本課程為教育部109年度智慧聯網技術課程推廣計畫之補助課程
教材內容也自教育部智慧聯網技術重點模組(人工智慧視覺感知運算系統模組)教材改編

人工智慧視覺運算方法實務與計算平台

謝東佑

可測及可靠系統實驗室

(Testable And Reliable Systems Lab., TARS)

國立中山大學電機系

Office: 工EC-7038

07-5252000 Ext. 4114

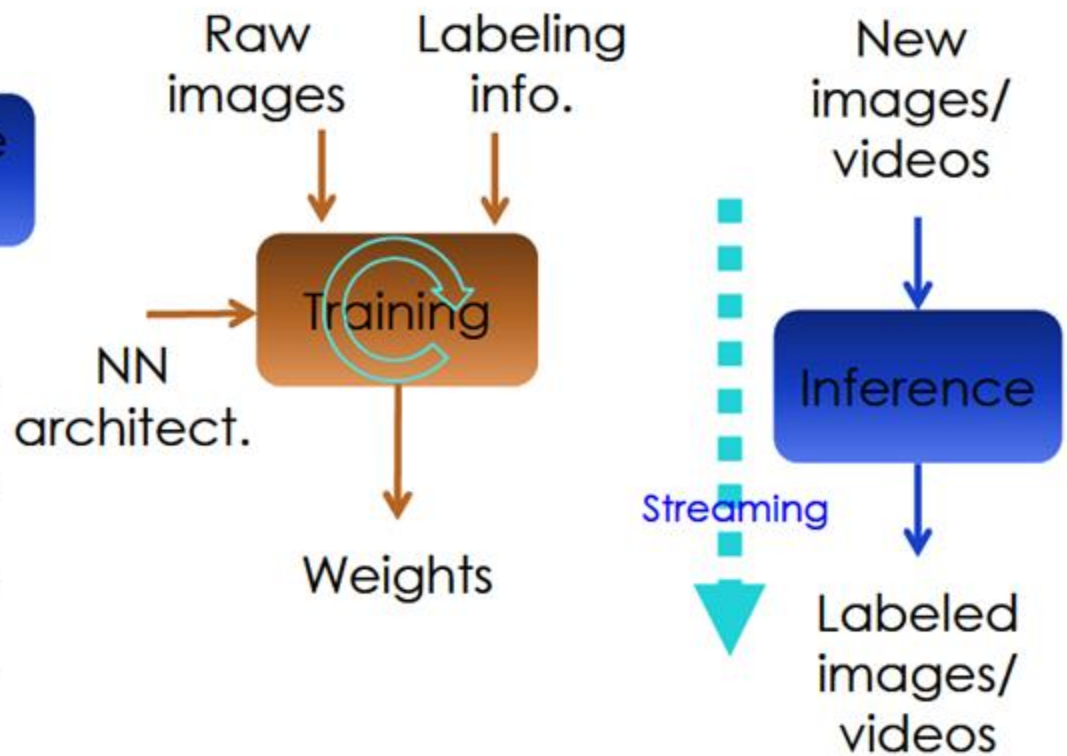
tyhsieh@mail.ee.nsysu.edu.tw

Keep feet on the ground

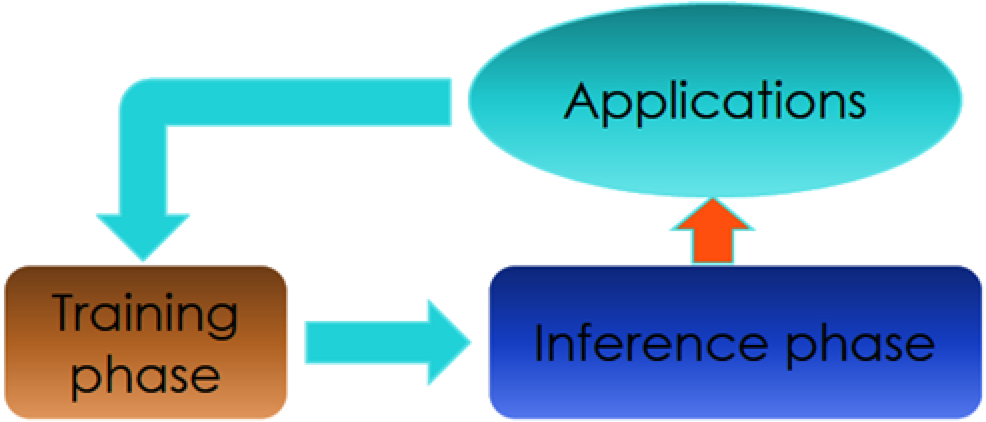
HOW TO OBTAIN ACCURATE AI VISUAL ALGORITHM?

Training VS Inference

Procedure	Sample amount	Algo. Type	Latency tolerance
Training	Finite	Close loop	High
Inference	Infinite	Open loop	Low



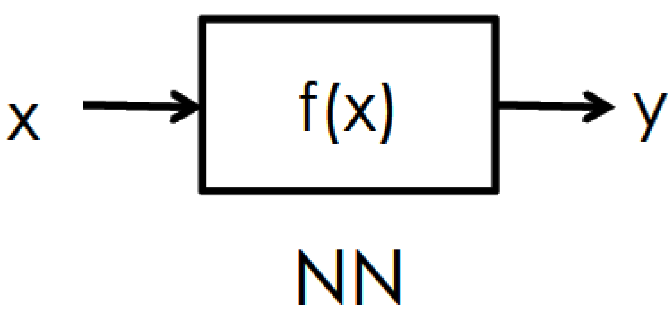
Training VS Inference

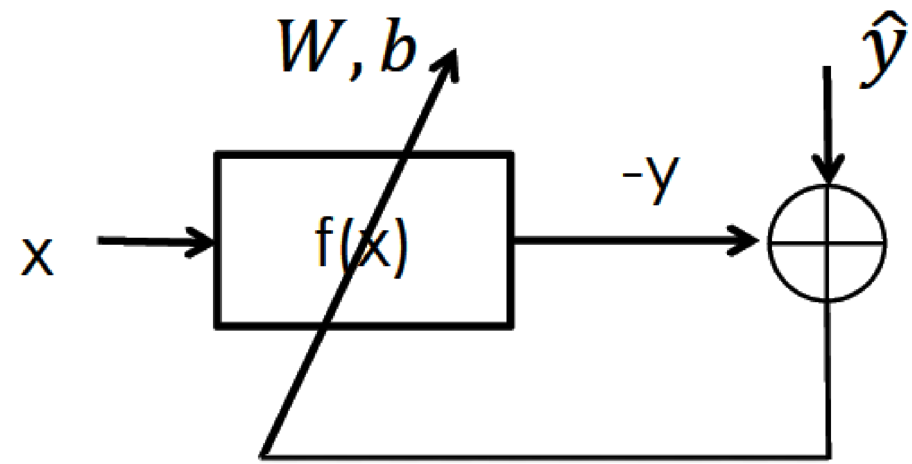


- Data labeling services
- Autonomous vehicle
- Security
- Biomedical imaging
- Smart robots
- Industry 4.0

Procedure	Sample amount	Algo. Type	Latency tolerance	Device quantity	Device unit price	Existing type	Operating period
Training	Finite	Close loop	High	Few	High	Cloud	Intermittent
Inference	Infinite	Open loop	Low	Many	Low	Edge	Continued

Training VS Inference





$$y = f(x) = W.X + b$$

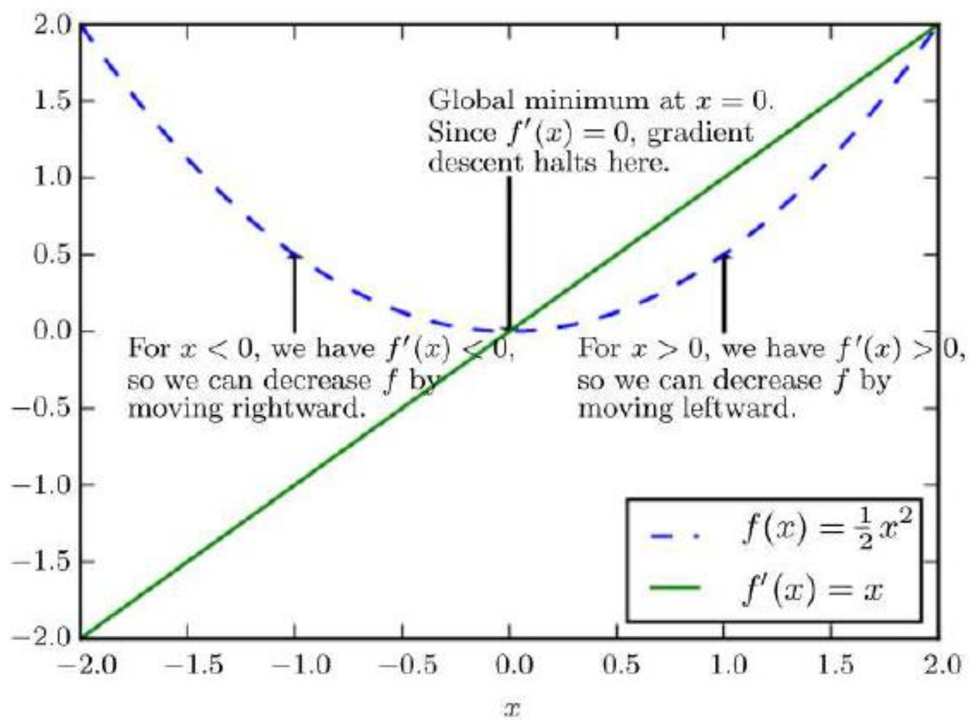
$$|\hat{y} - y| \to 0$$

Inference side

Training side

Gradient Descent (梯度下降法)

- For x<0, f'(x)<0, so we can decrease f by moving rightward
- For x>0, f'(x)>0, so we can decrease f by moving leftward
- The step size is correlated to the value of the gradient
  - Large gradient → large step size
  - Small gradient → small step size



Gradient Descent

$$f(x+\underline{\Delta x}) \simeq f(x) + \underline{\Delta x}\cdot f'(x)$$

$$x' = x + \Delta x$$

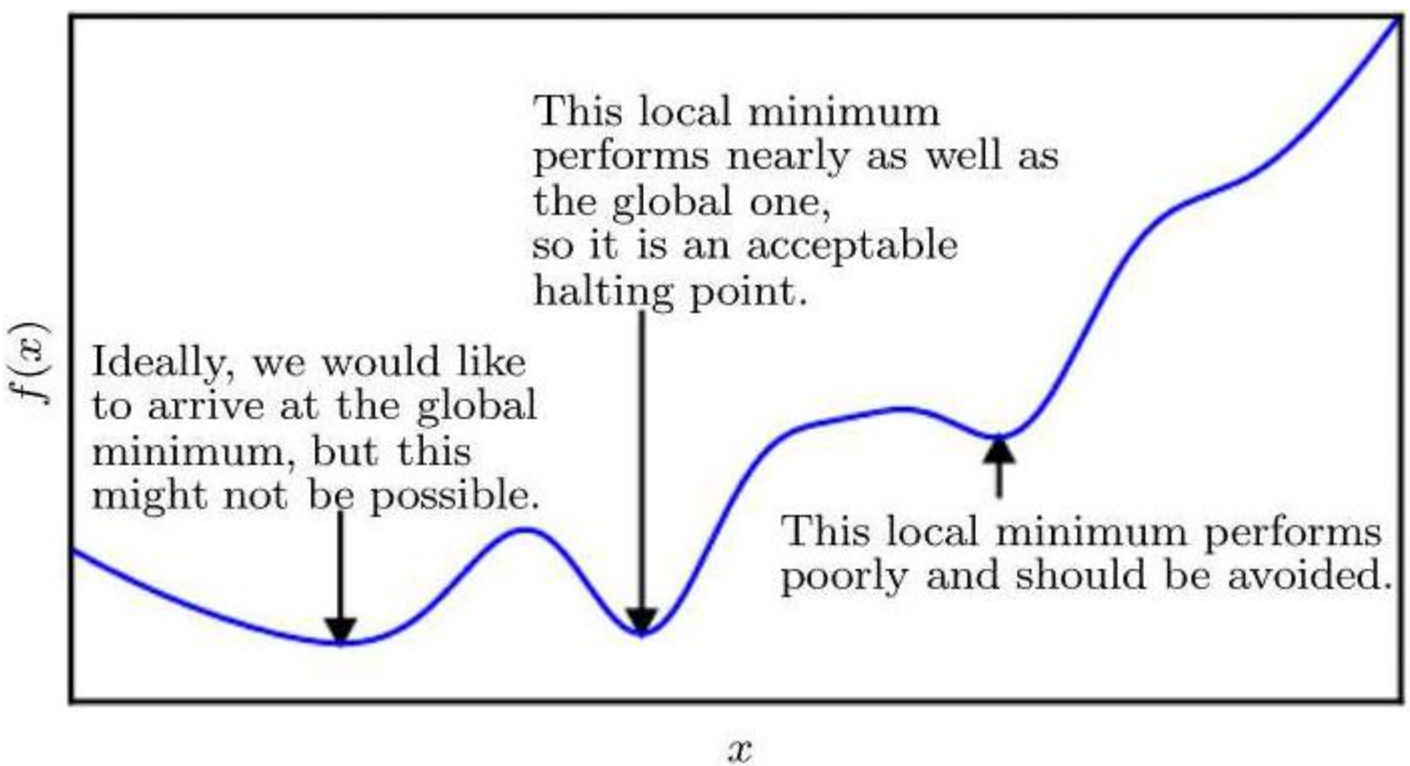
$$= x - \epsilon \underline{\nabla_x f(x)}$$

learning rate

Gradient of $f(x)$

Gradient Descent

- **Local minimum issue**
  - **The training did not converge to the global minimum**
  - **The step size may need to become large again**



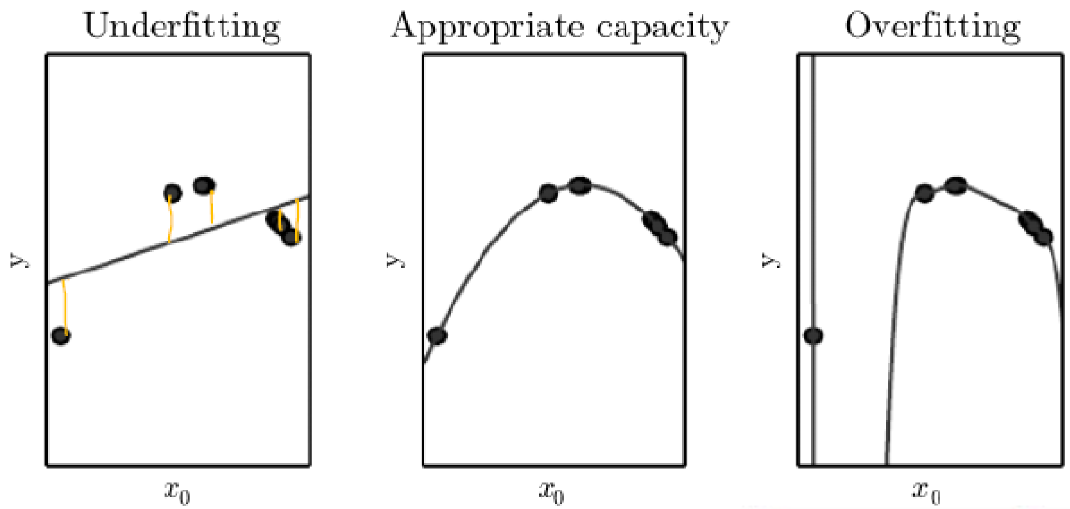
Generalization

- **Generalization- the ability to perform well on previously unseen inputs**
  - **Training error** $\| \overline{W} \cdot \underline{\overline{X}}^{(train)} - y^{(train)} \|_2^2$
  - **Test error (generalization error)** $\| \overline{W} \cdot \underline{\overline{X}}^{(test)} - y^{(test)} \|_2^2$
- **How well an AI algorithm will perform depends on**
  - **Make the training error small**
  - **Make the gap between training error and test error small**

Capacity, Underfitting, Overfitting

- **Capacity- the ability of a model to fit a wide variety of functions**
  - **Models with low capacity may struggle to fit the training set**
  - **Models with high capacity can overfit by memorizing properties of the training set that do not serve well on the test set**

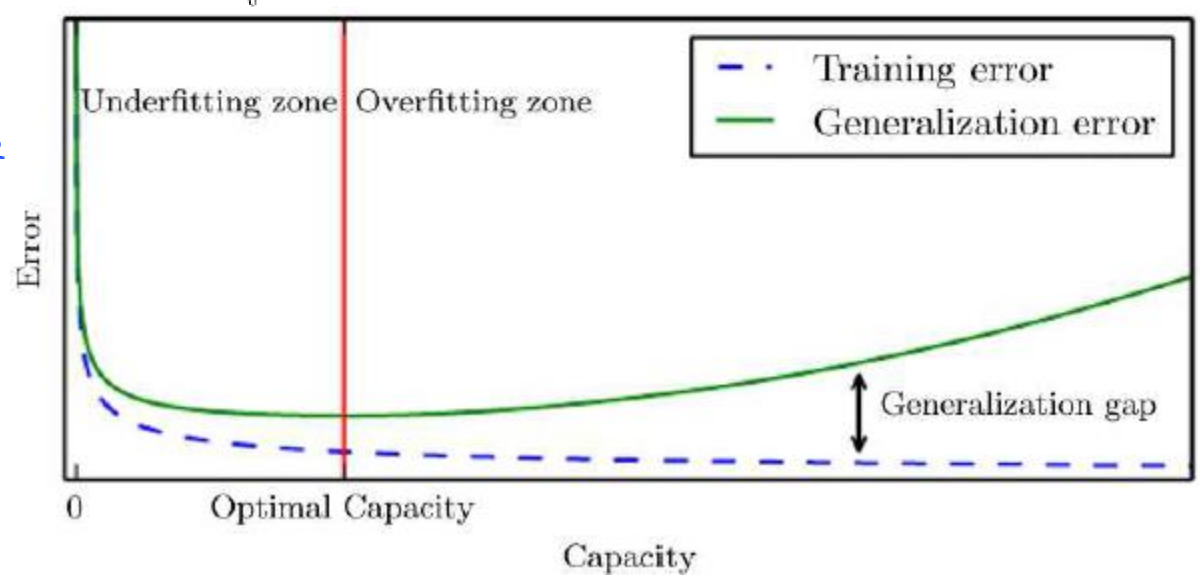
Capacity, Underfitting, Overfitting



$$y_9 = b + \sum_{i=1}^{9} w_i x^i$$

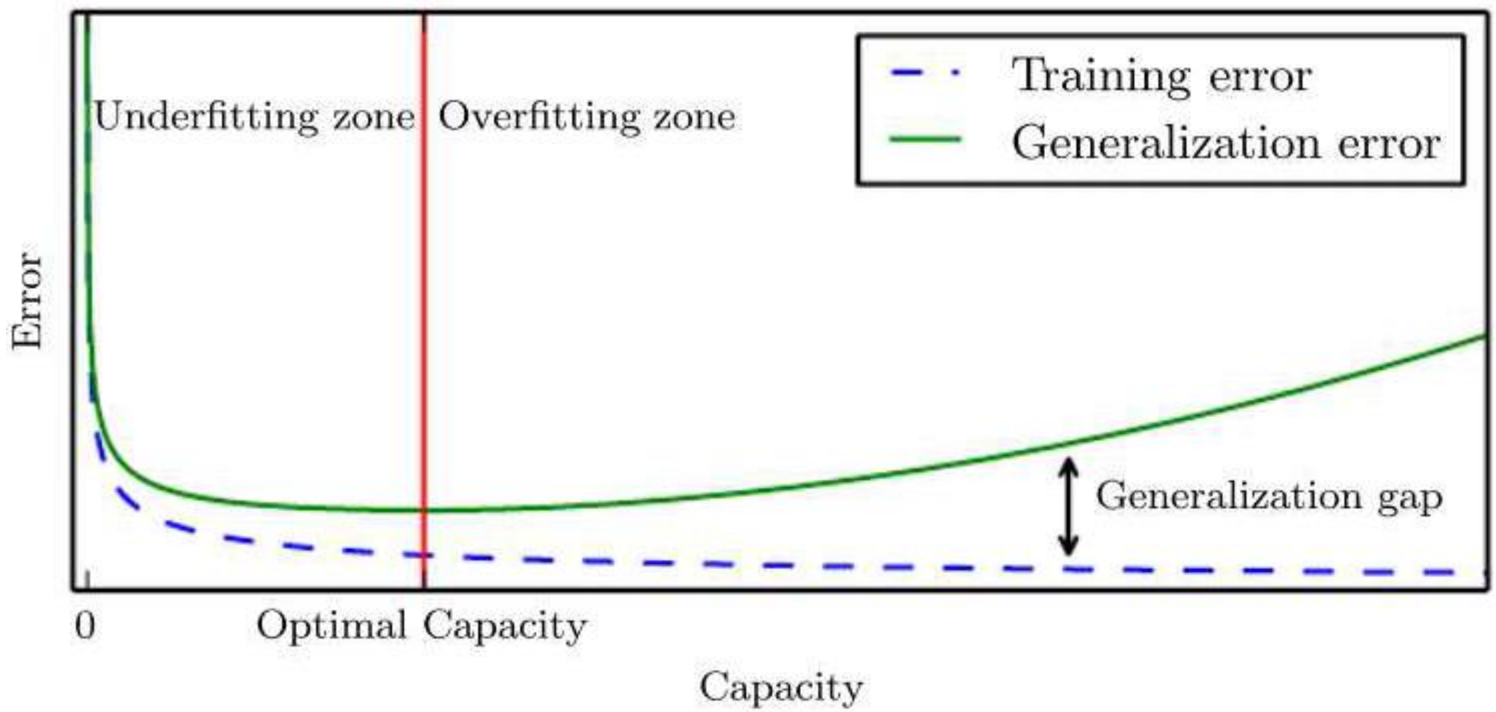
$$Y_1 = b + wx$$

$$Y_2 = b + w_1 x + w_2 x^2$$



How to Obtain an Accurate AI Algorithm?

- **Increase depth of NN**
- **Increase the variety of dataset**



The Variety of Dataset

- **ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness**

(a) Texture image

81.4%	**Indian elephant**
10.3%	indri
8.2%	black swan

(b) Content image

71.1%	**tabby cat**
17.3%	grey fox
3.3%	Siamese cat

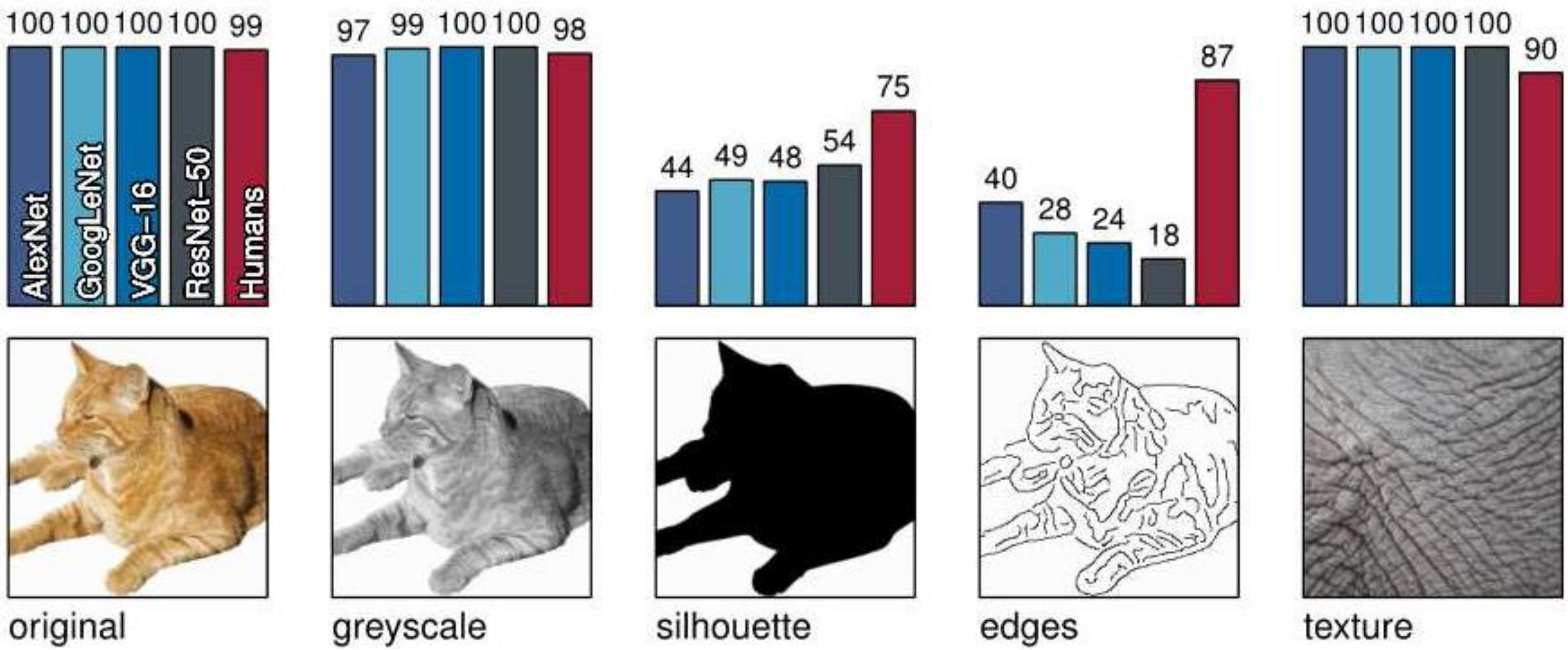
(c) Texture-shape cue conflict

63.9%	**Indian elephant**
26.4%	indri
9.6%	black swan

From a conference paper at ICLR 2019

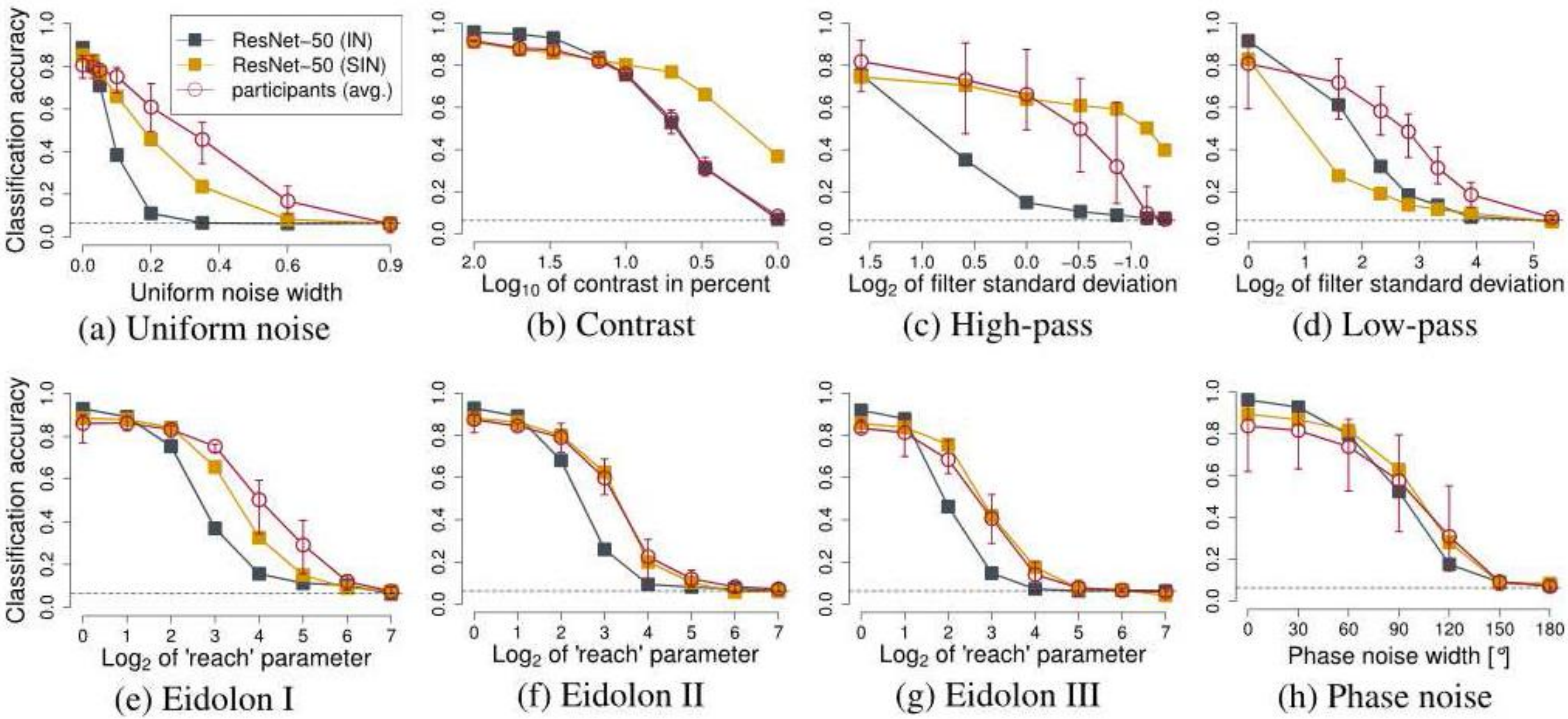
IMAGENET-TRAINED CNNS ARE BIASED TOWARDS TEXTURE; INCREASING SHAPE BIAS IMPROV
ACCURACY AND ROBUSTNESS

The Variety of Dataset



From a conference paper at ICLR 2019

Increasing Shape Bias Improves Accuracy and Robustness



From a conference paper at ICLR 2019

WHAT ARE KEY COMPUTATIONS IN AI VISUAL ALGORITHMS?

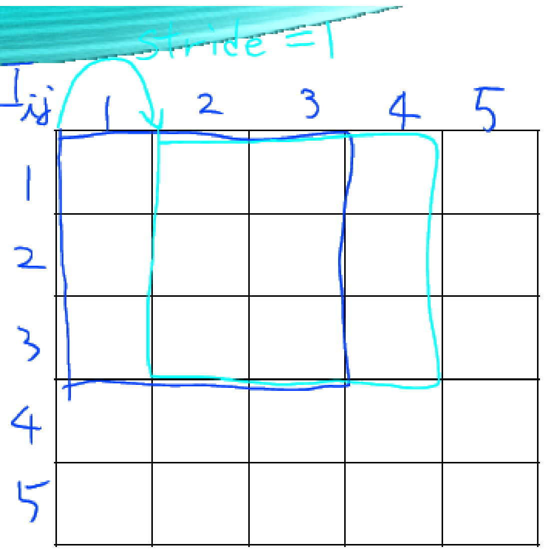
Key Computations in AI Visual Algorithms

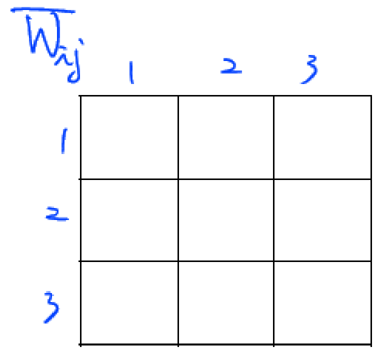
- Convolution: to extract features
- Pooling: down sampling with key info kept
- Activation (ReLU): to add non-linearity
- Residual layer: make deep NN convergent

Key Computations in AI Visual Algorithms

- **Convolution: to extract features**
- **Pooling: down sampling with key info kept**
- **Activation (ReLU): to add non-linearity**
- **Residual layer: make deep NN convergent**

Convolution





$$I_{11} \cdot \overline{W}_{11} + I_{12}\overline{W}_{12} + I_{13}\overline{W}_{13} +$$
$$I_{21}\overline{W}_{21} + I_{22}\overline{W}_{22} + I_{23}\overline{W}_{23} +$$
$$I_{31}\overline{W}_{31} + I_{32}\overline{W}_{32} + I_{33}\overline{W}_{33} = O_{11}$$

stride = 1

$$I_{12}\overline{W}_{11} + I_{13}\overline{W}_{12} + I_{14}\overline{W}_{13} +$$
$$I_{22}\overline{W}_{21} + I_{23}\overline{W}_{22} + I_{24}\overline{W}_{23} +$$
$$\underline{\hspace{10em}} = O_{12}$$

Feature map

↑

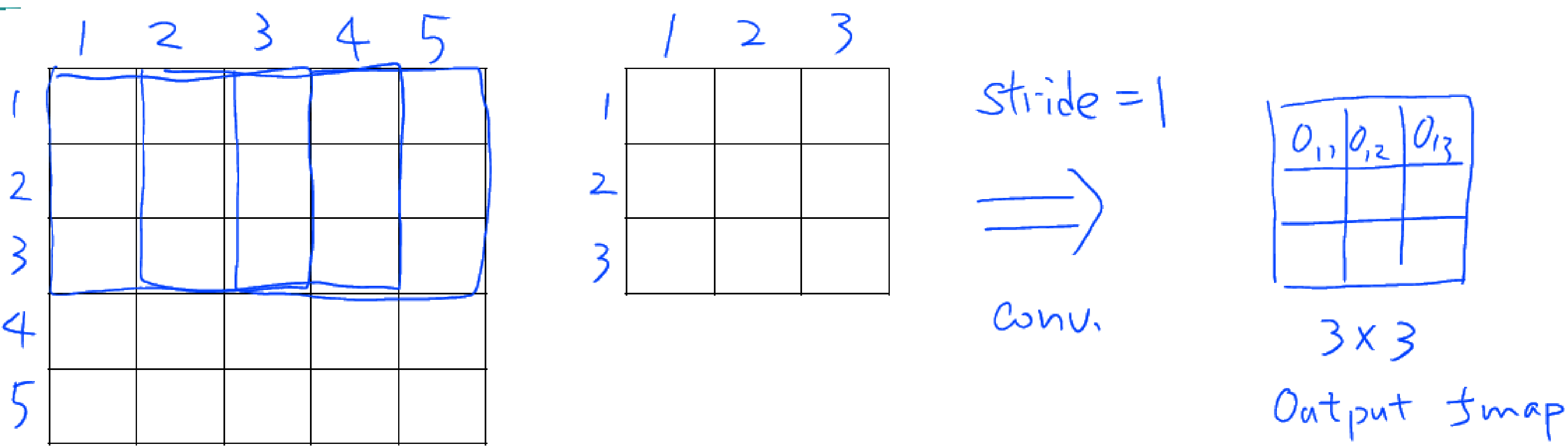
Input

Weight

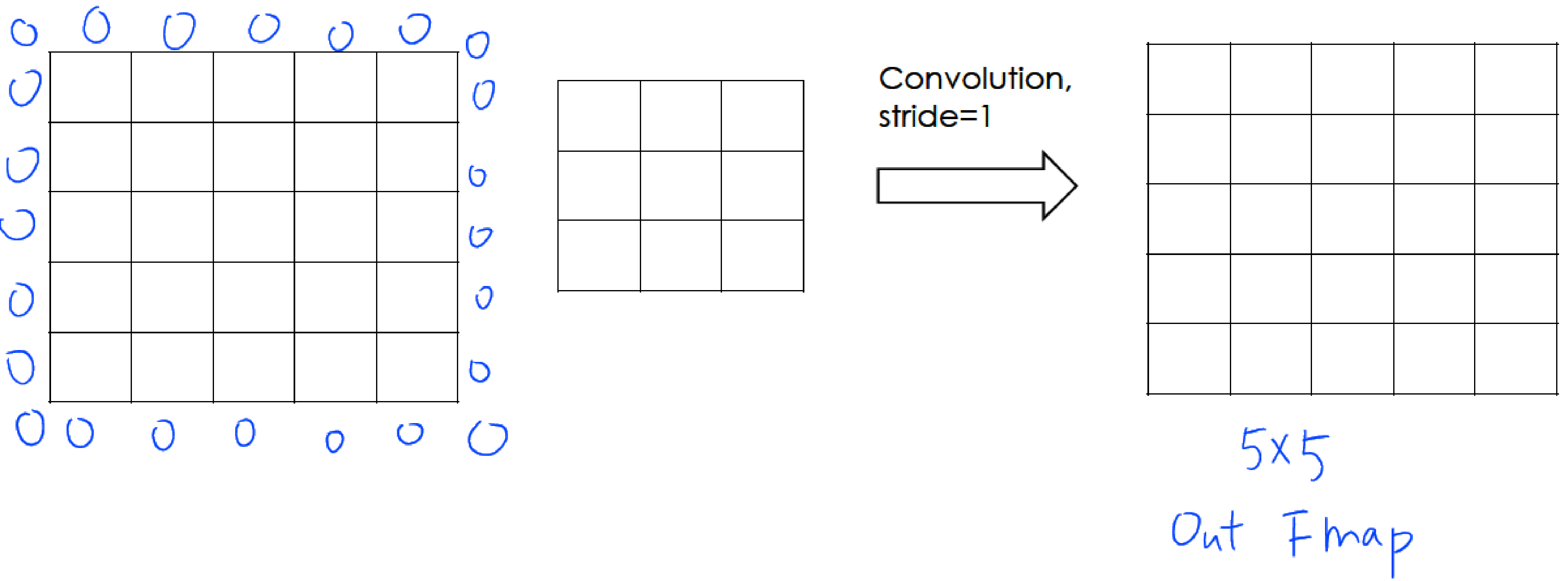
↑

Determined by training

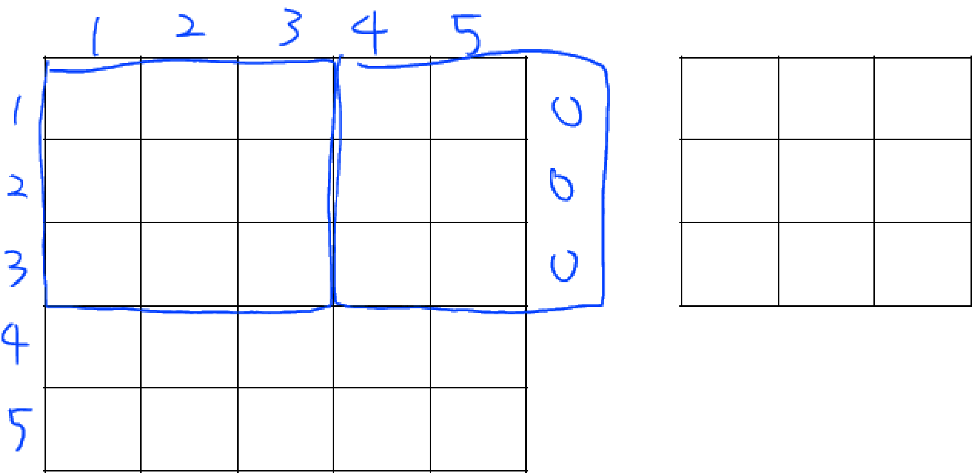
Convolution



Zero Padding



Stride More Than One

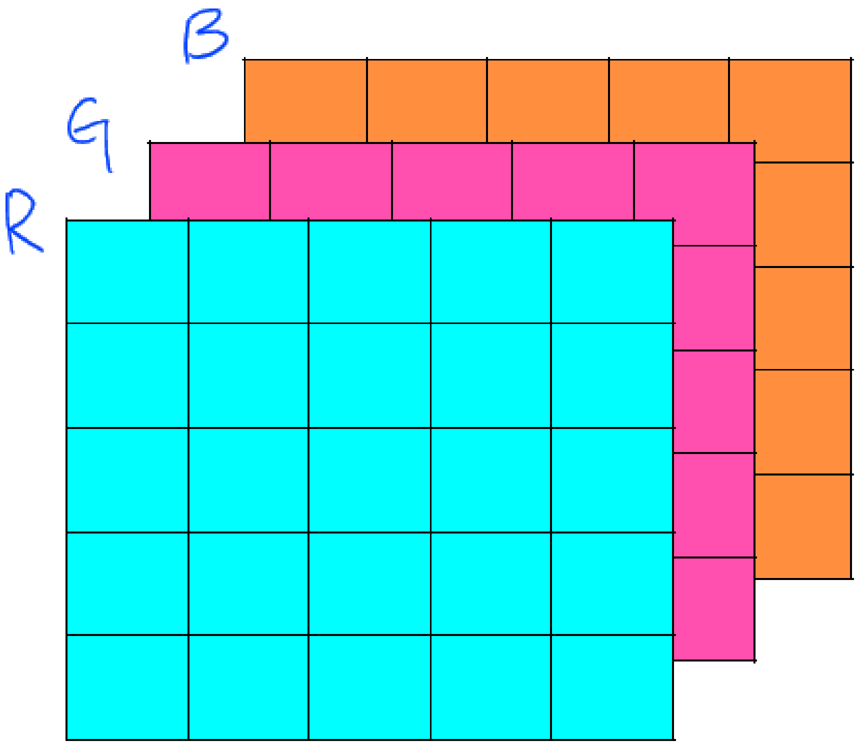


Stride = 3

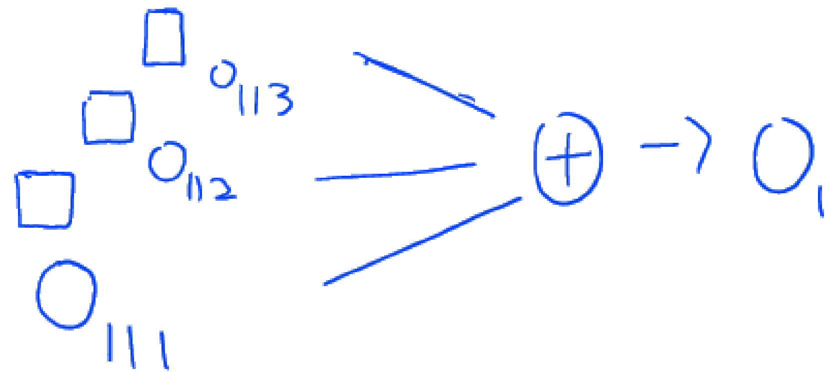
O_{11} the same as stride = 1

$O_{12} = I_{14} \cdot \overline{W_{11}} + I_{15} \cdot \overline{W_{12}} + I_{16} \overline{W_{13}} +$

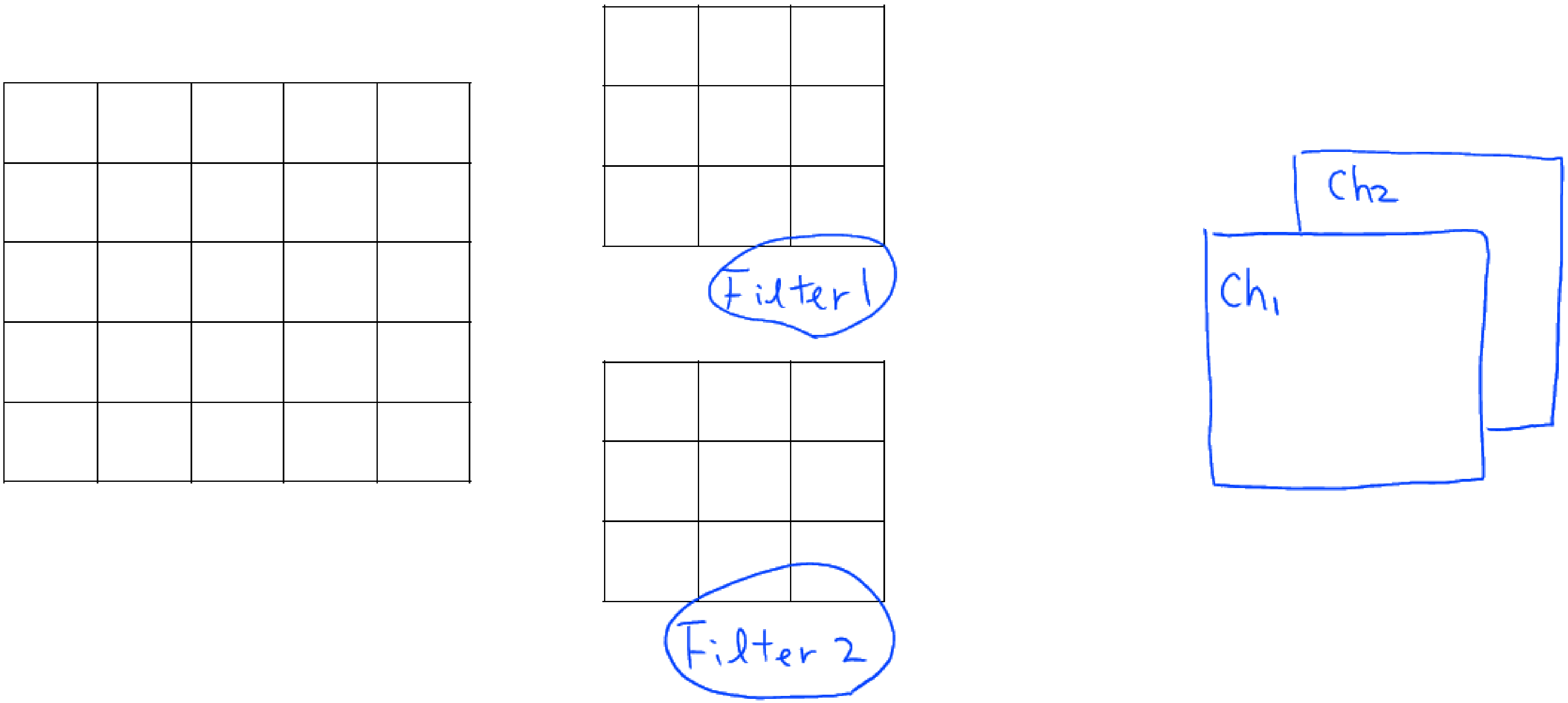
Multi-Channels



Color image



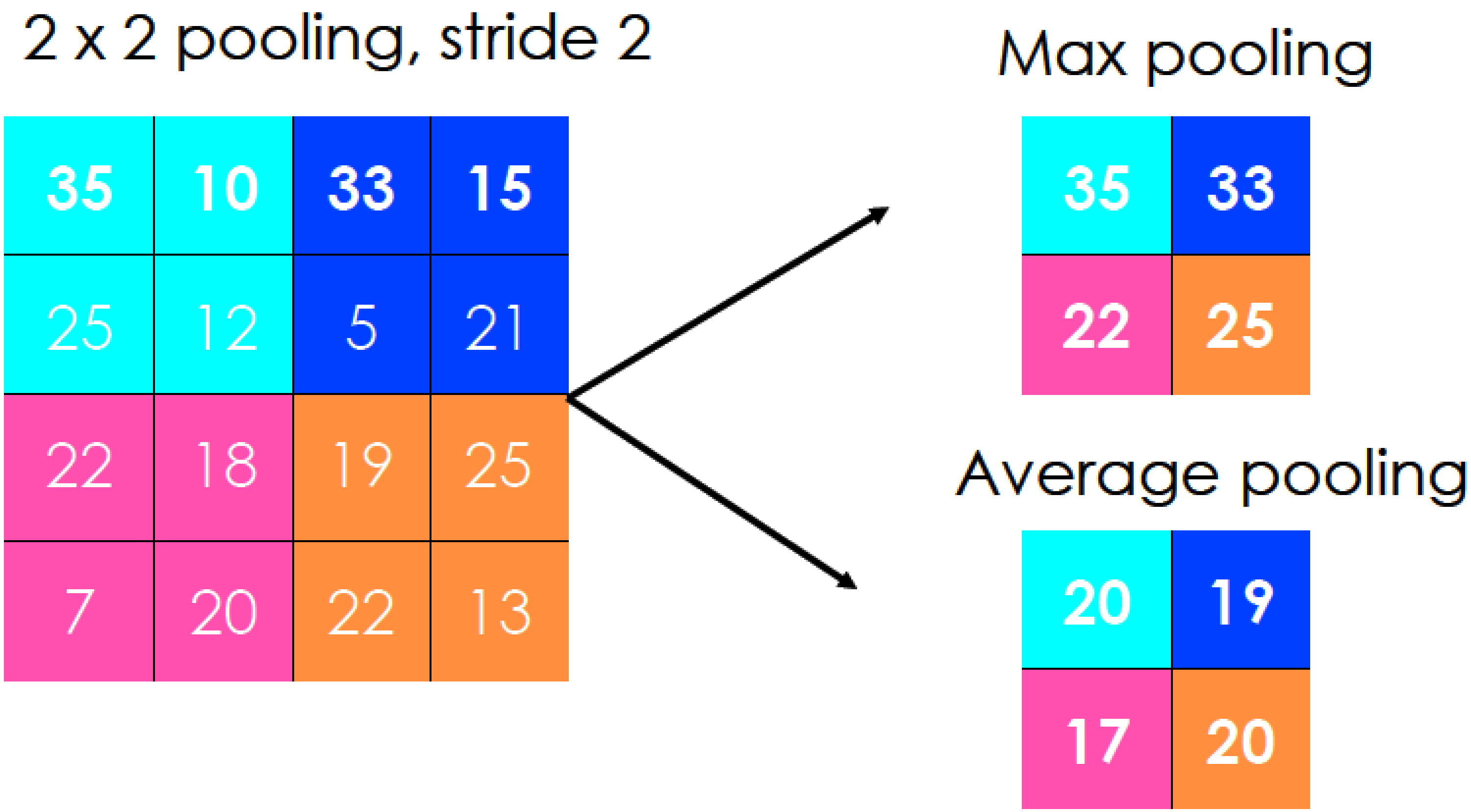
Multi-Filters



Key Computations in AI Visual Algorithms

- Convolution: to extract features
- Pooling: down sampling with key info kept
- Activation (ReLU): to add non-linearity
- Residual layer: make deep NN convergent

Pooling (Pool) Layer



Pool Layer Implementation

```
for (n=0;n<N;n++){
    for (m=0;m<M;m++){
        for (x=0;x<F;x++){
            for (y=0;y<E;y++){
                max =   -Inf;
                for (i=0;i<R;i++){
                    for (j=0;j<S;j++){
                        if(I[n][m][Ux+i][Uy+j]>max){
                            max =I[n][m][Ux+i][Uy+j];
                        }
                    }
                }
                O[n][m][x][y]   =   max;
            }
        }
    }
}
```

搜尋max pool值

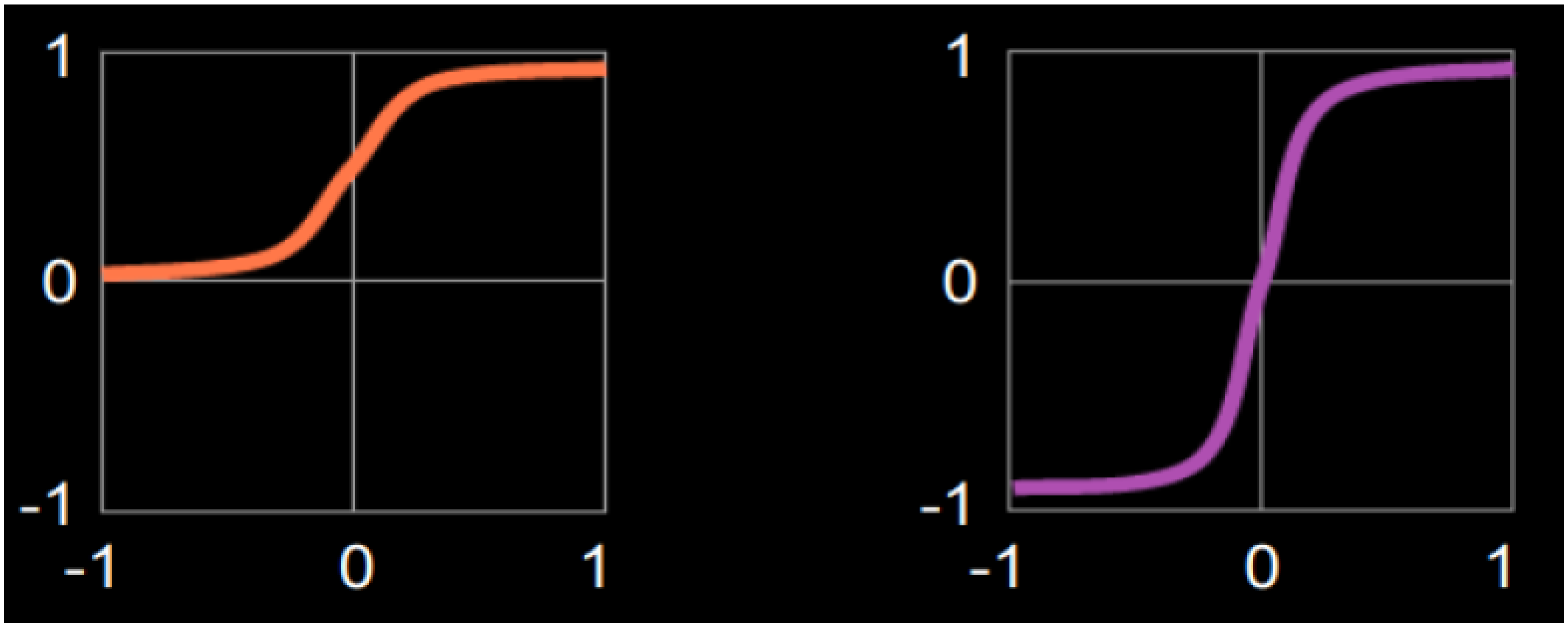
Key Computations in AI Visual Algorithms

- Convolution: to extract features
- Pooling: down sampling with key info kept
- **Activation (ReLU): to add non-linearity**
- Residual layer: make deep NN convergent

Traditional Activation Functions

Sigmoid

Hyperbolic Tangent



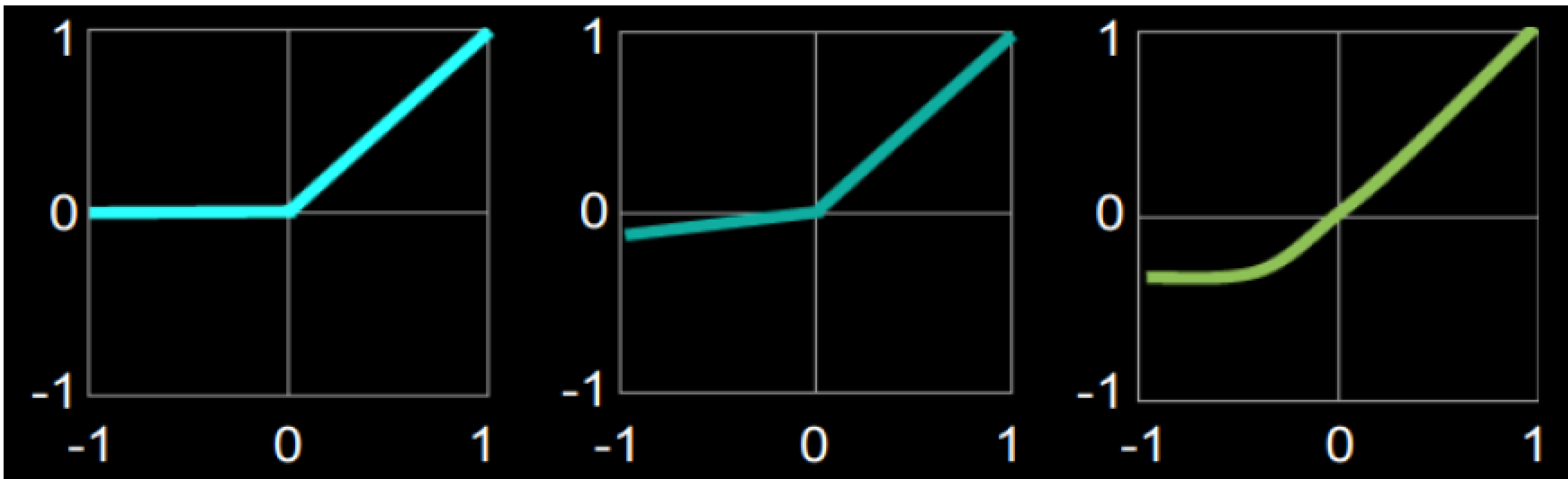
$$y = 1/(1+e^{-x})$$

$$y = (e^{x} - e^{-x})/(e^{x} + e^{-x})$$

Yu-Hsin Chen, Vivienne Sze, Joel Emer, "Hardware Architectures for Deep Neural Networks", ISCA, October 16, 2016

Modern Activation Functions

Rectified Linear Unit (ReLU) | Leaky ReLU | Exponential LU



$$y = \max(0, x)$$

$$y = max(ax, x)$$

a = small const

$$y = \begin{cases} x \\ \alpha(e^{x} - 1) \end{cases}$$

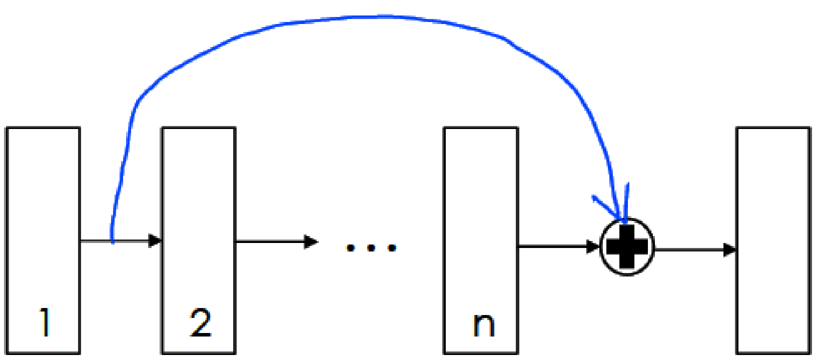
Yu-Hsin Chen, Vivienne Sze, Joel Emer, "Hardware Architectures for Deep Neural Networks", ISCA, October 16, 2016

Key Computations in AI Visual Algorithms

- Convolution: to extract features
- Pooling: down sampling with key info kept
- Activation (ReLU): to add non-linearity
- **Residual layer: make deep NN convergent**

Residual Layer

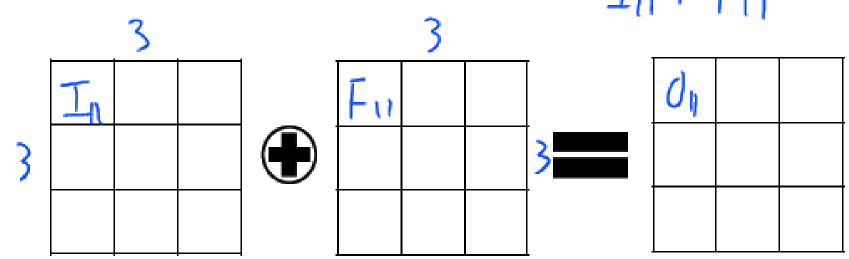
- **Also named shortcut or skip connection**
- **To solve the gradient vanishing problem**



: element-wise addition or concatenation

$$I_{ij} + F_{ij} = O_{ij}$$

$$I_{11} + F_{11} = O_{11}$$



The same dimension and channel numbers

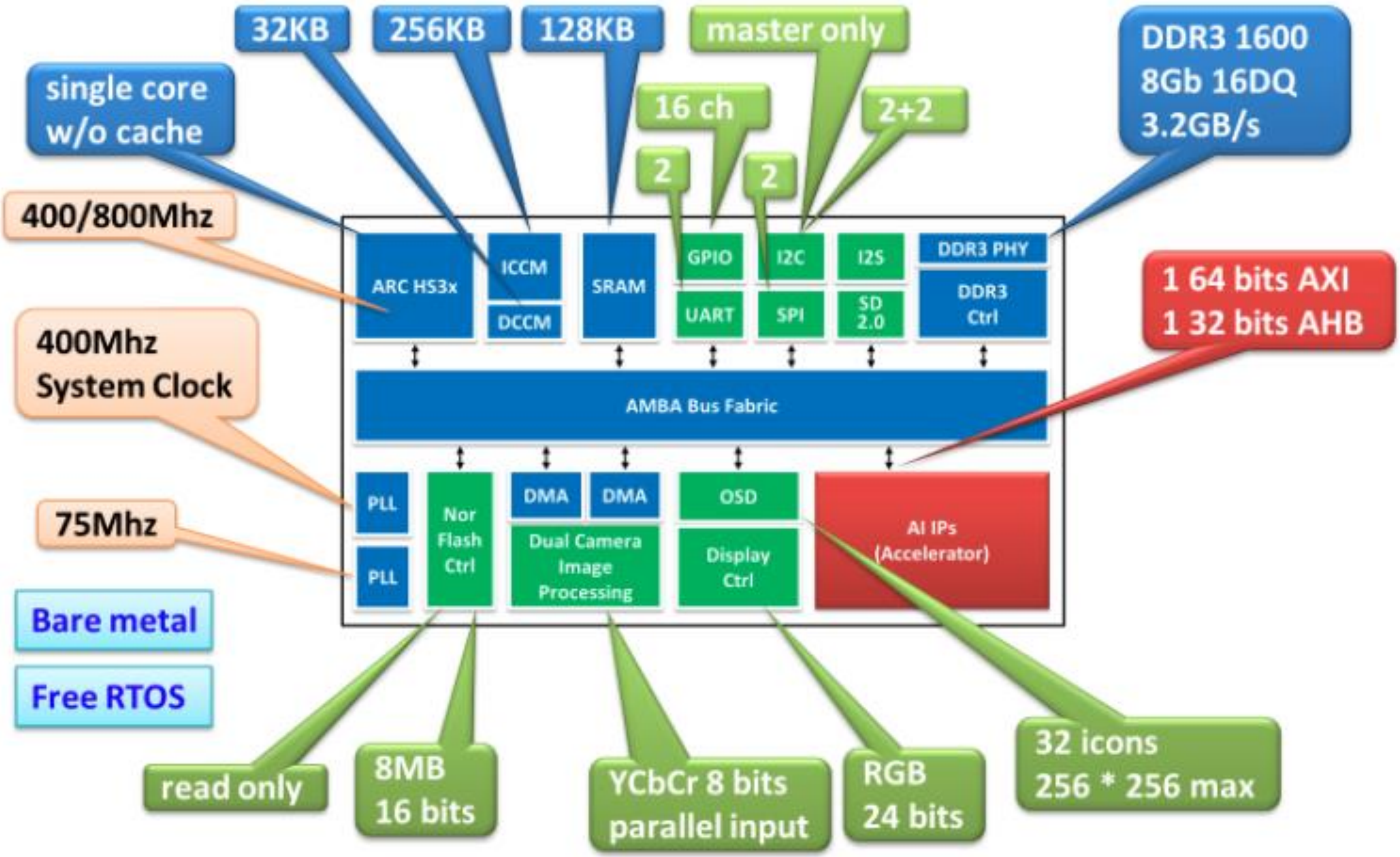
The same dimension number, but channel numbers are not necessary the same

WHAT ARE MAJOR CONCERNS FOR AN AI VISUAL COMPUTING SYSTEM?

A Computing System

- **Mainly composed of processing element (PE), cache/SRAM/on-chip buffer, DRAM, and peripherals.**
- **A PE is composed of one multiplier and one adder, or sometimes one multiplier and accumulator (MAC), several registers, and a local controller.**

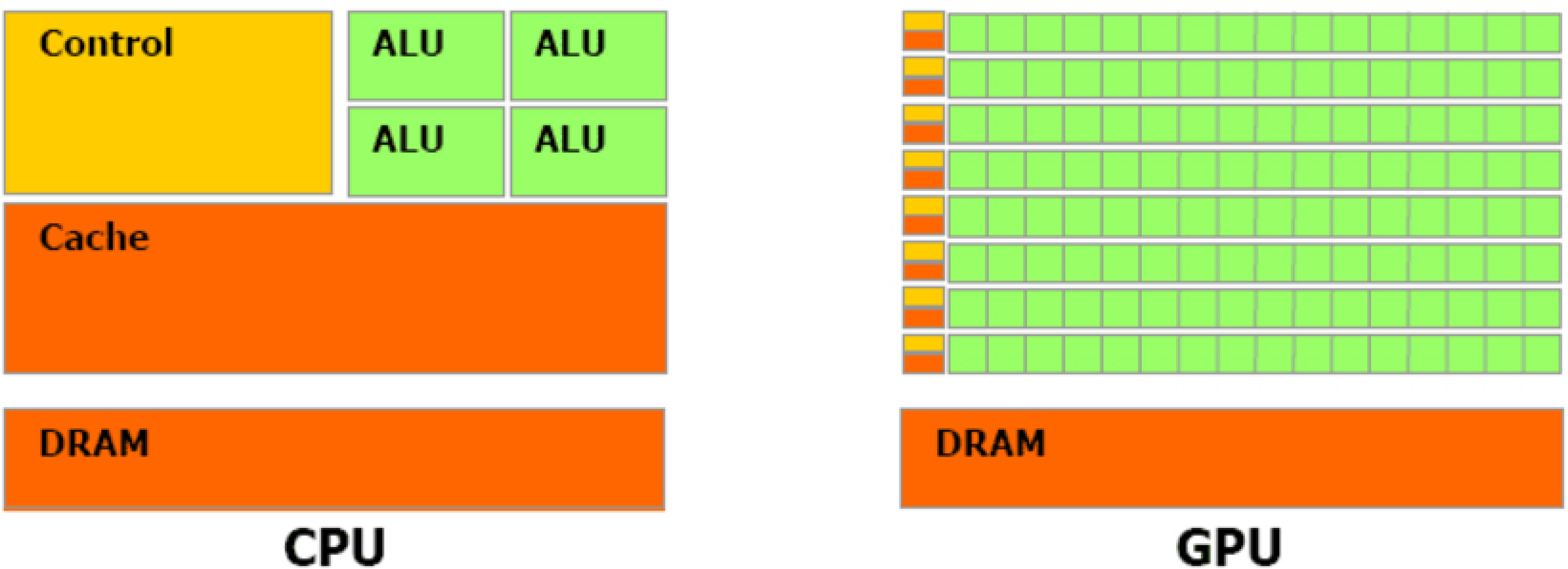
AI SOC Design Platform of TSRI



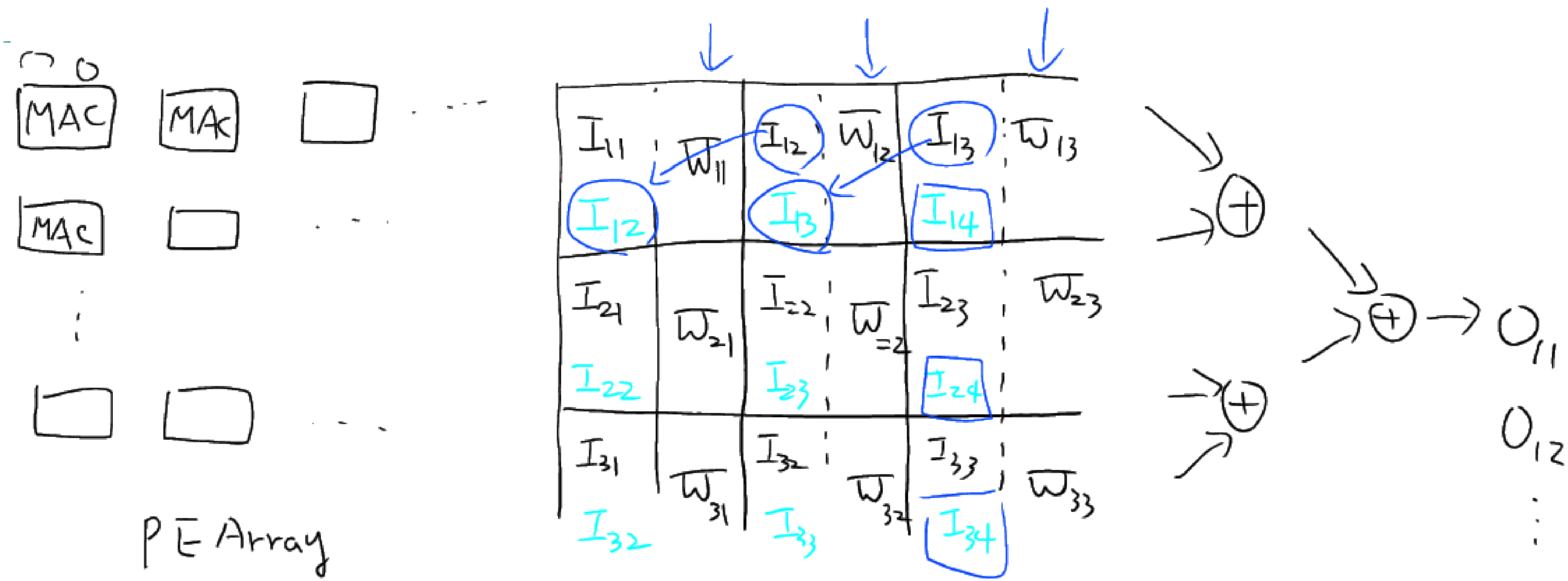
IEH

Parallel Computation

- **Utilize data independency**
  - **Convolution operation in AI visual networks**



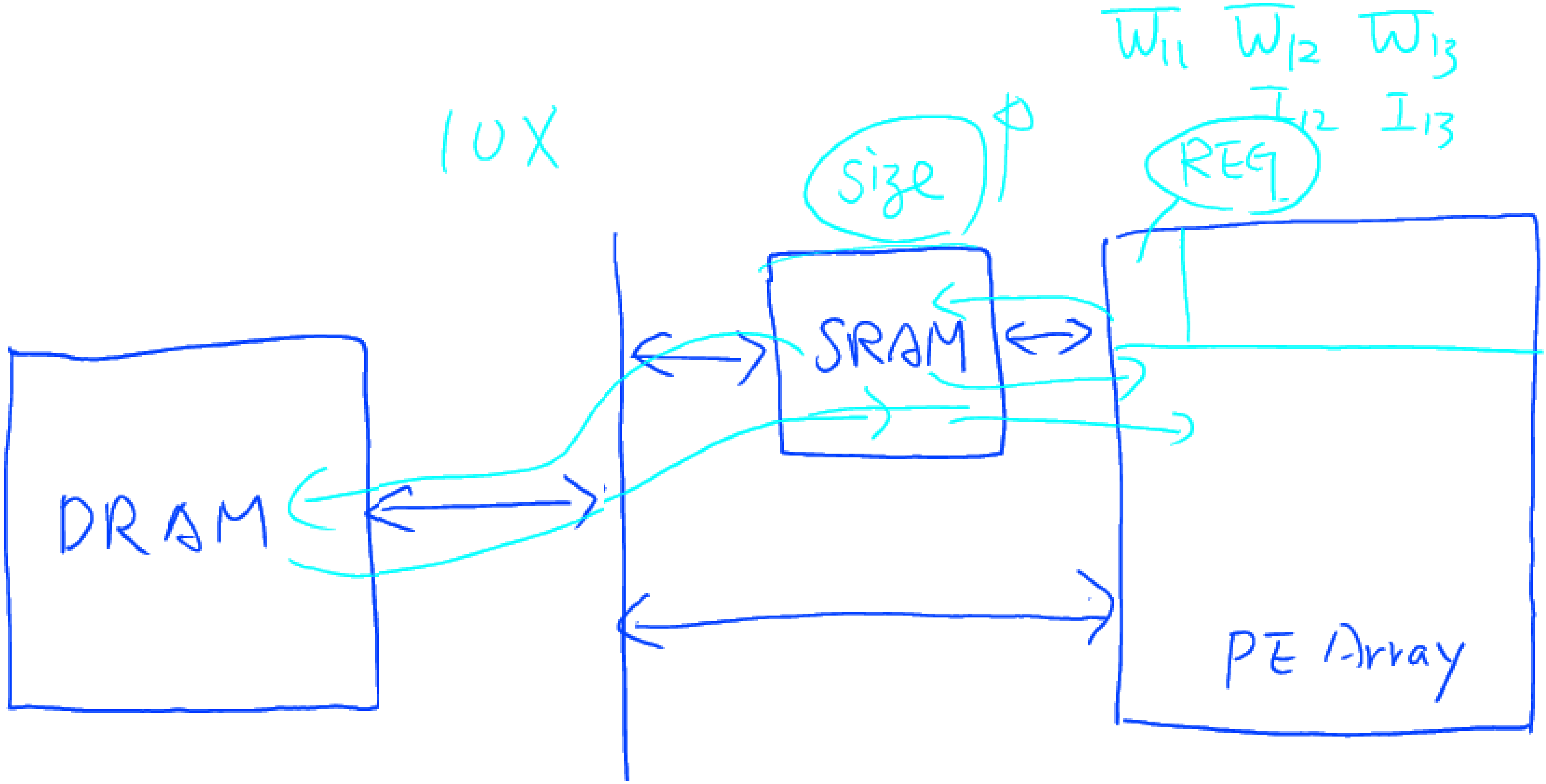
Parallel Computation



Data Reuse

- To lower down the DRAM transfer burden, and so that to accelerate the AI computation and save energy consumption.
- Need local storage. The size of local buffer versus the data reuse ratio is an important design trade-off.

Data Reuse



DRAM Bandwidth (DBW)

- **DRAM: Dynamic Random Access Memory, be used as a massive data storage in computing systems due to its low-cost merit.**

BASIS FOR COMPARISON	SRAM	DRAM
Speed	Faster	Slower
Size	Small	Large
Cost	Expensive	Cheap
Used in	Cache memory	Main memory
Density	Less dense	Highly dense
Construction	Complex and uses transistors and latches.	Simple and uses capacitors and very few transistors.
Single block of memory requires	6 transistors	Only one transistor.
Charge leakage property	Not present	Present hence require power refresh circuitry
Power consumption	Low	High

DBW Requirement for Computing CNNs

- Because of the limited on-chip cache size, data such as IFMs, weights, and OFMs are necessary to be moved between DRAM and SRAM. This forms the DRAM bandwidth requirement.
- Taking Agilev3 for example, the data transferred between DRAM and SRAM for 30 fps of 416*416 input image resolution may be as high as 3.02 GB/s based on that 72kB kernel SRAM and 169kB IFM SRAM are equipped on-chip.

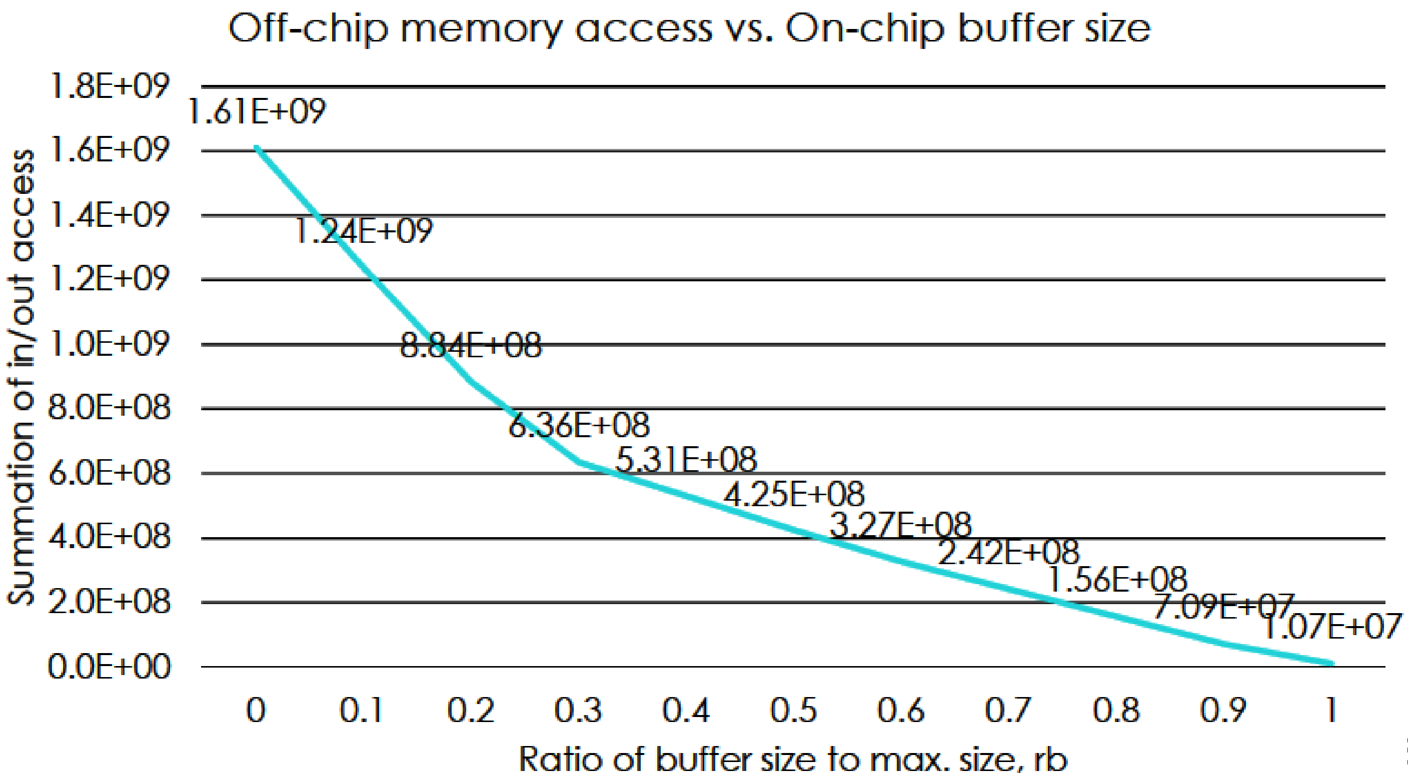
Total Available DBW

- The total bandwidth is the product of
  - **Base DRAM clock frequency**
  - **Number of data transfers per clock**: Two, in the case of double data rate (DDR, DDR2, DDR3, DDR4) memory.
  - **Memory bus (interface) width**: Each DDR, DDR2, or DDR3 memory interface is 64 bits wide. Those 64 bits are sometimes referred to as a line.
  - **Number of interfaces**: Modern personal computers typically usetwo memory interfaces (dual-channel mode) for an effective 128-bit bus width.
- For example, a computer with dual-channel memory and one DDR2-800 module per channel running at 400 MHz would have a theoretical maximum memory bandwidth of:

400,000,000 clocks per second × 2 lines per clock × 64 bits per line × 2 interfaces = 102,400,000,000 (102.4 billion) bits per second (in bytes, 12,800 MB/s or 12.8 GB/s)

Limited On-Chip Buffer Size Effect

- **NN: YOLO v2**
- **Input image resolution: 224*224**
- **Note**
  - **The data is for only a single frame**



Suggested DRAM Type

Width	Height	fps	DBW needed (GB/s) Original/optimized	Suggested DRAM type
416	416	30	3.02/1.93*	DDR-400, PC-3200
1080	720	30	13.57/8.67*	DDR4-2400, PC4-19200/DR3-1600, PC3-12800
1920	1080	30	36.19/23.13*	No available DRAM/DDR4-3200, PC4-25600

* DBW of integer AgileNet, a light-weight CNN for mobile use.

Names	Memory clock	I/O bus clock	Transfer rate	Theoretical bandwidth
DDR-200, PC-1600	100 MHz	100 MHz	200 MT/s	1.6 GB/s
DDR-400, PC-3200	200 MHz	200 MHz	400 MT/s	3.2 GB/s
DDR2-800, PC2-6400	200 MHz	400 MHz	800 MT/s	6.4 GB/s
DDR3-1600, PC3-12800	200 MHz	800 MHz	1600 MT/s	12.8 GB/s
DDR4-2400, PC4-19200	300 MHz	1200 MHz	2400 MT/s	19.2 GB/s
DDR4-3200, PC4-25600	400 MHz	1600 MHz	3200 MT/s	25.6 GB/s

Mobile AI Platforms

Platform	CPU	GPU	Performance	DBW (GB/s)
Jetson Nano	4 cortex A57	128 CUDA cores	472 GOPs	25.6
Jetson TX2	2 Denver cores and 4 cortex A57	256 pascal gpu cores	1.33 TOPs	59.7
Jetson AGX Xavier	8 Carmel cores and ARM 8.2 64b CPU	512 volta gpu cores with 64 tensor cores	32 TOPs	136.5

Suggested Platform

Width	Height	fps	Operation required (GOPS)	Network	Suggested platform
416	416	30	981	AgileV3	Jetson TX2 or Jetson Nano for 14 fps
1080	720	30	4405		Jetson TX2
1920	1080	30	11752		Jetson AGX Xavier

Width	Height	fps	Operation required (GOPS)	Network	Suggested platform
416	416	30	1560	YOLOv3	Jetson TX2/Jetson AGX Xavier
416	416	30	1803	YOLOv4	Jetson TX2/Jetson AGX Xavier

Comparisons of Detection NNs

Network	Word length	No. of conv. layers	Model size (MB)	Conv. IO* (Mega)	Required GOPS*	Year
Agilev3	FP32	43	65.39	2023.8	480	2019
YOLOv3	FP32	74	241.78	3352.5	980	2018
YOLOv4	FP32	109	251.15	4795.8	900	2020
YOLOv4-tiny	FP32	21	23.10	707.1	100	2020
HarDNet+SSD	FP32	88	98.04	2361.3	770	2019
YOLOv5 - s	FP16	70	14.2	1075.8	110	2020

* for 416x416 @ 30 fps

YOLOv3 (Up) vs YOLOv4 (Down)

Advanced Optimization Topic

- **Auto labeling of sample classes that are not available yet**
  - **Labelimg, Ezlabel, Ilabeler**

Pro's and Con's of Groundtruth Labeling Tools

Labeling tool	Pros	Cons
Labelimg	1. Easy to use 2. User-friendly Interface	1. Manual labeling, 2. Time-consuming for multiple images labeling 3. No video labeling.
EZLabel	1. User-friendly Interface 2. Auto image labeling 3. A cloud-based operation platform	1. No video labeling
Ilabeler-v1	1. Auto image labeling 2. Auto video labeling 3. Allow to interactively update image labeling result.	1. On-site computer operation only 2. Need sufficient computing resource