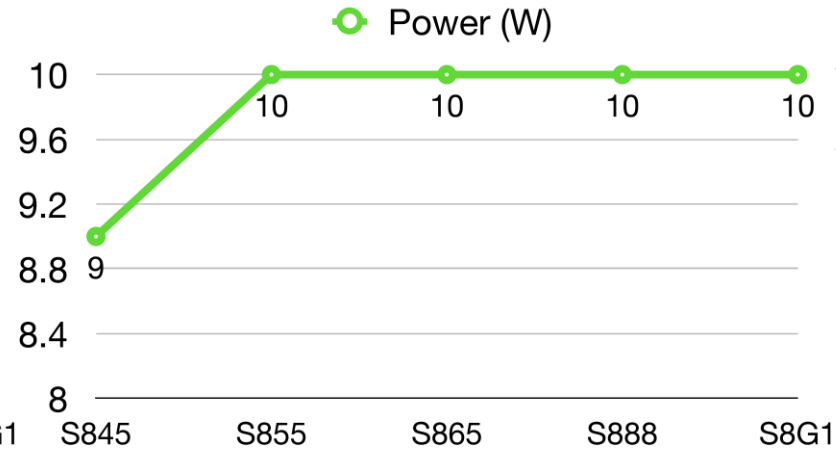
GPUs, Machine Learning, and EDA — Bill Dally

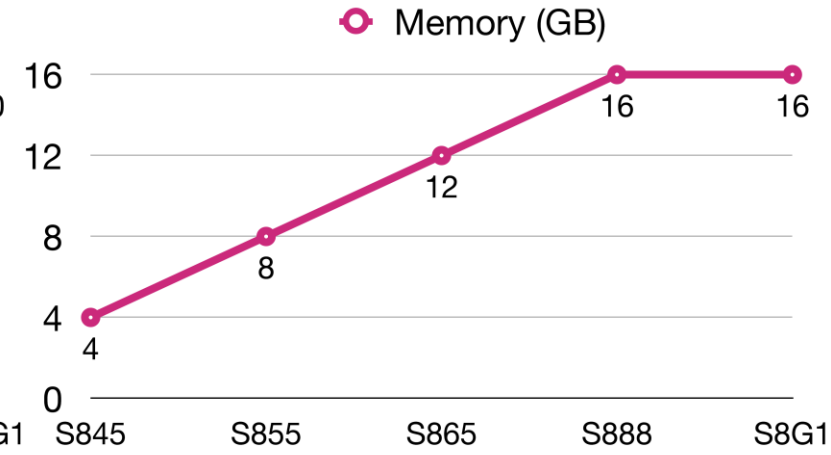
Edge AI Hardware

- Qualcomm Hexagon DSP
  - A family of digital signal processor (DSP) products by Qualcomm.
  - Designed to deliver performance with low power over a variety of applications



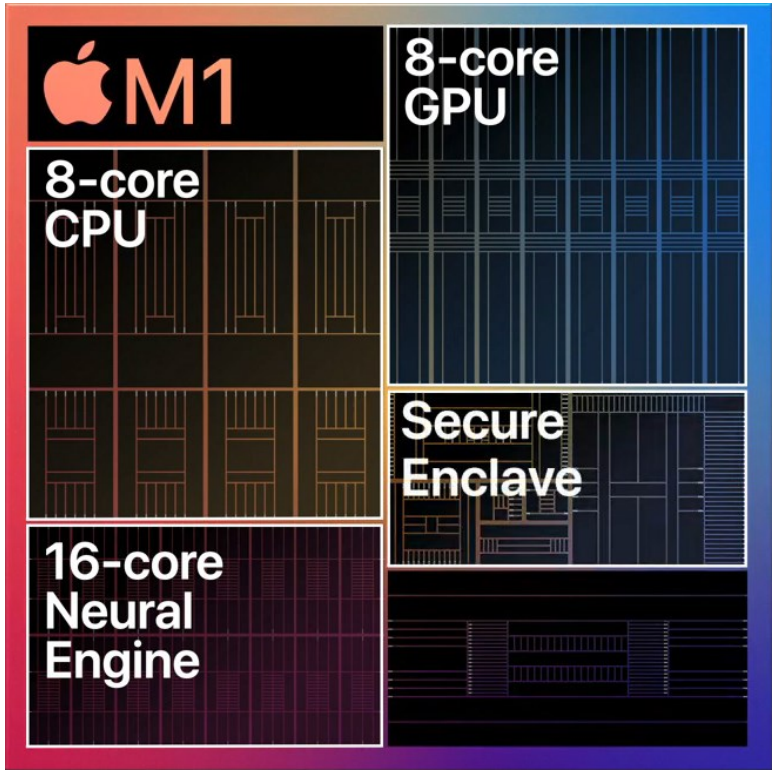


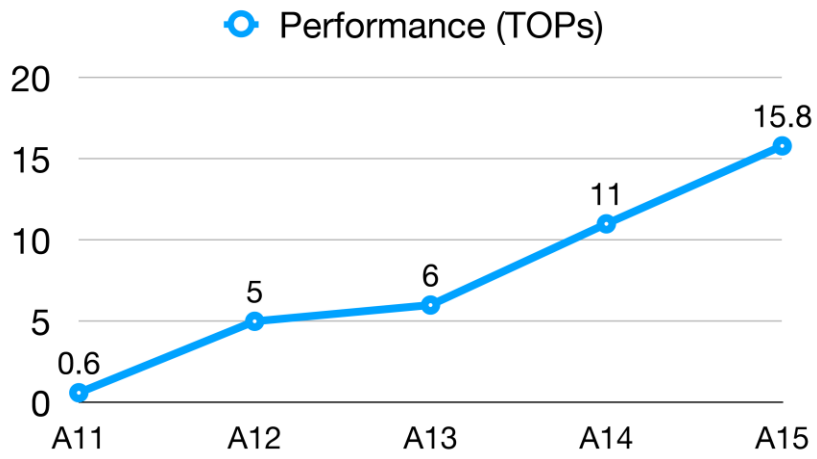


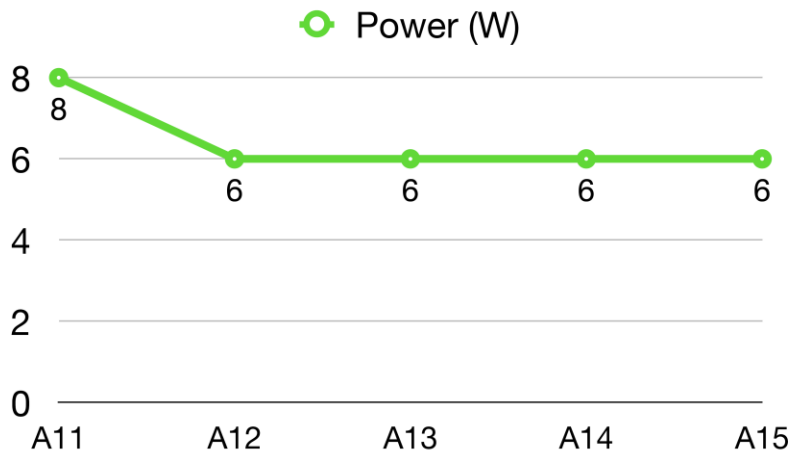


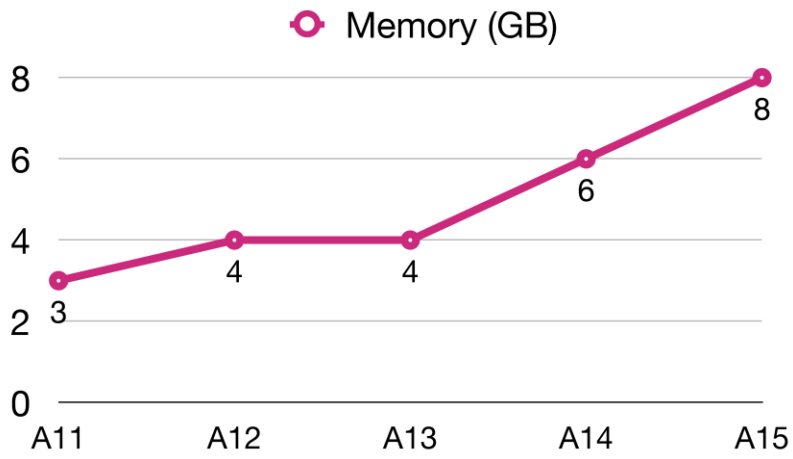
Edge AI Hardware

- Apple Neural Engine
  - An energy-efficient and high-throughput engine for ML inference on Apple silicon



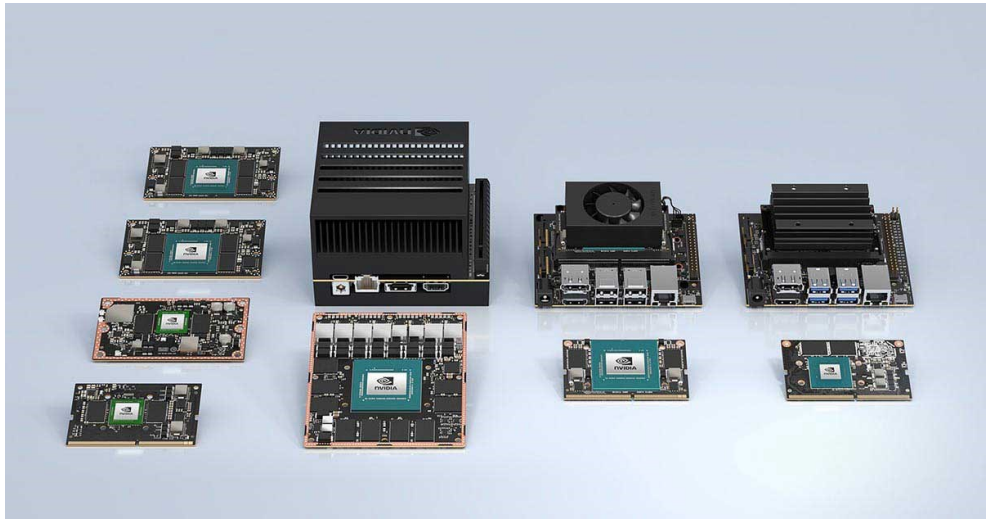


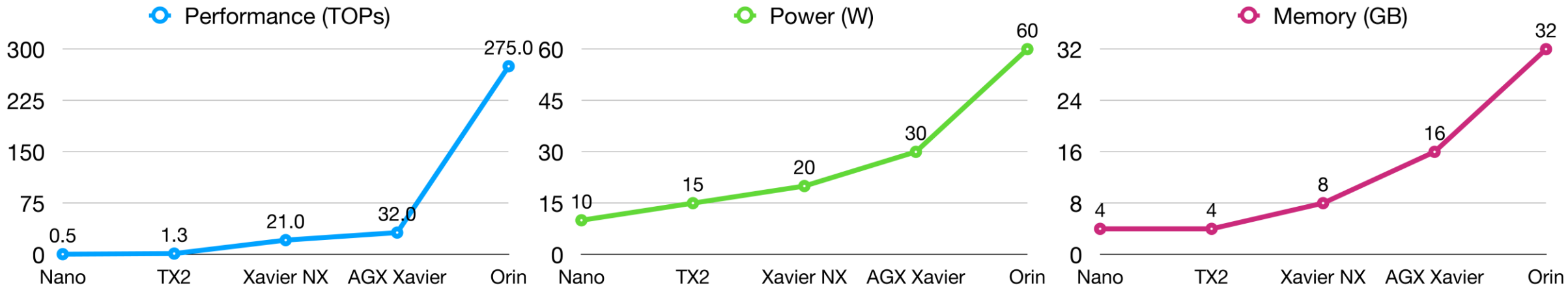




Edge AI Hardware

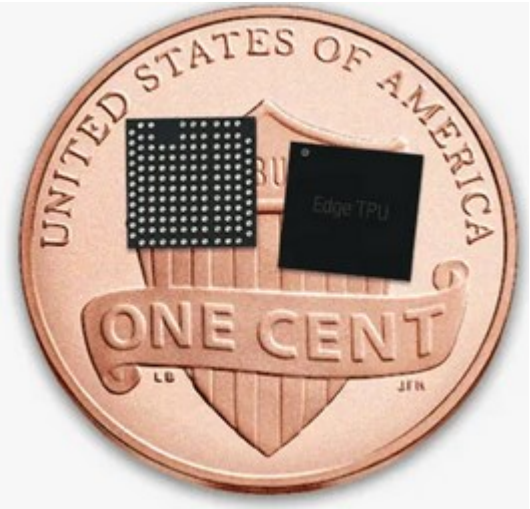
- Nvidia Jetson
  - A complete System on Module (SOM) that includes a GPU, CPU, memory, power management, high-speed interfaces, and more

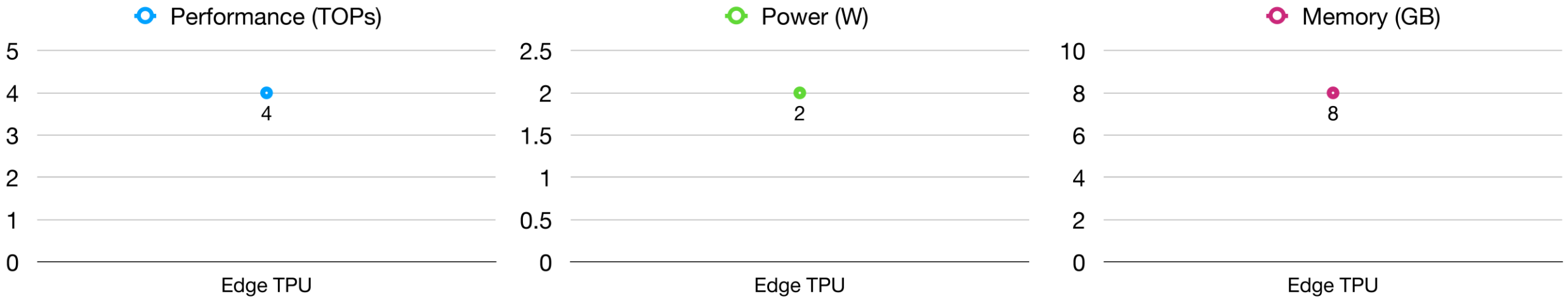




Edge AI Hardware

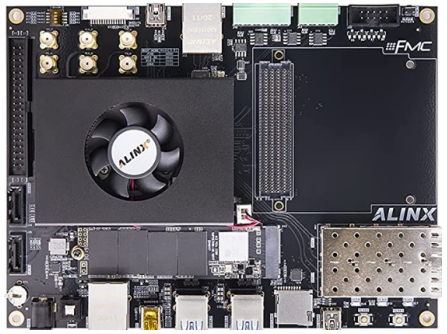
- Edge Tensor Processing Unit
  - An AI accelerator application-specific integrated circuit (ASIC) developed by Google for neural network machine learning, using Google‘s own TensorFlow software

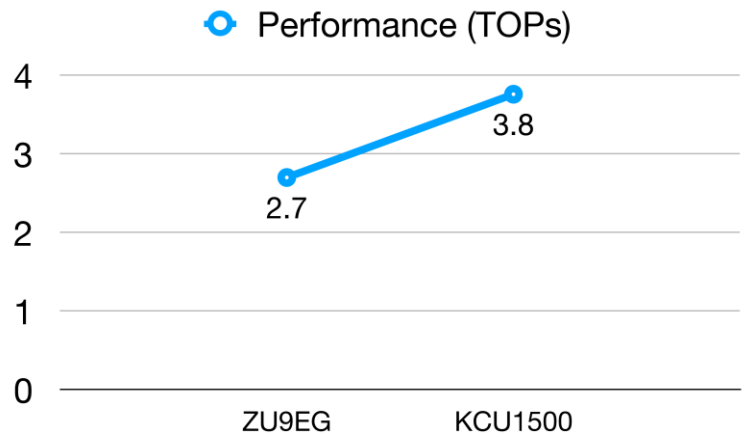


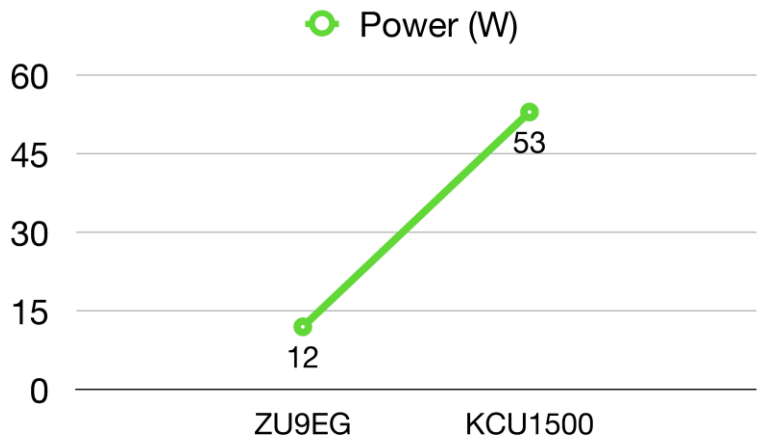


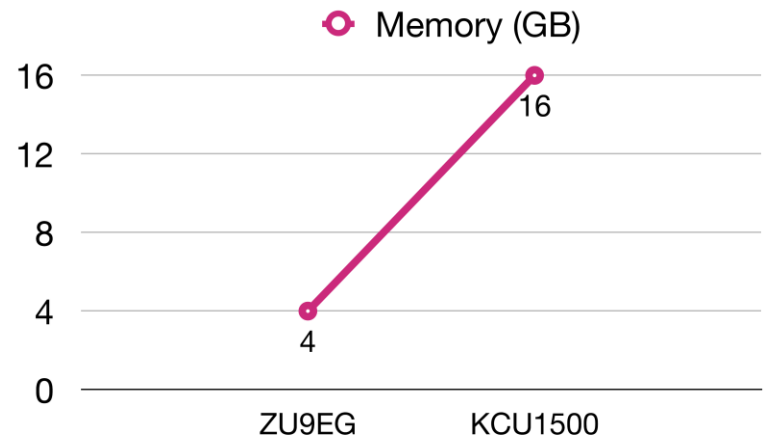
Edge AI Hardware

- FPGA-based Accelerator
  - Deliver higher performance compared to a fixed-architecture AI accelerator like a GPU due to efficiency of custom hardware acceleration



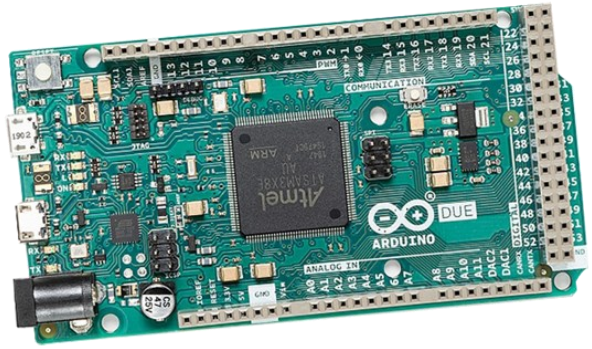


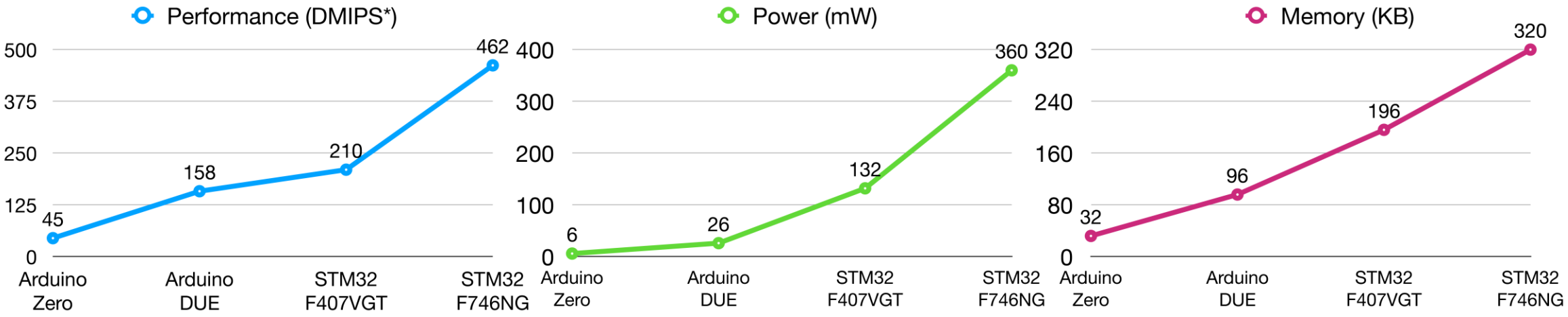




Edge AI Hardware

- Microcontrollers (MCU)
  - A compact integrated circuit designed for embedded systems
  - A typical microcontroller includes a processor, memory and input/output (I/O) peripherals on a single chip





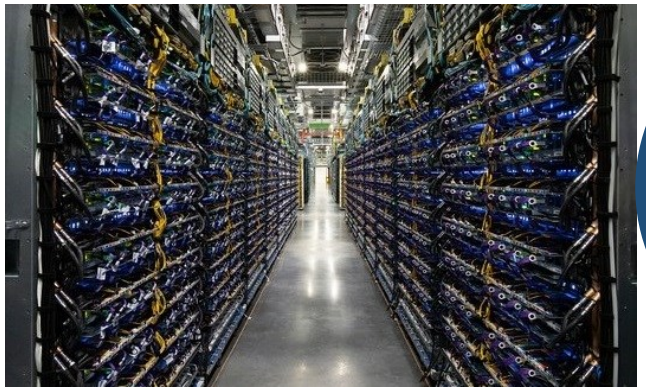
* Dhrystone Million Instructions Per Second (DMIPs) is an index for integer computation.

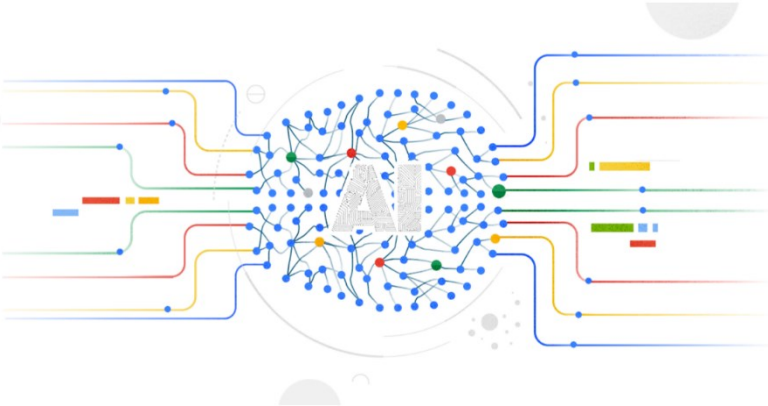
Efficient AI Models

Small Data

Lighter Algorithm

Efficient Hardware





Outline

- Demands for Efficient Model
- Efficiency Metrics

Efficiency Metrics

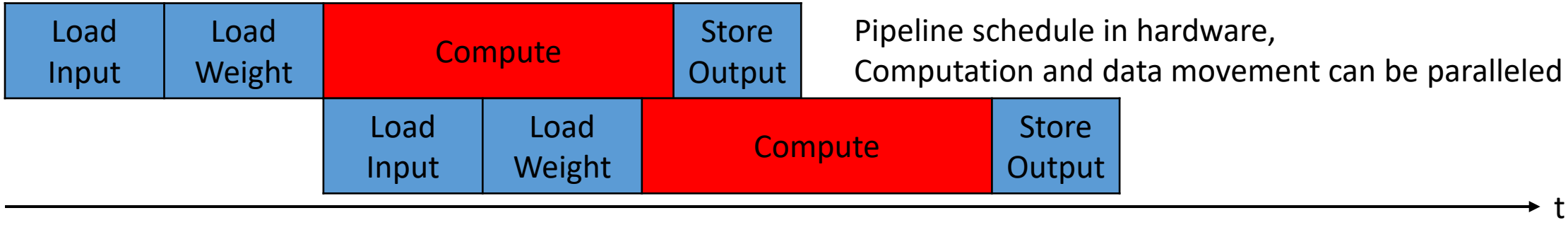
- Memory-Related
  - Number of parameters
  - Model size
  - Total/peak activations
- Computation Related
  - MACs
  - FLOPs

- **Smaller – Storage**
- **Faster – Latency**
- **Efficiency - Energy**

Latency

$$Latency \approx (T_{computation}, T_{data\_access})$$

- $T_{computation} \approx \dfrac{(Number\ of\ operations\ in\ NN\ model)}{(Number\ of\ operations\ that\ processor\ can\ process\ per\ second)}$ — Model Dependent / Hardware Dependent
- $T_{data\_access} \approx T_{(data\ movement\ of\ activations)} + T_{(data\ movement\ of\ weights)}$
- $T_{(data\ movement\ of\ weights)} \approx \dfrac{(NN\ model\ size)}{(Memory\ bandwidth\ of\ processor)}$ — Model Dependent / Hardware Dependent
- $T_{(data\ movement\ of\ activations)} \approx \dfrac{(Input\ activation\ size)+(Output\ activation\ size)}{(Memory\ Bandwidth\ of\ Processor)}$ — Model Dependent / Hardware Dependent

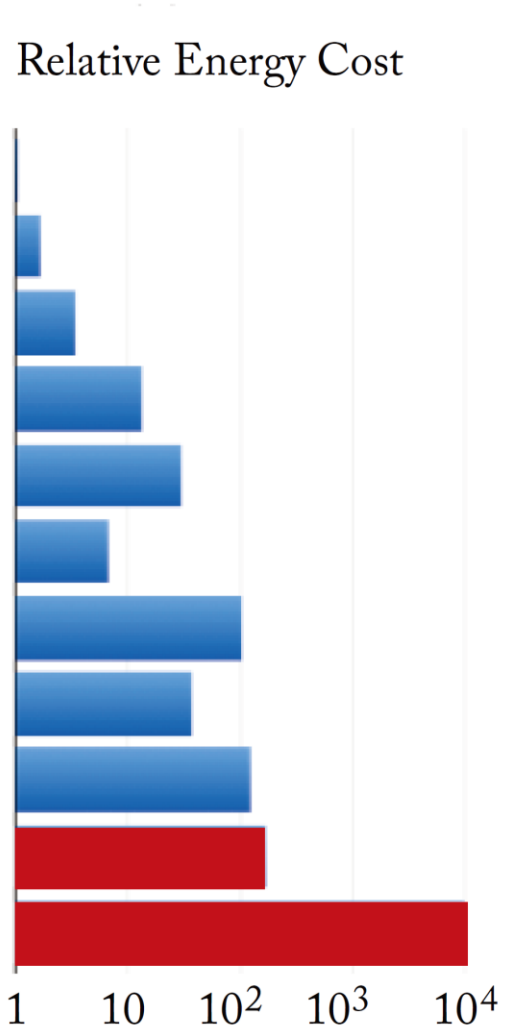


Pipeline schedule in hardware,
Computation and data movement can be paralleled

Energy Consumptions

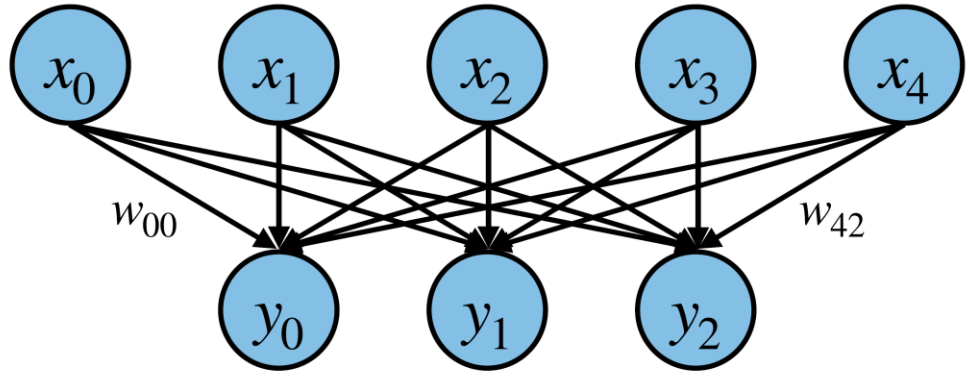
- 45nm process
- Energy consumption is dominated by the data movement
  - Capacitance of data movement tends to be much higher that the capacitance for arithmetic operations(PEs)
- Larger memories and longer interconnects
  - Consume more energy

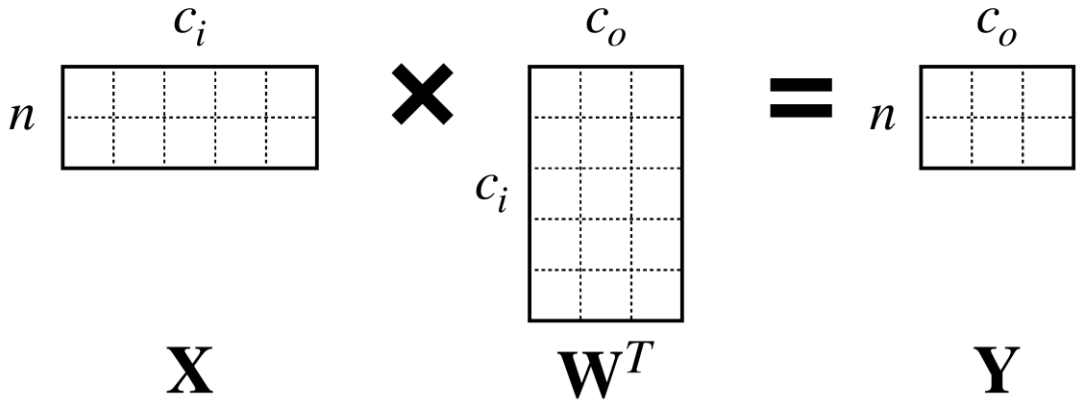
Operation:	Energy (pJ)
8b Add	0.03
16b Add	0.05
32b Add	0.1
16b FP Add	0.4
32b FP Add	0.9
8b Multiply	0.2
32b Multiply	3.1
16b FP Multiply	1.1
32b FP Multiply	3.7
32b SRAM Read (8KB)	5
32b DRAM Read	640



Number of Parameters

- Linear Layer





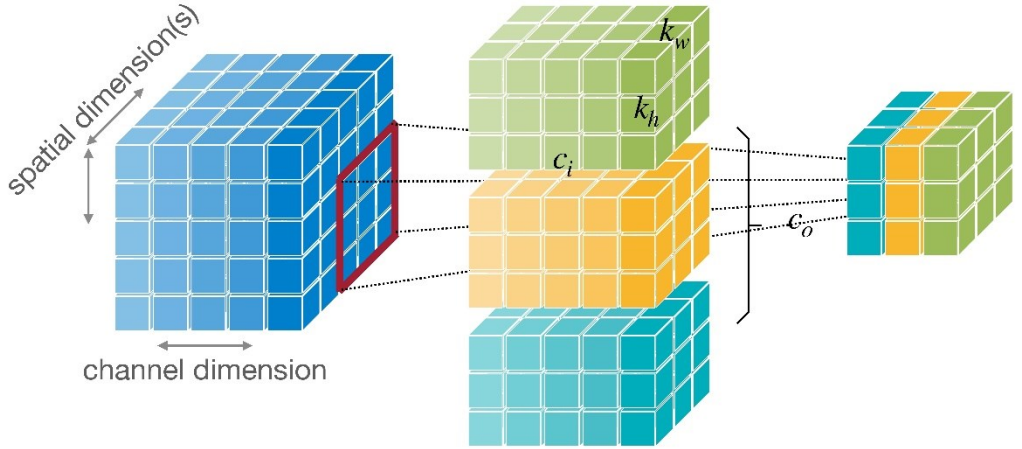
Layer Type	Number of Parameters (Bias ignored)
Linear Layer	$c_o \times c_i$
Convolution	
Grouped Convolution	
Depthwise Convolution	

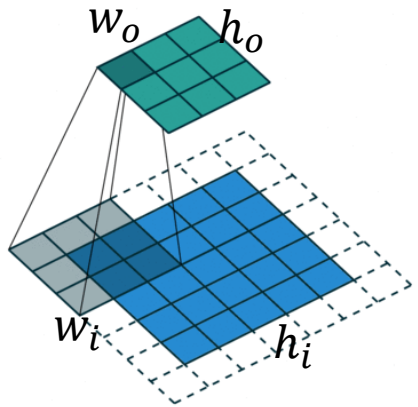
Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

Number of Parameters

- Convolution

Layer Type	Number of Parameters (Bias ignored)
Linear Layer	$c_o \times c_i$
Convolution	$c_o \times c_i \times k_h \times k_w$
Grouped Convolution	
Depthwise Convolution	



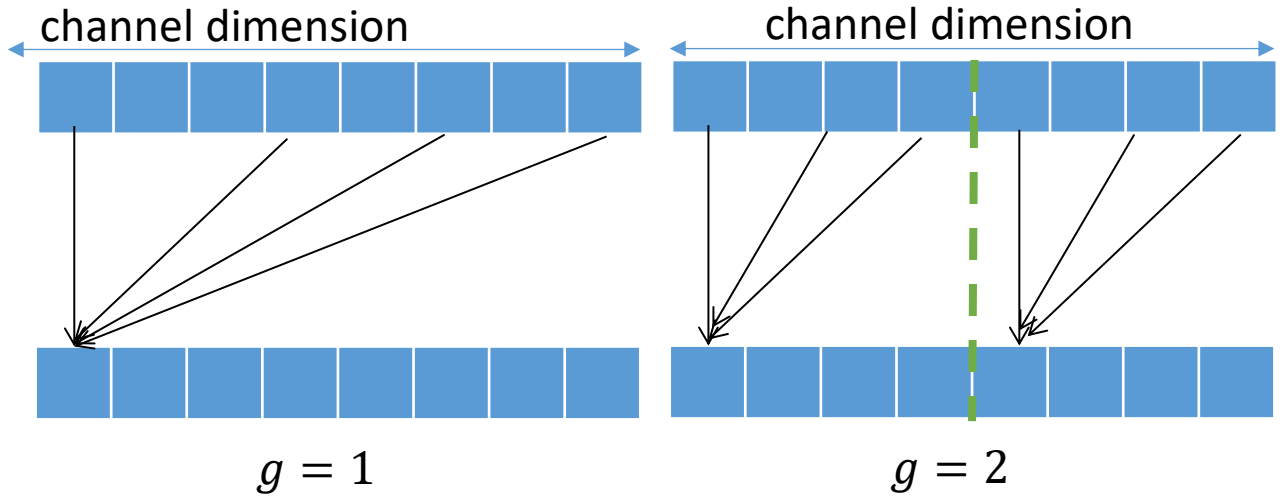


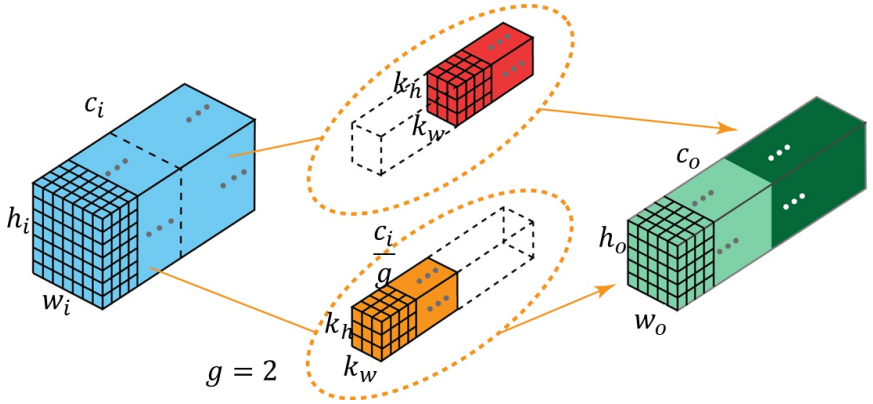
Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

Number of Parameters

- Grouped Convolution

Layer Type	Number of Parameters (Bias ignored)
Linear Layer	$c_o \times c_i$
Convolution	$c_o \times c_i \times k_h \times k_w$
Grouped Convolution	$c_o \times c_i \times k_h \times k_w \times \frac{1}{g}$
Depthwise Convolution	



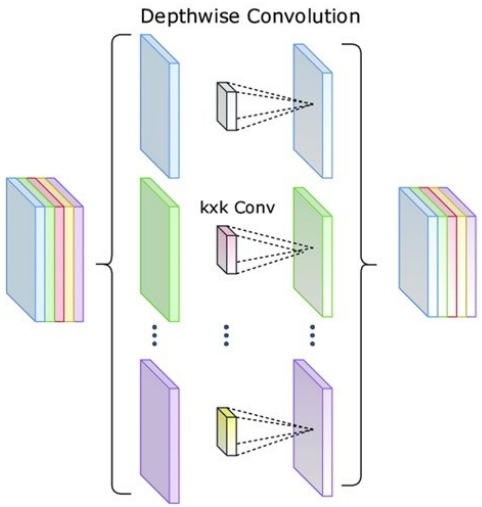


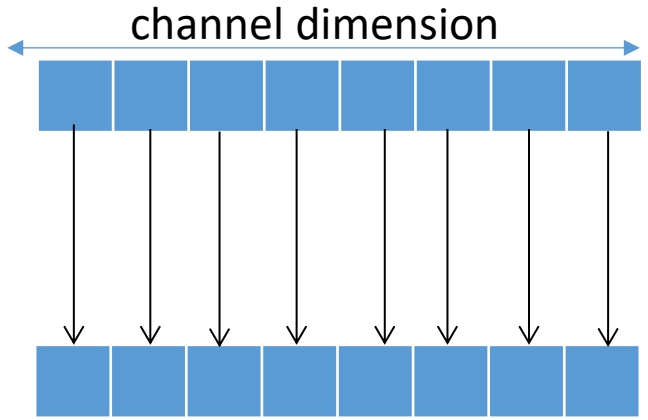
Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

Number of Parameters

- Depthwise Convolution

Layer Type	Number of Parameters (Bias ignored)
Linear Layer	$c_o \times c_i$
Convolution	$c_o \times c_i \times k_h \times k_w$
Grouped Convolution	$c_o \times c_i \times k_h \times k_w \times \frac{1}{g}$
Depthwise Convolution	$c_o \times k_h \times k_w$





$$g = c_i = c_o$$

Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

AlexNet Example

Layer	Kernel Shape	Groups	Output Shape	# of parameters
Image			$3 \times 224 \times 224$	
CONV	11×11	1	$96 \times 55 \times 55$	$96 \times 3 \times 11 \times 11 = 24{,}848$
MaxPool	3×3		$96 \times 27 \times 27$	
CONV	5×5	2	$256 \times 27 \times 27$	$256 \times 96 \times 5 \times 5/2 = 307{,}200$
MaxPool	3×3		$256 \times 13 \times 13$	
CONV	3×3	1	$384 \times 13 \times 13$	$384 \times 256 \times 3 \times 3 = 884{,}736$
CONV	3×3	2	$384 \times 13 \times 13$	$384 \times 384 \times 3 \times 3/2 = 663{,}552$
CONV	3×3	2	$256 \times 13 \times 13$	$384 \times 256 \times 3 \times 3/2 = 442{,}368$
MaxPool	3×3		$256 \times 6 \times 6$	
Linear			4096	$4096 \times (256 \times 6 \times 6) = 37{,}748{,}736$
Linear			4096	$4096 \times 4096 = 16{,}777{,}216$
Linear			1000	$1000 \times 4096 = 4{,}096{,}000$

Total 61M

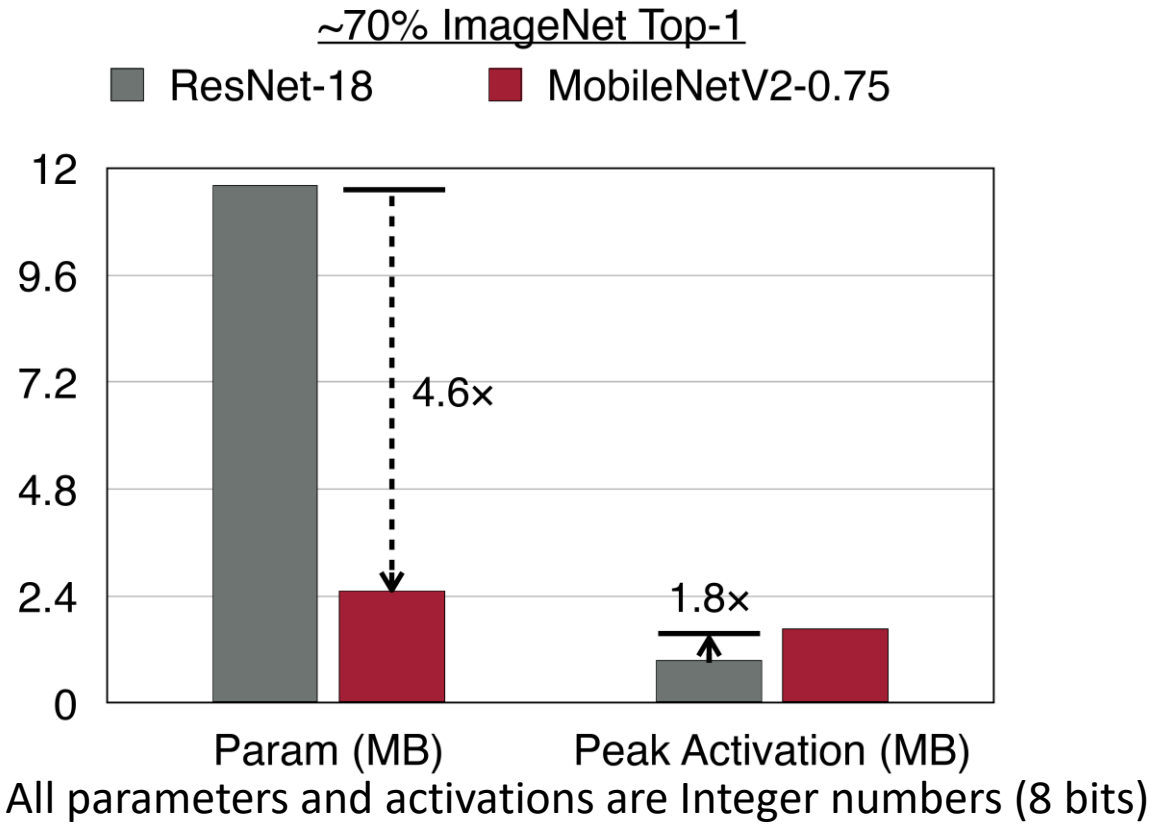
# of params dominant by Linear layer

Model Size

- Model size measures the storage for the weights of the given neural network
  - The common units for model size are: MB (megabyte), KB (kilobyte), bits
- In general, if the whole neural network uses the same data type and precision
  - $(Model\ size) = (\#\ of\ parameter) \times (bitwidth)$
- Example: AlexNet has 61M parameters
  - If all weights are stored with 32-bit numbers,
    - Total storage will be about $61M \times 4Bytes = 224MB$
  - If all weights are stored with 8-bit numbers
    - Total storage will be about $61M \times 1Bytes = 61MB$

Number of Activations

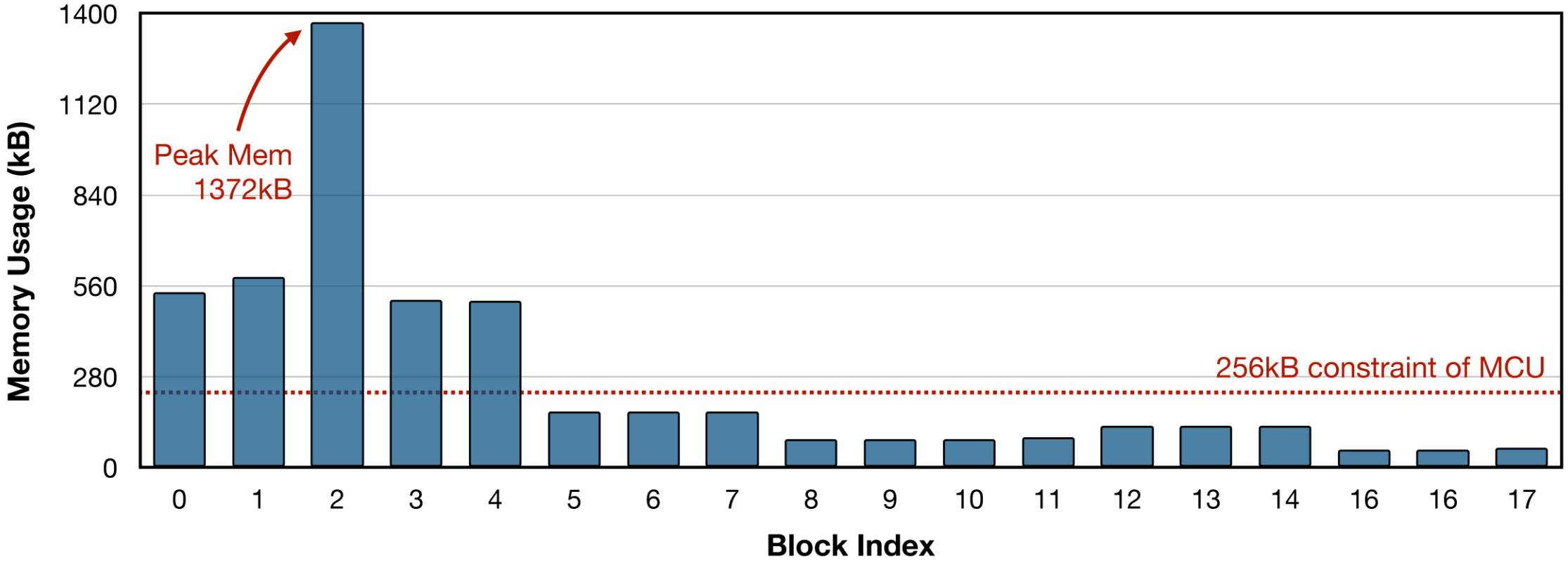
- Number of activation is the memory bottleneck in inference on IoT devices
  - Not number of parameters



All parameters and activations are Integer numbers (8 bits)

Imbalanced Memory Distribution

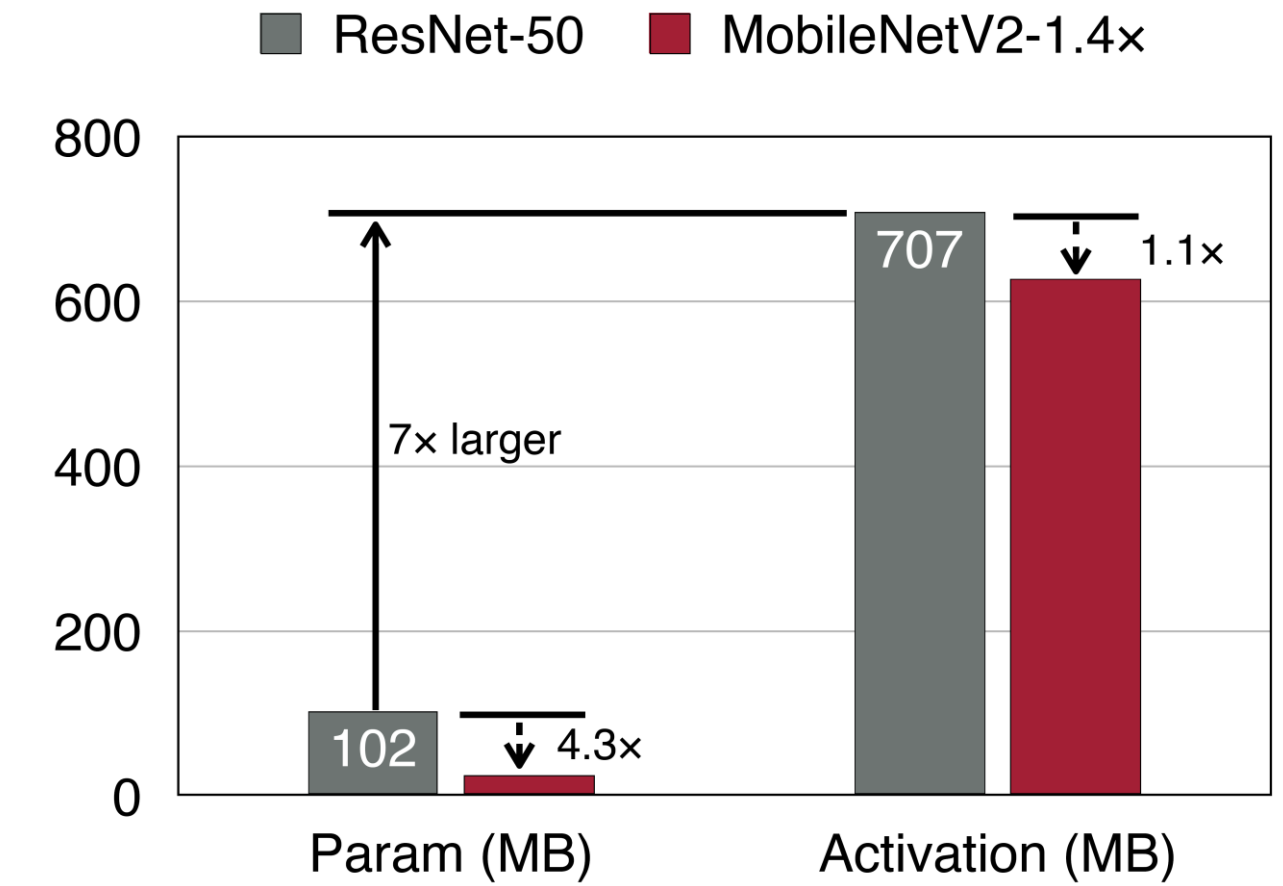
- MobileNetV2



Lin, J., Chen, W. M., Lin, Y., Gan, C., & Han, S. (2020). Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems*, *33*, 11711-11722.

Imbalanced Memory Distribution

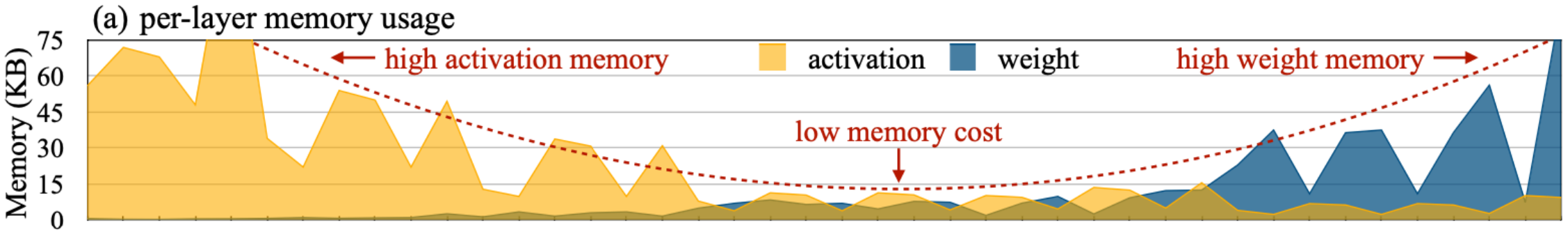
- The main memory bottleneck does not improve much



All parameters and activations are Floating-point numbers (32 bits)

Memory Distribution

- Activation and Weight



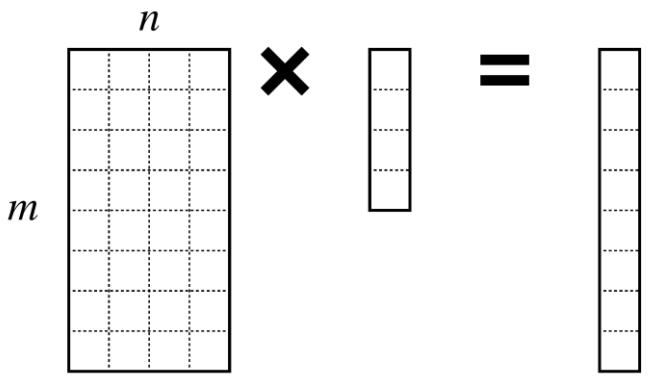
AlexNet Example

Layer	Kernel Shape	Groups	Output Shape	# of activation	Input + output
Image			$3 \times 224 \times 224$	$3 \times 224 \times 224 = 150{,}528$	
CONV	11×11	1	$96 \times 55 \times 55$	$96 \times 55 \times 55 = 290{,}400$	**440,928**
MaxPool	3×3		$96 \times 27 \times 27$	$96 \times 27 \times 27 = 69{,}984$	360,384
CONV	5×5	2	$256 \times 27 \times 27$	$256 \times 27 \times 27 = 186{,}624$	256,608
MaxPool	3×3		$256 \times 13 \times 13$	$256 \times 13 \times 13 = 43{,}264$	229,888
CONV	3×3	1	$384 \times 13 \times 13$	$384 \times 13 \times 13 = 64{,}896$	108,160
CONV	3×3	2	$384 \times 13 \times 13$	$384 \times 13 \times 13 = 64{,}896$	129,792
CONV	3×3	2	$256 \times 13 \times 13$	$256 \times 13 \times 13 = 43{,}264$	108,160
MaxPool	3×3		$256 \times 6 \times 6$	$256 \times 6 \times 6 = 9{,}216$	52,480
Linear			4096	4096	13,312
Linear			4096	4096	8,192
Linear			1000	1000	5,096
				Total 932,264	

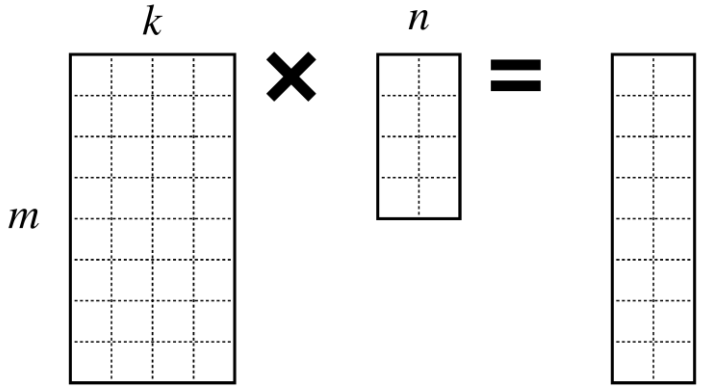
$$(peak\ \#of\ activation) \approx (\#\ of\ input\ acitvaion + \#\ of\ output\ activation)$$

Number of MAC Operations

- Multiply-Accumulate operation (MAC)
  - $a \leftarrow a + b \times c$
- Matrix-Vector Multiplication
  - $MACs = m \times n$
- General Matrix-Matrix Multiplication (GEMM)
  - $MACs = m \times n \times k$



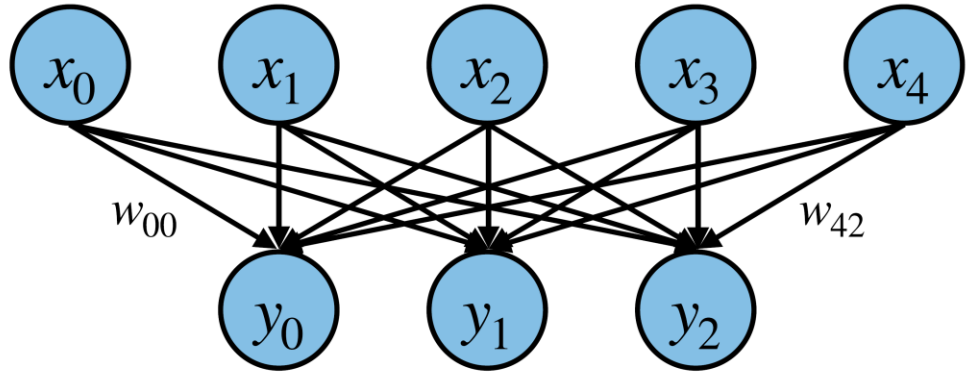
Matrix-Vector Multiplication

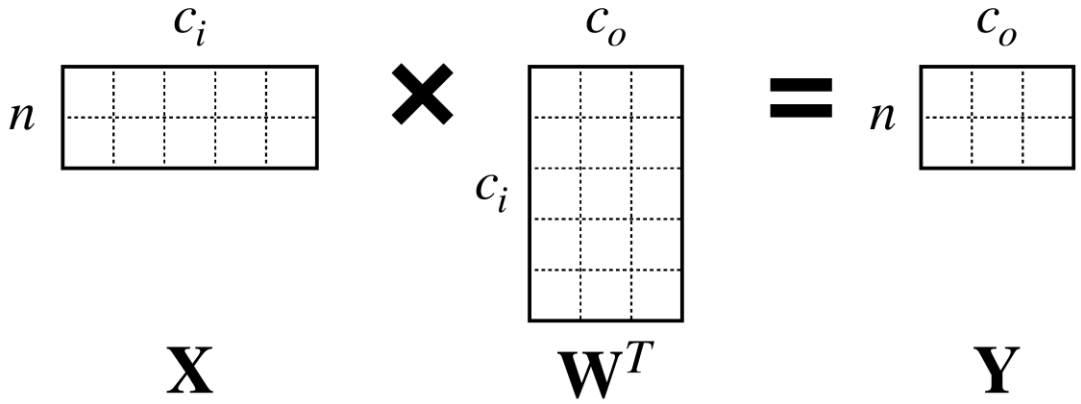


General Matrix-Matrix Multiplication (GEMM)

Number of MAC Operations

- Linear Layer





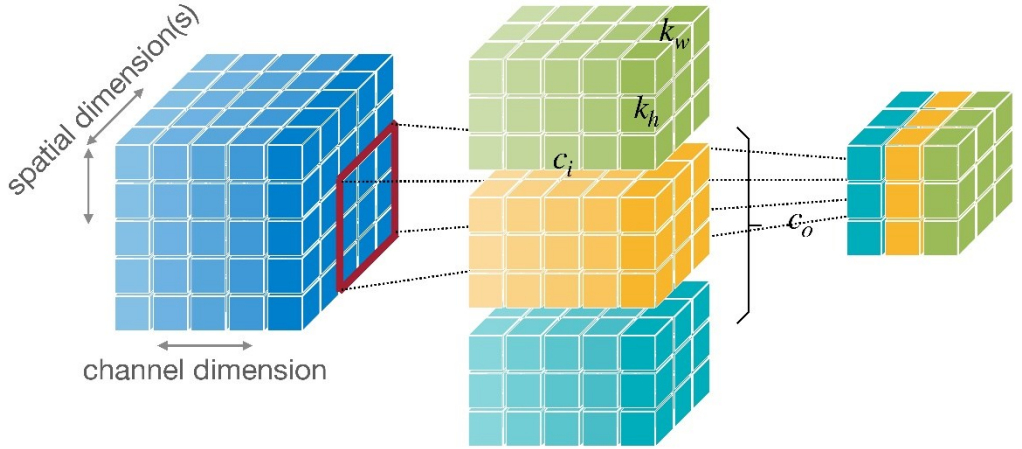
Layer Type	MACs (batch size n=1)
Linear Layer	$c_o \times c_i$
Convolution	
Grouped Convolution	
Depthwise Convolution	

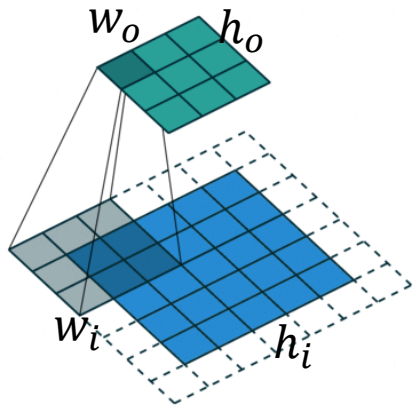
Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

Number of MAC Operations

- Convolution

Layer Type	MACs (batch size n=1)
Linear Layer	$c_o \times c_i$
Convolution	$c_i \times k_h \times k_w \times h_o \times w_o \times c_o$
Grouped Convolution	
Depthwise Convolution	



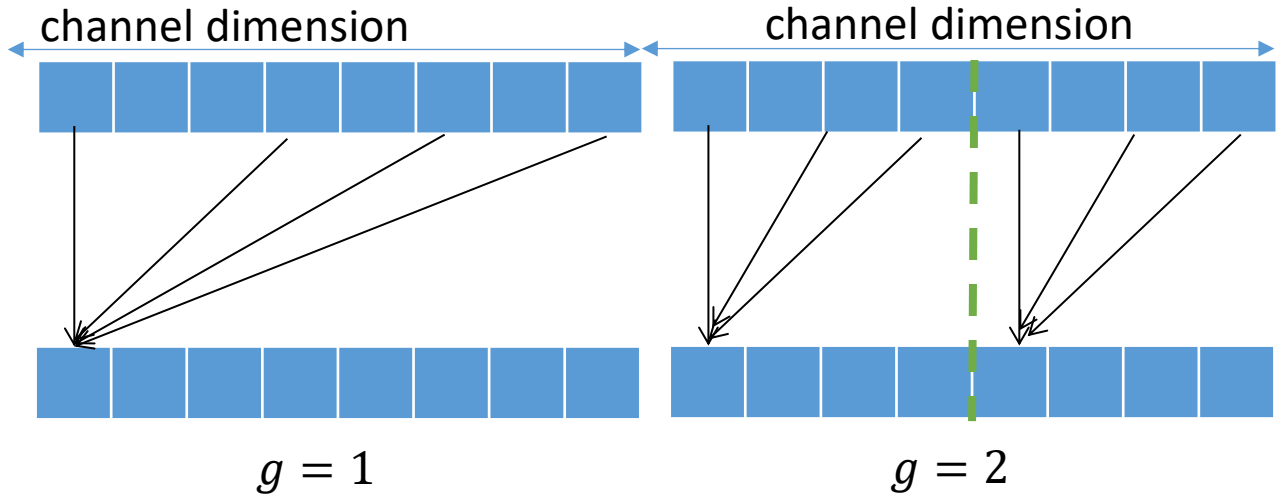


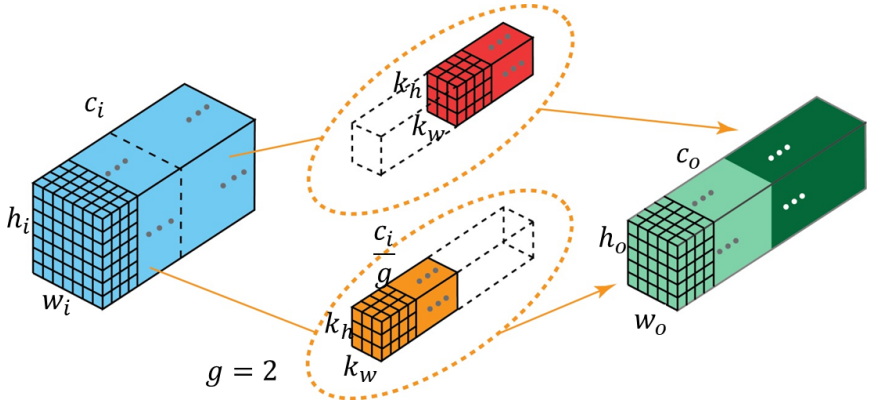
Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

Number of MAC Operations

- Grouped Convolution

Layer Type	MACs (batch size n=1)
Linear Layer	$c_o \times c_i$
Convolution	$c_i \times k_h \times k_w \times h_o \times w_o \times c_o$
Grouped Convolution	$\frac{c_i}{g} \times k_h \times k_w \times h_o \times w_o \times c_o$
Depthwise Convolution	



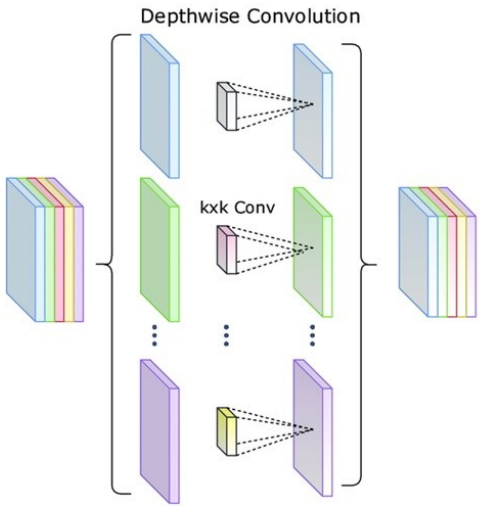


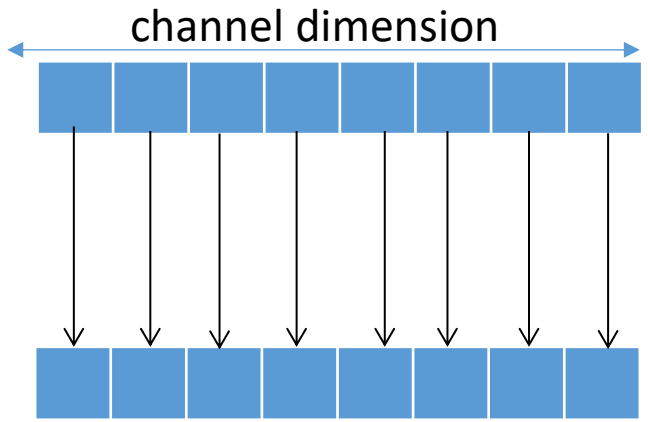
Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

Number of MAC Operations

- Depthwise Convolution

Layer Type	MACs (batch size n=1)
Linear Layer	$c_o \times c_i$
Convolution	$c_i \times k_h \times k_w \times h_o \times w_o \times c_o$
Grouped Convolution	$\frac{c_i}{g} \times k_h \times k_w \times h_o \times w_o \times c_o$
Depthwise Convolution	$k_h \times k_w \times h_o \times w_o \times c_o$





$g = c_i = c_o$

Notation	
n	Batch size
c_i, c_o	Input/output channel
w_i, w_o	Input/output width
h_i, h_o	Input/output height
k_w, k_h	Kernel width/height
g	Groups

AlexNet Example

Layer	Kernel Shape	Groups	Output Shape	Input + output	MACs
Image			$3 \times 224 \times 224$		
CONV	11×11	1	$96 \times 55 \times 55$	440,928	$96 \times 3 \times 11 \times 11 \times 55 \times 55 = 105{,}415{,}200$
MaxPool	3×3		$96 \times 27 \times 27$	360,384	
CONV	5×5	2	$256 \times 27 \times 27$	256,608	$256 \times 96 \times 5 \times 5 \times 27 \times 27/2 = 228{,}948{,}800$
MaxPool	3×3		$256 \times 13 \times 13$	229,888	
CONV	3×3	1	$384 \times 13 \times 13$	108,160	$384 \times 256 \times 3 \times 3 \times 13 \times 13 = 149{,}520{,}384$
CONV	3×3	2	$384 \times 13 \times 13$	129,792	$384 \times 384 \times 3 \times 3 \times 13 \times 13/2 = 112{,}140{,}288$
CONV	3×3	2	$256 \times 13 \times 13$	108,160	$256 \times 384 \times 3 \times 3 \times 13 \times 13/2 = 74{,}760{,}192$
MaxPool	3×3		$256 \times 6 \times 6$	52,480	
Linear			4096	13,312	$4096 \times (256 \times 6 \times 6) = 37{,}748{,}736$
Linear			4096	8,192	$4096 \times 4096 = 16{,}777{,}216$
Linear			1000	5,096	$1000 \times 4096 = 4{,}096{,}000$

Total 724M MACs

Number of FLOP

- Floating Point Operations (FLOP)
  - A multiply is a floating point operation
  - An add is a floating point operation
- 1 MAC = 2 FLOP
- Example
  - AlexNet has $724M\ MAC = 724M \times 2\ FLOP = 1.4G\ FLOP$
- FLOPS=Floating-point operations per second
  - $FLOPS = \frac{FLOP}{second}$

Today's AI is too big

Memory is expensive

How should we make deep learning more efficient?