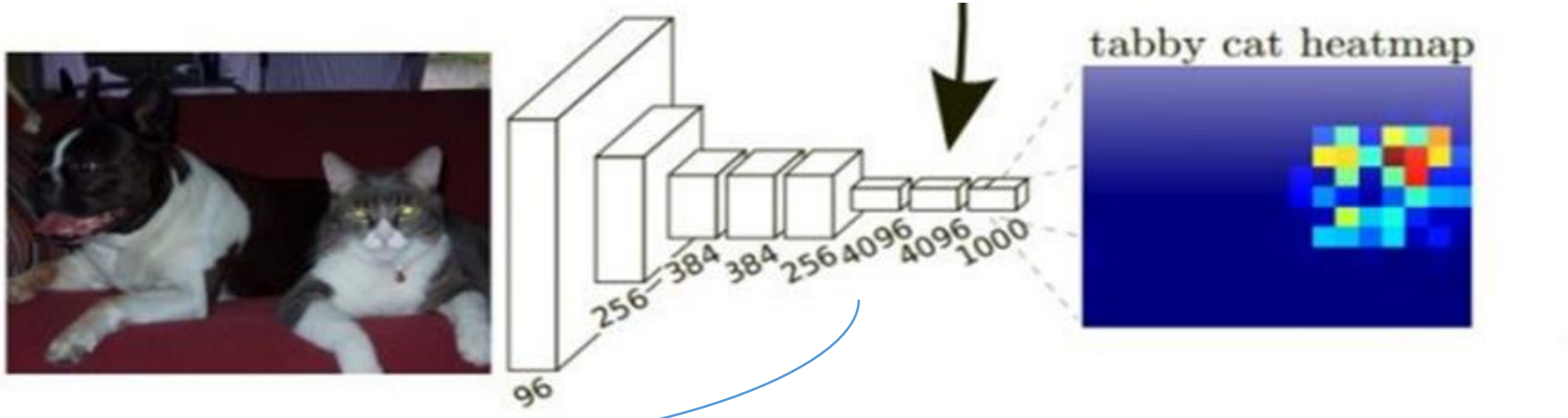
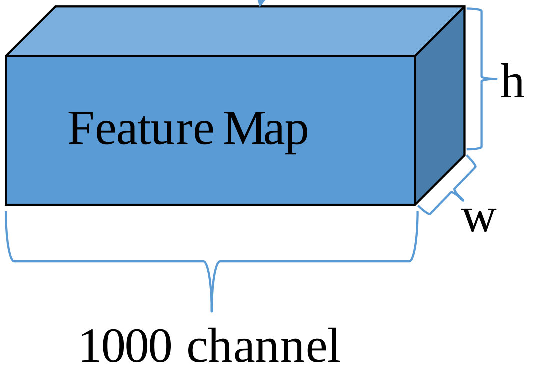
深度學習
Pytorch手把手實作
模型

黃志勝
義隆電子人工智慧研發部
國立陽明交通大學 AI學院 合聘助理教授
國立台北科技大學 電資學院 合聘助理教授

Classification

- Feature map可以做什麼? **這份資料強調在分類任務上。**





- ➢ 這張圖是貓還是狗 (Classification)
- ➢ 框出圖片內有貓和狗的位置 (Object detection)
- ➢ 從框出的物件(貓和狗)中的描繪出實際物件的輪廓 (Instance Segmentation)
- ➢ 把圖片描繪出物件(貓和狗)的輪廓 (Semantic Segmentation)

Classical Deep Learning Model

卷積神經網路(CNN)的核心

- 1. Convolution
- 2. Pool
- 3. Batch Normalization
- 4. Activation function

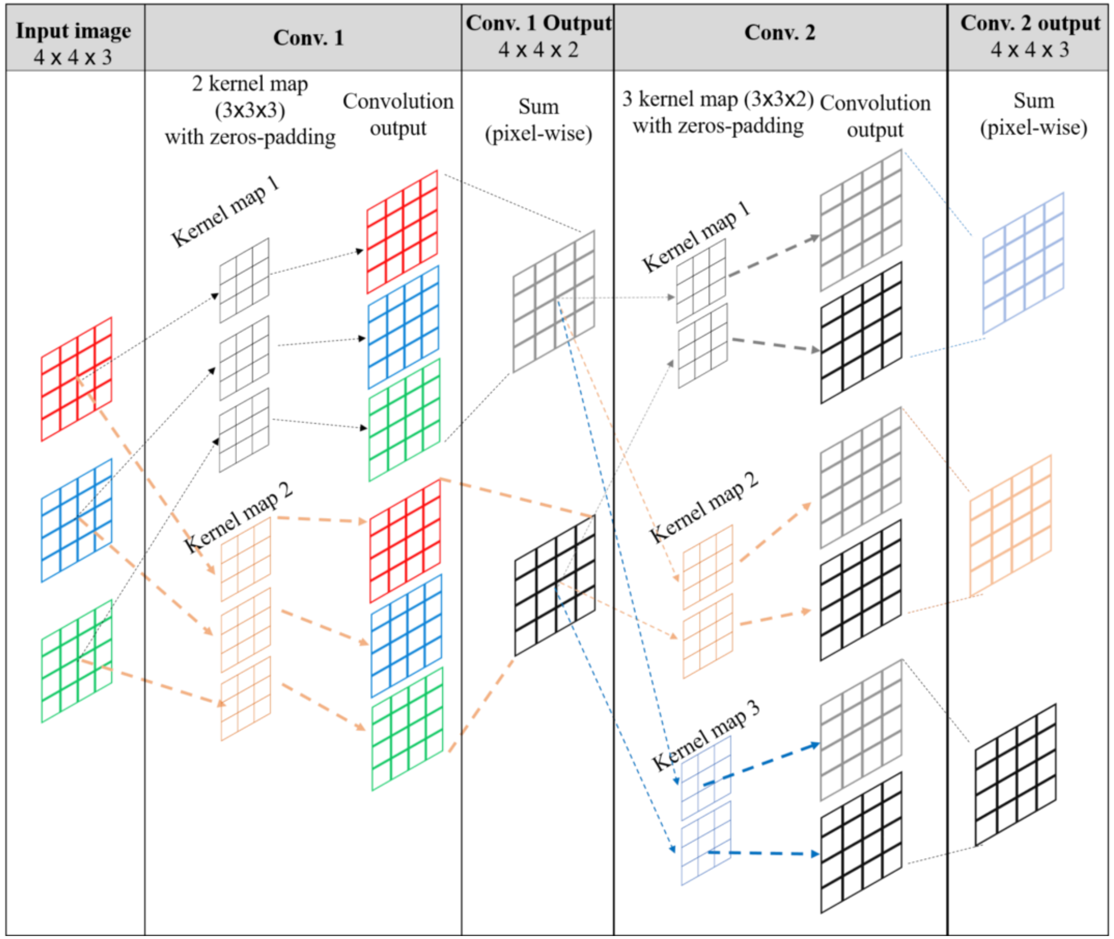
Classical Deep Learning Model

<u>**卷積神經網路(CNN)的核心**</u>

- 1. Convolution → 怎麼挑/設計更好的卷積結構 (大部分是這個)
- 2. Pool
- 3. Batch Normalization→Layer Normalization, Instance Normalization, Group Normalization.
- 4. Activation function→ ReLU家族系列

Convolution

Multi-channel Feature maps做Conv.



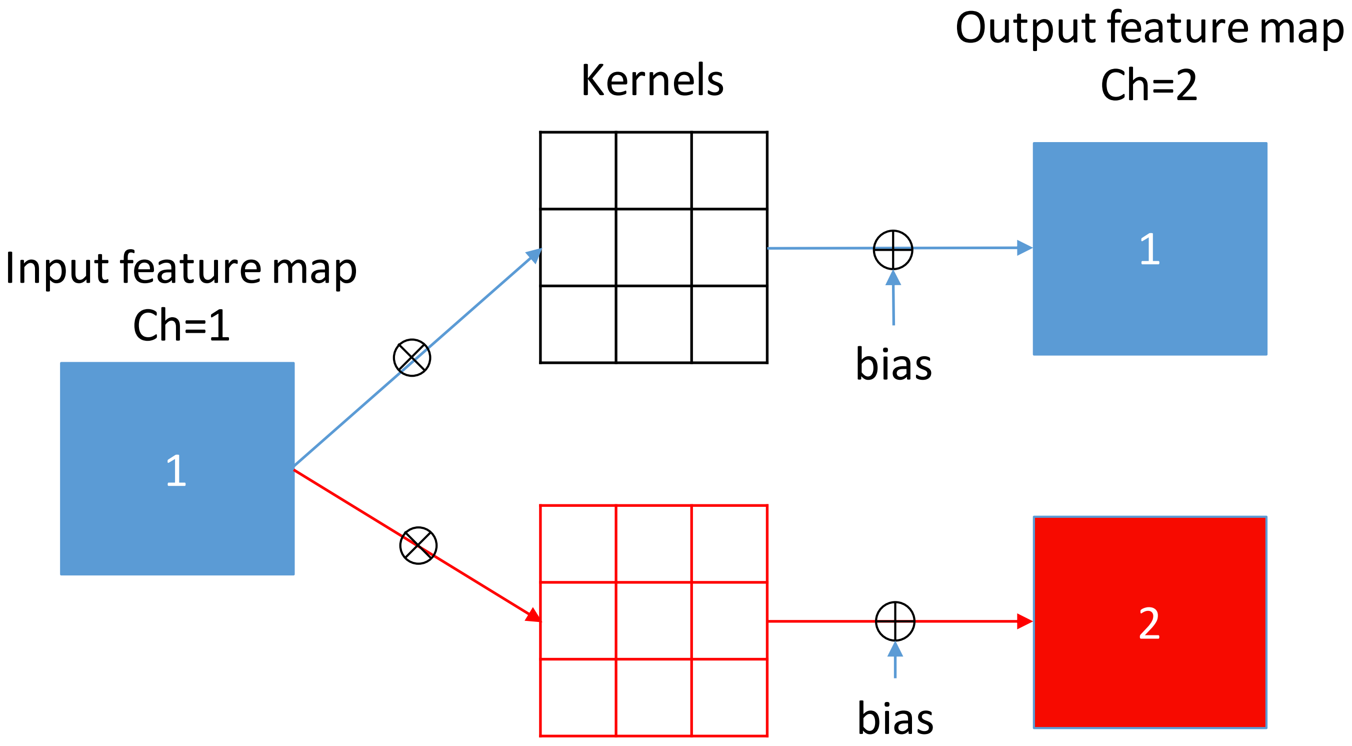
Reference: https://reurl.cc/RbM2r

Conv 1 :
參數量
Kernel map1=(3*3)*3=27
Kernel map2=(3*3)*3=27
27+27=54

Conv 2 :
參數量
Kernel map1=(3*3)*2=18
Kernel map2=(3*3)*2=18
Kernel map3=(3*3)*2=18
18+18+18=54

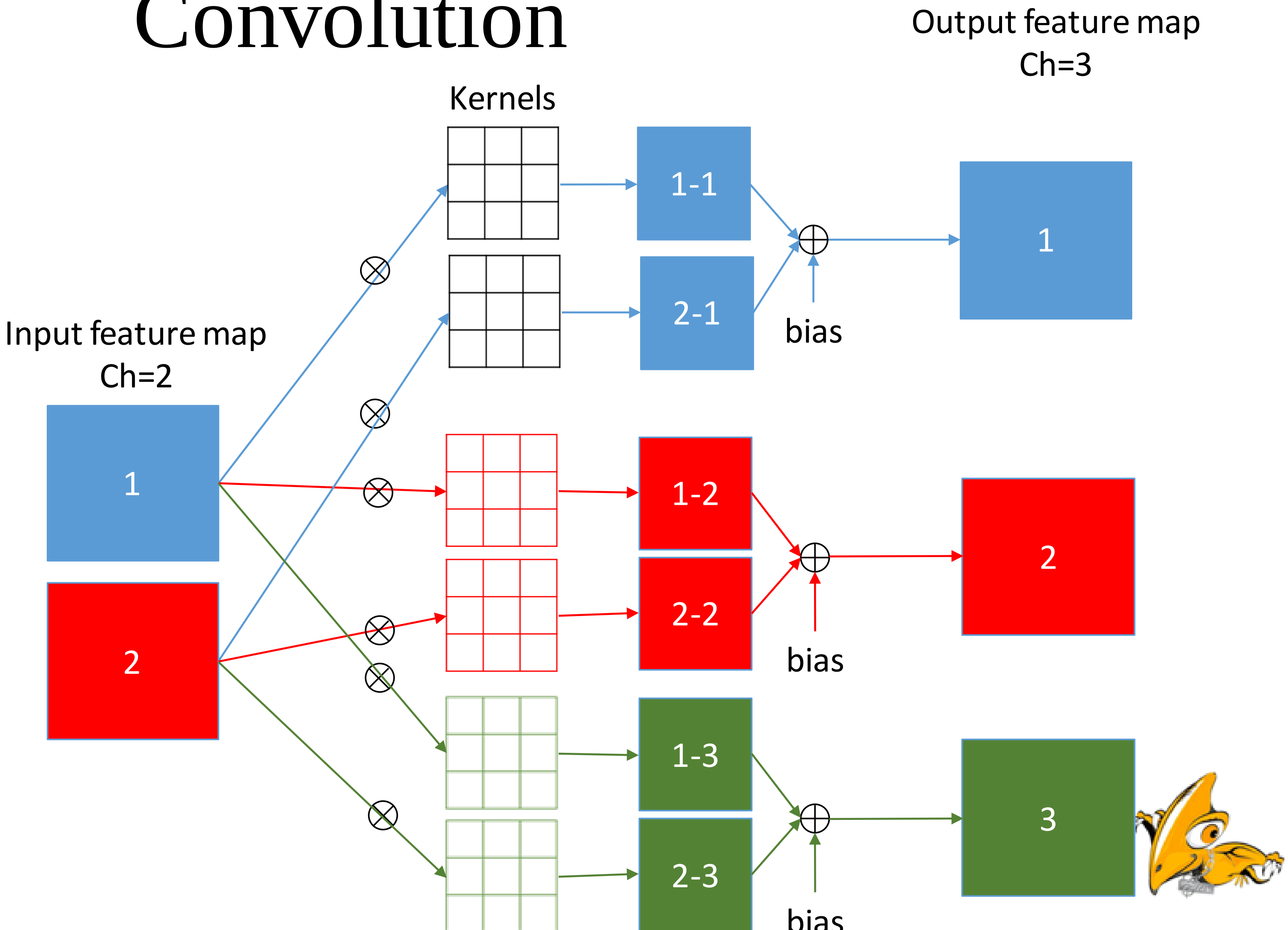
Convolution

- Example:
- In channel = 1,
- Output channel = 2
- Kernel size = 3
- bias



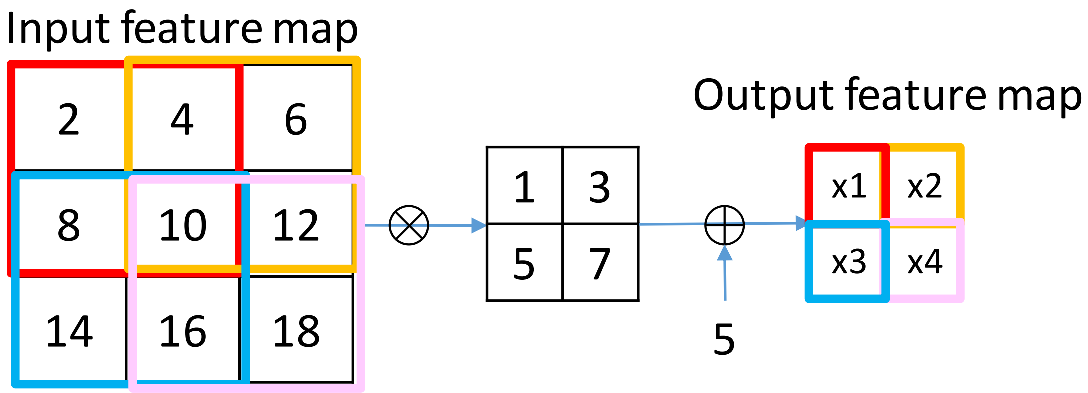
Convolution

- Example:
- In channel = 2,
- Output channel = 3
- Kernel size = 3
- bias



Example 1: Conv.

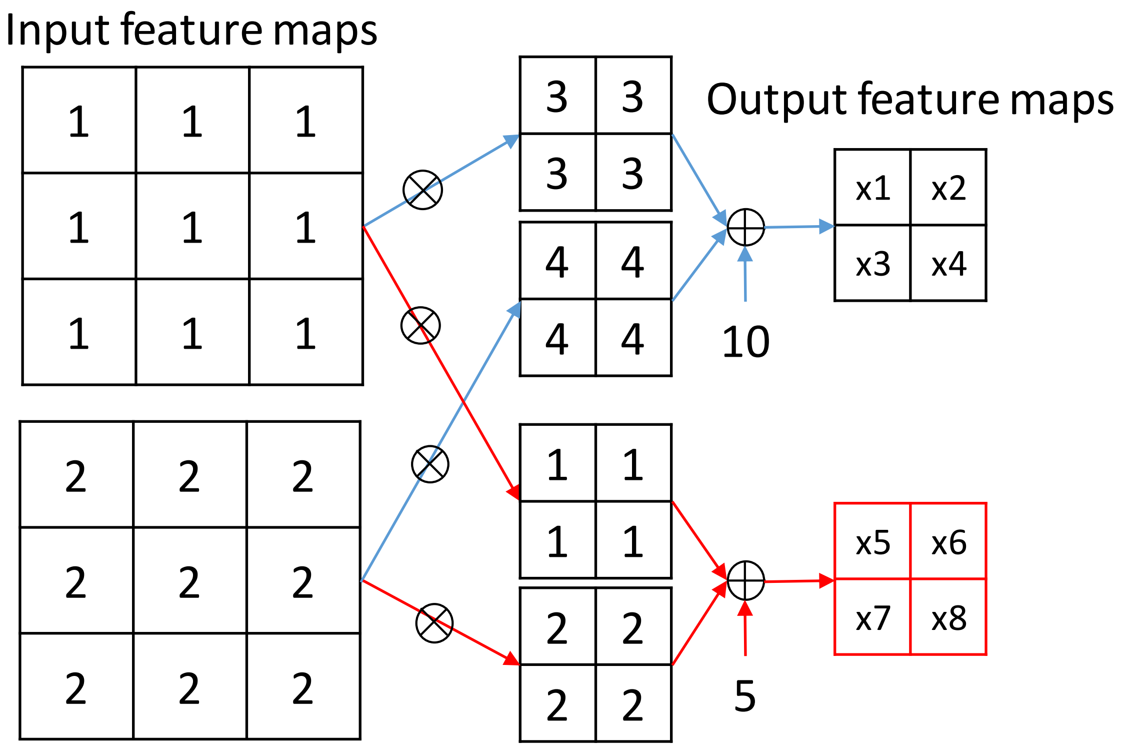
- Example:
- In channel = 1,
- Output channel = 1
- Kernel size = 2
- Bias=5



$$x1 = (2 \times 1 + 4 \times 3 + 8 \times 5 + 10 \times 7) + 5 = 129$$
$$x2 = (4 \times 1 + \ 6 \times 3 + 10 \times 5 + 12 \times 7) + 5 = 161$$
$$x3 = (8 \times 1 + 10 \times 3 + 14 \times 5 + 16 \times 7) + 5 = 225$$
$$x4 = (10 \times 1 + 12 \times 3 + 16 \times 5 + 18 \times 7) + 5 = 257$$

Example 2: Conv.

- Example:
- In channel = 2,
- Output channel = 2
- Kernel size = 2
- Bias = [10, 5]

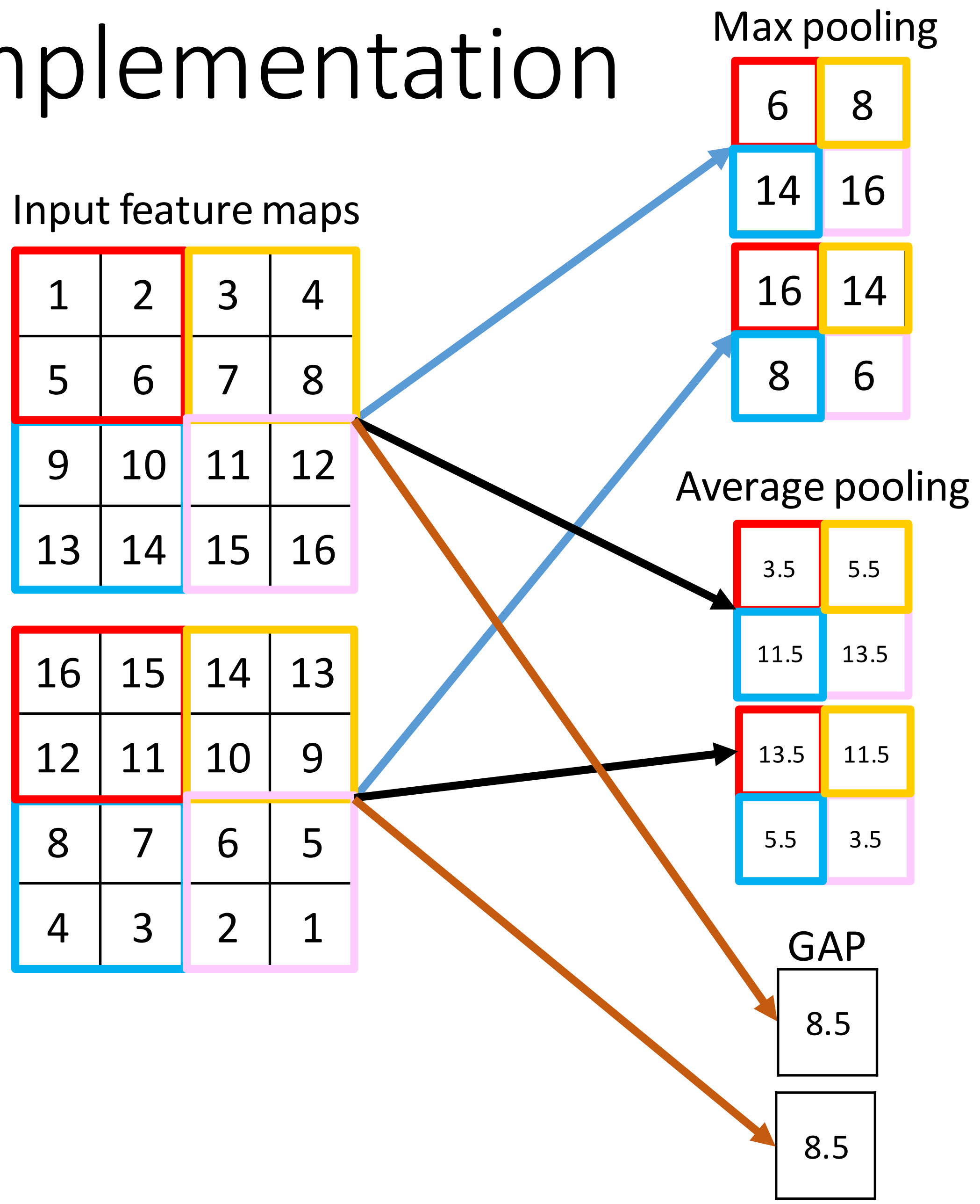


$$x1 = x2 = x3 = x4$$
$$= (1 \times 3 + 1 \times 3 + 1 \times 3 + 1 \times 3) + (2 \times 4 + 2 \times 4 + 2 \times 4 + 2 \times 4) + 10 = 12 + 32 + 10 = 54$$

$$x5 = x6 = x7 = x8$$
$$= (1 \times 1 + 1 \times 1 + 1 \times 1 + 1 \times 1) + (2 \times 2 + 2 \times 2 + 2 \times 2 + 2 \times 2) + 5 = 25$$

Pool Implementation

- 1. Max-pooling: ks=2, stride=2
- 2. Average pooling : ks=2, stride=2
- 3. Global Average Pooling



Activation function→ ReLU家族系列

- Sigmoid 、Tanh 。

ReLU(Rectified Linear Unit)系列：

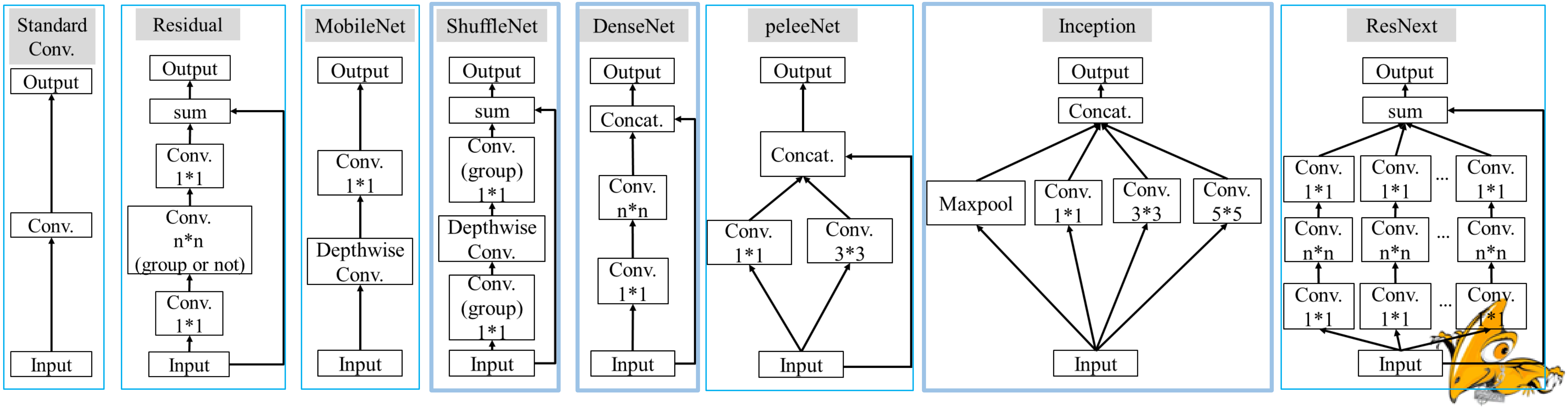
- ReLU
- Leaky ReLU
- ReLU6
- Mish

See Jupyter

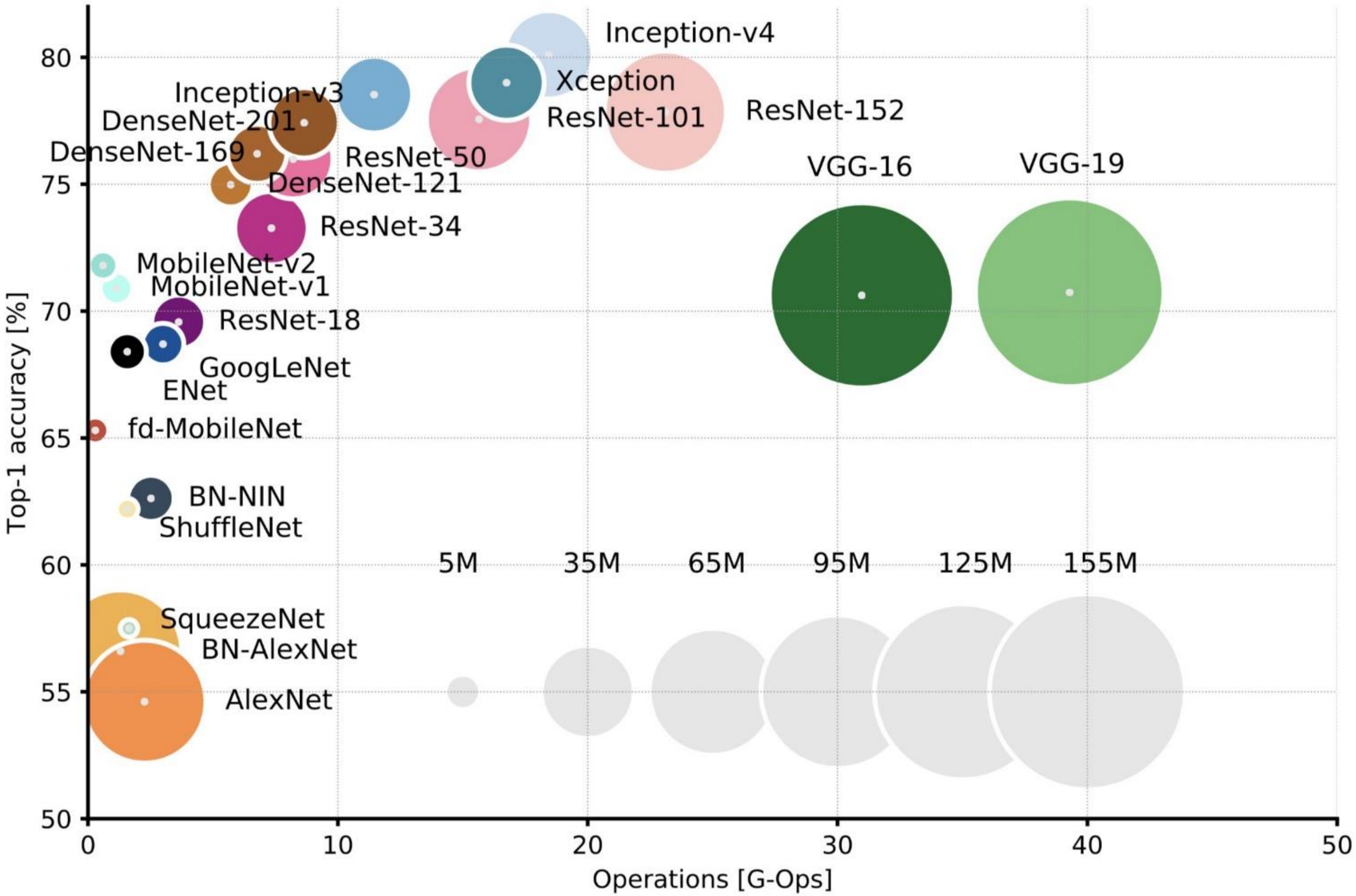
Classical Deep Learning Model

- Deep learning論文提出的NN component

1. 讓模型在每一層自己學哪個NN component大小合適(maxpool, 1*1 conv, 3*3 conv, 5*5 conv,… etc.)
2. 提出新的卷積架構減低標準卷積運算(group conv.)
3. 總和不同方法組合成新的NN component

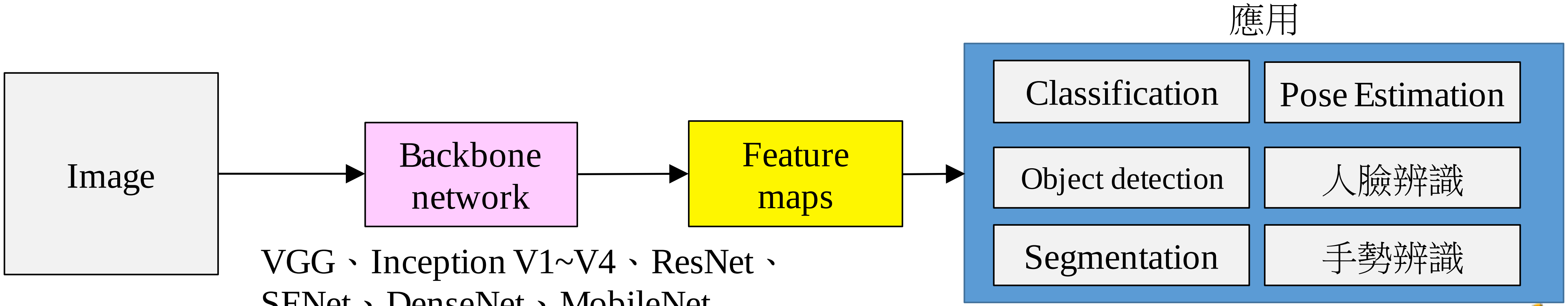


基於ImageNet比賽提出的分類模型



影像上的任務和深度學習模型的關聯

- 不論影像上的任務是什麼
- 現有的任務會採用深度學習當作Feature extraction的腳色，從深度學習模型中得到feature map，而這個feature extraction的主幹模型我們稱為Backbone。
- 從萃取出來的特徵圖在去做後面的應用(Classification、Object detection、Segmentation、Pose Estimation、手勢辨識和人臉辨識)。



VGG、Inception V1~V4、ResNet、SENet、DenseNet、MobileNet 都是建立在ImageNet分類基礎上。

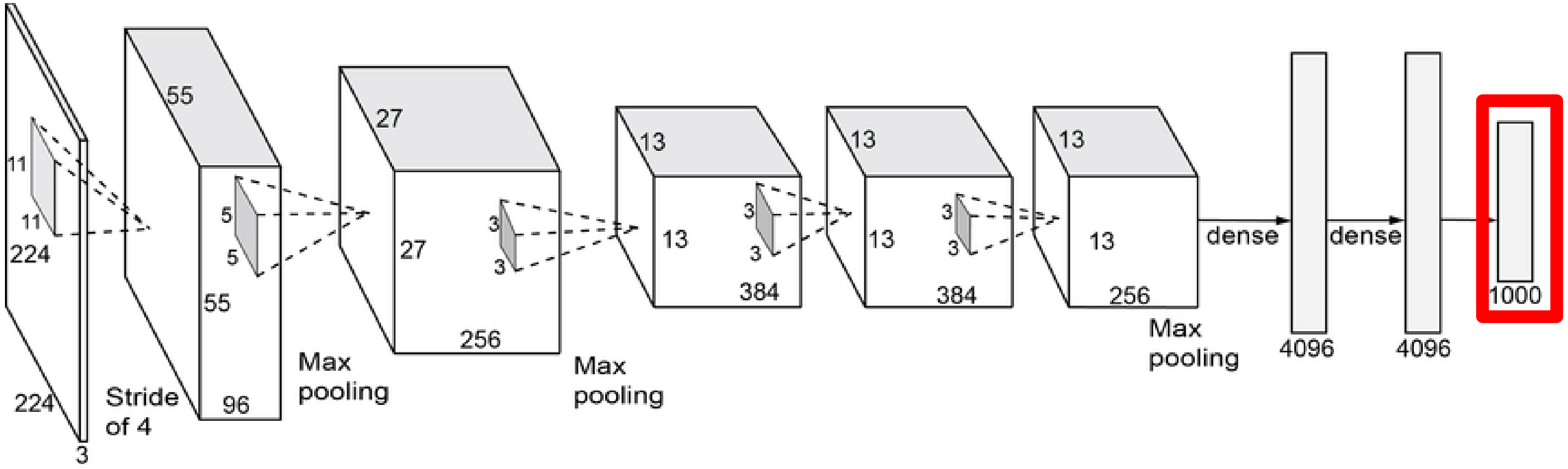
不同深度學習神經網路模型的演進

- **LeNet (1998)** － 算是第一個卷積神經網路
- AlexNet (ILSVRC2012) －第一個用GPU訓練的深度學習網路(ReLU)
- ZFNet (ILSVRC2013) － 第一個用deconv並且可視覺化解釋神經網路運作。
- NIN (Network In Network) – 提出1*1 conv和global average pooling
- **VGG (ILSVRC2014 2014, runner-up)** －加深網路。
- **GoogLeNet (ILSVRC2014, Winner)** －提出Inception Block (Inception V1)
- Inception V2 (2015) －提出Batch Normalization
- **ResNet (ILSVRC2015)** －提出 residual block。
- SENet (ILSVRC2017) － 提出squeeze-excitation block達到feature re-calibration
- **DenseNet (2016)** － 提出Dense block，讓前幾層的feature map可以繼續傳遞給後幾層卷積使用。
- **SqueezeNet(2016)**
- **MobileNet (2017)** －提出depthwise separable convolution。

模型結構 - Classification

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1 Conv / s1	3 × 3 × 512 dw 1 × 1 × 512 × 512	14 × 14 × 512 14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000



Pytorch Example

- 手刻 ResNet-18