Day 16

深度學習與電腦視覺

學習馬拉松

cupoy 陪跑專家：楊哲寧

CUPOY

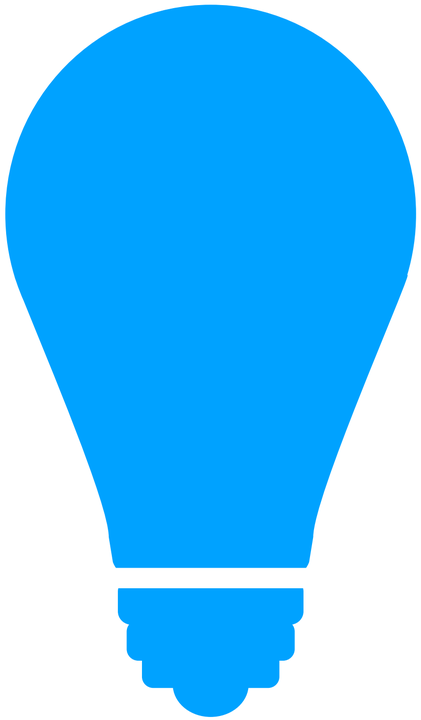
深度學習理論與實作

Image Augmentation
(圖像增強)

重要知識點

CUPOY

- 了解圖像增強的意義
- 了解如何使用 Keras 做 Image Augmentation
- 了解如何使用 Imgaug

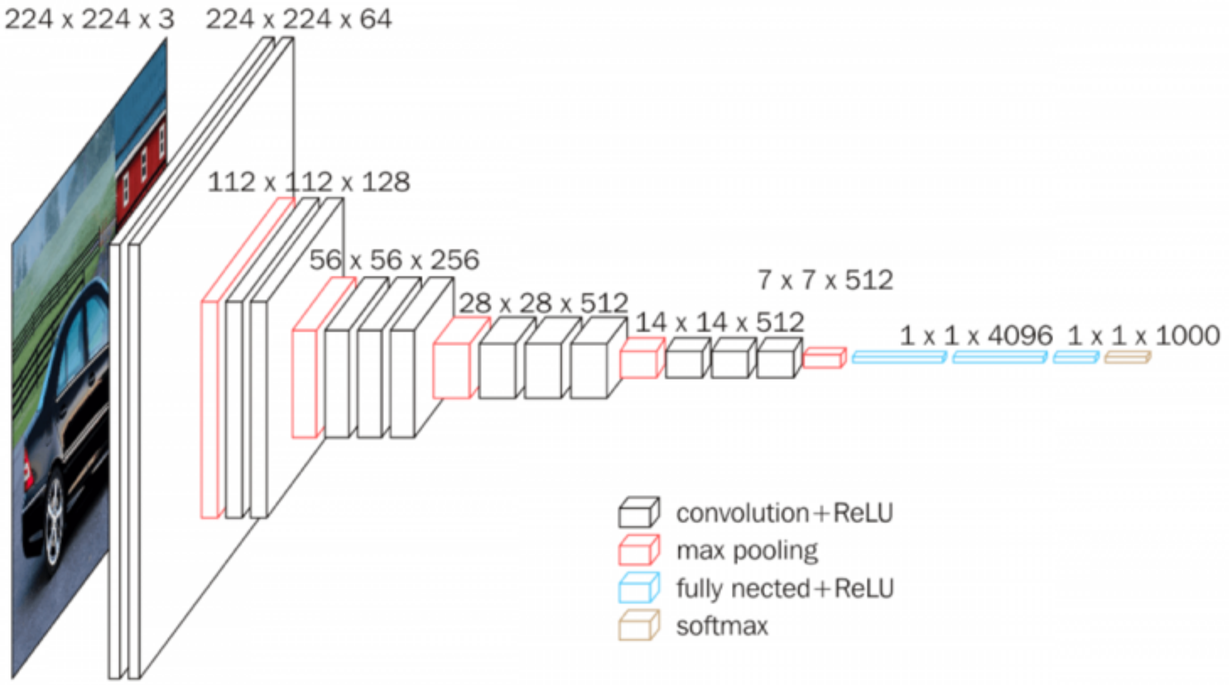


這次內容與作業會分為 2 個 parts 請參考
Part1、Part2，個別有對應的 ppt 與作業內容。

Part 01

彌補資料不足

通常我們想要訓練一個準確性高、泛化性好的 CNN 圖像辨識器，除了模型本身的架構外，最關鍵的就是我們所擁有的資料，資料量越大、越完整，不用太過於複雜的架構也能獲得不錯的效果！



參考來源： Neurohive

彌補資料不足

CUPOY

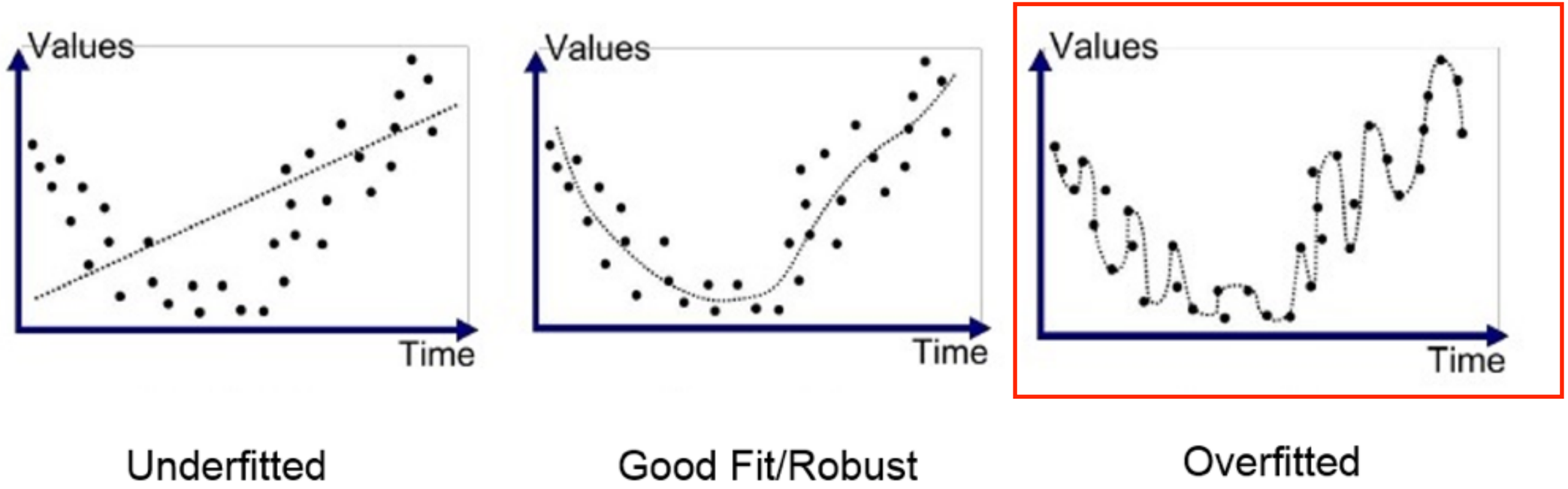
一般來說，圖像資料的收集往往是我們最大的痛點，而在資料有限的狀況下，我們又該如何自己創造新的資料呢？最簡單的方式就是透過 Image augmentation，我們藉由旋轉、裁切、增加噪點、白化等技術，將原本的圖片做了一些『轉換』，如此一來，我們就增加了許多的資料。

圖像增強示意

避免Overfitting

CUPOY

在訓練一個分類器時，大家應該很容易遇到 Overfitting 的狀況，也就是對 Training Data 過於完美的擬合，此時，透過適當的圖像增強，也能降低 Overfitting 的可能性。



Overfitting (當預測虛線完美吻合資料)

參考來源： Medium - Greyatom

Keras ImageDataGenerator

CUPOY

Keras 本身做 Augmentation 的工具為 ImageDataGenerator，
ImageDataGenerator 可以做到許多的圖像增強，以下列出幾項常見的項目為大家介紹：

featurewise_center：輸入為Boolean(True or False)，以每一張feature map為單位將平均值設為0
featurewise_std_normalization: 輸入為Boolean(True or False) ，以每一張feature map為單位將數值除以其標準差(上述兩步驟就是我們常見的Standardization)

Standardization 的用途在於使不同的輸入影像有相似的資料分佈範圍，這樣在收斂時會比較快，也較容易找到 Global Minimum。

Keras ImageDataGenerator

CUPOY

ImageDataGenerator中常見的Augmentation(輸入形式、內容)：

zca_whitening: Boolean，透過ZCA取出重要特徵(詳見：ZCA介紹)
rotation_range：整數值，控制隨機旋轉角度
width_shift_range：「浮點、整數、一維數」，圖像寬度上隨機偏移值
height_shift_range：「浮點、整數、一維數」，圖像高度上隨機偏移值
shear_range：浮點數，裁切範圍
zoom_range：浮點數或範圍，隨機縮放比例
horizontal_flip: Boolean，隨機水平翻轉
vertical_flip:Boolean，隨機垂直翻轉
rescale: 數值，縮放比例
dtype：輸出資料型態

Keras ImageDataGenerator

使用相當簡單，首先要創建一個 Generator，輸入想要使用的 Augmentation 與其參數：

```
from keras.preprocessing.image import ImageDataGenerator

img_gen=ImageDataGenerator(
featurewise_center=True,featurewise_std_normalization=True,rotation_r
ange=10,width_shift_range=0.1,height_shift_range=0.1,shear_range=0.1
,zoom_range=0.1,horizontal_flip=True,vertical_flip=False,dtype=np.float3
2)
```

接著再將影像輸入，這裡的影像是四維的[Batch_Size, Height, Width, Channels]

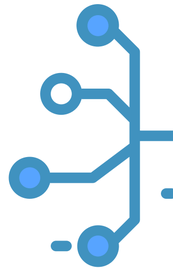
```
batch_gen = img_gen.flow(img,
 batch_size=4)
```

- 輸出影像就會隨機使用指定的Augmentation，詳細程式碼會在Homework中呈現

Imgaug

CUPOY

Imgaug 為一個更為泛用的第三方 Library (Github連結)，安裝方式最簡單的話使用 pip install imgaug，下面我們示範用 Imgaug 做 Image augmentation。



Imgaug

CUPOY

下方示範了最基本的使用方法，其原理與 ImageDataGenerator 相當相似，先創建 Augmentation generator，再將圖像輸入：

```
flipper = iaa.Fliplr(1.0) #水平翻轉機率==1.0

vflipper = iaa.Flipud(0.4) #垂直翻轉機率40%
img_1= vflipper.augment_image(img)

blurer = iaa.GaussianBlur(3.0)
img_2= blurer.augment_image(img) # 高斯模糊圖像( sigma of 3.0)

translater = iaa.Affine(translate_px={"x": -16}) #向左橫移16個像素

img_3= translater.augment_image(img)

scaler = iaa.Affine(scale={“y”: (0.8, 1.2)}) # 縮放照片高度，區間(0.8-1.2倍)

img_4= scaler.augment_image(img)
```

Imgaug

其輸出如下方：

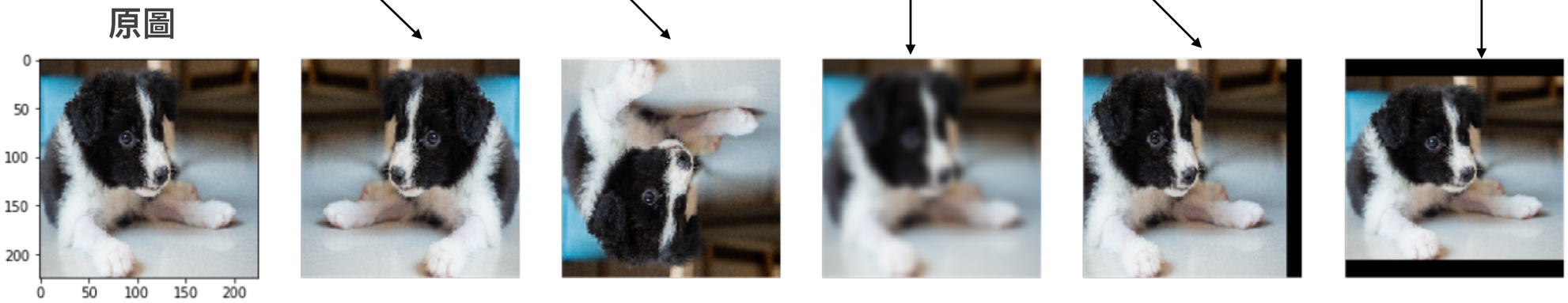
```
scaler = iaa.Affine(scale={“y”: (0.8, 1.2)})
# 縮放照片高度，區間(0.8-1.2倍)
```

```
flipper = iaa.Fliplr(1.0) #水平翻轉機率==1.0
```

```
vflipper = iaa.Flipud(0.4) #垂直翻轉機率40%
```

```
blurer = iaa.GaussianBlur(3.0)
```

```
translater = iaa.Affine(translate_px={"x":
-16}) #向左橫移16個像素
```



Part 02

Imgaug

CUPOY

更為常見的方法是將多種 Augmentation 包裝在一起

```
Sometimes = lambda aug: iaa.Sometimes(0.5, aug) # Sometimes(0.5, ...) 代表每次都有50%的機率運用不同的Augmentation

seq = iaa.Sequential([iaa.Crop(px=(0, 16)), iaa.Fliplr(0.4),
                      sometimes(iaa.CropAndPad(percent=(-0.05, 0.1), pad_mode=ia.ALL, pad_cval=(0, 255) )),
                      sometimes(iaa.Affine(scale={"x": (0.8, 1.2), "y": (0.8, 1.2)}, translate_percent={"x": (-0.2, 0.2), "y": (-0.2, 0.2)},
                                           rotate=(-10, 10), shear=(-8, 8), order=[0, 1], cval=(0, 255), mode=ia.ALL )),
                      sometimes(iaa.Superpixels(p_replace=(0, 1.0), n_segments=(20, 200))),
                      sometimes(iaa.Sharpen(alpha=(0, 0.2), lightness=(0.1, 0.4))), # sharpen images
                      sometimes(iaa.Emboss(alpha=(0, 0.3), strength=(0, 0.5))), # emboss images
                     ],random_order=True)
```

用Sequential打包所有想要應用的Augmentation

```
images_aug = seq.augment_images(img_combine) ## Image Augmentation
```

Imgaug-pipeline

CUPOY

我們也可以將自己定義的 Augmentation 混合 imgaug 打造一個 pipeline

我們自己定義了一個隨機白點的 Augmentation

```python
class RandomDot(object):
    '''Function to randomly add white dot noise to the image
    Parameters
    ----------
    color: int
        color of the noise dot
    probability: float
        probability of each pixel to be added noise dot
    '''
    def __init__(self, color, probability=0.01):
        assert 0.0 < probability < 1.0
        self.color = color
        self.probability = probability

    def __call__(self, image):
        sample = np.random.choice([0.0, 1.0], size=(image.shape[0], image.shape[1]),
                                  p=[1 - self.probability, self.probability])
        sample = sample * self.color
        #sample = np.repeat(np.expand_dims(sample, axis=2), 3, axis=2)
        sample = np.expand_dims(sample, axis=2)
        image = np.expand_dims(image, axis=2)
        image = image + sample
        image = np.clip(image, 0.0, 255.0)
        return image
```

Imgaug-pipeline

CUPOY

混合不同 Augmentation

```
class TrainAugmentations(object):
    def __init__(self):

        img_seq = iaa.Sequential([

             sometimes(iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.03*255), per_channel=0.5)),
             sometimes(iaa.Sharpen(alpha=(0, 0.2), lightness=(0.1, 0.4))), # sharpen images
             sometimes(iaa.Emboss(alpha=(0, 0.3), strength=(0, 0.5))), # emboss images
            ],random_order=True)

        self.aug_pipeline = Compose([
            RandomBrightness(16), #make image brighter or darker
            ImgAugSequence(img_seq),
        ])

    def __call__(self, image,  mask_0,mask_1,mask_2):
        image = self.aug_pipeline(image)
        return image
```

Imgaug-pipeline

CUPOY

其中需要定義一個 **ImgAugSequence** 來包裝 Imgaug 中的 Augmentation，以及 **Compose** 來包裝所有的Augmentation。

```python
class TrainAugmentations(object):
    def __init__(self):

        img_seq = iaa.Sequential([

             sometimes(iaa.AdditiveGaussianNoise(loc=0, scale=(0.0, 0.03*255), per_channel=0.5)),
             sometimes(iaa.Sharpen(alpha=(0, 0.2), lightness=(0.1, 0.4))), # sharpen images
             sometimes(iaa.Emboss(alpha=(0, 0.3), strength=(0, 0.5))), # emboss images
            ],random_order=True)

        self.aug_pipeline = Compose([
            RandomBrightness(16), #make image brighter or darker
            ImgAugSequence(img_seq),
        ])

    def __call__(self, image,  mask_0,mask_1,mask_2):
        image = self.aug_pipeline(image)
        return image
```

```python
class ImgAugSequence(object):
    def __init__(self, sequence):
        self.sequence = sequence

    def __call__(self, image):
        image = self.sequence.augment_image(image)

        return image

class Compose(object):
    def __init__(self, transforms):
        self.transforms = transforms

    def __call__(self, image):
        for t in self.transforms:
            image = t(image)
        return image
```

Imgaug-pipeline

CUPOY

另一個常見應用在 Semantic segmentation 中，我們希望標籤Mask跟輸入影像有相同的 Augmentation(如平移、縮放)，這時候要加上，**to_deterministic()**，意思是**將隨機性關閉**，使訓練影像與其標籤(mask)有相同的Augmentation。(什麼是 Semantic segmentation?)

```
class MaskAugSequence(object):
    def __init__(self, sequence):
        self.sequence = sequence

    def __call__(self, image, mask):
        sequence = self.sequence.to_deterministic()
        image = sequence.augment_image(image)
        mask = sequence.augment_image(mask)
            return image, mask
```

解題時間 Let’s Crack It

CUPOY

請跳出 PDF 至官網 Sample Code & 作業開始解題