Computer Vision

第十二周 网络构建

庞彦

yanpang@gzhu.edu.cn

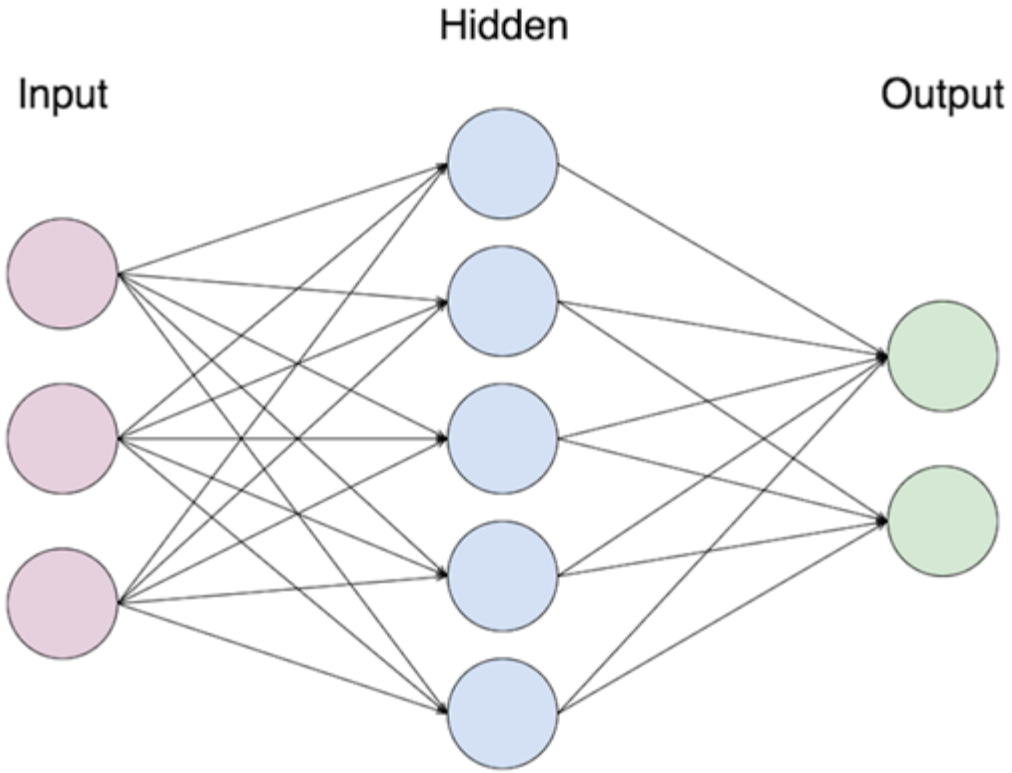
Fall 2023

七月在线
JULYEDU.COM

01 Activation Functions

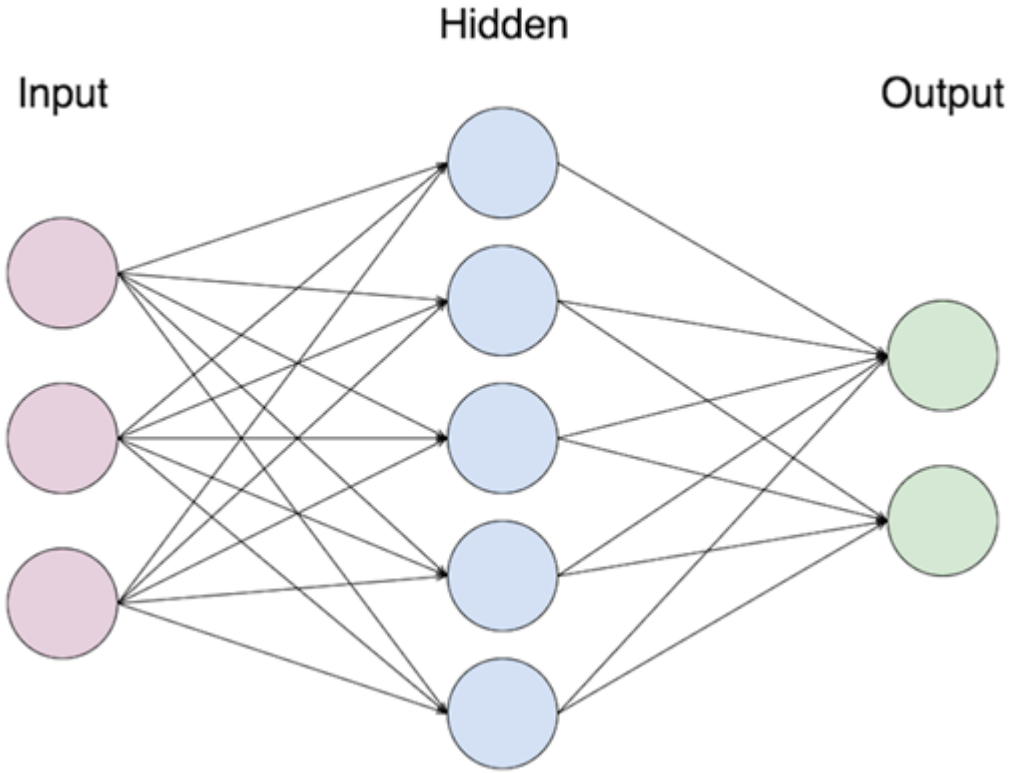
激活函数

Neural Network



有模有样

Neural Network

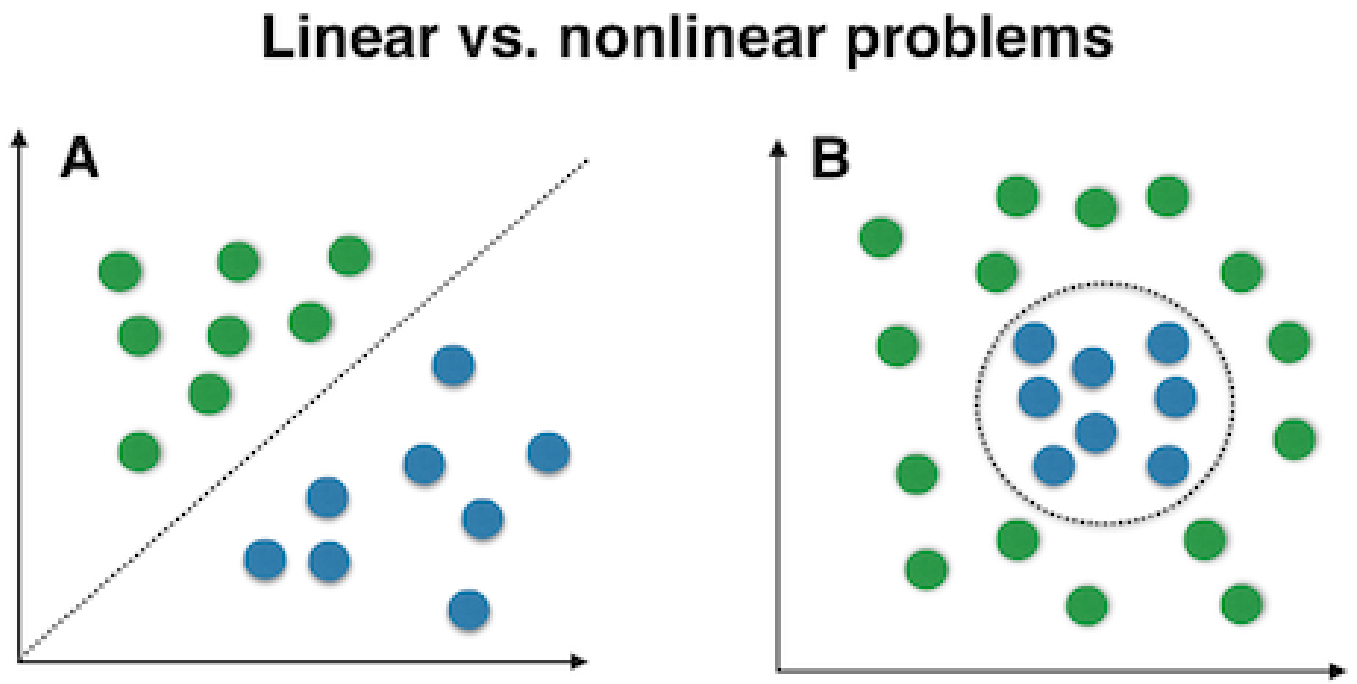


$$H = XW_h + b_h$$

$$O = HW_o + b_o$$

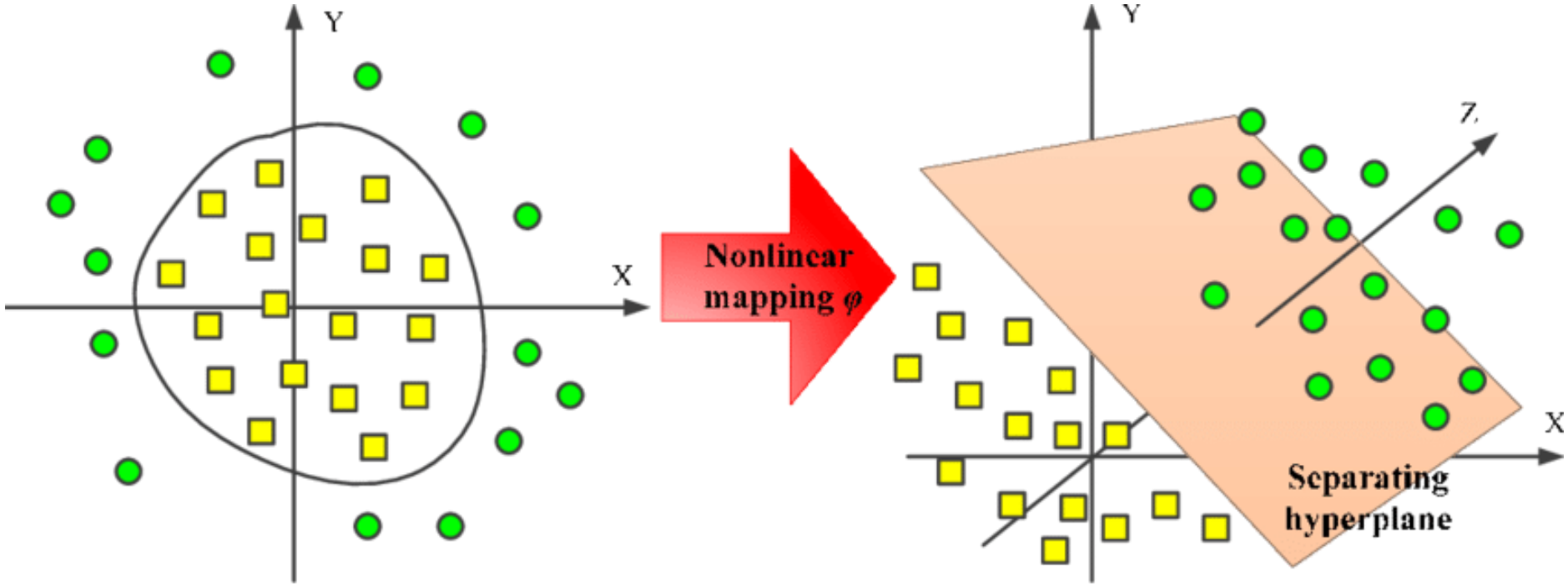
$$\begin{aligned} O &= (XW_h + b_h)W_o + b_o \\ &= XW_hW_o + b_hW_o + b_o \\ &= XW + b \end{aligned}$$

Nonlinear

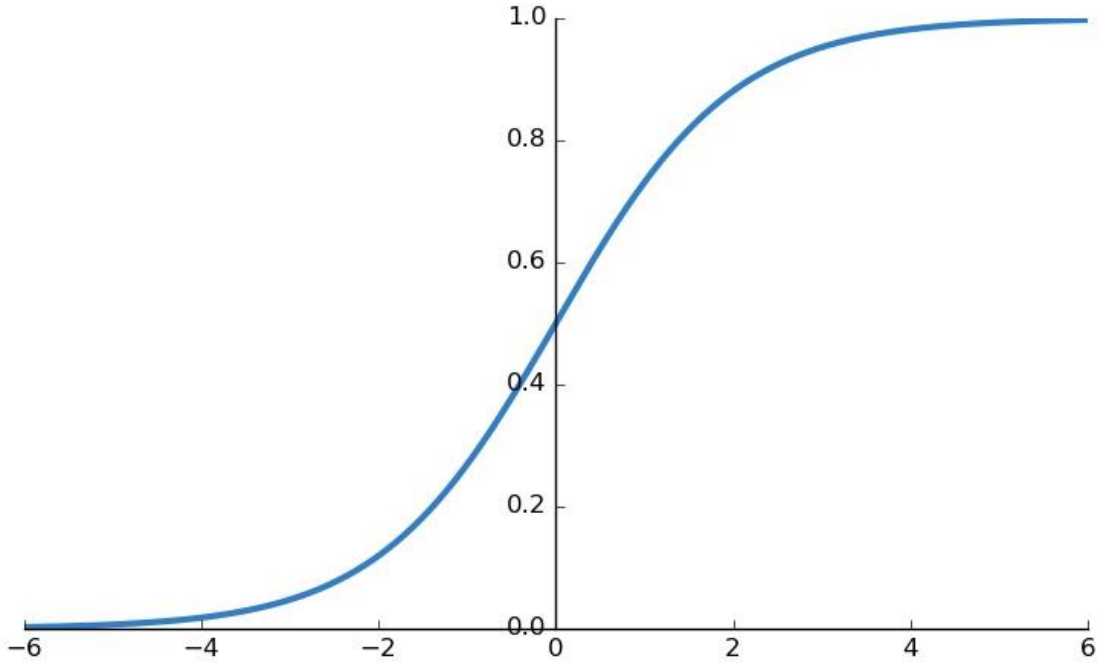


有模有样

Nonlinear



Sigmoid



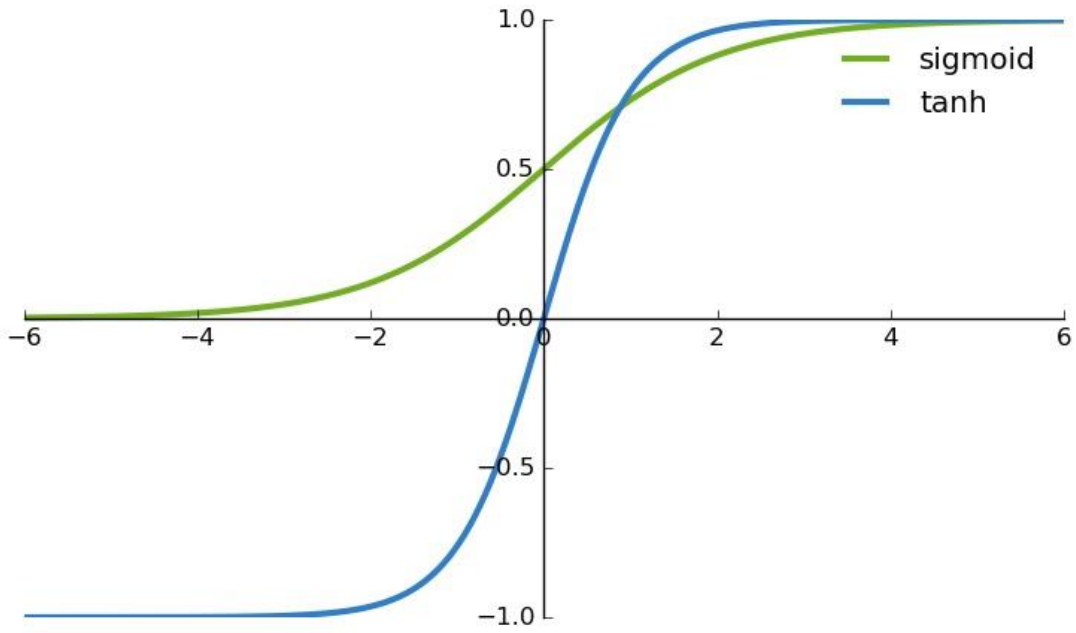
$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

$$\sigma'(z) = \frac{e^{-z}}{(1 + e^{-z})^2}$$

$$= \sigma(z)(1 - \sigma(z))$$

有模有样

tanh

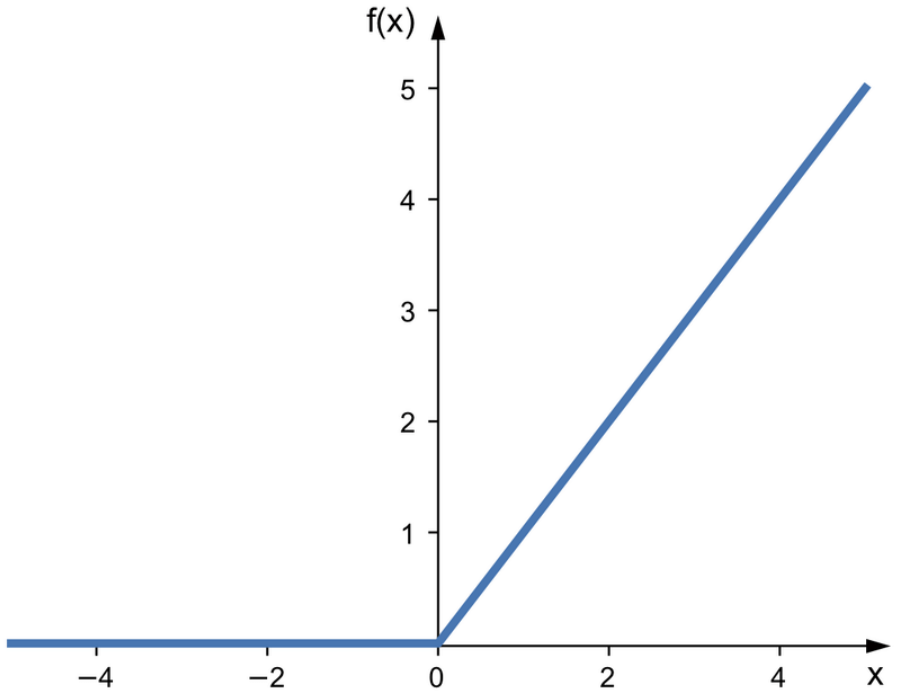


$$\sigma(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\sigma'(z) = \frac{4}{(e^z + e^{-z})^2}$$

$$= 1 - \sigma(z)^2$$

ReLU： Rectified Linear Unit

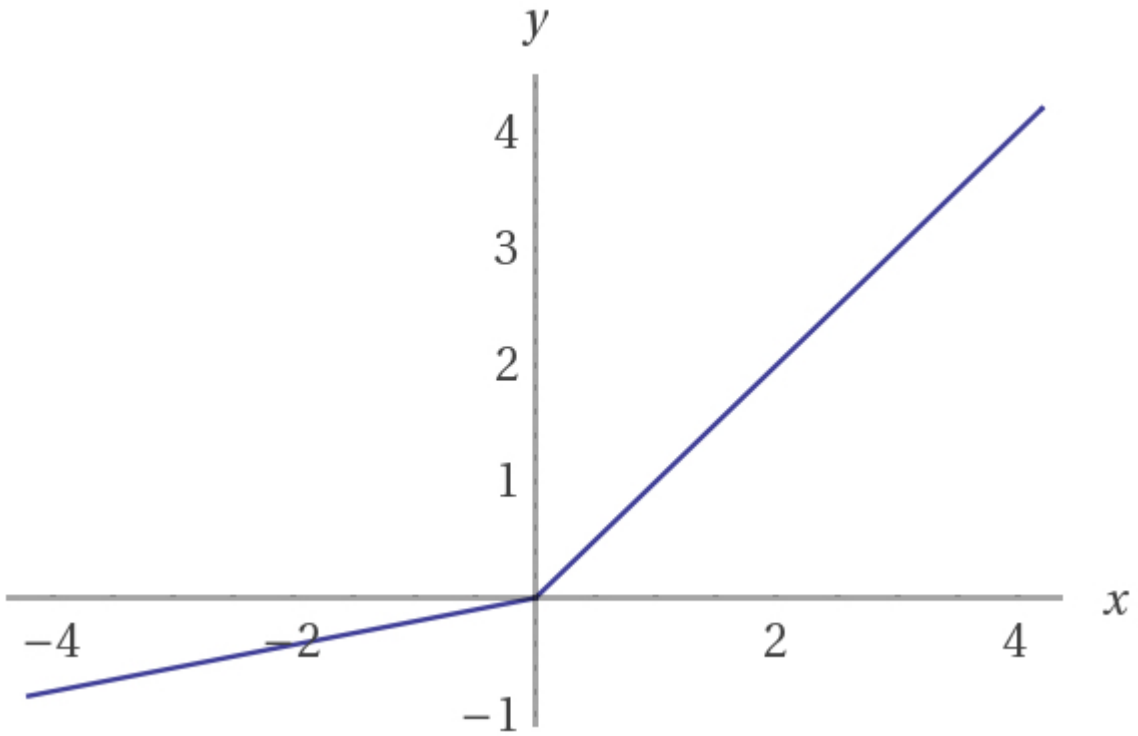


$$\sigma(z) = \begin{cases} z, & if\ z > 0 \\ 0, & if\ z < 0 \end{cases}$$

$$\sigma'(z) = \begin{cases} 1, & if\ z > 0 \\ 0, & if\ z < 0 \end{cases}$$

有模有样

Leaky ReLU

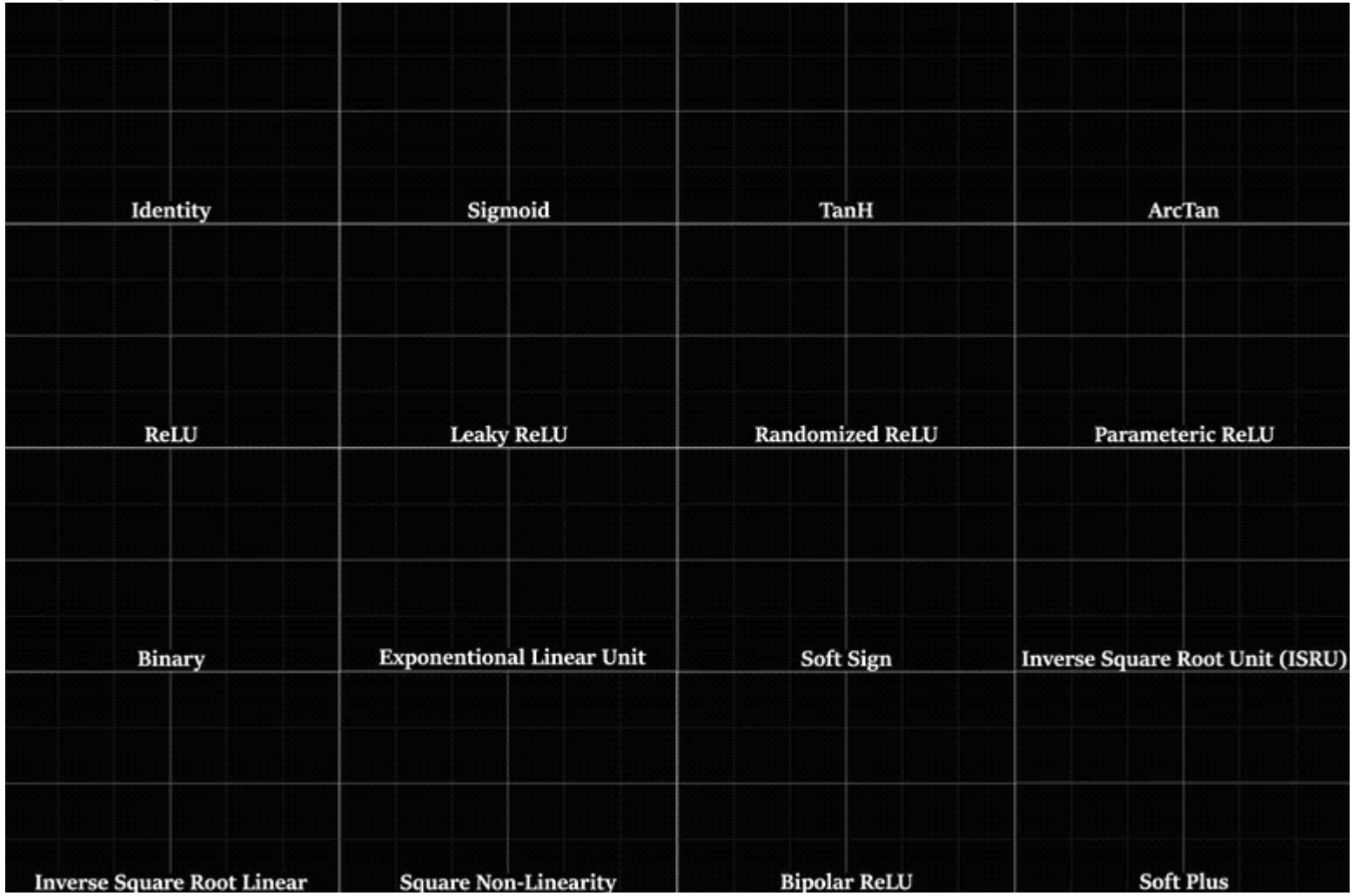


$$\sigma(z) = \begin{cases} z, & if\ z > 0 \\ az, & if\ z < 0 \end{cases}$$

$$\sigma'(z) = \begin{cases} 1, & if\ z > 0 \\ a, & if\ z < 0 \end{cases}$$

有模有样

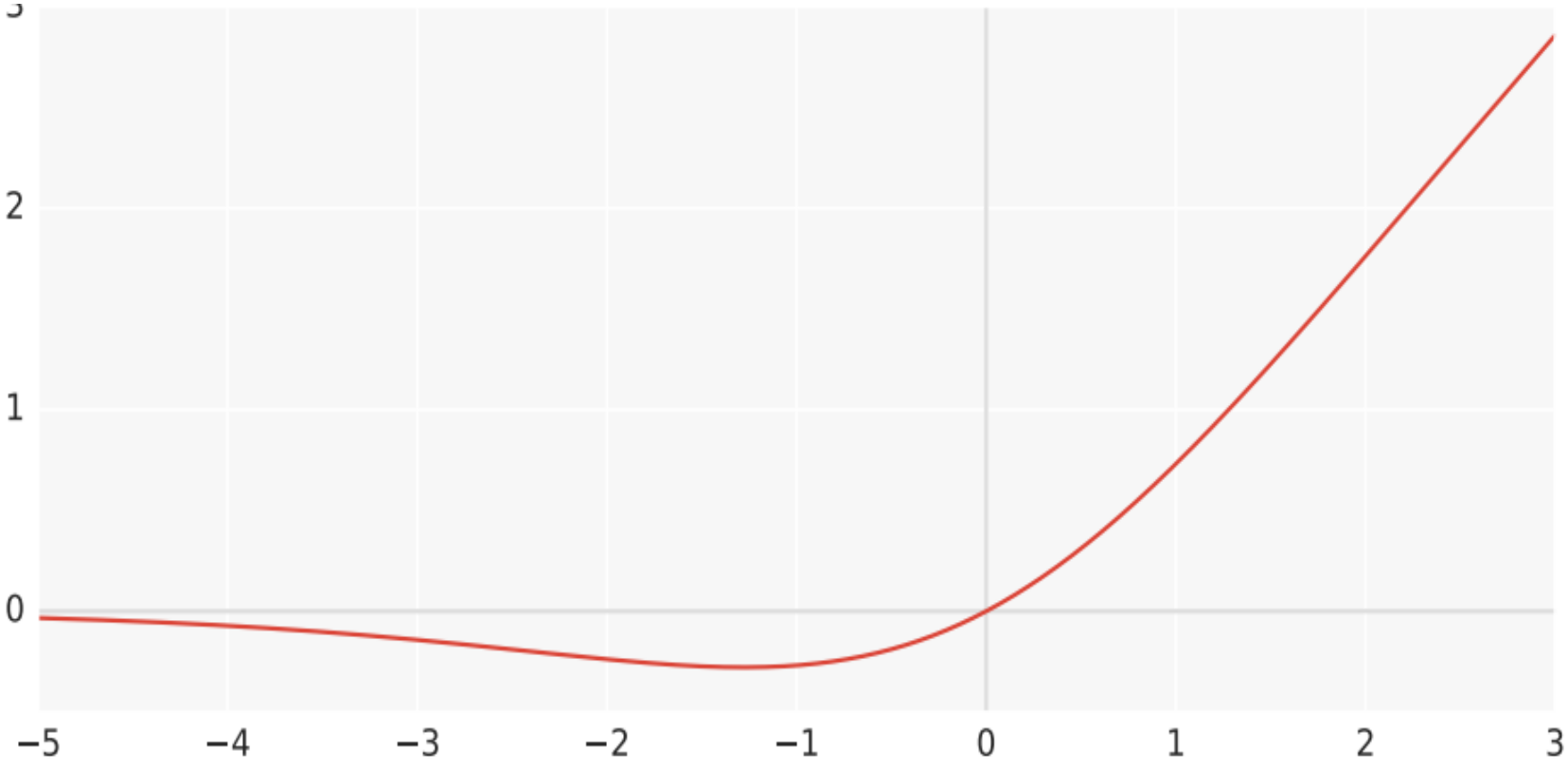
More ReLU



有模有样

https://mlfromscratch.com/activation-functions-explained/#/

Swish



$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

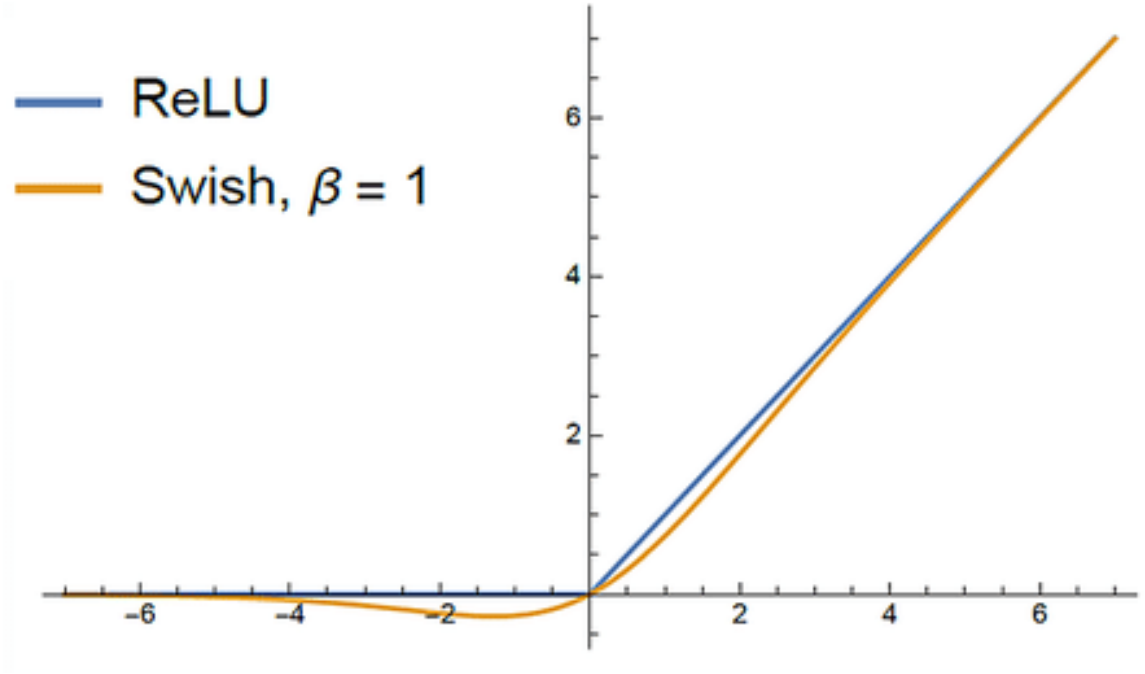
$$f(z) = z \cdot \sigma(\beta z)$$

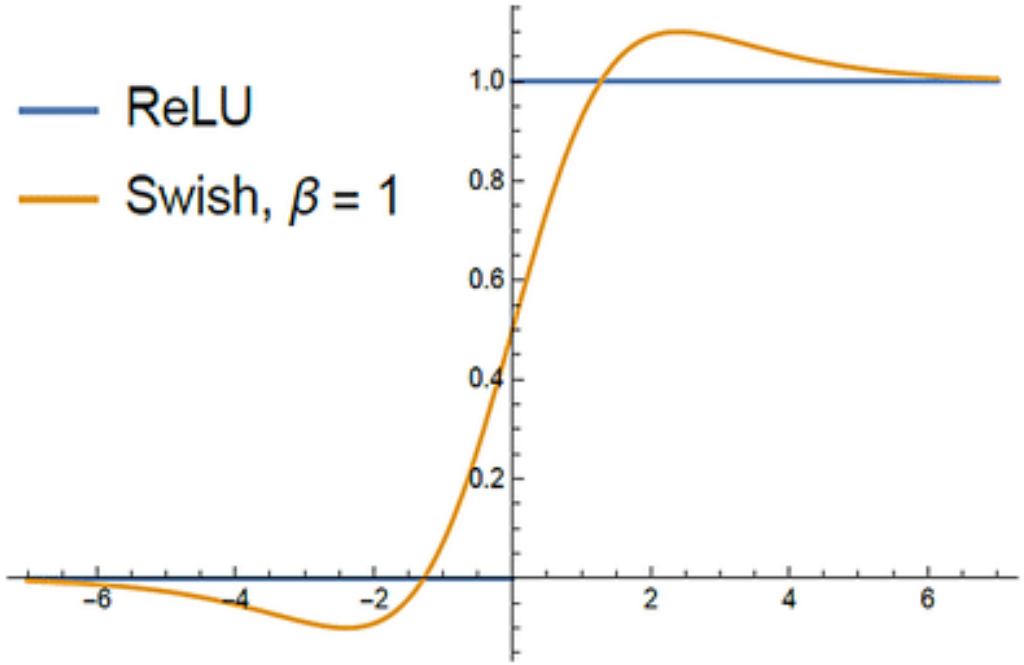
有模有样

Swish

$$f(z) = z \cdot \sigma(\beta z)$$

$$f(z)' = \beta f(z) + \sigma(\beta z)(1 - \beta f(x))$$





有模有样

Swish

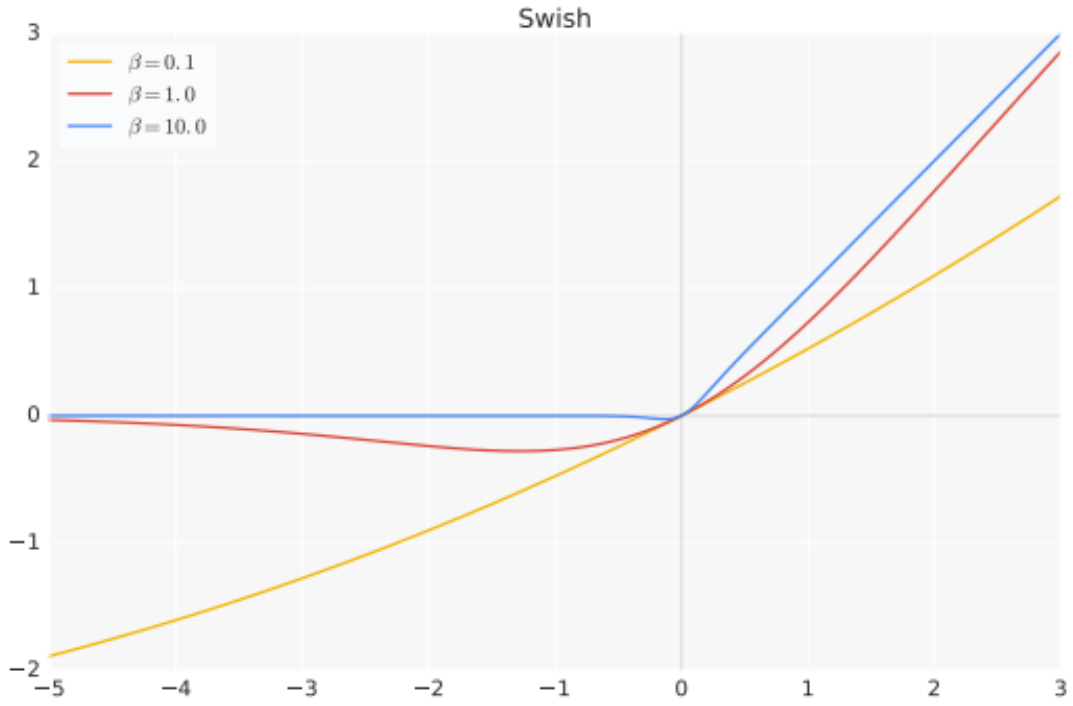


Figure 4: The Swish activation function.

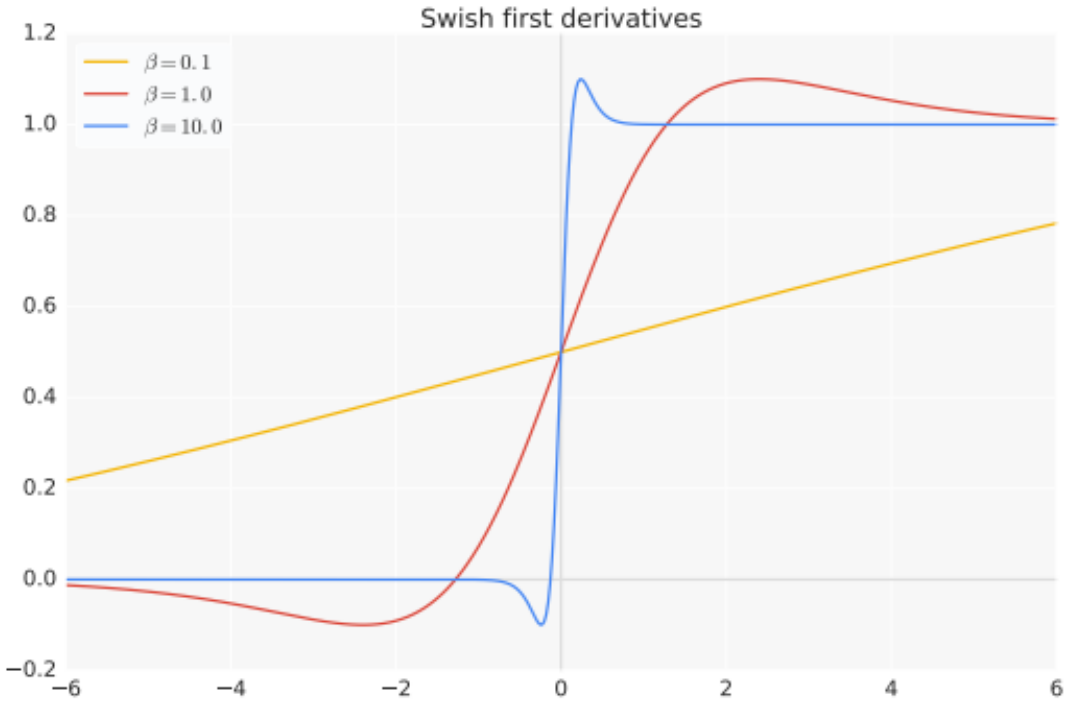
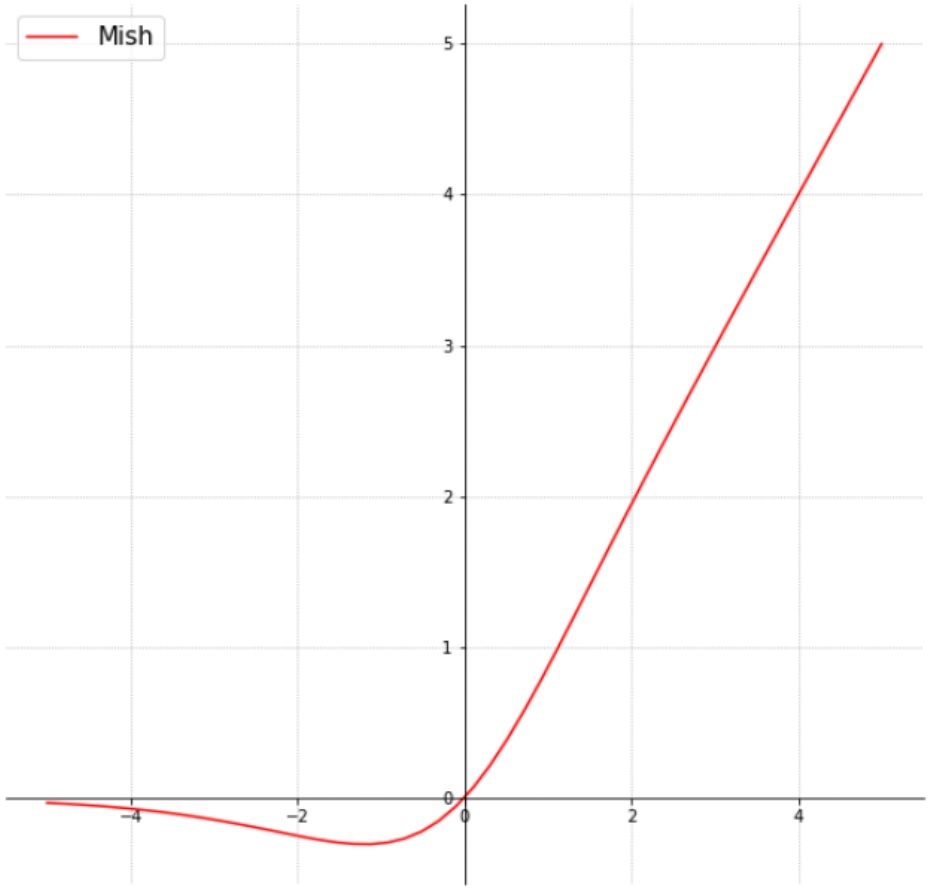


Figure 5: First derivatives of Swish.

有模有样

Mish



$$f(z) = z \cdot tanh(\ln(1 + e^z))$$

$$f(z)' = \frac{e^z \omega}{\delta^2}$$

$$\omega = 4(z + 1) + 4e^{2z} + e^{3z} + e^z(4z + 6)$$

$$\delta = 2e^z + e^{2z} + 2$$

Activation Function	Mean Accuracy	Mean Loss	Standard Deviation (Accuracy)
Mish	**87.48%**	**4.13%**	**0.3967**
Swish	87.32%	4.22%	0.414

有模有样

七月在线
JULYEDU.COM

02 Normalization

齐次化

Normalization

Why do we need Normalization ?

Normalization techniques can decrease your model's training time by a huge factor. Let me state some of the benefits of using Normalization.

- ✓ It normalizes each feature so that they maintains the contribution of every feature, as some feature has higher numerical value than others. This way our network can be unbiased(to higher value features).
- ✓ It reduces **Internal Covariate Shift**. It is the change in the distribution of network activations due to the change in network parameters during training. To improve the training, we seek to reduce the internal covariate shift.
- ✓ It makes the Optimization faster because normalization doesn't allow weights to explode all over the place and restricts them to a certain range.
- ✓ An unintended benefit of Normalization is that it helps network in Regularization(only slightly, not significantly).

Normalization

How Normalization layers behave in Distributed training ?

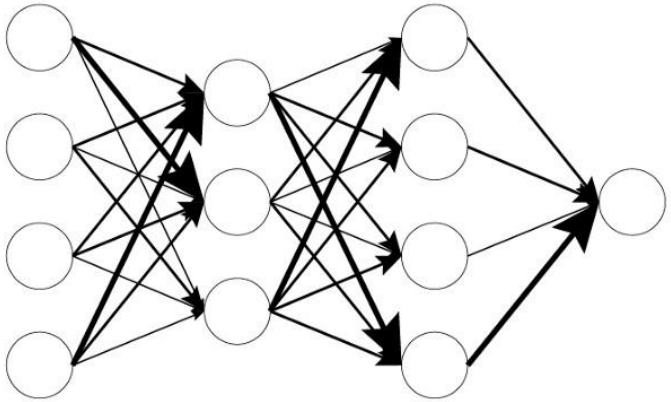
Which Normalization technique should you use for your task like CNN, RNN, style transfer etc ?

What happens when you change the batch size of dataset in your training ?

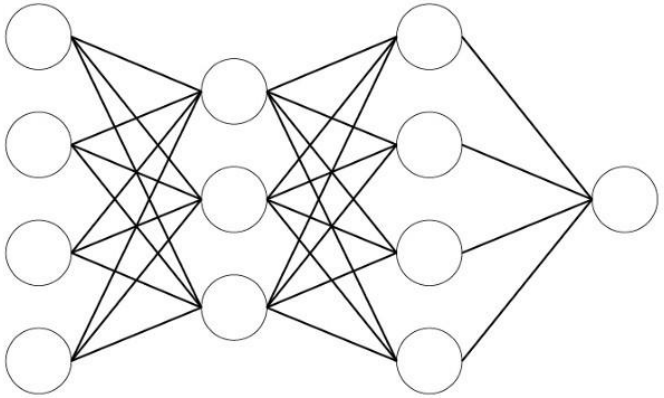
Which norm technique would be the best trade-off for computation and accuracy for your network ?

Batch Normalization

Batch normalization is a method that normalizes activations in a network across the mini-batch of definite size.



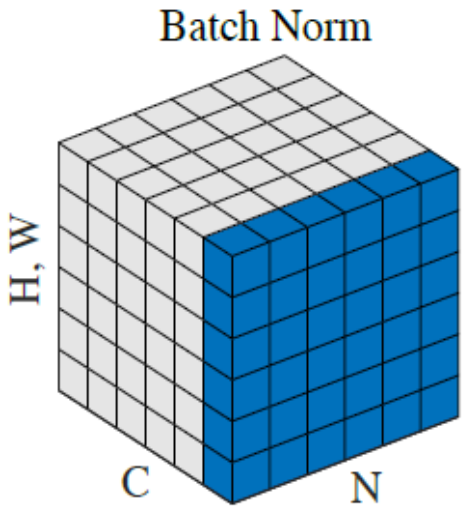
- **Raw** signal
- **High interdependancy** between distributions
- **Slow** and **unstable** training



- **Normalized** signal
- **Mitigated interdependancy** between distributions
- **Fast** and **stable** training

Batch Normalization

If we would like to represent real numbers, we have to give up perfect precision.



Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

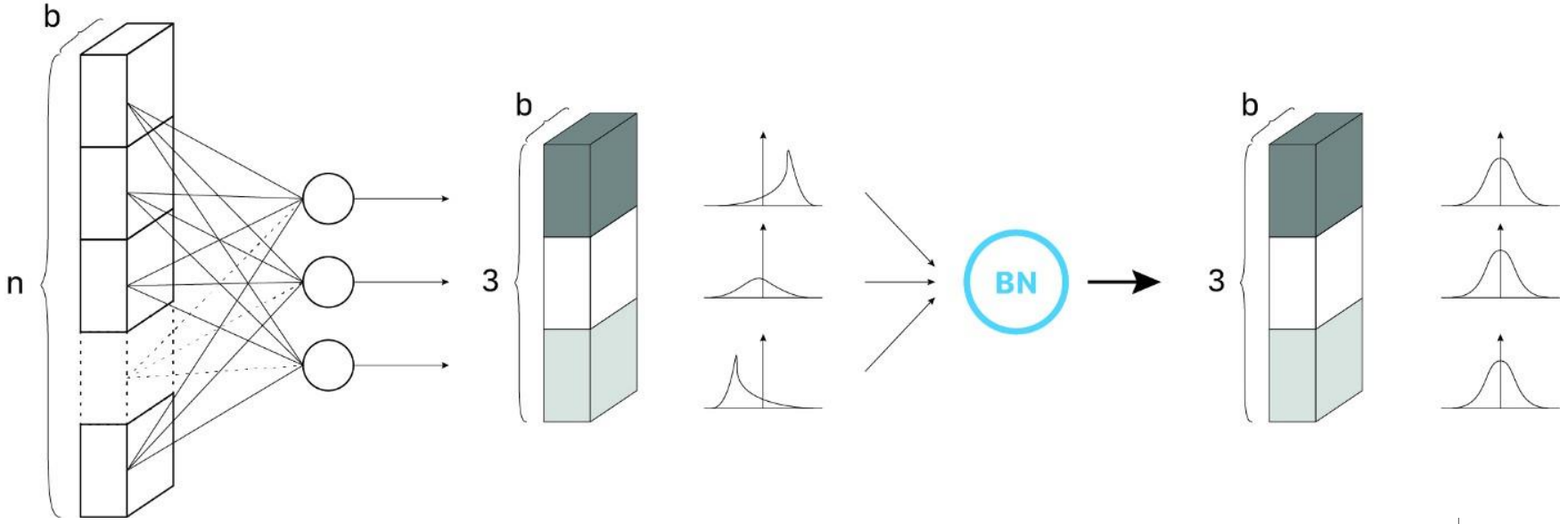
$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

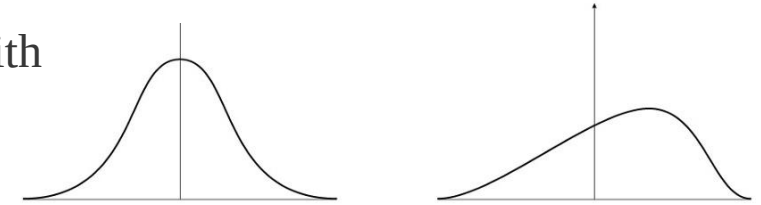
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

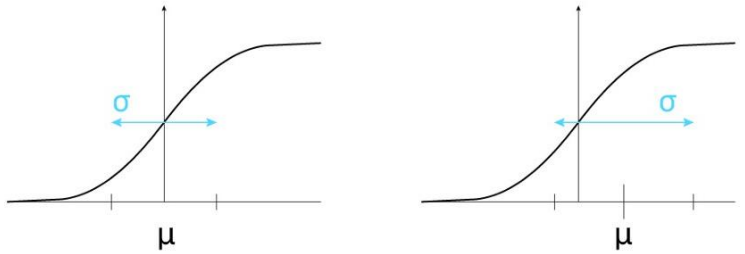
Batch Normalization



Batch Normalization first step. Example of a 3-neurons hidden layer, with a batch of size b. Each neuron follows a standard normal distribution.



- γ allows to adjust the standard deviation ;
- β allows to adjust the bias, shifting the curve on the right or on the left side.



Batch Normalization

Problems associated with Batch Normalization:

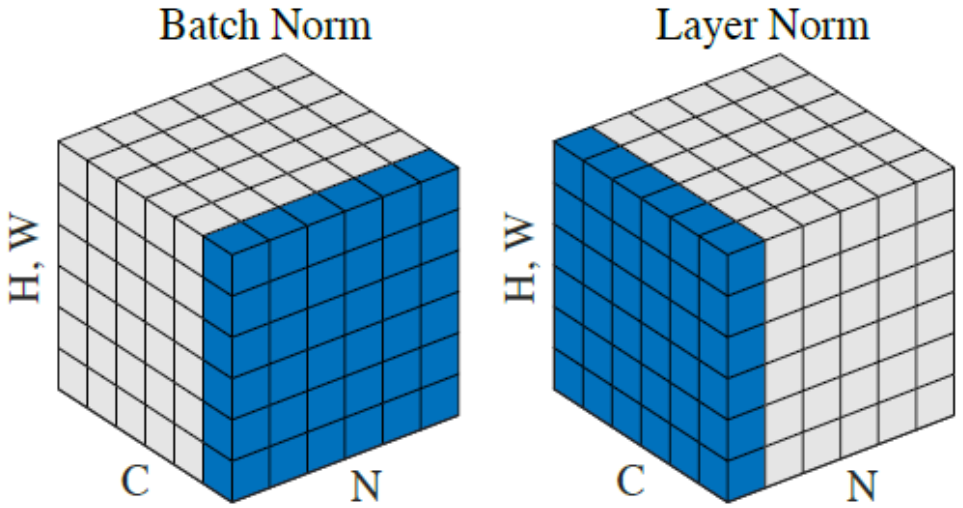
Variable Batch Size → If batch size is of 1, then variance would be 0 which doesn't allow batch norm to work. Furthermore, if we have small mini-batch size then it becomes too noisy and training might affect. There would also be a problem in distributed training. As, if you are computing in different machines then you have to take same batch size because otherwise γ and β will be different for different systems.

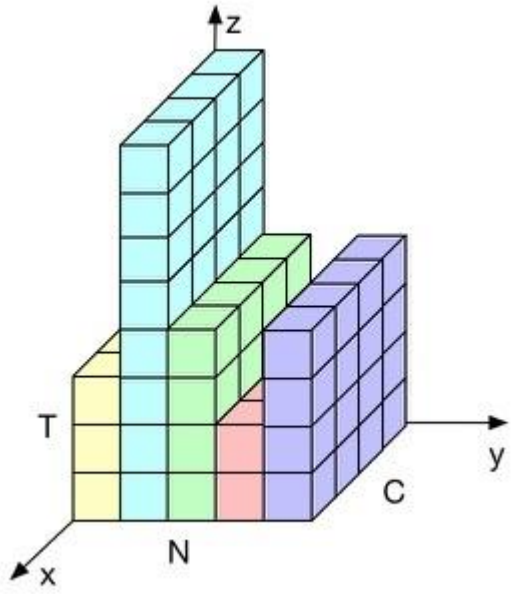
Recurrent Neural Network → In an RNN, the recurrent activations of each time-step will have a different story to tell(i.e. statistics). This means that we have to fit a separate batch norm layer for each time-step. This makes the model more complicated and space consuming because it forces us to store the statistics for each time-step during training.

Layer Normalization

Layer normalization improves the training speed for various neural network models.

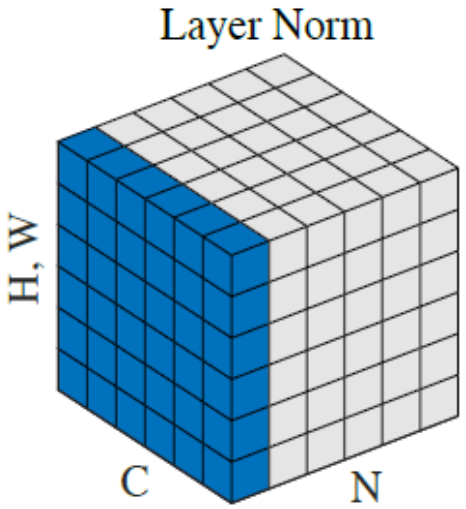
Layer normalization normalizes input across the features instead of normalizing input features across the batch dimension in batch normalization.





Layer Normalization

If we would like to represent real numbers, we have to give up perfect precision.

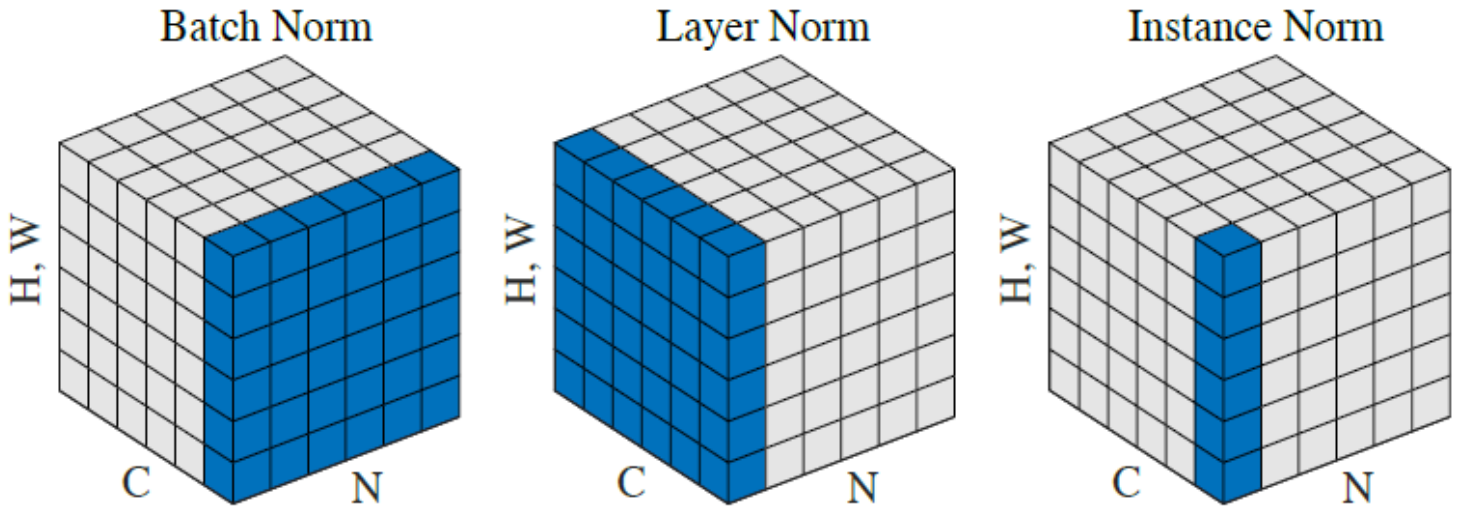


$$\mu^l = \frac{1}{H}\sum_{i=1}^{H} a_i^l \qquad \sigma^l = \sqrt{\frac{1}{H}\sum_{i=1}^{H}\left(a_i^l - \mu^l\right)^2}$$

Instance Normalization

Layer normalization and **instance normalization** is very similar to each other but the difference between them is that instance normalization normalizes across each channel in each training example instead of normalizing across input features in a training example.

Unlike batch normalization, the **instance normalization layer** is applied at test time as well(due to non-dependency of mini-batch).



Instance Normalization

Figure 1: Artistic style transfer example of Gatys et al. (2016) method.

$$y_{tijk} = \frac{x_{tijk} - \mu_{ti}}{\sqrt{\sigma_{ti}^2 + \epsilon}}, \quad \mu_{ti} = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H} x_{tilm},$$

$$\sigma_{ti}^2 = \frac{1}{HW}\sum_{l=1}^{W}\sum_{m=1}^{H}(x_{tilm} - mu_{ti})^2.$$

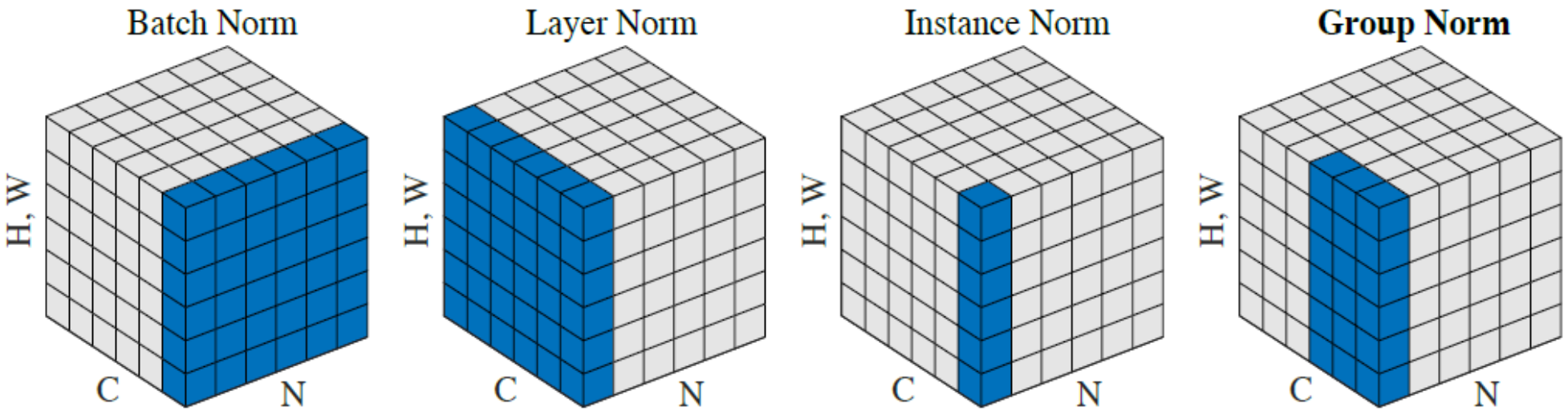
(a) Content image. (b) Stylized image.

(c) Low contrast content image. (d) Stylized low contrast image.

Group Normalization

Group Normalization normalizes over group of channels for each training examples.

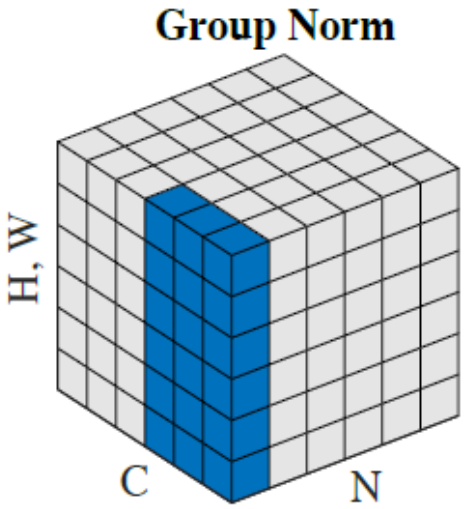
- When we put all the channels into a single group, group normalization becomes Layer normalization.
- When we put each channel into different groups it becomes Instance normalization.



Group Normalization

Group Normalization divides the channels into groups and computes within each group the mean and variance for normalization.

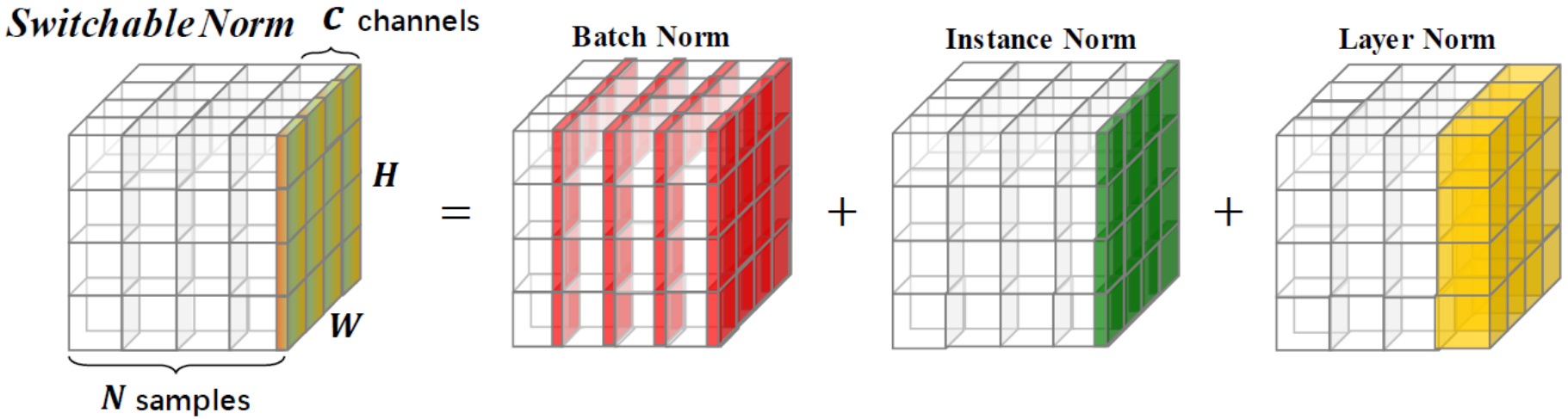
The computation of group normalization is independent of batch sizes, and its accuracy is stable in a wide range of batch sizes.




```
def GroupNorm(x, gamma, beta, G, eps=1e−5):
    # x: input features with shape [N,C,H,W]
    # gamma, beta: scale and offset, with shape [1,C,1,1]
    # G: number of groups for GN

    N, C, H, W = x.shape
    x = tf.reshape(x, [N, G, C // G, H, W])

    mean, var = tf.nn.moments(x, [2, 3, 4], keep_dims=True)
    x = (x − mean) / tf.sqrt(var + eps)

    x = tf.reshape(x, [N, C, H, W])

    return x ∗ gamma + beta
```

Switchable Normalization

Switchable Normalization is a normalization technique that is able to learn different normalization operations for different normalization layers in a deep neural network in an end-to-end manner.



https://github.com/switchablenorms/Switchable-Normalization

Switchable Normalization

SN has an intuitive expression

$$\hat{h}_{ncij} = \gamma \frac{h_{ncij} - \Sigma_{k\in\Omega} w_k \mu_k}{\sqrt{\Sigma_{k\in\Omega} w'_k \sigma_k^2 + \epsilon}} + \beta, \tag{3}$$

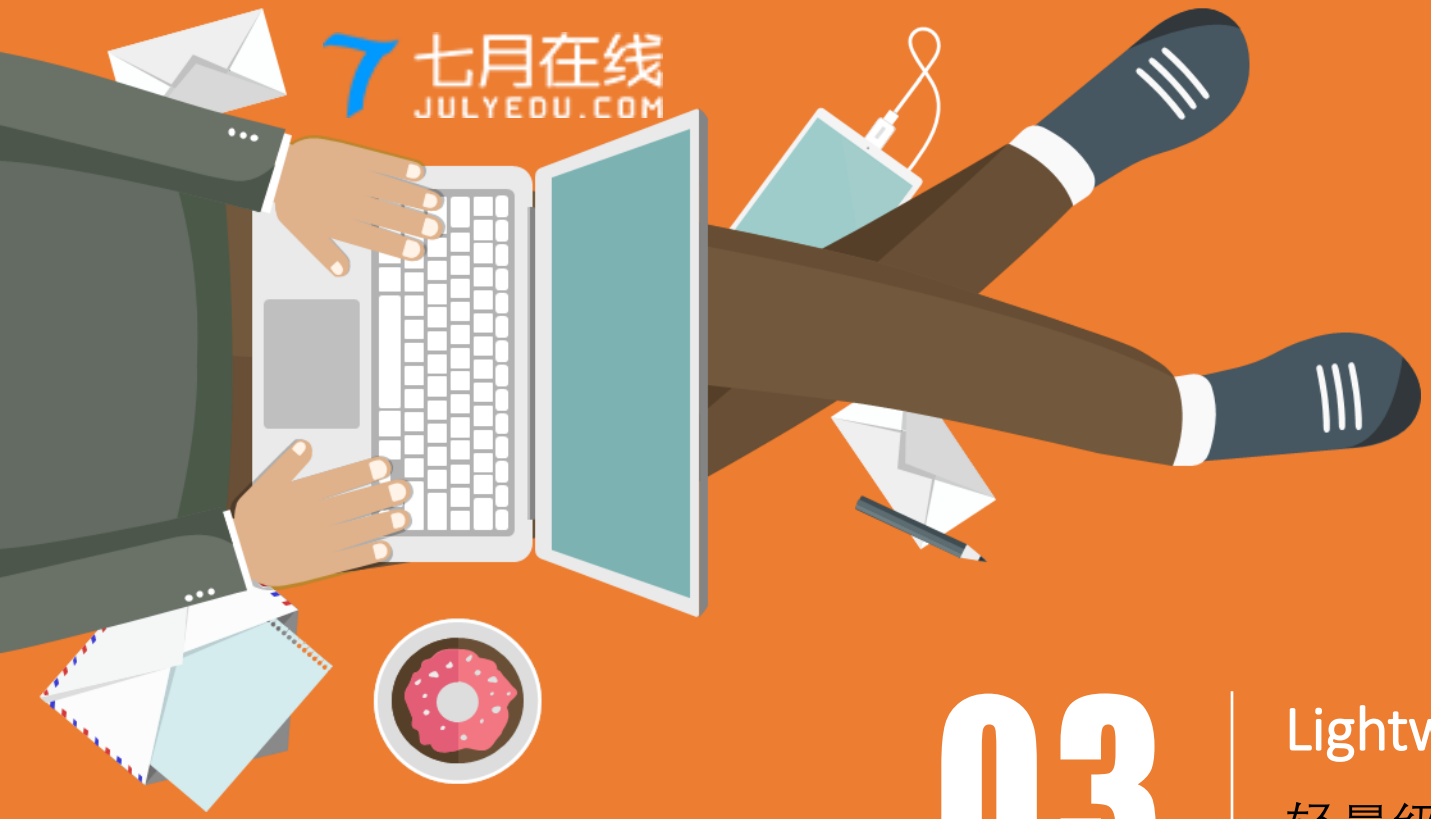
where Ω is a set of statistics estimated in different ways. In this work, we define $\Omega = \{\text{in}, \text{ln}, \text{bn}\}$ the same as above where μ_k and σ_k^2 can be calculated by following Eqn.(2). However, this strategy leads to large redundant computations. In fact, the three kinds of statistics of SN depend on each other. Therefore we could reduce redundancy by reusing computations,

$$\begin{aligned}
\mu_{\text{in}} &= \frac{1}{HW}\sum_{i,j}^{H,W} h_{ncij}, \quad \sigma^2_{\text{in}} = \frac{1}{HW}\sum_{i,j}^{H,W}(h_{ncij} - \mu_{\text{in}})^2, \\
\mu_{\text{ln}} &= \frac{1}{C}\sum_{c=1}^{C}\mu_{\text{in}}, \quad \sigma^2_{\text{ln}} = \frac{1}{C}\sum_{c=1}^{C}(\sigma^2_{\text{in}} + \mu^2_{\text{in}}) - \mu^2_{\text{ln}}, \\
\mu_{\text{bn}} &= \frac{1}{N}\sum_{n=1}^{N}\mu_{\text{in}}, \quad \sigma^2_{\text{bn}} = \frac{1}{N}\sum_{n=1}^{N}(\sigma^2_{\text{in}} + \mu^2_{\text{in}}) - \mu^2_{\text{bn}},
\end{aligned} \tag{4}$$

showing that the means and variances of LN and BN can be computed based on IN. Using Eqn.(4), the computational complexity of SN is $\mathcal{O}(NCHW)$, which is comparable to previous work.

Furthermore, w_k and w'_k in Eqn.(3) are importance ratios used to weighted average the means and variances respectively. Each w_k or w'_k is a scalar variable, which is shared across all channels. There are $3 \times 2 = 6$ importance weights in SN. We have $\Sigma_{k\in\Omega} w_k = 1$, $\Sigma_{k\in\Omega} w'_k = 1$, and $\forall w_k, w'_k \in [0, 1]$, and define

$$w_k = \frac{e^{\lambda_k}}{\Sigma_{z\in\{\text{in},\text{ln},\text{bn}\}} e^{\lambda_z}} \quad \text{and} \quad k \in \{\text{in}, \text{ln}, \text{bn}\}. \tag{5}$$



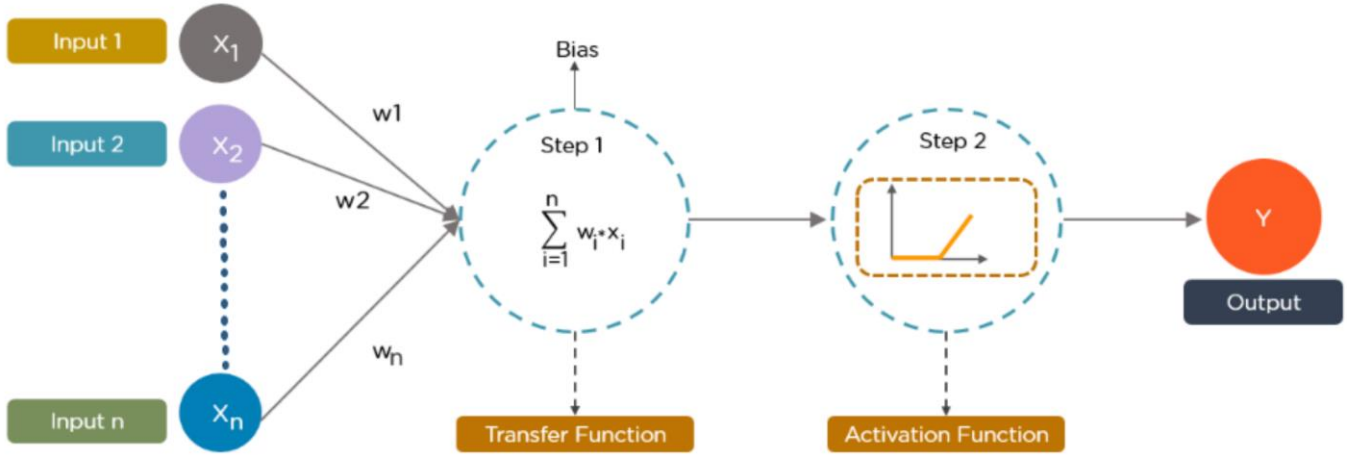
03 Lightweight Neural Networks

轻量级神经网络

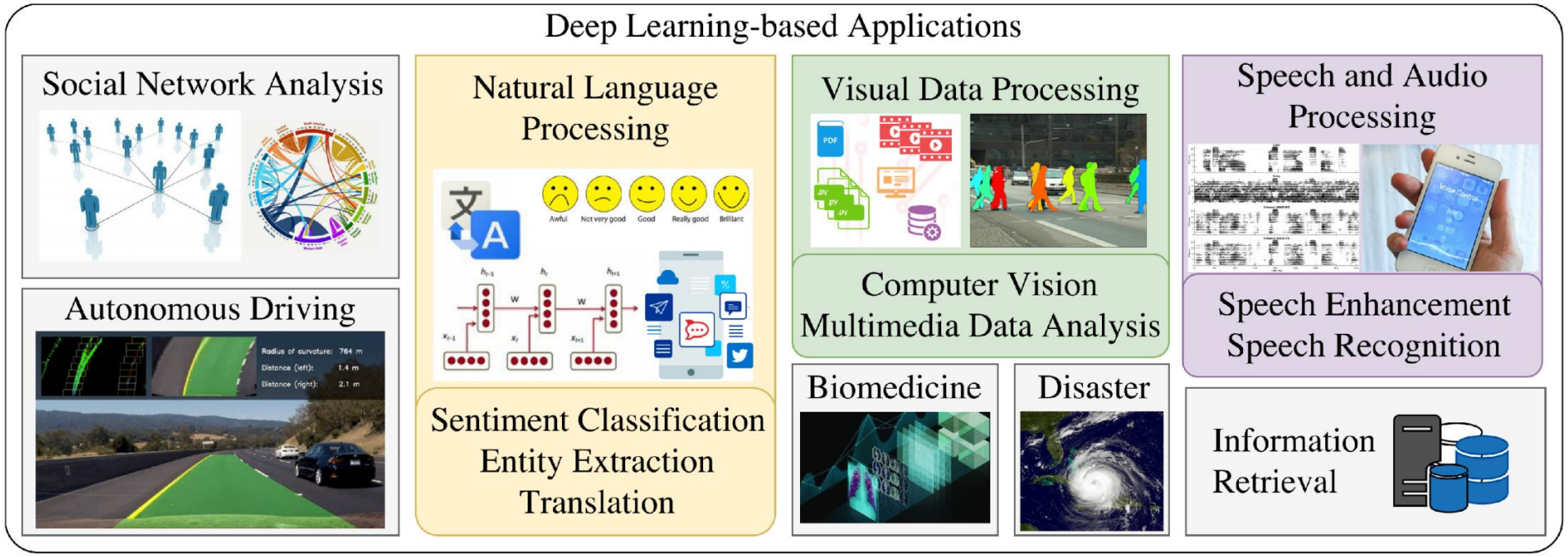
Lightweight Neural Networks

A neural network is structured like the human brain and consists of artificial neurons, also known as nodes.

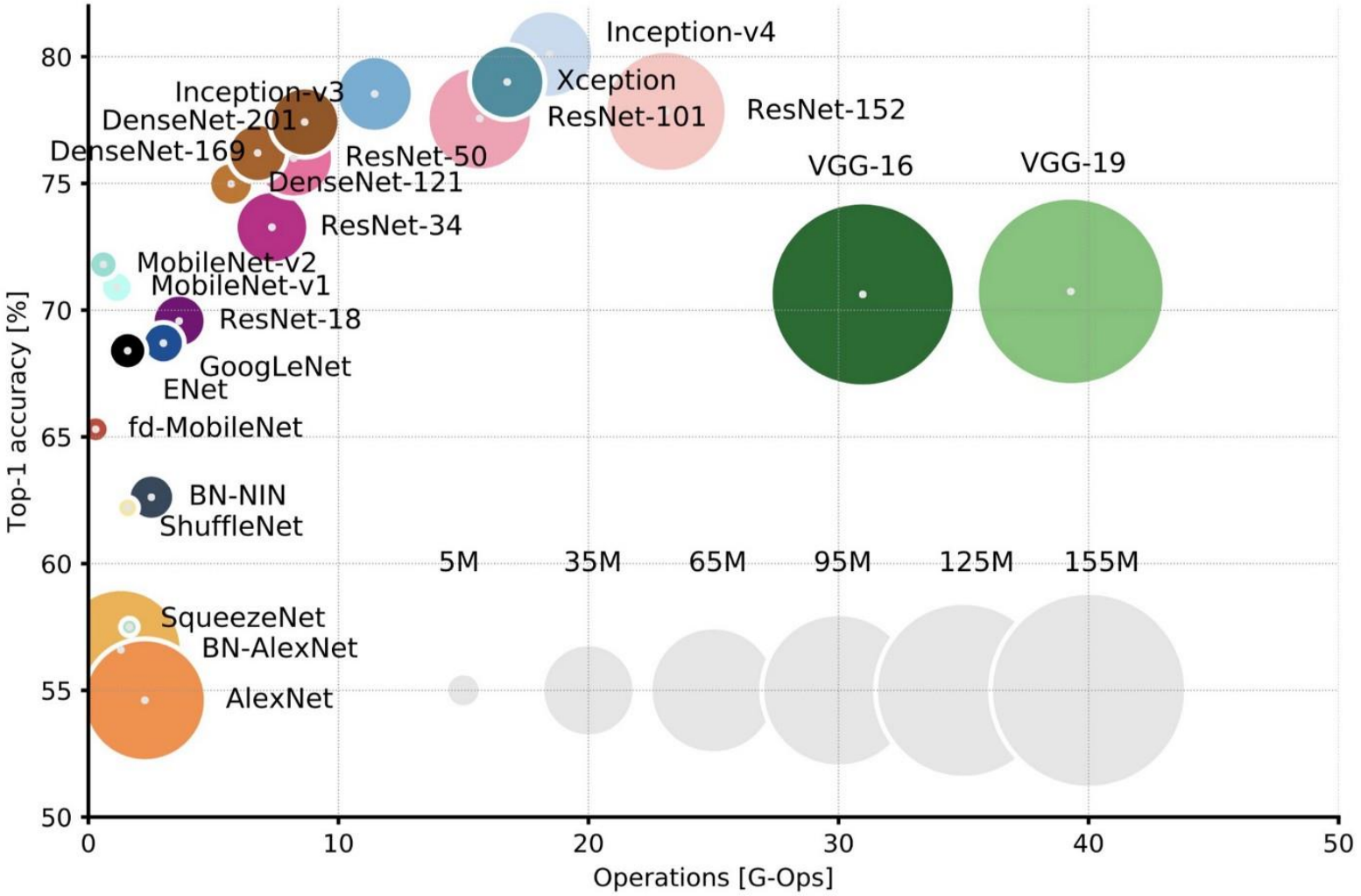
These nodes are stacked next to each other in three layers: the input layer, the hidden layers, and the output layer.



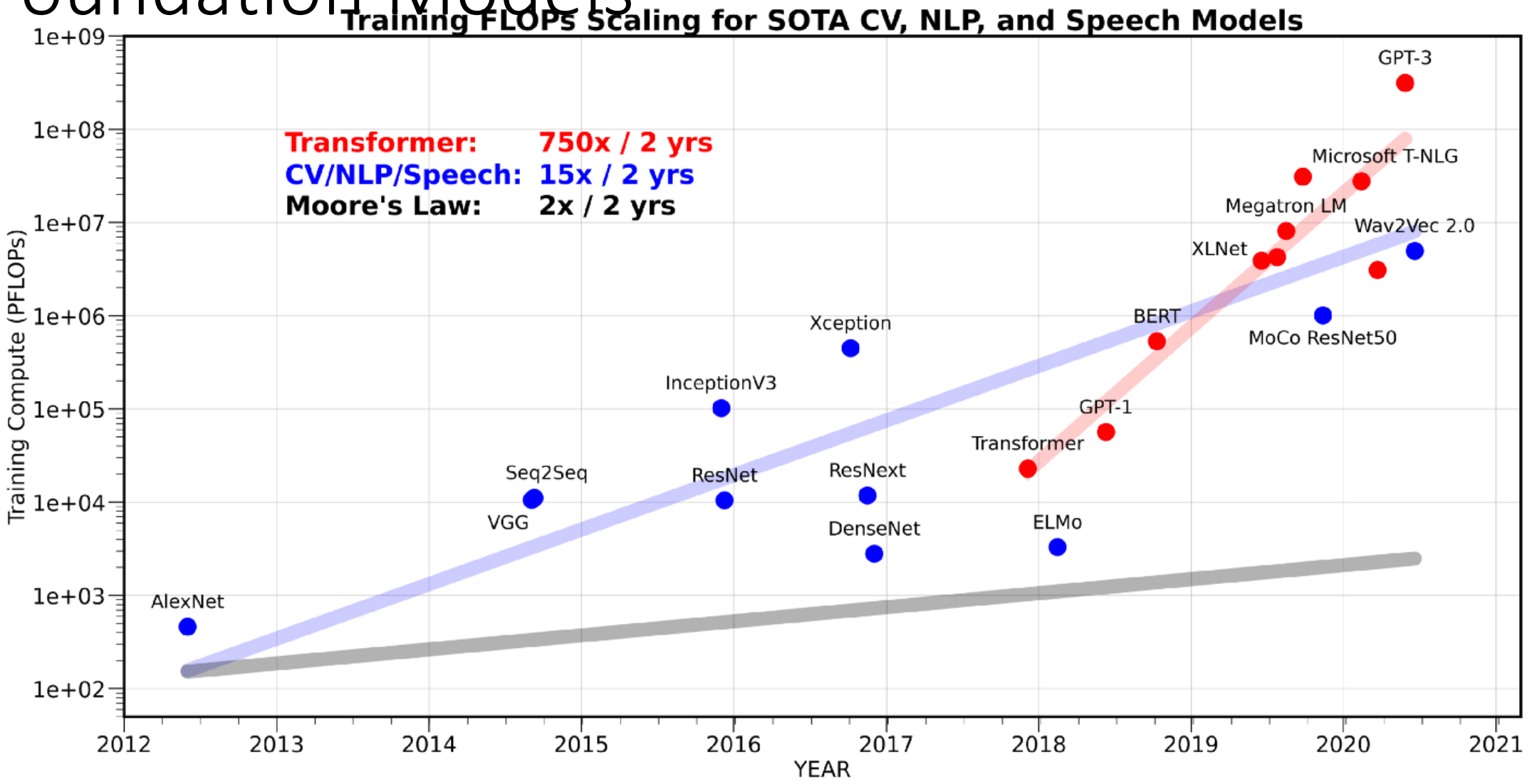
Some Popular Deep Learning Applications



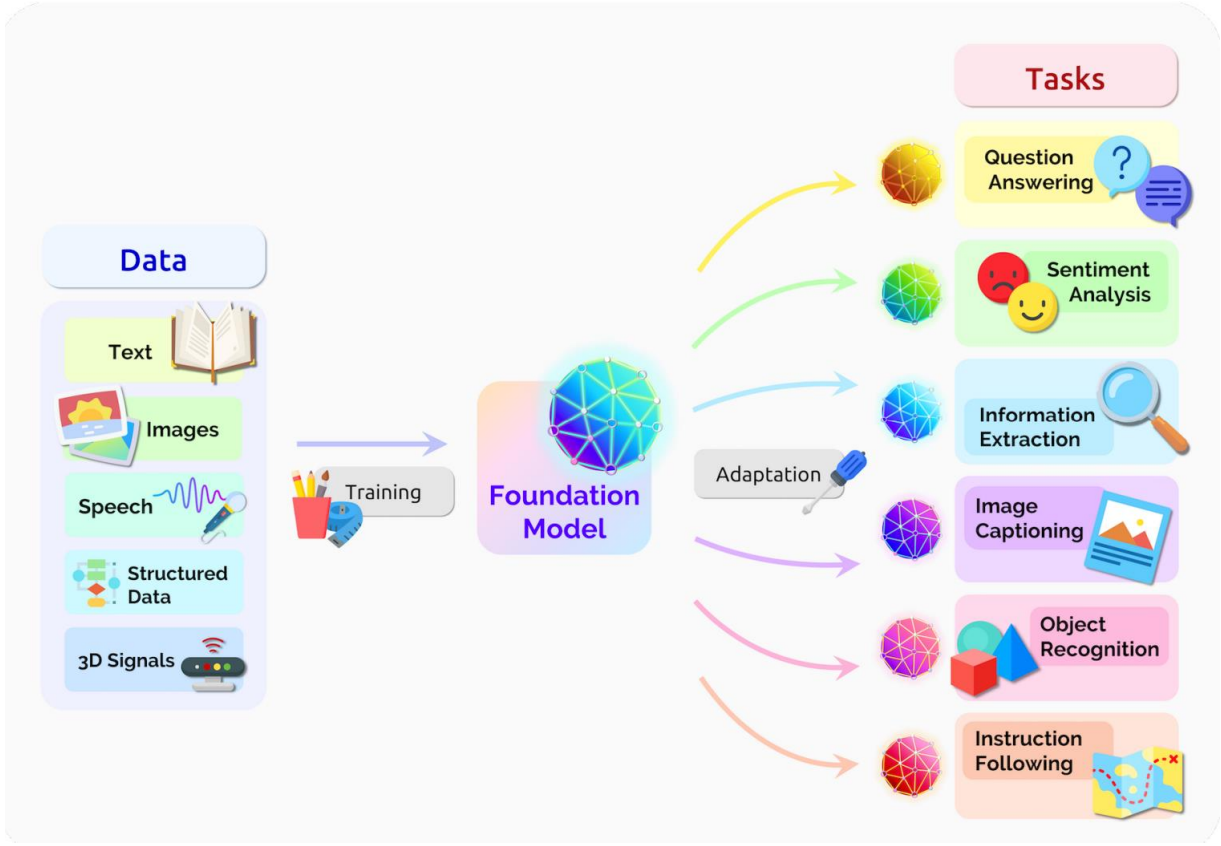
Foundation Models



Foundation Models

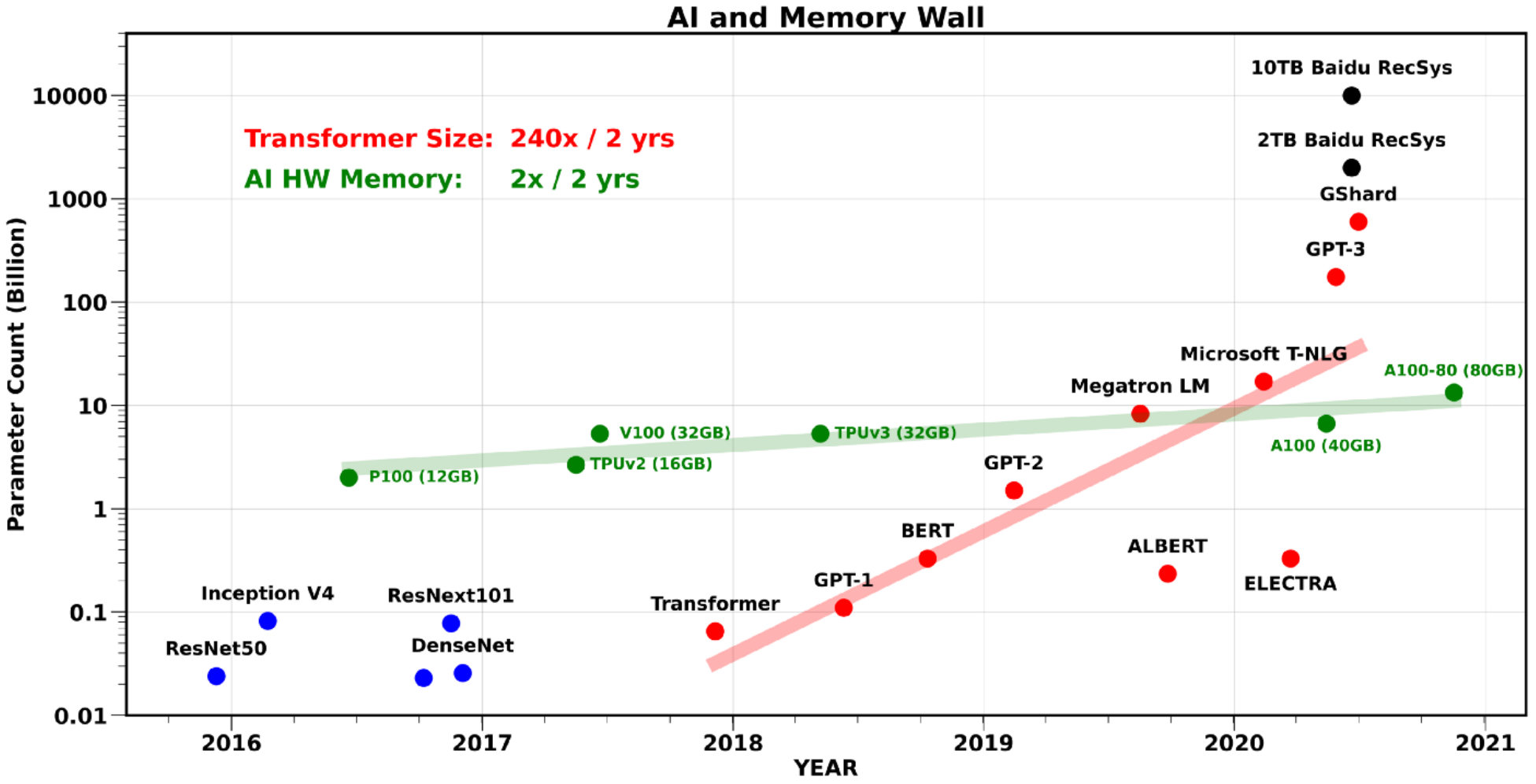


Foundation Models

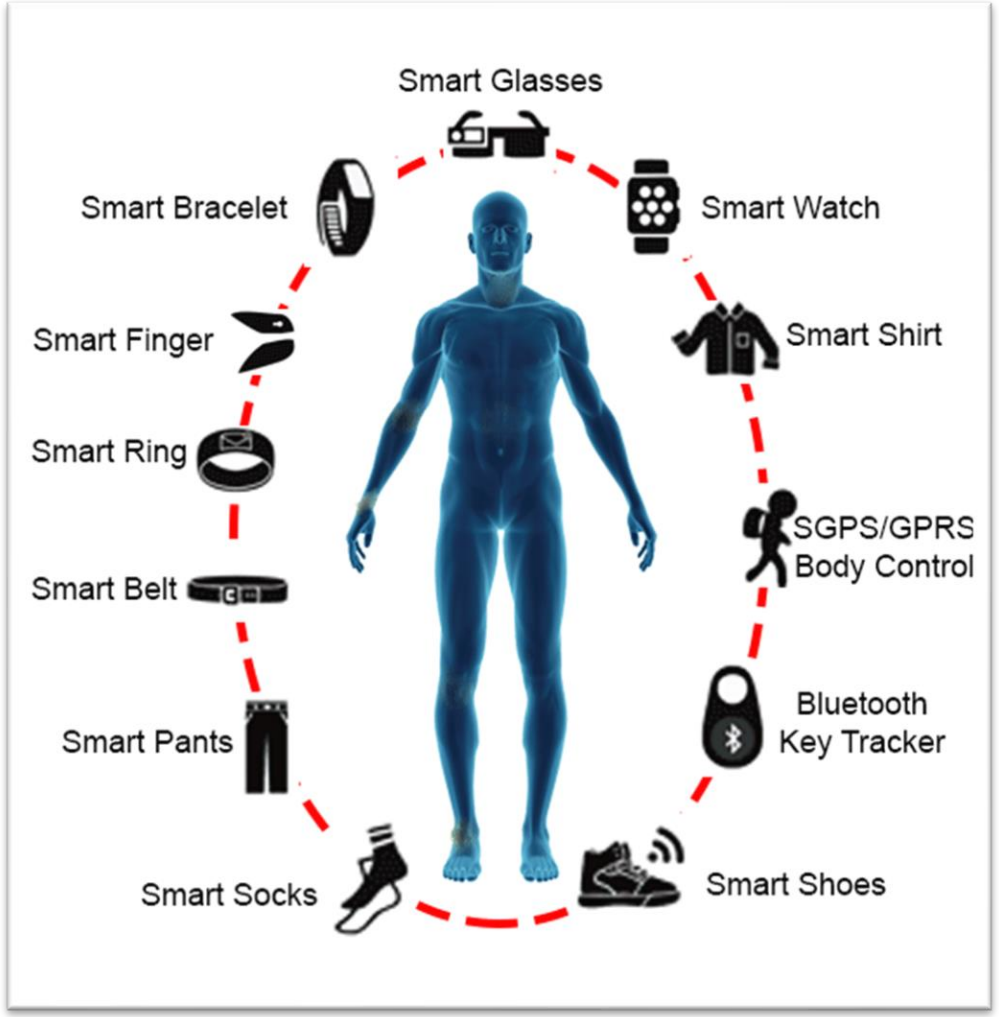


A foundation model can centralize the information from all the data from various modalities. This one model can then be adapted to a wide range of downstream tasks.

Foundation Models



Smart Devices

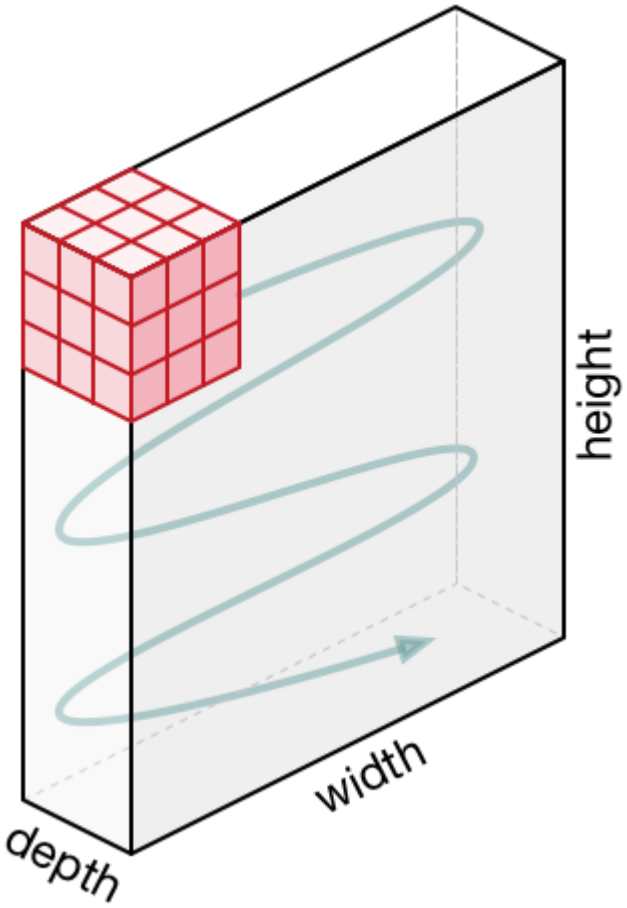


Solution

Architecture Design: 设计更高效的「网络计算方式」（主要针对卷积方式），从而使网络参数减少的同时，不损失网络性能。比如：MobileNet 、MobileViT 、 ShuffleNet

Model Compression: 即在已经训练好的模型上进行压缩，使得网络携带更少的网络参数，同时尽量减少网络性能损失。

Standard Convolution



0	0	0	0	0	0	…
0	156	155	156	158	158	…
0	153	154	157	159	159	…
0	149	151	155	158	159	…
0	146	146	149	153	158	…
0	145	143	143	148	158	…
…	…	…	…	…	…	…

Input Channel #1 (Red)

0	0	0	0	0	0	…
0	167	166	167	169	169	…
0	164	165	168	170	170	…
0	160	162	166	169	170	…
0	156	156	159	163	168	…
0	155	153	153	158	168	…
…	…	…	…	…	…	…

Input Channel #2 (Green)

0	0	0	0	0	0	…
0	163	162	163	165	165	…
0	160	161	164	166	166	…
0	156	158	162	165	166	…
0	155	155	158	162	167	…
0	154	152	152	157	167	…
…	…	…	…	…	…	…

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

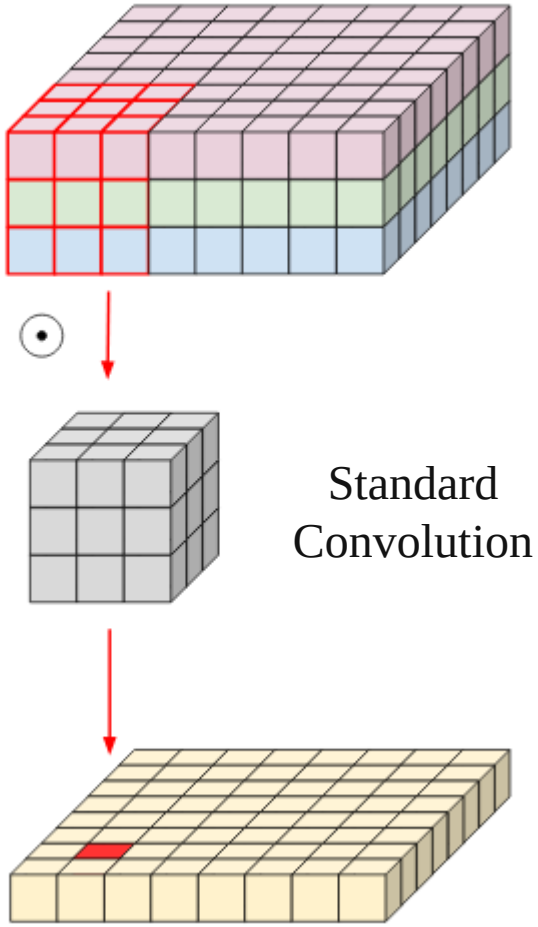
$308 + (-498) + 164 + 1 = -25$

Bias = 1

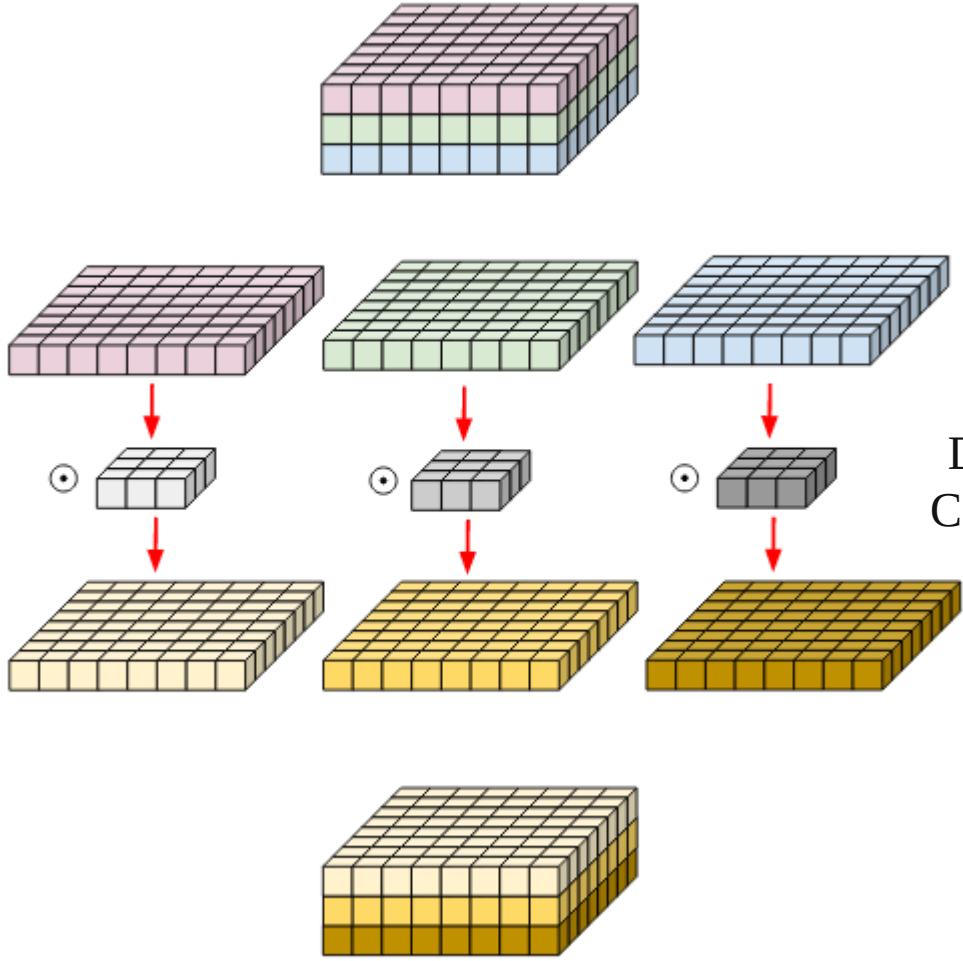
Output

-25				…
				…
				…
				…
…	…	…	…	…

Depthwise Convolution

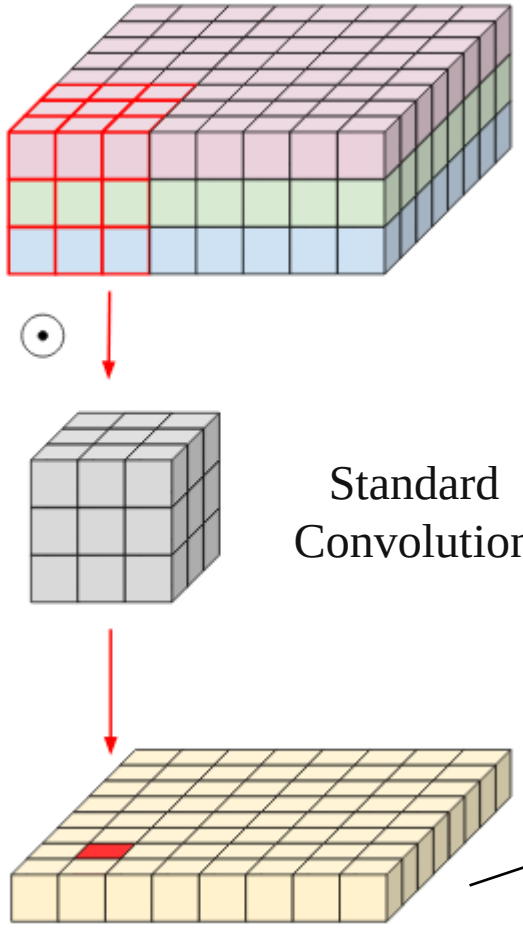


Standard Convolution

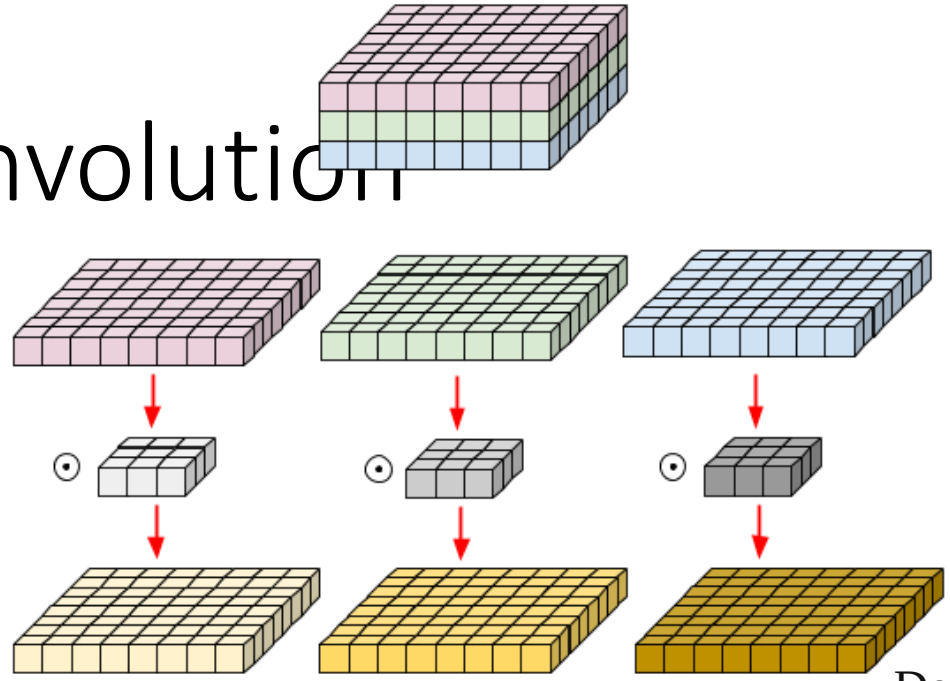


Depthwise Convolution

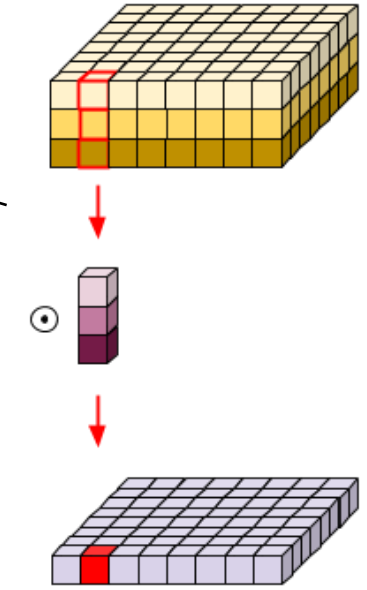
Depthwise Separable Convolution



Standard Convolution

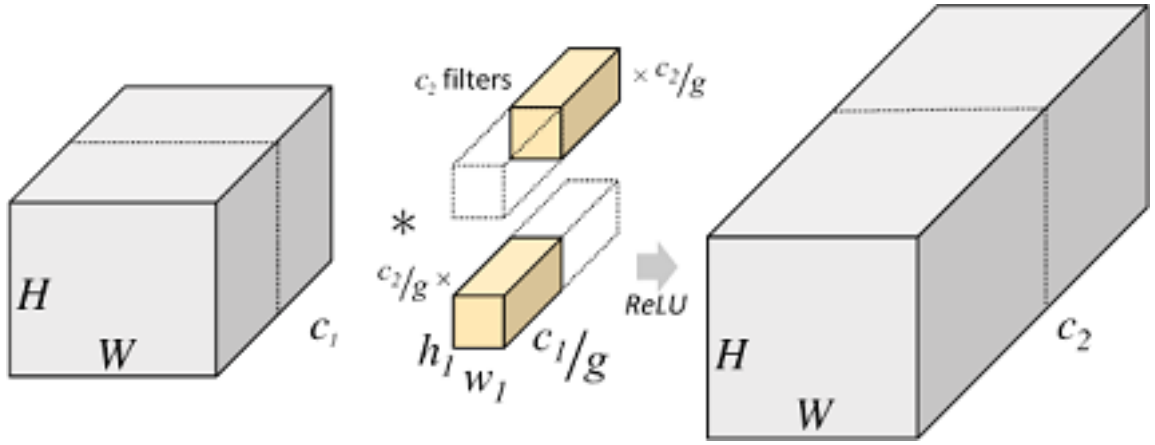


Depthwise Convolution



$$\frac{k \times k \times I + I \times O}{k \times k \times I \times O} = \frac{1}{O} + \boxed{\frac{1}{k \times k}}$$

Group Convolution



Group Convolution与Standard Convolution相比，
总的计算量为 $1/g$，参数量也是标准卷积的 $1/g$ 。

Dilated Convolution

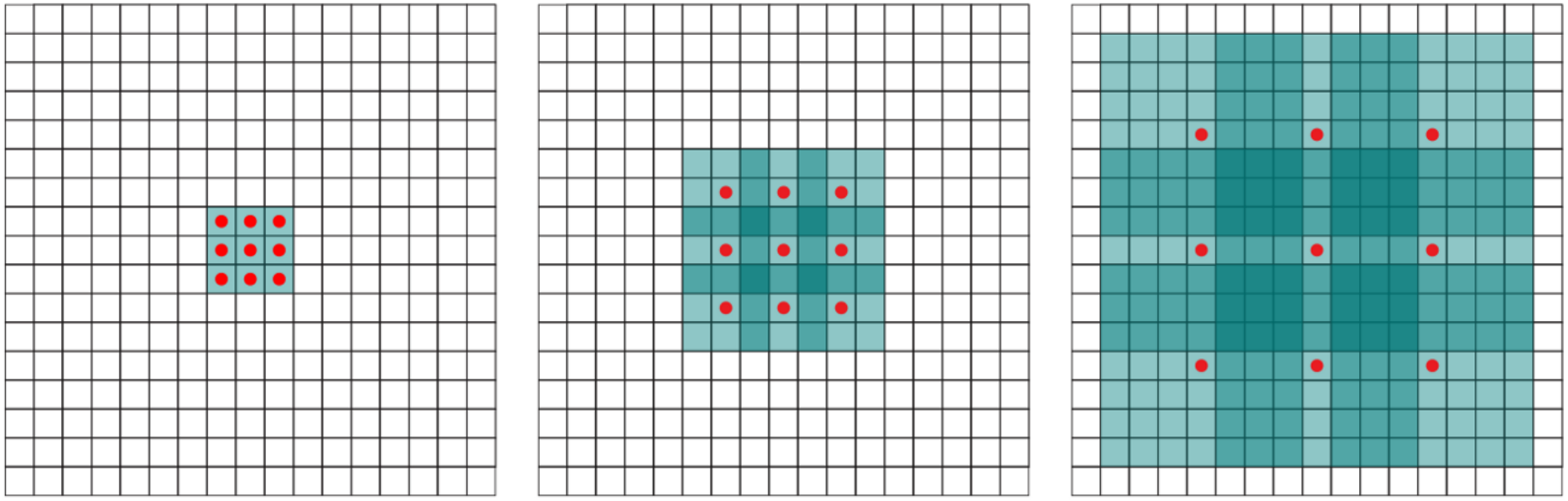
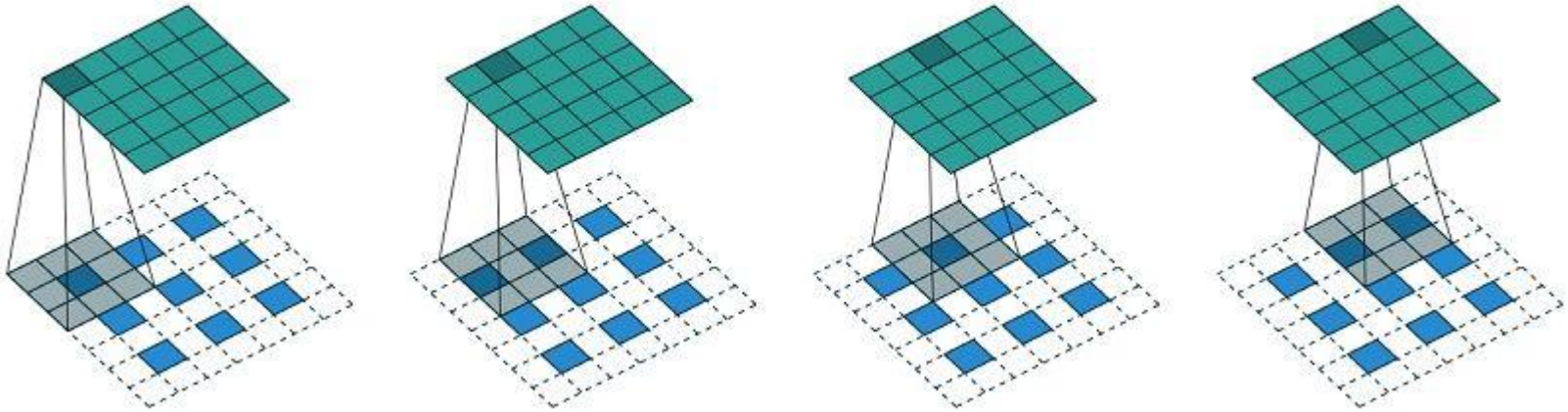


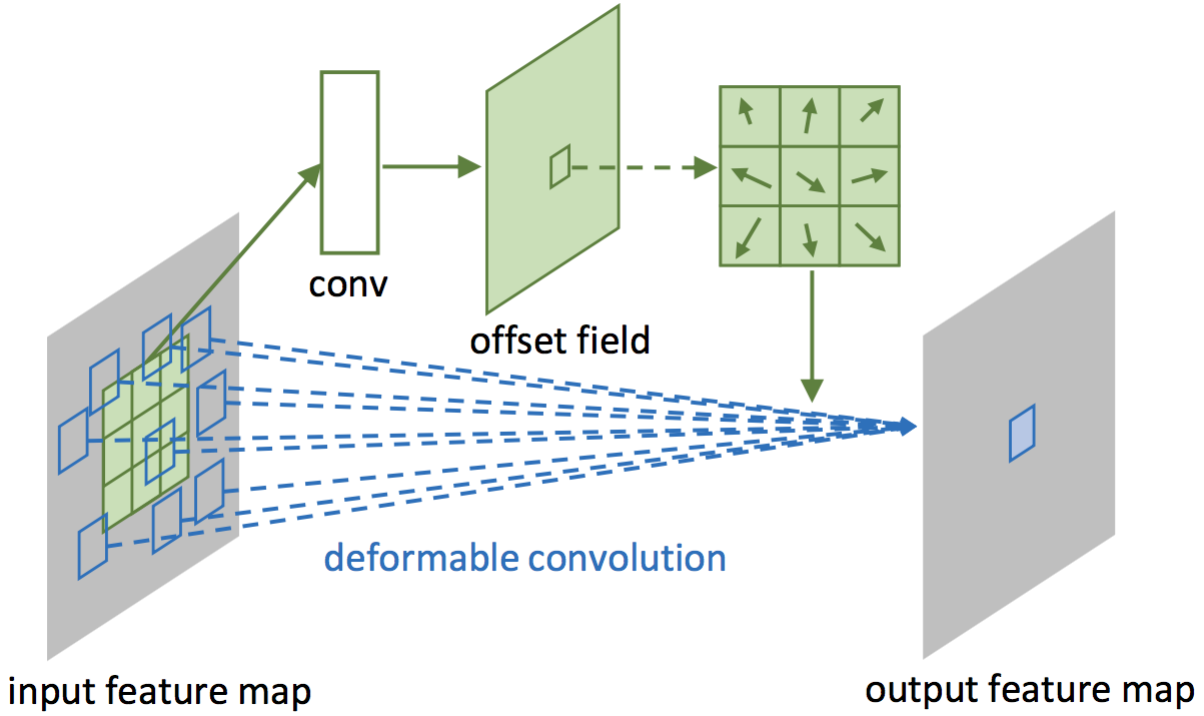
Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

Transposed Convolutions



Transposed Convolutions与Dilated Convolution操作完全相反。

Deformable Convolution



选择题

不属于轻量型神经网络的网络结构是( )

（A）MobileNet　　（B）ShuffleNet

（C）MobileViT　　（D）DenseNet

Q&A