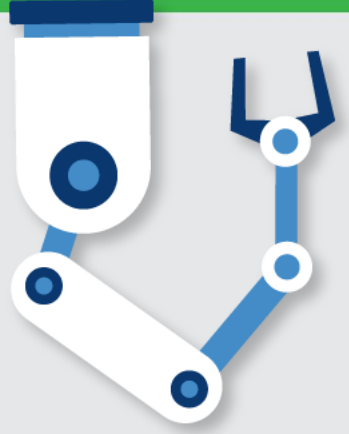
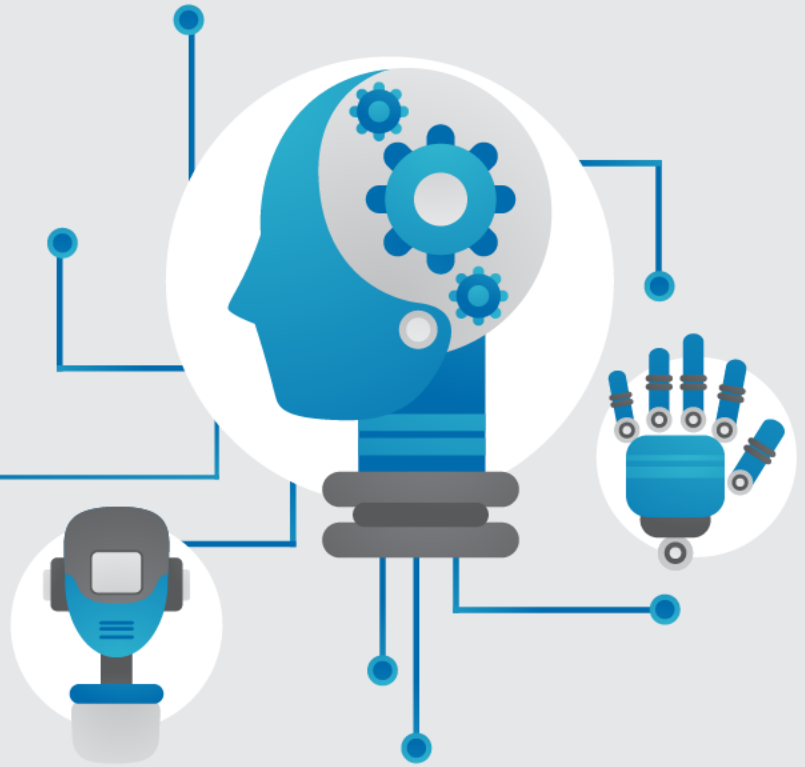
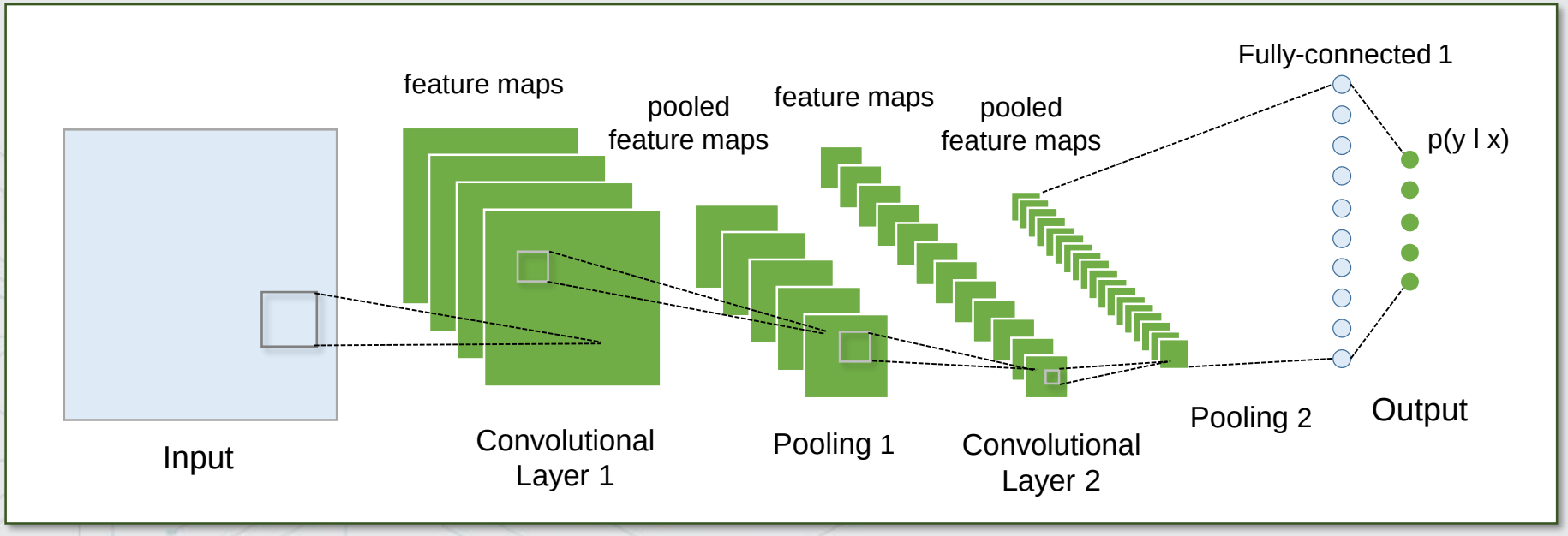


CNN經典模型介紹



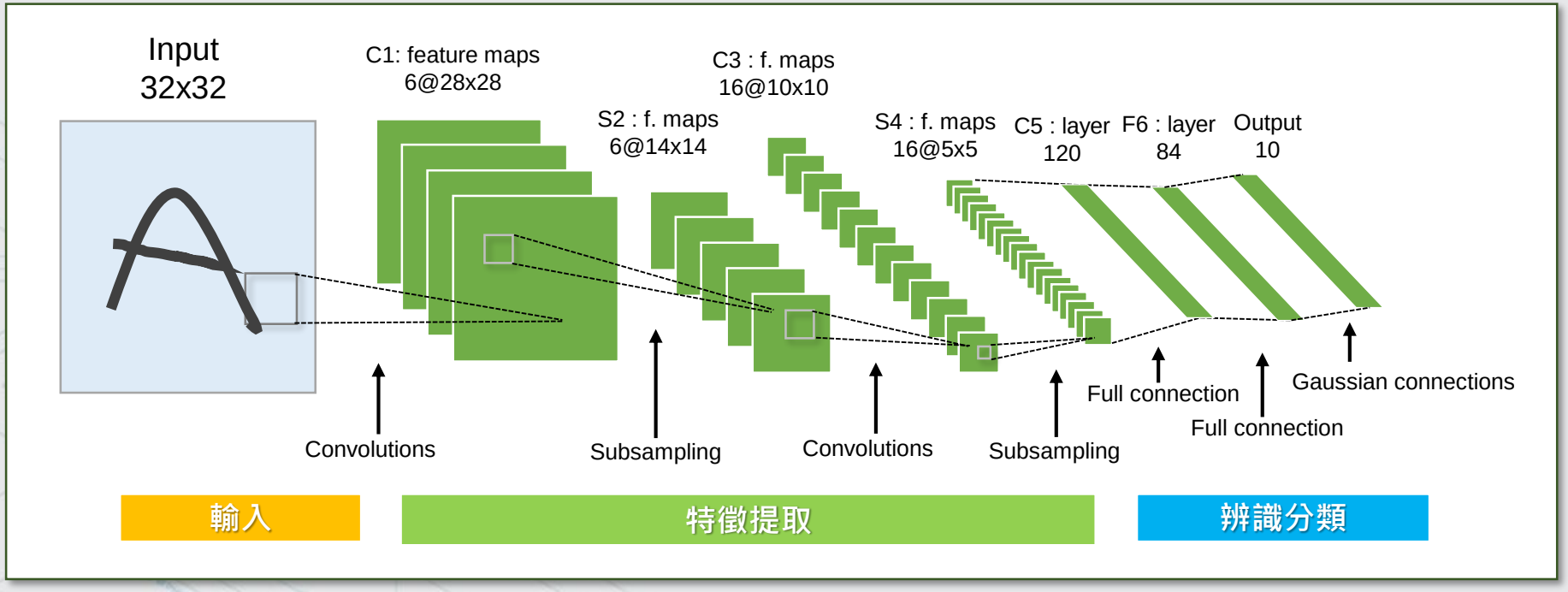
CNN經典模型

- LeNet
- VGGNet
- AlexNet
- ResNet



LeNet

- 是由Yann LeCun在1998年提出，用於手寫數字、字母的識別。
- 利用卷積、池化等操作提取特徵，避免了大量的計算成本，最後再使用全連接神經網絡進行分類識別，這個網路也是最近深度神經網路架構的基礎。



LeNet 架構

- LeNet論文網址：http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf
- 模型架構

Layer		Feature Map	Size	Kernel Size	Stride	Activation
0	Input	1	32X32	-	-	-
1	Convolution	6	28X28	5X5	1	tanh
2	Average Pooling	6	14X14	2X2	2	
3	Convolution	16	10X10	5X5	1	tanh
4	Average Pooling	16	5X5	2X2	2	
5	FC	-	120	-	-	tanh
6	FC	-	84	-	-	tanh
7	Output	-	10	-	-	softmax

LeNet 實作

- 範例程式碼片段

```
import keras
from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense, Flatten
from keras.layers import Conv2D, AveragePooling2D

#Build LetNet model
def LetNet(input_shape, num_classes):
    model = Sequential()
    model.add(Conv2D(filters=6, kernel_size=(5, 5), strides=(1,1),
                     input_shape=input_shape, activation='tanh', padding='same'))
    model.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Conv2D(filters=16, kernel_size=(5, 5), strides=(1,1),
                     input_shape=input_shape, activation='tanh', padding='same'))
    model.add(AveragePooling2D(pool_size=(2, 2), strides=(2, 2)))
    model.add(Flatten())
    model.add(Dense(120,activation = 'tanh'))
    model.add(Dense(84,activation = 'tanh'))
    model.add(Dense(num_classes,activation='softmax'))
    return model
```

AlexNet

- 是由A. Krizhevsky, I. Sutskever和G. E. Hinton於2012年發表。
- AlexNet 在ILSVRC（ImageNet Large Scale Visual Recognition Challenge）-2012競賽中以top-5 15.3%的錯誤率，遠超越第二名的26.2%。使得CNN開始被重視，之後的比賽也都是由CNN拿下冠軍，深度學習正式大爆發。

- AlexNet論文網址

https://www.nvidia.cn/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf

AlexNet 架構

AlexNet的特點

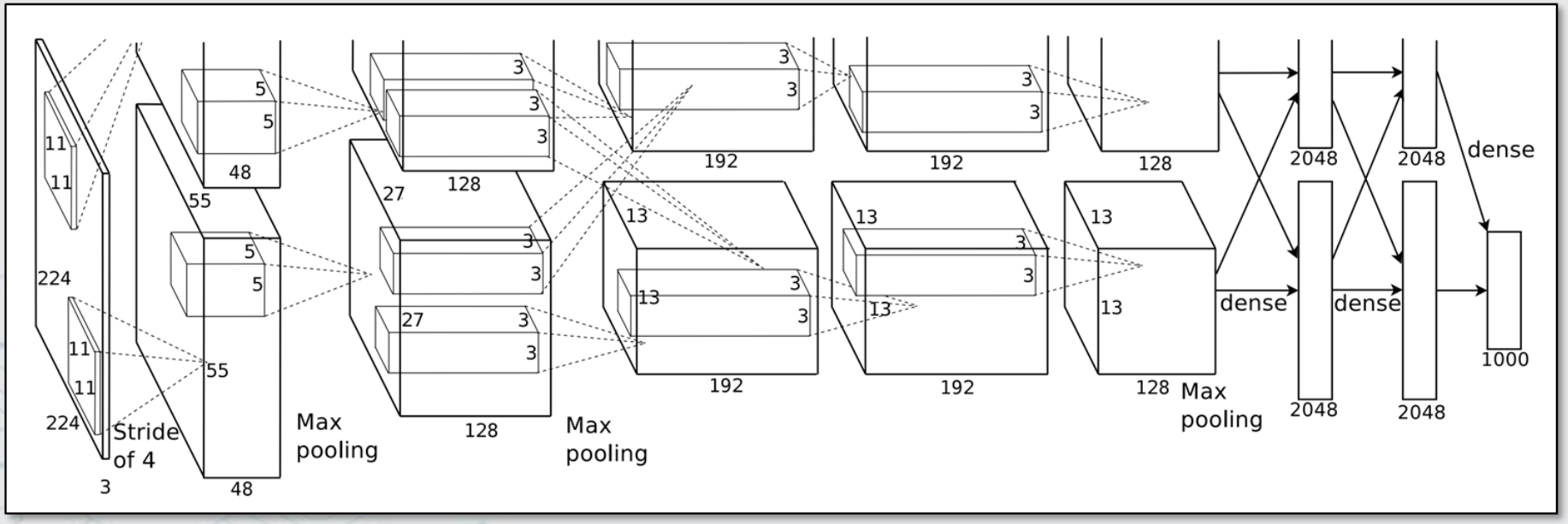
a. 使用了非線性的激活函數（Activation function）：Relu，
在AlexNet以前，大部分的網路都是用tanh當作激活函數

b. 使用了Data augmentation 和 Dropout 來防止Overfitting

c. 使用mini-batch SGD（也成隨機梯度下降）來加快訓練

AlexNet 架構

- AlexNet包含8個學習層，5個卷積層和3個完全連接層



引用文獻 Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks,2012.

AlexNet 實作

- 範例程式碼片段

```
#Build AlexNet model
def AlexNet(input_shape, num_classes):
    model = Sequential()

    model.add(Conv2D(filters=96, input_shape=input_shape,
          kernel_size=(11,11),strides=(4,4), activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))
    model.add(BatchNormalization())

    model.add(Conv2D(filters=256, kernel_size=(11,11), strides=(1,1),
           activation='relu', padding='same'))
    model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))
    model.add(BatchNormalization())

    model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
          activation='relu', padding='same'))
    model.add(BatchNormalization())

    model.add(Conv2D(filters=384, kernel_size=(3,3), strides=(1,1),
          activation='relu', padding='same'))
    model.add(BatchNormalization())
```

AlexNet 實作

- 範例程式碼片段

```
model.add(Conv2D(filters=256, kernel_size=(3,3), strides=(1,1),
      activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2), padding='same'))
model.add(BatchNormalization())

model.add(Flatten())

model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.4))
model.add(BatchNormalization())

model.add(Dense(4096, activation='relu'))
model.add(Dropout(0.4))
model.add(BatchNormalization())

model.add(Dense(1000, activation='relu'))
model.add(Dropout(0.4))
model.add(BatchNormalization())

model.add(Dense(10, activation='softmax'))

return model
```

VGGNet

Published as a conference paper at ICLR 2015

VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION

Karen Simonyan* **& Andrew Zisserman+**
Visual Geometry Group, Department of Engineering Science, University of Oxford
{karen,az}@robots.ox.ac.uk

onal network depth on its
Our main contribution is
using an architecture with
a significant improvement
shing the depth to 16–19
mageNet Challenge 2014
ond places in the localisa-
v that our representations
ate-of-the-art results. We
ublicly available to facili-
ations in computer vision.

- VGGNet是由K. Simonyan 和 A. Zisserman在2014年提出，並且獲得ILSVRC分類項目中的亞軍。
- VGGNet論文
  https://arxiv.org/pdf/1409.1556.pdf

VGGNet

VGGNet的特性

a. 結構簡潔，包含5層卷積層、3層全連接層、softmax輸出層。層與層之間使用MaxPooling（最大池化）分開，所有隱層的激活函數都採用ReLU函數。

b. VGG的卷積層使用多個較小卷積核（3x3）代替一個較大的卷積核，一方面可以減少參數，另一方面相當於進行了更多的非線性映射，可增加網絡的擬合/表達能力。

c. 小池化核，相較於AlexNet的3x3的池化核，VGG全部採用2x2的池化核。

d. 通道數多，VGG網路第一層的通道數為64，後面每層都加倍，最多到512個通道，通道數的增加，使得更多的特徵可以被提取出來。

VGGNet 架構

網路架構

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

ConvNet Configuration				VGG16	VGG19
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

引用文獻 **Karen Simonyan,Andrew Zisserman,Very Deep Convolutional Networks for Large-Scale Image Recognition,2015.**

VGGNet 參數(I)

keras.applications.vgg16.VGG16(include_top=True, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)

參數

a. include_top：是否包括頂層的全連接層

b. weights：None代表隨機初始化， 'imagenet' 代表加載在 ImageNet 上預訓練的權值

c. input_tensor：Keras tensor 作為模型的輸入
（即 layers.Input() 輸出的 tensor ）

d. input_shape：輸入尺寸元組，僅當include_top=False 時有效，應為長為3的tuple，指定輸入圖片的形狀，圖片的寬高必須大於48。

VGGNet 參數(II)

keras.applications.vgg16.VGG16(include_top=True, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)

- pooling：當include_top為False時，該參數指定了特徵提取時的池化方式

a. None 代表不池化

b. 'avg' 代表採用平均池化（AveragePooling2D）

c. 'max' 代表採用最大池化（MaxPooling2D）

- classes：圖片分類的類別數，僅當include_top為True並且不加載預訓練權值時可用

VGGNet 實作

› 範例程式碼片段

```
# modify the model
def modified_model(img_shape, num_classes):
    model_vgg16 = VGG16(include_top=False,weights='imagenet')
    # Make vgg16 model layers as non trainable
    for layer in model_vgg16.layers:
        layer.trainable = False

    img_input = Input(shape=img_shape)
    img_model_vgg16 = model_vgg16(img_input)

    x = Flatten(name='flatten')(img_model_vgg16)
    x = Dense(256, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    x = Dense(num_classes, activation='softmax')(x)

    modified_model = Model(inputs=img_input, outputs=x)
    modified_model.compile(loss='sparse_categorical_crossentropy',
        optimizer='adam', metrics=['acc'])

    return modified_model
```

VGGNet 實作

› 範例程式碼片段

```
# loading the data
(x_train, y_train), (x_test, y_test) = mnist.load_data()

# converting it to RGB
x_train = [cv2.cvtColor(cv2.resize(i, (32,32)), cv2.COLOR_GRAY2BGR) for i in x_train]
x_train = np.concatenate([arr[np.newaxis] for arr in x_train]).astype('float32')

x_test = [cv2.cvtColor(cv2.resize(i, (32,32)), cv2.COLOR_GRAY2BGR) for i in x_test]
x_test = np.concatenate([arr[np.newaxis] for arr in x_test]).astype('float32')

# training the model
model = modified_model(x_train.shape[1:], len(set(y_train)))
```

ResNet

- ResNet是由K. He, X. Zhang, S. Ren, 和 J. Sun在2015年提出，是當年的ILSVRC分類項目的冠軍。
  ResNet為ILSVRC競賽上首次錯誤率低於人類表現。
- ResNet論文
  https://arxiv.org/pdf/1512.03385.pdf

- 殘差神經網路的名稱指的是在傳統卷積神經網路中加入殘差學習（residual learning）的概念，解決了深層網路中梯度消失和精度下降（訓練集）的問題，使網路能夠越來越深，既保證了精確度，又控制了速度。

ResNet 概念

- ResNet Block

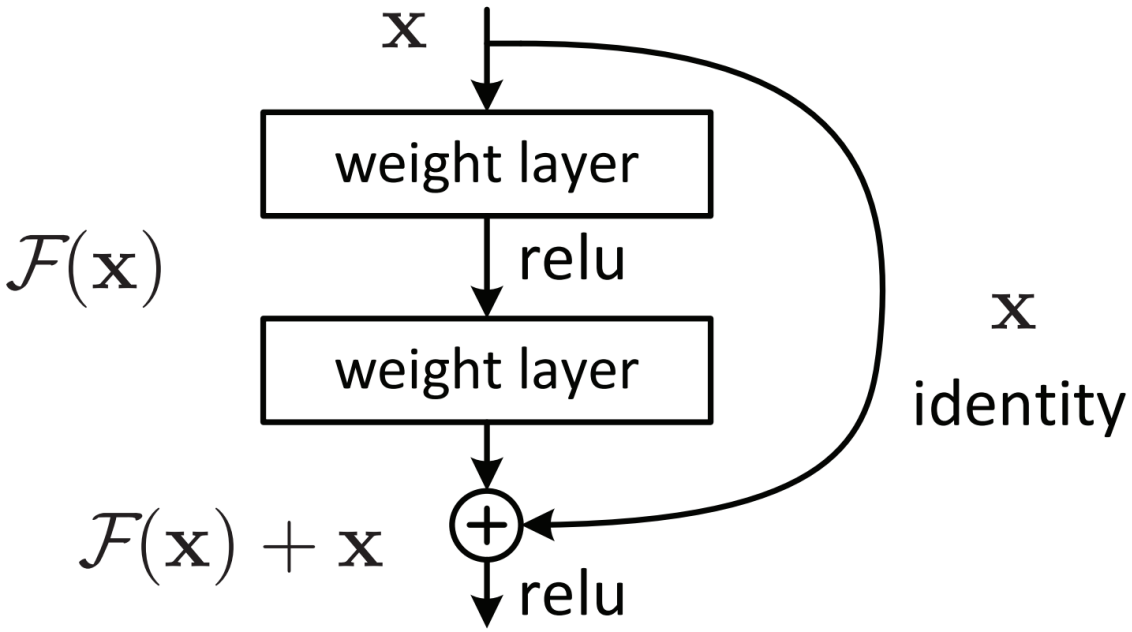


Figure 2. Residual learning: a building block.

引用文獻 **Kaiming He,Xiangyu Zhang,Shaoqing Ren,Jian Sun,Deep Residual Learning for Image Recognition,2015.**

ResNet 概念

網路架構

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table 1. Architectures for ImageNet. Building blocks are shown in brackets (see also Fig. 5), with the numbers of blocks stacked. Down-sampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2.

引用文獻 **Kaiming He,Xiangyu Zhang,Shaoqing Ren,Jian Sun,Deep Residual Learning for Image Recognition,2015.**

ResNet 實作

keras.applications.resnet.ResNet50(include_top=True, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)

› **參數**

a. include_top：是否包括頂層的全連接層

b. weights：None 代表隨機初始化， 'imagenet' 代表加載在 ImageNet 上預訓練的權值

c. input_tensor：Keras tensor 作為模型的輸入（即 layers.Input() 輸出的 tensor ）

d. input_shape：輸入尺寸元組，僅當include_top=False 時有效，應為長為3的tuple，指定輸入圖片的形狀，圖片的寬高必須大於 48

ResNet

keras.applications.resnet.ResNet50(include_top=True, weights='imagenet', input_tensor=None, input_shape=None, pooling=None, classes=1000)

- pooling：當include_top為False時，該參數指定了特徵提取時的池化方式

a. None 代表不池化

b. 'avg' 代表採用平均池化（AveragePooling2D）

c. 'max' 代表採用最大池化（MaxPooling2D）

- classes：圖片分類的類別數，僅當include_top為True並且不加載預訓練權值時可用

ResNet 實作

› 範例程式碼片段

```
# modify the model
def modified_model(img_shape, num_classes):
    model_resnet = ResNet50(include_top=False,weights='imagenet')
    # Make resnet50 model layers as non trainable
    for layer in model_resnet.layers:
        layer.trainable = False

    img_input = Input(shape=img_shape)
    img_model_resnet = model_resnet(img_input)

    x = Flatten(name='flatten')(img_model_resnet)
    x = Dense(256, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    x = Dense(num_classes, activation='softmax')(x)

    modified_model = Model(inputs=img_input, outputs=x)
    modified_model.compile(loss='sparse_categorical_crossentropy',
                  optimizer='adam', metrics=['acc'])

    return modified_model
```