RESNET & DENSENET

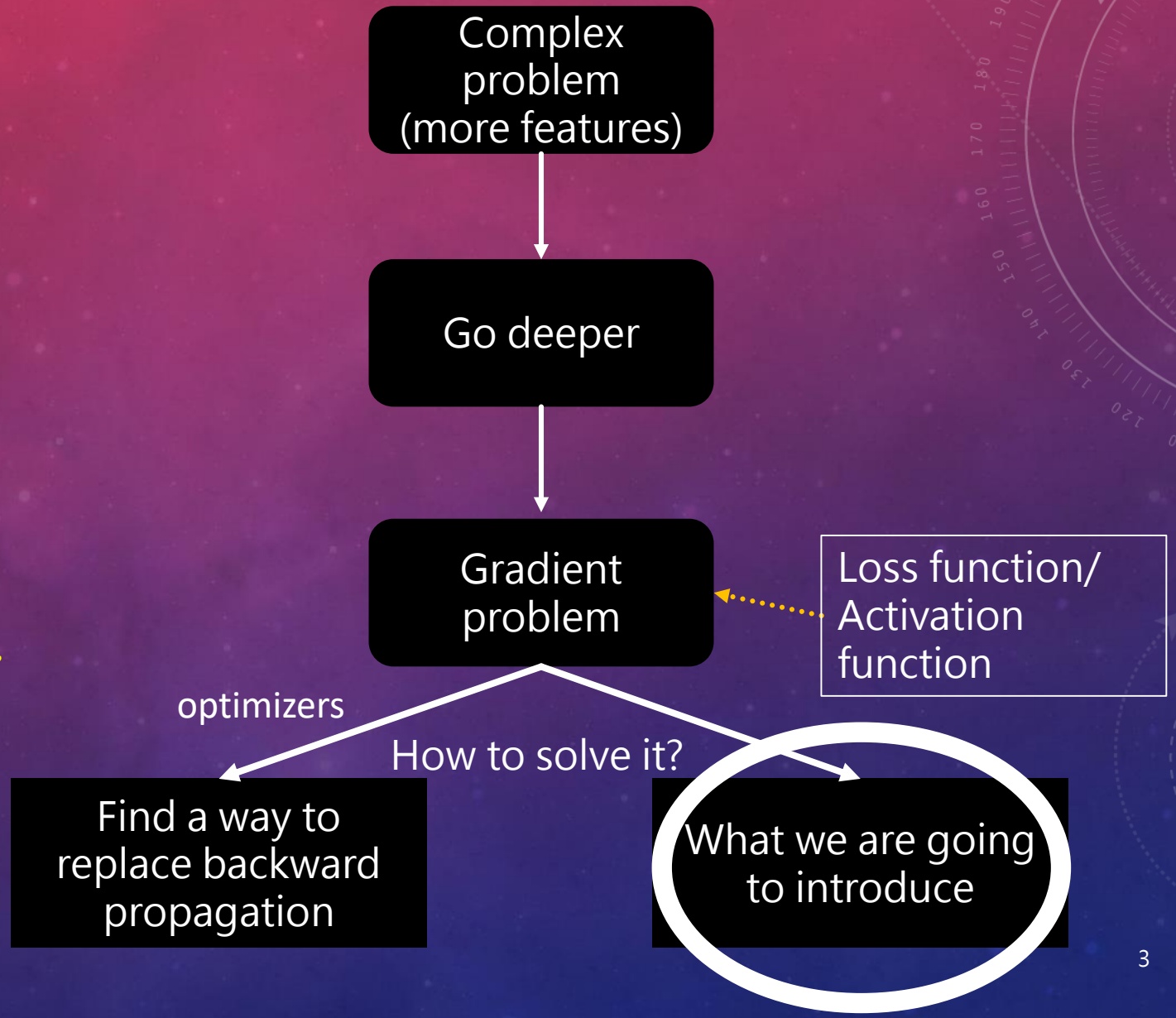
張家瑋 博士

助理教授

國立臺中科技大學資訊工程系

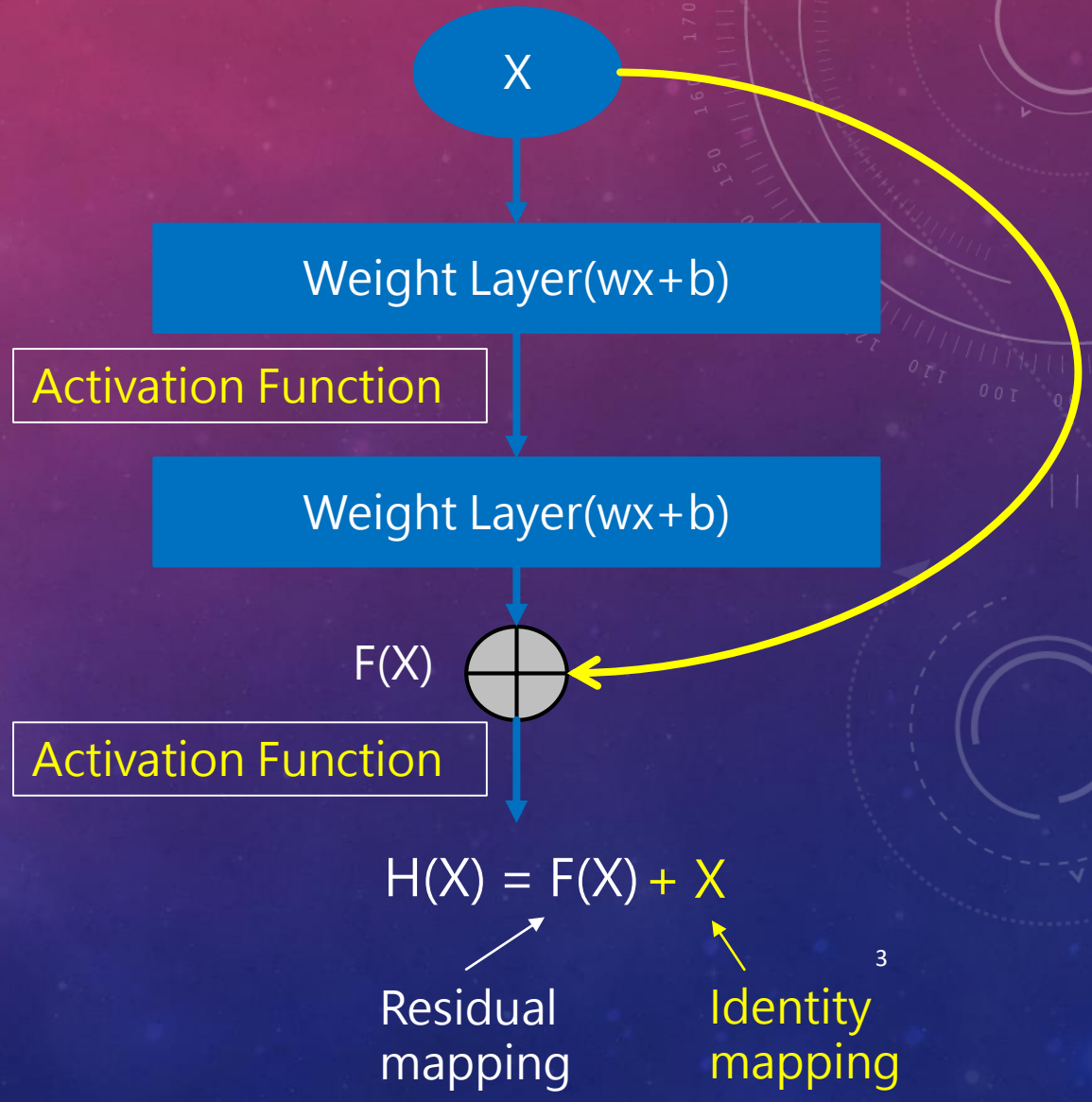
梯度爆炸/消失

- 深度學習隨著層數增加，效果越好。
- 使用全連接網絡時，有梯度消失/爆炸現象（gradient vanishing / exploding），使用卷積層以及歸一初始化，BN（劃重點）後解決問題;
- 準確率惡化（degradation），訓練時神經網絡收斂，隨著層數的增加，訓練誤差和驗證誤差不是一直減小，在達到一定層數後，誤差反而隨之增長。



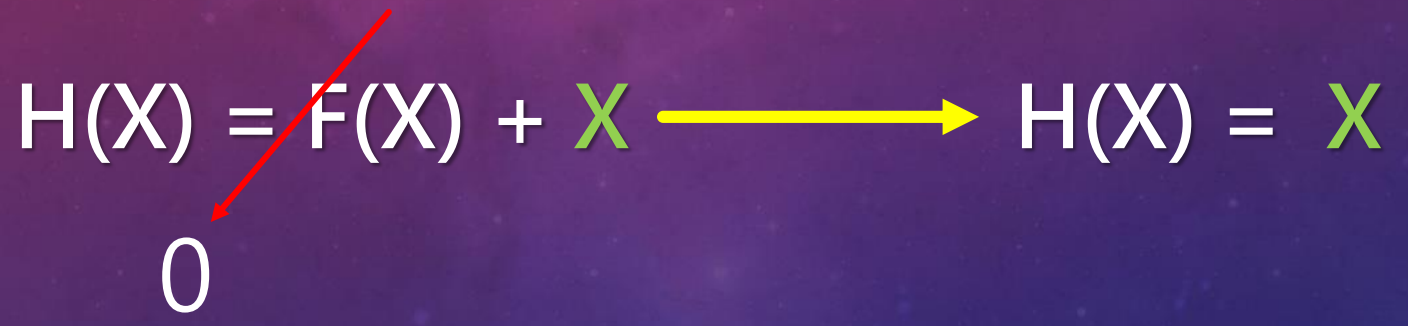
Residual Neural Network

- Shortcut connection



恒等映射

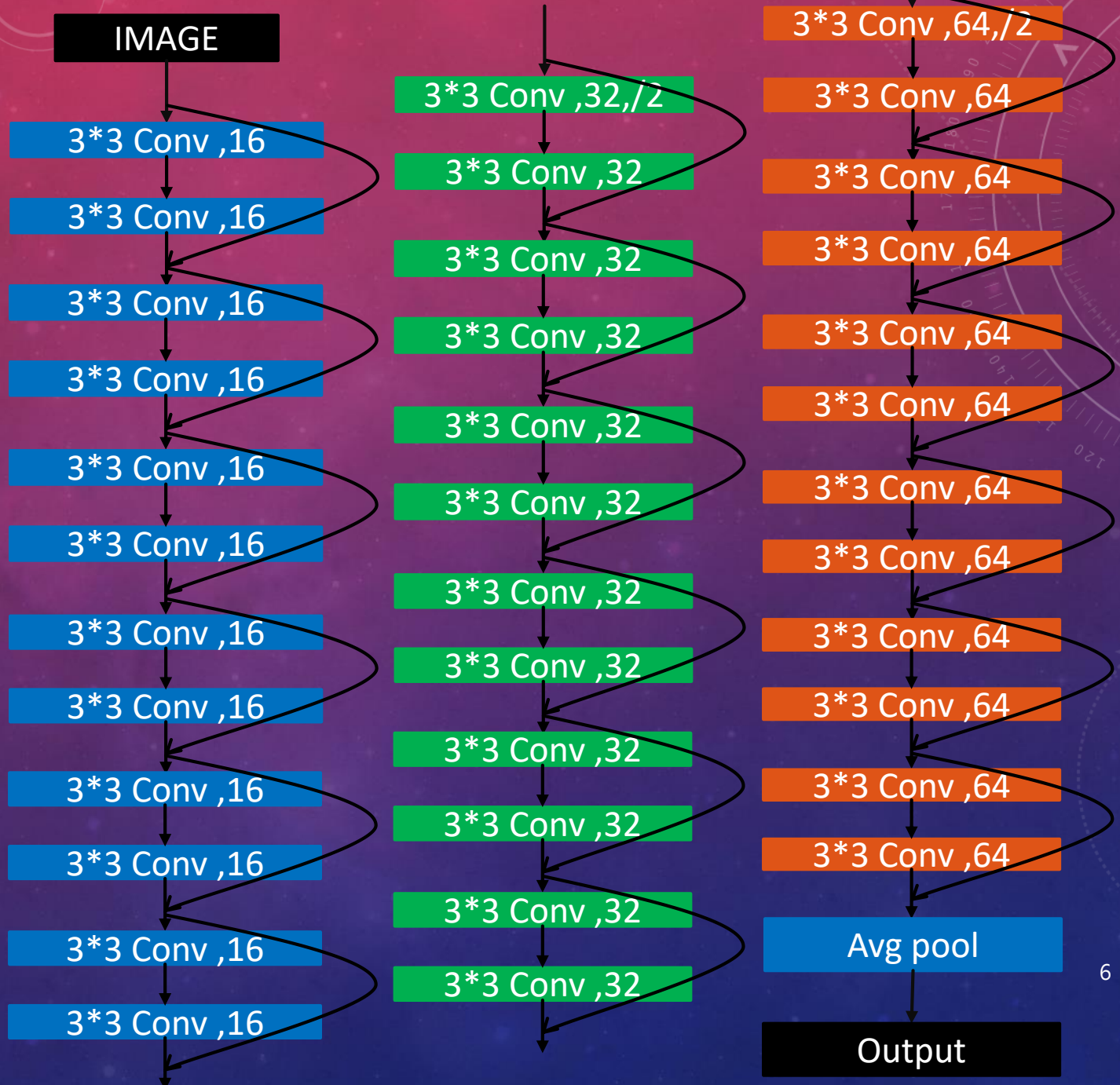
- 在更深層的網路中，為了不要使前層參數被更深層網路擾動過大，有人提出等效傳遞，ResNet 便有效達成這個方法。
- 對於輸出H(X)=F(X)+X而言，為了使恒等映射(identity mapping)，優化目標即能夠使F(X)夠小就可以使H(X)=X了。
- 意即如果下層的誤差變大，會自動將 F (x) → 0 處理。



貢獻

1. 解決了網絡惡化問題，準確度隨層數增加而增加。
2. 實現簡單，訓練方便，方向傳播使用 end to end 的 SGD，不用修改optimizers 。
3. 不增加網絡參數，可以看到152層的 Resnet 比 VGG-19 的網絡參數還要少，計算複雜度降低。

RESNET 39 LAYERS



實際案例

IMPLEMENT ON CIFAR10 WITH KERAS

RESNET-CIFAR10(1/8)

引入所需套件:

```
import keras
from keras.models import Model
from keras.utils import to_categorical                    #使label做one-hot encoding
from keras.datasets import cifar10                        #本次實例使用dataset
from keras.layers import Dense,Conv2D,Activation          #Dense建構一般層、Conv2D建構CNN層
from keras.layers import BatchNormalization               #可加速收斂、控制overfitting，所以不用Dropout
from keras.layers import AveragePooling2D,Input,Flatten   #Flatten將多維輸入壓成一維，用於CNN層過渡到fully connected
from keras.regularizers import l2                         #使用l2正規化減少overfitting
from keras.callbacks import ModelCheckpoint               #調用fit中callback參數，ModelCheckpoint每一個epoch存一次模型
from keras.callbacks import LearningRateScheduler         #動態調整學習率
from keras.callbacks import ReduceLROnPlateau             #訓練停滯時降低學習率
import matplotlib.pyplot as plt                           #可視化套件
```

RESNET-CIFAR10(1/8)

載入資料集與正規化:

```
###---load dataset---###
(x_train,y_train),(x_test,y_test) = cifar10.load_data()

###---normalization---###
x_train = x_train/255
x_test = x_test/255
```

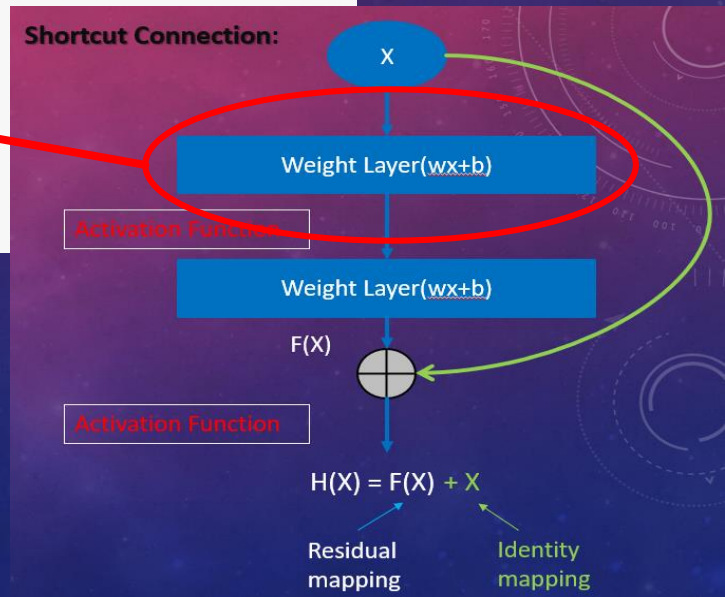
對Label做one-hot encoding:

```
###---one-hot encoding---###
y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

RESNET-CIFAR10(2/8)

定義 weight layer:

```
###---define resnet_block---###
#使用keras中functional方式建立神經網路層 : ly=NN(參數)(input)#
def resnet_block(inputs,filters=16,kernel_size=(3,3),strides=1,activation='relu'):
    x = Conv2D(filters,kernel_size=kernel_size,strides=strides,
               padding='same',kernel_initializer='he_normal',
               kernel_regularizer=l2(1e-4))(inputs)
    x = BatchNormalization()(x)
    if(activation):
        x = Activation(activation)(x)
    return x
```



RESNET-CIFAR10(3/8)

創建ResNet(39層):

輸入&2~13層:

```
###---adding layers---###
def resnet20(input_shape):
    inputs = Input(shape = input_shape)

    x = resnet_block(inputs)                                    #輸入層

    for ly in range(6):                                         #六層filter=16的shortcut connection
        w1 = resnet_block(inputs=x)
        w2 = resnet_block(inputs=w1,activation=None)
        x = keras.layers.add([x,w2])                            #ResNet精隨!!!! 加上identity mapping
        x = Activation('relu')(x)
```

Shortcut connection

RESNET-CIFAR10(4/8)

創建ResNet(39層):
14~25層:

```
for ly2 in range(6):                                        #六層filter=32的shortcut connection
    if ly2 == 0:
        w1 = resnet_block(inputs=x,filters=32,strides=2)    #為使前一層的輸出shape與本層的輸入shape一致，strides=2
    else:
        w1 = resnet_block(inputs=x,filters=32)
    w2 = resnet_block(inputs=w1,activation=None,filters=32)

    if ly2 == 0:
        x = Conv2D(32,kernel_size=3,strides=2,padding='same',    #對第一層filter=32的x多做一次CNN使輸出shape與w2一致
                   kernel_initializer='he_normal',
                   kernel_regularizer=l2(1e-4))(x)
    x = keras.layers.add([x,w2])                            #ResNet精隨!!!! 加上identity mapping
    x = Activation('relu')(x)
```

RESNET-CIFAR10(5/8)

創建ResNet(39層):
26~37層&avgpool&輸出層:

```
for ly3 in range(6):                                          #六層filter=64的shortcut connection
    if ly3 == 0:
        w1 = resnet_block(inputs=x,filters=64,strides=2)
    else:
        w1 = resnet_block(inputs=x,filters=64)
    w2 = resnet_block(inputs=w1,activation=None,filters=64)

    if ly3 == 0:
        x = Conv2D(64,kernel_size=3,strides=2,padding='same',     #對第一層filter=64的x多做一次CNN使輸出shape與w2一致
                   kernel_initializer='he_normal',
                   kernel_regularizer=l2(1e-4))(x)
    x = keras.layers.add([x,w2])                                  #ResNet精隨!!!! 加上identity mapping
    x = Activation('relu')(x)

x = AveragePooling2D(pool_size=2)(x)
y = Flatten()(x)
outputs = Dense(10,activation='softmax',kernel_initializer='he_normal')(y)  #輸出層

model = Model(inputs=inputs,outputs=outputs)
return model
```

RESNET-CIFAR10(6/8)

設定loss_function、optimizer:

```
###---調整模型參數###
model = resnet20((32,32,3))                          #cifar10的shape(32,32,3)
model.compile(loss='categorical_crossentropy',
              optimizer=keras.optimizers.Adam(),
              metrics=['accuracy'])
model.summary()
```

Callback:

```
###---Callback---###
checkpoint = ModelCheckpoint(filepath='./cifar10_resnet.h5',
                             monitor = 'val_acc',verbose=1,
                             save_best_only=True)

def lr_sch(epoch):
    if epoch<50:
        return 1e-3
    if 50<=epoch<100:
        return 1e-4
    if epoch>=100:
        return 1e-5

lr_scheduler = LearningRateScheduler(lr_sch)
lr_reducer = ReduceLROnPlateau(monitor='val_acc',factor=0.2,patience=5,
                               mode='max',min_lr=1e-3)
callbacks = [checkpoint,lr_scheduler,lr_reducer]
```

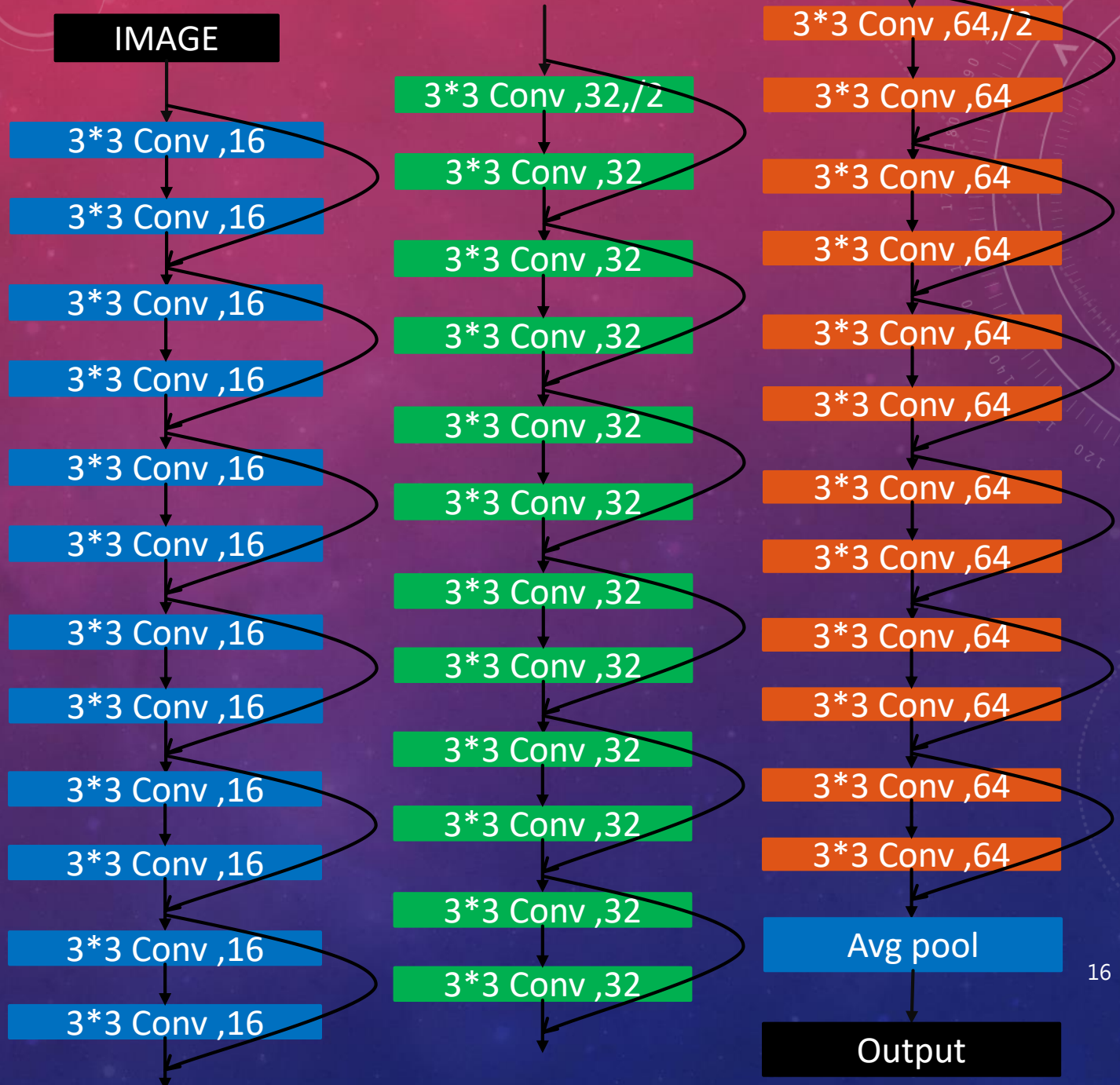
RESNET-CIFAR10(7/8)

Fitting data & print result:

```
###---餵資料---###
model.fit(x_train,y_train,batch_size=64,epochs=200,validation_data=(x_test,y_test),verbose=1,callbacks=callbacks)


###---印出結果---###
scores = model.evaluate(x_test,y_test,verbose=1)
print('Test loss:',scores[0])
print('Test accuracy:',scores[1])
```

RESNET CIFAR10(8/8)



參考資料

ResNet

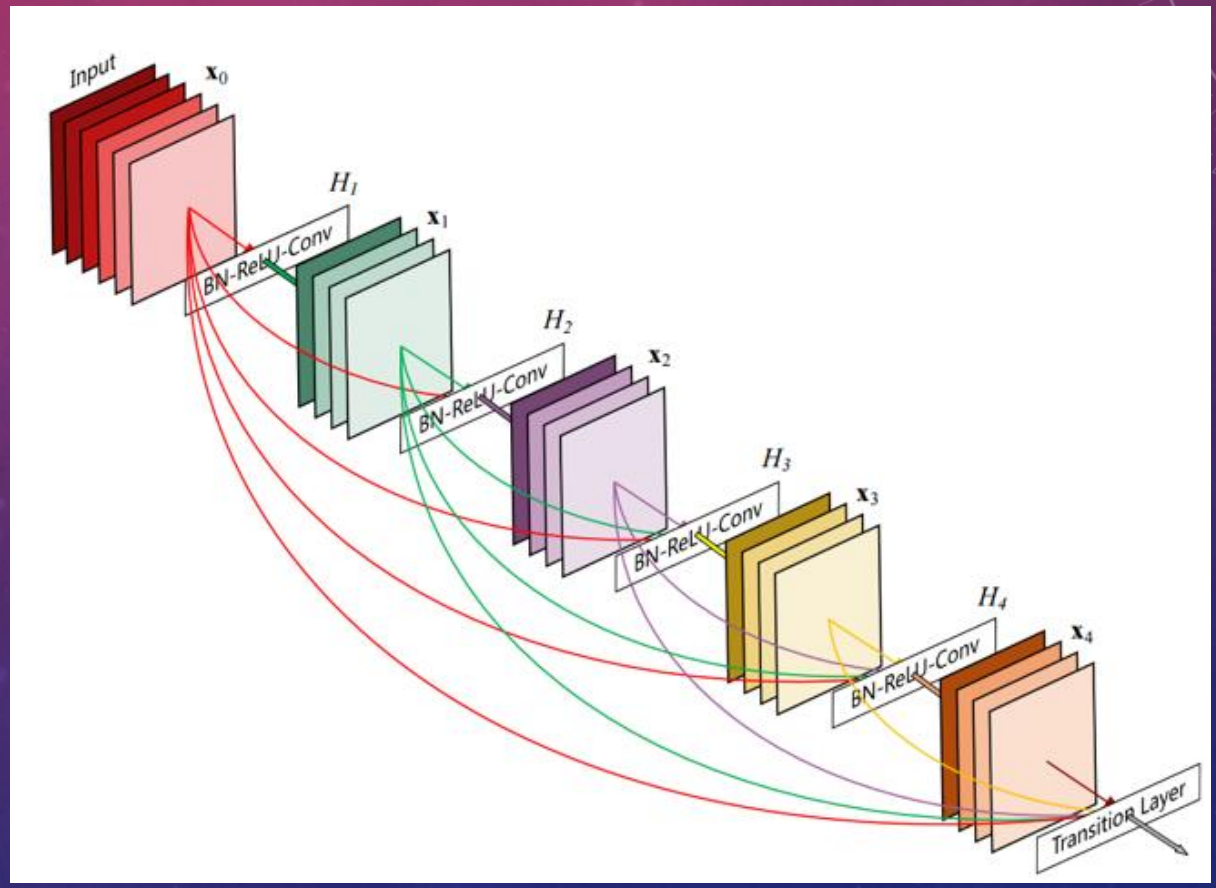
- http://bangqu.com/4Zy3e7.html?fbclid=IwAR1G-HIQdDmJmmHsEvLwUoKKnLJsSPRf7ffFv0Lx0sJjXrvja0nqrcTKxa4
- https://medium.com/@sidereal/cnns-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more-666091488df5
- https://arxiv.org/pdf/1512.03385.pdf
- https://codertw.com/%E4%BA%BA%E5%B7%A5%E6%99%BA%E6%85%A7/216/
- https://blog.csdn.net/rogerchen1983/article/details/79353972?utm_source=blogxgwz2&fbclid=IwAR3o9ifurlcbWtzDRbWfJkxCBbEDYaS70YghSxZVoYTw_eRBUIpdSffS8iY

Keras

- https://blog.csdn.net/tsyccnh/article/details/78865167
- https://keras-cn.readthedocs.io/en/latest/

DENSENET

DenseNet 是一種具有密集連接的卷積神經網絡。在該網絡中，任何兩層之間都有直接的連接，也就是說，網絡每一層的輸入都是前面所有層輸出的並集，而該層所學習的特徵圖也會被直接傳給其後面所有層作為輸入。

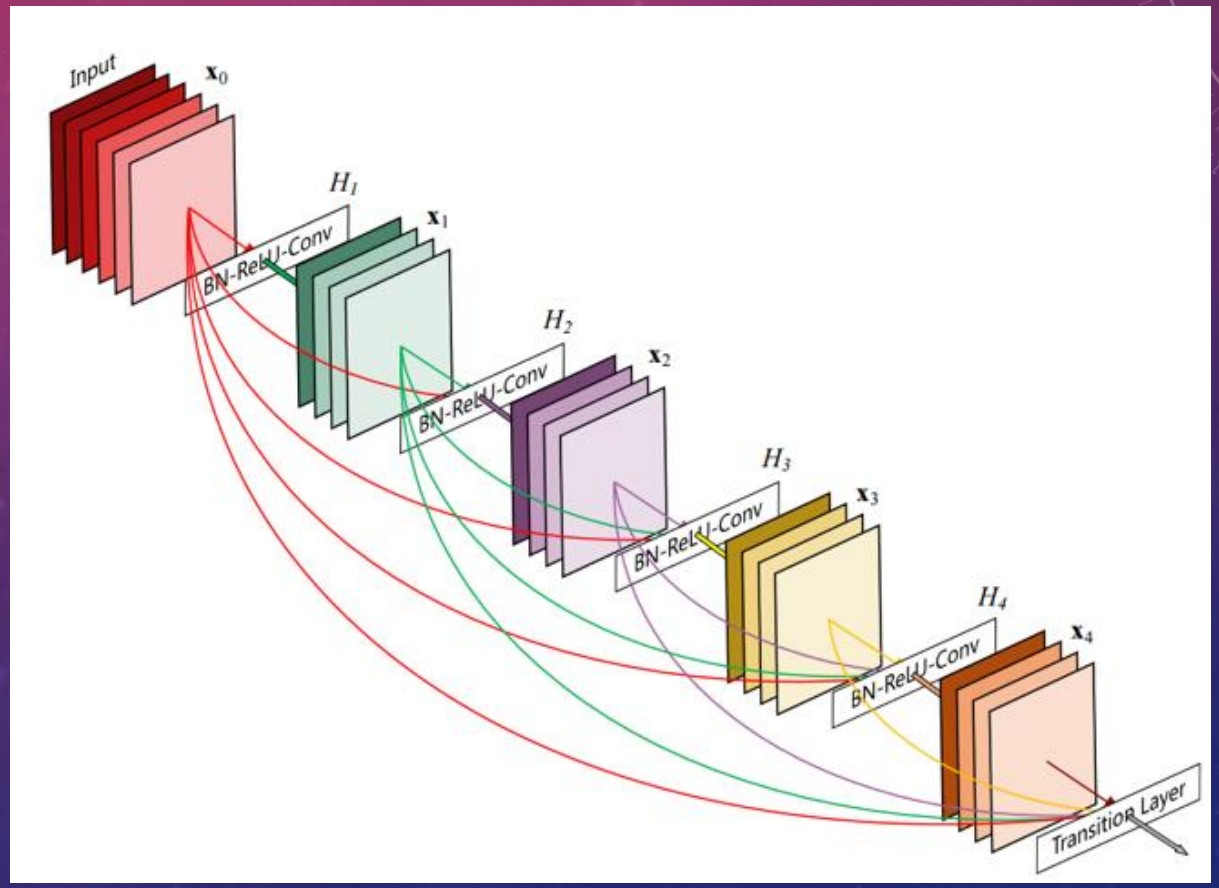


啟發

- **隨機深度網絡（Deep networks with stochastic depth）**在訓練過程中的每一步都隨機地「扔掉」（drop）一些層，可以顯著的提高 ResNet 的泛化性能。
  1. 神經網絡其實並不一定要是一個遞進層級結構，也就是說網絡中的某一層可以不僅僅依賴於緊鄰的上一層的特徵，而可以依賴於更前面層學習的特徵。
  2. 隨機扔掉很多層也不會破壞算法的收斂，說明了 ResNet 具有比較明顯的冗餘性，網絡中的每一層都只提取了很少的特徵（即所謂的殘差）。實際上，將訓練好的 ResNet 隨機的去掉幾層，對網絡的預測結果也不會產生太大的影響。

DENSENET

- 讓網絡中的每一層都直接與其前面層相連，實現特徵的重複利用；同時把網絡的每一層設計得特別「窄」，只學習非常少的特徵圖，達到降低冗餘性的目的。
- 第一點是第二點的前提，沒有密集連接，不可能把網絡設計得太窄的，否則訓練會出現 under-fitting 現象



優點

1. **省參數**。在 ImageNet 分類數據集上達到同樣的準確率，DenseNet 所需的參數量不到 ResNet 的一半。
2. **省計算**。達到與 ResNet 相當的精度，DenseNet 所需的計算量也只有 ResNet 的一半左右。
3. **抗過擬合**。DenseNet 具有非常好的抗過擬合性能，尤其適合於訓練數據相對匱乏的應用。

THANK YOU