

HW02 Classification

1. 前處理

- Train/Valid [改變的機率值為 0.1]

尺寸調整	Resize((224, 224))
將圖像轉為 Tensor	ToTensor()
空間操作	RandomHorizontalFlip()
	RandomVerticalFlip()
	RandomRotation(10)
	RandomPerspective(distortion_scale=0.6, p=1.0)
	RandomAffine(degrees=(30, 70), translate=(0.1, 0.3), scale=(0.5, 0.75))
	ElasticTransform(alpha=250.0)
顏色操作	ColorJitter(brightness=0.4, contrast=0.4, saturation=0.4, hue=0.1)
	RandomGrayscale(p=0.1)
	RandomInvert(p=0.1)
	RandomPosterize(bits=2, p=0.1)
	RandomSolarize(threshold=1.0)
噪聲操作	AddGaussianNoise(mean=0.0, std=0.05)
	AddPoissonNoise(lam=0.1)
	AddSpeckleNoise(noise_level=0.1)
	AddSaltPepperNoise(salt_prob=0.05, pepper_prob=0.05)
高斯模糊	GaussianBlur(kernel_size=(5, 9), sigma=(0.1, 5.))
標準化 Normalize [-1,1]	Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

- Test

尺寸調整	Resize((224, 224))
將圖像轉為 Tensor	ToTensor()
標準化 Normalize [-1,1]	Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])

補充: 由於測試圖片加入了翻轉變形、改變顏色跟雜訊干擾等等變化，以此在前處理時訓練及驗證圖片也要加入這些圖片變化操作，確保後續訓練能獲的更好的模型效果

2. 訓練模型選擇

- 先前: 自己寫的 CNN 模型
 - 使用自己寫的訓練模型，效果很差，精準度在 0.02~0.03 徘徊，比 baseline 要求的 0.05 還要低，距離最終目標還有很多路要走。

```
class SimpleCNN(nn.Module):
    def __init__(self, numClasses):
        super(SimpleCNN, self).__init__()
        self.features = nn.Sequential(
            nn.Conv2d(3, 32, kernel_size=3, stride=1, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2),
            nn.Conv2d(64, 128, kernel_size=3, padding=1),
            nn.ReLU(inplace=True),
            nn.MaxPool2d(kernel_size=2, stride=2)
        )
        self.classifier = nn.Sequential(
            nn.Linear(128*28*28, 512), # 28*28 from the image size (224/2/2/2 = 28)
            nn.ReLU(inplace=True),
            nn.Linear(512, numClasses),
        )

    def forward(self, x):
        x = self.features(x)
        x = x.view(x.size(0), -1)
        x = self.classifier(x)
        return x
```

- 最終: 套用 CNN pre-trained 模型 (特徵提取)
 - 使用老師提供的 [TRANSFER LEARNING FOR COMPUTER VISION TUTORIAL](#)，裡面有一個部分關於 pre-trained image classification models，有許多 pre-trained 可供使用，我使用幾個 CP 值較高的模型，以及舉例上有使用的“resnet50”以下表個的模型作為訓練模型使用。

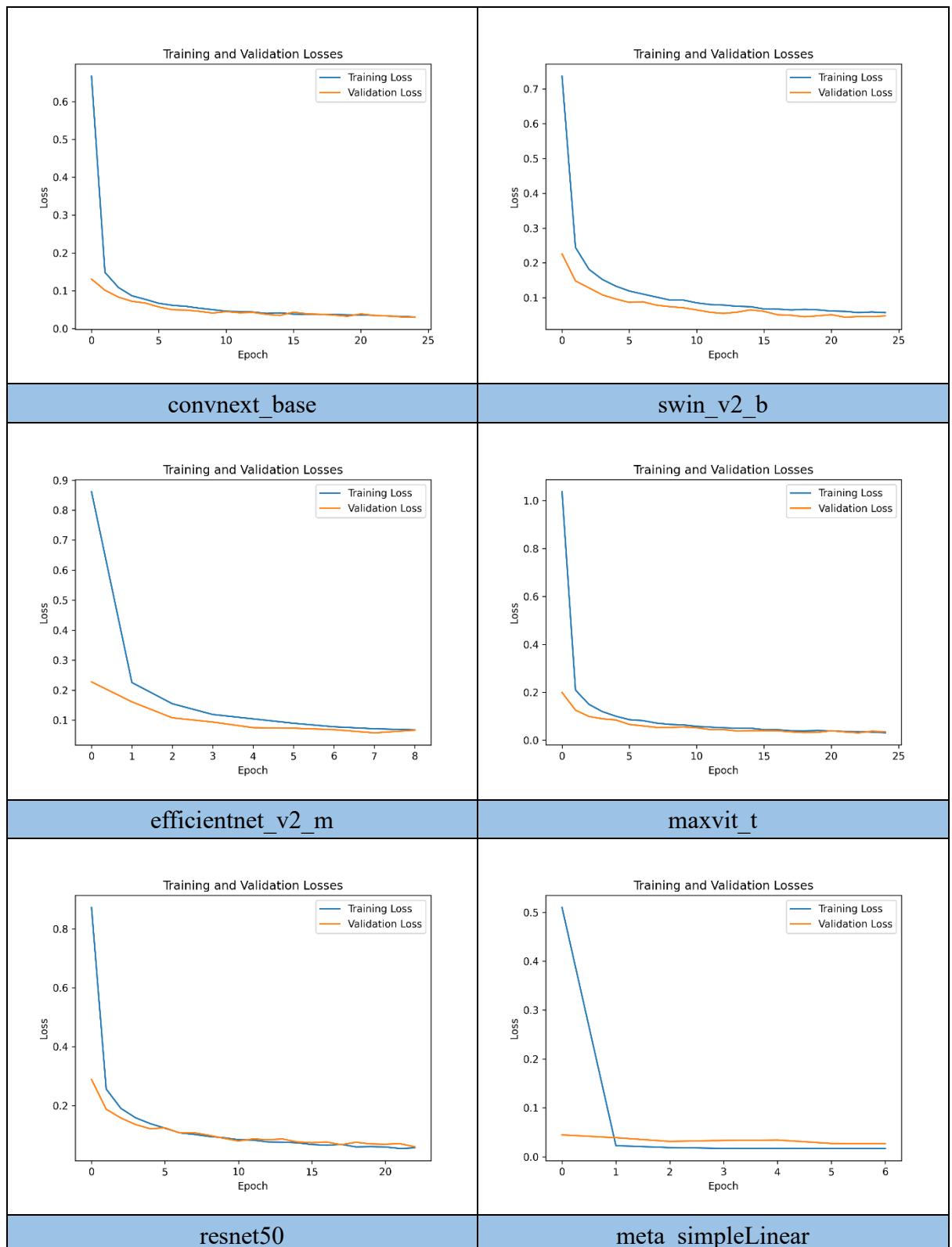
Pre-trained model	
convnext_base	swin_v2_b
efficientnet_v2_m	maxvit_t
resnet50	

- 做完 Pre-trained model 後，再套一層自己的 **linear model**，將 CNN 模型的特徵映射到最後的分類空間，題目設計是要辨識出 50 隻角色，所以這裡的“分類空間=50”

Pre-trained model
meta_simpleLinear

```
class SimpleLinear(nn.Module):
    def __init__(self, input_dim, num_classes):
        super(SimpleLinear, self).__init__()
        self.fc = nn.Linear(input_dim, 3 * input_dim)
        self.relu = nn.ReLU()
        self.fc2 = nn.Linear(3 * input_dim, num_classes)
        init.kaiming_uniform_(self.fc.weight, nonlinearity='relu')
    def forward(self, x):
        return self.fc2(self.relu(self.fc(x)))
```

- 訓練與驗證的損失率 Loss



3. 環境變數

- .env:

第一次這樣寫，好處是將模型名稱，資料夾路徑等等，定義再一起，方便管理，想換成其他 pre-trained 可以直接從這裡進行修改，不用每個檔案有用的地方都改。

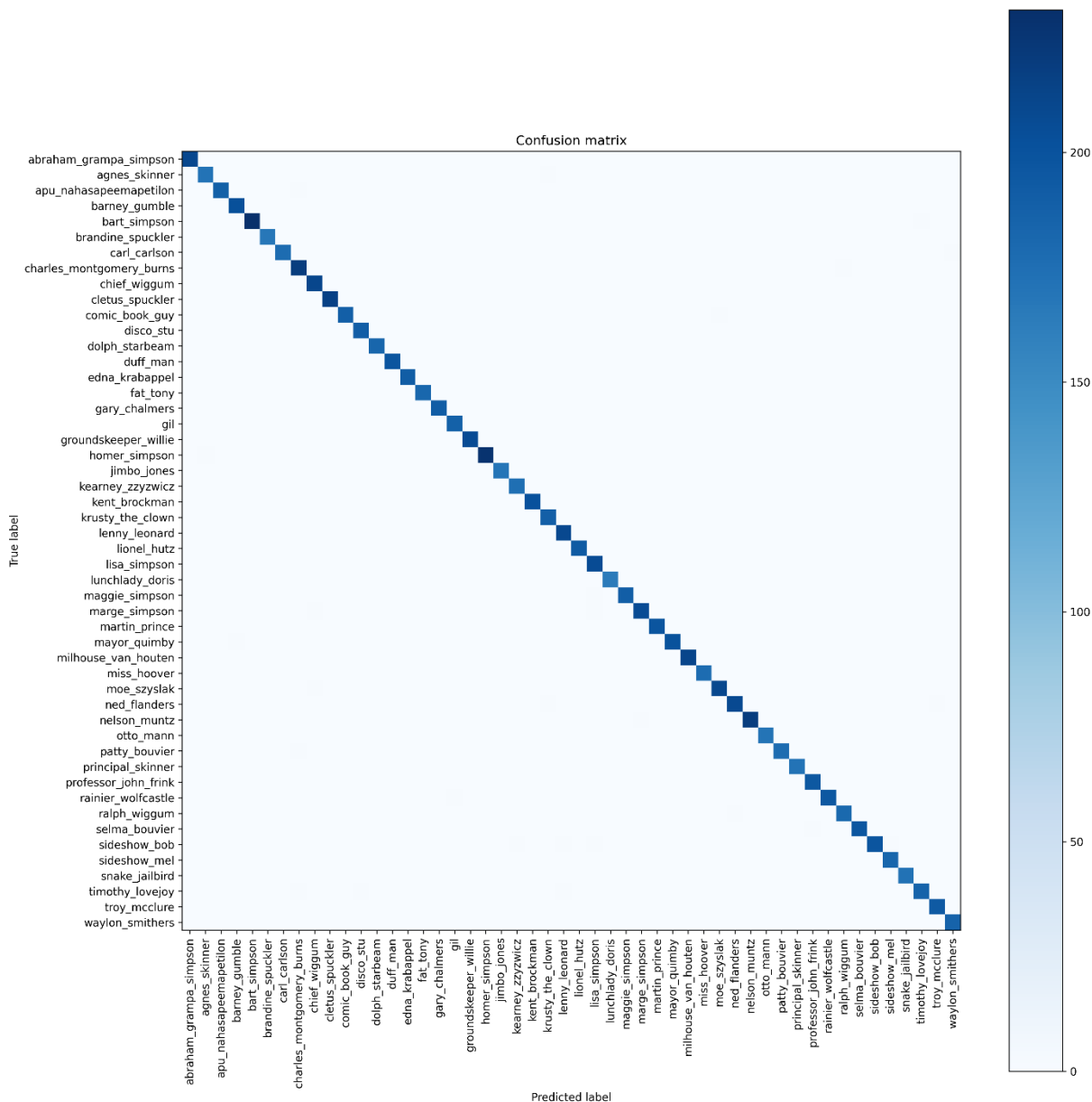
- 使用方法:

下載: `pip install python-dotenv`

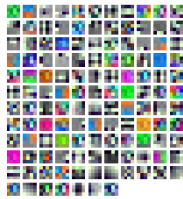
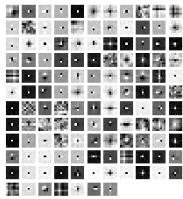
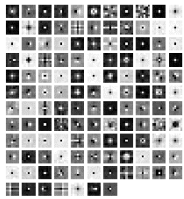
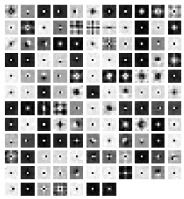
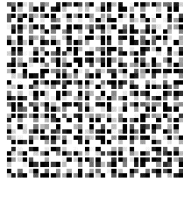
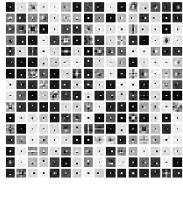


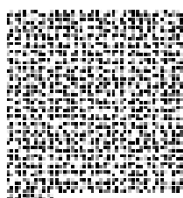
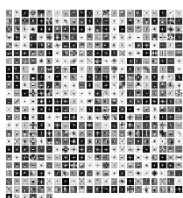
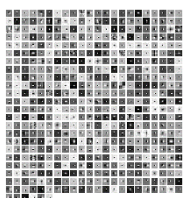
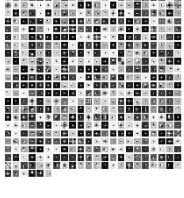
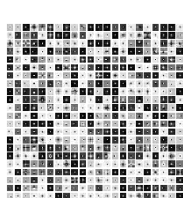
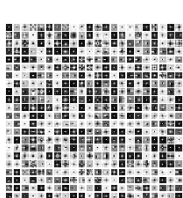
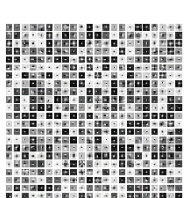
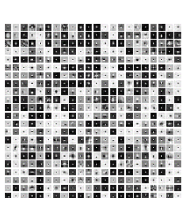
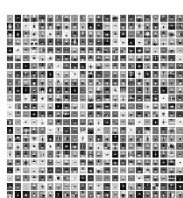
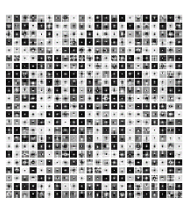
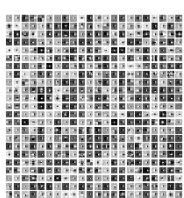
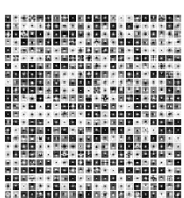
```
from dotenv import load_dotenv
load_dotenv()

TRAIN_DATA_DIR = os.getenv('TRAIN_DATA_DIR')
TEST_DATA_DIR = os.getenv('TEST_DATA_DIR')
TRAIN_VAL_RATIO = float(os.getenv('TRAIN_VAL_RATIO'))
BATCH_SIZE = int(os.getenv('BATCH_SIZE'))
```

4. 混淆矩陣(Confusion Matrix)



5. 每層權重(Layer filter)

			
			
			
			
			
..... (省略)			
..... (省略)			
