

AI Computing Systems Backgrounds and ML Basics

Instructor: Siting Liu

School of Information Science and Technology (SIST), ShanghaiTech University

Shanghai Institute of Semiconductor Innovation (SISI)

<https://faculty.sist.shanghaitech.edu.cn/liust/EE219/EE219.html>



Computer Architecture - EE219
faculty.sist.shanghaitech.edu.cn

立志成才 报国裕民



What is Going on?

- Basic computation of neural networks
 - General matrix multiplication (GEMM)
- Contemporary computing hardware
- Things to consider for different computing systems
- International and domestic AI computing companies and startups.



General matrix multiplication (GEMM)

- An MLP example

```
for (int n = 0; n < N; n++) {  
    for (int k = 0; k < K; k++) {  
        OA[n][k] = 0; // Initialization  
        for (int m = 0; m < M; m++) {  
            OA[n][k] += IA[n][m] * W[m][k];  
        }  
        OA[n][k] = Activation(OA[n][k]);  
    }  
}
```

Software

Programming
language

C/C++;
OpenMP/SIMD intrinsic;
Python; CUDA
(numpy/pytorch)

Scheduling

Different
execution
order

Memory
layout

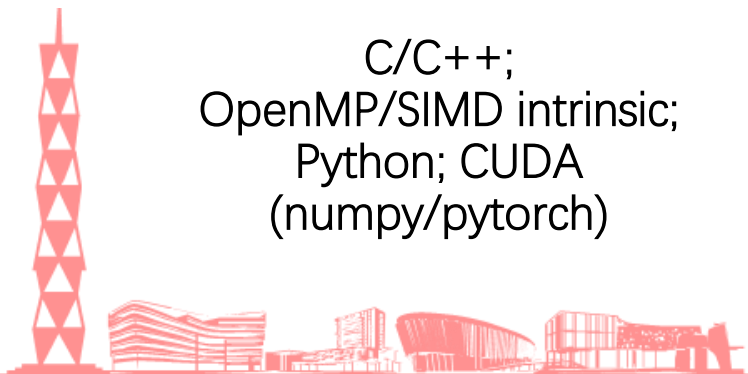
Number
representation

Memory
hierarchy

Hardware

Computing
platform

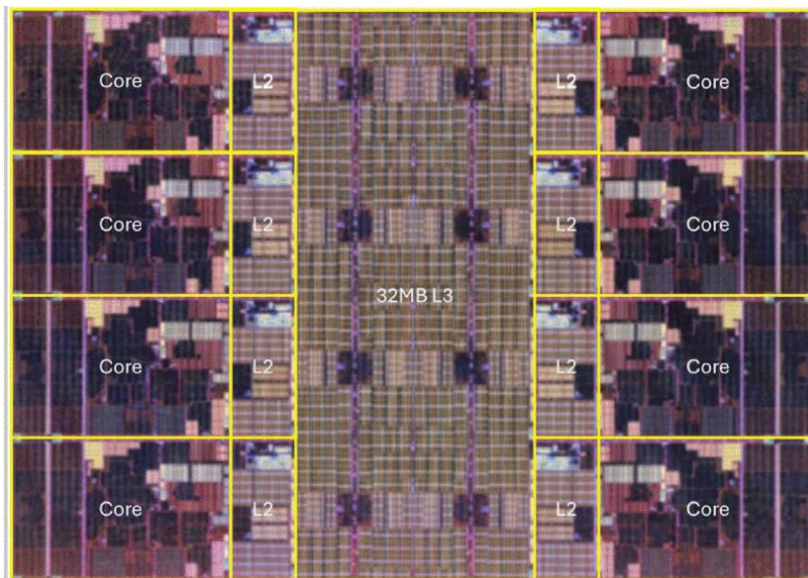
CPU; GPU;
FPGA; SoC;
ASIC; etc.



Contemporary computing hardware: an Overview



• CPU



Attribute	"Zen 4"	"Zen 5"
L1/L2 BTB	1.5K/7K	16K/8K
Return Address Stack	32	52
ITLB L1/L2	64/512	64/2048
Fetch/Decoded Instruction Bytes/Cycle	32	64
Op Cache Associativity	12-way	16-way
Op Cache Bandwidth	9 macro-ops	12 inst or fused inst
Dispatch Bandwidth (Macro-Ops/Cycle)	6	8
ALU/AGU	4/3	6/4
Integer PRF (Reg/Flag)	224/126	240/192
Vector PRF	192	384
FP Pre-Scheduler Queue	64	96
FP Scheduler	2x32	3x38
FP Pipes	3	4
Vector Width	256b	256b/512b
ROB/Retire Queue	320	448
LS Mem Pipes Supporting Loads/Stores	3/1	4/2
DTLB L1/L2	72/3072	96/4096
L1 Data Cache	32KB/8-way	48KB/12-way
L2 per Core	1MB/8-way	1MB/16-way
L2 Bandwidth	32B/clock	64B/clock

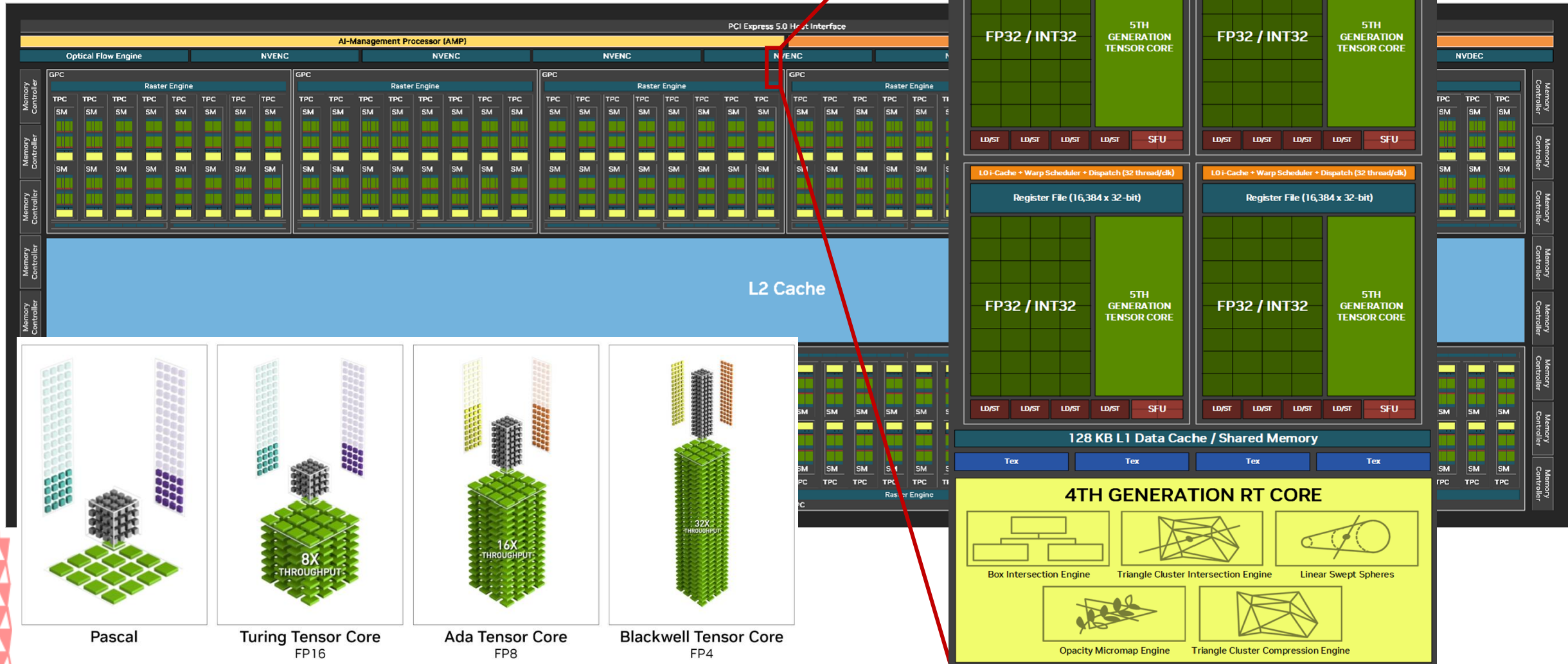
Teja Singh et al., "Zen 5": The AMD High-Performance 4nm x86-64 Microprocessor Core," ISSCC, San Francisco, CA, USA, 2025.

The diagram illustrates the L3 cache architecture in a 3D perspective. It shows a top layer labeled "8 MB V-Cache Slice" and a bottom layer labeled "4 MB L3 Cache Slice". The bottom layer is divided into a grid of "32B Per Core" blocks. A central "L3 Fabric" is shown with "32 Lines x 2 Rung's". Red arrows indicate data flow between the V-Cache and the L3 Cache. A red box highlights a "4 MB L3 Cache Slice" within the bottom layer.

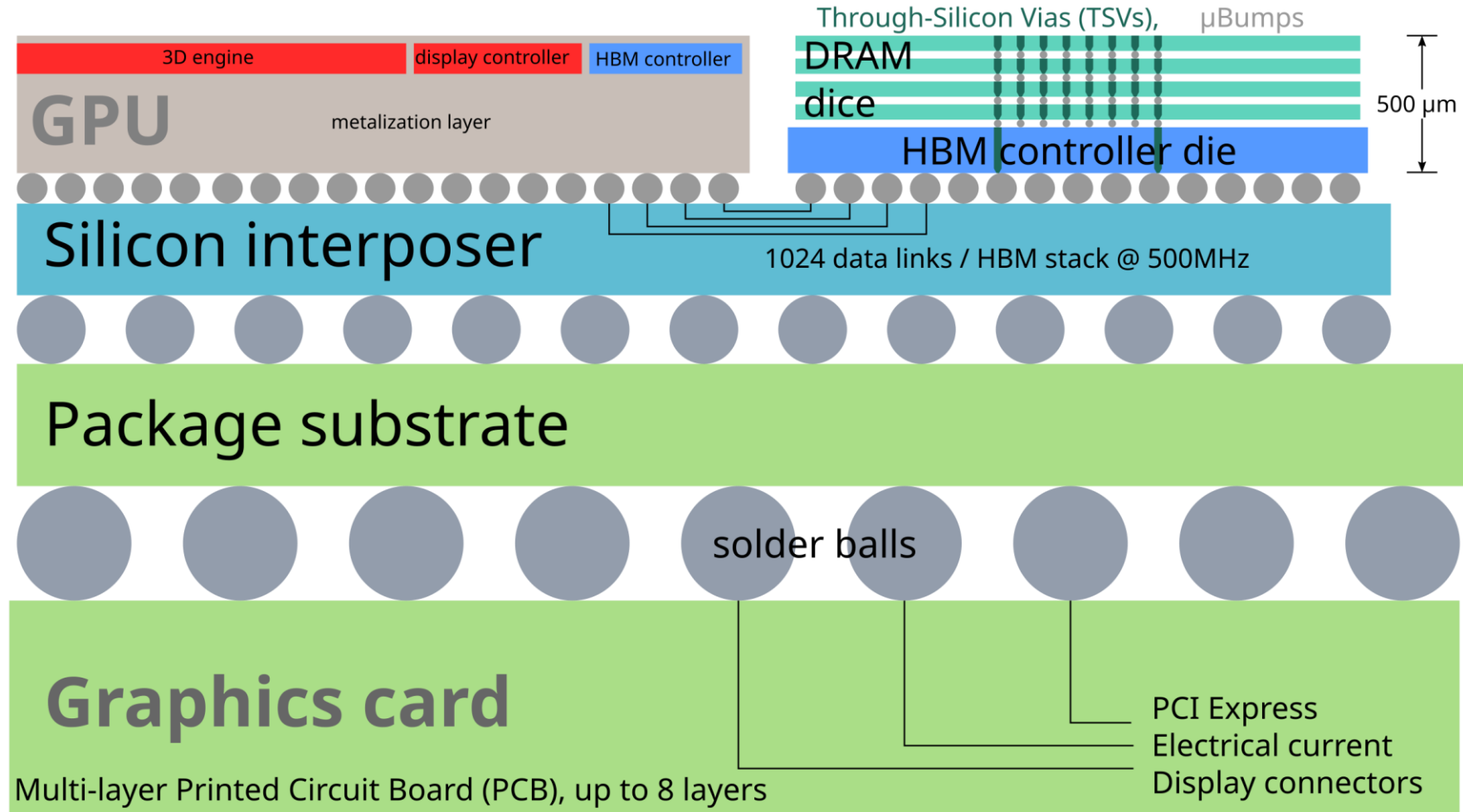
A black and white photograph of a film strip. The film strip is oriented vertically. On the left edge, the words "THE END OF THE LINE" are printed in a bold, sans-serif font, repeated multiple times. On the right edge, the name "W. J. WILSON" is printed in a similar bold, sans-serif font, also repeated multiple times. The film strip itself shows a series of frames, though the individual images within the frames are not clearly discernible. The overall appearance is that of a vintage film strip.

[illegible]

GB202 Blackwell GPU



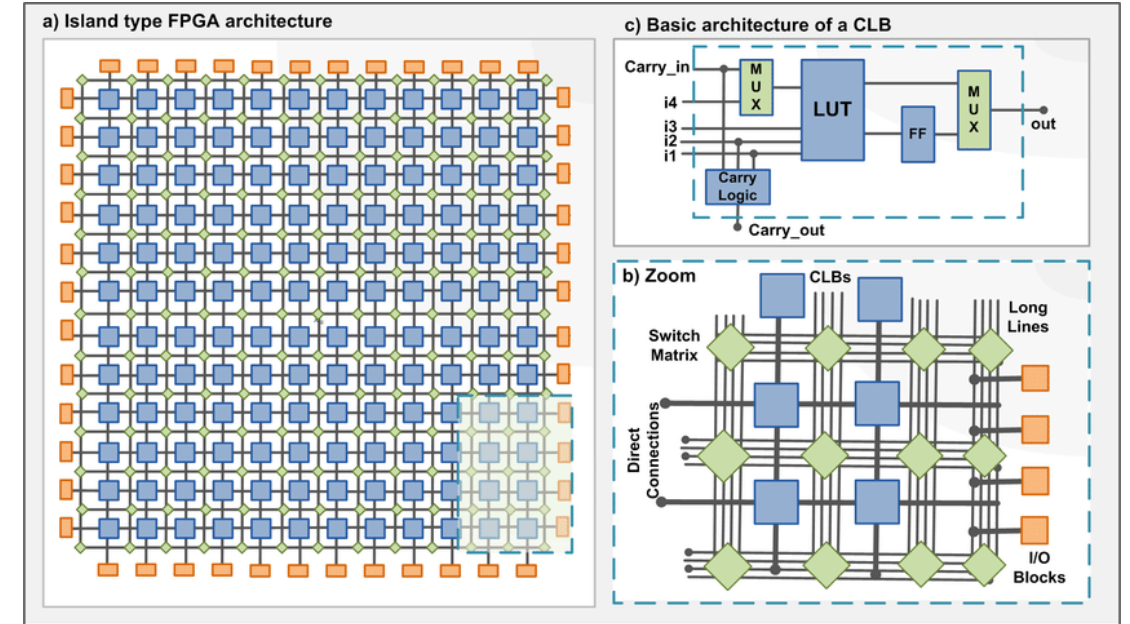
2.5D/3D Integration



https://en.wikipedia.org/wiki/High_Bandwidth_Memory

FPGA

- CLB (Configurable logic blocks) in Xilinx FPGA
 - 5/6-input LUT
 - Distributed memory and shift register logic
 - Dedicated carry logic for arith.
 - Wide multiplexer
 - Dedicated storage elements (FFs/latches)
- Adaptive logic module (ALM) block diagram Intel
 - 3/4-input LUT
 - Registers and adders
- Modern FPGAs are integrated with Block RAM, DSP, various interface, even AI accelerator, etc.
- Support hardware description language, less for (C, Python, etc.)



Generic FPGA hardware architecture [3]

[1] Xilinx. UltraScale Architecture Configurable Logic Block (User Guide).

[2] Altera. FPGA Architecture White Paper.

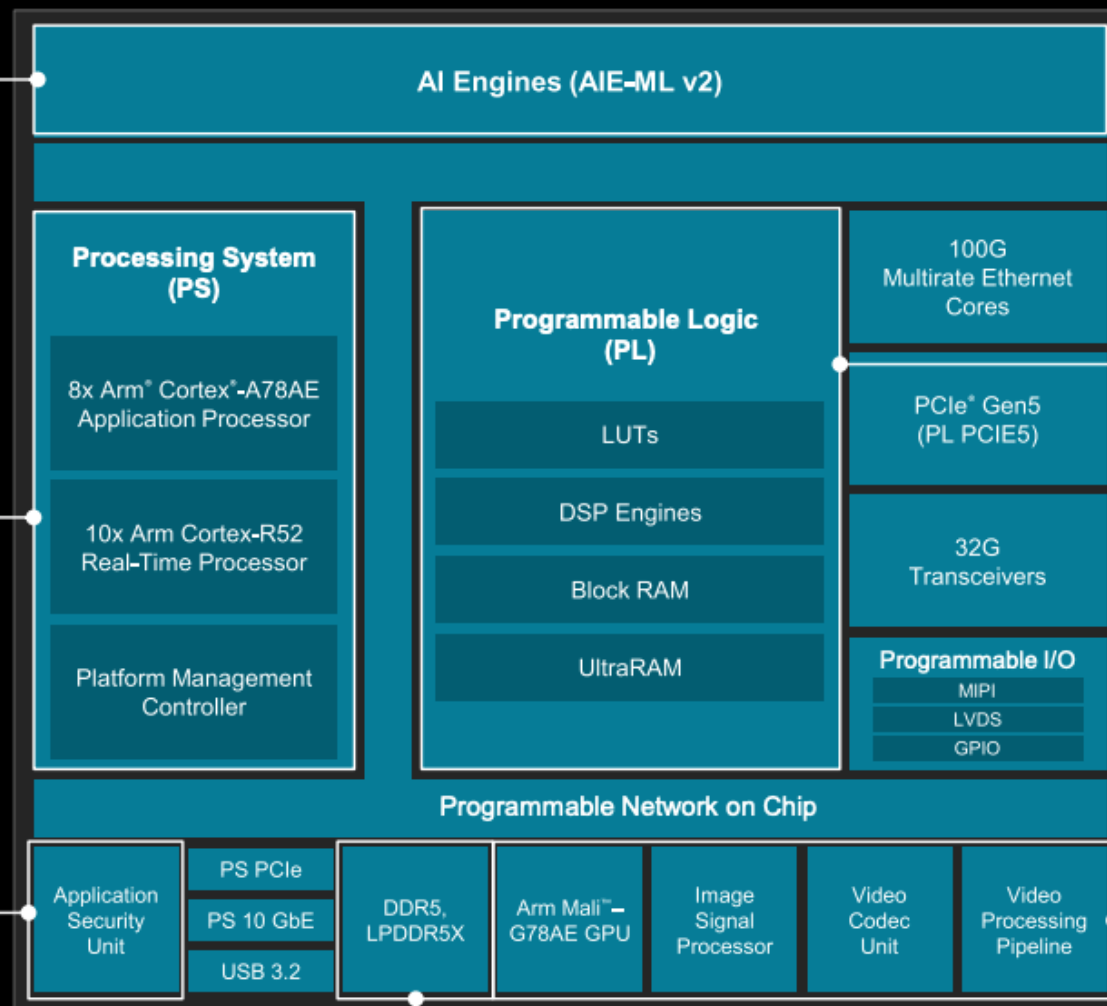
[3] https://www.researchgate.net/publication/265125404_A_High-level_Methodology_for_Automatically_Generating_Dynamically_Reconfigurable_Systems_using_IP-XACT_and_the_UML_MARTE_Profile/figures?lo=1

AMD Versal™ AI Edge Series Gen 2 Overview

**Next-Generation AI Engines
for Efficient AI Inference**
Up to 3X TOPs/Watt*

**High-Performance Integrated
CPUs for Postprocessing**
Up to 10X Scalar Compute*

Enhanced Safety & Security
Support for ASIL D, SIL 3



**World-Class Programmable
Logic for Flexible, Real-Time
Preprocessing**

- Sensor Fusion, Data Conditioning
- New Hard Image/Video Processing

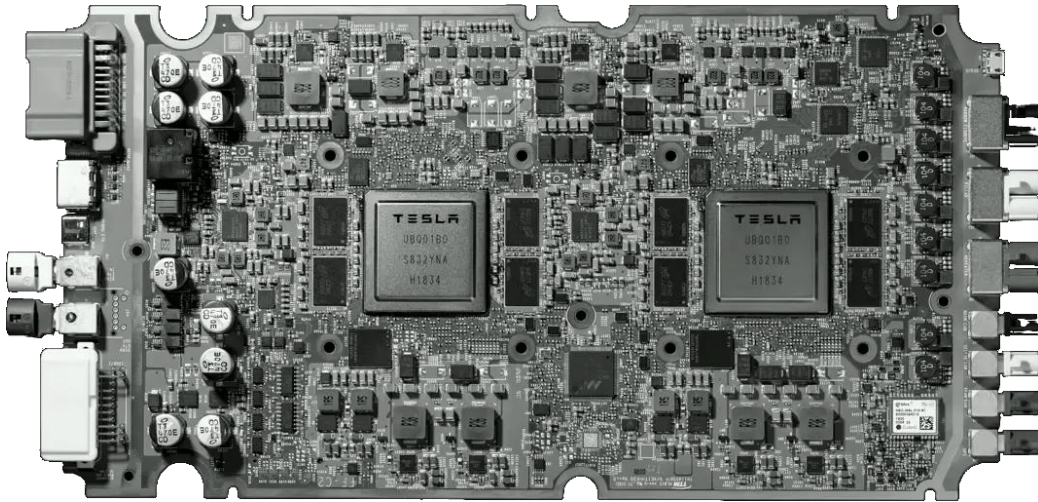
**User Selectable System Memory
Interfaces for Data Intensive
Accesses**

- Up to 170 GBytes/sec Bandwidth*

ASICs

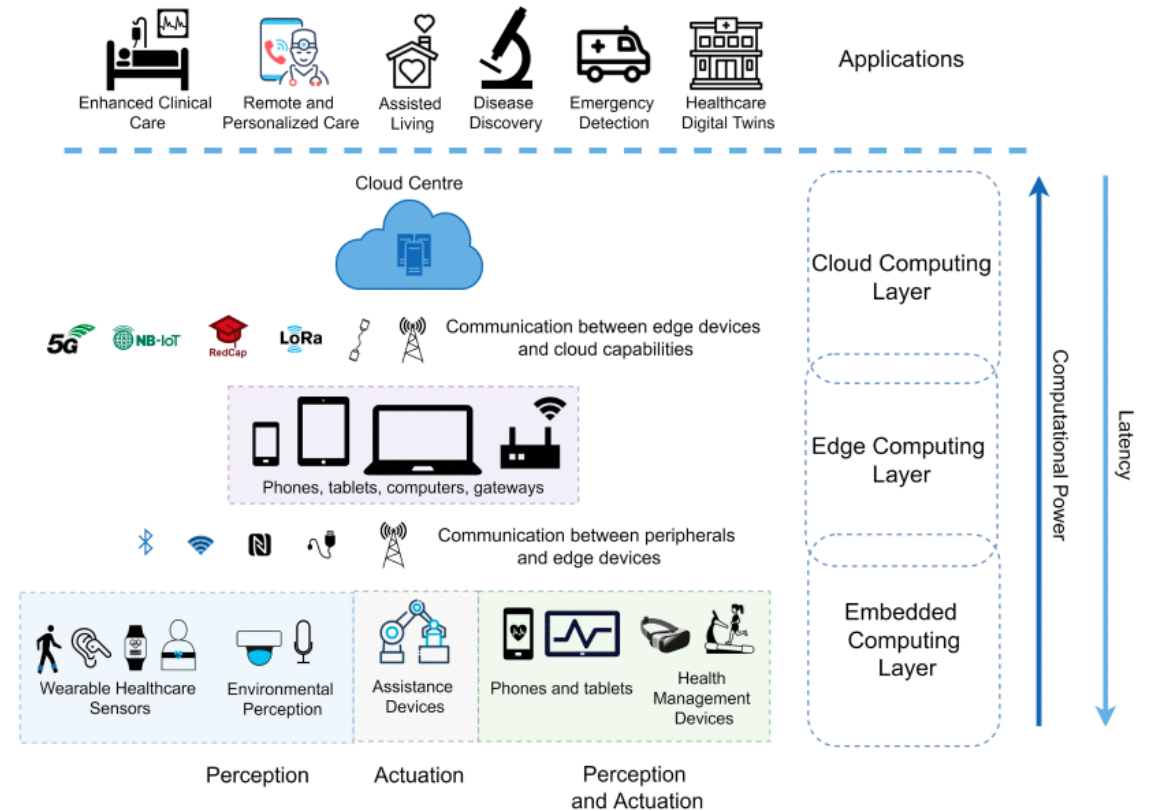
- Pervasive AI
- Autonomous driving, smartphone, etc.
- Wearables (smart watch, implanted medical devices, etc.)

Energy constraint



From Wikichip.

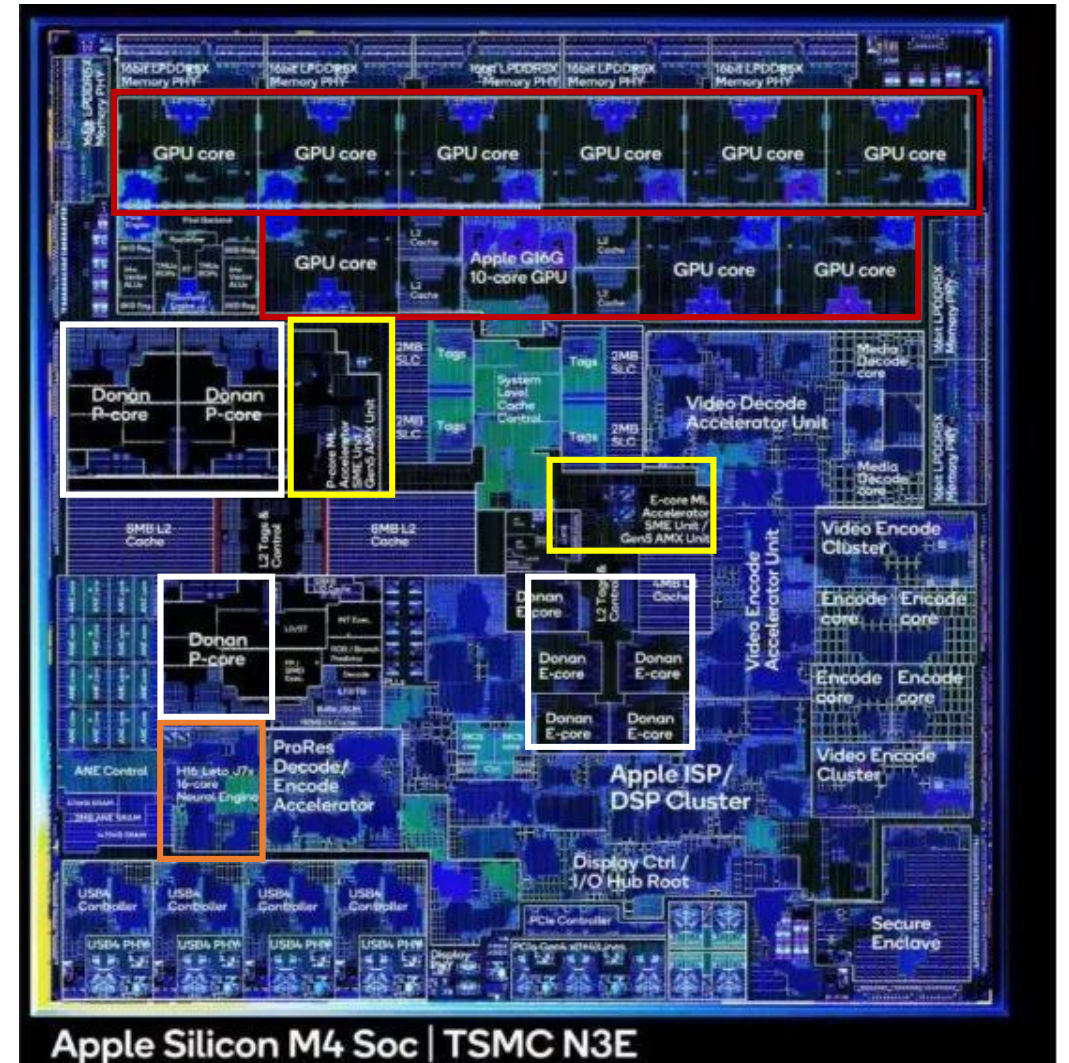
S. Baker and W. Xiang, "Artificial Intelligence of Things for Smarter Healthcare: A Survey of Advancements, Challenges, and Opportunities," in IEEE Communications Surveys & Tutorials, vol. 25, no. 2, pp. 1261-1293, Secondquarter 2023



立志成才 报国裕民

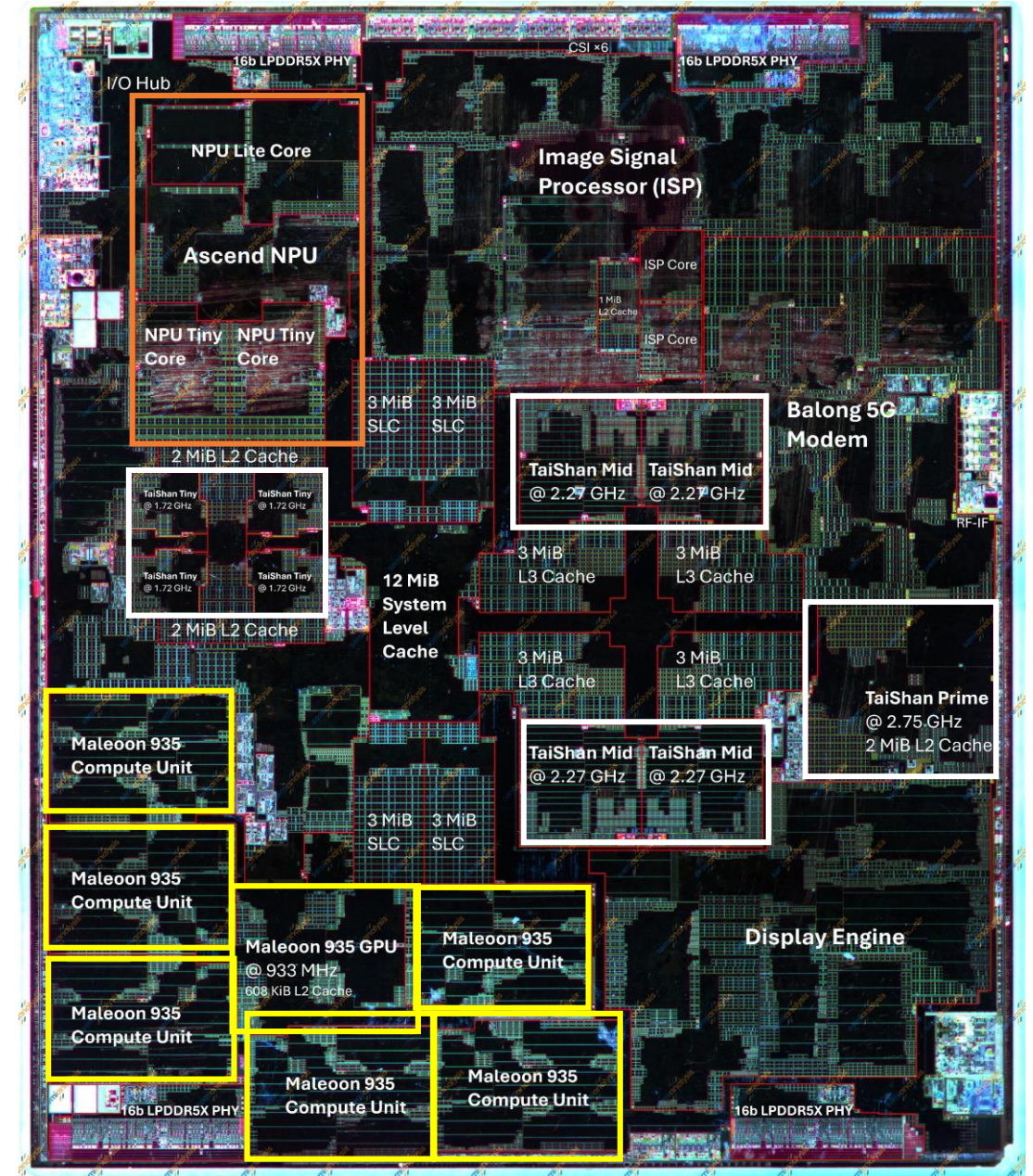
SoCs

- Highly heterogeneous
 - Mixed P-core & E-core with AMX (matrix)
 - 10-core GPU
 - 16-core NPU
 - Various hardware encoders and decoders (video, ProRes, media)
 -
- Increase the difficulty of programming



SoCs (Huawei 9030)

- Highly heterogeneous
 - Mixed Prime-core/Mid-core/Tiny-core
 - Maleoon GPU
 - Ascend NPU with different sizes
 - Balong 5G Modem
 - ISP core
 -



SoC Comparison

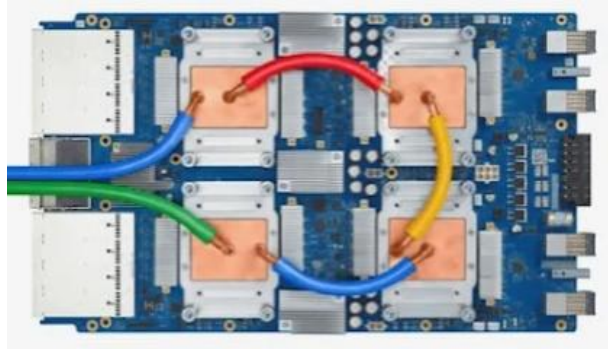
Company	Name	Process	CPU	GPU	NPU	Other features
HiSilicon	Kirin 9030	N+3	1+3+4 (2.75 GHz + 2.27 GHz + 1.72 GHz)	Maleoon GPU with 6 CUs (马良)	Ascend 1 Lite core 2 Tiny cores (昇腾)	Support LPDDR5X
Apple	M4	TSMC N3E	3/4 P-cores 4/6 E-cores	8/10-core GPU	16-core neural engine (38 TOPs)	Unified memory



ASICs

- Application-specific integrated circuit (ASIC)
 - Both on cloud and edge

Google TPU v3
420 TFLOPS
128 GB HBM



4.8 cm
with Google TPU Edge
INT8 4 TOPS
2 TOPS per watt

Atlas 300I Pro
204.8 GB/s bandwidth
140 TOPS INT8
70 TFLOPS FP16
LPDDR4X 24 GB



4 cm
Atlas 200
Ascend 310 chipset
8 W
INT8 22 TOPS
FP16 11 TFLOPS

6.5 cm

5 cm

AI IC Startups

Too many to list
[Startups in Shanghai](#)

Domestic GPU
“4 little dragon”

公司名称	核心产品/技术方向	市场定位与应用场景
瀚博半导体	VA系列：云端AI推理加速卡	云端视频处理：视频解码+AI推理，广泛应用于短视频、直播、云游戏、元宇宙等。
燧原科技	云燧：云端AI训练和推理加速卡	云端训练与推理：为数据中心提供全栈AI算力解决方案，对标NVIDIA。
地平线	征程系列：车载AI芯片 旭日系列：边缘AI芯片	智能驾驶、智能座舱、泛AIoT。
黑芝麻智能	华山系列：自动驾驶计算芯片	智能驾驶：提供基于车规级芯片的自动驾驶解决方案。
亿铸科技	基于ReRAM的存算一体AI芯片	云端大算力推理：利用存算一体技术突破传统架构的“内存墙”瓶颈，能效比高。
芯翼信息	NB-IoT芯片、AIoT SoC	物联网终端：将AI能力集成到物联网芯片中，用于低功耗终端设备。
肇观电子	N系列：视觉处理AI芯片	视觉AI：专注于计算机视觉，应用于安防、机器人、AR/VR、车载等。
ThinkForce	边缘/云端AI加速芯片及计算卡	边缘计算、智能安防、工业检测等。
天数智芯	通用GPU（GPGPU）	云端训练与推理：提供兼容CUDA生态的通用计算芯片，用于AI和高性能计算。
沐曦集成电路	GPU（通用图形处理器）	AI训练、高性能计算、图形渲染。
壁仞科技	通用GPU（BR系列）	AI训练与推理、高性能计算。
昆仑芯科技	AI加速卡（前身为百度芯片部）	云端AI推理与训练：最初为百度搜索等业务优化，现已独立运营。

Chaos is a Ladder!

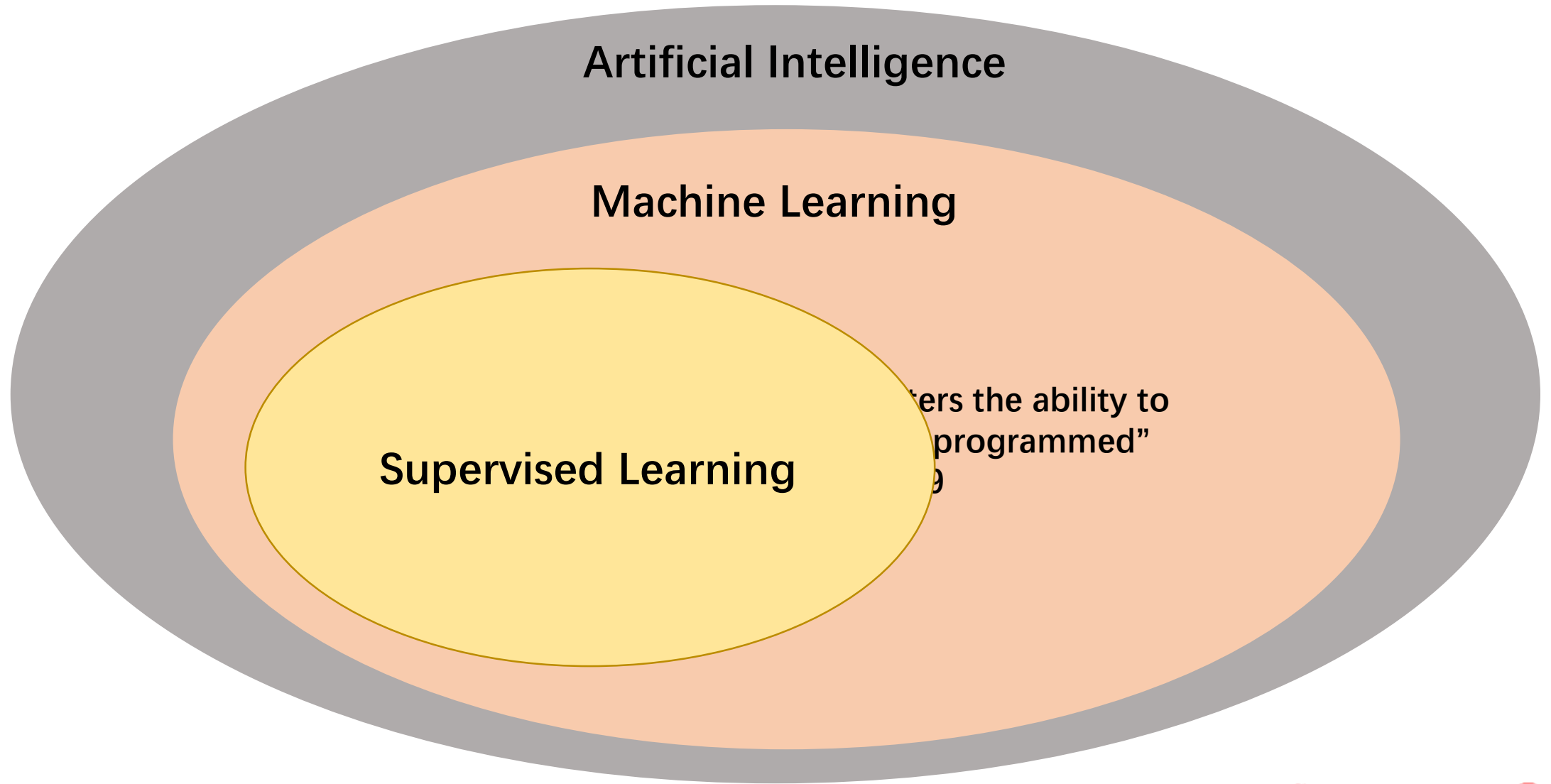
www.gameofquotes.blogspot.com



"Chaos isn't a pit. Chaos is a ladder.
Many who try to climb it fail and never get to try again.
The fall breaks them. And some are given a chance to climb,
but they refuse. They cling to the realm or the gods or love.
Illusions. Only the ladder is real. The climb is all there is."

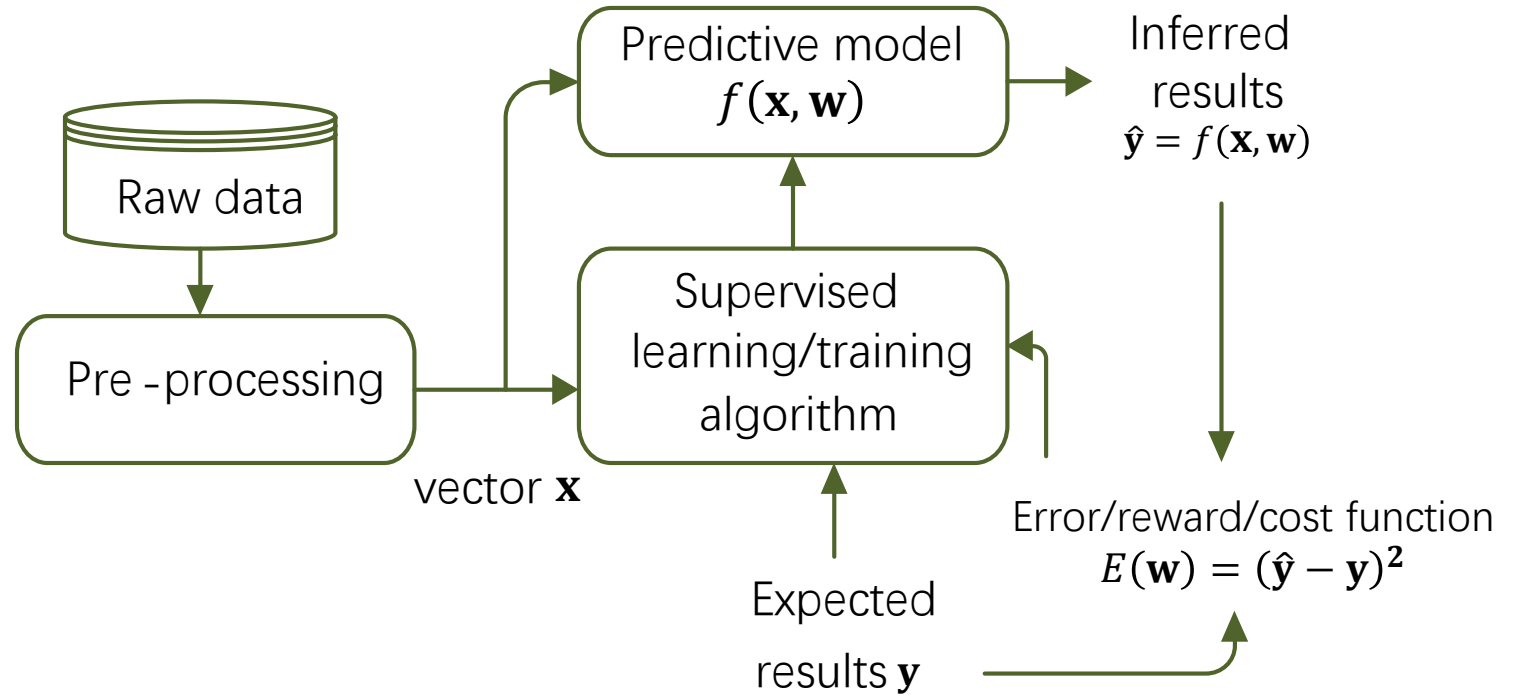
www.gameofquotes.blogspot.com

Basics on Neural Networks

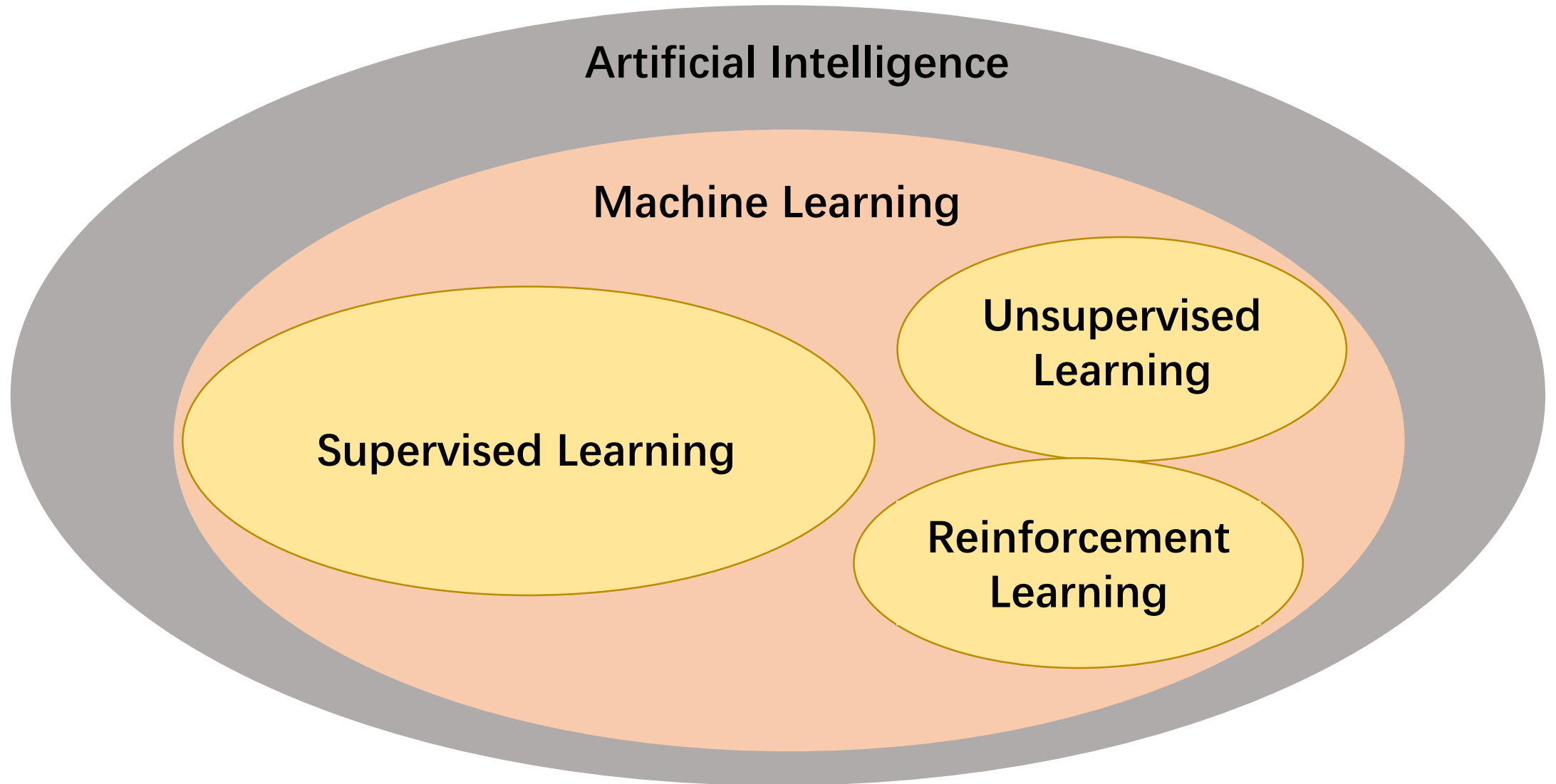


Supervised Learning

- Can be deep or not
- Can be combined with unsupervised learning (semi-supervised learning)
- Classification, regression



Basics on Neural Networks



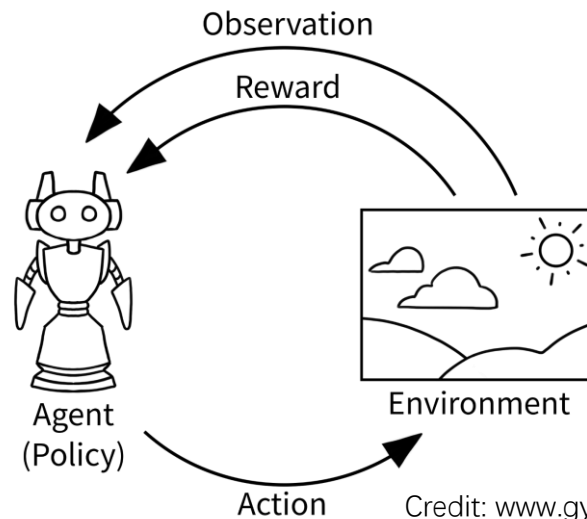
Unsupervised & Reinforcement Learning

Unsupervised learning

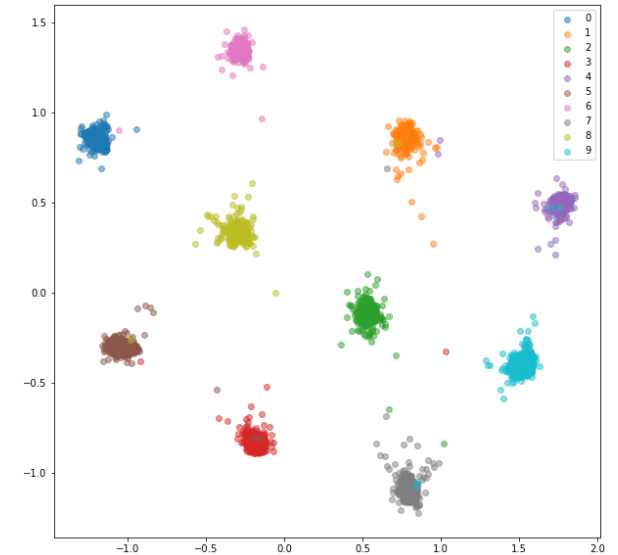
- Learning without labels
- For example, clustering, word embedding (e.g., word2vec, GloVe)

Reinforcement learning

- Interact with an environment
- No fixed dataset
- AlphaGo, AI players

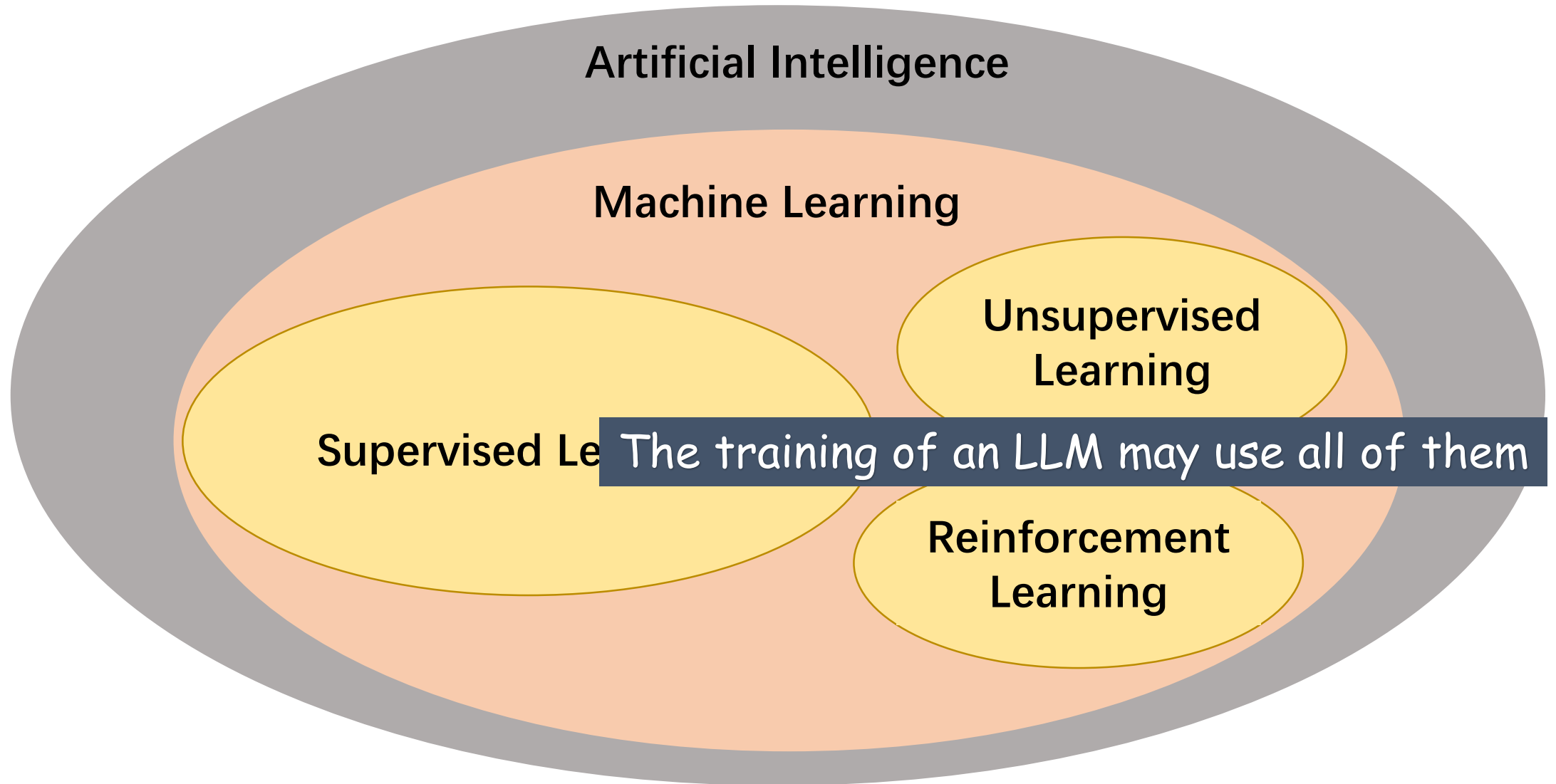


Credit: www.gymnasium.dev

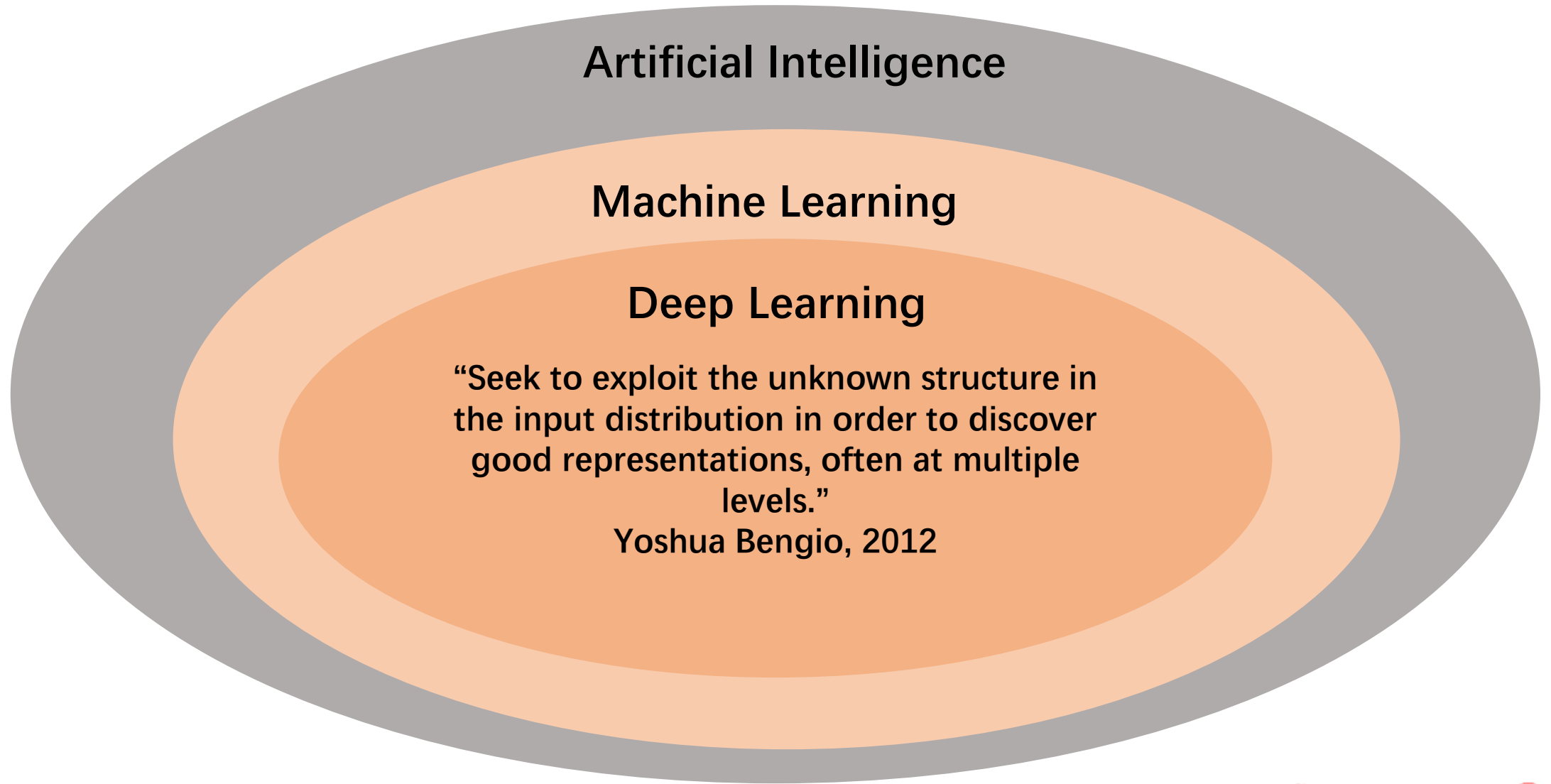


Courtesy of Prof. Honglan Jiang, SJTU

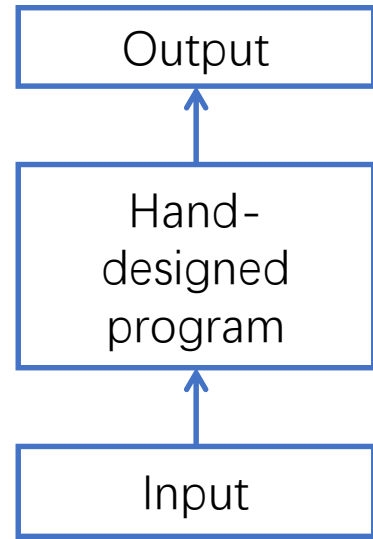
Basics on Neural Networks



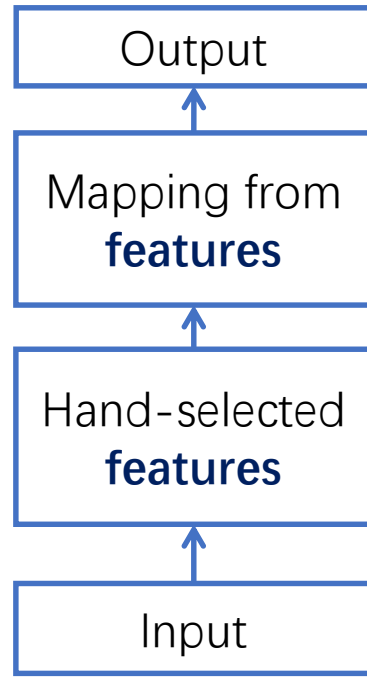
Neural Network Basics



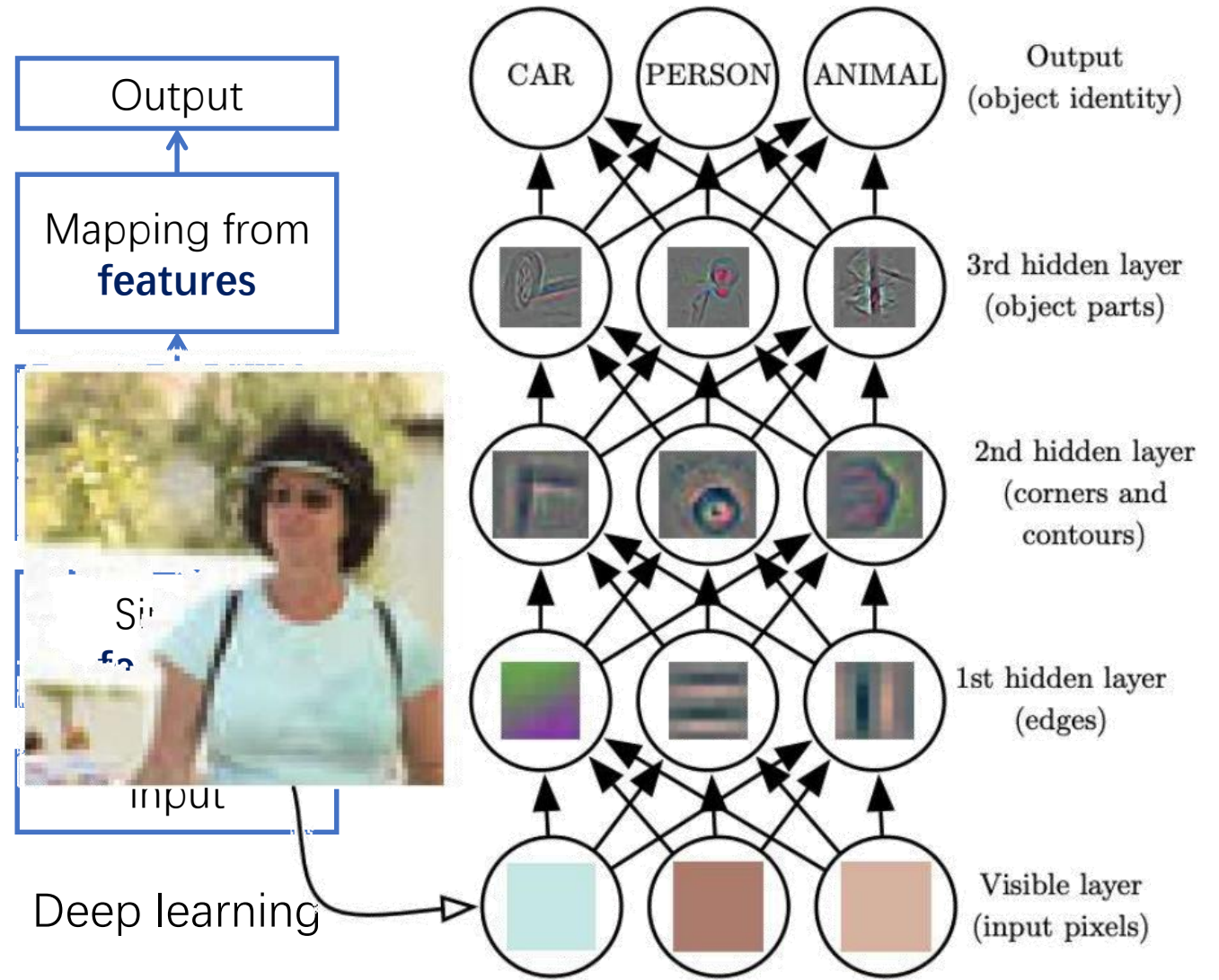
Evolution of AI



Rule-based system



Classic machine learning



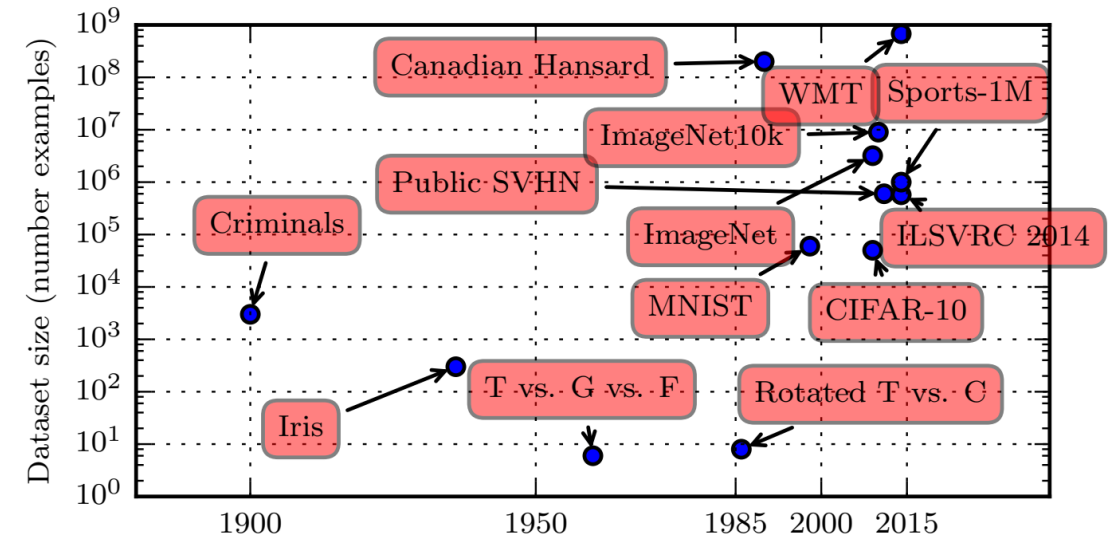
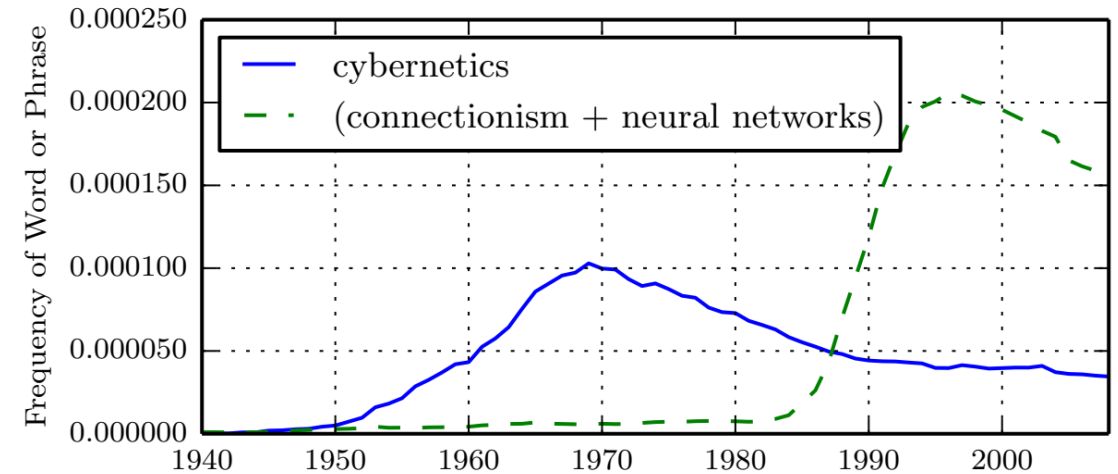
Historical Trends

Three waves of NNs

- Cybernetics
- Connectionism
- Deep learning (current wave)

Growing datasets

- Dataset size
- Different types of data (image, text, speech, video, 3D shape, etc.)



Historical Trends (Cont'd)

ImageNet

- Over 14,000,000 images
- 1,000 classes initially
- ImageNet10k

CIFAR10

- 60,000 32x32 color images
- 50,000 for training
- 10,000 for testing
- 10 classes
- 6,000 images/class

airplane



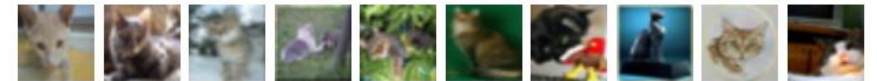
automobile



bird



cat



deer



dog



frog



horse



ship



truck



<http://www.cs.utoronto.ca/~kriz/cifar.html>

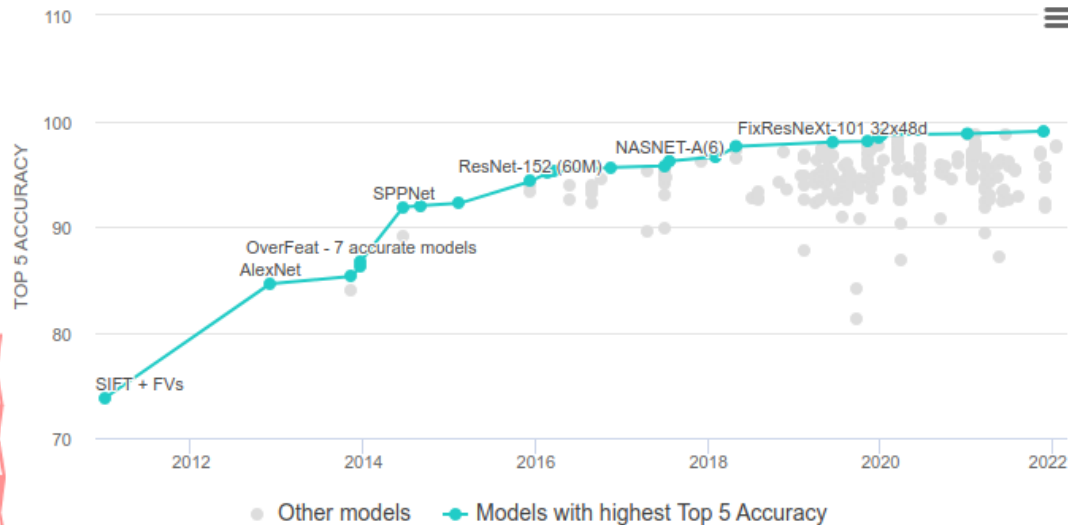
Historical Trends (Cont'd)

- Growing model size
- Better accuracy

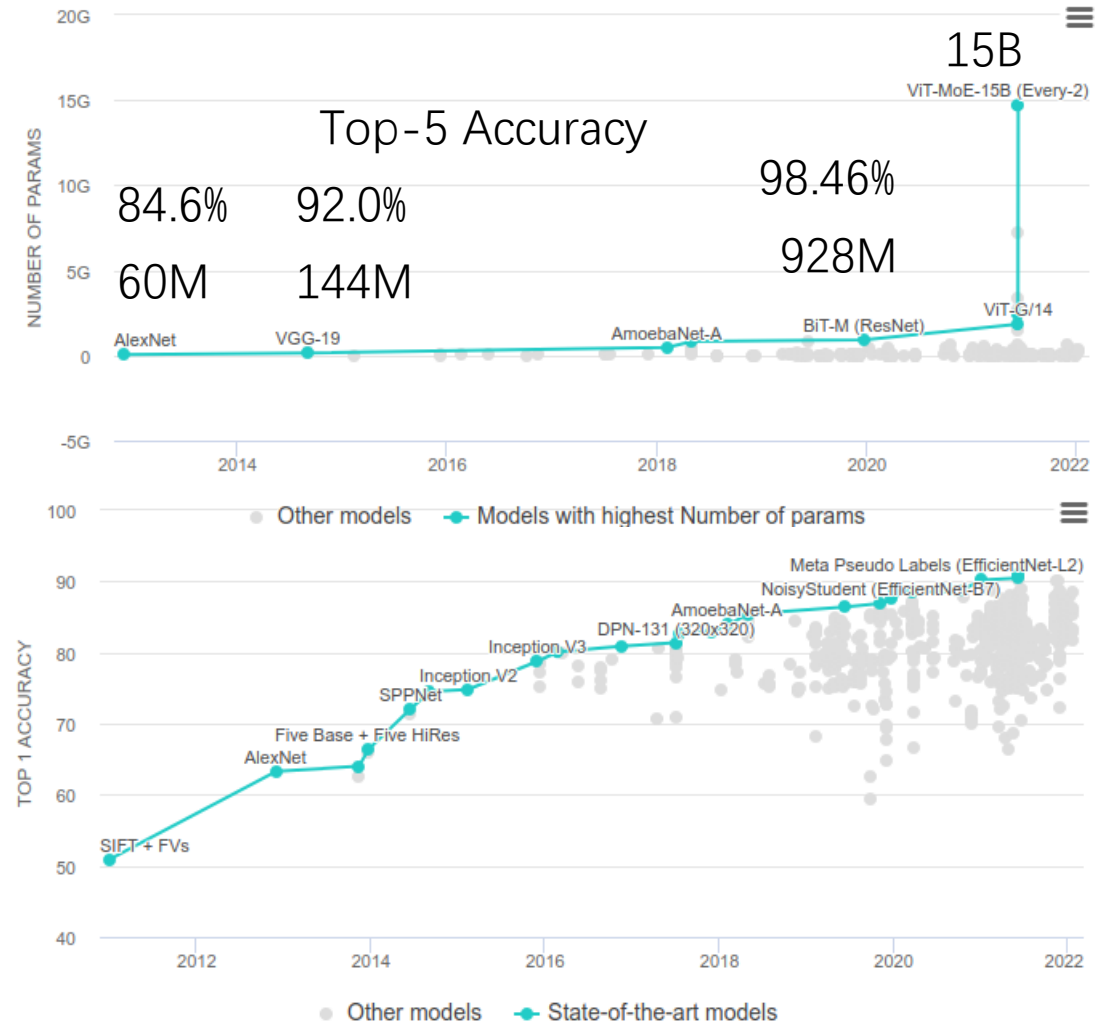
ILSVRC

ImageNet Large Scale Visual Recognition Challenge

Largely due to the availability of faster hardware.



Pics from <https://paperswithcode.com/sota/image-classification-on-imagenet>



立志成才 报国裕民

Keys of a Machine Learning Algorithm

“A computer program is said to learn from **experience (E)** with respect to some **task (T)** and some **performance measure (P)**, if its performance on T, as measured by P, improves with experience E”, Tom Mitchell, 1998

Example: spam classification

T: Predict if an email is spam or not

E: Observe users label emails as spam or not

P: Probability of emails that are correctly predicted



Dataset/Experience (E)

- **A dataset is a collection of many examples/samples.**

Generalization:

- Algorithm performs well on new, previously unseen inputs

Partition the dataset into:

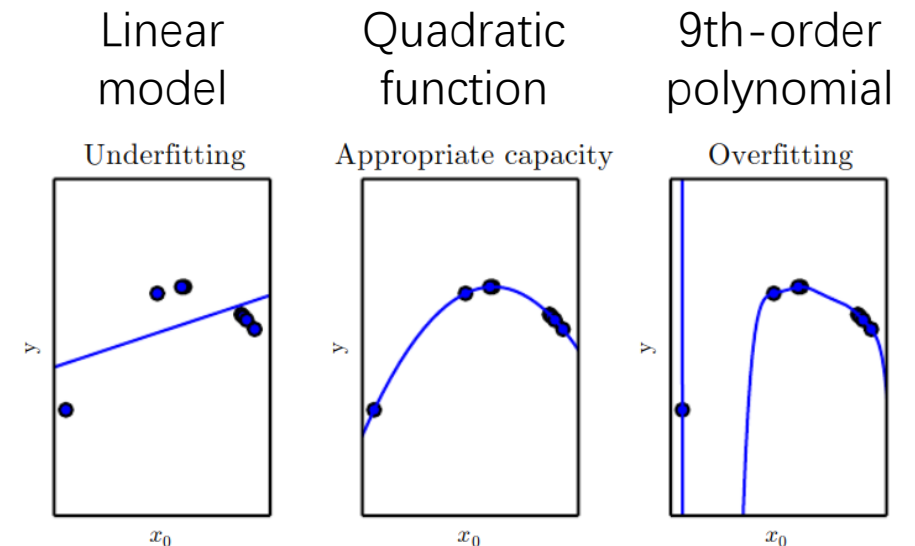
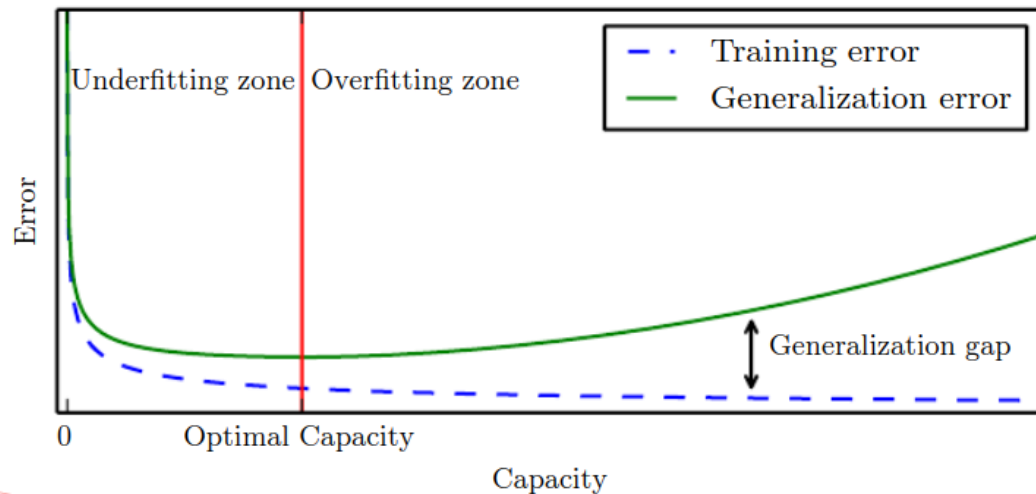
- Training set
- Validation set (cross-validation)
- Test set



Model (T) & Capacity

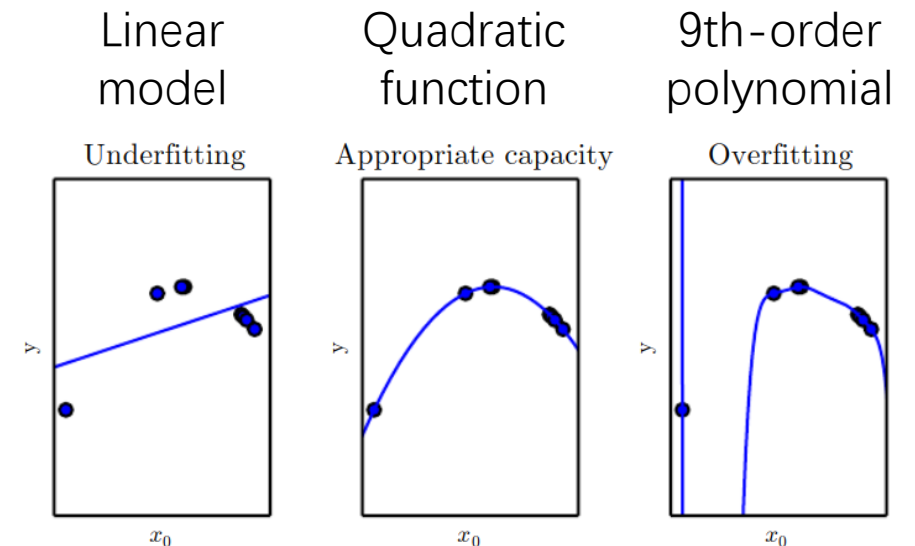
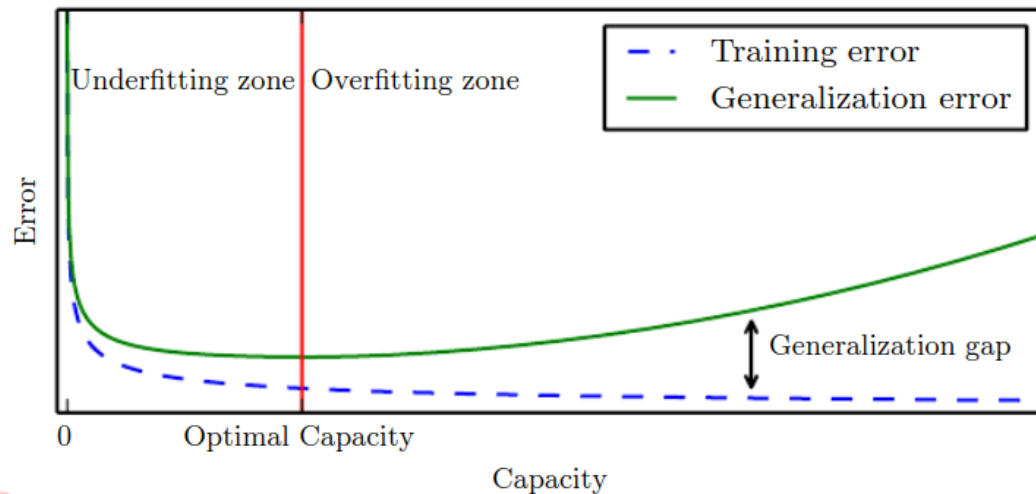
Capacity: model's ability to fit a wide variety of functions

- Overfitting vs. Underfitting:
- Hypothesis space
- Occam's razor principle
- Simpler models are more likely to generalize
- Also apply to hardware design



Cost/Loss Function (Optimize P)

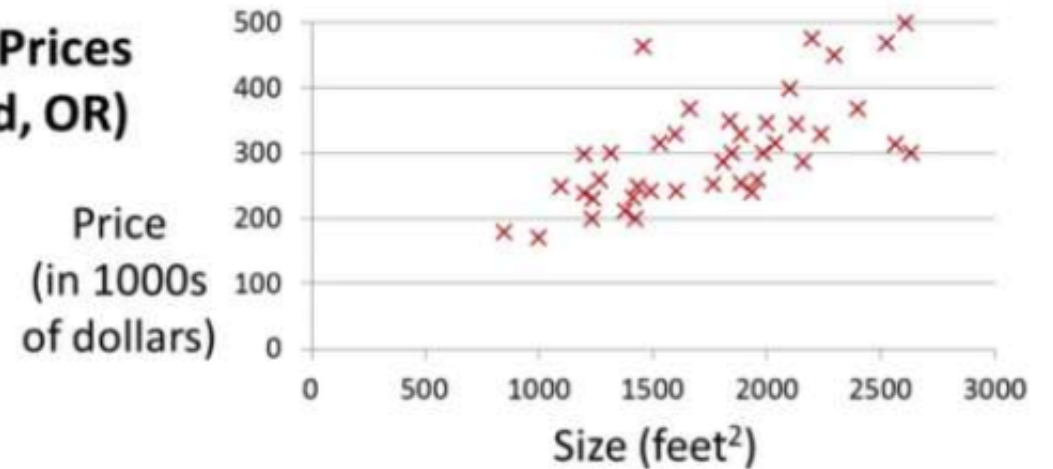
- Measure the difference between target/label and output for given data
- Quantify the error between predicted and expected values in the training set
 - Cost function = Mean squared error (MSE) = $\sum_{samples} (pred - exp)^2$
- Regularization: weight decay
 - Cost function = $MSE + \lambda \sum_i w_i^2$ (L2 regularization, $\lambda > 0$)



Example: Linear Regression

- Dataset (E):
 - (x, y) x: size and y: price
 - m training samples
- Model (T):
 - $f(x) = w_0 + w_1 \times x$
- Cost function (P):
 - Mean squared error
$$MSE = \frac{1}{m} \sum_{i=0}^m (f(x_i) - y_i)^2$$
 - Optimization method:
 - Gradient descent, etc.

**Housing Prices
(Portland, OR)**



Size in feet ² (x)	Price (\$) in 1000's (y)
2104	460
1416	232
1534	315
852	178
...	...

Machine Learning, Andrew Ng

Optimization: Gradient Descent

- Gradient descent: $w[n] = w[n-1] - \overset{\text{learning rate}}{\mu} \frac{\partial L}{\partial w[n-1]}$, iteratively. w usually is initialized with some small random values. $\swarrow$ w at step n

- Example:

- $f(x) = w_0 + w_1 x$
- $L(w_1, w_0) = MSE = \frac{1}{m} \sum_{i=0}^m (f(x_i) - y_i)^2 =$
 $\frac{1}{m} \sum_{i=0}^m (w_0 + w_1 x_i - y_i)^2$ m : # of sample

- Compute the gradient:

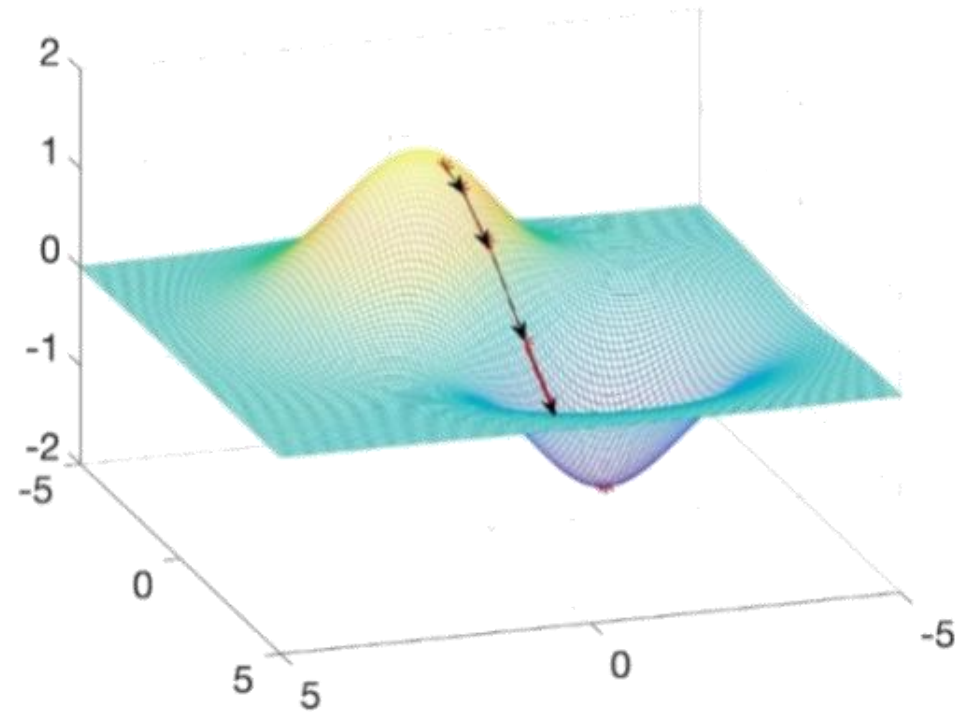
$$\frac{\partial L}{\partial w_0} = \frac{1}{m} \sum_{i=0}^m 2(w_0 + w_1 x_i - y_i)$$

$$\frac{\partial L}{\partial w_1} = \frac{1}{m} \sum_{i=0}^m 2(w_0 + w_1 x_i - y_i) x_i$$

- Iterative update:

$$w_0[n] = w_0[n-1] - \frac{\mu}{m} \sum_{i=0}^m 2(w_0[n-1] + w_1[n-1] x_i - y_i)$$

$$w_1[n] = w_1[n-1] - \frac{\mu}{m} \sum_{i=0}^m 2(w_0[n-1] + w_1[n-1] x_i - y_i) x_i$$



What if m is very big?

Stochastic Gradient Descent

- Stochastic gradient descent (SGD) (mini-batch learning)

$$w_0[n] = w_0[n-1] - \mu \frac{N}{m} \sum_{i=0}^{m/N} 2(w_0[n-1] + w_1[n-1]x_i - y_i)$$

$$w_1[n] = w_1[n-1] - \mu \frac{N}{m} \sum_{i=0}^{m/N} 2(w_0[n-1] + w_1[n-1]x_i - y_i)x_i$$

Batch size

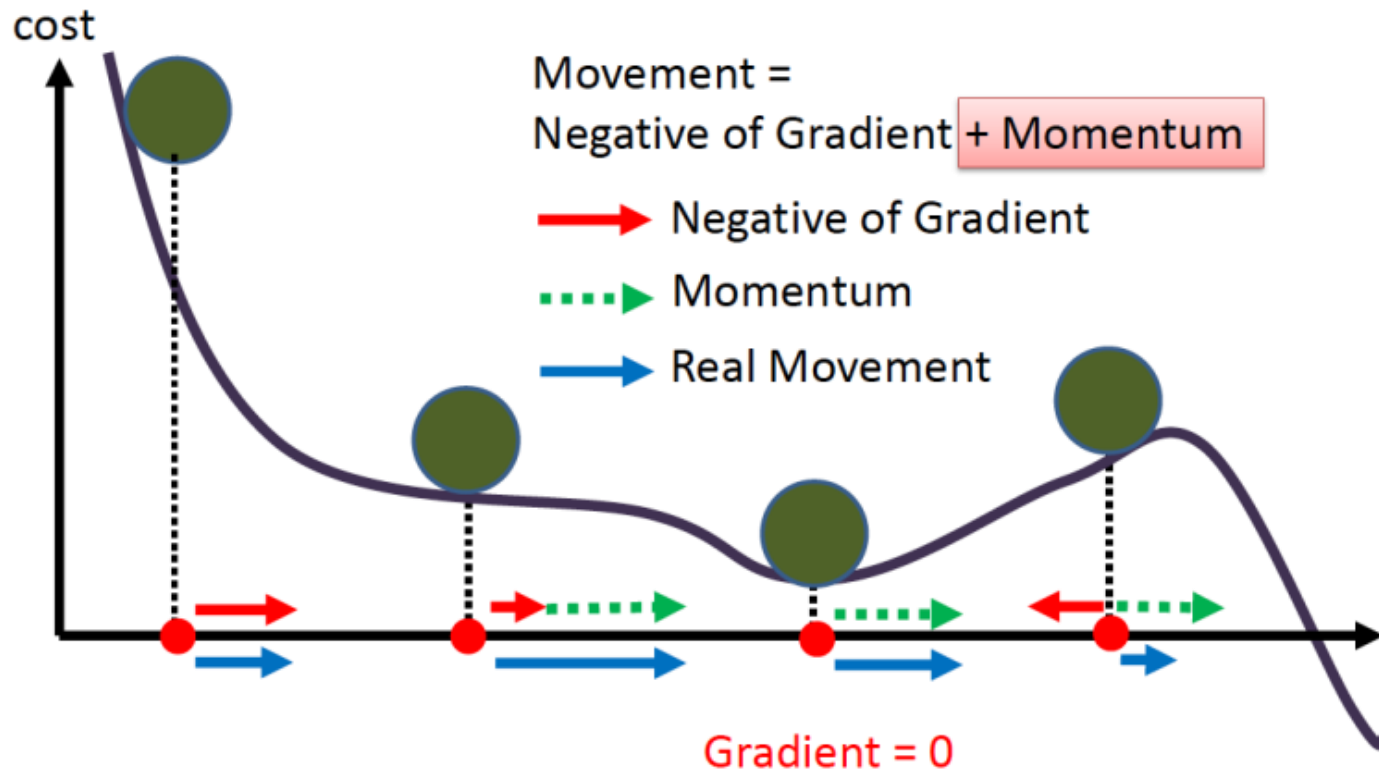
Each batch is randomly sampled without replacement, until an epoch is complete.

- On-line learning: when $m = N$.



Stochastic Gradient Descent with Momentum

- Momentum: keep the direction of previous update.



SGD

$$w[n] = w[n-1] - \mu \frac{\partial L}{\partial w[n-1]}$$

while True:

```
dw = compute_gradient(L)
w -= learning_rate * dw
```

SGD+Momentum

$$v[n] = \beta v[n-1] + \frac{\partial L}{\partial w[n-1]}$$

$$w[n] = w[n-1] - \mu v[n]$$

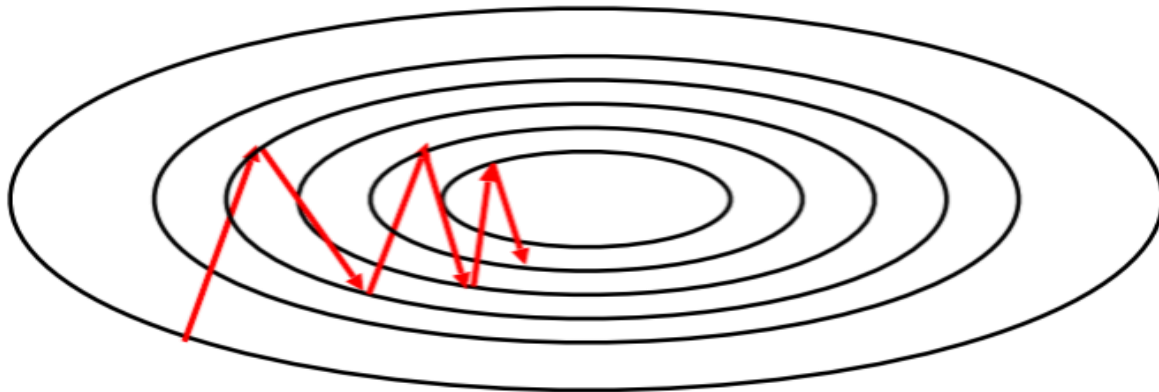
v = 0

while True:

```
dw = compute_gradient(x)
v = beta * x + dw
w -= learning_rate * v
```

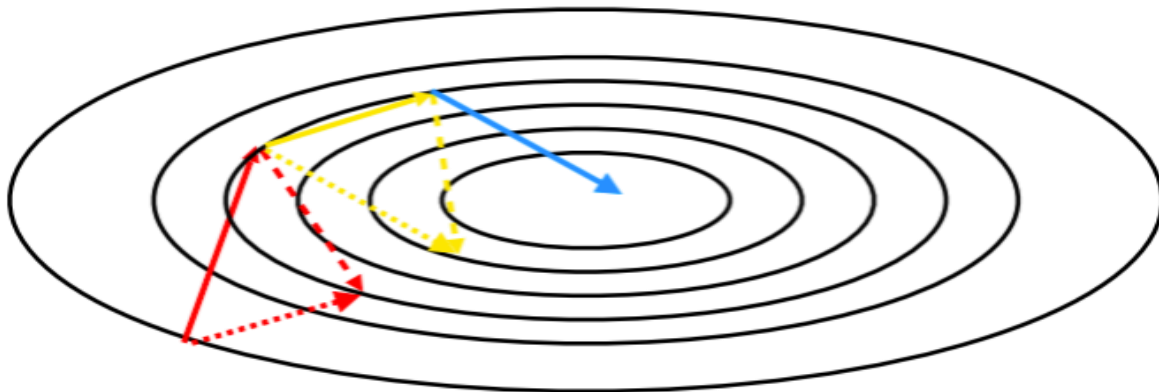
Stochastic Gradient Descent with Momentum

- Momentum: keep the direction of previous update.



Naïve GD

What are the drawbacks?



SGD

SGD

$$w[n] = w[n-1] - \mu \frac{\partial L}{\partial w[n-1]}$$

while True:

```
dw = compute_gradient(L)
w -= learning_rate * dw
```

SGD+Momentum

$$v[n] = \beta v[n-1] + \frac{\partial L}{\partial w[n-1]}$$

$$w[n] = w[n-1] - \mu v[n]$$

v = 0

while True:

```
dw = compute_gradient(x)
v = beta * x + dw
w -= learning_rate * v
```

Adam Optimizer

- 1. First and second momentum. Let $\frac{\partial L}{\partial w[n-1]} = g_{n-1}$

$$v[n] = \beta_1 v[n-1] + (1 - \beta_1) g_{n-1}$$

$$u[n] = \beta_2 u[n-1] + (1 - \beta_2) g_{n-1}^2$$

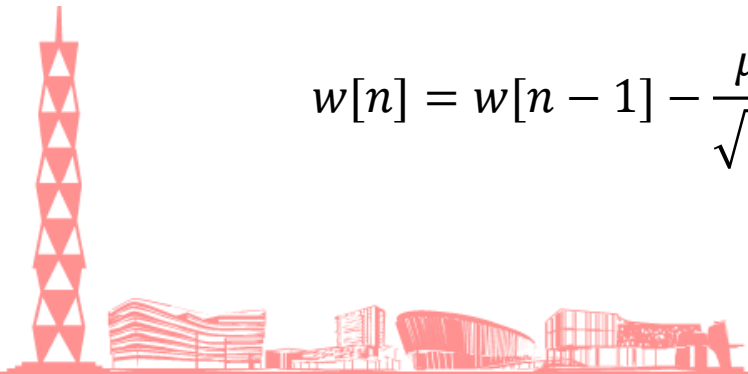
- 2. First and second momentum estimate correction

What are the drawbacks?

$$\hat{v}[n] = \frac{v[n]}{1 - \beta_1^n} \quad \hat{u}[n] = \frac{u[n]}{1 - \beta_2^n}$$

- 3. Weight update

$$w[n] = w[n-1] - \frac{\mu \hat{v}[n]}{\sqrt{\hat{v}_t} + \epsilon}$$



Adam Optimizer

- 1. First and second momentum. Let $\frac{\partial L}{\partial w[n-1]} = g_{n-1}$

$$v[n] = \beta_1 v[n-1] + (1 - \beta_1) g_{n-1}$$

$$u[n] = \beta_2 u[n-1] + (1 - \beta_2) g_{n-1}^2$$

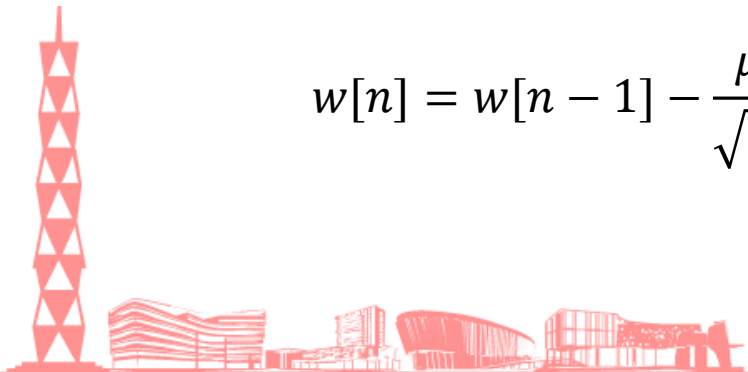
- 2. First and second momentum estimate correction

What are the drawbacks?

$$\hat{v}[n] = \frac{v[n]}{1 - \beta_1^n} \quad \hat{u}[n] = \frac{u[n]}{1 - \beta_2^n}$$

- 3. Weight update

$$w[n] = w[n-1] - \frac{\mu \hat{v}[n]}{\sqrt{\hat{v}_t} + \epsilon}$$



The Other Optimizers

Adadelta

Adagrad

Adam

AdamW

SparseAdam

Adamax

ASGD

NAdam

RAdam

RMSprop

Rprop

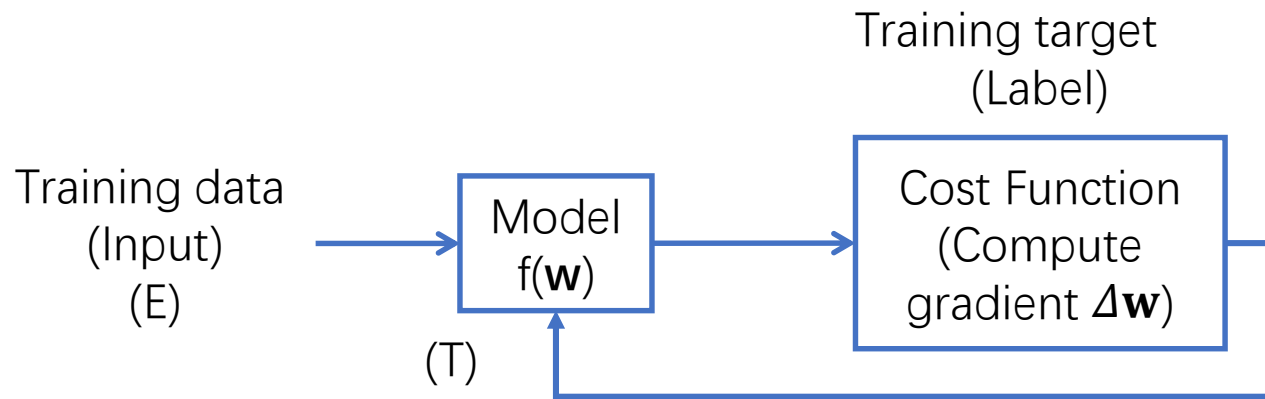
SGD

See <https://pytorch.org/docs/stable/optim.html> for more.



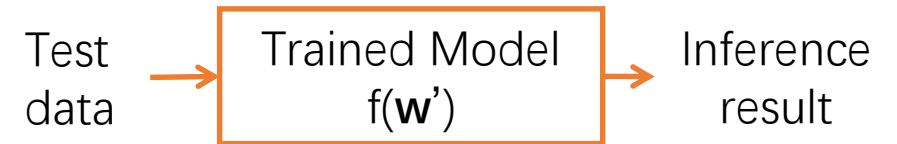
Wrap-it-up

- Model training: Data, model, cost function & optimizer
- Inference: compute with the optimized weights & new input data



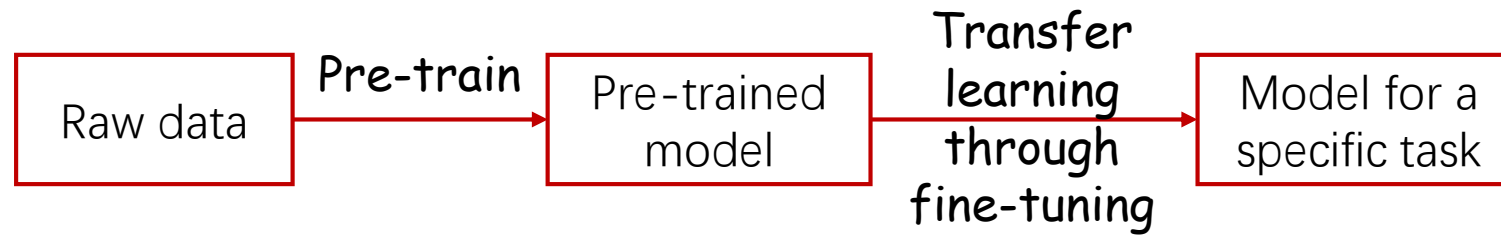
$$\mathbf{w} = \mathbf{w} - \mu \Delta \mathbf{w}$$

(Train until converge, or
certain condition is met)



The Other Training Paradigm

- Transfer learning
 - Pre-train and fine-tuning



Usually use GD-based (Adam/AdamW) optimizer with smaller step size and/or fixed some of the weights.
E.g. image recognition; chatGPT

