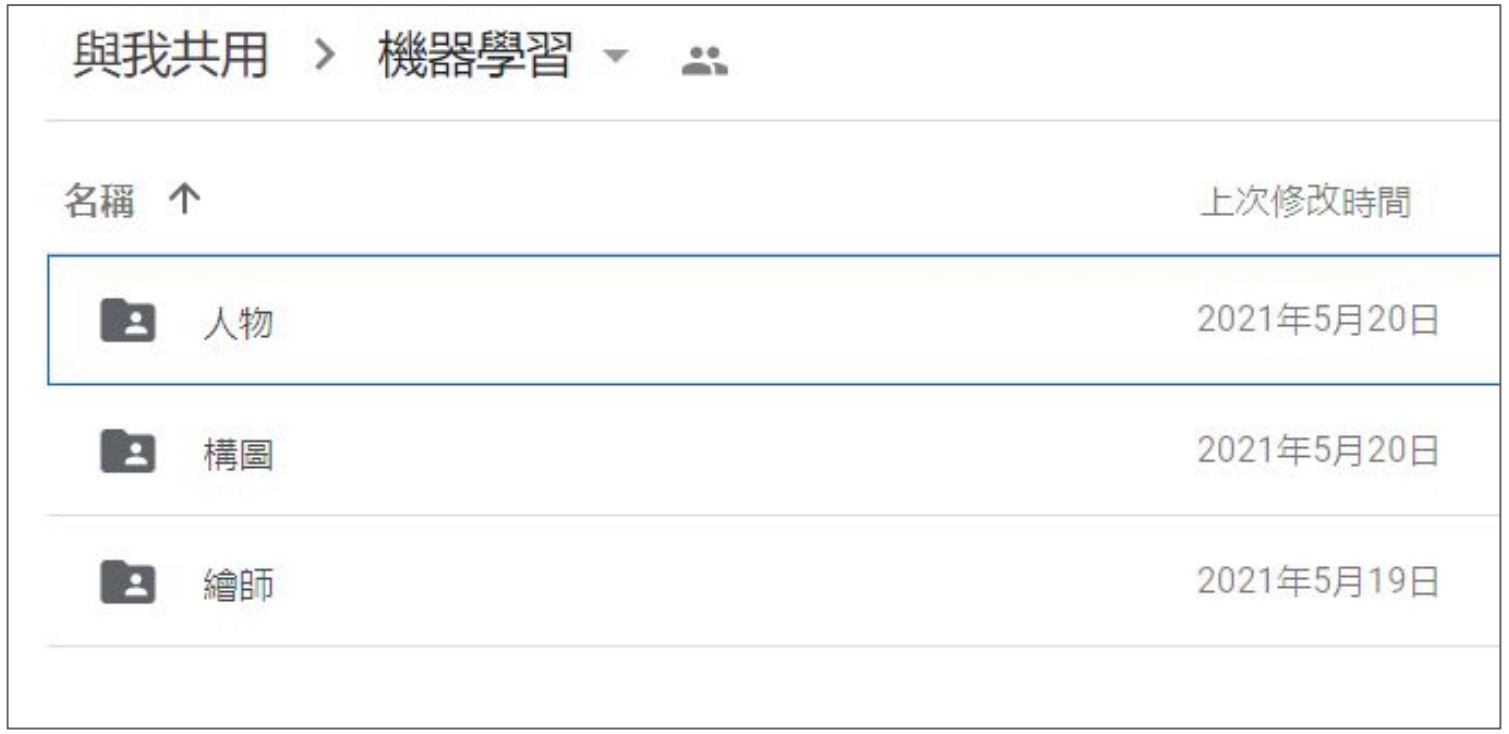
機器學習

目錄

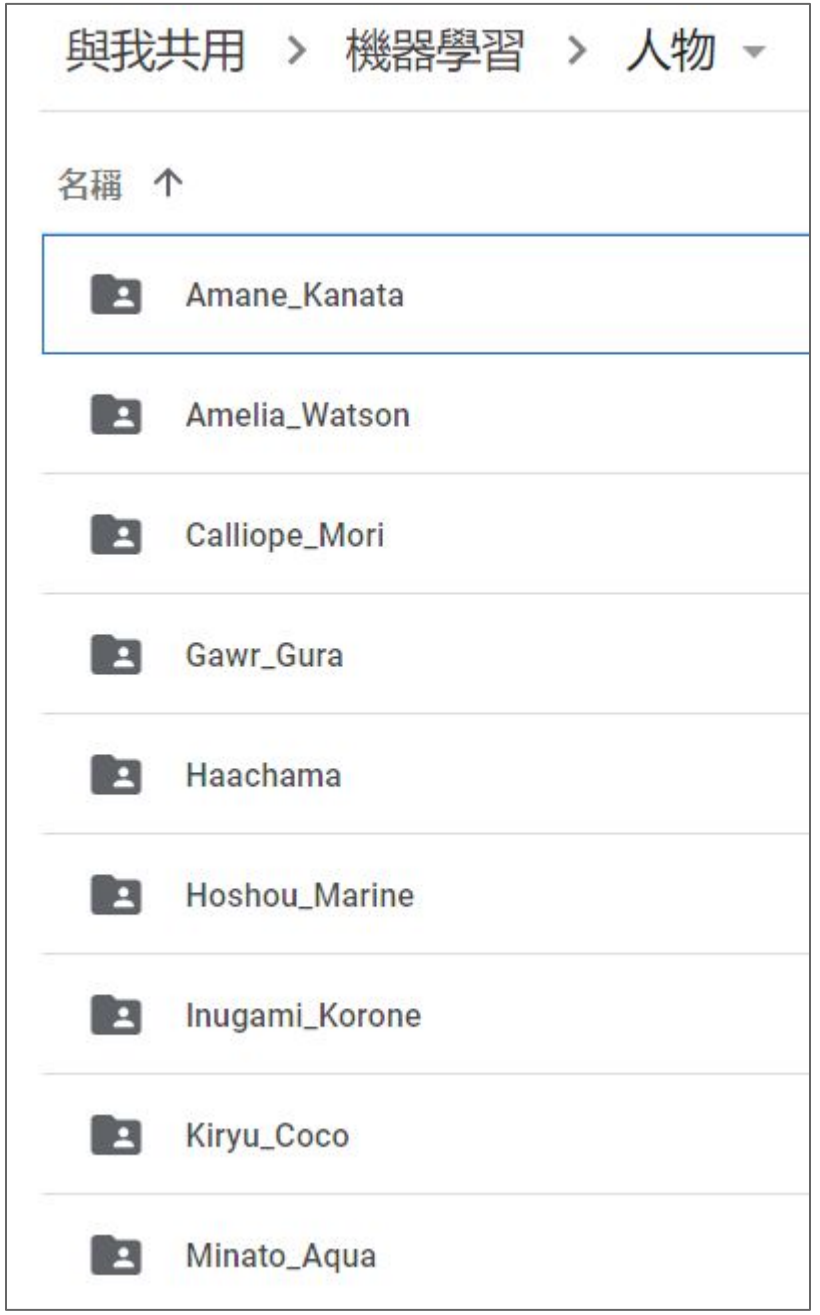
一、事前準備



時間範圍：2021年5月、6月
蒐集位置：twitter, pixiv
訓練資料圖數量：20+

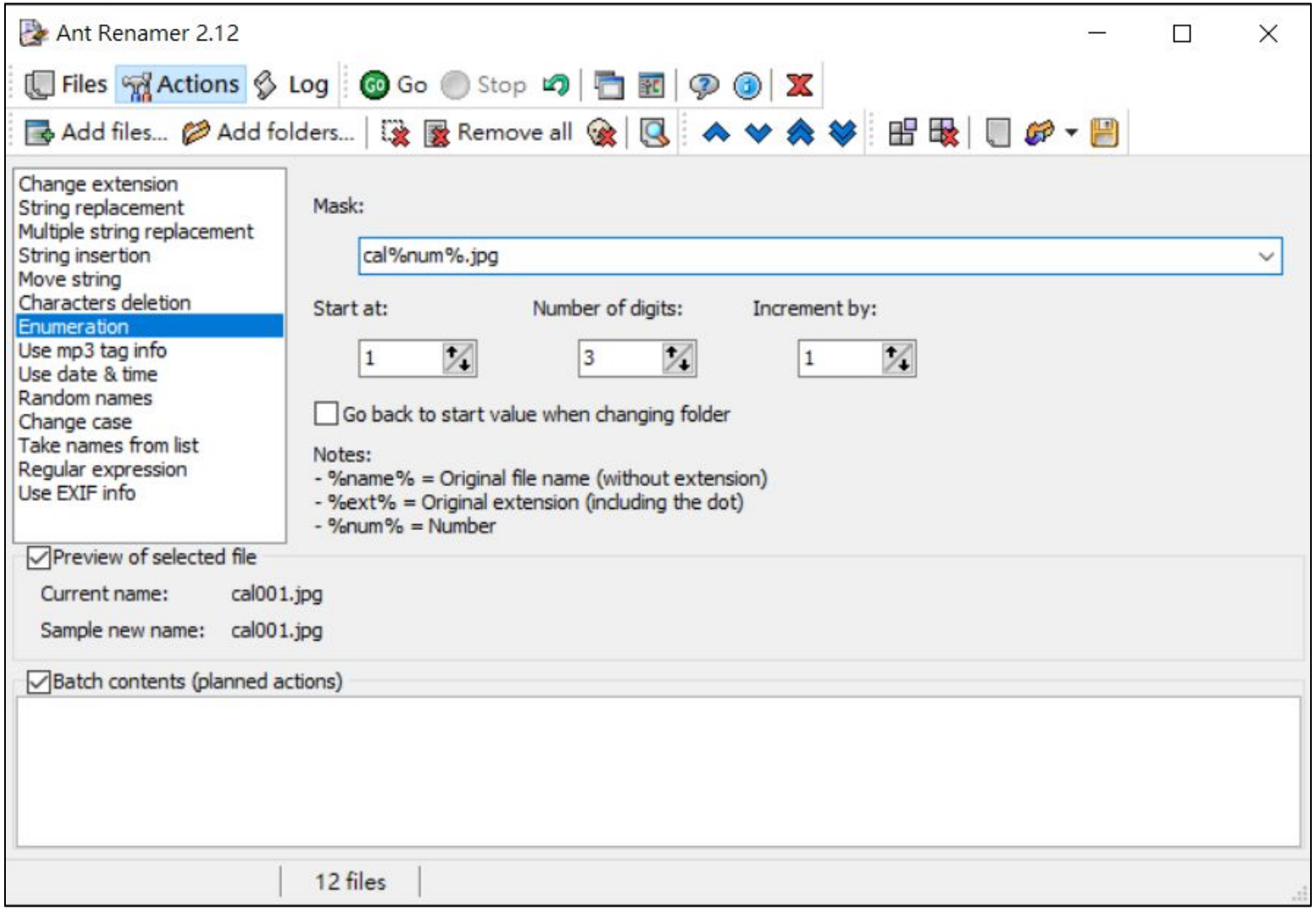
訓練資料集


與我共用 > 機器學習 > 繪師
名稱
古塔TSUMI
米山舞
焦茶
Daisuke Richard
Zeen Chin


與我共用 > 機器學習 > 人物
名稱
Amane_Kanata
Amelia_Watson
Calliope_Mori
Gawr_Gura
Haachama
Hoshou_Marine
Inugami_Korone
Kiryu_Coco
Minato_Aqua


與我共用 > 機器學習 > 構圖
名稱
0未分類
九宮格
三角
中央
支點/對比
放射狀
斜線
對稱
L或倒T型

實用小工具: ant renamer



二、分析模型

- LeNet
  - 早期模型，由卷積層、池化層、全連接層以及最後一層 Gaussian 連接層所組成
- AlexNet
  - 上課使用之模型，與前者不同為用Dropout抑制過擬合，與使用ReLU減少梯度消失現象
- VGGNet
  - 與AlexNet類似，但VGG增加一系列大小為3x3的小尺寸卷積核和pooling層，取得更好的效果
- Inception (GoogLeNet)
- WGAN
- YoloV3 (未完成)

三、結果分析

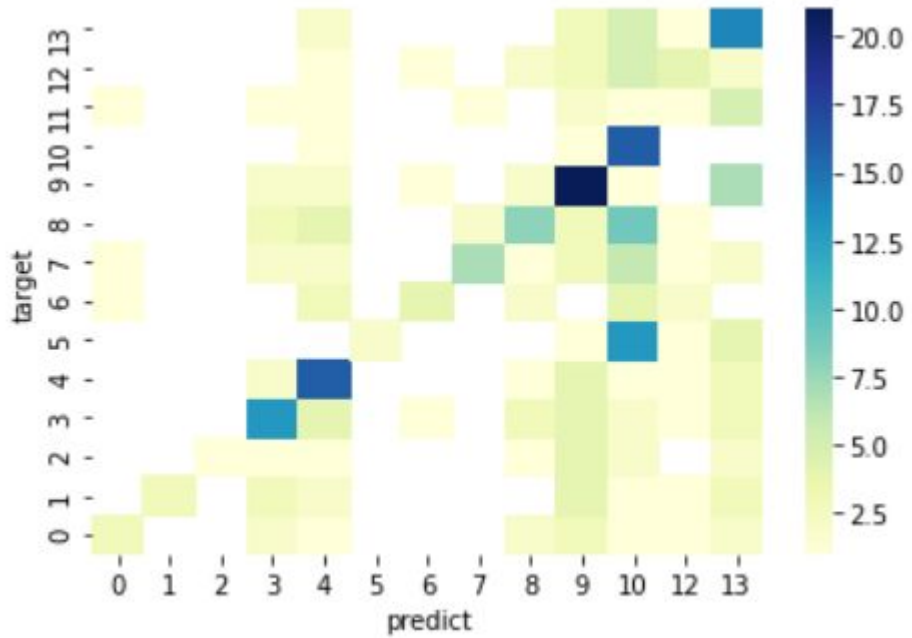
- 人物角色辨識
  - 針對全體的基本分析
  - 針對部分的基本分析，並提升精準度
- 畫風辨識
  - 基本分析
  - Inception
  - WGAN
  - YoloV3

CNN-人物角色辨識

入力 [52]:
```
# 針對 training set, 如果有 test set 可比較
print (CATEGORIES)
acc_matrix(model,X, y)
```

```
['Amane_Kanata', 'Amelia_Watson', 'Calliope_Mori', 'Gawr_Gura', 'Haachama', 'Hoshou_Marine', 'Inugami_Korone', 'Kiryu_Coco', 'M
inato_Aqua', 'Nakiri_Ayame', 'Natsuiro_Matsuri', 'Nekomata_Okayu', 'Sakura_Miko', 'Uruha_Rushia']
```

predict	0	1	2	3	4	5	6	7	8	9	10	12	13
target													
0	3.0	NaN	NaN	2.0	1.0	NaN	NaN	NaN	2.0	3.0	1.0	1.0	2.0
1	NaN	3.0	NaN	3.0	2.0	NaN	NaN	NaN	NaN	4.0	1.0	1.0	3.0
2	NaN	NaN	1.0	1.0	1.0	NaN	NaN	NaN	1.0	4.0	2.0	NaN	2.0
3	NaN	NaN	NaN	13.0	4.0	NaN	1.0	NaN	3.0	4.0	2.0	1.0	3.0
4	NaN	NaN	NaN	2.0	16.0	NaN	NaN	NaN	1.0	4.0	1.0	1.0	3.0
5	NaN	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	1.0	13.0	1.0	4.0
6	1.0	NaN	NaN	NaN	3.0	NaN	4.0	NaN	2.0	NaN	4.0	2.0	NaN
7	1.0	NaN	NaN	2.0	2.0	NaN	NaN	7.0	1.0	3.0	6.0	1.0	2.0
8	NaN	NaN	NaN	3.0	4.0	NaN	NaN	2.0	8.0	3.0	9.0	1.0	NaN
9	NaN	NaN	NaN	2.0	2.0	NaN	1.0	NaN	2.0	21.0	1.0	NaN	7.0
10	NaN	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	1.0	16.0	NaN	NaN
11	1.0	NaN	NaN	1.0	1.0	NaN	NaN	1.0	NaN	2.0	1.0	1.0	5.0
12	NaN	NaN	NaN	NaN	1.0	NaN	1.0	NaN	2.0	3.0	5.0	4.0	2.0
13	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	3.0	5.0	1.0	14.0



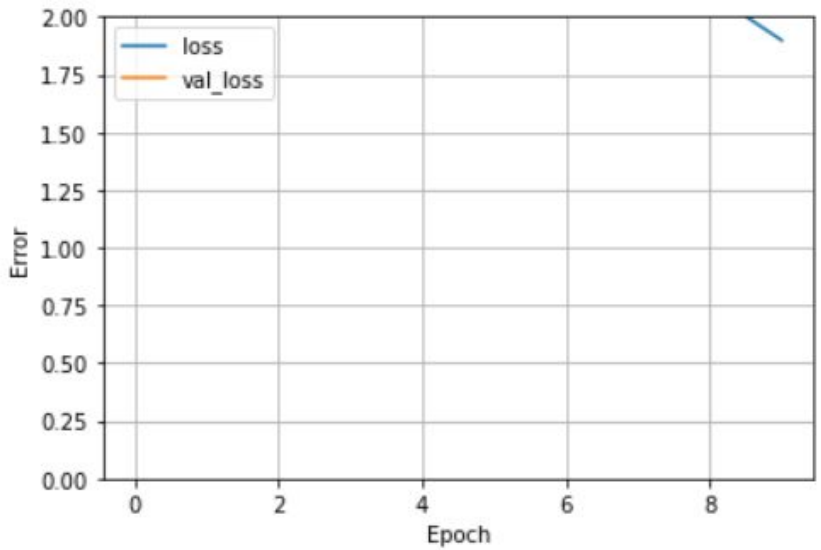
入力 [53]:

```
import pandas as pd
# history 轉為 dataframe 格式
hist = pd.DataFrame(history.history)
# 新增 epoch 欄位
hist['epoch'] = history.epoch
# 顯示 epoch, loss, val_loss
hist.tail()
```

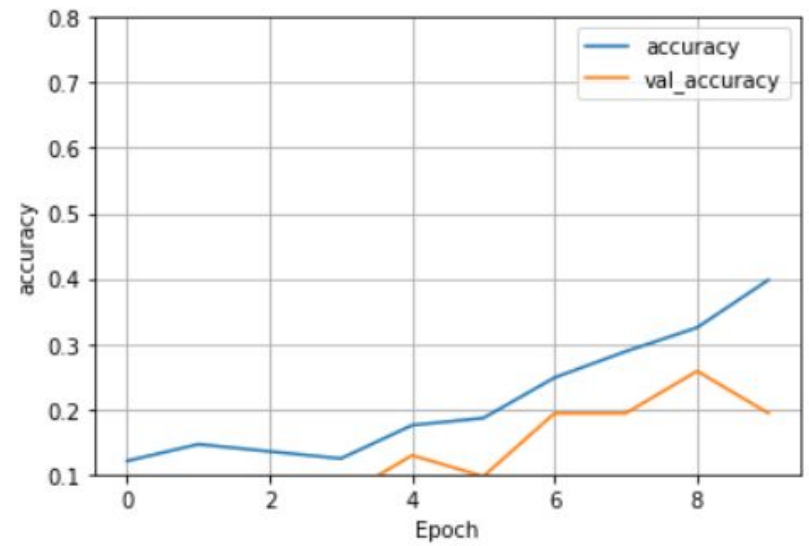
出力[53]:

	loss	accuracy	val_loss	val_accuracy	epoch
5	2.426796	0.186131	2.509262	0.096774	5
6	2.333289	0.248175	2.443762	0.193548	6
7	2.245581	0.288321	2.477624	0.193548	7
8	2.106127	0.324818	2.501650	0.258065	8
9	1.897979	0.397810	2.555658	0.193548	9

入力 [54]:
```
# 繪圖，顯示損失函數下降的趨勢
def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss')
  plt.ylim([0, 2])
  plt.xlabel('Epoch')
  plt.ylabel('Error')
  plt.legend()
  plt.grid(True)
plot_loss(history)
```



[55]:
```
# 繪圖，顯示正確率上升的趨勢
def plot_acc(history):
  plt.plot(history.history['accuracy'], label='accuracy')
  plt.plot(history.history['val_accuracy'], label='val_accuracy')
  plt.ylim([0.1, 0.8])
  plt.xlabel('Epoch')
  plt.ylabel('accuracy')
  plt.legend()
  plt.grid(True)
plot_acc(history)
```



jupyter TF4_CNN_faces (2) (自動保存)

Logout

File Edit View Insert Cell Kernel Widgets Help

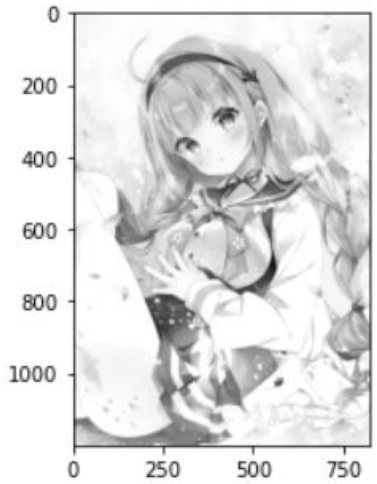
信頼されていません | Python 3

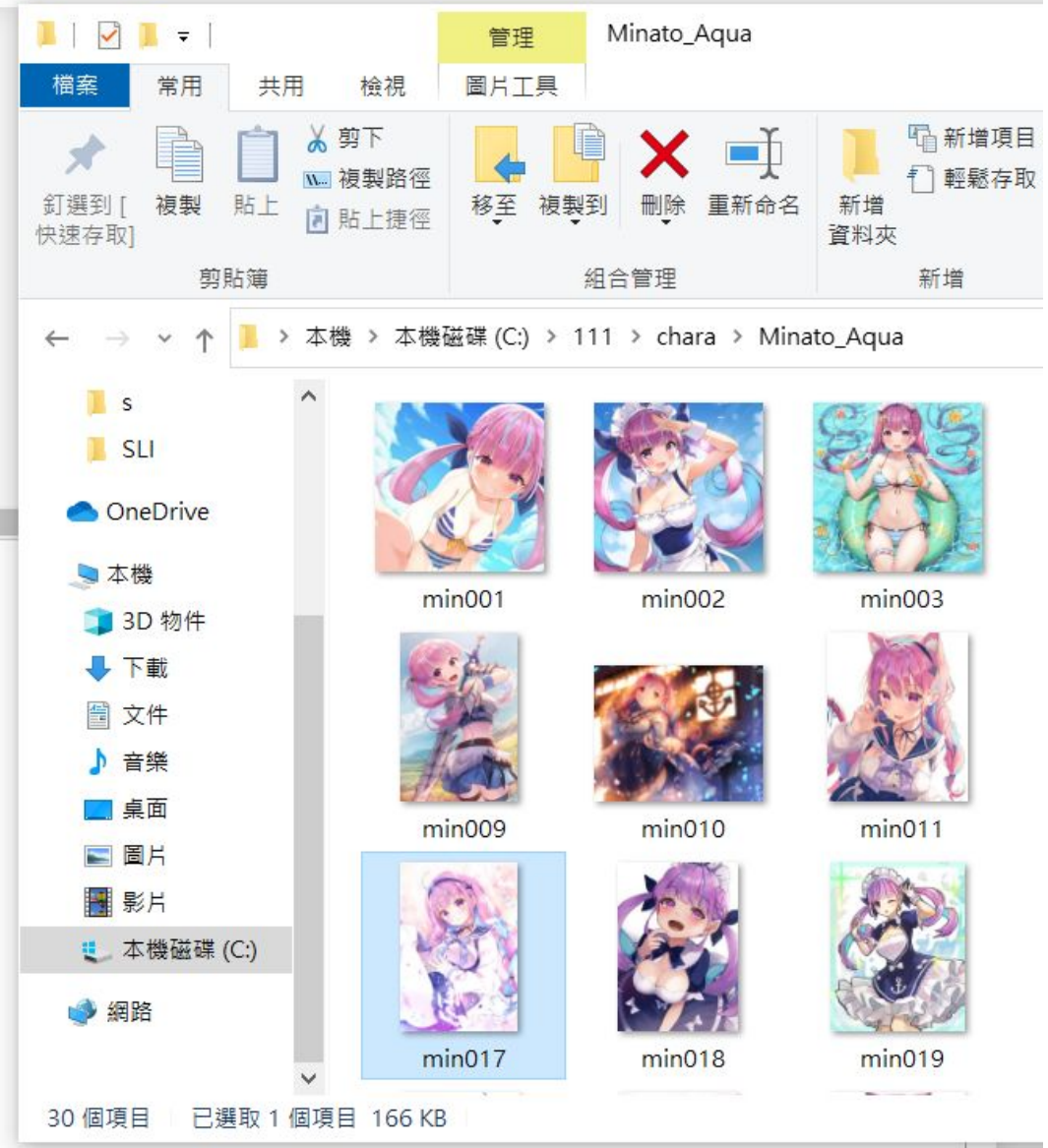
▶ 実行 コード

```
# float
new_array=new_array.astype(float)
# tensor, creat a batch 目的是與 model 訓練時的 input shape 一致
img_tf = tf.expand_dims(new_array, 0)
# prediction
predictions = model.predict(img_tf)
# score，顯示各類別的機率
score =predictions[0]

print (filepath)
print (score)
print("這應該是 {} 有百分之 {:.0f} 的信心 "
    .format(CATEGORIES[np.argmax(score)], 100 * np.max(score)))
plt.imshow(img_array, cmap='gray')
plt.show()
```

```
C:/111/chara/Minato_Aqua/min017.jpg
[0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
這應該是 Haachama 有百分之 100 的信心
```

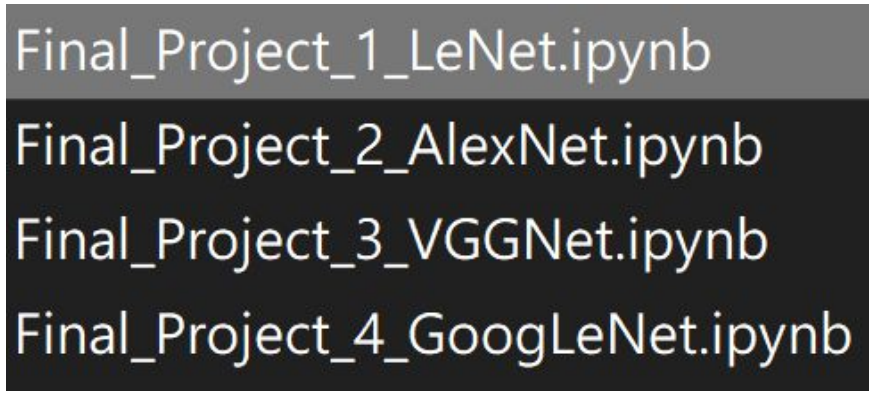




很顯然的....我們的模型出了點問題.....

解決問題方向 :

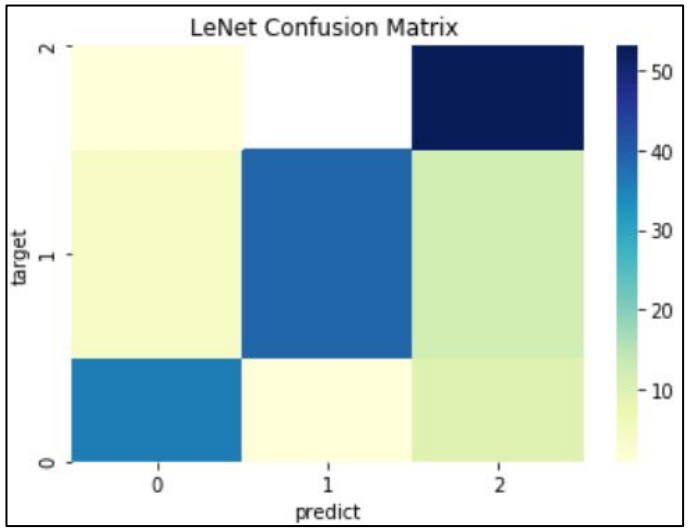
- 演算法

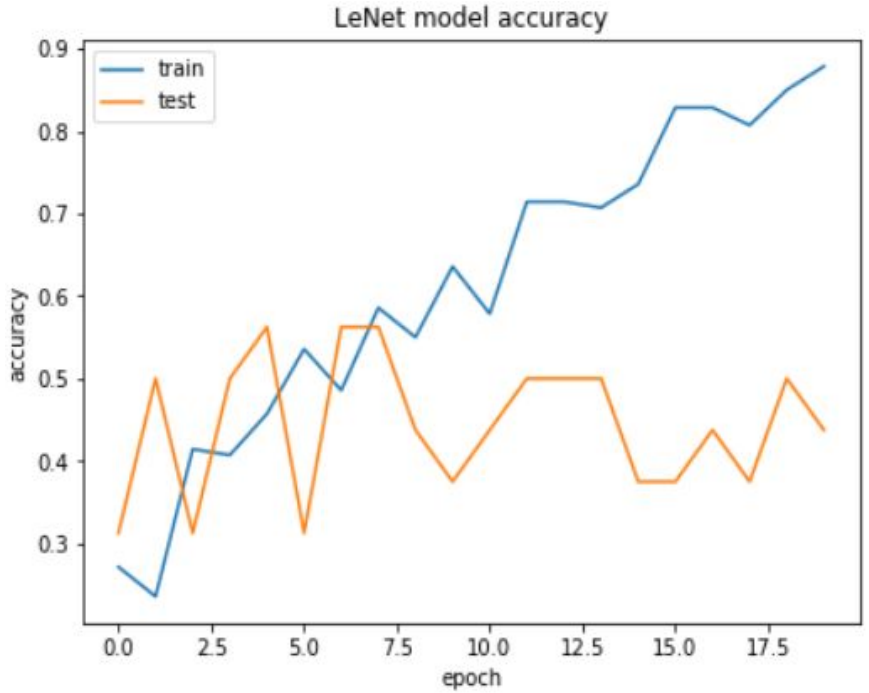


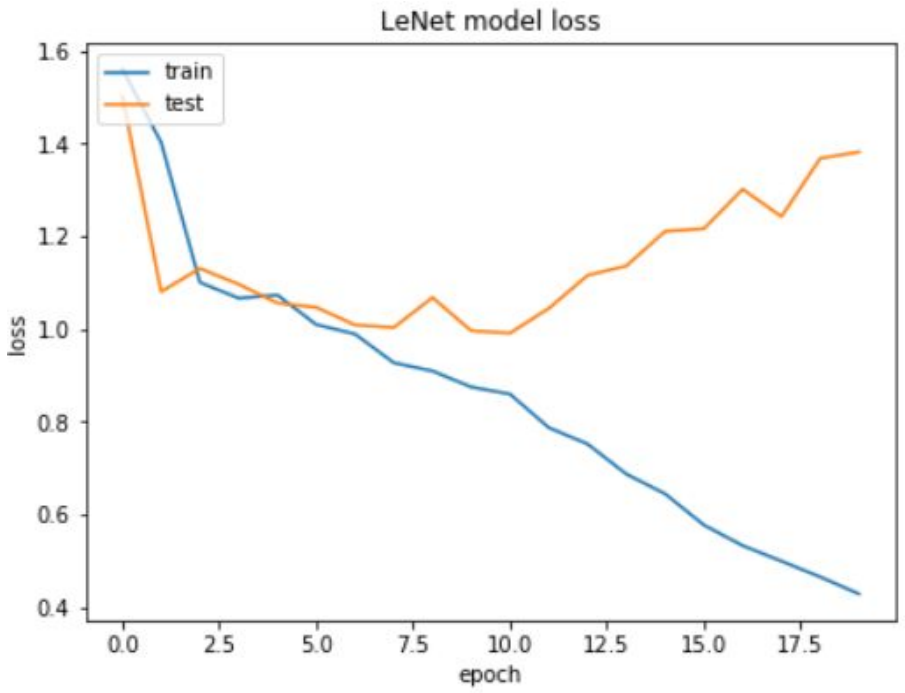
- 圖檔品質與數量
  - 針對Minato_Aqua, Nakiri_Ayame, Uruha_Rushia增加圖片數量 (60以上)
  - 提升轉換成pickledata的圖片檔的解析度 (300 x 300)

LeNet

predict target	0	1	2
0	36.0	1.0	10.0
1	4.0	39.0	12.0
2	1.0	NaN	53.0

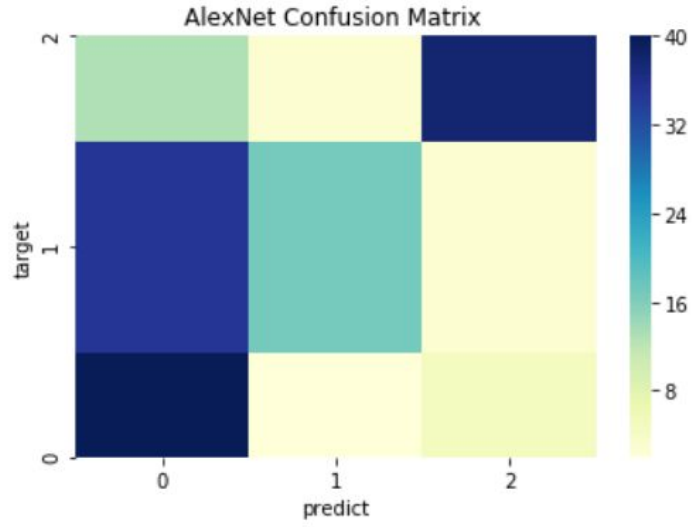


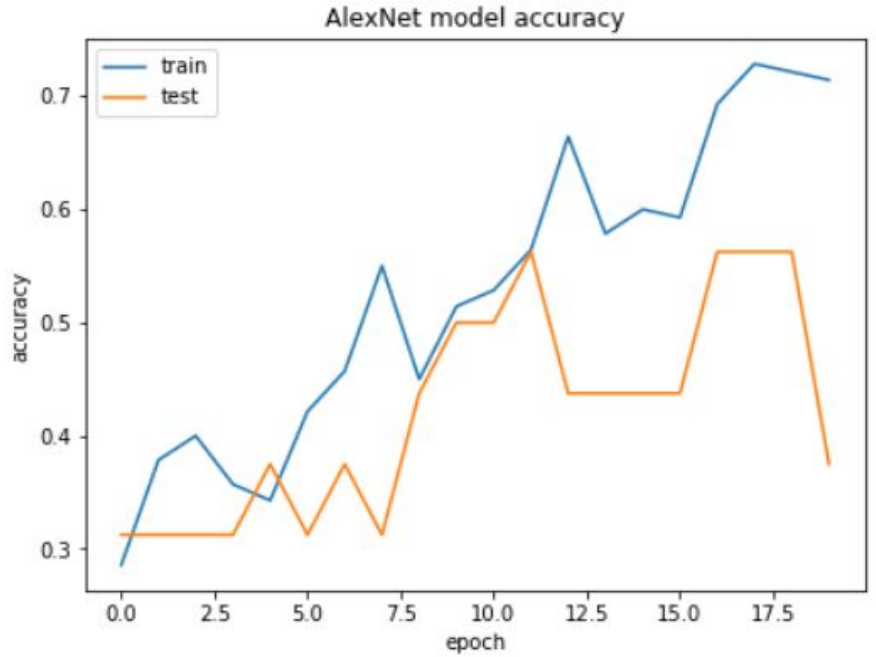


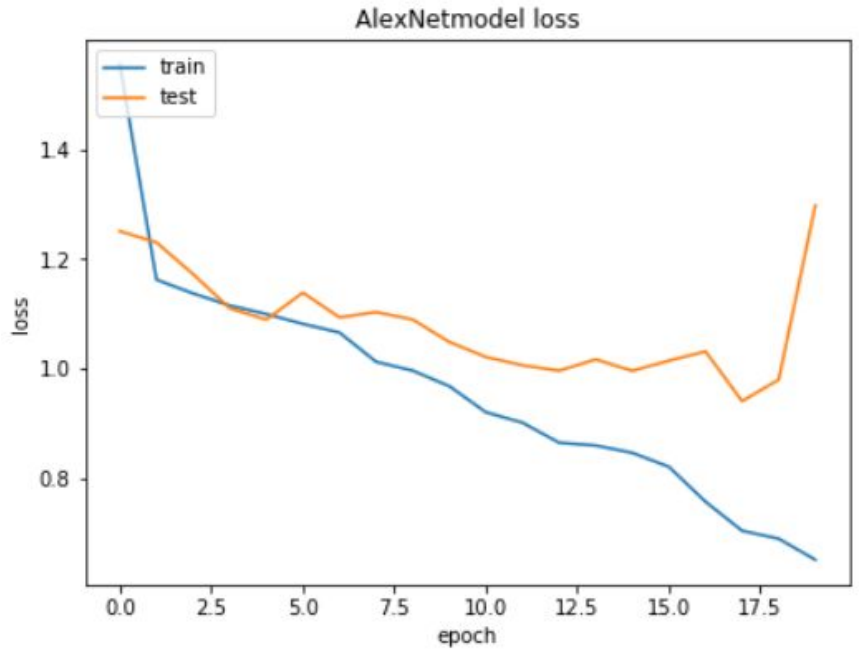


AlexNet

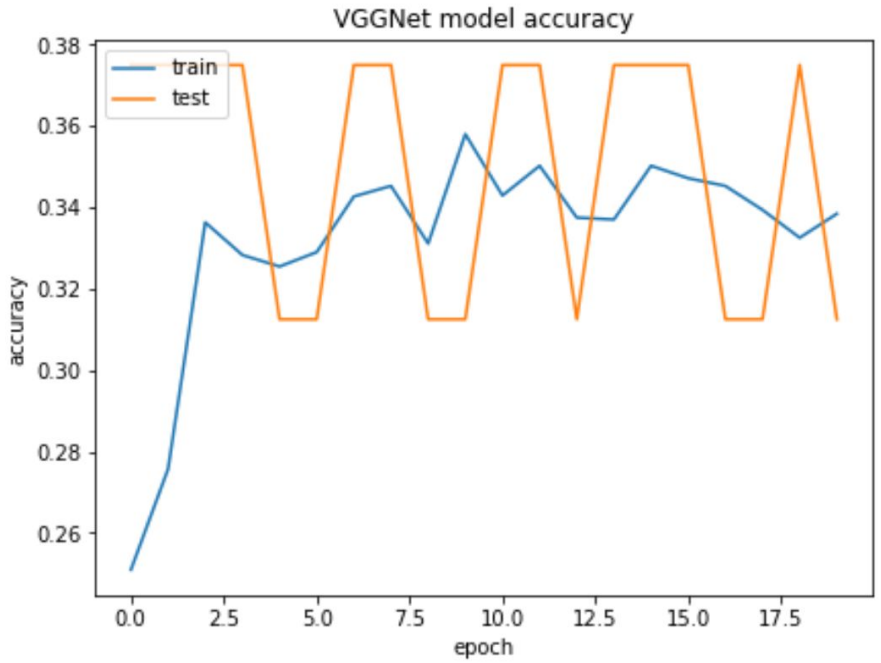
predict	0	1	2
target			
0	40	2	5
1	35	17	3
2	13	3	38

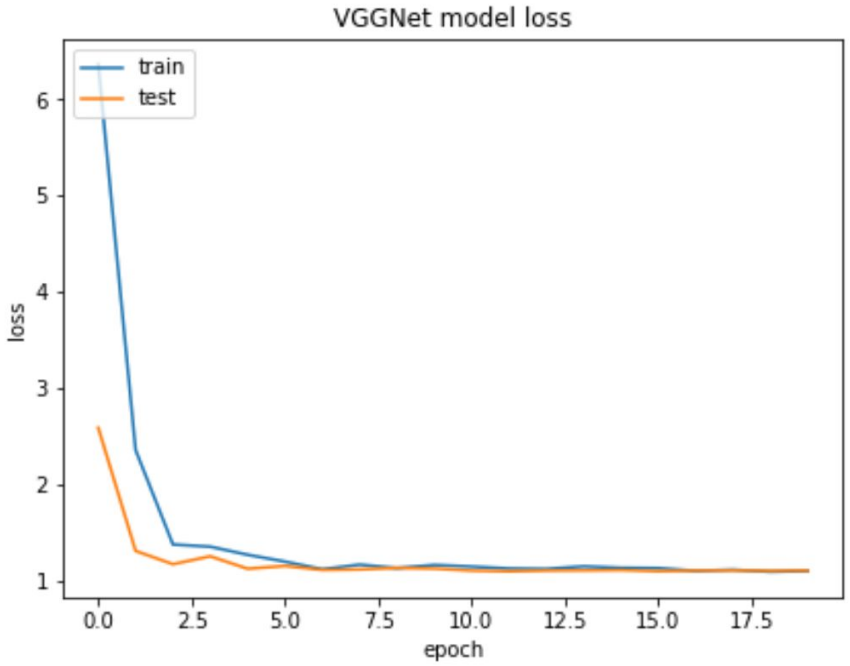






VggNet


VGGNet model accuracy
train
test
accuracy
epoch
0.38
0.36
0.34
0.32
0.30
0.28
0.26
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5


VGGNet model loss
train
test
loss
epoch
6
5
4
3
2
1
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

CNN:畫風辨識

- 與預期不同: 比角色好辨識許多

fit (這要花一點時間)

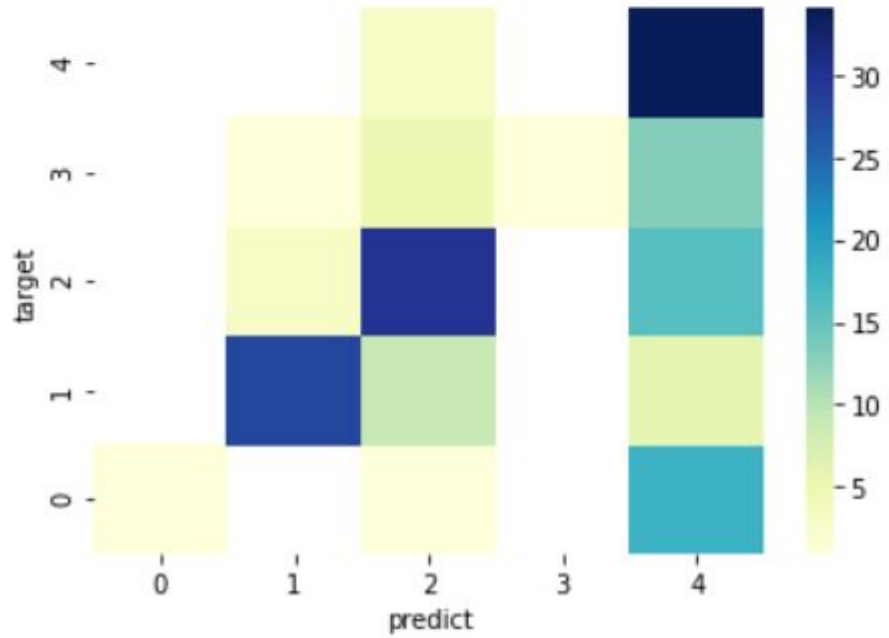
入力 [28]:
```
history =model.fit(X,y, batch_size=32, epochs=10,validation_split=0.1,verbose=2)
```

```
Epoch 1/10
5/5 - 1s - loss: 1.8202 - accuracy: 0.1711 - val_loss: 1.8794 - val_accuracy: 0.1176
Epoch 2/10
5/5 - 0s - loss: 1.6196 - accuracy: 0.2697 - val_loss: 1.6643 - val_accuracy: 0.4706
Epoch 3/10
5/5 - 0s - loss: 1.5529 - accuracy: 0.3553 - val_loss: 1.7622 - val_accuracy: 0.1176
Epoch 4/10
5/5 - 0s - loss: 1.5400 - accuracy: 0.3487 - val_loss: 1.6074 - val_accuracy: 0.2353
Epoch 5/10
5/5 - 0s - loss: 1.5218 - accuracy: 0.2961 - val_loss: 1.5434 - val_accuracy: 0.3529
Epoch 6/10
5/5 - 0s - loss: 1.4579 - accuracy: 0.3947 - val_loss: 1.4877 - val_accuracy: 0.5294
Epoch 7/10
5/5 - 0s - loss: 1.3775 - accuracy: 0.4934 - val_loss: 1.3601 - val_accuracy: 0.5294
Epoch 8/10
5/5 - 0s - loss: 1.2956 - accuracy: 0.4934 - val_loss: 1.4185 - val_accuracy: 0.5294
Epoch 9/10
5/5 - 0s - loss: 1.2625 - accuracy: 0.5000 - val_loss: 1.2390 - val_accuracy: 0.6471
Epoch 10/10
5/5 - 0s - loss: 1.2190 - accuracy: 0.5132 - val_loss: 1.2336 - val_accuracy: 0.6471
```

```
print (CATEGORIES)
acc_matrix(model,X, y)
```

```
['Daisuke_Richard', 'jiaocha', 'mishanwu', 'TSUMI', 'Zeen_Chin']
```

predict	0	1	2	3	4
target					
0	1.0	NaN	1.0	NaN	18.0
1	NaN	28.0	9.0	NaN	6.0
2	NaN	3.0	30.0	NaN	16.0
3	NaN	1.0	5.0	1.0	13.0
4	NaN	NaN	3.0	NaN	34.0



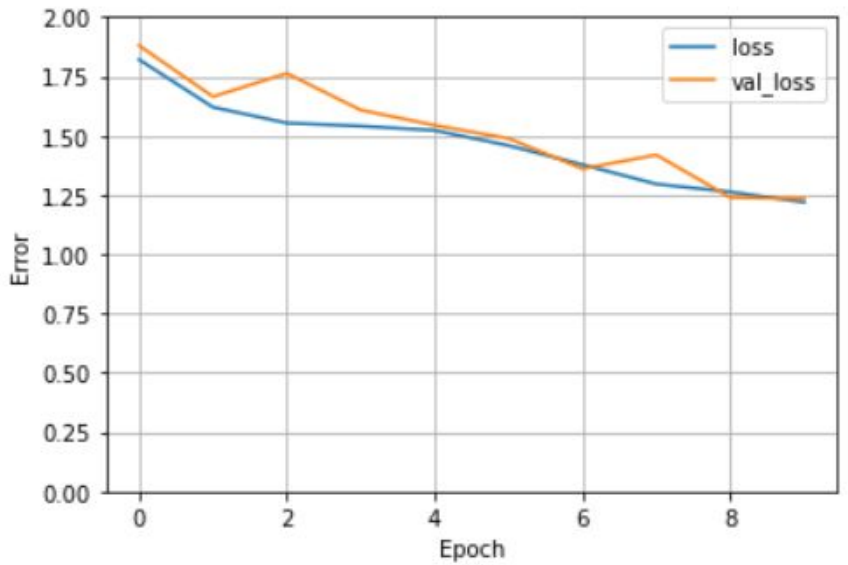
入力 [31]:

```
import pandas as pd
# history 轉為 dataframe 格式
hist = pd.DataFrame(history.history)
# 新增 epoch 欄位
hist['epoch'] = history.epoch
# 顯示 epoch, loss, val_loss
hist.tail()
```

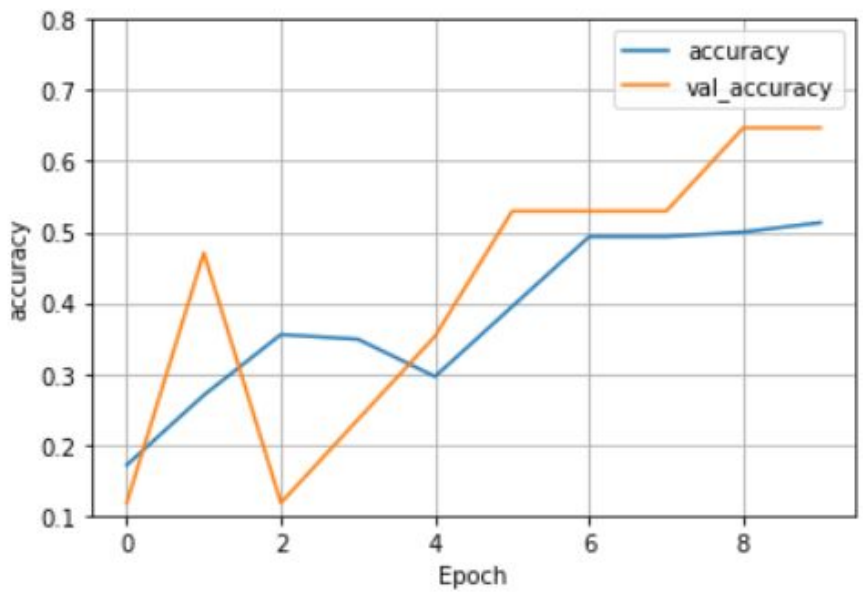
出力[31]:

	loss	accuracy	val_loss	val_accuracy	epoch
5	1.457915	0.394737	1.487714	0.529412	5
6	1.377530	0.493421	1.360143	0.529412	6
7	1.295563	0.493421	1.418492	0.529412	7
8	1.262537	0.500000	1.239008	0.647059	8
9	1.219000	0.513158	1.233648	0.647059	9

入力 [32]:
```
# 繪圖，顯示損失函數下降的趨勢
def plot_loss(history):
  plt.plot(history.history['loss'], label='loss')
  plt.plot(history.history['val_loss'], label='val_loss
  plt.ylim([0, 2])
  plt.xlabel('Epoch')
  plt.ylabel('Error')
  plt.legend()
  plt.grid(True)
plot_loss(history)
```



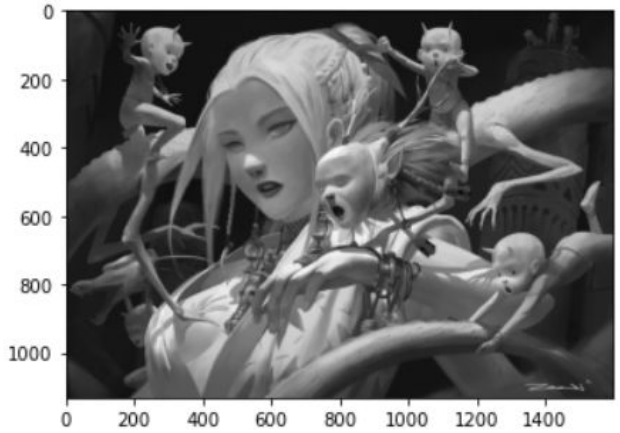
入力 [34]:
```
# 繪圖，顯示正確率上升的趨勢
def plot_acc(history):
  plt.plot(history.history['accuracy'], label='accuracy')
  plt.plot(history.history['val_accuracy'], label='val_accuracy')
  plt.ylim([0.1, 0.8])
  plt.xlabel('Epoch')
  plt.ylabel('accuracy')
  plt.legend()
  plt.grid(True)
plot_acc(history)
```

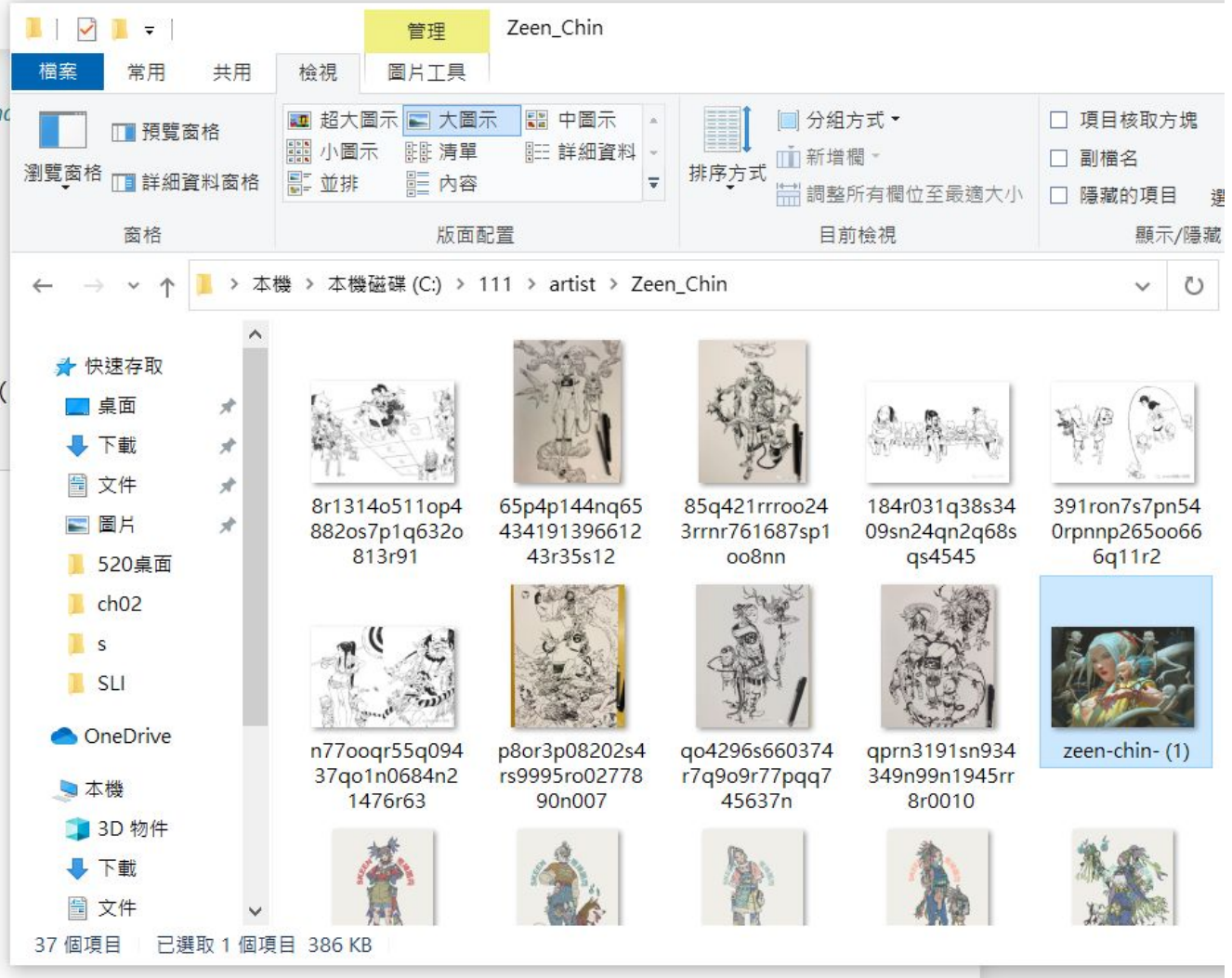


```
# float
new_array=new_array.astype(float)
# tensor, creat a batch 目的是與 model 訓練時的 input sha
img_tf = tf.expand_dims(new_array, 0)
# prediction
predictions = model.predict(img_tf)
# score，顯示各類別的機率
score =predictions[0]

print (filepath)
print (score)
print("這應該是 {} 有百分之 {:.0f} 的信心 "
    .format(CATEGORIES[np.argmax(score)], 100 * np.max(
plt.imshow(img_array, cmap='gray')
plt.show()
```

```
C:/111/artist/Zeen_Chin/zeen-chin- (1).jpg
[0. 0. 0. 0. 1. 0. 0.]
這應該是 Zeen_Chin 有百分之 100 的信心
```





結論: 正確率較高 還算適合此方面應用

InceptionV3

jupyter Untitled 最終チェックポイント: 上星期四06:16 (自動保存)

Logout

File Edit View Insert Cell Kernel Widgets Help 信頼済み Python 3

▶ 実行 コード

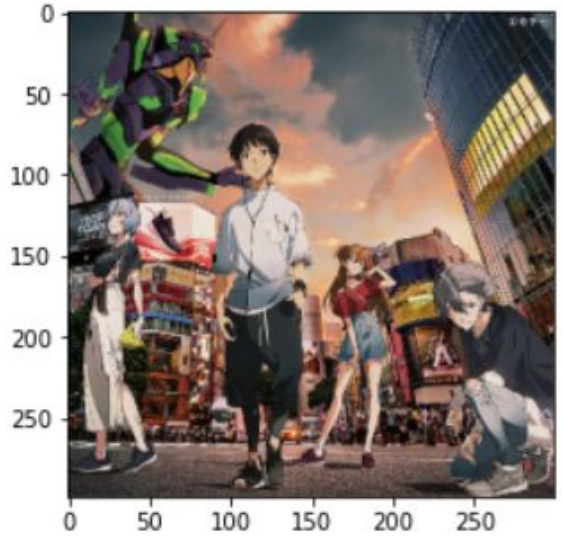
入力 [20]:
```
from tensorflow.keras.applications.inception_v3 import preprocess_input
from tensorflow.keras.applications.inception_v3 import decode_predictions
```

入力 [23]:
```
import matplotlib.pyplot as plt
def read_img(img_path, resize=(299,299)):
    img_string = tf.io.read_file(img_path)
    img_decode = tf.image.decode_image(img_string)
    img_decode = tf.image.resize(img_decode, resize)
    img_decode = tf.expand_dims(img_decode, axis=0)
    return img_decode

img_path = 'Untitled Folder/8447.jpg'
img = read_img(img_path)
plt.imshow(tf.cast(img, tf.uint8)[0])
```

出力[23]:
```
<matplotlib.image.AxesImage at 0x1c1a67ab940>
```

出力[23]: <matplotlib.image.AxesImage at 0x1c1a67ab940>



入力 [24]:
```
img = preprocess_input(img)
preds = model.predict(img)
print("Predicted:", decode_predictions(preds, top=3)[0])
```

```
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/imagenet_class_index.json
40960/35363 [==================================] - 0s 0us/step
Predicted: [('n06596364', 'comic_book', 0.47411594), ('n02817516', 'bearskin', 0.23792689), ('n07248320', 'book_jacket', 0.0273
64124)]
```

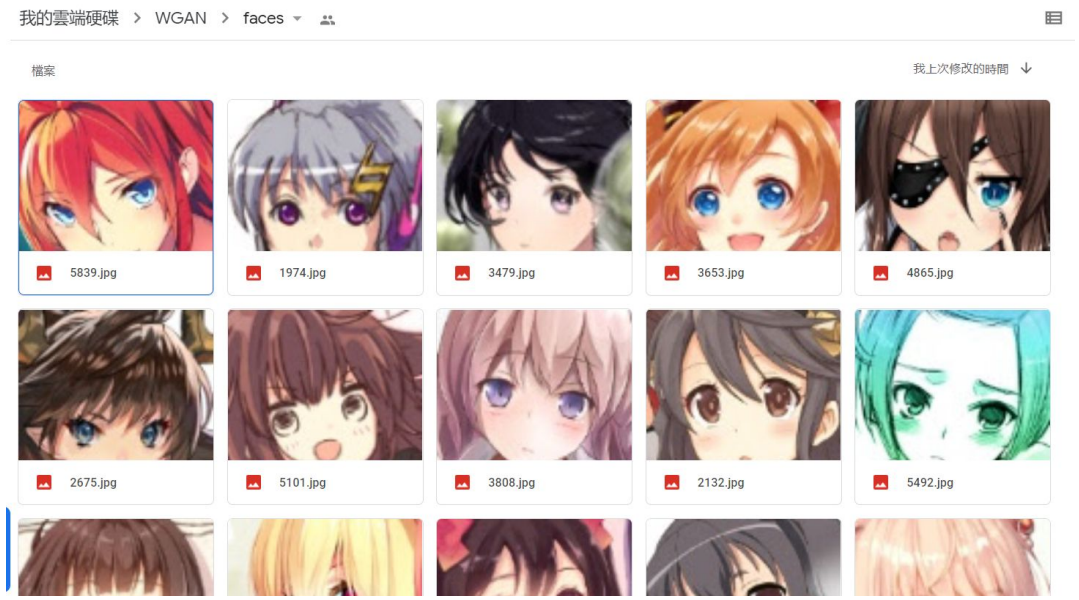
入力 [ ]:

結論: 適用範圍與研究目的不符

WGAN(未完全)

- 資料處理上的困難點 : 分類、圖像處理

- 所以我用網路資料集先嘗試



預期:


043


045


047

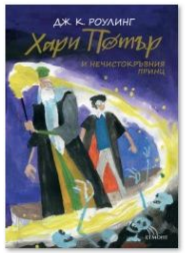

054

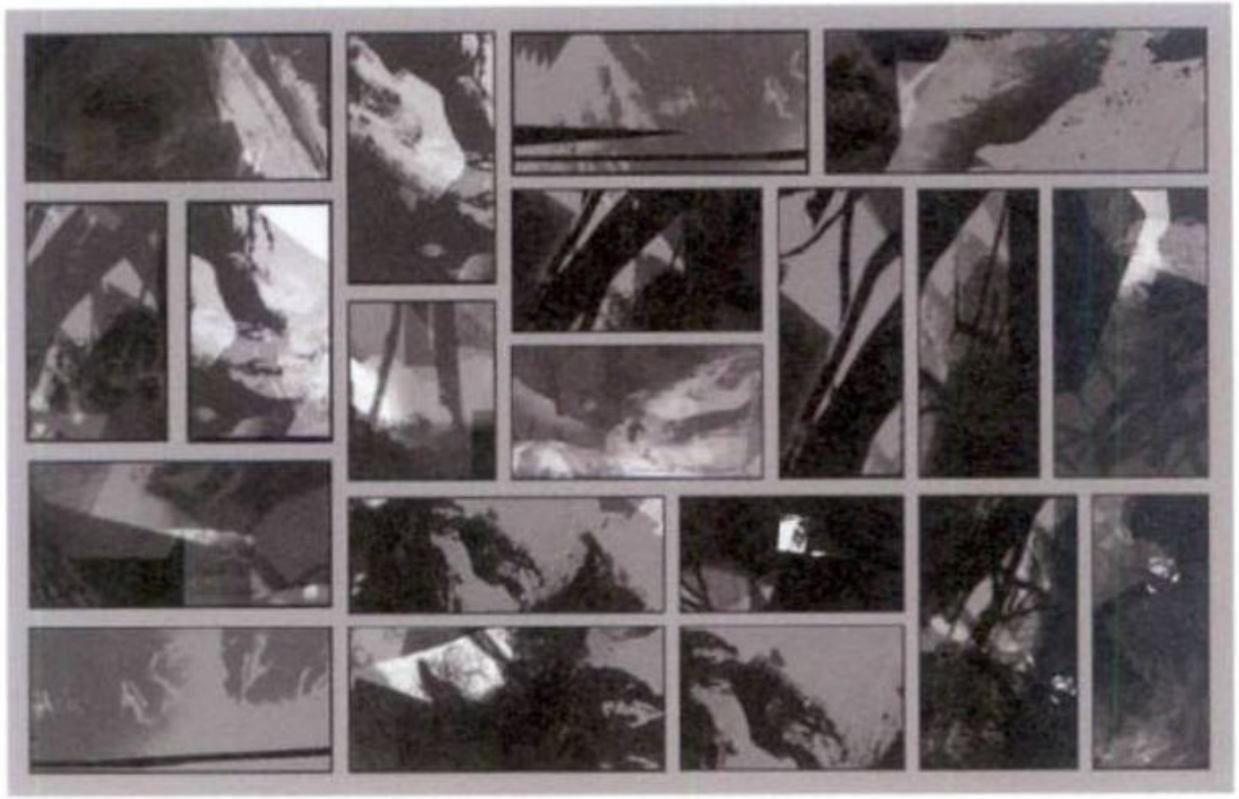

125


227


228


379

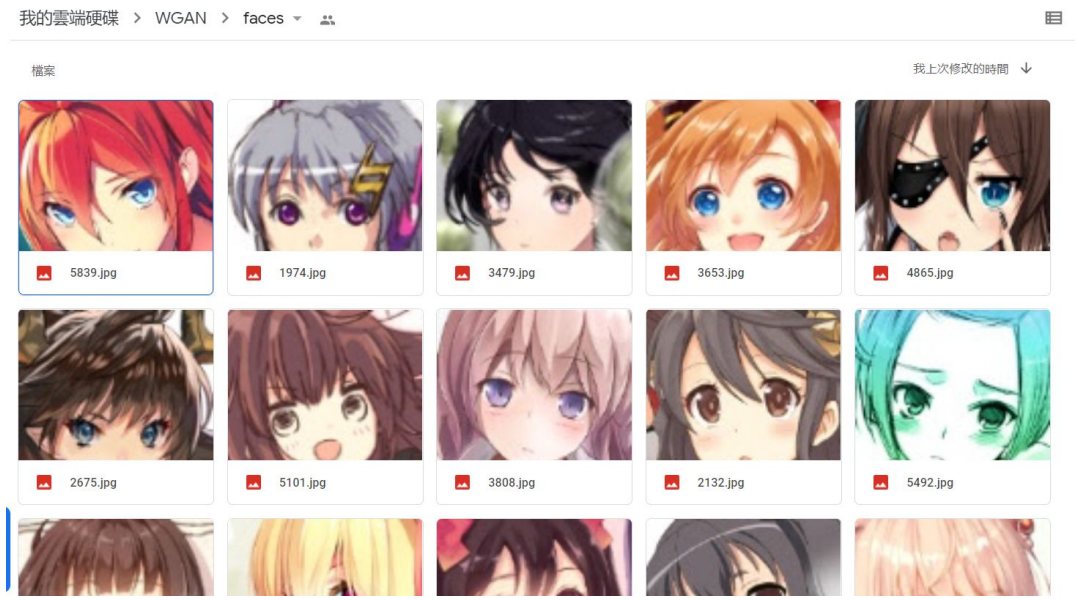




WGAN(未完全)

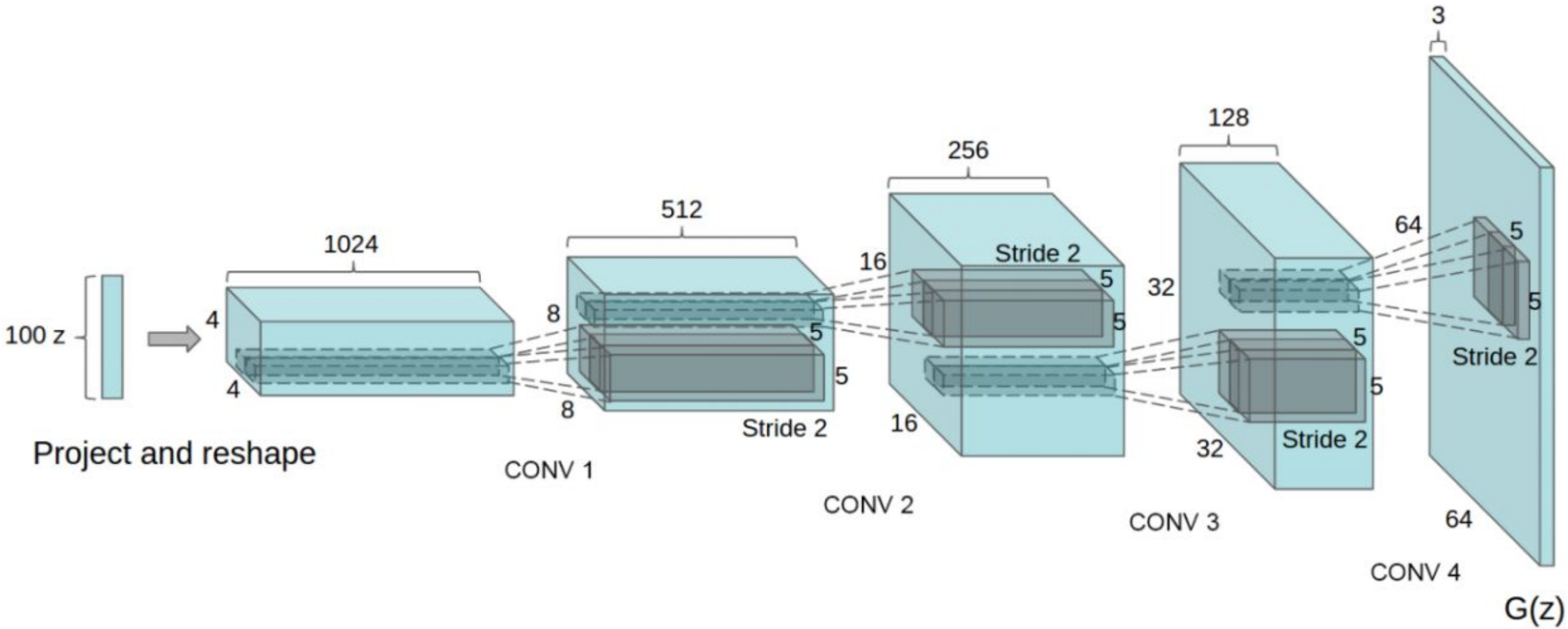
- 資料處理上的困難點 : 分類、圖像處理

- 所以我用網路資料集先嘗試

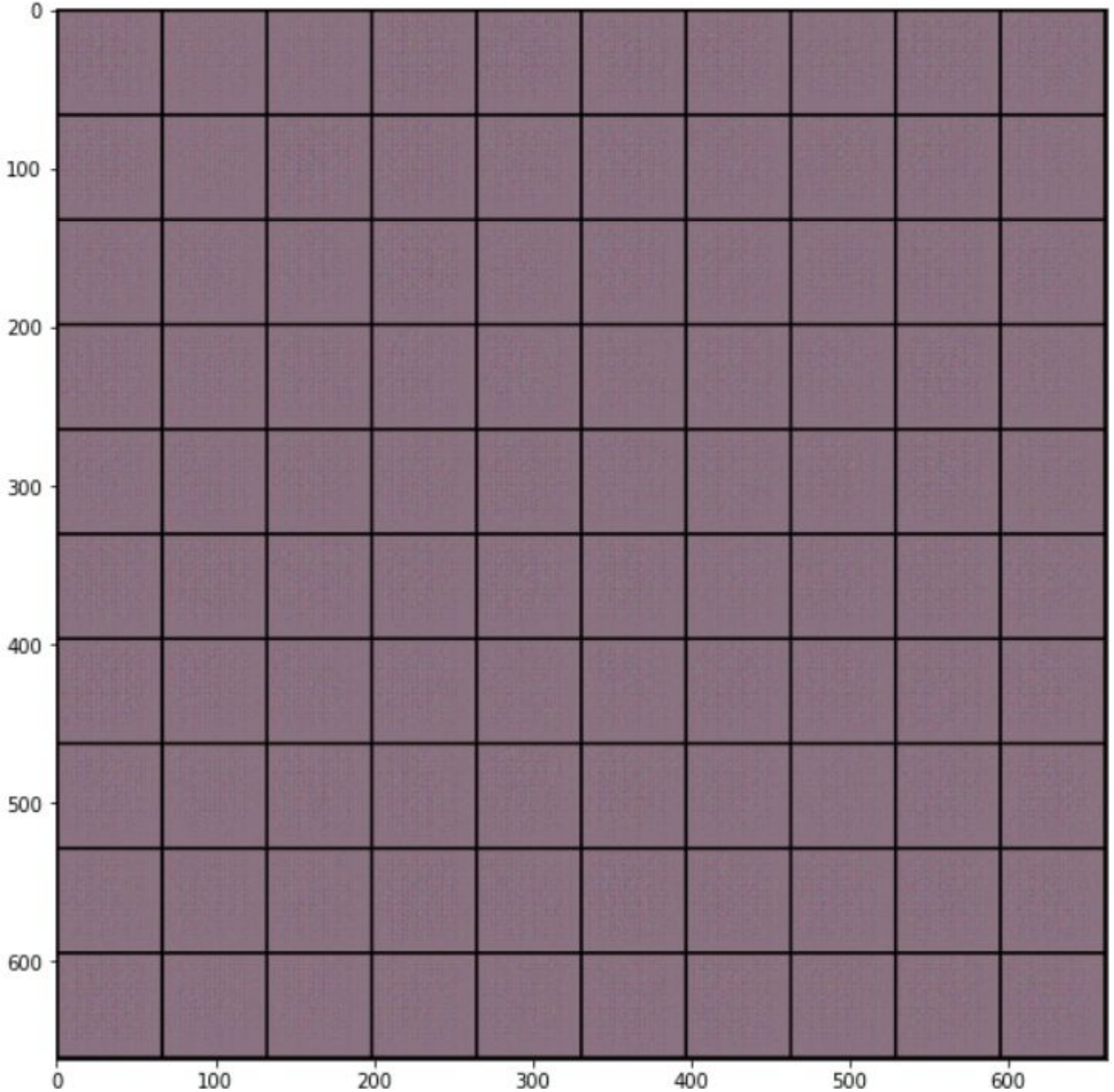


Model

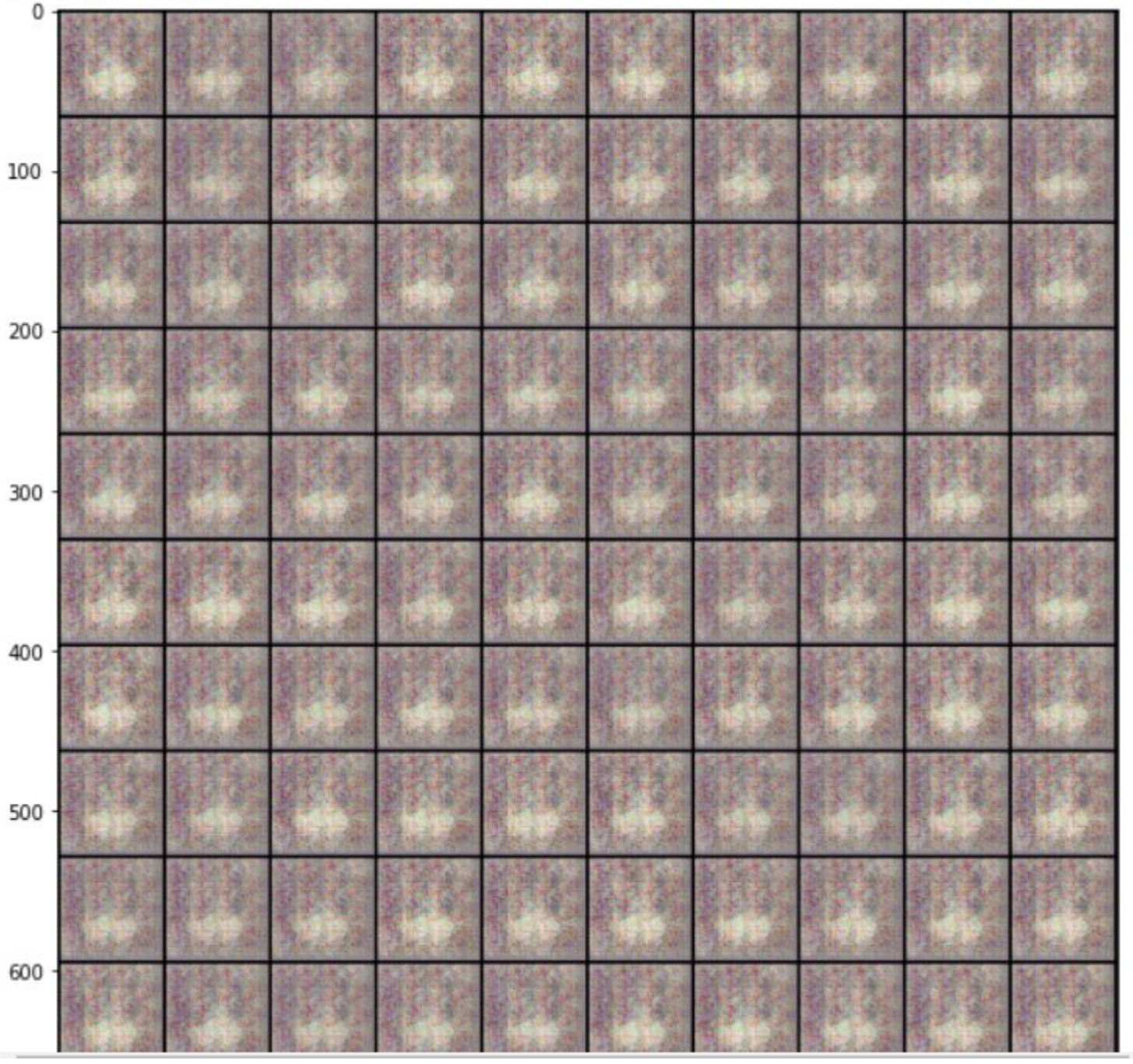
使用 DCGAN 作為 baseline mode。圖示為 DCGAN 架構示意圖。

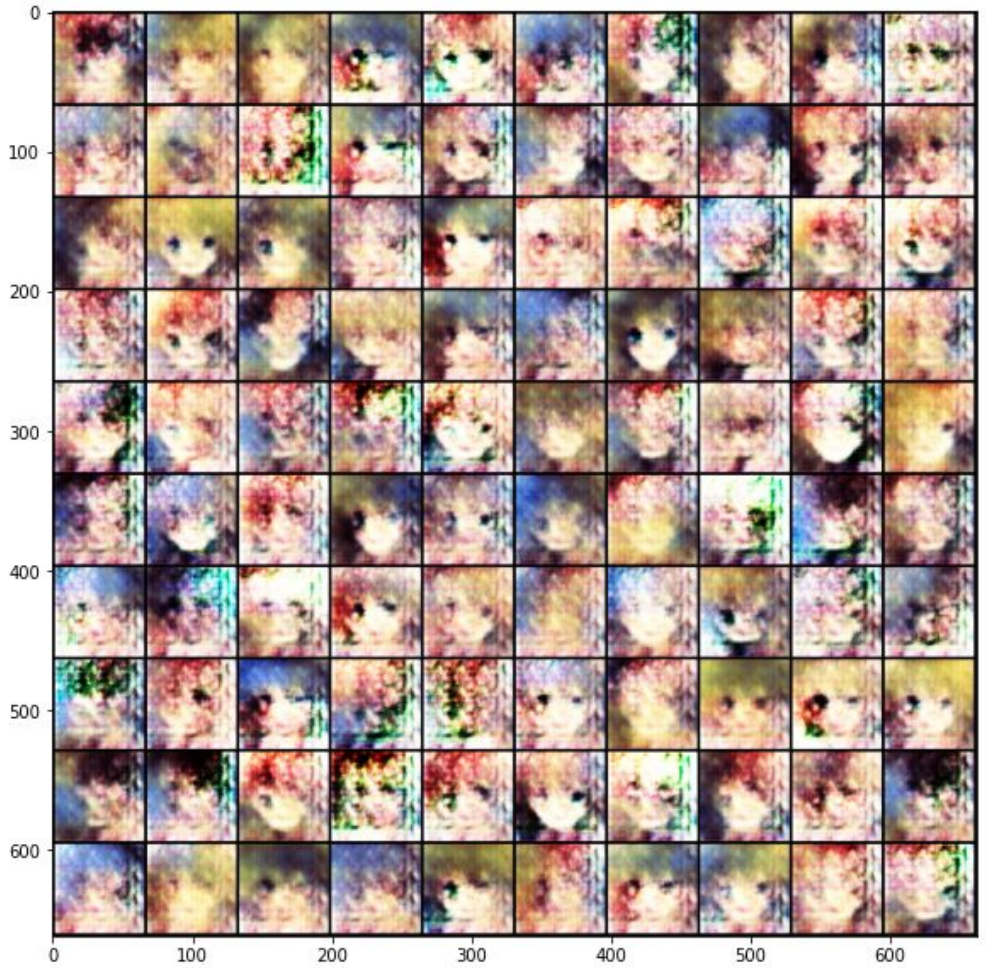


```
/usr/local/lib/python3.7/dist-packages/torch/utils/data/dataloader.py:477: UserWarning: This DataLoader will create 4 worker processes in tot
  cpuset_checked))
Epoch [1/20] 4/4 Loss_D: -0.0006 Loss_G: -0.5022 | Save some samples to /content/drive/My Drive/WGAN/logs/wgan_Epoch_001.jpg.
```



```
Epoch [19/20] 4/4 Loss_D: -0.0120 Loss_G: -0.4998 | Save some samples to /content/drive/My Drive/WGAN/logs/wgan_Epoch_019.jpg.
```






```
# generate images and save the result
same_seeds(0)
n_output = 20
z_sample = Variable(torch.randn(n_output, z_dim)).cuda()
imgs_sample = (G(z_sample).data + 1) / 2.0
save_dir = os.path.join(workspace_dir, 'logs')
filename = os.path.join(save_dir, f'wgan_result.jpg')
torchvision.utils.save_image(imgs_sample, filename, nrow=10)
# show image
grid_img = torchvision.utils.make_grid(imgs_sample.cpu(), nrow=10)
plt.figure(figsize=(10,10))
plt.imshow(grid_img.permute(1, 2, 0))
plt.show()
```

YOLOV3(未完成)

- 困難點:訓練

- 解方:權重檔、yolov3-tiny