Εύρηκα

運用CNN實現食物種類及份量辨識

作者：林宸宇
黃偉翔
指導老師：彭天健老師

大綱

- 研究動機
- 研究目的
- 研究方法
- 研究結果
- 參考資料

研究動機

- 接觸人工智慧
- 美食愛好者
- 與營養師的長談

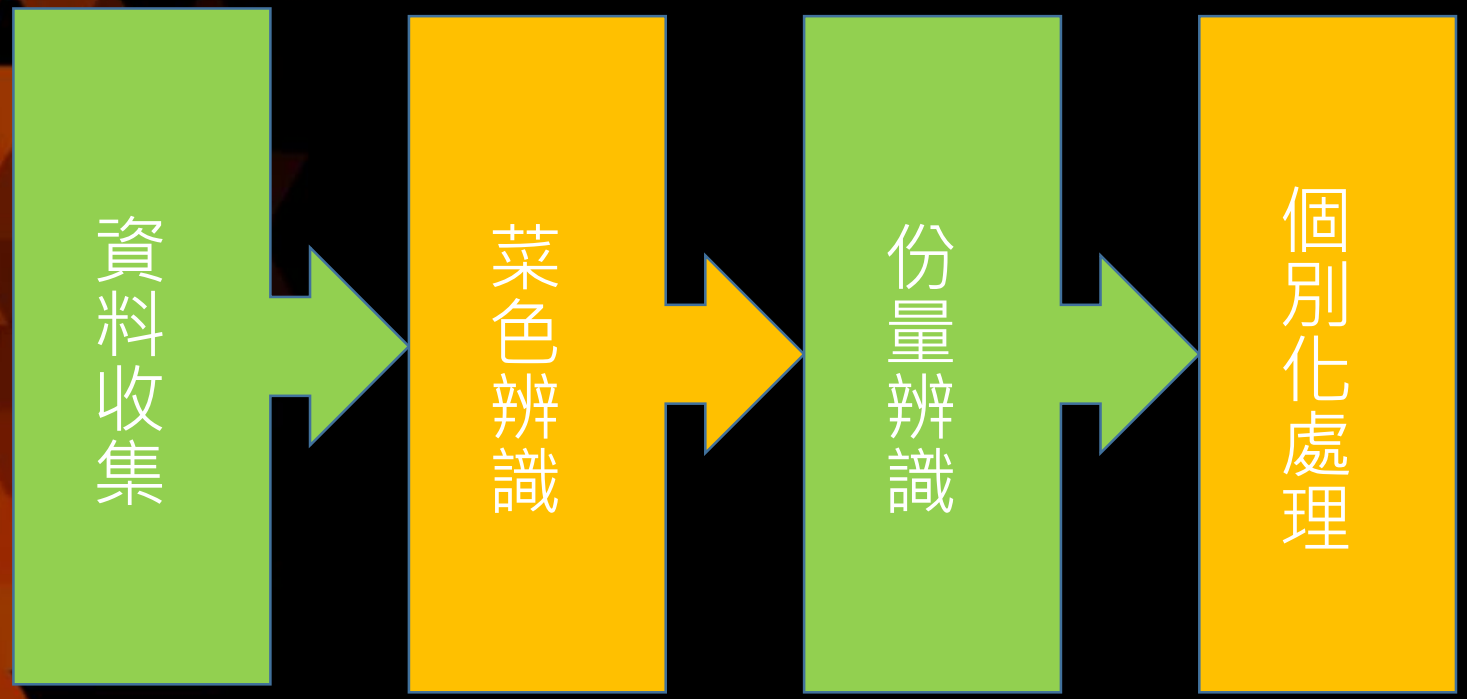
研究目的

Input

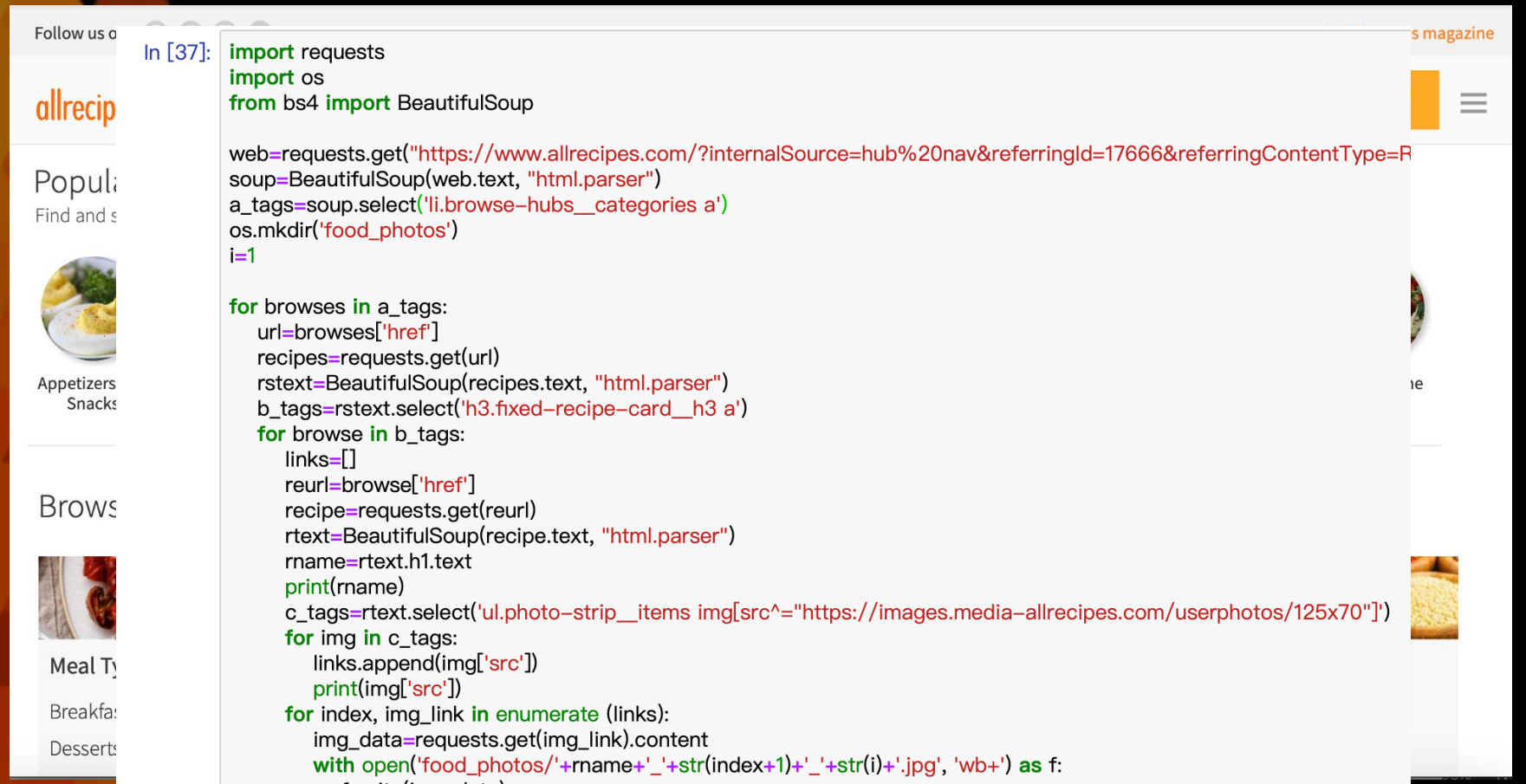
Convolution Neural Network

Output
一份
番茄炒蛋

研究方法



資料收集

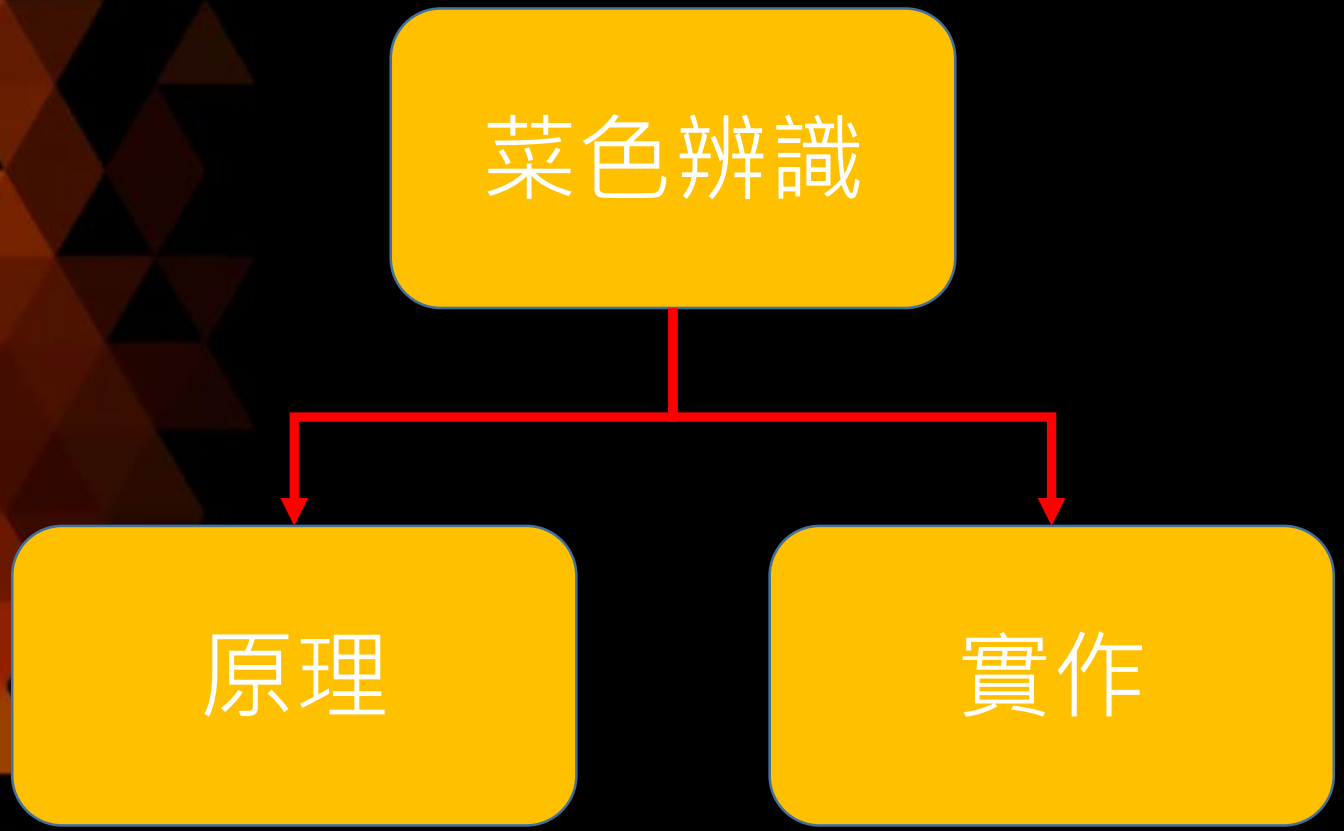

Follow us o
s magazine
allrecip
Popul
Find and s
Appetizers
Snacks
Brows
Meal Ty
Breakfas
Desserts
In [37]:
import requests
import os
from bs4 import BeautifulSoup
web=requests.get("https://www.allrecipes.com/?internalSource=hub%20nav&referringId=17666&referringContentType=R
soup=BeautifulSoup(web.text, "html.parser")
a_tags=soup.select('li.browse-hubs__categories a')
os.mkdir('food_photos')
i=1
for browses in a_tags:
url=browses['href']
recipes=requests.get(url)
rstext=BeautifulSoup(recipes.text, "html.parser")
b_tags=rstext.select('h3.fixed-recipe-card__h3 a')
for browse in b_tags:
links=[]
reurl=browse['href']
recipe=requests.get(reurl)
rtext=BeautifulSoup(recipe.text, "html.parser")
rname=rtext.h1.text
print(rname)
c_tags=rtext.select('ul.photo-strip__items img[src^="https://images.media-allrecipes.com/userphotos/125x70"]')
for img in c_tags:
links.append(img['src'])
print(img['src'])
for index, img_link in enumerate (links):
img_data=requests.get(img_link).content
with open('food_photos/'+rname+'_'+str(index+1)+'_'+str(i)+'.jpg', 'wb+') as f:
f.write(img_data)
f.close()
i+=1
del links

Google-images-download

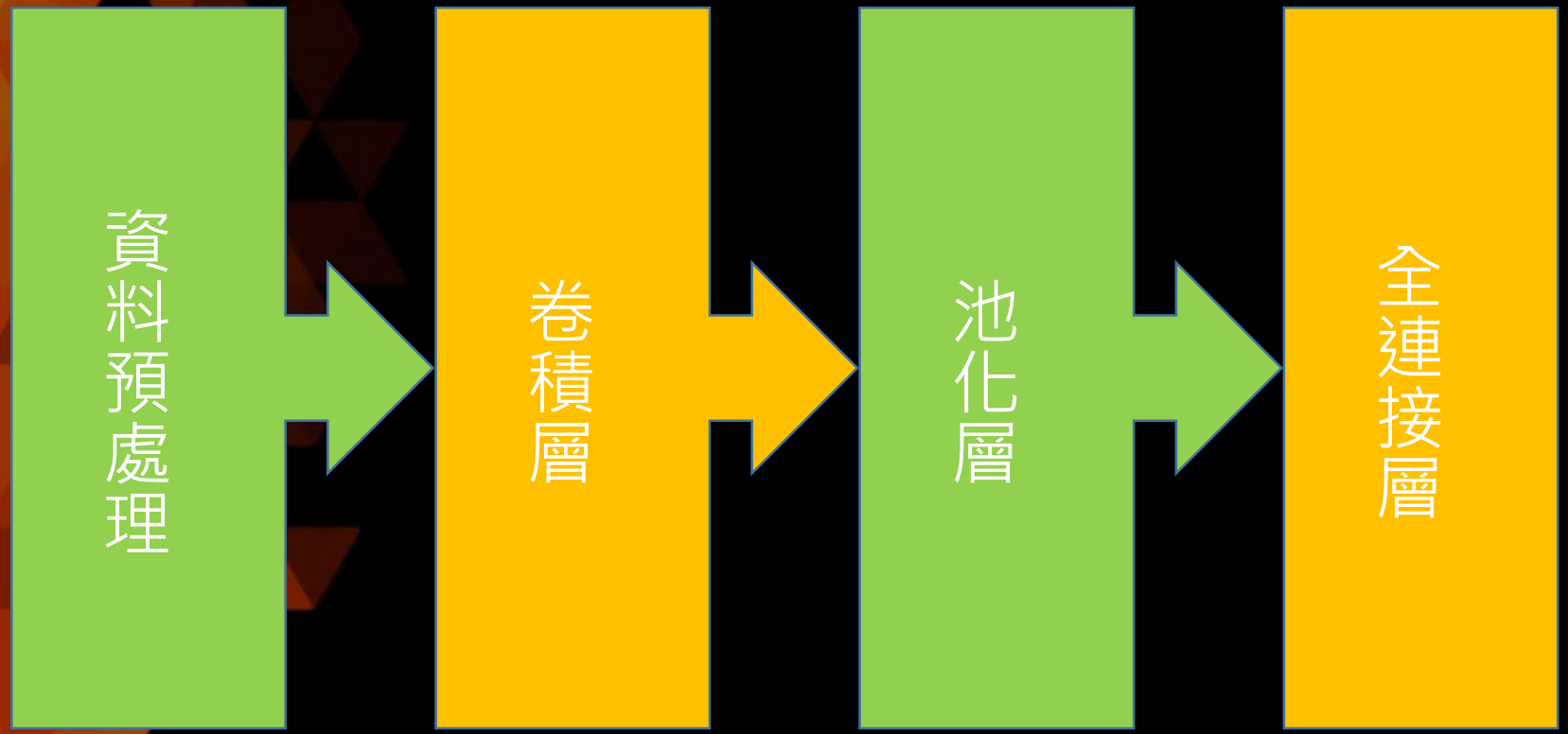
程式碼：

```
from google_images_download import google_images_download
response = google_images_download.googleimagesdownload()
list=['番茄炒蛋']
for examples in list:
    arguments = {"keywords":example
    paths = response.download(argun
    print(paths)
```

執行結果：


菜色辨識
原理
實作

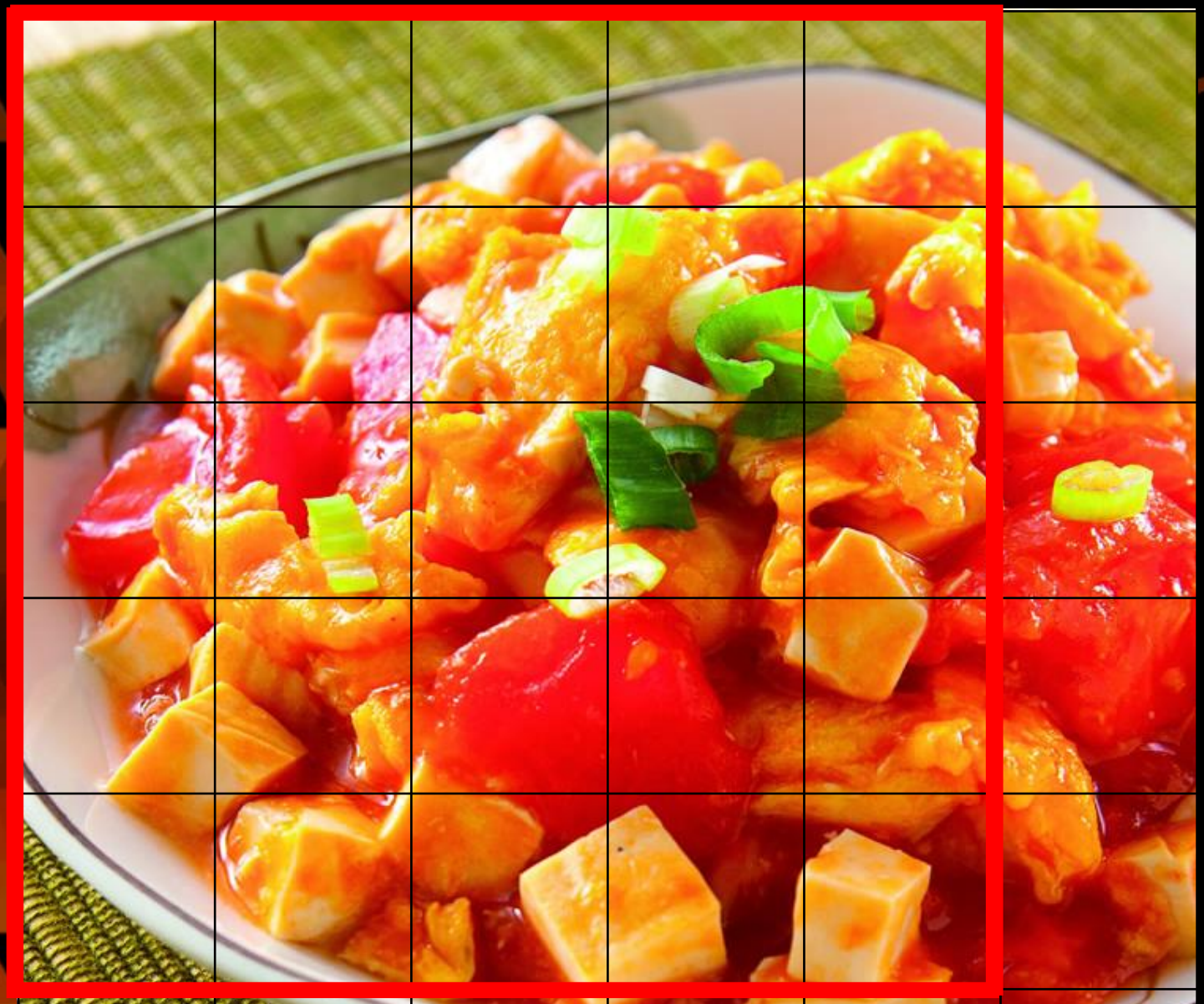
CNN卷積神經網路



資料預處理

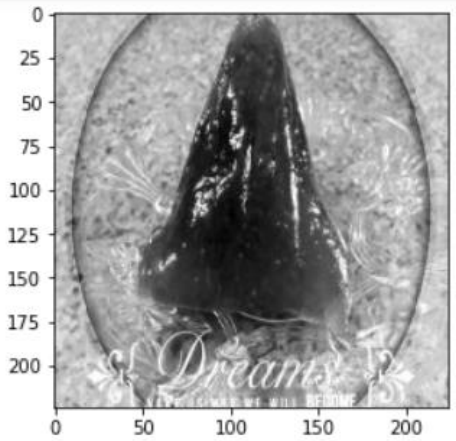
```
In [3]: def create_training_data() :
            for category in CATEGORIES_train :
                path = os.path.join(DATADIR,category)
                class_num = CATEGORIES_train.index(category)
                print(path)
                for img in os.listdir(path) :
                    try:
                        new_path = os.path.join(path,img)
                        img_array = mpimg.imread(new_path)#, cv2.IMREAD_GRAYSCALE)
                        new_array = cv2.resize(img_array, (img_size, img_size))
                        if new_array.shape==(img_size,img_size,3):
                            for i in range(img_size-crop_size):
                                x=i
                                for j in range(img_size-crop_size):
                                    y=j
                                    crop_array=new_array[y: y+crop_size, x: x+crop_size]
                                    training_data.append([crop_array, class_num])
                                    flip_array=cv2.flip(crop_array, 1)
                                    training_data.append([flip_array, class_num])
                    except Exception as e :
                        pass
```

Data augmentation



圖片型態

```
training_data = []
for category in CATEGORIES_train :
        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES_train.index(category)
        for img in os.listdir(path) :
            try:
                new_path = os.path.join(path,img)
                img_array = mpimg.imread(new_path,1)
                new_array = cv2.cvtColor(img_array, cv2.COLOR_BGR2GRAY)
                training_data.append([new_array, class_num])
                if class_num == 1 and size <= 10:
                    plt.imshow(new_array, cmap='gray')
                    plt.show()
            except Exception as e :
                pass
```

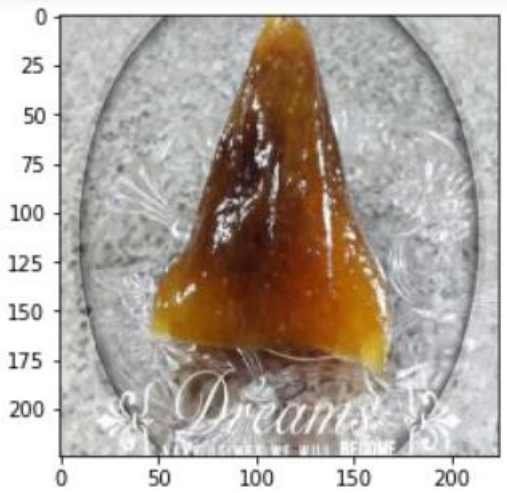



```
In [20]: for img in training_data[:1] :
             print (img)
```

```
[array([[ 69,  69,  70, ...,  73,  72,  72],
        [ 67,  68,  68, ...,  71,  70,  70],
        [ 66,  67,  67, ...,  68,  69,  69],
        ...,
        [ 25,  25,  27, ..., 206, 170,  90],
        [ 26,  26,  26, ..., 181, 110,  38],
        [ 26,  27,  27, ..., 126,  49,  35]], dtype=uint8), 0]
```

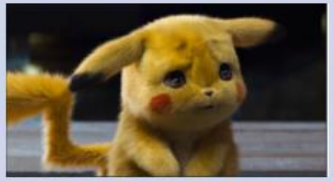
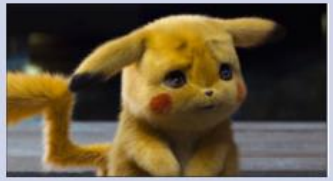
圖片型態

```python
training_data = []
for category in CATEGORIES_train :
        path = os.path.join(DATADIR,category)
        class_num = CATEGORIES_train.index(category)
        for img in os.listdir(path) :
            try:
                new_path = os.path.join(path,img)
                img_array = mpimg.imread(new_path,1)
                training_data.append([img_array, class_num])
                if class_num == 1 :
                    plt.imshow(img_array, cmap='gray')
                    plt.show()
            except Exception as e :
                pass
```



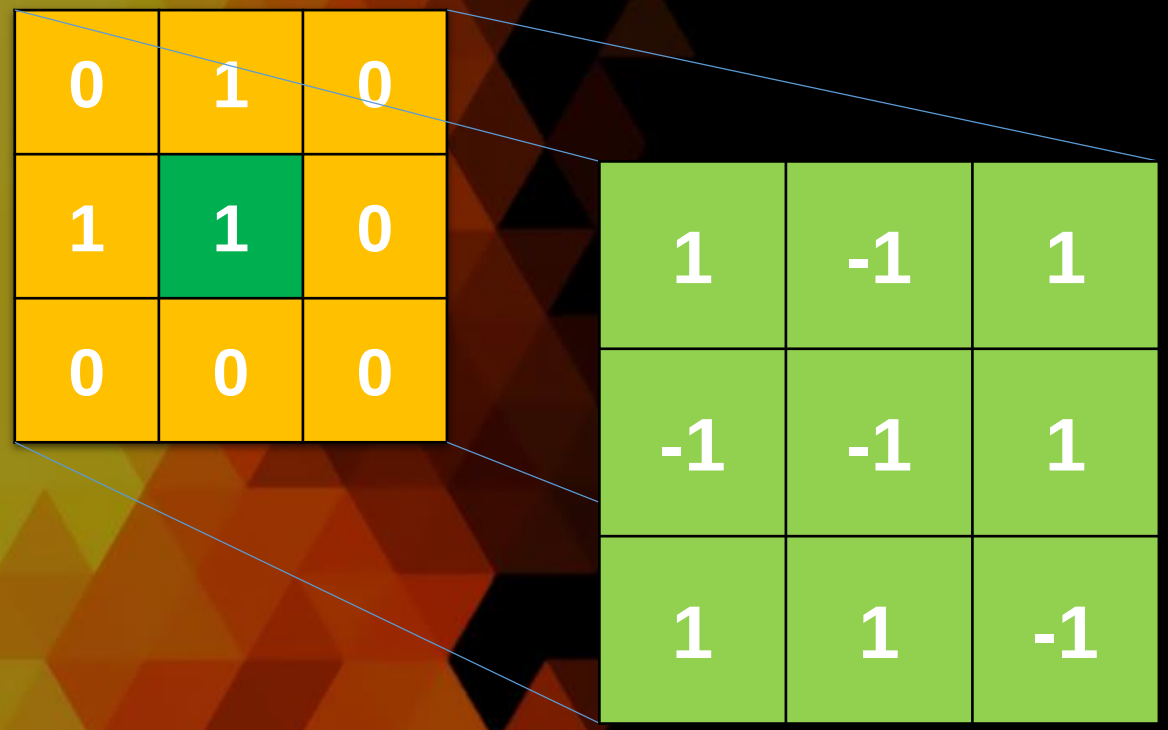

```
In [5]: for img in training_data[:1] :
            print (img)

[array([[[135,  65,  53],
         [135,  65,  53],
         [136,  66,  54],
         ...,
         [131,  69,  58],
         [131,  69,  56],
         [132,  69,  54]],

        [[133,  63,  51],
         [134,  64,  52],
         [134,  64,  52],
         ...,
         [127,  67,  56],
         [129,  67,  54],
         [129,  67,  54]],

        [[130,  62,  49],
         [131,  63,  50],
         [131,  63,  50],
         ...,
         [123,  65,  54],
         [125,  65,  54],
         [125,  65,  54]],

        ...,

        [[ 43,  23,  22],
         [ 43,  23,  22],
         [ 43,  25,  25],
         ...,
         [205, 204, 209],
         [171, 169, 172],
         [ 91,  89,  92]],

        [[ 44,  24,  23],
         [ 44,  24,  23],
         [ 44,  24,  23],
         ...,
         [180, 180, 182],
         [111, 109, 112],
         [ 39,  37,  40]],

        [[ 44,  24,  23],
         [ 45,  25,  24],
         [ 45,  25,  24],
         ...,
         [125, 125, 127],
         [ 50,  48,  51],
         [ 38,  33,  37]]], dtype=uint8), 0]
```

卷積層Convolution

Original Image	3x3 Kernel	After Image	
	[0 0 0] [0 1 0] [0 0 0]		原圖不變
	[1 1 1] [1 -7 1] [1 1 1]		銳化
	[-1 -1 -1] [-1 8 -1] [-1 -1 -1]		邊緣強化
	[-1 -1 0] [-1 0 1] [0 1 1]		浮雕

https://miro.medium.com/max/1400/1*Ai489J5xlRaszTHlS3rK5w.png

卷積層Convolution



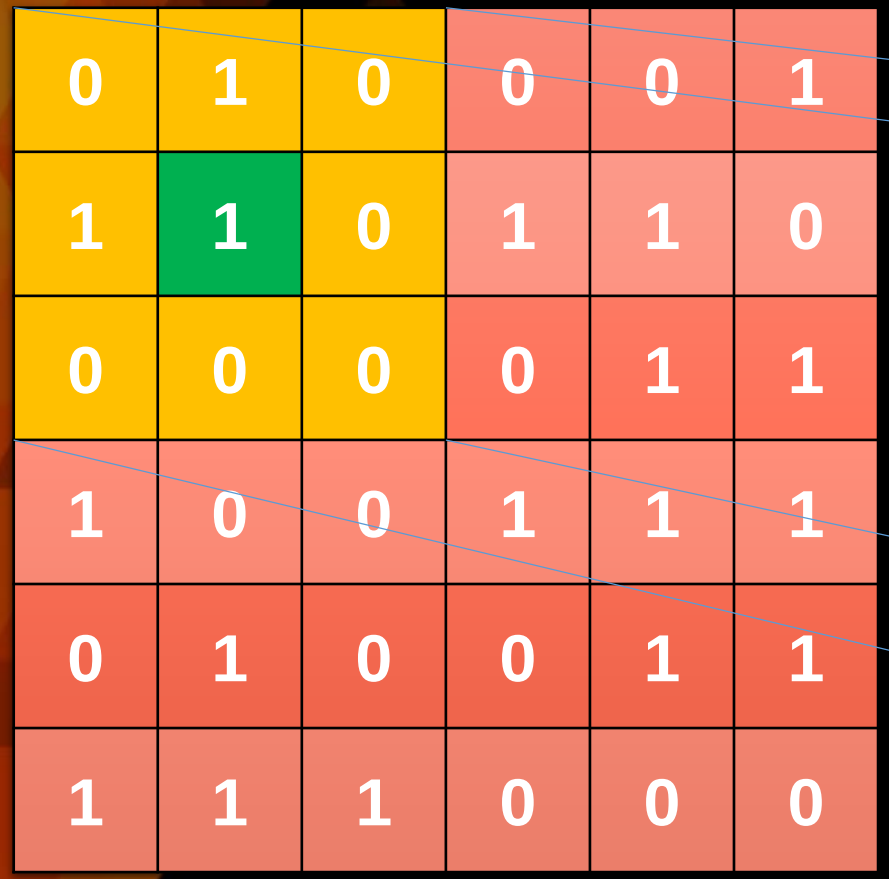
filter

-3

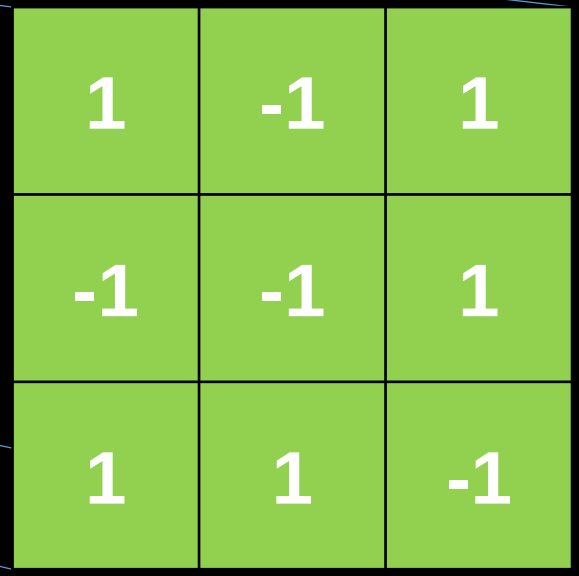
[0×1+1×(-1)+0×1+
1×(-1)+1×(-1)+0×1+
0×1+0×1+0×(-1)]
= **-3**

4×4 feature map

卷積層Convolution

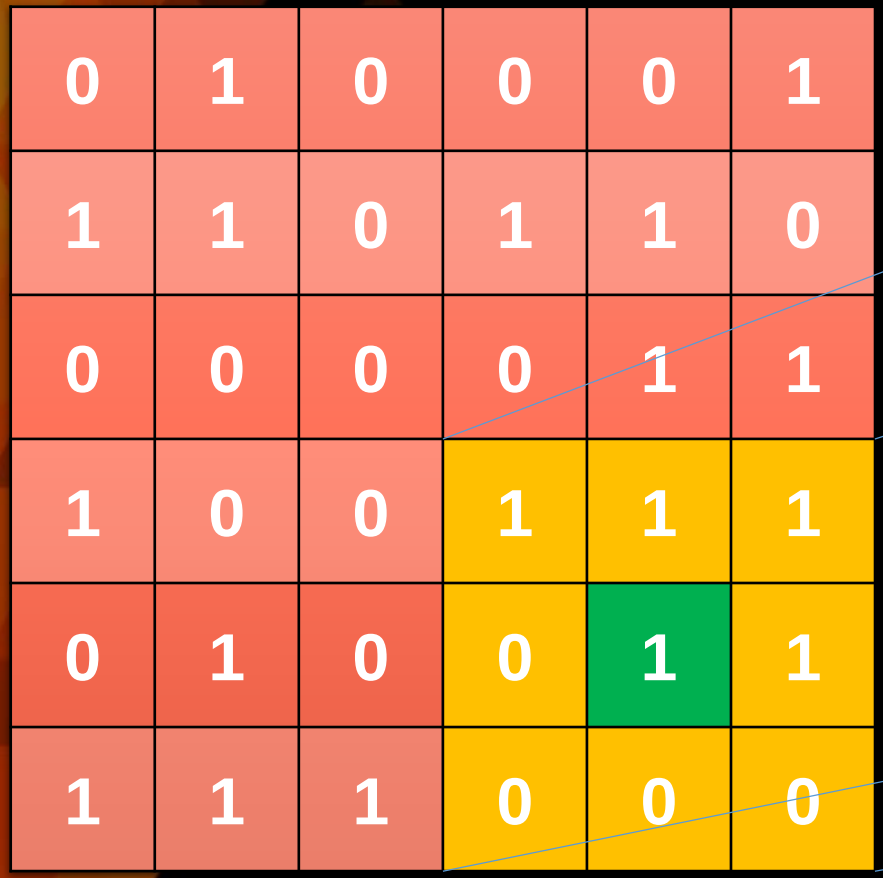


6×6 image

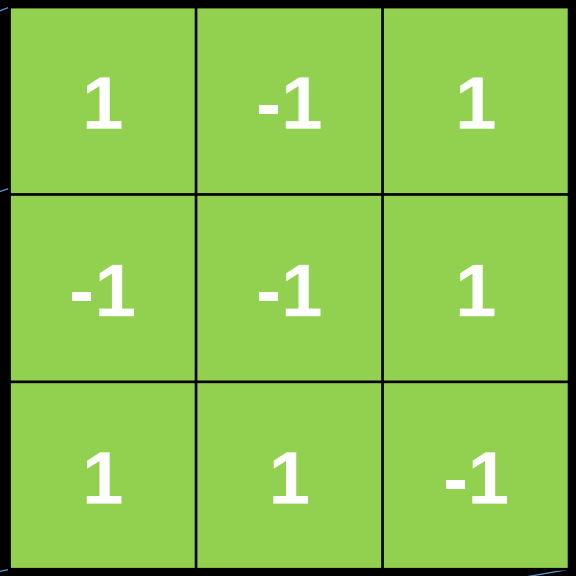


filter

卷積層Convolution



6×6 image



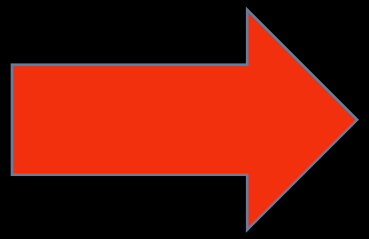
filter

卷積層Convolution

池化層Pooling(MaxPooling)

3	1	-1	4
0	-4	-3	1
0	2	0	-1
1	-2	2	1

4×4 image



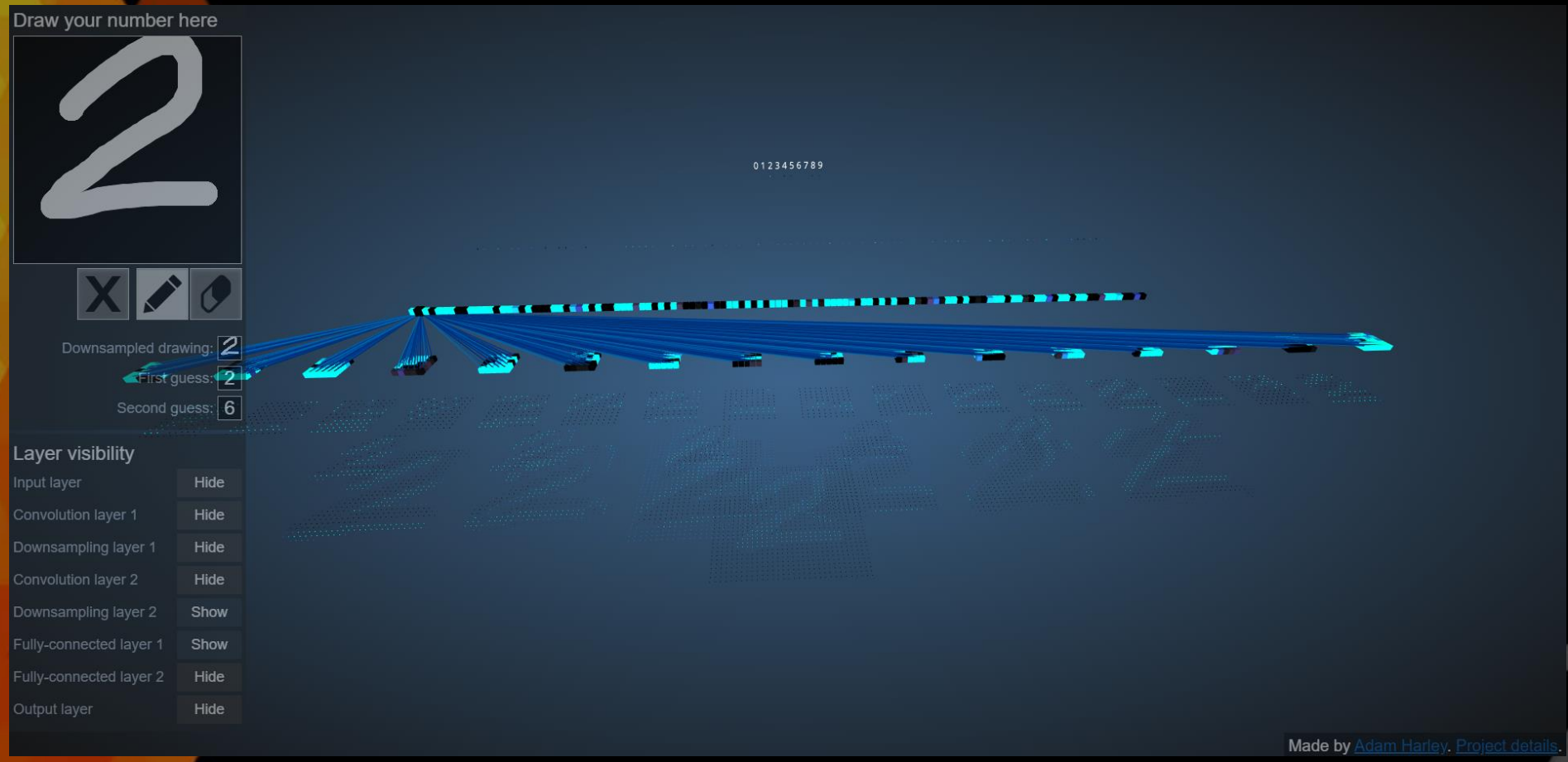
Max Pooling

3	4
2	2

2×2 image

Flatten

Fully Connected Layer



https://www.cs.ryerson.ca/~aharley/vis/conv/

神經網路結構

```
In [5]: model = Sequential()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 224, 224, 64)      1792
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 224, 224, 64)      36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 112, 112, 64)      0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 112, 112, 128)     73856
_________________________________________________________________
leaky_re_lu_1 (LeakyReLU)    (None, 112, 112, 128)     0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 112, 112, 128)     147584
_________________________________________________________________
leaky_re_lu_2 (LeakyReLU)    (None, 112, 112, 128)     0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 56, 56, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 256)       295168
_________________________________________________________________
leaky_re_lu_3 (LeakyReLU)    (None, 56, 56, 256)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 56, 56, 256)       590080
_________________________________________________________________
leaky_re_lu_4 (LeakyReLU)    (None, 56, 56, 256)       0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 56, 56, 256)       590080
_________________________________________________________________
leaky_re_lu_5 (LeakyReLU)    (None, 56, 56, 256)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 28, 28, 256)       0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 28, 28, 512)       1180160
_________________________________________________________________
leaky_re_lu_6 (LeakyReLU)    (None, 28, 28, 512)       0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 28, 28, 512)       2359808
_________________________________________________________________
leaky_re_lu_7 (LeakyReLU)    (None, 28, 28, 512)       0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 28, 28, 512)       2359808
_________________________________________________________________
leaky_re_lu_8 (LeakyReLU)    (None, 28, 28, 512)       0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 14, 14, 512)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
leaky_re_lu_9 (LeakyReLU)    (None, 14, 14, 512)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
leaky_re_lu_10 (LeakyReLU)   (None, 14, 14, 512)       0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 14, 14, 512)       2359808
_________________________________________________________________
leaky_re_lu_11 (LeakyReLU)   (None, 14, 14, 512)       0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 7, 7, 512)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              51382272
_________________________________________________________________
leaky_re_lu_12 (LeakyReLU)   (None, 2048)              0
_________________________________________________________________
dense_2 (Dense)              (None, 2048)              4196352
_________________________________________________________________
leaky_re_lu_13 (LeakyReLU)   (None, 2048)              0
_________________________________________________________________
dense_3 (Dense)              (None, 10)                20490
=================================================================
Total params: 70,313,802
Trainable params: 70,313,802
Non-trainable params: 0
_________________________________________________________________
```

研究結果

```
Train on 51123 samples, validate on 12781 samples
Epoch 1/10
51123/51123 [==============================] - 835s 16ms/step - loss: 1.5144 - accuracy: 0.4644 - val_loss: 0.8196 - val_accur
acy: 0.7638
Epoch 2/10
```

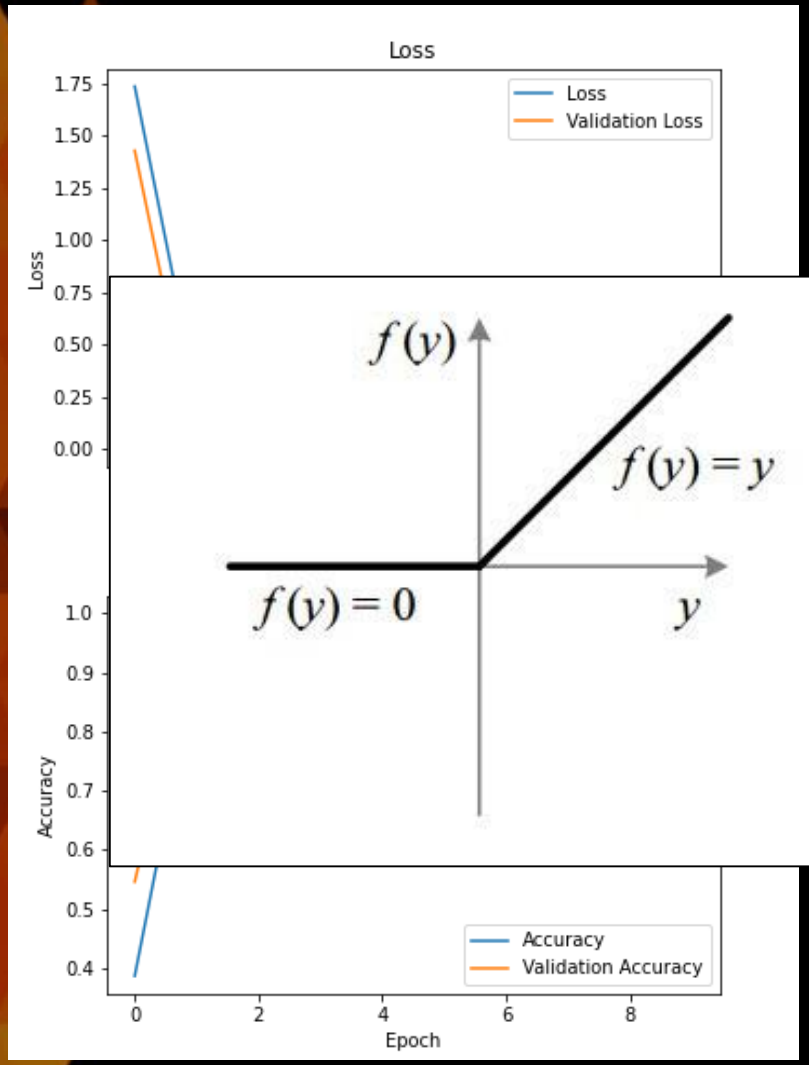
```
def prepare(filepath):
    IMG_SIZE = 224
    img_array = cv2.imread(filepath)
    new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))  # resize image to match model's expected sizing
    return new_array.reshape(-1, IMG_SIZE, IMG_SIZE, 3)  # return the image with shaping that TF wants.
```

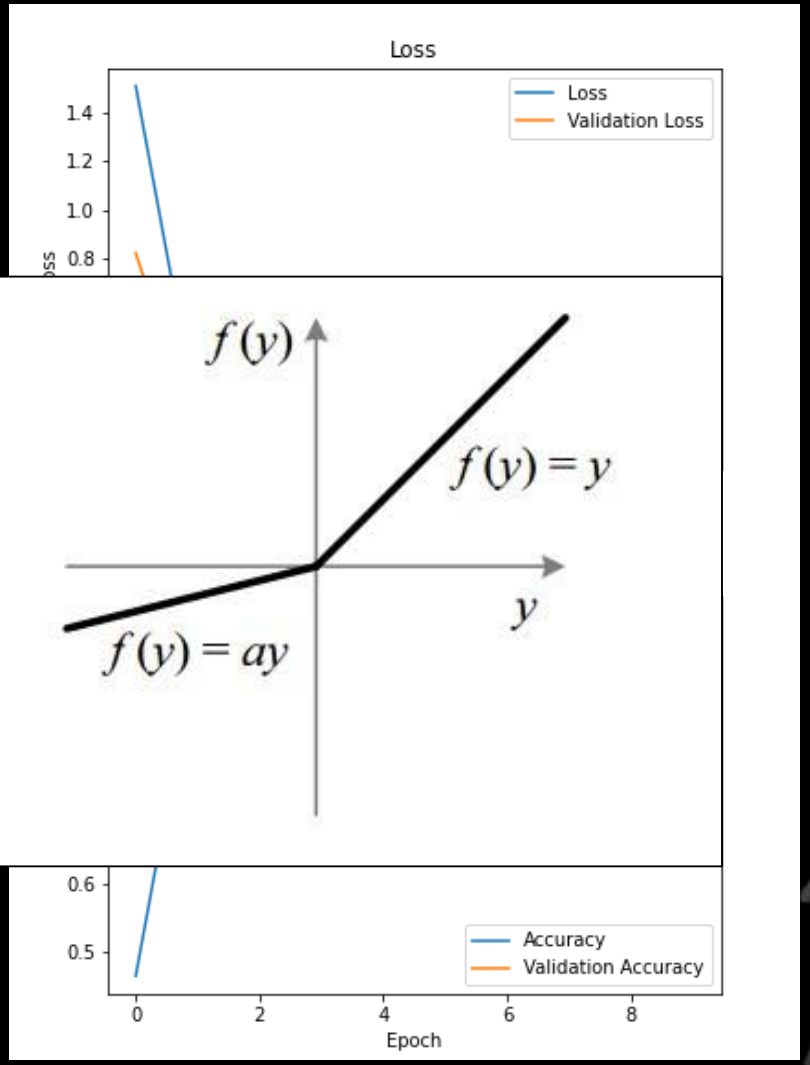
```
from keras.applications.vgg16 import preprocess_input, decode_predictions
prediction = model.predict([prepare('/Users/leolin/Documents/高中專題/downloads/2/269檔案.jpg')])
print (prediction)
p=np.argmax(prediction)
print(CATEGORIES_train[int(p)])
```

```
[[0. 1. 0. 0. 0. 0. 0. 0.]]
2
```

```
al_accuracy: 1.0000
```

Model比較





Relu

Leakey RELU

Object Detection

個別化處理

參考資料

- 經典模型Alexnet的技術研究
- https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
- 神經網路學習套件：keras&tensorflow
- https://keras.io/
- https://www.tensorflow.org/
- 臺大電機李宏毅教授的線上教學資源
- http://speech.ee.ntu.edu.tw/~tlkagk/courses_ML20.html

特別感謝

- 彭天健老師
- 張詠裕叔叔
- 張又心營養師
- 一路支持我們的家長


Εύρηκα
Thanks for listening!
π