

深度學習

-補充範例

-卷積神經網路在看什麼?

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

台北科技大學 業師



Outline

- 補充說明

1. Iteration or Epoch

2. MNIST實例: Batch normalization and ReLU 對模型影響

3. MNIST實例: Residual Block對模型影響

- CNN取什麼特徵?

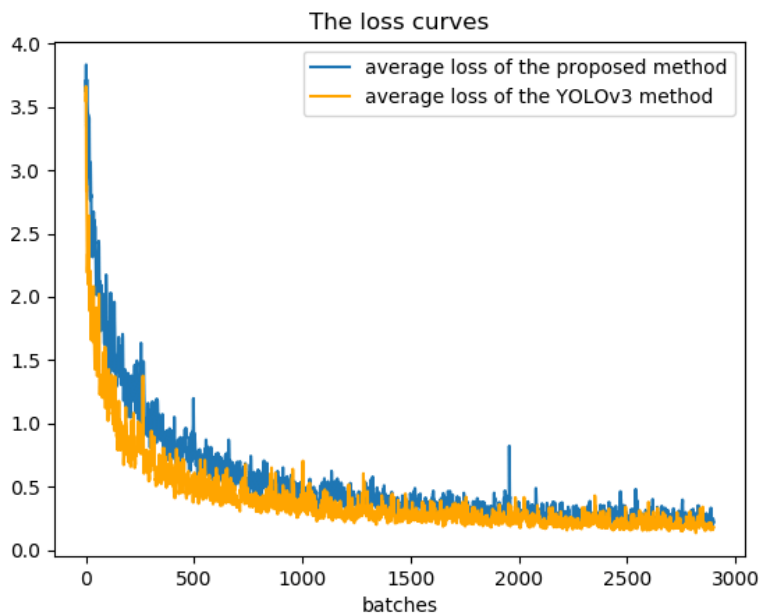
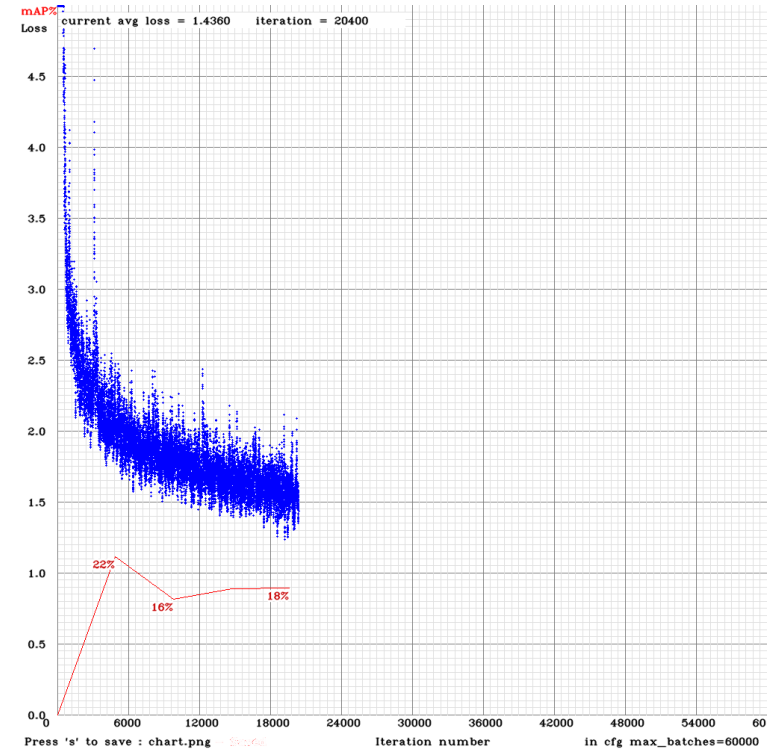
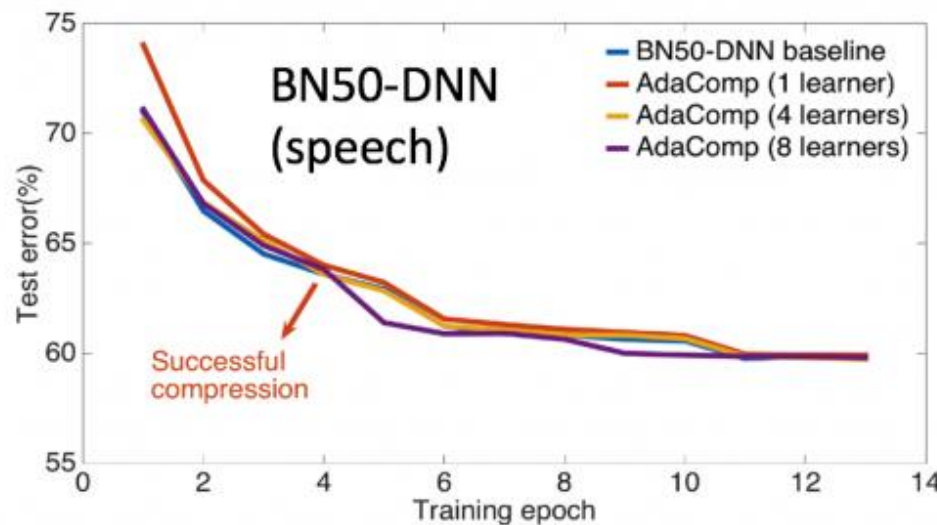
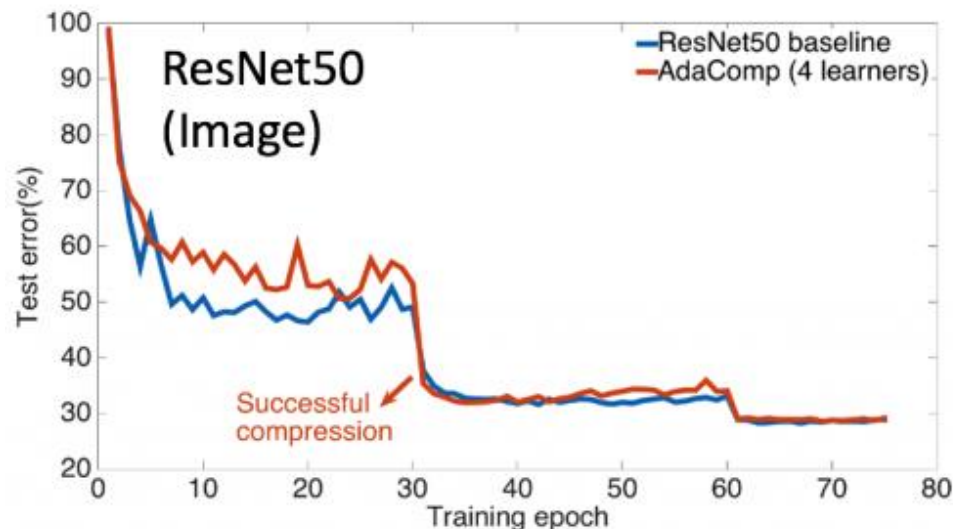


補充說明

• 訓練模型名稱說明

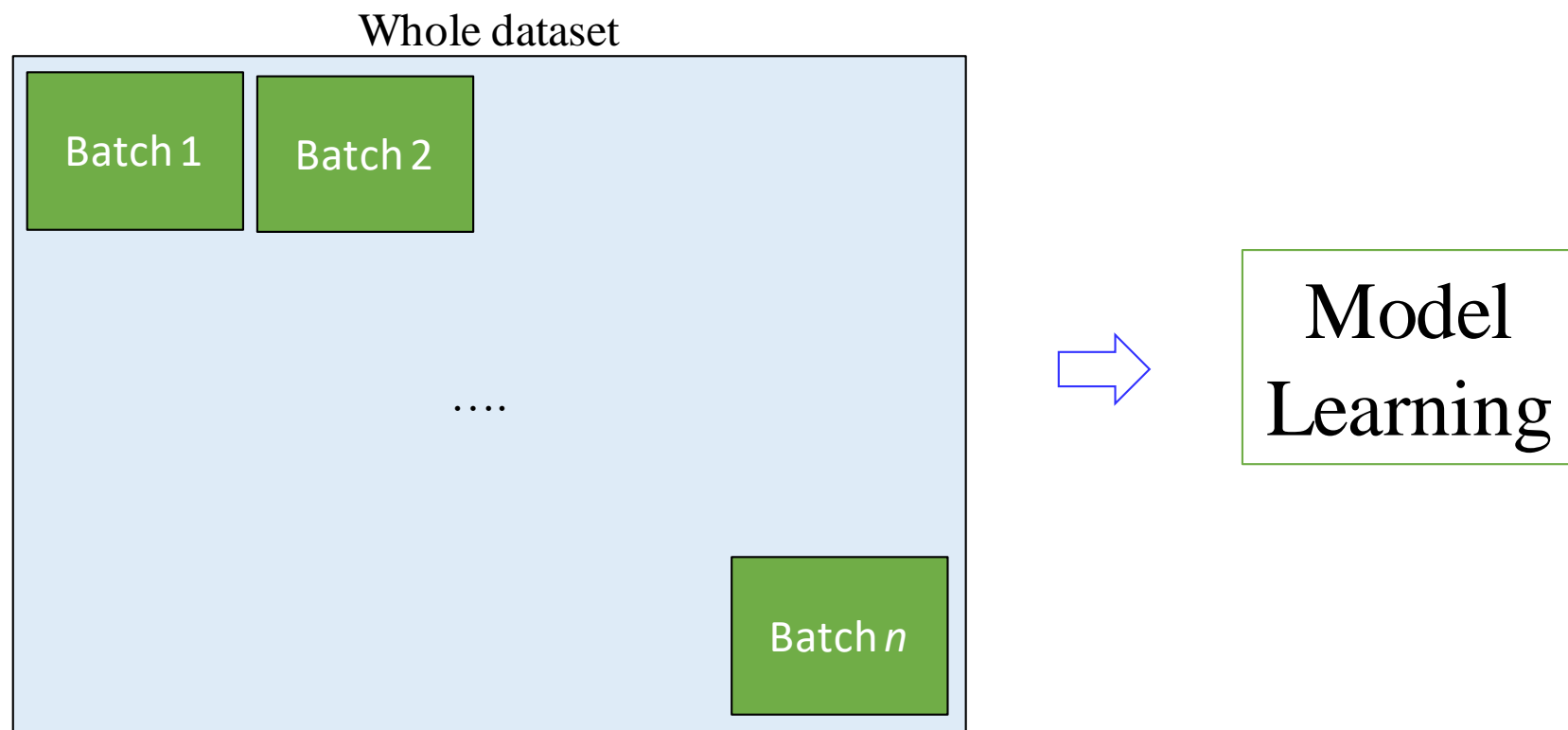
Learning iteration

Learning Epoch



補充說明

- 學習深度學習模型，因為倒傳遞需要特徵圖、權重和**Gradients**(當前和上一次的)等，因此需要大量記憶體，因此不可能一次全部的資料都到記憶體，所以採用**batch-learning**。



一次batch進去做完訓練模型稱為一個iteration。

所有的資料(Whole dataset)都學過一次稱為一個epoch。



Batch normalization and ReLU 對模型影響

- 上週提到Sigmoid和BN，測試看看當模型深層採用Sigmoid和BN的影響
- MNIST: 建立一個10層Conv層和一個FC層的神經網路

CONV層

- 1. Conv. + Sigmoid
- 2. Conv. + ReLU
- 3. Conv. + Sigmoid + BN
- 4. Conv. + ReLU + BN

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = Conv(1, 32, 3, 1)
        self.conv2 = nn.Sequential(
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1)
        )
        self.fc = nn.Linear(32, 10)
```



Batch normalization and ReLU 對模型影響

- 1. Conv. + Sigmoid
- 2. Conv. + ReLU

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = Conv(1, 32, 3, 1)
        self.conv2 = nn.Sequential(
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1)
        )
        self.fc = nn.Linear(32, 10)
```

	Sigmoid	ReLU
Epoch	acc	acc
1	10.09	11.35
2	9.82	11.35
3	11.35	11.35
4	11.35	11.35
5	11.35	11.35
6	11.35	11.35
7	11.35	11.35
8	11.35	11.35
9	11.35	11.35
10	11.35	11.35
11	11.35	11.35
12	11.35	11.35
13	11.35	11.35
14	11.35	11.35
15	11.35	11.35

- 沒有BN，不論用 Sigmoid或是ReLU都無法將模型訓練起來。



Batch normalization and ReLU 對模型影響

- 3. Conv. + Sigmoid + BN
- 4. Conv. + ReLU + BN

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = Conv(1, 32, 3, 1)
        self.conv2 = nn.Sequential(
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1),
            Conv(32, 32, 3, 1)
        )
        self.fc = nn.Linear(32, 10)
```

	BN+Sigmoid	BN+ReLU
Epoch	acc	acc
1	9.48	69.62
2	9.8	93.4
3	10.1	90.89
4	22.84	98.08
5	41.51	99.15
6	69.51	98.91
7	41.2	99.16
8	46.54	99.2
9	66.63	99.28
10	51.88	99.29
11	86.88	99.27
12	77.89	99.31
13	86.72	99.3
14	90.74	99.3
15	89.56	99.27

- 有BN
- ReLu比Sigmoid好。



Residual Block對模型影響

- 建立一個 4 2 層的卷積神經網路，每一個 Residual block 有兩個 Conv. ◦
- 用MNIST資料庫測試 ◦

```
class ResidualBlock(nn.Module):
    def __init__(self, in_channels,
out_channels, kernel_size=3, stride=1,
groups=1):
        padding = (kernel_size - 1) // 2
        super().__init__()
        self.conv1 = nn.Conv2d(in_channels,
out_channels, kernel_size, stride, padding,
groups=groups, bias=False)
        self.conv2 = nn.Conv2d(out_channels,
out_channels, kernel_size, stride, padding,
groups=groups, bias=False)
        self.bn1 = nn.BatchNorm2d(out_channels)
        self.bn2 = nn.BatchNorm2d(out_channels)
        self.relu = nn.ReLU()

    def forward(self, x):
        residual = x
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)

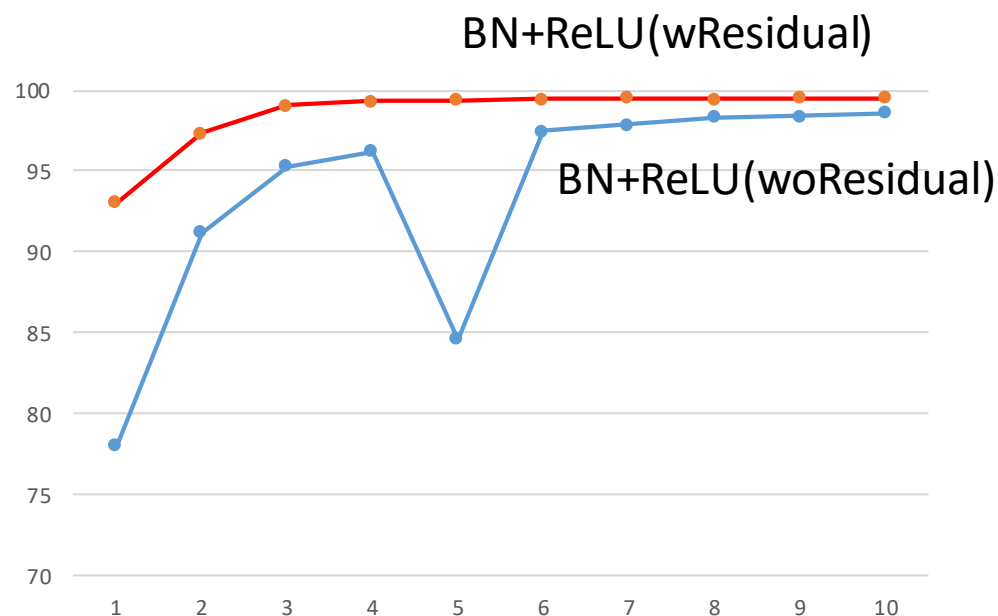
        x = self.conv2(x)
        x = self.bn2(x)

        x += residual
        x = self.relu(x)
        return x
```

```
self.conv1 = Conv(1, 32, 3, 1)
self.conv2 = nn.Sequential(
    ResidualBlock(32, 32, 3, 1), #1
    ResidualBlock(32, 32, 3, 1), #2
    ResidualBlock(32, 32, 3, 1), #3
    ResidualBlock(32, 32, 3, 1), #4
    ResidualBlock(32, 32, 3, 1), #5
    ResidualBlock(32, 32, 3, 1), #6
    ResidualBlock(32, 32, 3, 1), #7
    ResidualBlock(32, 32, 3, 1), #8
    ResidualBlock(32, 32, 3, 1), #9
    ResidualBlock(32, 32, 3, 1), #10
    ResidualBlock(32, 32, 3, 1), #11
    ResidualBlock(32, 32, 3, 1), #12
    ResidualBlock(32, 32, 3, 1), #13
    ResidualBlock(32, 32, 3, 1), #14
    ResidualBlock(32, 32, 3, 1), #15
    ResidualBlock(32, 32, 3, 1), #16
    ResidualBlock(32, 32, 3, 1), #17
    ResidualBlock(32, 32, 3, 1), #18
    ResidualBlock(32, 32, 3, 1), #19
    ResidualBlock(32, 32, 3, 1), #20
    ResidualBlock(32, 32, 3, 1), # 21
    ResidualBlock(32, 32, 3, 1), # 22
    ResidualBlock(32, 32, 3, 1), # 23
    ResidualBlock(32, 32, 3, 1), # 24
    ResidualBlock(32, 32, 3, 1), # 25
    ResidualBlock(32, 32, 3, 1), # 26
    ResidualBlock(32, 32, 3, 1), # 27
    ResidualBlock(32, 32, 3, 1), # 28
    ResidualBlock(32, 32, 3, 1), # 29
    ResidualBlock(32, 32, 3, 1), # 30
    ResidualBlock(32, 32, 3, 1), # 31
    ResidualBlock(32, 32, 3, 1), # 32
    ResidualBlock(32, 32, 3, 1), # 33
    ResidualBlock(32, 32, 3, 1), # 34
    ResidualBlock(32, 32, 3, 1), # 35
    ResidualBlock(32, 32, 3, 1), # 36
    ResidualBlock(32, 32, 3, 1), # 37
    ResidualBlock(32, 32, 3, 1), # 38
    ResidualBlock(32, 32, 3, 1), # 39
    ResidualBlock(32, 32, 3, 1), # 40
)
self.fc = nn.Linear(32, 10)
```



Residual Block對模型影響



```

self.conv1 = Conv(1, 32, 3, 1)
self.conv2 = nn.Sequential(
    ResidualBlock(32, 32, 3, 1), #1
    ResidualBlock(32, 32, 3, 1), #2
    ResidualBlock(32, 32, 3, 1), #3
    ResidualBlock(32, 32, 3, 1), #4
    ResidualBlock(32, 32, 3, 1), #5
    ResidualBlock(32, 32, 3, 1), #6
    ResidualBlock(32, 32, 3, 1), #7
    ResidualBlock(32, 32, 3, 1), #8
    ResidualBlock(32, 32, 3, 1), #9
    ResidualBlock(32, 32, 3, 1), #10
    ResidualBlock(32, 32, 3, 1), #11
    ResidualBlock(32, 32, 3, 1), #12
    ResidualBlock(32, 32, 3, 1), #13
    ResidualBlock(32, 32, 3, 1), #14
    ResidualBlock(32, 32, 3, 1), #15
    ResidualBlock(32, 32, 3, 1), #16
    ResidualBlock(32, 32, 3, 1), #17
    ResidualBlock(32, 32, 3, 1), #18
    ResidualBlock(32, 32, 3, 1), #19
    ResidualBlock(32, 32, 3, 1), #20
    ResidualBlock(32, 32, 3, 1), # 21
    ResidualBlock(32, 32, 3, 1), # 22
    ResidualBlock(32, 32, 3, 1), # 23
    ResidualBlock(32, 32, 3, 1), # 24
    ResidualBlock(32, 32, 3, 1), # 25
    ResidualBlock(32, 32, 3, 1), # 26
    ResidualBlock(32, 32, 3, 1), # 27
    ResidualBlock(32, 32, 3, 1), # 28
    ResidualBlock(32, 32, 3, 1), # 29
    ResidualBlock(32, 32, 3, 1), # 30
    ResidualBlock(32, 32, 3, 1), # 31
    ResidualBlock(32, 32, 3, 1), # 32
    ResidualBlock(32, 32, 3, 1), # 33
    ResidualBlock(32, 32, 3, 1), # 34
    ResidualBlock(32, 32, 3, 1), # 35
    ResidualBlock(32, 32, 3, 1), # 36
    ResidualBlock(32, 32, 3, 1), # 37
    ResidualBlock(32, 32, 3, 1), # 38
    ResidualBlock(32, 32, 3, 1), # 39
    ResidualBlock(32, 32, 3, 1), # 40
)
self.fc = nn.Linear(32, 10)

```

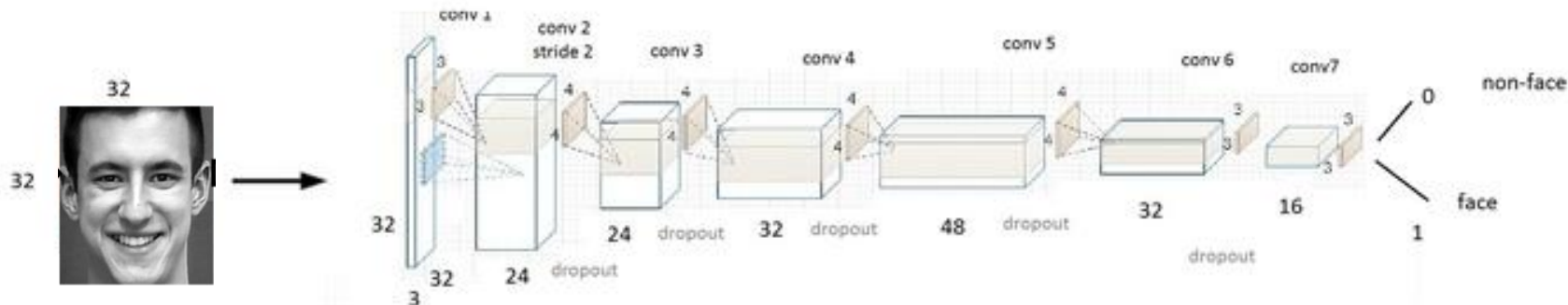
Outline

- 補充說明
 1. Iteration or Epoch
 2. MNIST實例: Batch normalization and ReLU 對模型影響
 3. MNIST實例: Residual Block對模型影響

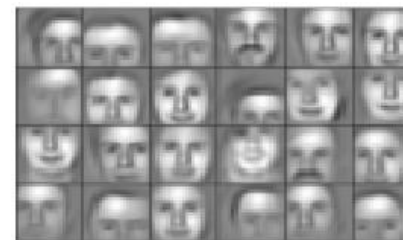
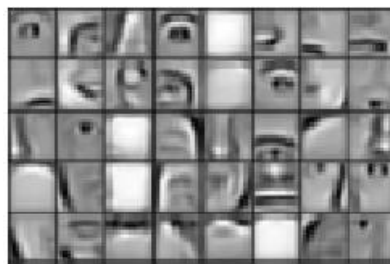
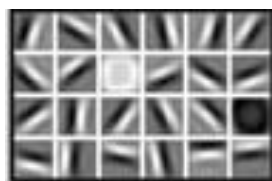
- CNN取什麼特徵?



以CNN(Convolutional Neural Network)做人臉影像識別



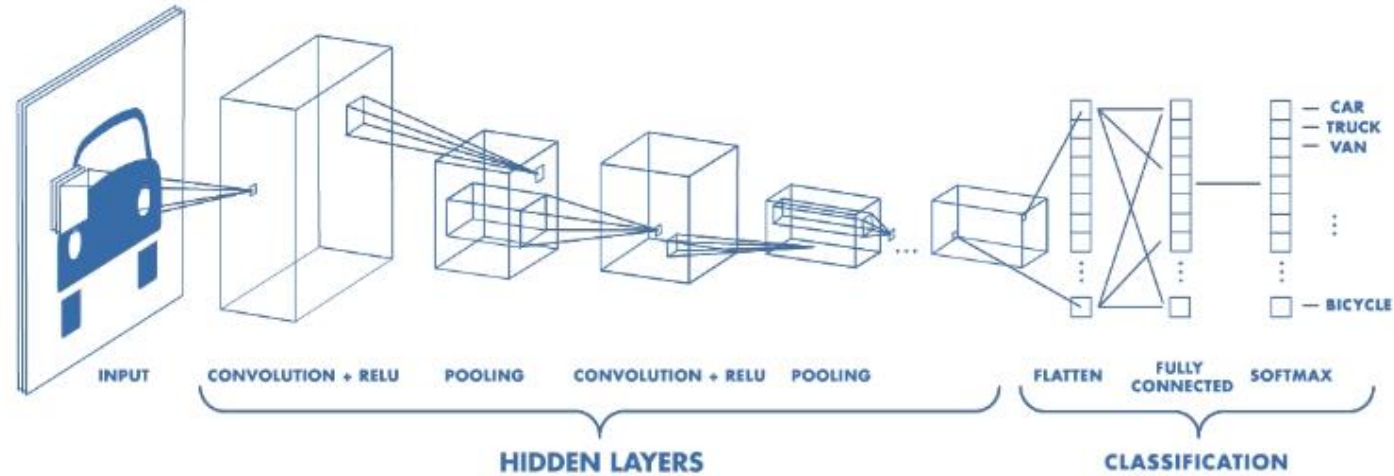
Source: Face detection based on deep convolutional neural networks exploiting incremental facial part learning (Dec 2016)



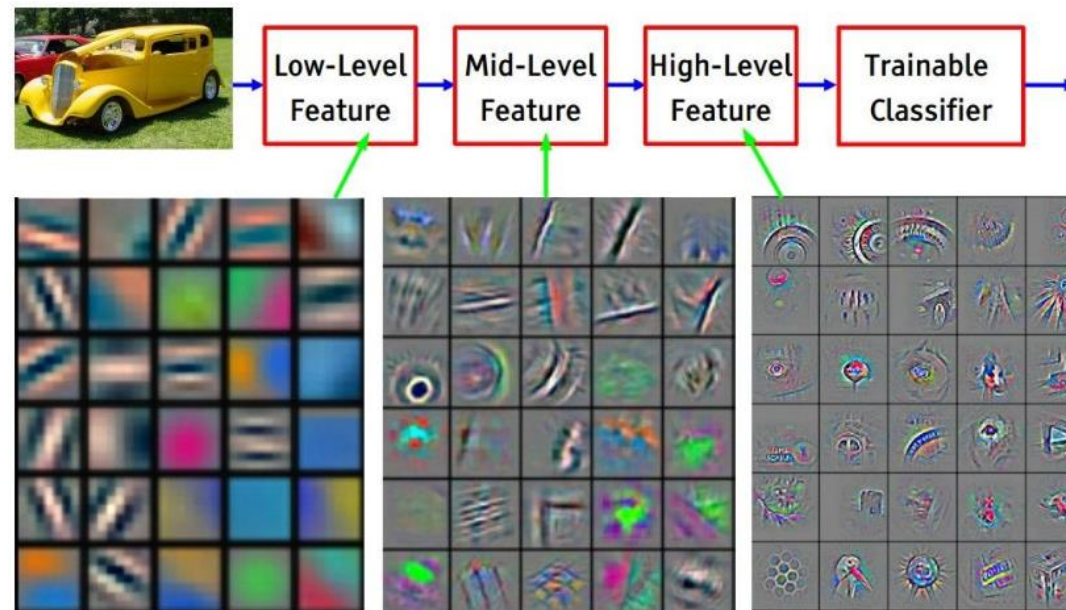
Source: Feature Visualization of Convnet trained on ImageNet from [Zeiler & Fergus 2013]



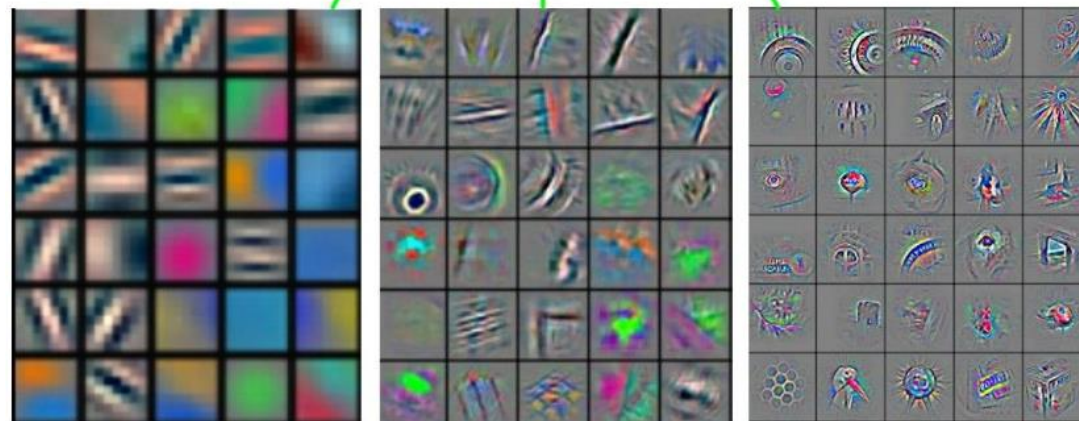
CNN (Convolutional Neural Network) Vehicle Classification



Source: <https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks-1489512765771.html>



CNN取什麼特徵?



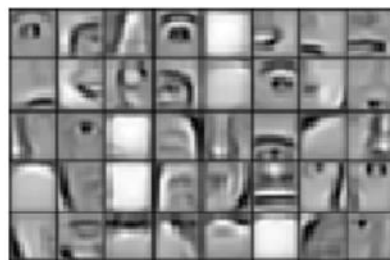
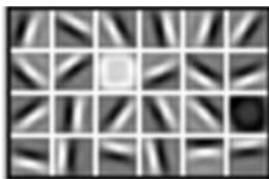
CNN取什麼特徵?

MNIST 手寫數字資料庫

訓練一個五層的CNN

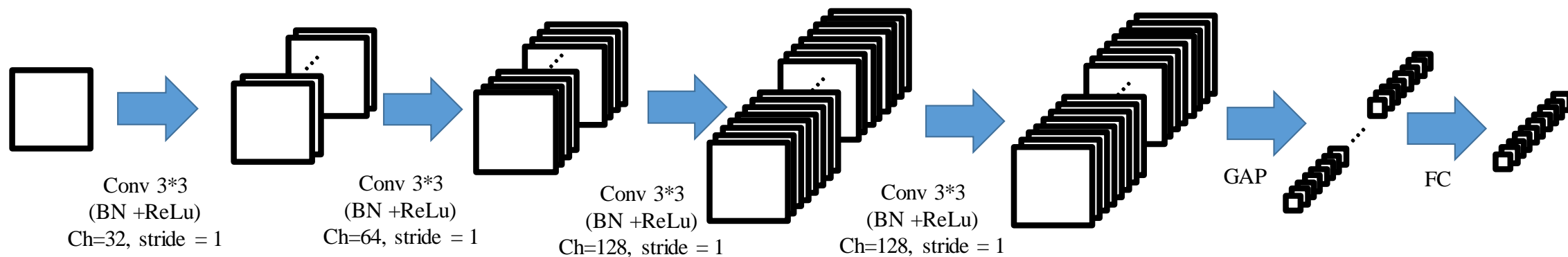
我們觀察一下這個模型的特徵圖。

```
self.conv1 =ConvBNReLU(1, 32, 3, 1)
self.conv2 =ConvBNReLU(32, 64, 3, 1)
self.conv3 =ConvBNReLU(64, 128, 3, 1)
self.conv4 =ConvBNReLU(128, 128, 3, 1)
self.fc = nn.Linear(128, 10)
```

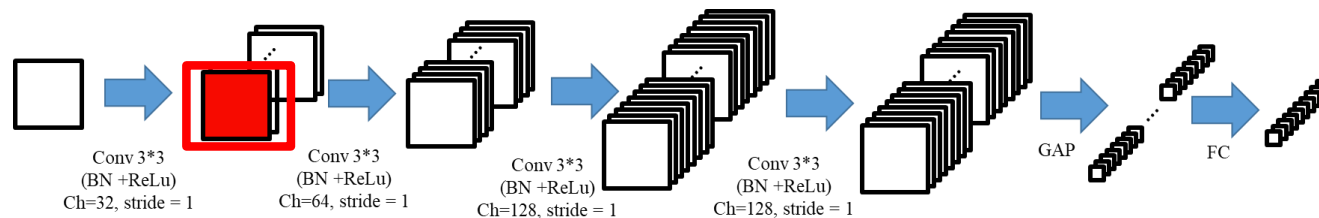


CNN取什麼特徵?

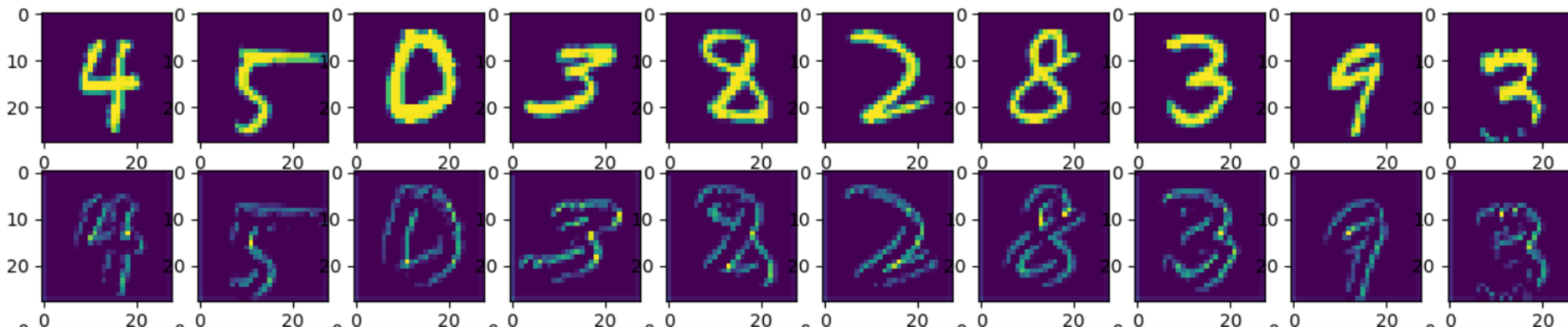
```
self.conv1 = ConvBNReLU(1, 32, 3, 1)
self.conv2 = ConvBNReLU(32, 64, 3, 1)
self.conv3 = ConvBNReLU(64, 128, 3, 1)
self.conv4 = ConvBNReLU(128, 128, 3, 1)
self.fc = nn.Linear(128, 10)
```



CNN取什麼特徵?

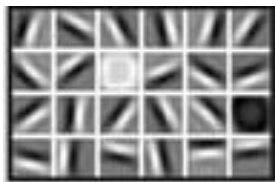


輸入圖



徵圖都只取第一張
第一層卷積
輸出特徵圖

Low-Level
Feature



Low-level Feature只取邊的特徵
就如同Canny Edge Detection

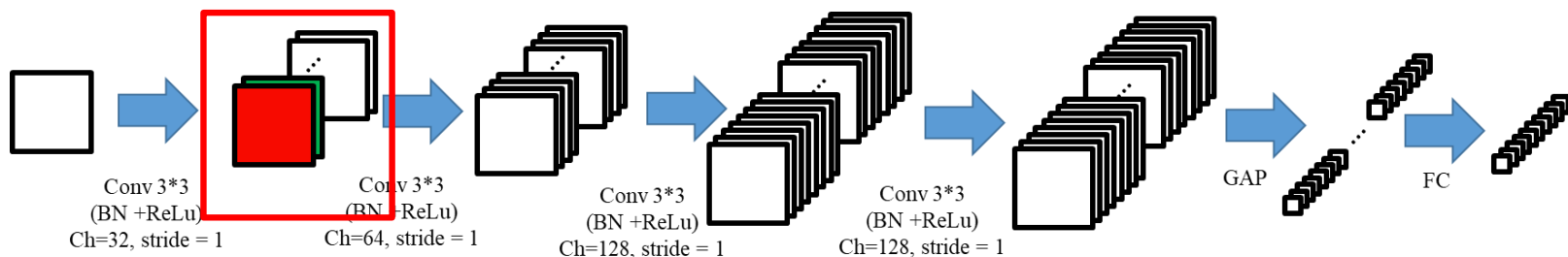
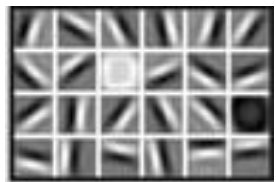
一兩層簡單的卷積就只保留圖像的邊

在卷積神經網路的Low-level Feature
則是讓“邊的部分被強化”，所以可以
發現上圖的邊顏色變強了。



第一層卷積輸出特徵圖，取不同channel特徵圖

Low-Level Feature



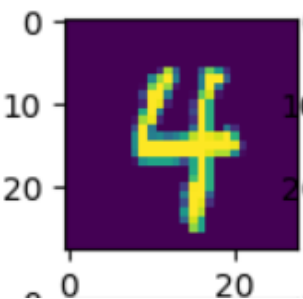
Channel 1

Channel 2

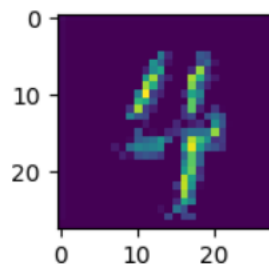
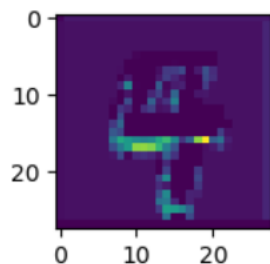
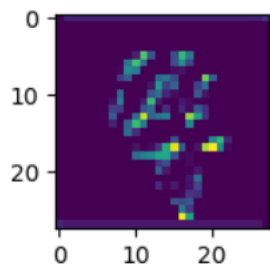
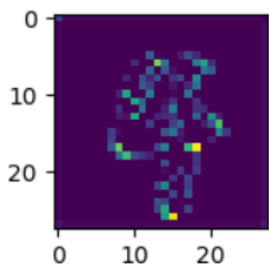
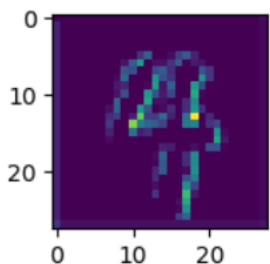
Channel 3

Channel 4

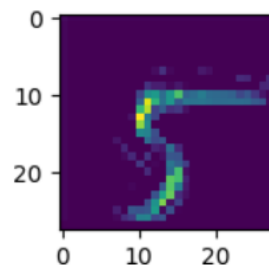
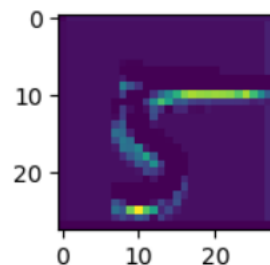
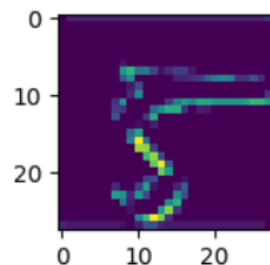
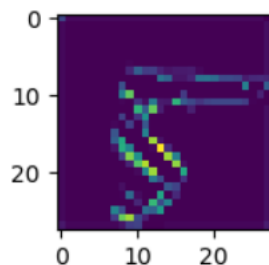
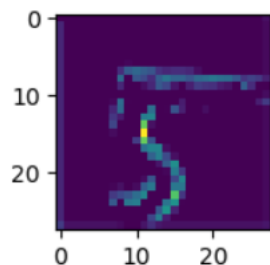
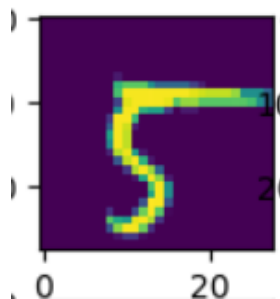
Channel 5



Conv1

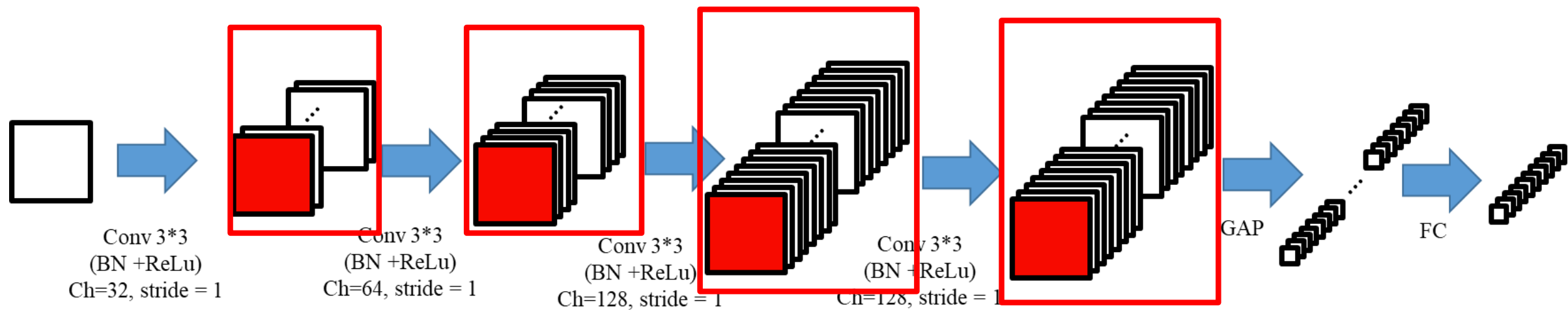


Low-Level Feature



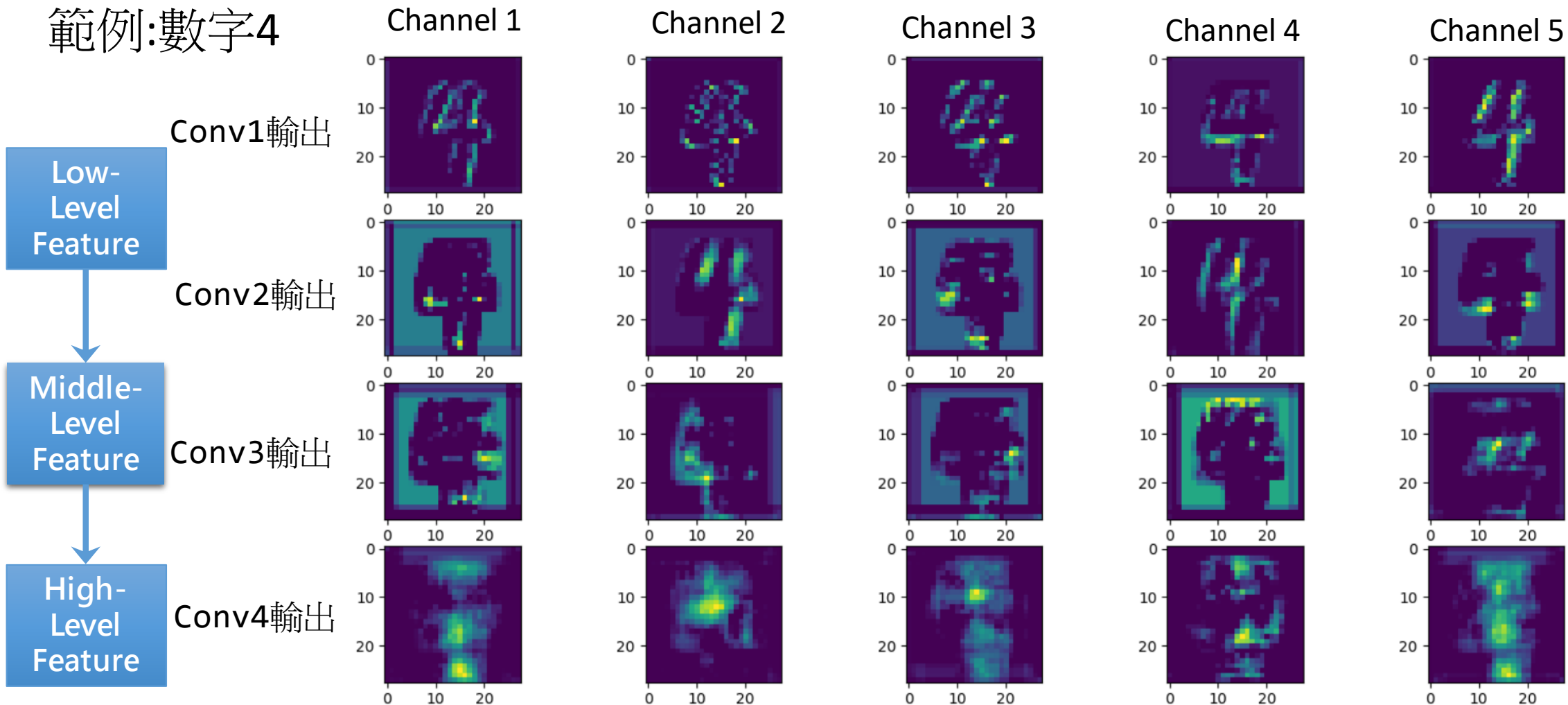
CNN取什麼特徵?

- 每一層卷積輸出特徵圖，取不同channel特徵圖



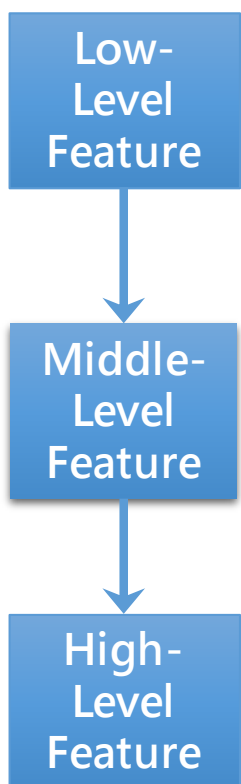
每一層卷積輸出特徵圖 取不同channel特徵圖

範例:數字4



每一層卷積輸出特徵圖 取不同channel特徵圖

範例:數字5

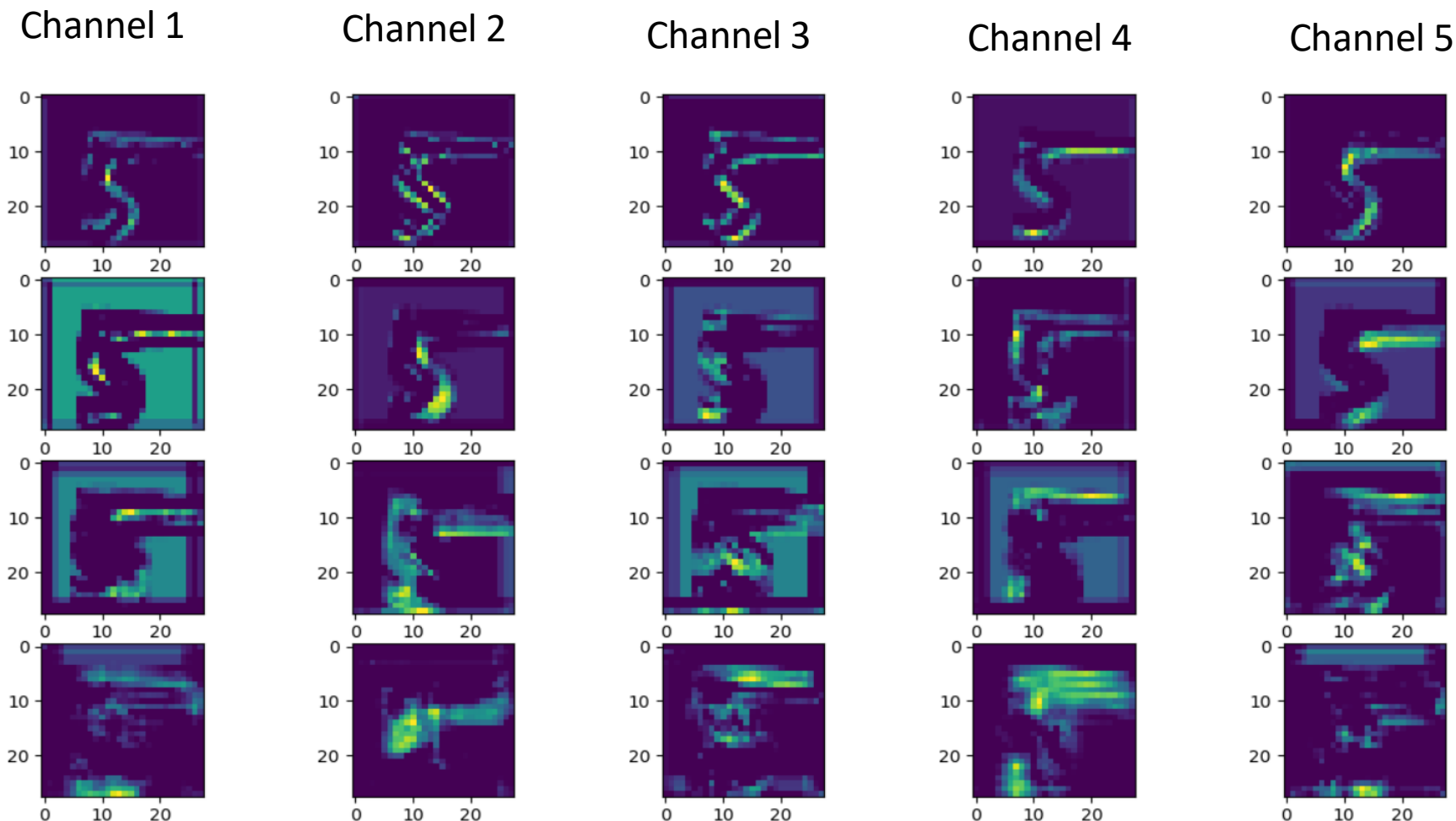


Conv1輸出

Conv2輸出

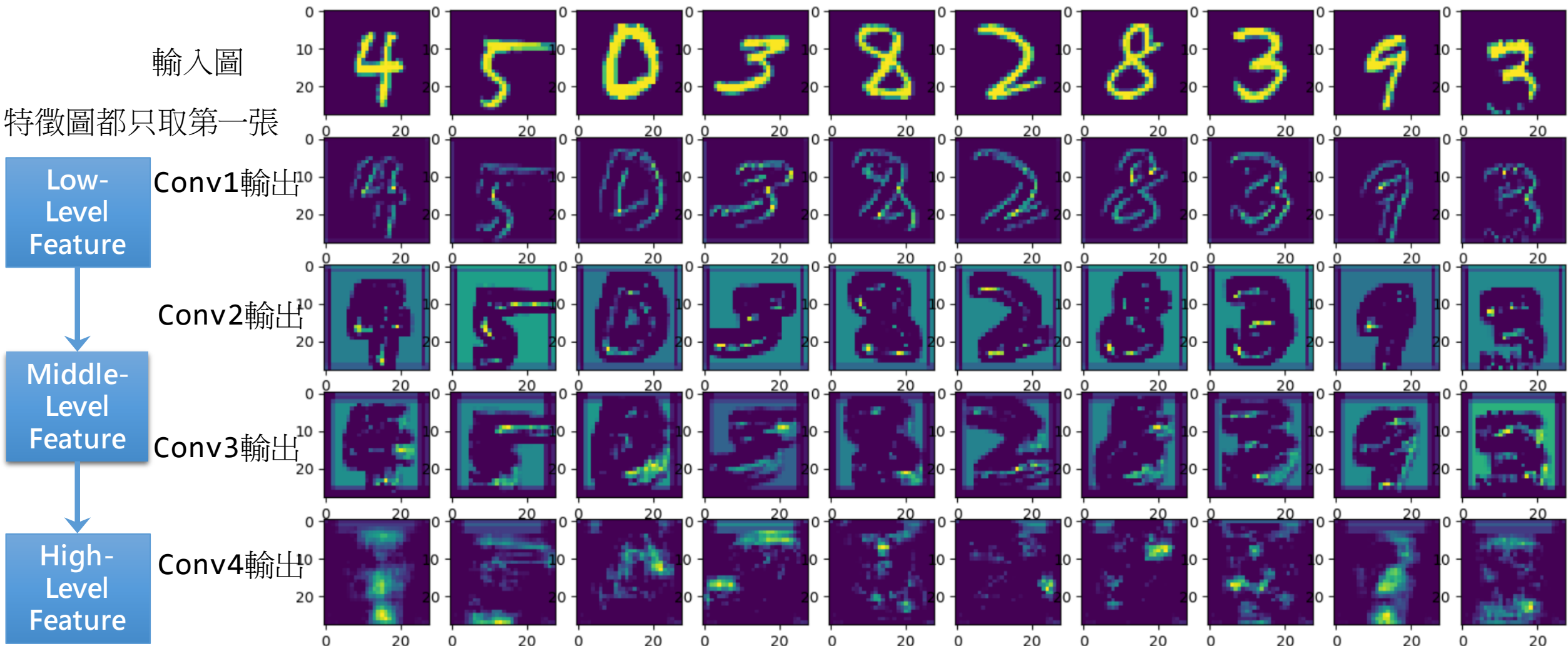
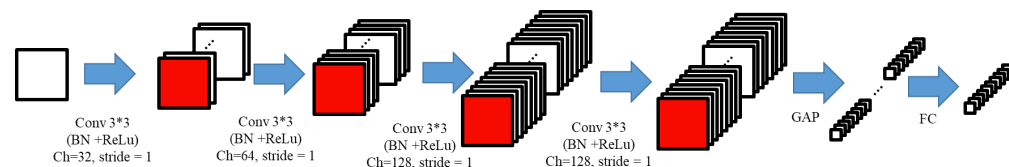
Conv3輸出

Conv4輸出



CNN不同層取什麼特徵?

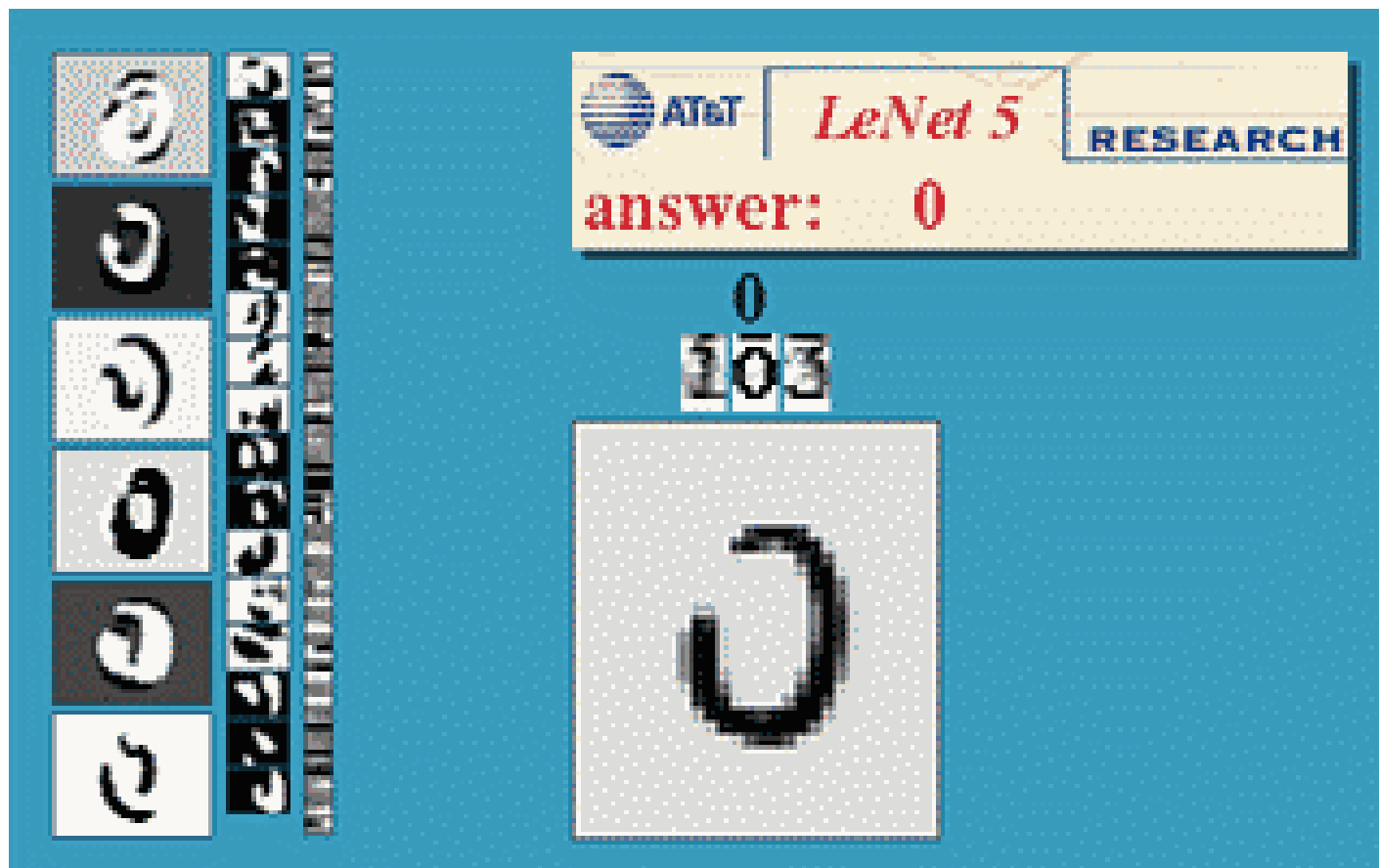
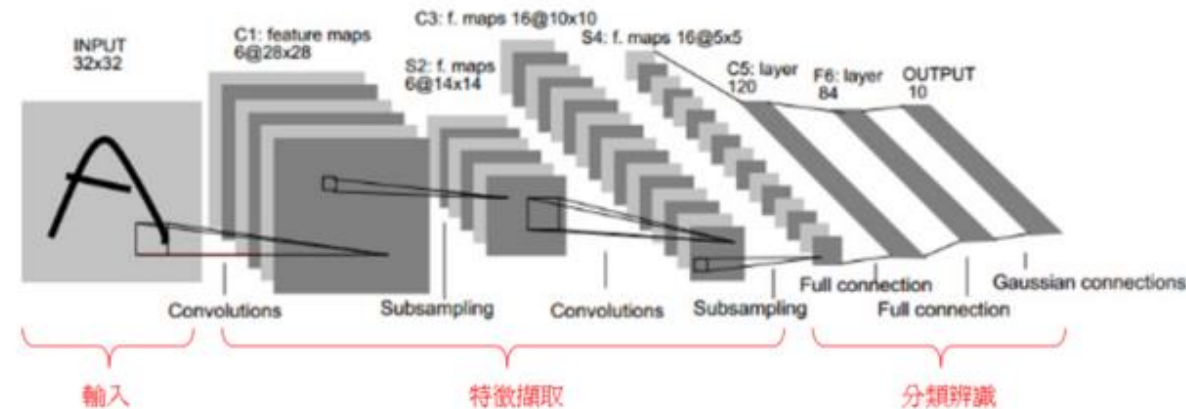
範例: 不同數字在不同層的第一個特徵圖。



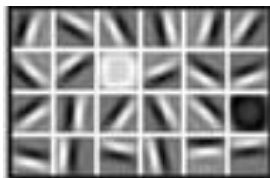
Conv4的第一個特徵圖，看的是垂直線的特徵，所以只有4和9會有反應。

CNN不同層取什麼特徵?

- LeNet5 from Yann LeCun

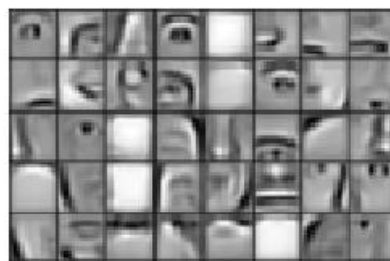


CNN取什麼特徵?



Low-level Feature

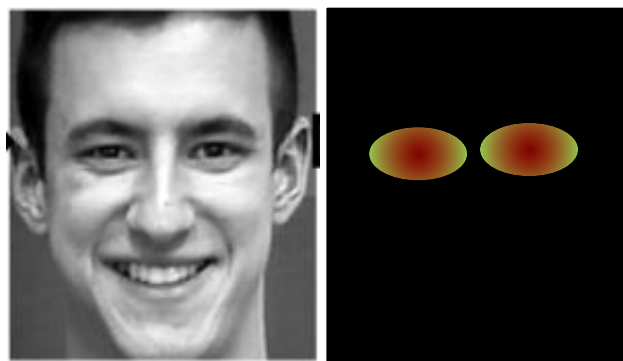
看到的是邊的部分會被強化，邊在特徵圖上的輸出值較大。



Middle-Level Feature

看到的是圖上器官的在feature map的輸出值較大。

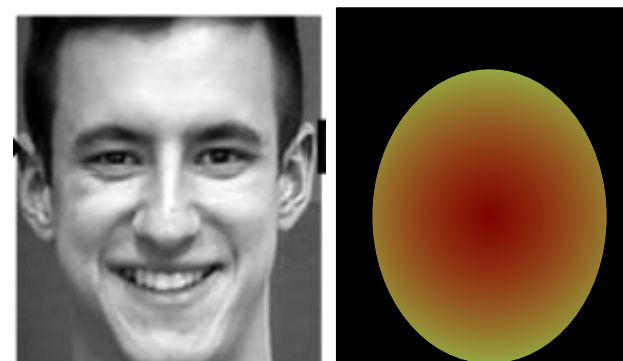
EX: 特徵圖(表示眼睛的反應)



High-Level Feature

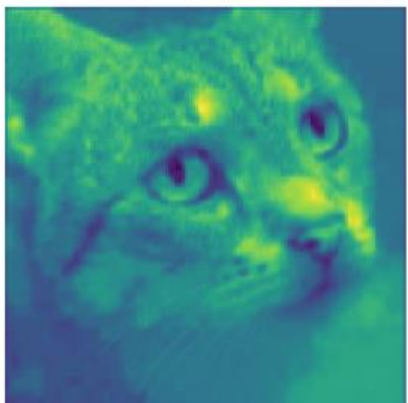
看到的是圖上臉的在feature map的輸出值較大。

EX: 特徵圖(表示臉的反應)

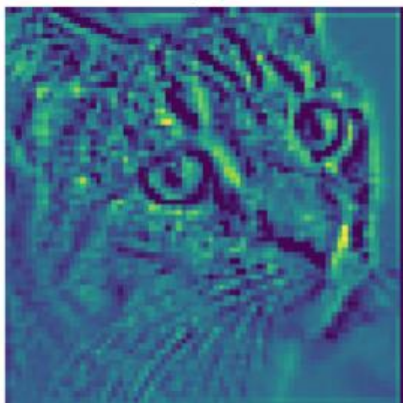


CNN取什麼特徵?

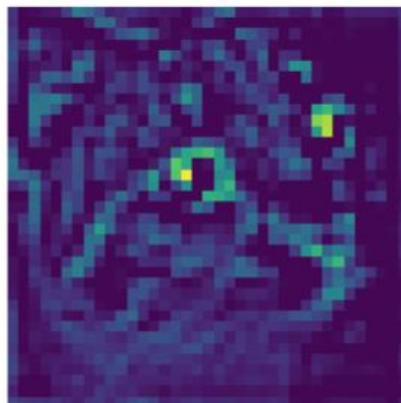
block1_conv1



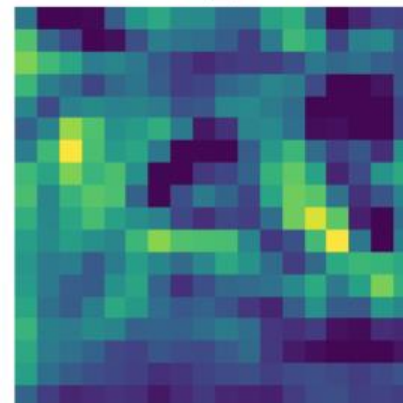
block2_conv1



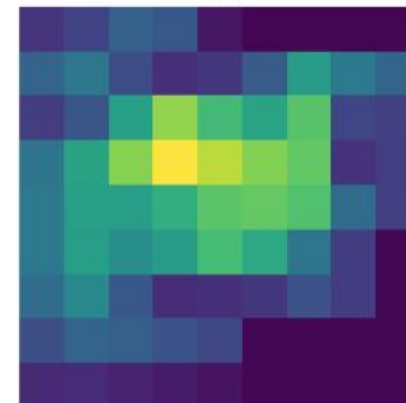
block3_conv1



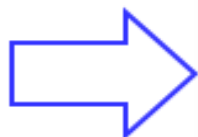
block4_conv1



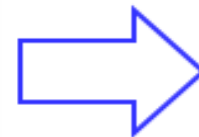
block5_conv1



Low-Level
Feature



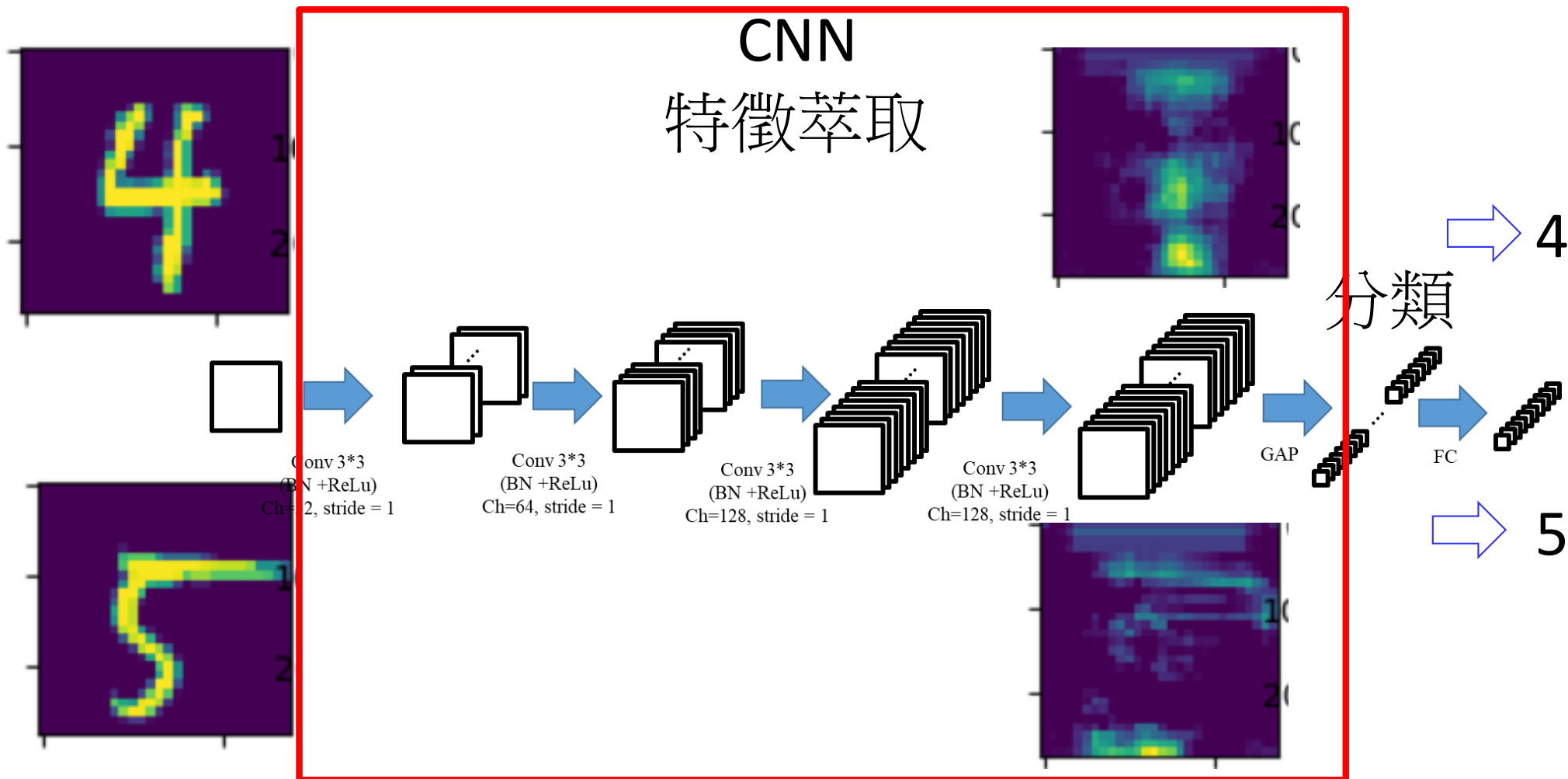
Middle-Level
Feature



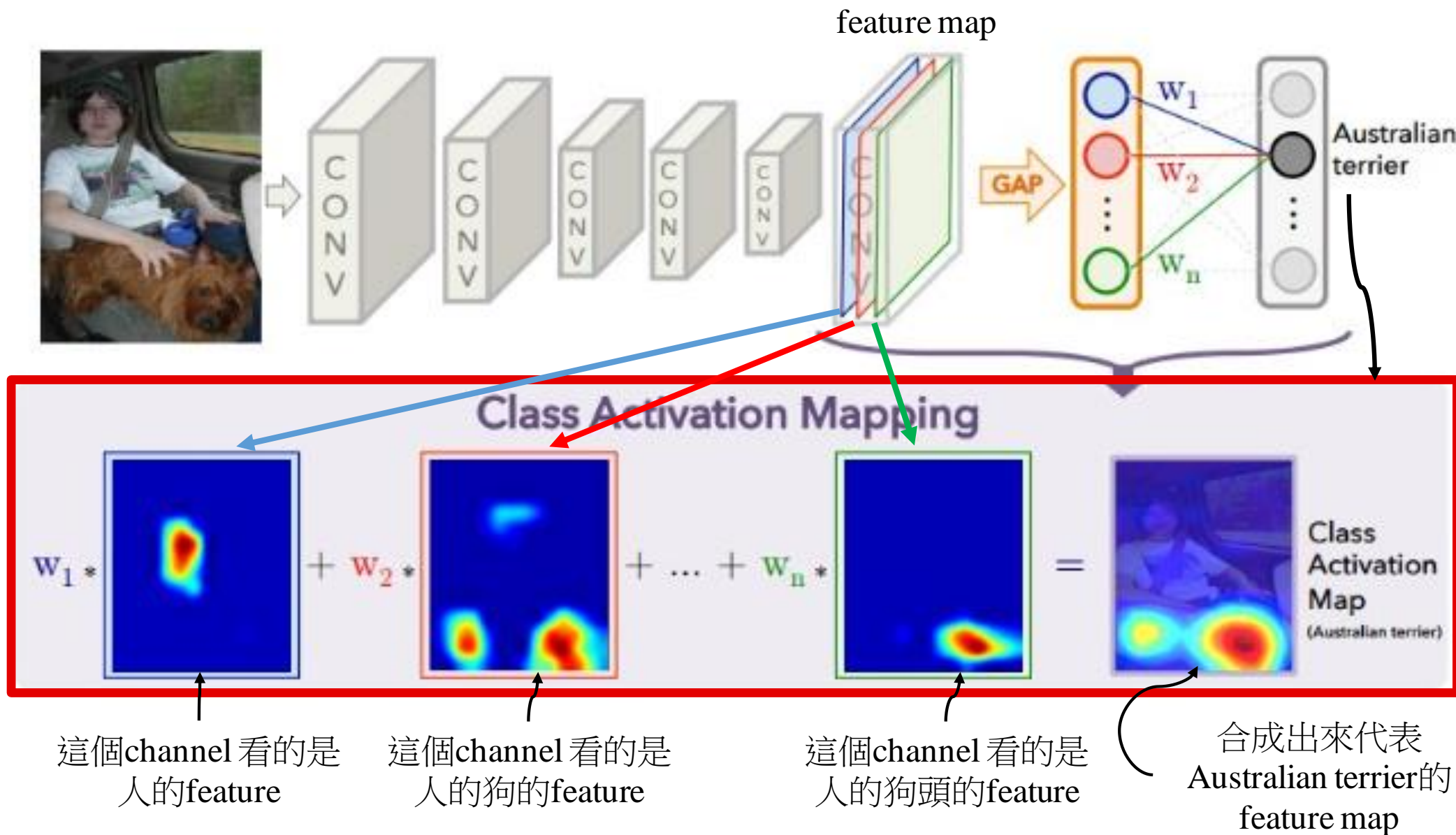
High-Level
Feature



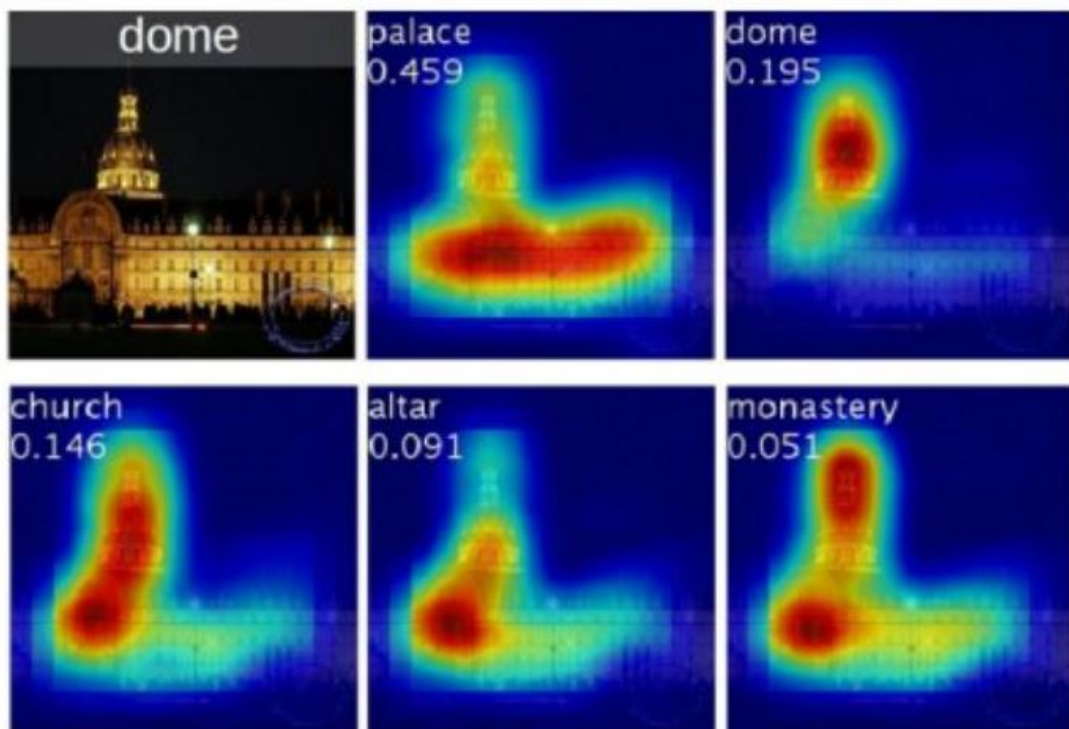
CNN取什麼特徵?



觀察模型針對這張圖在特定類別看得區塊



觀察模型針對這張圖在特定類別看得區塊



Class activation maps of top 5 predictions



Class activation maps for one object class

