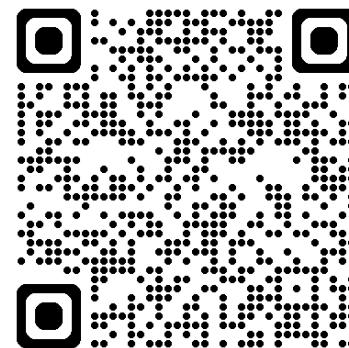




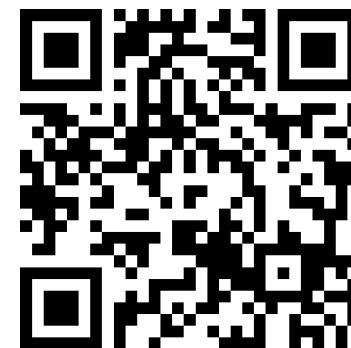
深度學習 Deep Learning

Convolutional Neural Networks (CNN)

Instructor: 林英嘉 (Ying-Jia Lin)
2025/10/08



[Course GitHub](#)



[Slido # DL1008](#)

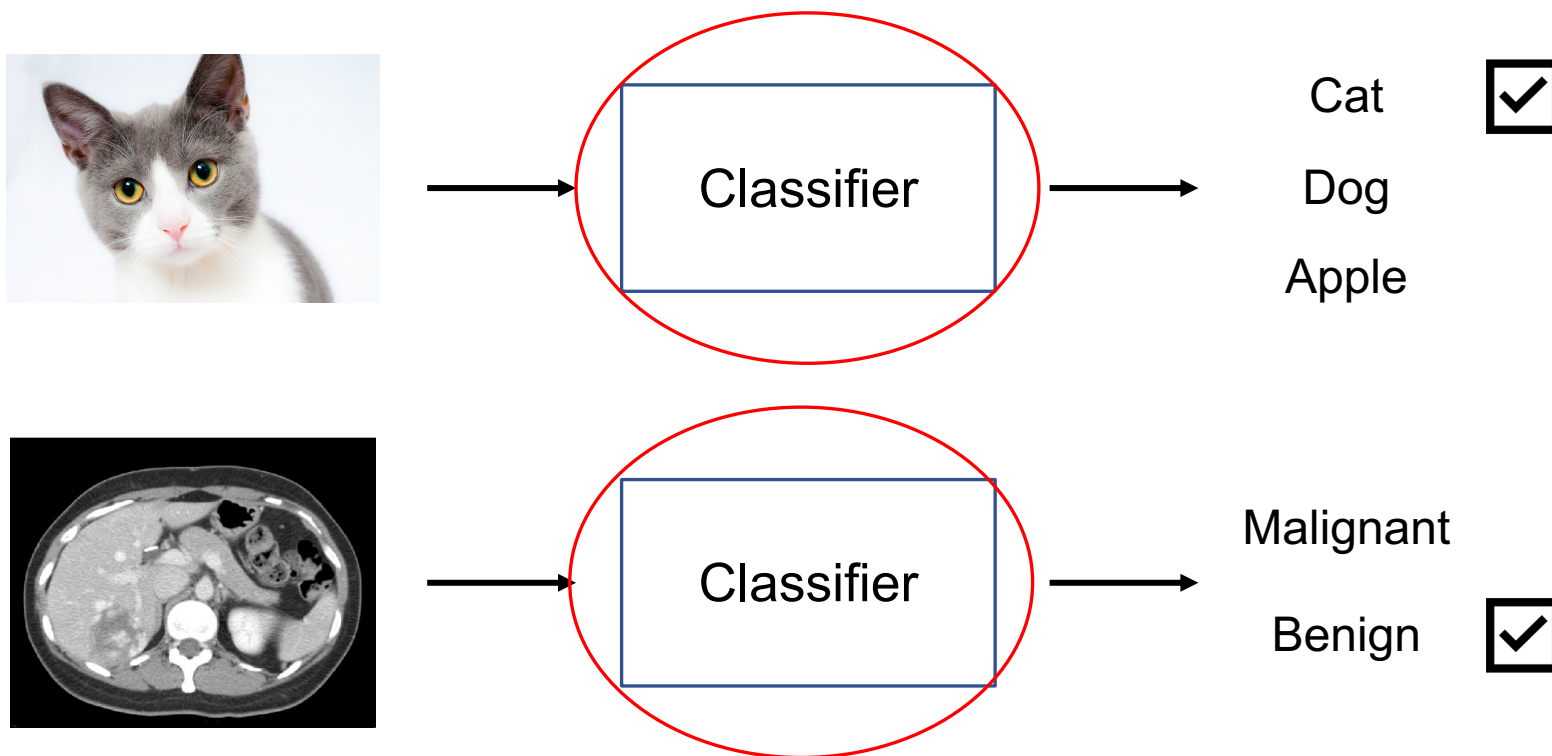
Outline

- Images [10 min]
- Convolutional Neural Networks [50 min]
- CV (Computer Vision) in PyTorch [40 min]
- HW2 [30 min]
- Quiz [20 min]



什麼是機器學習？(以電腦視覺為例)

Machine Learning function!



LeNet-5: Early Convolutional Neural Networks

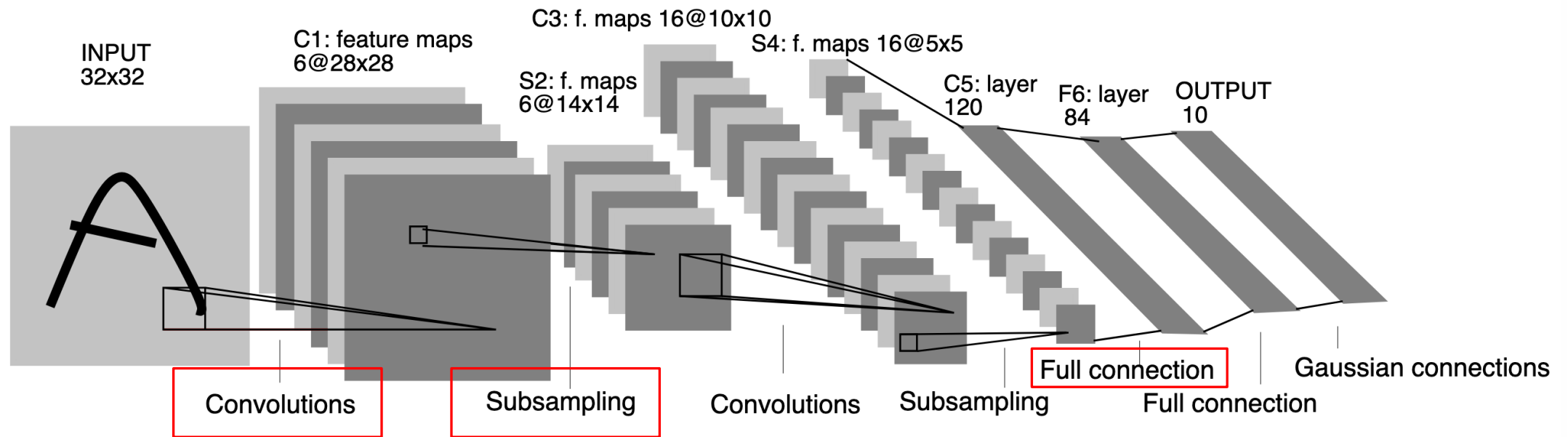


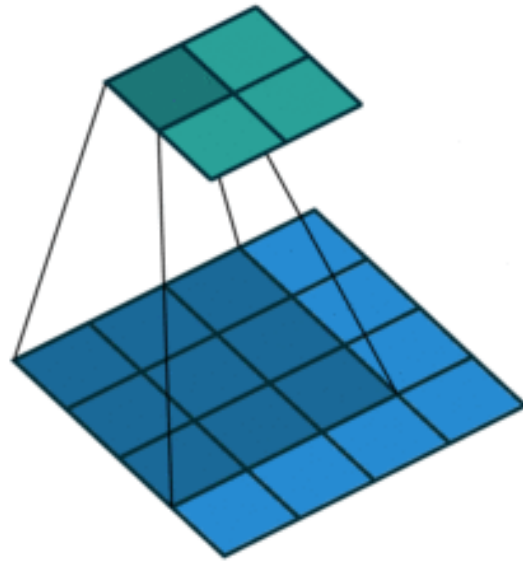
Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278-2324.



Examples of Convolutions

Padding = 0, stride = 1



Padding = 0, stride = 2

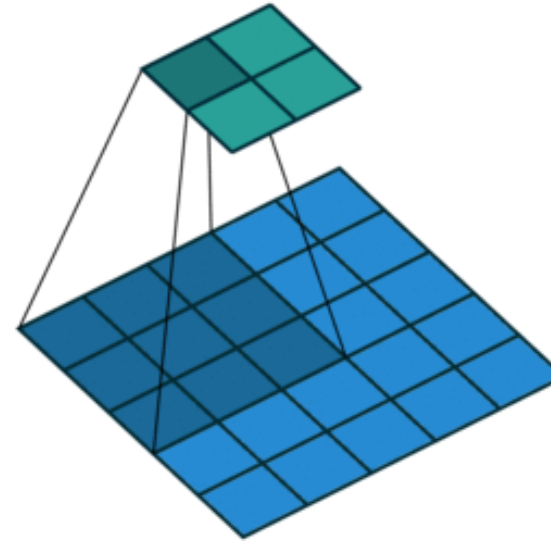
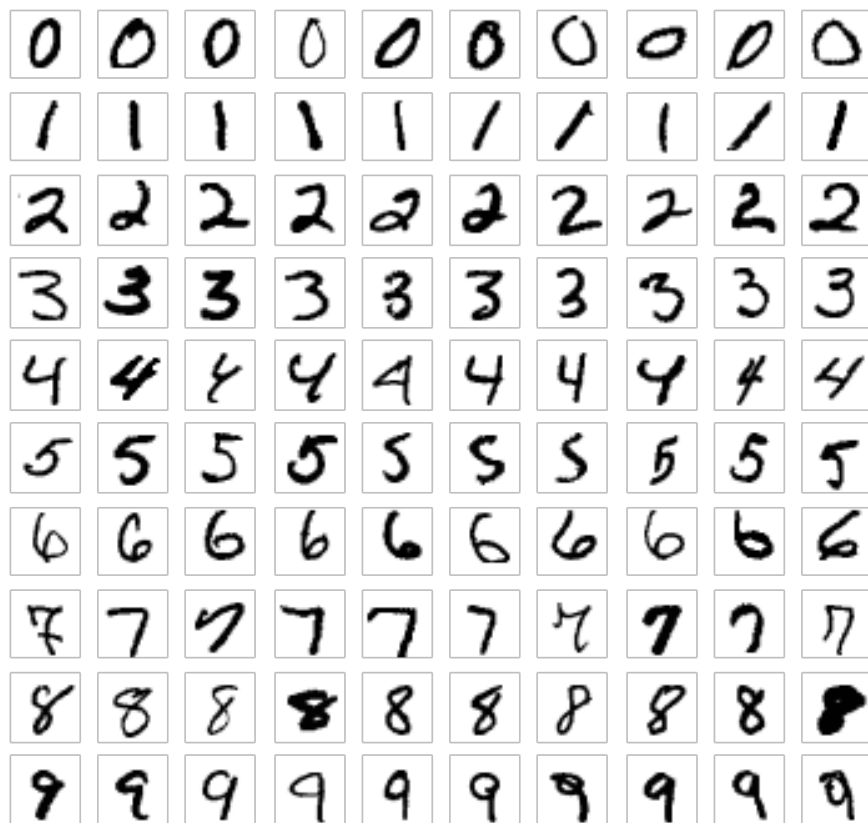


Figure source:
<https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html>



Images

MNIST



<http://yann.lecun.com/exdb/mnist/>

CIFAR-10

airplane

automobile

bird

cat

deer

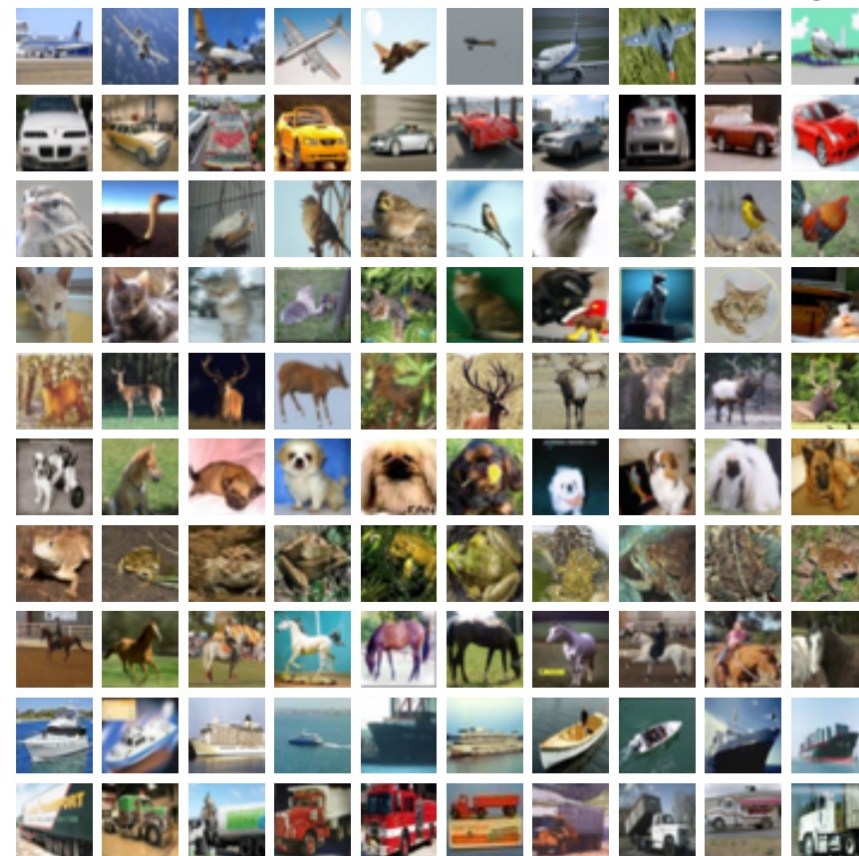
dog

frog

horse

ship

truck

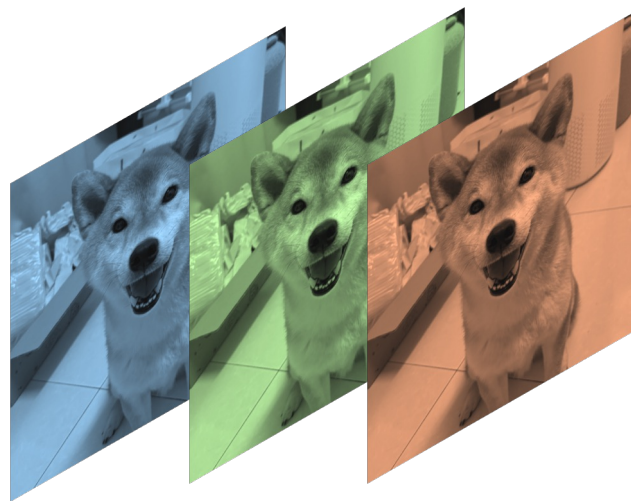


<https://www.cs.toronto.edu/~kriz/cifar.html>



RGB Images (彩色影像)

- RGB 代表紅色 (Red)、綠色 (Green) 及藍色 (Blue)



Blue
Channel

Green
Channel

Red
Channel

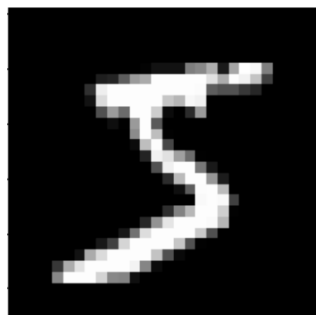
Example shape:
(3, 128, 128)

channels height width



Gray-scale Images (灰階影像)

- 灰階影像只有一個 channel



Example shape:

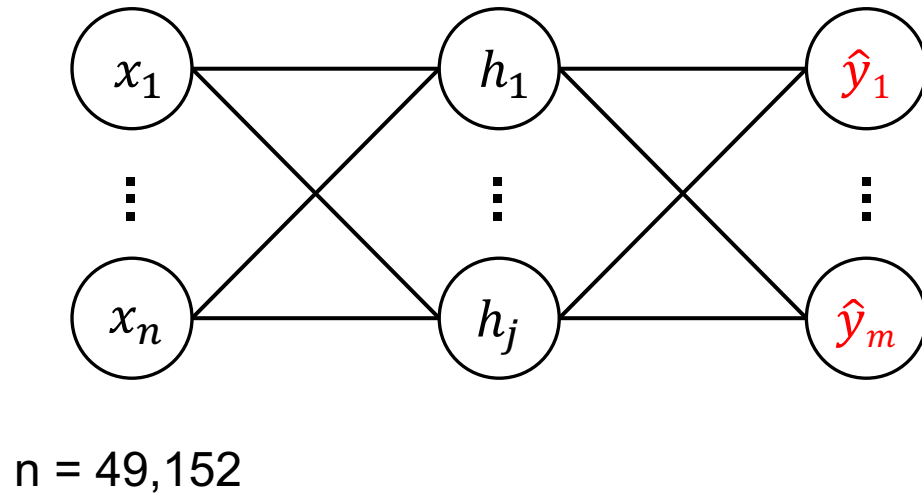
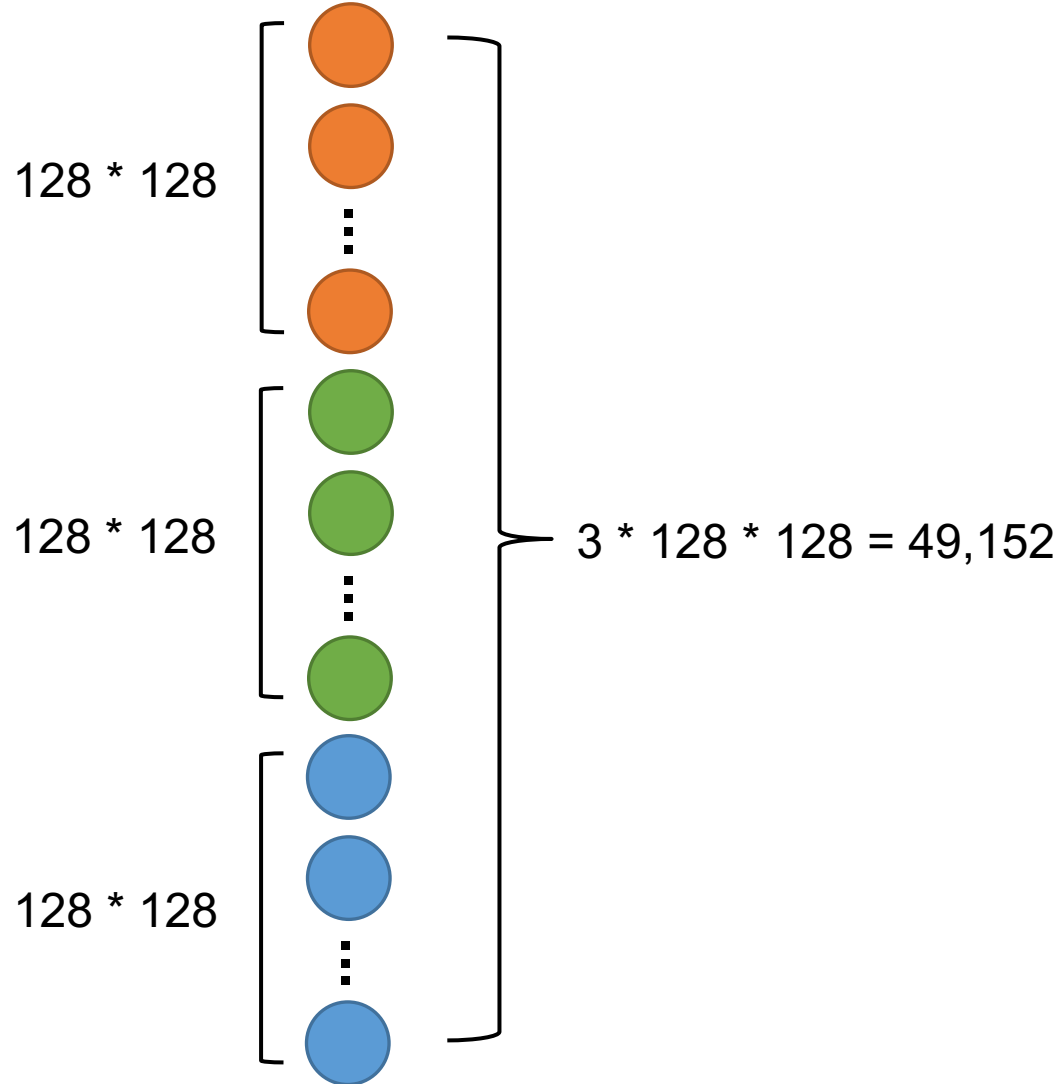
(1, 128, 128)

channels height width

↑ ↑ ↑



Image inputs are too big for an MLP



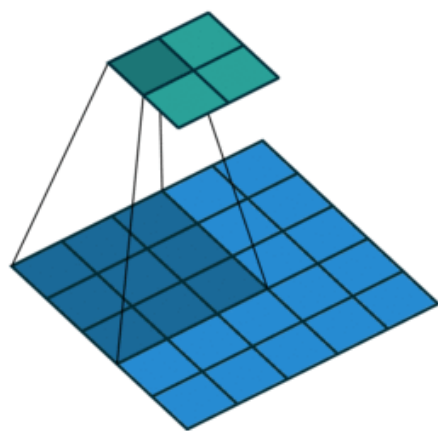
Convolutional Neural Networks

CNN

卷積層有兩個重要的設定參數

<https://docs.pytorch.org/docs/stable/generated/torch.nn.Conv2d.html>

- **stride (int):** 步長，代表卷積一次走幾個像素點
- **padding (int):** 影像的四個邊要不要補 0



Padding = 0, stride = 2

0	0	0	0	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

(白色代表原始影像區域，橘紅色代表padding)

Figure source:
<https://hannibunny.github.io/mlbook/neuralnetworks/convolutionDemos.html>



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1
---	---



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1	1
---	---	---



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1	1	2
---	---	---	---



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1	1	2
-1			



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1	1	2
-1	1		



Convolutions (stride = 1)

沒有 padding

Stride = 1

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1	1	2
-1	1	1	0
0	0	1	2
0	0	1	2

feature map



Convolutions (stride = 1)

沒有 padding

Stride = 2

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1



Convolutions (stride = 2)

沒有 padding

→

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1


Filter

1	1
---	---



Convolutions (stride = 2)

Stride = 2



1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

1	0	0
0	1	0
0	0	-1

Filter

1	1
0	



Convolutions (stride = 2)

沒有 padding

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0

Stride = 2

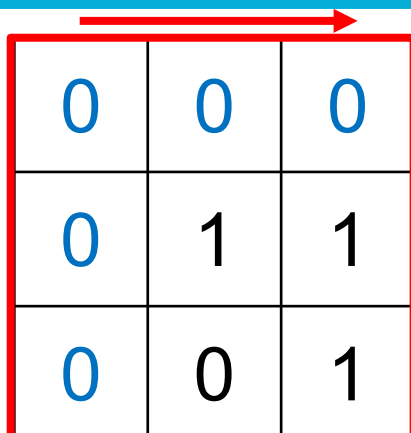
1	0	0
0	1	0
0	0	-1

Filter

1	1
0	1



Convolutions (stride = 2, padding = 1)



0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

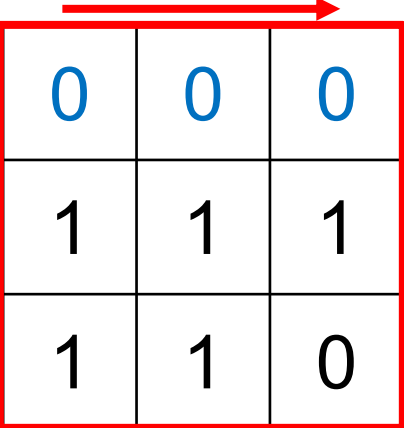
有 padding

0		
---	--	--

Padding



Convolutions (stride = 2, padding = 1)



0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

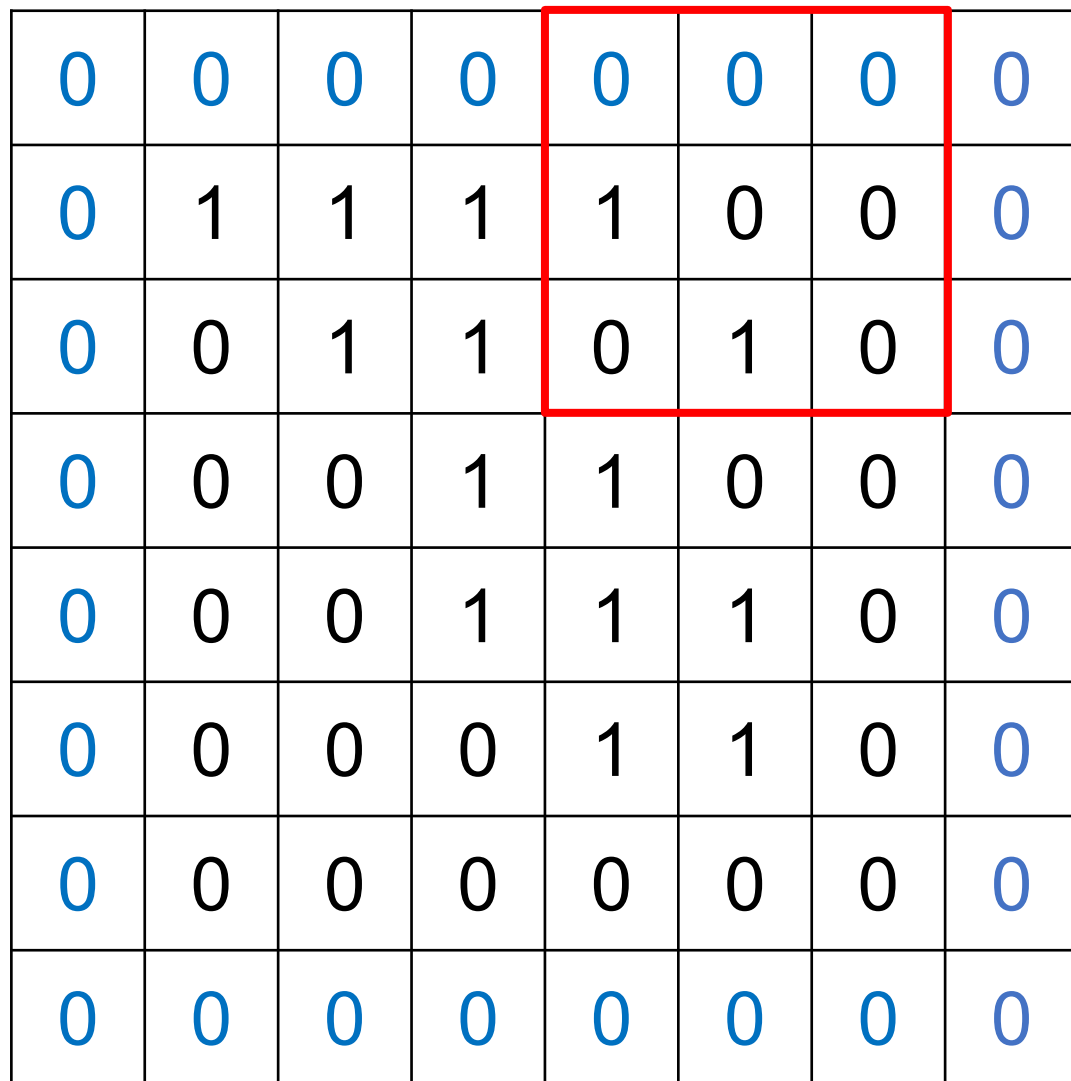
有 padding

0	1	
---	---	--

Padding



Convolutions (stride = 2, padding = 1)



The diagram illustrates a 1D convolution operation. The input is an 8x8 grid of values, where the first four columns are blue (padding) and the last four are black (input). A 3x3 red box highlights the current receptive field. A red arrow above the grid points to the right, labeled 'Stride = 2'. To the right of the grid is a 3x3 green 'Filter' matrix. Further right is a 1x3 white output vector, where the last element is red. A blue box labeled '有 padding' is in the top right. A small neural network icon is in the bottom left.

0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1


Filter

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

有 padding

Padding

Convolutions (stride = 2, padding = 1)



0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

有 padding

0	1	0
0		

Padding



Convolutions (stride = 2, padding = 1)

0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

0	1	0
0	1	

Padding



Convolutions (stride = 2)

padding=1

0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter


0	1	0
0	1	0

Padding



Convolutions (stride = 2)

padding=1



0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

0	1	0
0	1	0
0		

Padding

Convolutions (stride = 2)

padding=1

0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

0	1	0
0	1	0
0	0	

Padding



Convolutions (stride = 2)

padding=1

0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	0
0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Stride = 2

1	0	0
0	1	0
0	0	-1

Filter

0	1	0
0	1	0
0	0	2

Padding



Filters 數目與 output_channels 的關係

- Filters 的數目可以調整
- 對輸入層來說，1 個 filter 照顧 3 個 channels 的資料，輸出 1 個 feature map
- 使用 64 個 filters，就會輸出 64 個 feature maps
- 可調整參數為: $\text{num_filters} * \text{input_channels} * \text{filter_size} * \text{filter_size}$



Convolutions (RGB)

filters 數值可能都是不同的！(為了方便計算，簡報使用一樣的數字)

R	G	B	1	1	1
			0	1	1
			0	0	1
			0	0	1
			0	0	1
			0	0	0
			0	0	0

1	0	0
0	1	0
0	0	-1

(R) Filter

1	0	0
0	1	0
0	0	-1

(G) Filter

1	0	0
0	1	0
0	0	-1

(B) Filter

1	1	1
0	1	1
0	0	1

1	0	1
0	1	1
0	0	1

1	0	0
0	1	1
1	0	1

先各自與Filter
做內積後相加

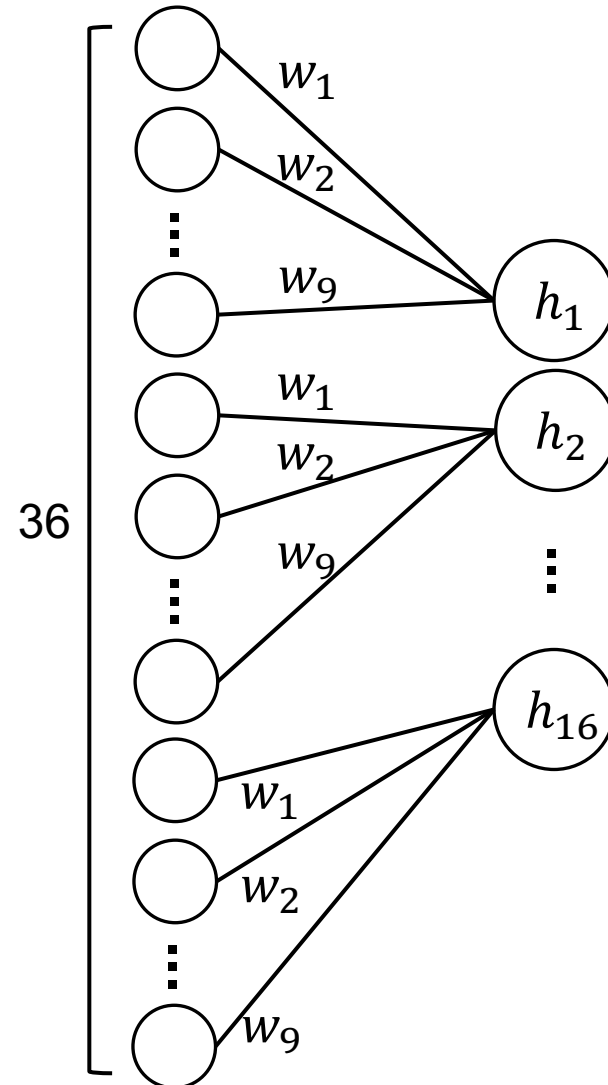
3	0	0
0	3	0
0	0	-3

↓
3



Convolutional layer is not fully-connected

1	1	1	1	0	0
0	1	1	0	1	0
0	0	1	1	0	0
0	0	1	1	1	0
0	0	0	1	1	0
0	0	0	0	0	0



1	0	0
0	1	0
0	0	-1

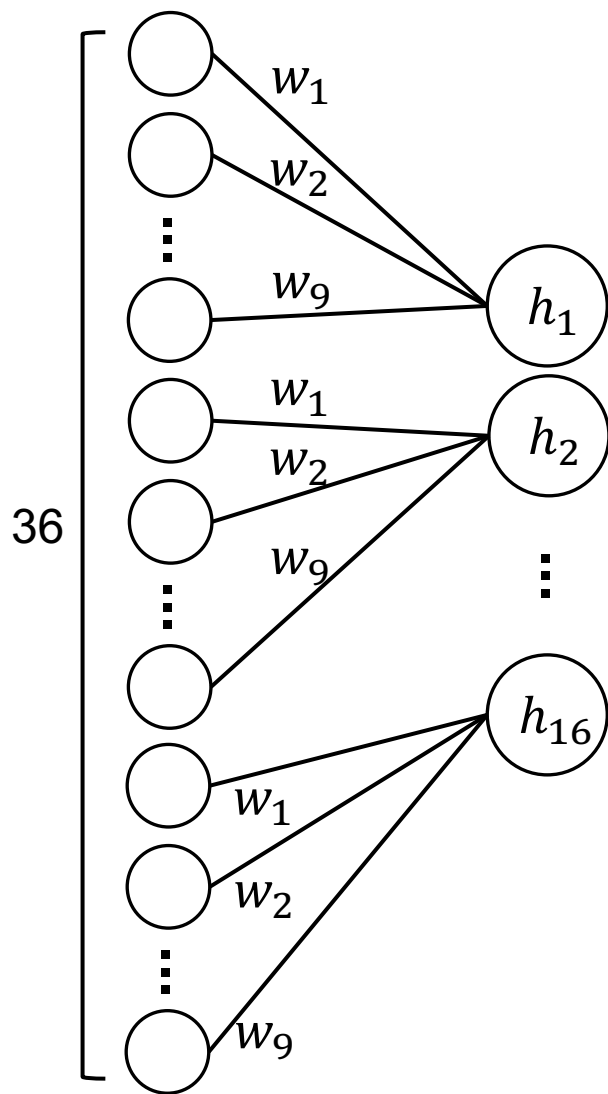
Filter, stride=1

h_1	h_2	h_3	h_4
h_5	h_6	h_7	h_8
h_9	h_{10}	h_{11}	h_{12}
h_{13}	h_{14}	h_{15}	h_{16}

feature map



Convolutional layer vs. FC layer 參數量

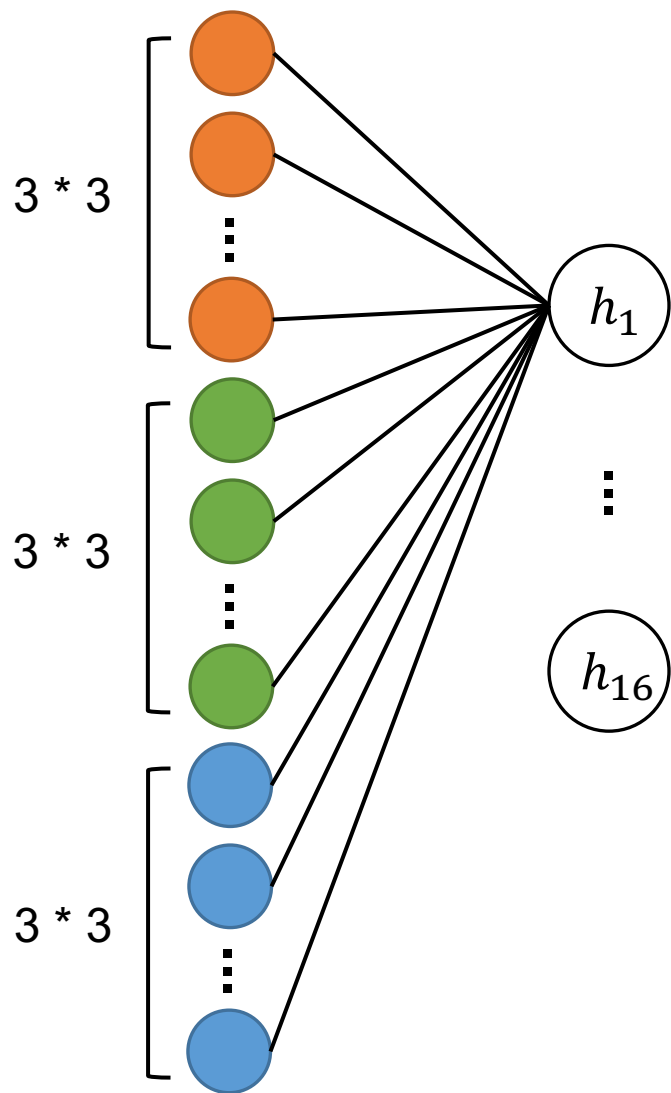


FC: fully-connected

Grayscale images	參數量比較 (不算 bias 數)
FC layer	$36 * 16 = 576$
Conv layer	$3 * 3 = 9$



Convolutional layer vs. FC layer 參數量



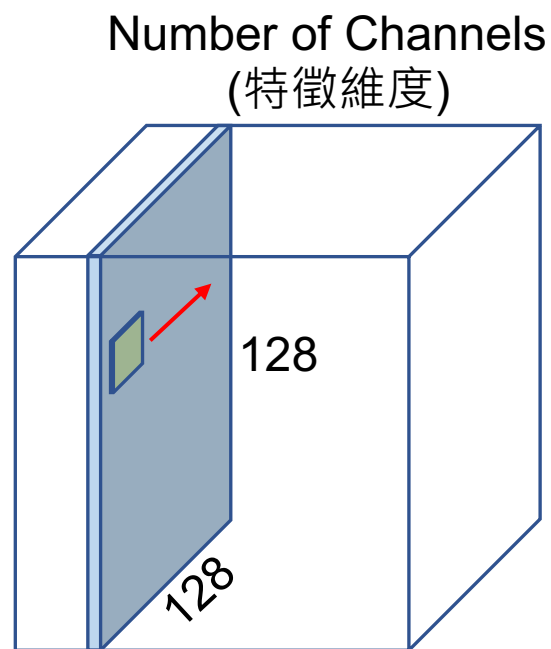
FC: fully-connected

RGB images	參數量比較 (不算 bias 數)
FC layer	$3 * 128 * 128 * 16 = 786,432$
Conv layer (num_filter = 1)	$3 * 3 * 3 = 27$
Conv layer (num_filter = 16)	$3 * 3 * 3 * 16 = 432$



Implementation in PyTorch

- 我們使用 `torch.nn.Conv2d`
- Why not 3d?



2d 卷積僅沿著空間維度
(Height, Width) 進行



Pooling (example of Max Pooling)

1	3	1	2
-1	1	1	0
0	1	1	0
0	0	1	2



參數：

- kernel_size=2
- stride = 2

3	2
1	2



Pooling (example of **Average** Pooling)

1	3	1	2
-1	1	1	0
0	1	1	0
0	0	1	2



參數：

- kernel_size=2
- stride = 2

1	1
0.25	1



Pooling (example of Max Pooling)

1	3	1	2
-1	1	1	0
0	1	1	0
0	0	1	2



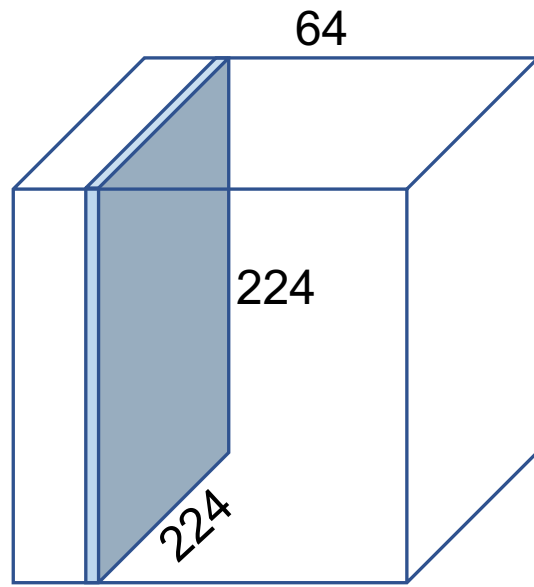
參數：

- kernel_size=2
- stride = 1

3	3	2
1	1	1
1	1	2



2D Pooling on RGB Images

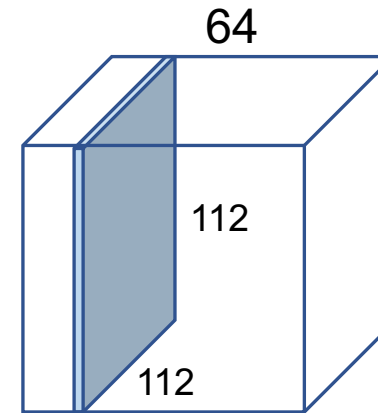


224 x 224 x 64
(H x W x C)



參數：

- kernel_size=2
- stride = 2



112 x 112 x 64
(H x W x C)



[Important Notes] Meaning of Pooling

- To maintain the most relevant information
 - Summarizing the features within the region covered by the filter
- Shortage:
 - No theoretical guarantee for making better model performance

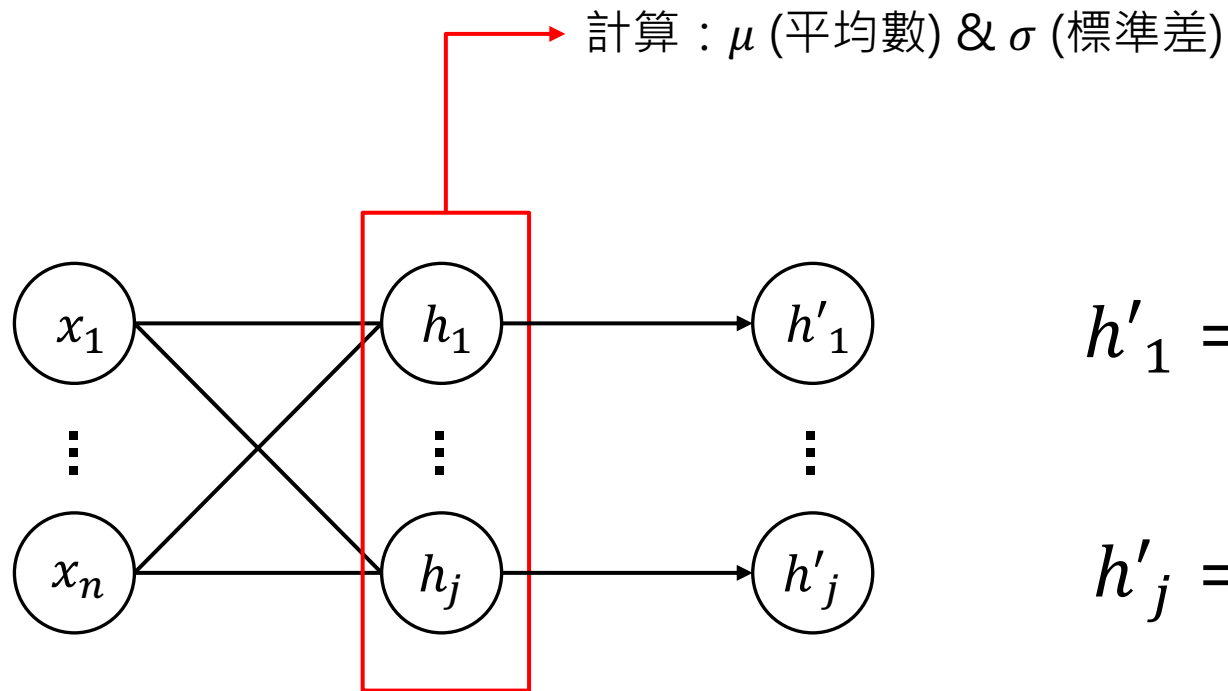


CNN 的結局

- 通常不會只有一層的 Convolutional Layer
- 隨著層數增加，通常 Feature map 會越來越小，因為：
 - Convolutional Layer 可能會縮小 feature map (如果沒有 padding)
 - Pooling 也會縮小 feature map
- 經過了多次的 Convolution + Pooling 後的 feature map 為壓縮後的重要資訊



Batch Normalization (BN)

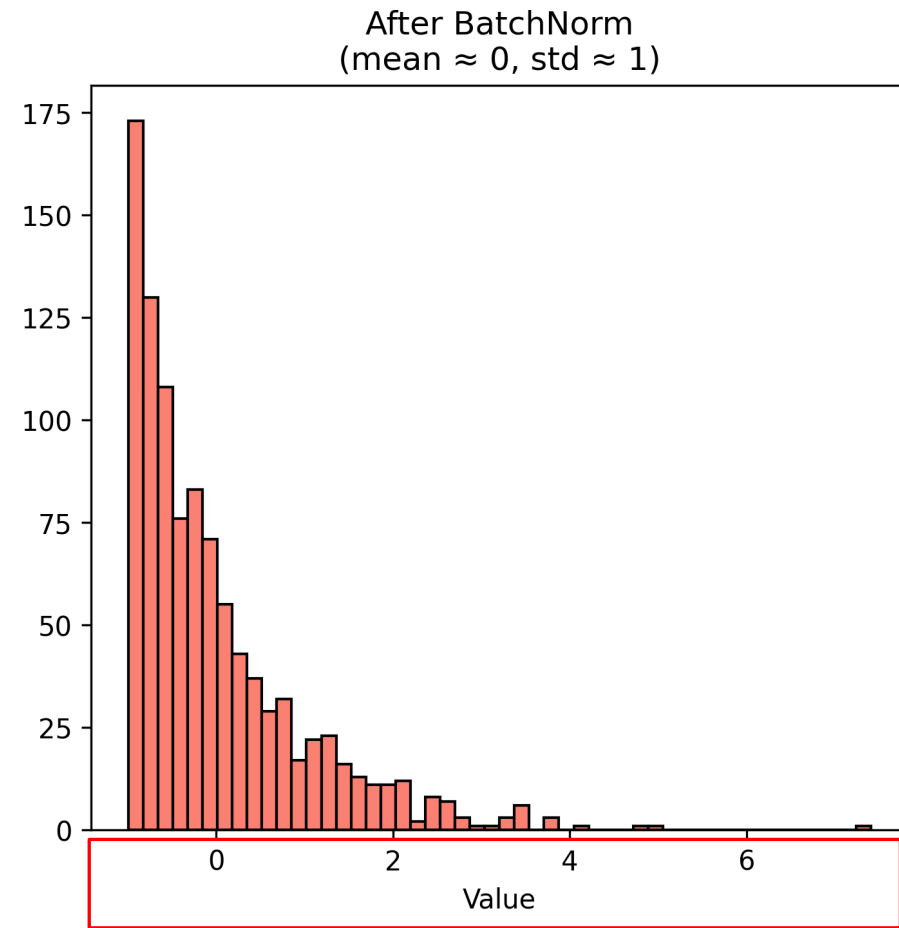
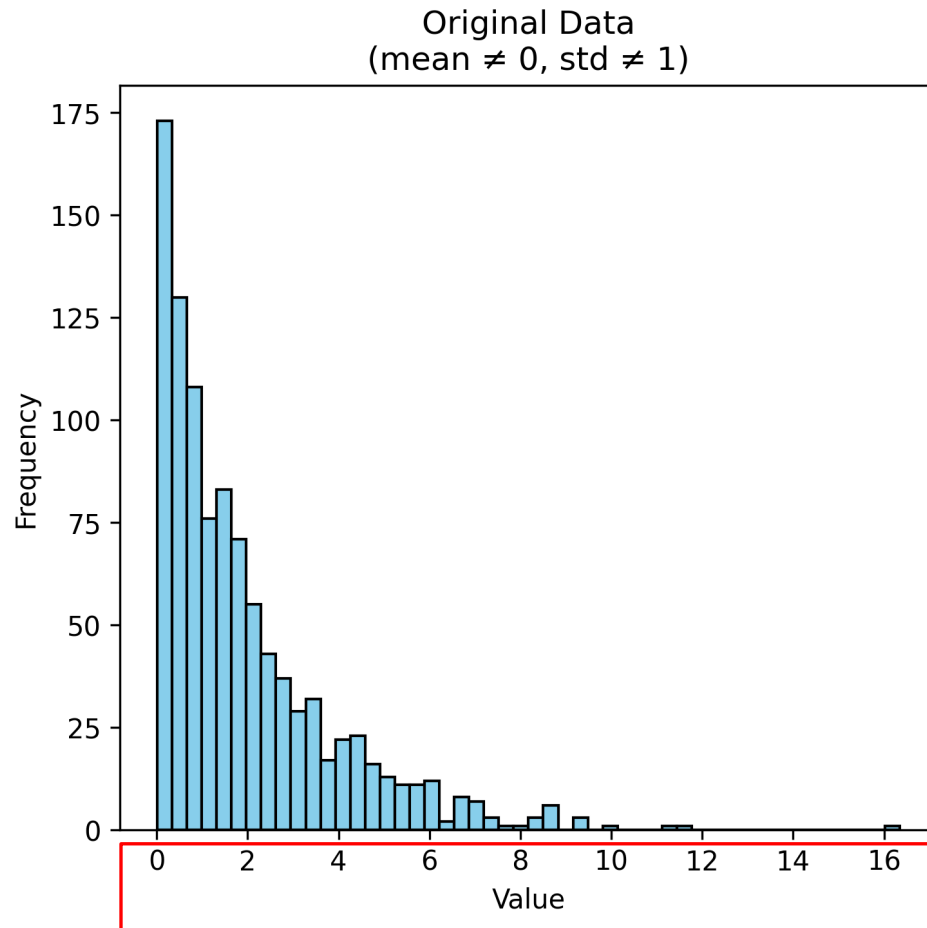


$$h'_1 = \frac{h_1 - \mu}{\sigma}$$

$$h'_j = \frac{h_j - \mu}{\sigma}$$



Value comparison example with BN

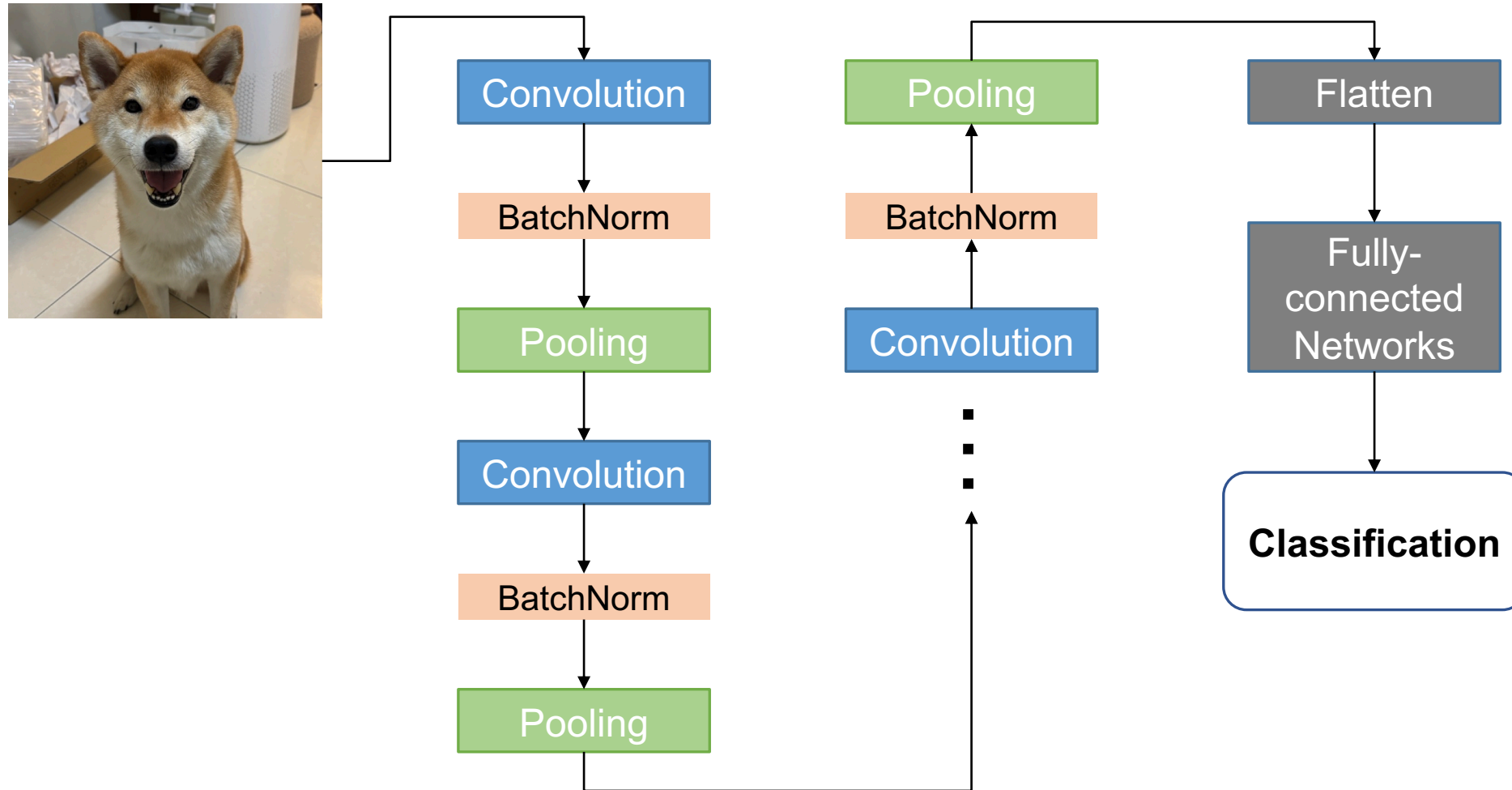


BN Summary

- BN 對數值進行轉換，使得轉換後的數值的平均值為零、標準差為1，類似於統計上的 Z-score standardization
- BN 可以：
 - 減少不同特徵尺度帶來的偏差
 - 提高模型訓練時的穩定性

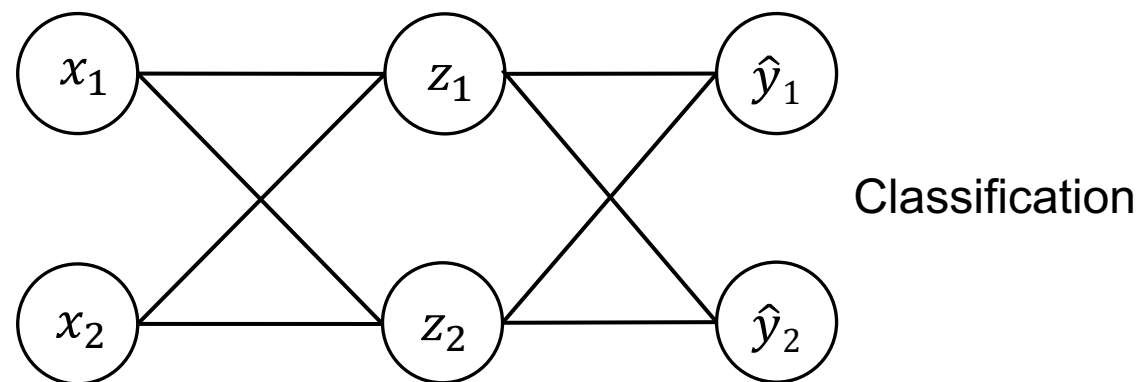
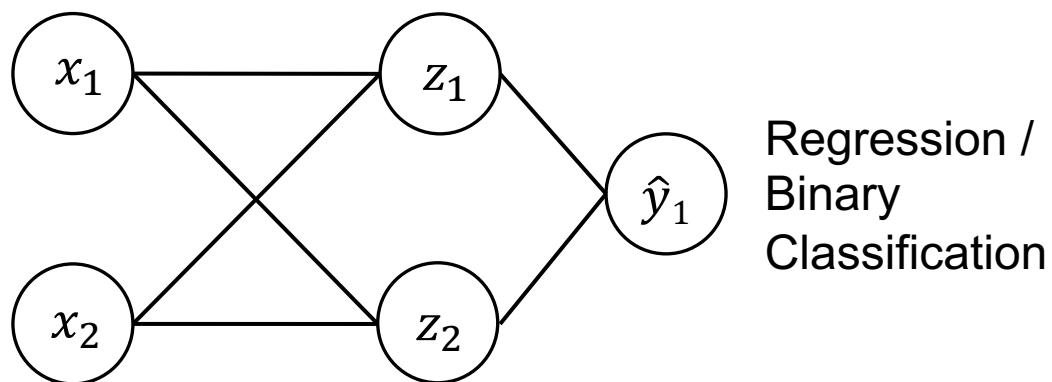


The whole CNN



Fully-connected Networks

- 常被稱為全連接層 (Fully-connected layers, FC)
- Multi-layer perceptron (MLP) 即是由 FC 所構成
- 注意! CNN 不是全連接層



Why Flatten?

- 以分類任務來說，我們需要 Fully-connected layers
- CNN 常被用於分類任務



CNN Summary

- 為影像資料量身定做，用來捕捉空間資訊 (Spatial Information)
- Parameter sharing (參數共享機制)
 - 降低用 MLP 來進行影像任務的參數量



ConvNets are everywhere

Object Detection with YOLO

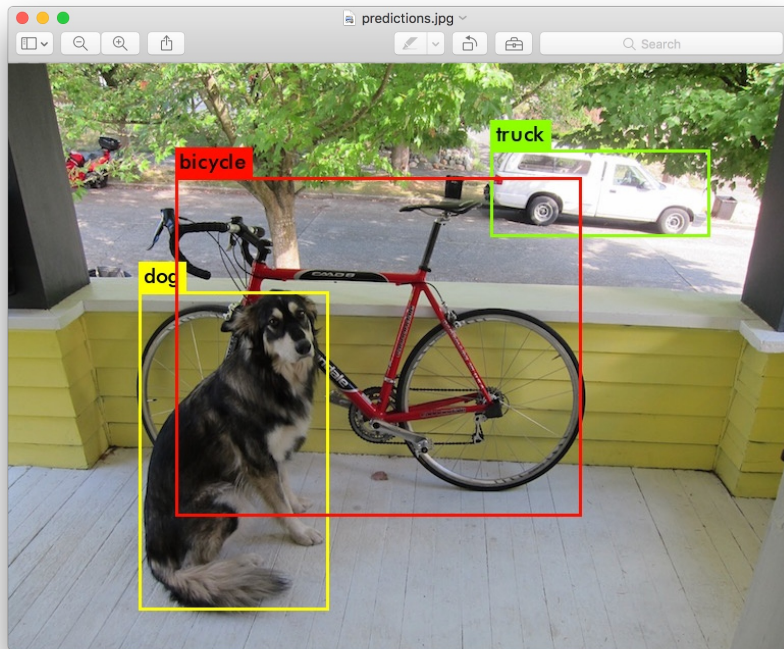


Figure source: <https://pjreddie.com/darknet/yolo/>

Style Transfer with CNN



Figure source: Gatys, Leon A., Alexander S. Ecker, and Matthias Bethge. "Image style transfer using convolutional neural networks." CVPR. 2016.



Advanced CNNs

- [ResNet] Deep Residual Learning for Image Recognition
 - <https://arxiv.org/abs/1512.03385>
- [DenseNet] Densely Connected Convolutional Networks
 - <https://arxiv.org/abs/1608.06993>
- https://slazebni.cs.illinois.edu/spring17/lec04_advanced_cnn.pdf



Additional Links

- **PyTorch CNN example:**

https://pytorch.org/tutorials/beginner/blitz/cifar10_tutorial.html

- **Batch Normalization:**

- https://wandb.ai/wandb_fc/GroupNorm/reports/Group-Normalization-in-Pytorch-With-Examples---VmlldzoxMzU0MzMzMy
- <https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739/>



Thank you!

Instructor: 林英嘉

 yjlin@cgu.edu.tw

TA: 劉美辰

 m1461014@cgu.edu.tw