DENSENET

張家瑋 博士

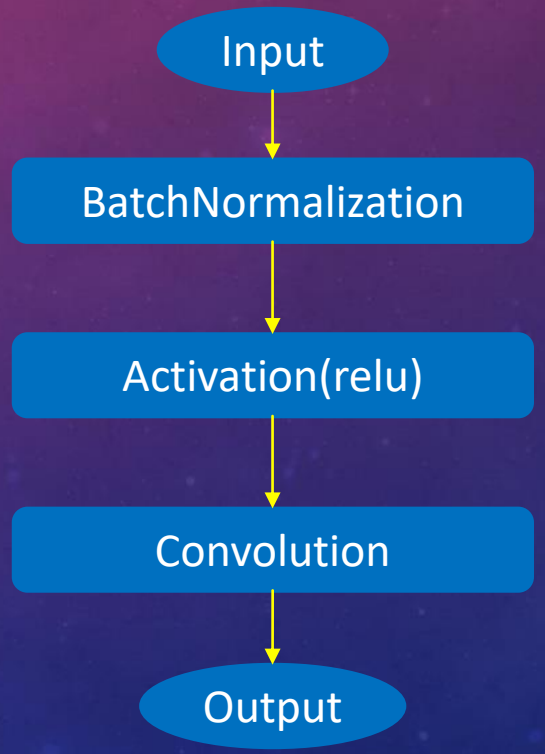
助理教授

國立臺中科技大學資訊工程系

DENSE CONVOLUTIONAL NEURAL NETWORK

- Dense connectivity
  $X_l = H_l([X_0, X_1, X_2 \ldots, X_{l-1}])$
- Composite Function(transition layer)

Composite Function:

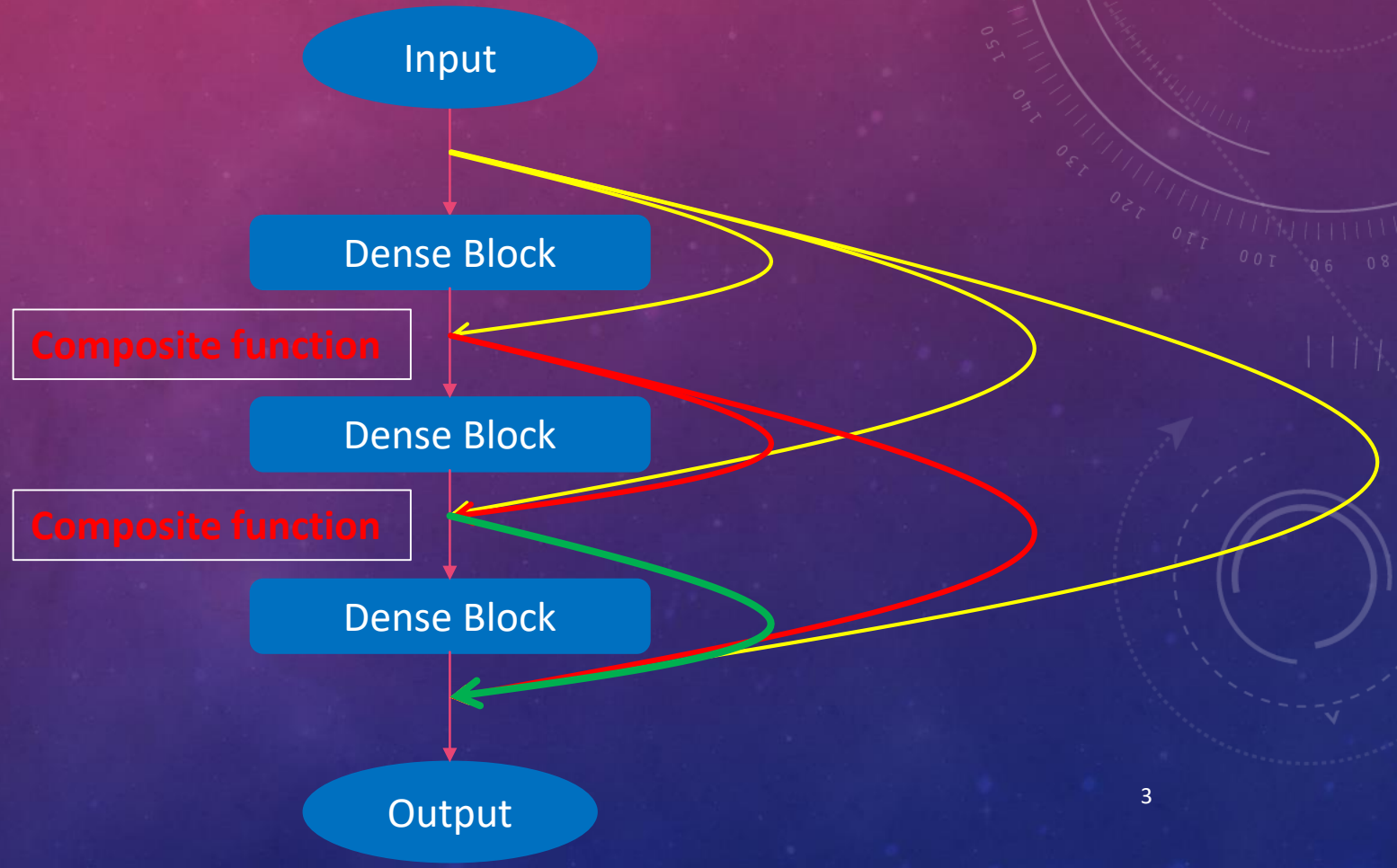


DENSE BLOCK

- L層有L(L+1)/2連結

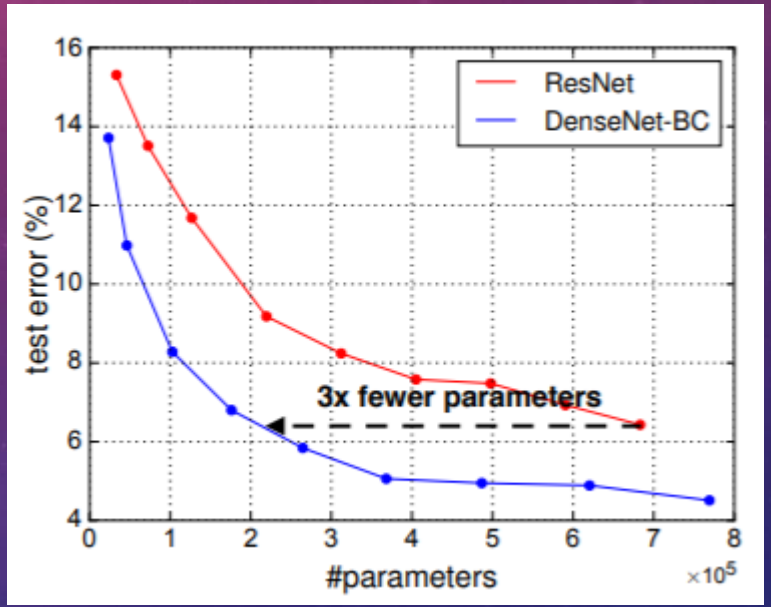
1+2+3+…+L = L(L+1)/2

Dense Block :



DENSENET VS RESNET

- Adding vs Concatenation
- ResNet利用identity mapping
- DenseNet使feature更完美利用



截至https://arxiv.org/pdf/1608.06993.pdf

範例

IMPLEMENT ON CIFAR10 WITH KERAS

DENSENET-CIFAR10(1/8)

引入所需套件:

```
import keras
from keras.datasets import cifar10                                  #本次使用資料集
from keras.models import Model                                      #keras建立模型方式
from keras.layers import Dense,Conv2D,Input,AveragePooling2D        #Dense建立一般層、Conv2D建立CNN層
from keras.layers import Flatten,MaxPooling2D,BatchNormalization    #Flatten使CNN過渡到一般層
from keras.layers import Concatenate,Activation                     #利用Concatenate結合feature_map
from keras.optimizers import Adam                                   #優化器
from keras.callbacks import ModelCheckpoint,ReduceLROnPlateau       #可選擇使用，可參考ResNet.py
from keras.utils import to_categorical                              #做one-hot encoding用
```

DENSENET-CIFAR10(2/8)

載入資料集:

```
###---載入資料集---###

(x_train,y_train),(x_test,y_test) = cifar10.load_data()
```

正規化:

```
###---normalization---###

x_train = x_train/255
x_test = x_test/255
```

One-hot encoding:

```
###---one-hot encoding---###

y_train = to_categorical(y_train,10)
y_test = to_categorical(y_test,10)
```

DENSENET-CIFAR10(3/8)

定義Dense Block:

```
###---Dense Block---###
def add_denseblock(inputs,filters):
    x = BatchNormalization()(inputs)
    x = Activation('relu')(x)
    conv = Conv2D(filters,(3,3),padding='same')(x)
    conca = Concatenate(axis=-1)([inputs,conv])      #與ResNet不同之處，ResNet使用add方式將輸入合併
    return conca
```

定義Transition Layer(composite function):

```
###---Transition---###
def add_transition(inputs,filters):
    x = BatchNormalization()(inputs)
    x = Activation('relu')(x)
    x = Conv2D(filters,(1,1),padding='same')(x)
    avg = AveragePooling2D(pool_size=(2,2))(x)
    return avg
```

DENSENET-CIFAR10(4/8)

定義輸出層：

```
###---Output_Layer---###
def output_layer(inputs):
    x = BatchNormalization()(inputs)
    x = Activation('relu')(x)
    avg = AveragePooling2D(pool_size=(2,2))(x)
    y = Flatten()(avg)
    output = Dense(10,activation='softmax')(y)
    return output
```

DENSENET-CIFAR10(5/8)

建立模型:

```
###---建立模型---###

inputs = Input(shape=(32,32,3))
first_conv = Conv2D(32,(3,3),padding='same')(inputs)  #第一層CNN

for i in range(3):                                    #建立四層dense_block中間有三層transition_layer
    if i == 0:
        block = add_denseblock(first_conv,32)
    else:
        block = add_denseblock(transition,32)
    transition = add_transition(block,32)
block = add_denseblock(transition,32)                 #第四層dense_block
output = output_layer(block)
model = Model(inputs=[inputs], outputs=[output])
model.summary()
```

DENSENET-CIFAR10(6/8)

調整參數:

```
###---調整參數---###

model.compile(loss='categorical_crossentropy',
              optimizer=Adam(),
              metrics=['accuracy'])
model.fit(x_train,y_train,batch_size=64,epochs=200,verbose=1,
          validation_data=(x_test,y_test))
```

印出結果:

```
###---印出結果---###

score = model.evaluate(x_test,y_test,verbose=1)
print('Test loss:',score[0])
print('Test accuracy:',score[1])

###---Done---###
```

THANK YOU