

The background is a gradient from dark red at the top to dark blue at the bottom, with a starry space-like texture. Overlaid on the left side are several concentric circles and arcs, some with degree markings (40, 150, 160, 170, 180, 190, 200, 210, 220, 230, 240, 250, 260) and arrows, suggesting a technical or scientific theme.

DENSENET

張家瑋 博士

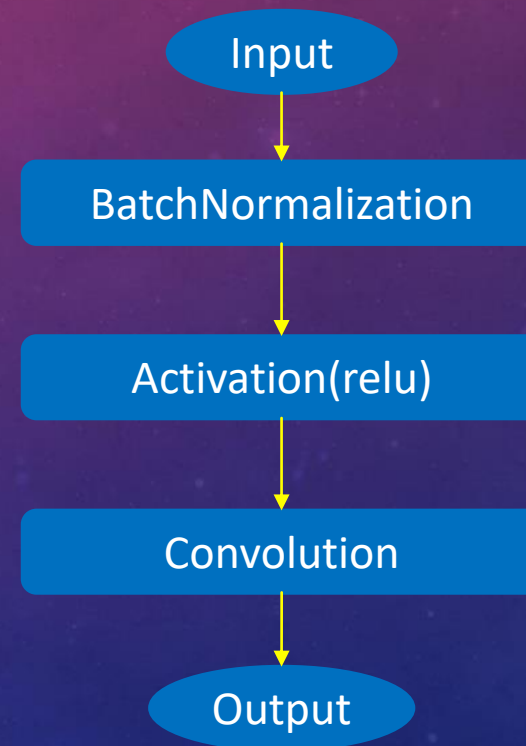
助理教授

國立臺中科技大學資訊工程系

DENSE CONVOLUTIONAL NEURAL NETWORK

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

Composite Function:

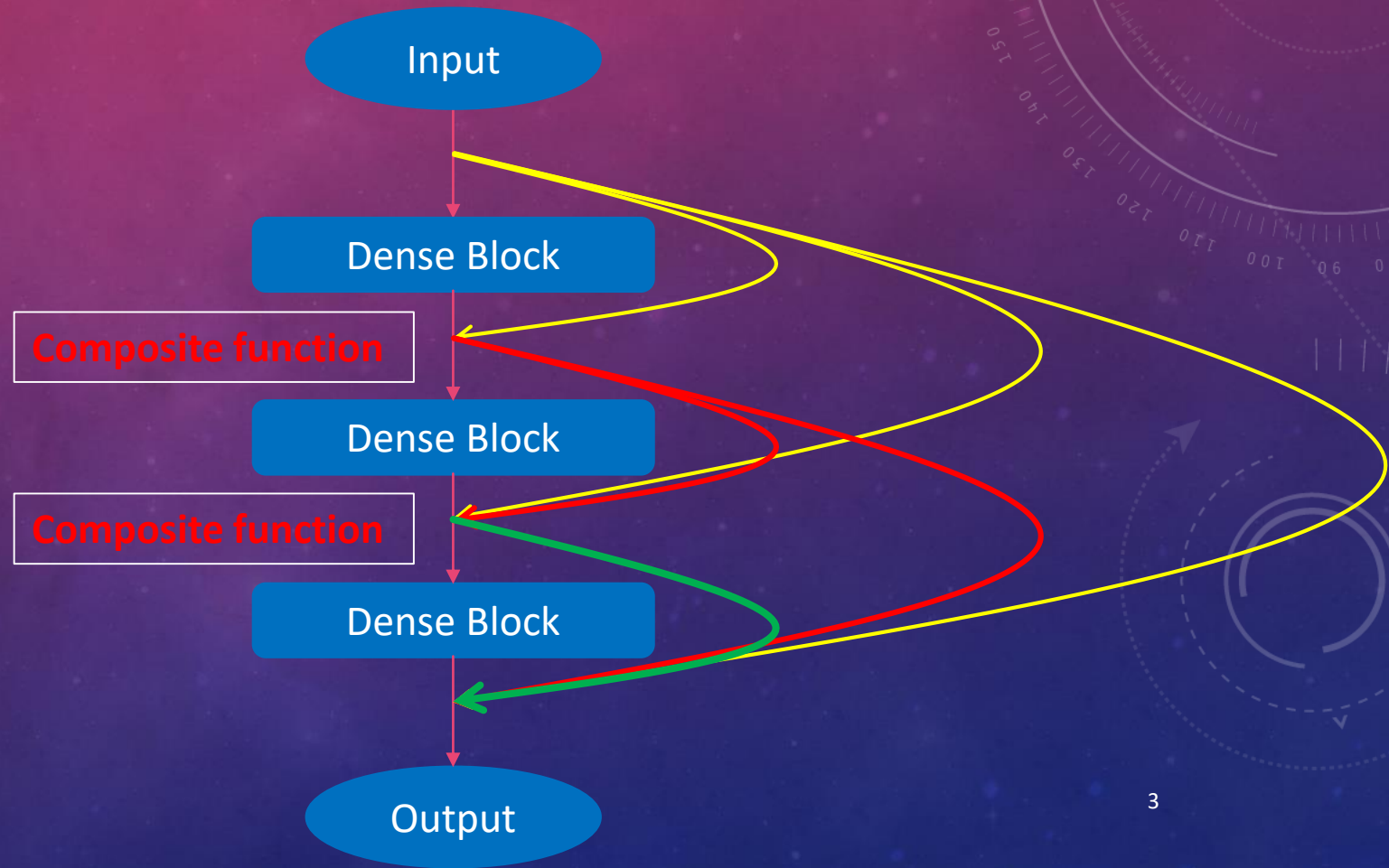


DENSE BLOCK

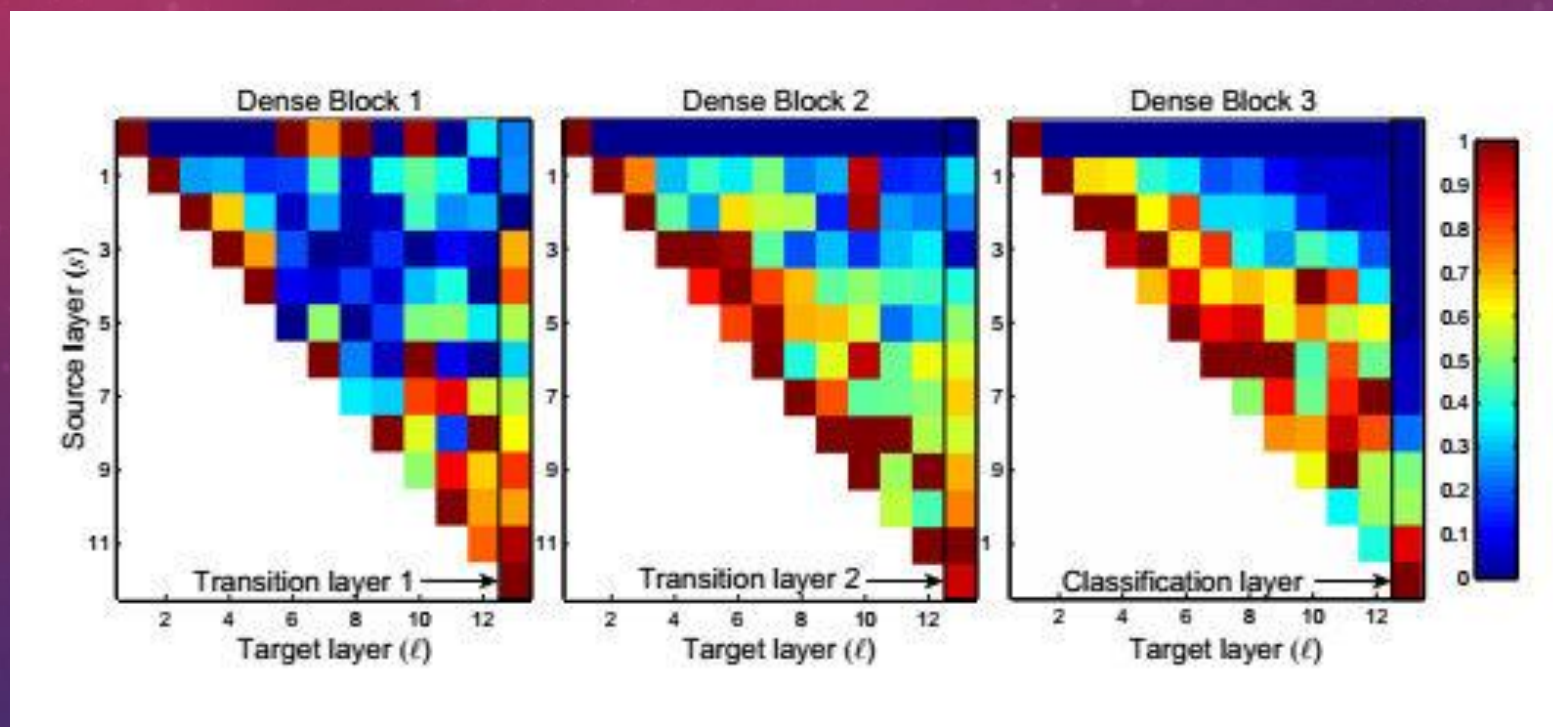
- L層有 $L(L+1)/2$ 連結

$$1+2+3+\dots+L = L(L+1)/2$$

Dense Block :

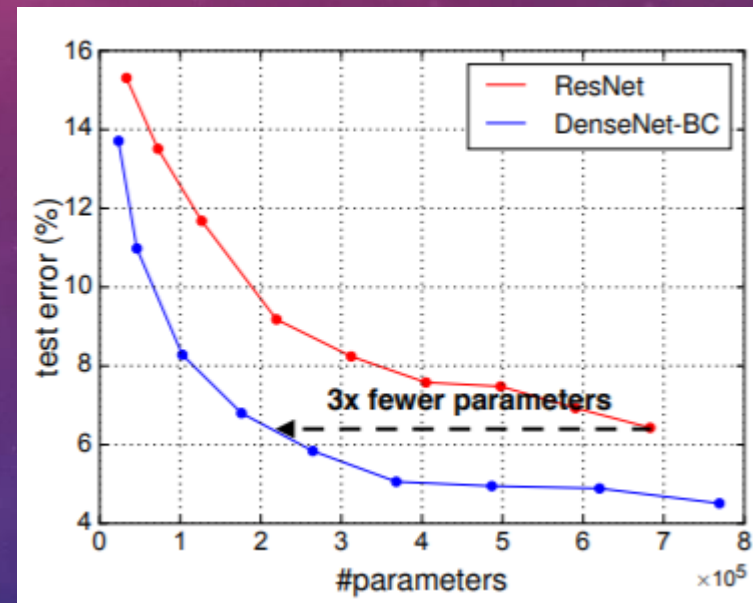


DENSENET 優勢



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>

The background is a gradient from dark red at the top to dark blue at the bottom, speckled with white dots. Overlaid on the left side are several concentric circles and a radial scale. The scale is a large arc with tick marks and numbers ranging from 40 to 260 in increments of 10. Smaller concentric circles with arrows are scattered across the image, some solid and some dashed.

範例

IMPLEMENT ON CIFAR10 WITH KERAS

DENSENET-CIFAR10(1/8)

引入所需套件:

```
import keras
from keras.datasets import cifar10
from keras.models import Model
from keras.layers import Dense, Conv2D, Input, AveragePooling2D
from keras.layers import Flatten, MaxPooling2D, BatchNormalization
from keras.layers import Concatenate, Activation
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
from keras.utils import to_categorical
```

#本次使用資料集
#keras 建立模型方式
#Dense 建立一般層、Conv2D 建立CNN層
#Flatten 使CNN過渡到一般層
#利用Concatenate 結合feature_map
#優化器
#可選擇使用，可參考ResNet.py
#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])
    return conca
```

#與ResNet不同之處，ResNet使用add方式將輸入合併

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