Keras

Simple. Flexible. Powerful.

2020/10/21

草莓甜甜圈

陳星文 陳薔 劉益程

目錄

1. 簡介 & 安裝
2. 神經網路 (NN)
3. 應用於電腦視覺 (CNN)
4. 生成式深度學習 (Deep dream)
5. 補充

簡介 https://www.chainnews.com/zh-hant/articles/916826219156.htm

- 快速又簡單的實現深度神經網路，使用者只需要加入正確的參數
- 2015年由Google工程師**François Chollet**發明
- 支援不同運算後台，例如TensorFlow、Microsoft Cognitive Toolkit、Theano或PlaidML (最主流的後台是tensorflow)

- 2019年底，**Keras v2.3.0** 正式發佈，並且不再更新，是最後一個支援Tensorflow之外後端的keras
- **應開始改用TensorFlow 2.0 和 tf.keras!!**

安裝

舊的keras(使用Tensorflow當後端)

```
> pip install tensorflow

> pip install Keras
```

```
import keras
keras.__version__
```

```
'2.2.4'
```

tf.keras(建議使用!!)

```
# Requires the latest pip
pip install --upgrade pip

# Current stable release for CPU and GPU
pip install tensorflow
```

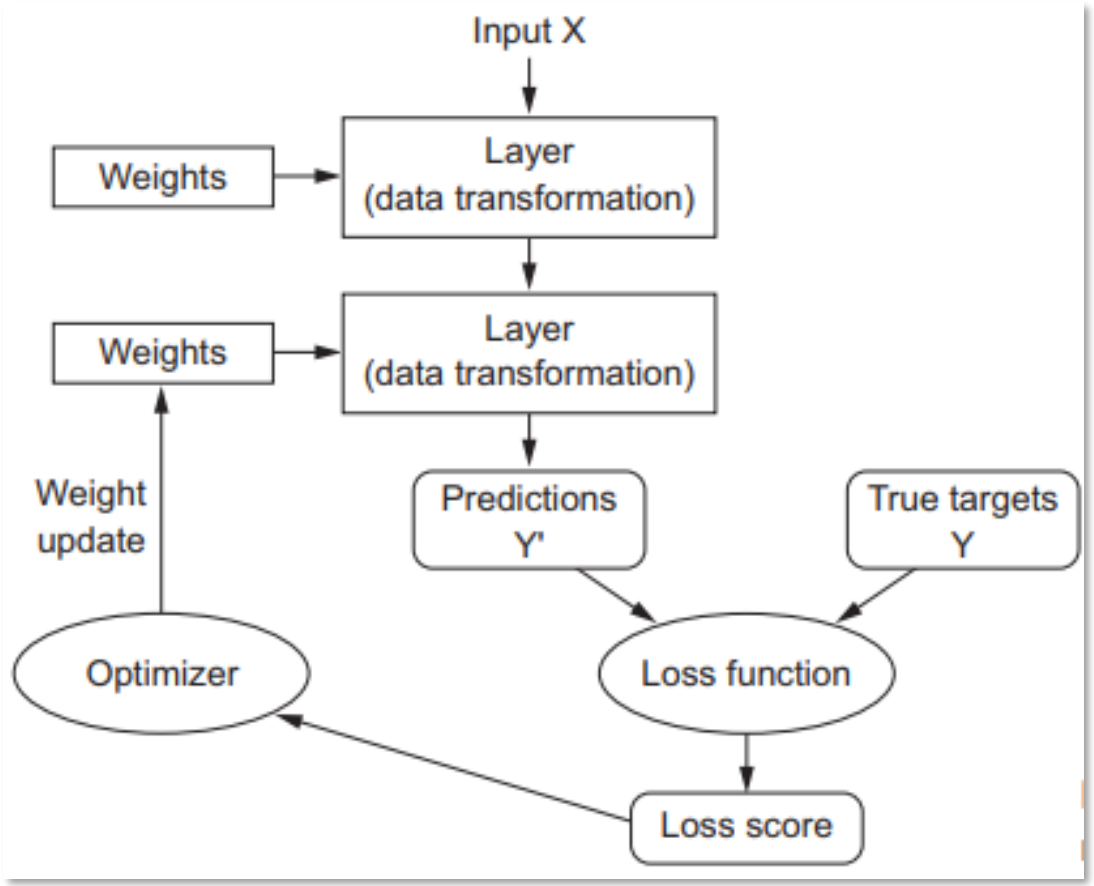
```
In [1]: import tensorflow.keras as keras
        keras.__version__

Out[1]: '2.4.0'
```

1. 簡介 & 安裝
2. **神經網路(NN)**
3. 應用於電腦視覺 (CNN)
4. 生成式深度學習 (Deep dream)
5. 補充

Neural Network

要訓練一個神經網絡，通常都包含：

(1)Layers
(2)Input data and targets
(3)Loss function &Optimizer



(1)Layers

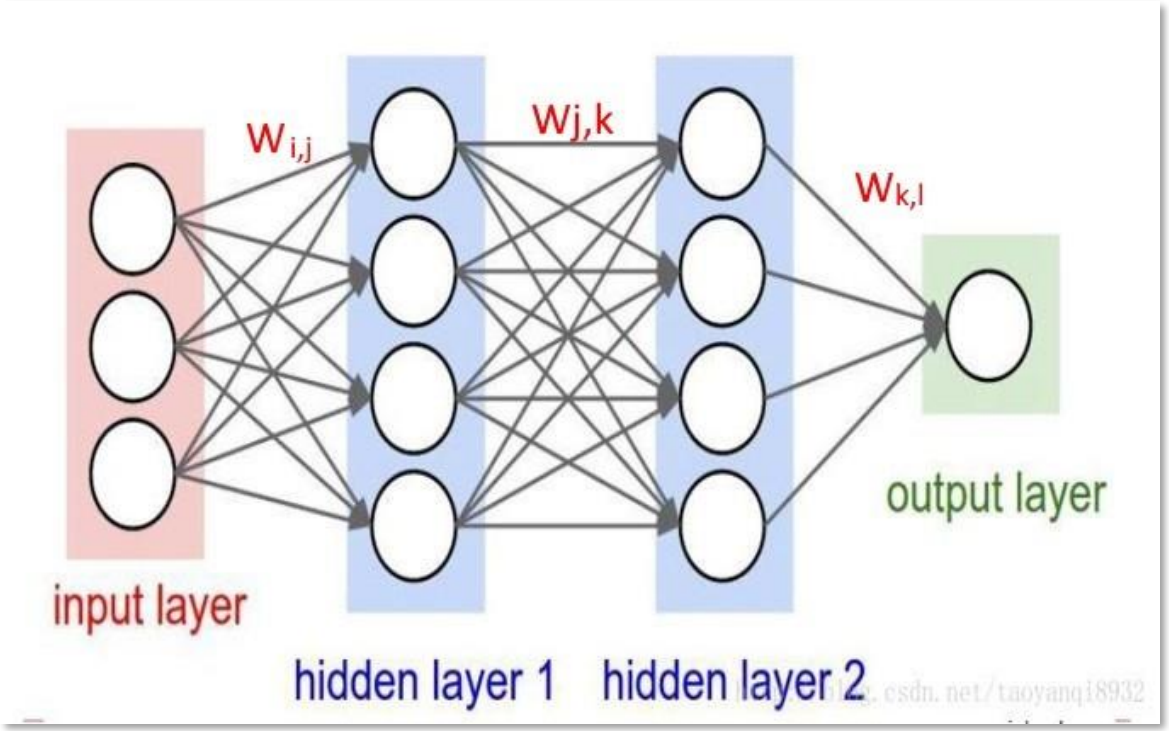
Layer(層)是神經網絡的基本結構。Input Layer就是接收信號的神經元，Hidden Layer就是隱藏層，而Output Layer就是做出反應的輸出層，而各神經元傳導的力量大小，稱為**權重(Weight,** 以W表示)，也就是模型要求解的參數。

Hidden Layer及Output Layer上每一個節點(圓圈)的值等於上一層所有節點的加權總和，如下式

$$y_j = \sum x_{i,j} * W_{i,j}$$

但此式只能進行線性切割，因此通常會做以下處理:

- 將上述的公式乘以一個非線性函數，稱為Activation Function。依據問題挑選適合的函常見的如sigmoid、softmax、relu等等。
- 使用更多層 Hidden Layer。



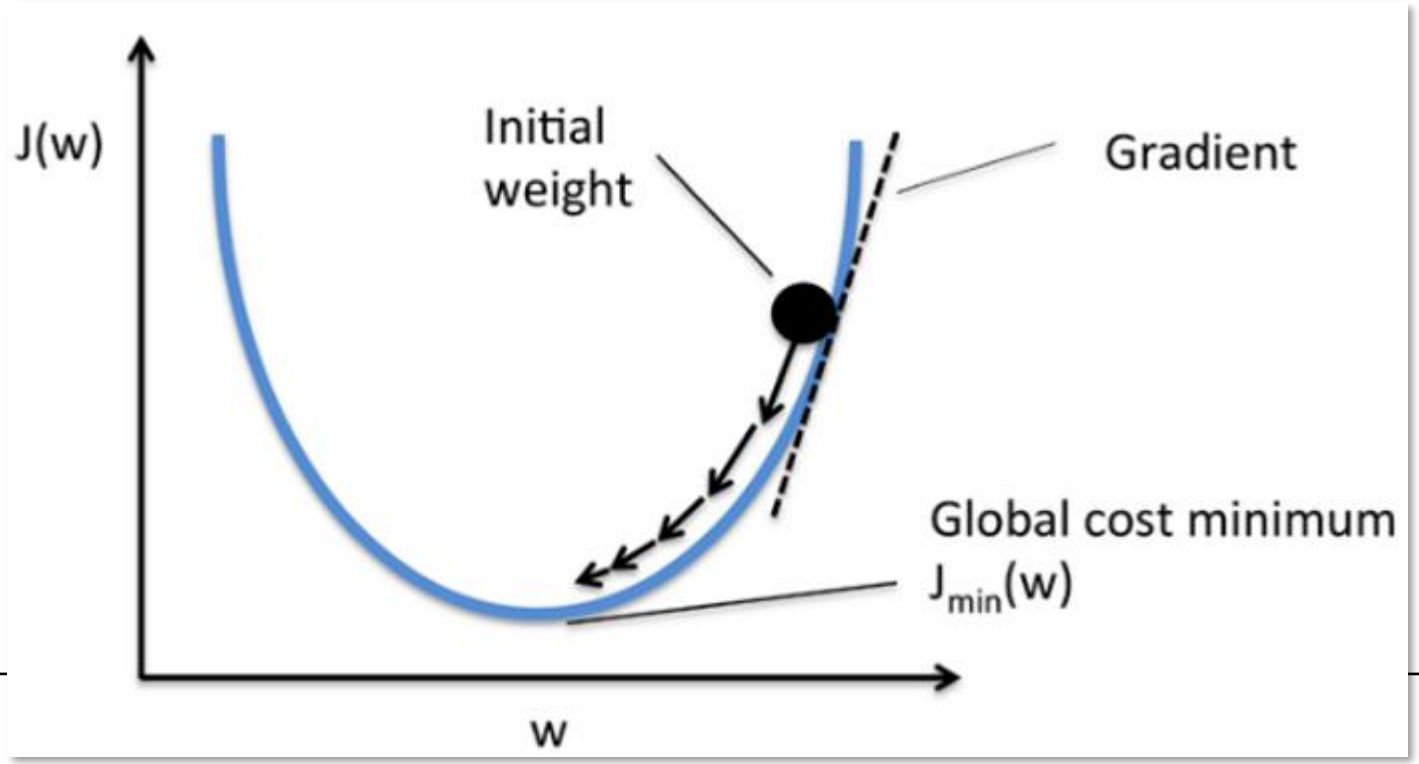
(2)Input data & Targets

- **Input data:** 用來學習的資料
- **Targets:** 每筆資料的答案

(3)Loss functions & optimizers

Loss function: 用來評估模型的好壞，在訓練過程中要最小化。依據不同問題也會使用不同的loss function。

Optimizer: 決定一個神經網路如何更新權重(基於Loss function)，一般常用的優化函數是梯度下降(Gradient Descent)。



Keras

用Keras實做一個基本的神經網絡!

models.Sequential()

- 用Keras實做一個基本的神經網絡!

```
from keras import models
from keras import layers
network = models.Sequential()
```

- 先從keras import models及layers

- Sequential()用來建立序貫(Sequential)模型，是結構較簡單的模型，單輸入單輸出，且層之間只有相鄰關係。

add(), layers.Dense()

add(): 增加Layers

```
network.add(layers.Dense(512, activation='relu', input_shape=(28 * 28,)))
network.add(layers.Dense(10, activation='softmax'))
```

- Dense()中512代表輸出的節點數。activation代表activation function的種類。input_shape是輸入層(Input layer)代表輸入的維度。
- 此例中的Input layer需input 28*28的資料，激活函數是relu，然後輸出512個節點到下一層。
- 輸出層最後輸出10個節點，激活函數是softmax。

compile()

compile(): optimizer決定優化器，loss決定損失函數，metrics則是評估模型的準則。

```
network.compile(optimizer='rmsprop',
                loss='categorical_crossentropy',
                metrics=['accuracy'])
```

- 此例中的優化器使用rmsprop(Root-Mean-Square prop)。
- loss function使用cross entropy (前面的categorical代表response是多類別的變數)。
- 評估模型使用Accuracy(準確率)。

fit()

fit(): 用來訓練模型。

```
network.fit(train_x, train_y, epochs=5, batch_size=128)
```

- 先分別放Training Set的x和y。
- epochs為訓練次數。
- batch_size是對總樣本數分組，每組包含的樣本數。
- 還有一些參數未在上式中，如shuffle可以隨機打亂數據，validation_split可以選擇百分之多少的資料做交叉驗證。

evaluate(), predict

evaluate(): 用來評估模型，會回傳loss和accuracy兩個數值。

```
test_loss, test_acc = network.evaluate(test_x, test_y)
```

predict(): 用來做預測，可以獲得輸入資料對應的輸出。

```
model.predict(test_x)
```

1 簡介 & 安裝

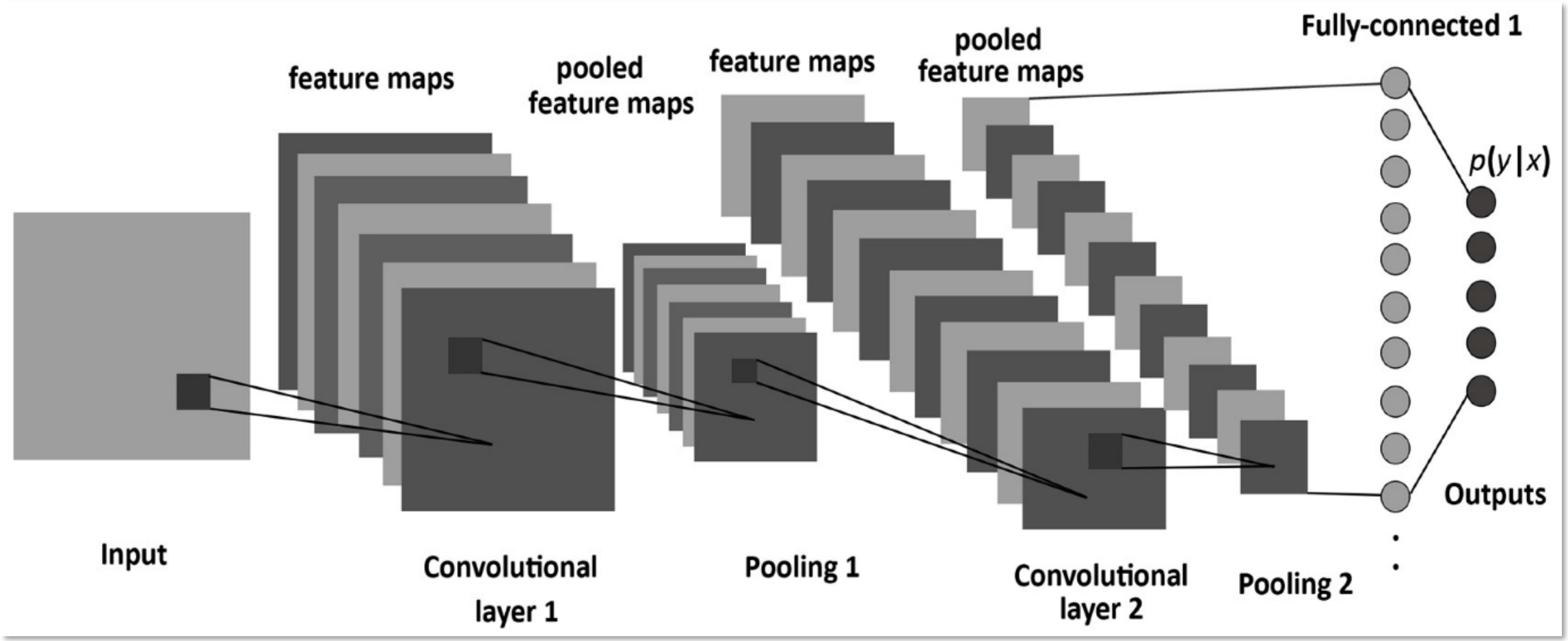
2 神經網路(NN)

3 **應用於電腦視覺 (CNN)**

4 生成式深度學習 (Deep dream)

5 補充

CNN(卷積神經網路)



範例資料集 https://www.kaggle.com/c/dogs-vs-cats/data

- 使用貓狗圖片辨識資料集 (單位: 張)

	Train data	Validation data
貓	1000	500
狗	1000	500

(1)使用生成器預處理

```
In [20]: from keras.preprocessing.image import ImageDataGenerator

         train_dir = 'train_data'
         validation_dir = 'validation_data'

         dataGen = ImageDataGenerator(rescale = 1.0/255)
```

ImageDataGenerator(): 自動將資料做特定處理

- rescale: 把所有數字縮放

(1)使用生成器預處理

```
train_generator = dataGen.flow_from_directory(train_dir,
                                              target_size = (150, 150),
                                              batch_size = 20,
                                              class_mode = 'binary')
validation_generator = dataGen.flow_from_directory(validation_dir,
                                                   target_size = (150, 150),
                                                   batch_size = 20,
                                                   class_mode = 'binary')
```

```
Found 2000 images belonging to 2 classes.
Found 1000 images belonging to 2 classes.
```

flow_from_directory(): 自動將特定資料夾的圖片處理並label

- directory: 目標資料夾，**每類資料**都必須包含在一個**子資料夾**裡面
- target_size: 將圖像resize
- batch_size: 一次產生多少張
- class_mode: label的形式，可以是"categorical", "binary", "sparse"或"None"

(2)使用一個單純的CNN

```
# add
from keras import layers
from keras import models

model_CNN = models.Sequential()
model_CNN.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Conv2D(128, (3, 3), activation='relu'))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Conv2D(128, (3, 3), activation='relu'))
model_CNN.add(layers.MaxPooling2D((2, 2)))
model_CNN.add(layers.Flatten())
model_CNN.add(layers.Dense(512, activation='relu'))
model_CNN.add(layers.Dense(1, activation='sigmoid'))
```

layers.Conv2D(): 卷積層，2D指移動方向

- filters: 卷積核數量
- kernel_size: kernel大小
- strides: kernel每次移動步數，預設1
- padding: 邊邊填充方法，預設'valid'

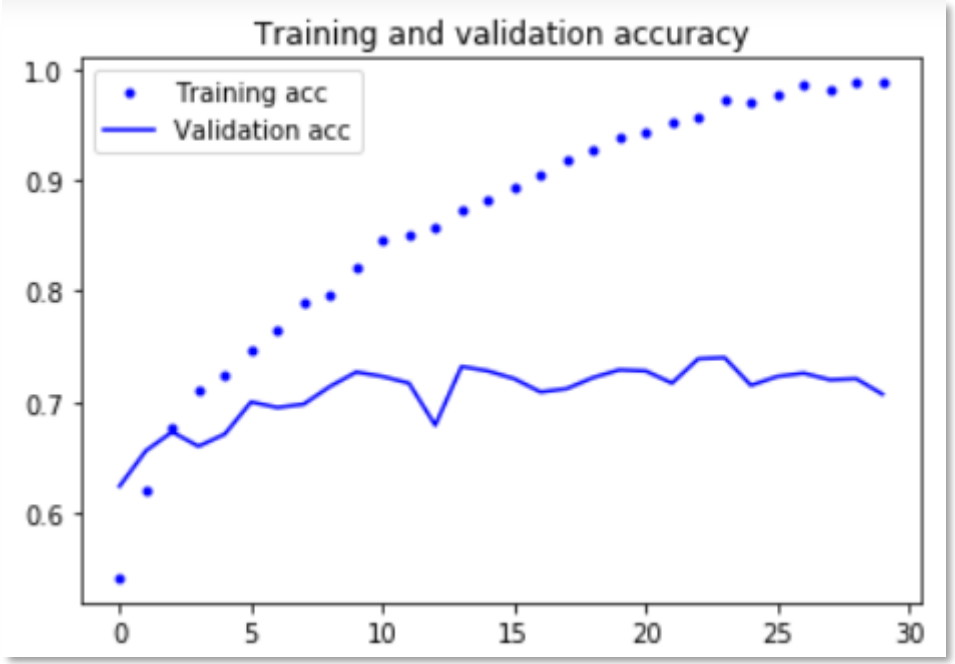
layers.Maxpooling2D(): 池化層，放大重要特徵

- pool_size: 做一次maxpooling的大小

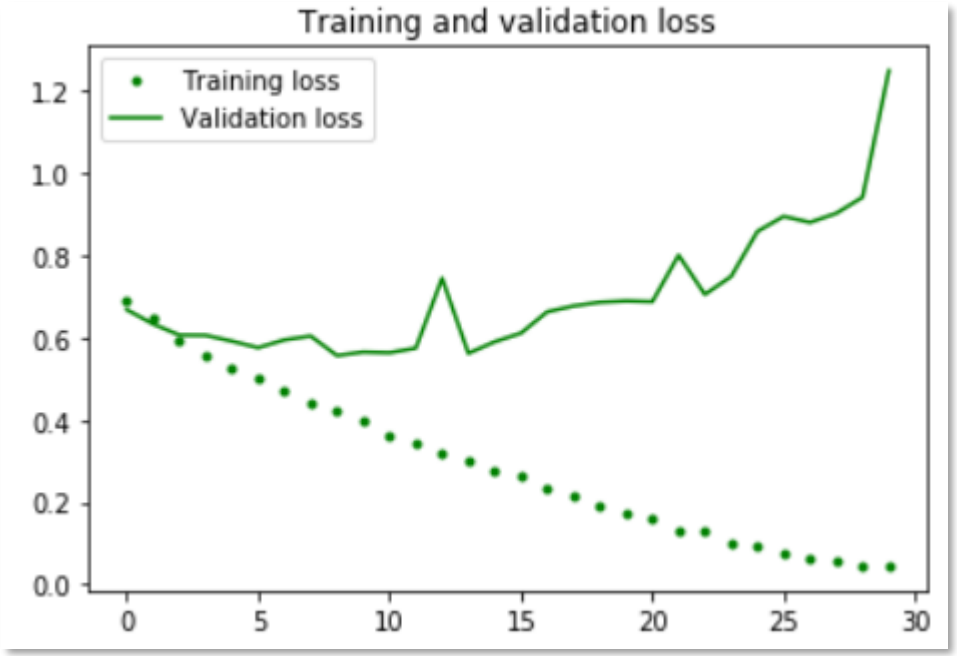
(2)使用一個單純的CNN

結果:

X = epoch, Y = acc



X = epoch, Y = loss



(3)使用圖像增強(Data Augumentation), 並加入Dropout層

原因: 資料集太小，需避免overfitting

(3)使用圖像增強(Data Augumentation), 並加入Dropout層

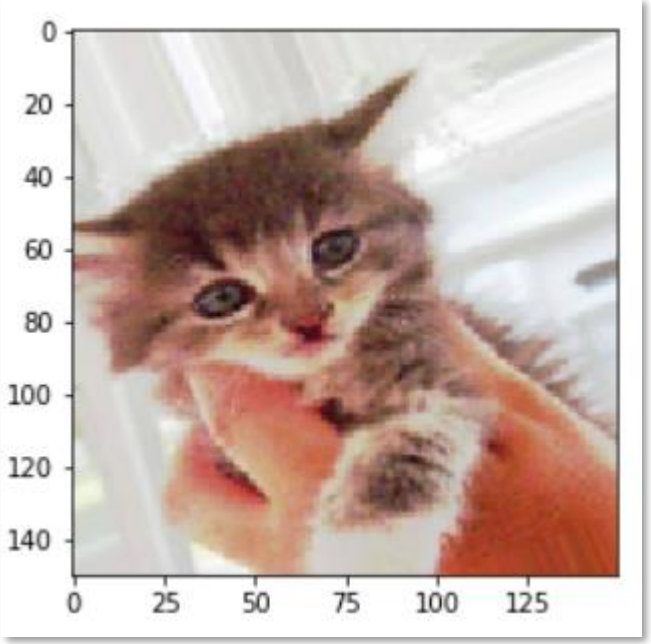
```
In [67]: datagen_augmetation = ImageDataGenerator(rotation_range = 40,
                                                   width_shift_range = 0.2,
                                                   height_shift_range = 0.2,
                                                   shear_range = 0.2,
                                                   zoom_range = 0.2,
                                                   horizontal_flip = True,
                                                   fill_mode = 'nearest')
```

ImageDataGenerator(): 自動將資料做特定處理

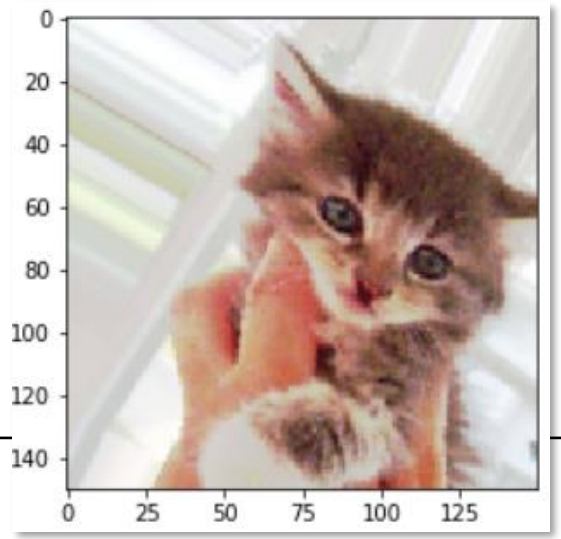
- rotation_range: 0~180，隨機旋轉時的角度範圍
- width_shift_range / height_shift_range : 水平/垂直方向平移的範圍
- shear_range: 固定某座標值時，另一個座標值縮放的比例
- zoom_range: 隨機縮放的範圍
- horizontal_flip: 50%機率水平翻轉
- fill_mode: 填充像素值的方法

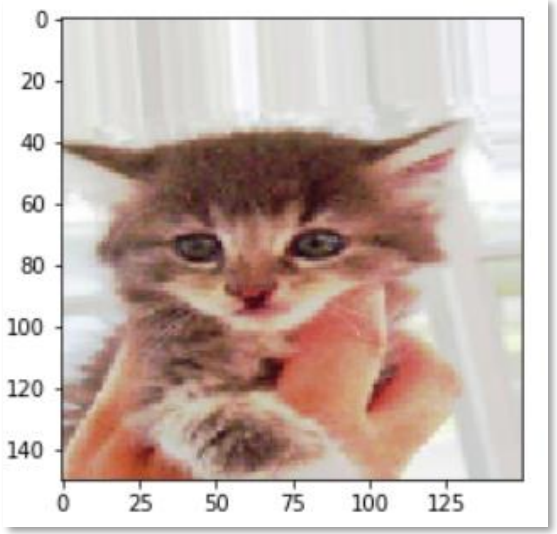
(3)使用圖像增強(Data Augumentation), 並加入Dropout層

Original:



Output





Keras

(3)使用圖像增強(Data Augumentation), 並加入Dropout層

```
model_CNN_dropout = models.Sequential()
model_CNN_dropout.add(layers.Conv2D(32, (3, 3), activation='relu',
                        input_shape=(150, 150, 3)))
model_CNN_dropout.add(layers.MaxPooling2D((2, 2)))
model_CNN_dropout.add(layers.Conv2D(64, (3, 3), activation='relu'))
model_CNN_dropout.add(layers.MaxPooling2D((2, 2)))
model_CNN_dropout.add(layers.Conv2D(128, (3, 3), activation='relu'))
model_CNN_dropout.add(layers.MaxPooling2D((2, 2)))
model_CNN_dropout.add(layers.Conv2D(128, (3, 3), activation='relu'))
model_CNN_dropout.add(layers.MaxPooling2D((2, 2)))

model_CNN_dropout.add(layers.Flatten())
model_CNN_dropout.add(layers.Dropout(0.5)) # flatten之後加入dropout層
model_CNN_dropout.add(layers.Dense(512, activation='relu'))
model_CNN_dropout.add(layers.Dense(1, activation='sigmoid'))

model.compile(loss='binary_crossentropy',
              optimizer=optimizers.RMSprop(lr=1e-4),
              metrics=['acc'])
```

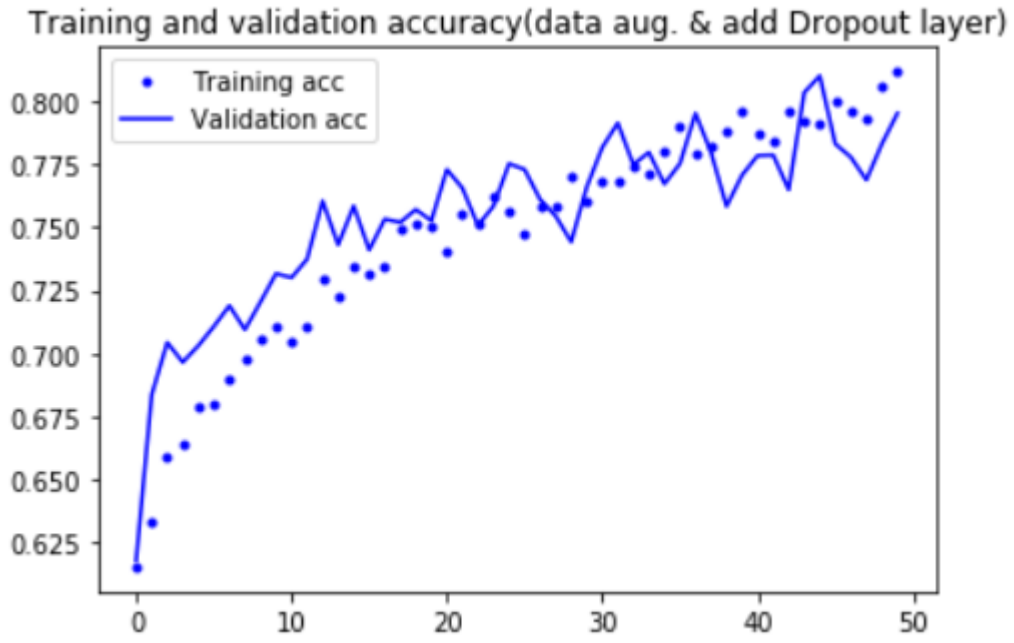
layers.Dropout():避免overfitting

- rate: 隨機選多少比例的數字會變成0

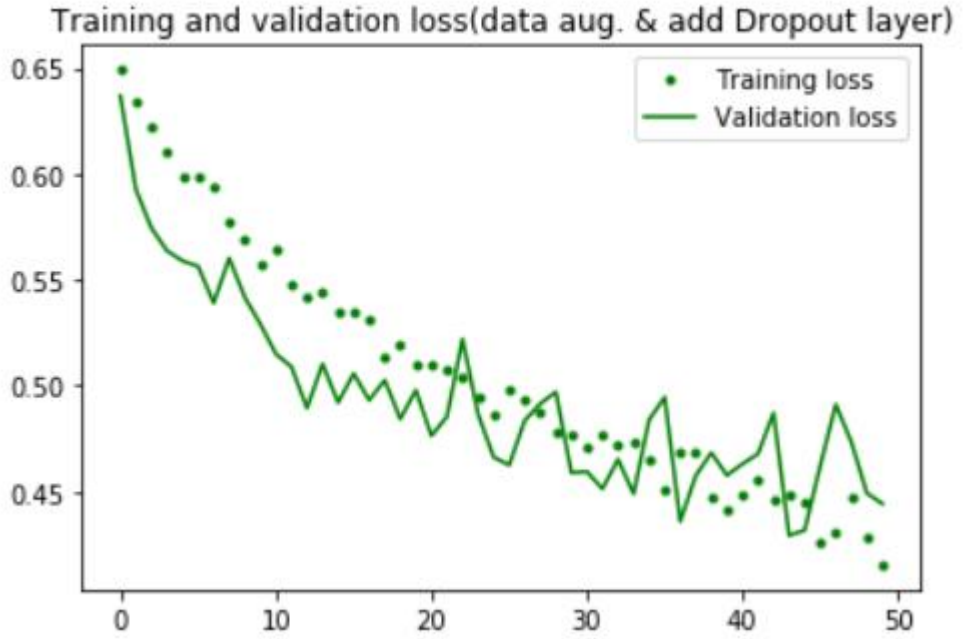
(3)使用圖像增強(Data Augumentation), 並加入Dropout層

結果:

`X = epoch, Y = acc`



`X = epoch, Y = loss`



(4)視覺化訓練過程

```
In [115]: # 將模型實例化, 使其可以input一個圖片張量, output模型前8層layer的結果
          from keras import models

          layer_outputs = [layer.output for layer in model_CNN_dropout.layers[:8]]

          # Model()可以允許一個input, 多個output, 例如activation_model的output就是8個numpy arrary
          activation_model = models.Model(inputs = model_CNN_dropout.input, outputs = layer_outputs)
```

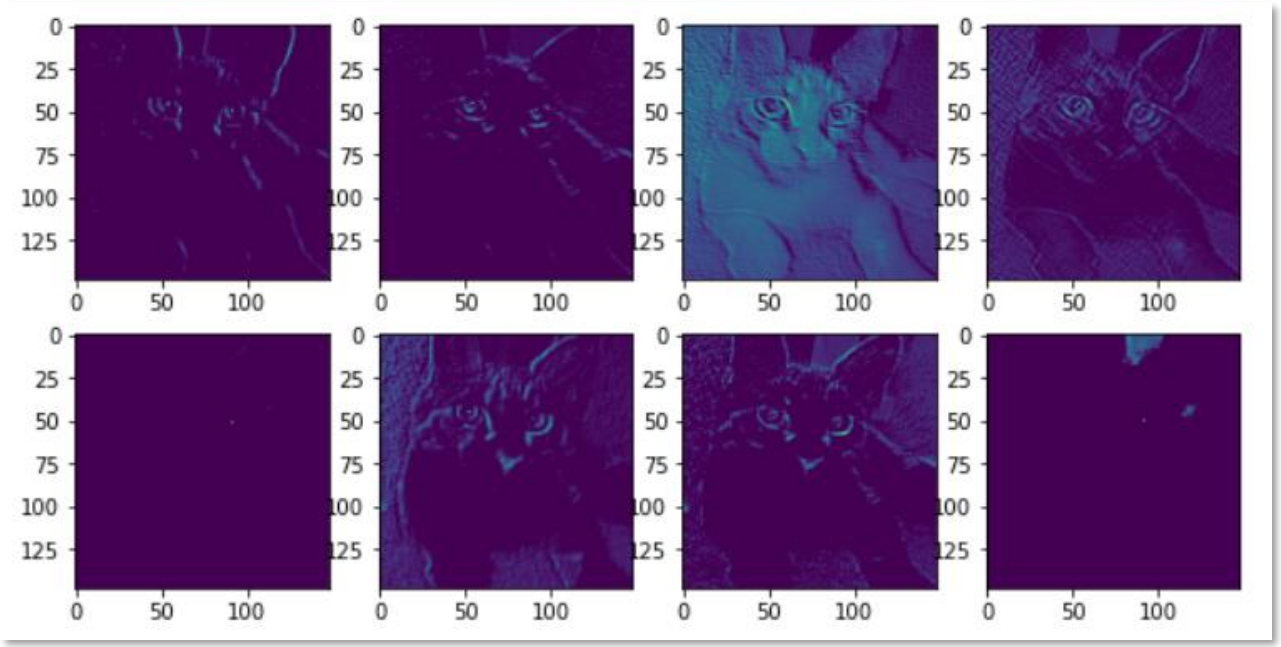
Model(): 可自訂input和output, 允許多個output

一張圖

8層layer分別的輸出結果

(4)視覺化訓練過程

畫出第一層layer的前8個輸出結果:



(第一層layer總共會輸出32個)

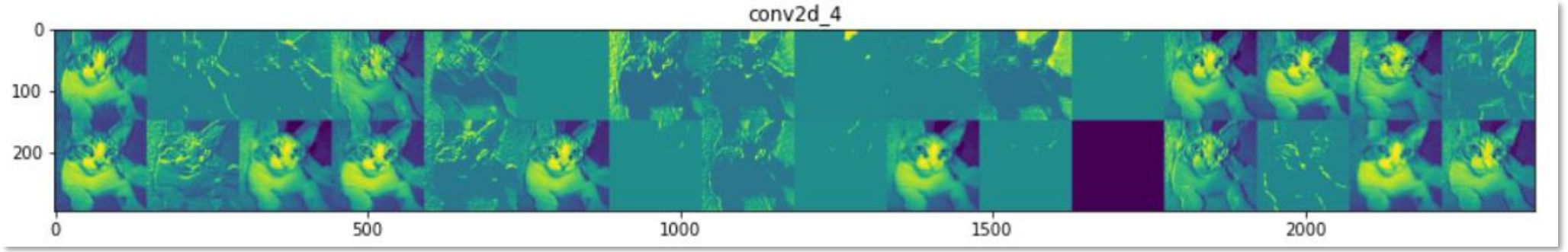
```
activations = activation_model.predict(img_tensor) # 將圖片放入模型

first_layer_activation = activations[0]
```

Keras

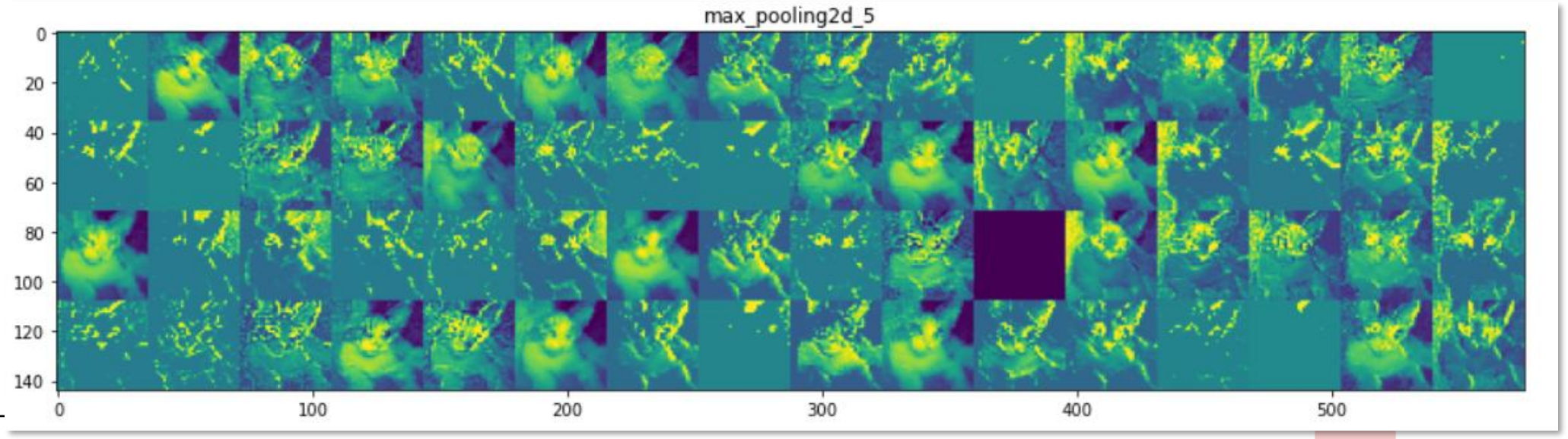
(4)視覺化訓練過程

畫出某些layer的所有輸出結果:

Layer1



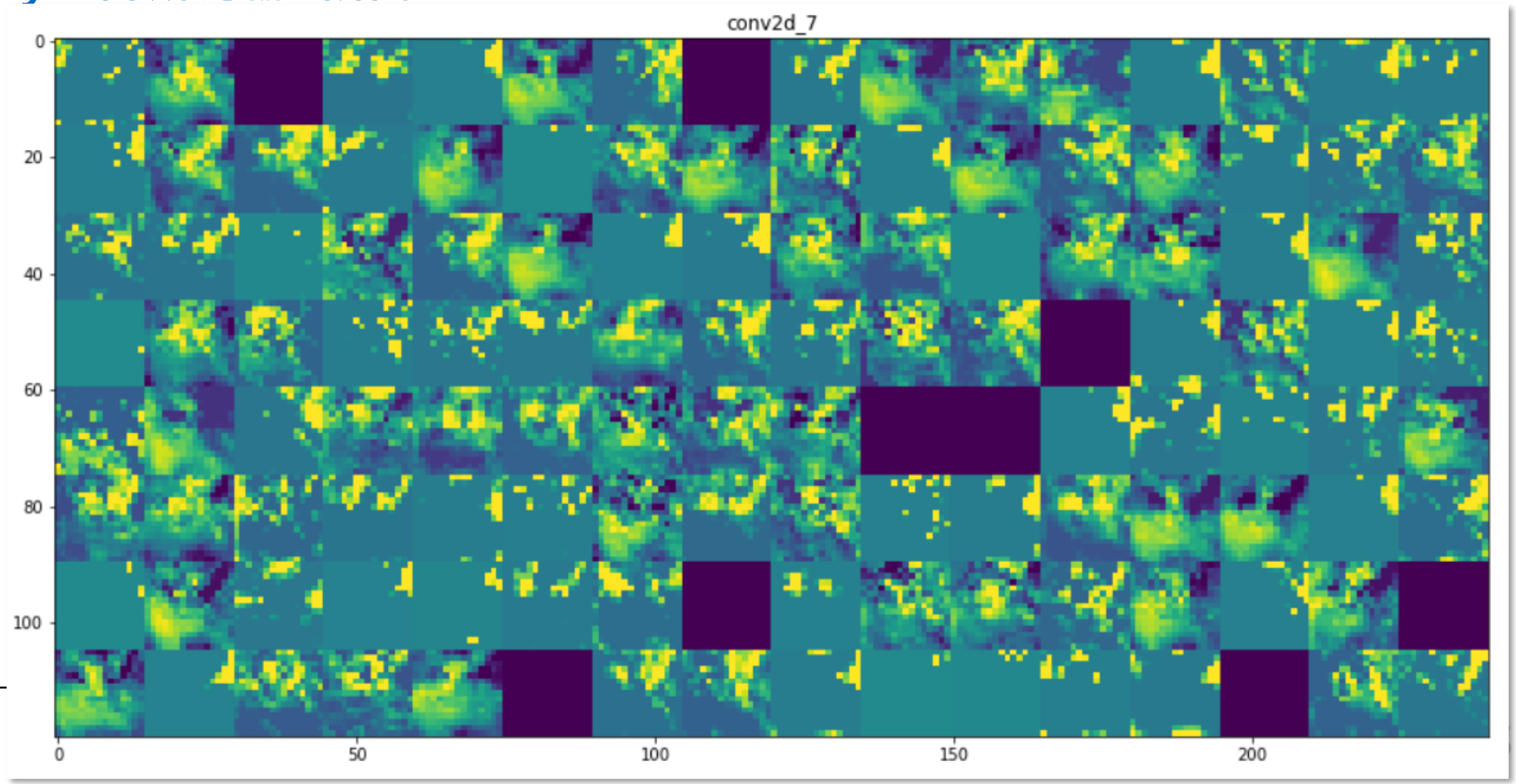
Layer4



(4)視覺化訓練過程

畫出某些layer的所有輸出結果：

Layer7



(4)視覺化訓練過程

層數越深，輸出結果會越來越抽象，

但與類別相關的訊息會越多！

1. 簡介 & 安裝
2. 神經網路(NN)
3. 應用於電腦視覺 (CNN)
4. 生成式深度學習 (Deep dream)
5. 補充

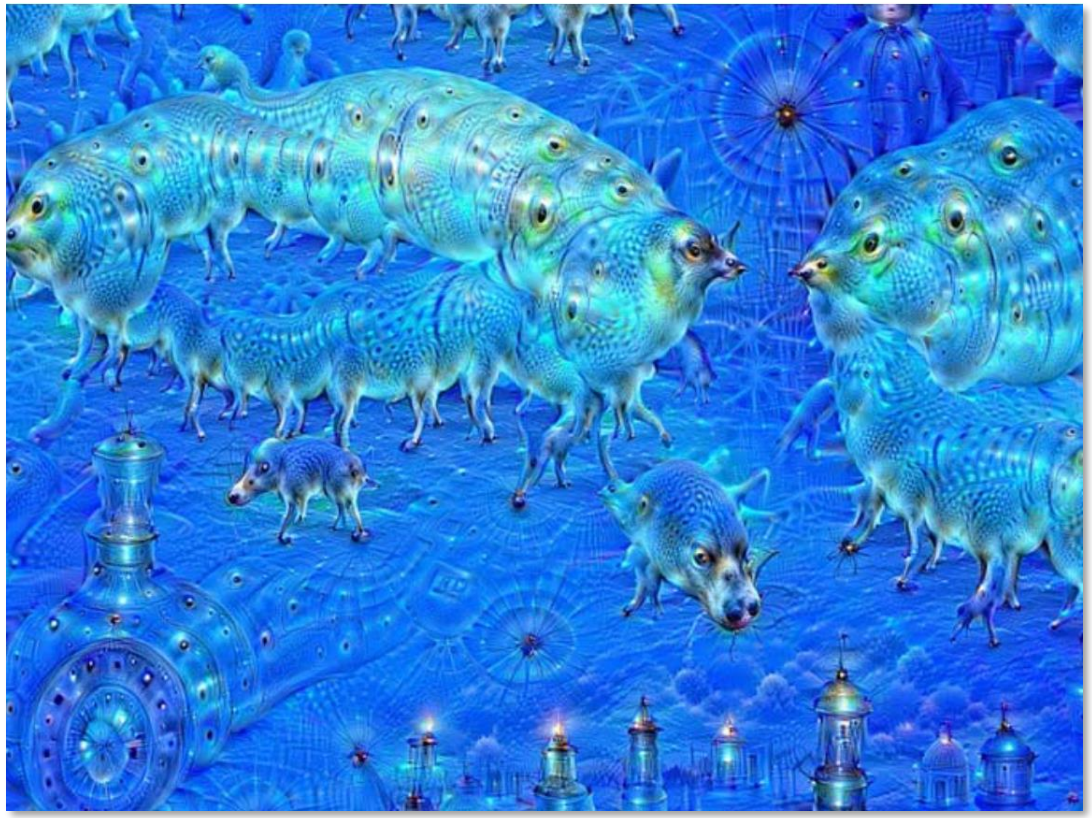
DeepDream

DeepDream 是由 Google 在 2015 年創建的計算機視覺程序，它使用卷積神經網絡透過算法空想性錯視來查找和增強圖像中的圖案，從而在故意過度處理的圖像中創建夢幻般的幻覺外觀。

DeepDream

DeepDream 卷積神經網路在 ImageNet 上訓練

`ImageNet` 中狗和鳥的樣本特別多



DeepDream 演算法

反向運作卷積神經網路

對卷積神經網路的輸入圖像做梯度上升，以使卷積神經網路靠層的過濾器激活最大化

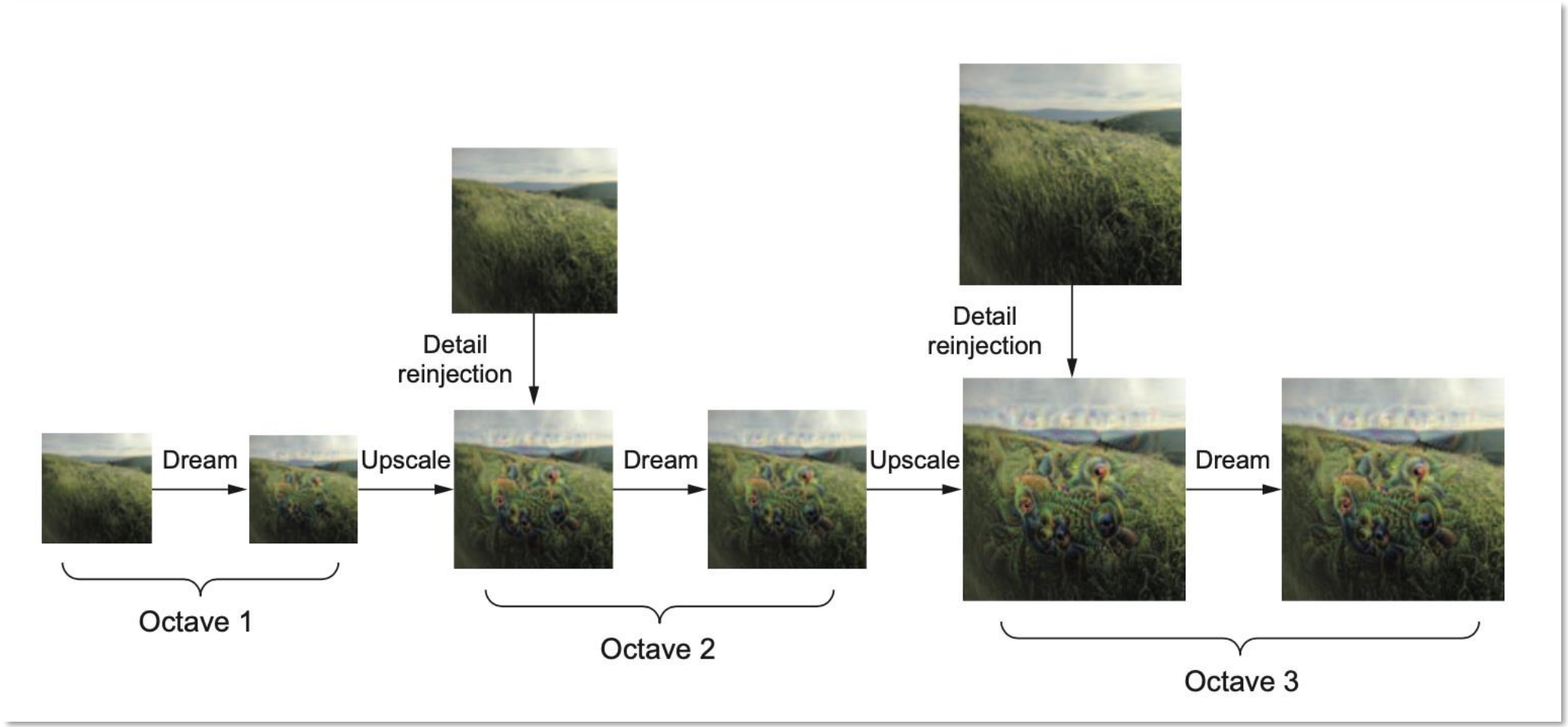
特點：

- 嘗試將所有層都激活最大化，因此需要同時將大量特徵的視覺化混合在一起
- 不能從空白、帶有微噪音的輸入開始！
- 從現有的圖像開始，因此產生的效果能夠根據已經存在的視覺元素作扭曲
- 輸入的圖像是在不同的尺度（octave）上進行處理的，因此可以提高可視化的質量

DeepDream 流程

1. 讀取原始圖片
2. 定義尺度（octave）大小（從小到大）
3. 調整原始圖片至最小尺度
4. 在每個（從小到大）的尺度執行：
   - 梯度上升，將loss最大化
   - 將圖片放大至下一尺度
   - 重新注入放大之後損失的細節
5. 直到放大至原始圖片大小則停止。透過計算縮小尺度的圖片與原始圖片的差異，可以定量描述放大時的細節損失

DeepDream 流程



用Keras實現DeepDream

載入預訓練的Inception V3模型

```
import keras
from keras.applications import inception_v3
from keras import backend as K
```

Inception V1

- Inception就是在廣度上增加，並使用不同的捲積核大小(3x3，5x5，1x1)，來提取不同的特徵。
- 引入 1x1 的捲積層降維(降低計算參數量)

Inception V2

- 將5x5捲積核用兩個3x3代替，兩個3x1代替3x3卷積核
- 減少參數量，卻不會降低準確度

Inception V3

- 將7x7分解成 1x7, 7x1 兩個一維捲積與將 3x3分解成 1x3, 3x1；5x5則換成兩個3x3。
- 將1個conv拆成2個convolutions來增加網絡的非線性

用Keras實現DeepDream

設置DeepDream配置

```
layer_contributions = {
    'mixed2': 0.2,
    'mixed3': 3.,
    'mixed4': 2.,
    'mixed5': 1.5,
}
```

包含層的名稱及係數，係數代表該層對於要最大化損失的貢獻大小

`model.summary()`

層的名稱編碼在Inception_V3之中，
可透過model.summary()列出所有層的名稱

用Keras實現DeepDream

定義需要最大化的損失

```
# Define the loss.
loss = K.variable(0.)
for layer_name in layer_contributions:
    # Add the L2 norm of the features of a layer to the loss.
    coeff = layer_contributions[layer_name]
    activation = layer_dict[layer_name].output

    # We avoid border artifacts by only involving non-border pixels in the loss.
    scaling = K.prod(K.cast(K.shape(activation), 'float32'))
    loss += coeff * K.sum(K.square(activation[:, 2: -2, 2: -2, :])) / scaling
```

將該層特徵的L2 norm添加到loss中

為了避免出現邊界偽影，損失中僅包函非邊界的像素！

用Keras實現DeepDream

梯度上升過程

```
# This holds our generated image
dream = model.input

# Compute the gradients of the dream with regard to the loss.
grads = K.gradients(loss, dream)[0]

# Normalize gradients.
grads /= K.maximum(K.mean(K.abs(grads)), 1e-7)

# Set up function to retrieve the value
# of the loss and gradients given an input image.
outputs = [loss, grads]
fetch_loss_and_grads = K.function([dream], outputs)

def eval_loss_and_grads(x):
    outs = fetch_loss_and_grads([x])
    loss_value = outs[0]
    grad_values = outs[1]
    return loss_value, grad_values

def gradient_ascent(x, iterations, step, max_loss=None):
    for i in range(iterations):
        loss_value, grad_values = eval_loss_and_grads(x)
        if max_loss is not None and loss_value > max_loss:
            break
        print('...Loss value at', i, ':', loss_value)
        x += step * grad_values
    return x
```

將梯度標準化！

給定一張輸出圖像，設置一個Keras函數來獲取損失值和梯度值

Keras

用Keras實現DeepDream

輔助函數

```
import scipy
from keras.preprocessing import image

def resize_img(img, size):
    img = np.copy(img)
    factors = (1,
               float(size[0]) / img.shape[1],
               float(size[1]) / img.shape[2],
               1)
    return scipy.ndimage.zoom(img, factors, order=1)


def save_img(img, fname):
    pil_img = deprocess_image(np.copy(img))
    scipy.misc.imsave(fname, pil_img)
def preprocess_image(image_path):
    # Util function to open, resize and format pictures
    # into appropriate tensors.
    img = image.load_img(image_path)
    img = image.img_to_array(img)
    img = np.expand_dims(img, axis=0)
    img = inception_v3.preprocess_input(img)
    return img


def deprocess_image(x):
    # Util function to convert a tensor into a valid image.
    if K.image_data_format() == 'channels_first':
        x = x.reshape((3, x.shape[2], x.shape[3]))
        x = x.transpose((1, 2, 0))
    else:
        x = x.reshape((x.shape[1], x.shape[2], 3))
    x /= 2.
    x += 0.5
    x *= 255.
    x = np.clip(x, 0, 255).astype('uint8')
    return x
```

Scipy

前處理：
打開圖像、改變圖像大小、將圖像格式轉換為Inception_V3模型能夠處理的張量

將張量轉換為有效圖像

用Keras實現DeepDream

在多個連續尺度上運行梯度上升

```
import numpy as np

# Playing with these hyperparameters will also allow you to achieve new effects

step = 0.01  # Gradient ascent step size
num_octave = 3  # Number of scales at which to run gradient ascent
octave_scale = 1.4  # Size ratio between scales
iterations = 20  # Number of ascent steps per scale

# If our loss gets larger than 10,
# we will interrupt the gradient ascent process, to avoid ugly artifacts
max_loss = 10.

# Fill this to the path to the image you want to use
base_image_path = '/home/ubuntu/data/original_photo_deep_dream.jpg'

# Load the image into a Numpy array
img = preprocess_image(base_image_path)

# We prepare a list of shape tuples
# defining the different scales at which we will run gradient ascent
original_shape = img.shape[1:3]
successive_shapes = [original_shape]
for i in range(1, num_octave):
    shape = tuple([int(dim / (octave_scale ** i)) for dim in original_shape])
    successive_shapes.append(shape)
```

Numpy

steps:梯度上升步長
num_octave:梯度上升尺度個數
octave_scale:兩尺度間的大小比例
iterations:在每個尺度上執行梯度上升的步數

將基礎圖像加載成一個Numpy數組

準備一個由形狀元祖組成的列表，
定義執行梯度上升的不同尺度

用Keras實現DeepDream

在多個連續尺度上運行梯度上升

```
# Reverse list of shapes, so that they are in increasing order
successive_shapes = successive_shapes[::-1]

# Resize the Numpy array of the image to our smallest scale
original_img = np.copy(img)
shrunk_original_img = resize_img(img, successive_shapes[0])

for shape in successive_shapes:
    print('Processing image shape', shape)
    img = resize_img(img, shape)
    img = gradient_ascent(img,
                          iterations=iterations,
                          step=step,
                          max_loss=max_loss)
    upscaled_shrunk_original_img = resize_img(shrunk_original_img, shape)
    same_size_original = resize_img(original_img, shape)
    lost_detail = same_size_original - upscaled_shrunk_original_img

    img += lost_detail
    shrunk_original_img = resize_img(original_img, shape)
    save_img(img, fname='dream_at_scale_' + str(shape) + '.png')

save_img(img, fname='final_dream.png')
```

執行梯度上升，改變夢境圖像

將丟失的細節重新注入到夢境圖像中

1. 簡介 & 安裝
2. 神經網路(NN)
3. 應用於電腦視覺 (CNN)
4. 生成式深度學習 (Deep dream)
5. 補充

補充

- **Deep learning with Python(use Keras) 這是一本書!**

https://tanthiamhuat.files.wordpress.com/2018/03/deeplearningwithpython.pdf

- **Keras**

https://codertw.com/%E7%A8%8B%E5%BC%8F%E8%AA%9E%E8%A8%80/568066/#outline__0_1

Thanks!