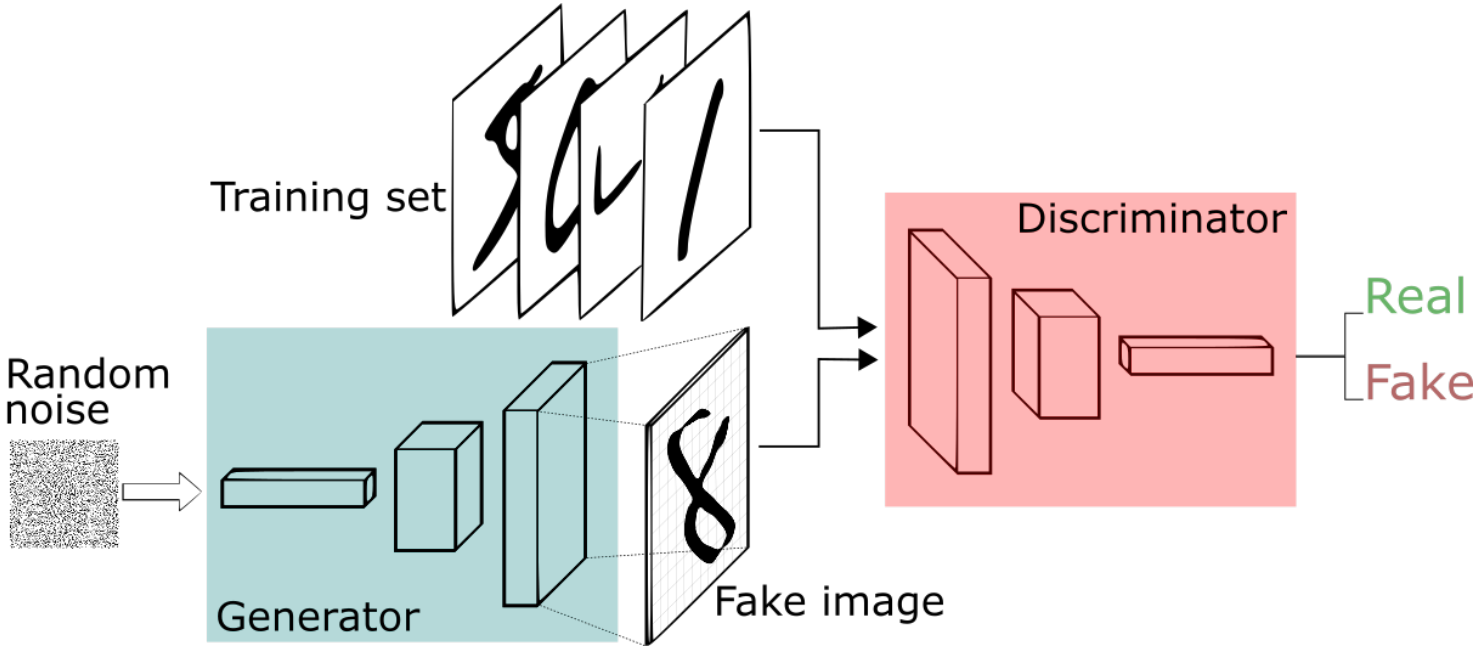
Introduction to Natural Language Processing

GAN - Generative Adversarial Network

RB308-3

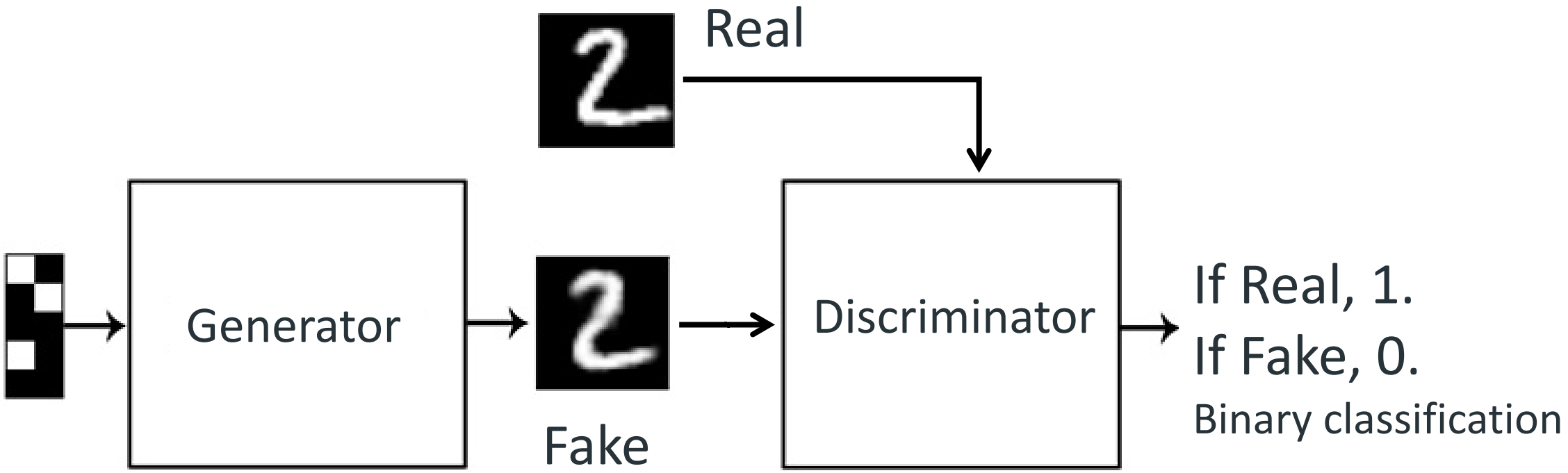
羅上堡

1. GAN – Generator & Discriminator
2. GAN – keras example



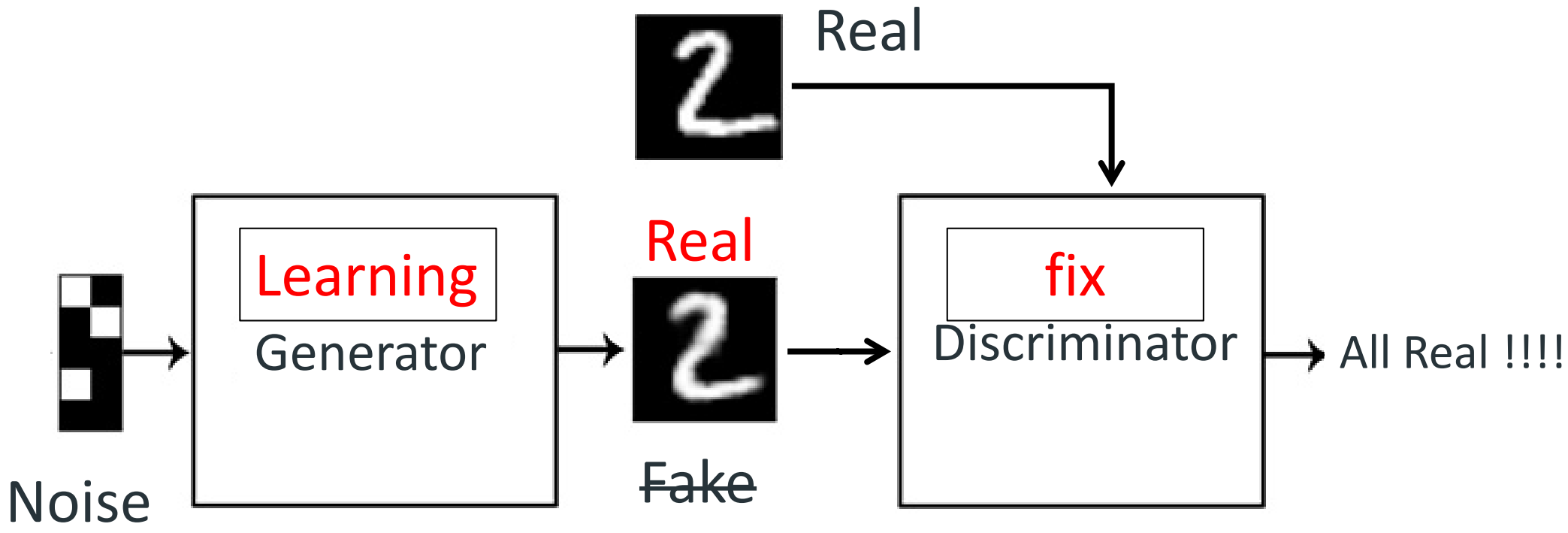
GAN - 生成對抗網路

- GAN由兩個網路所組成➔ Generator 、Discriminator。



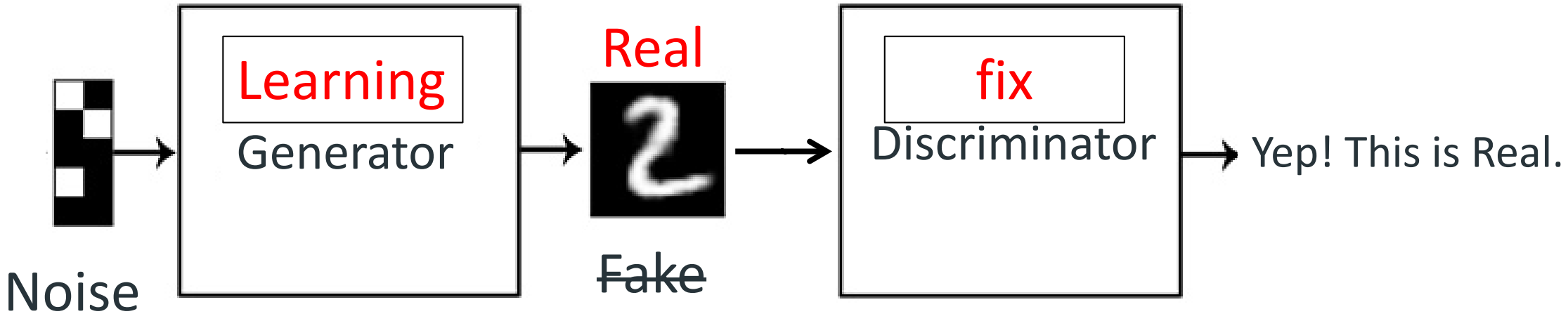
GAN – Generator 生成器

- Generator 生成器
  - 目標：讓Discriminator判別不出，產生出來的圖判定為Fake。



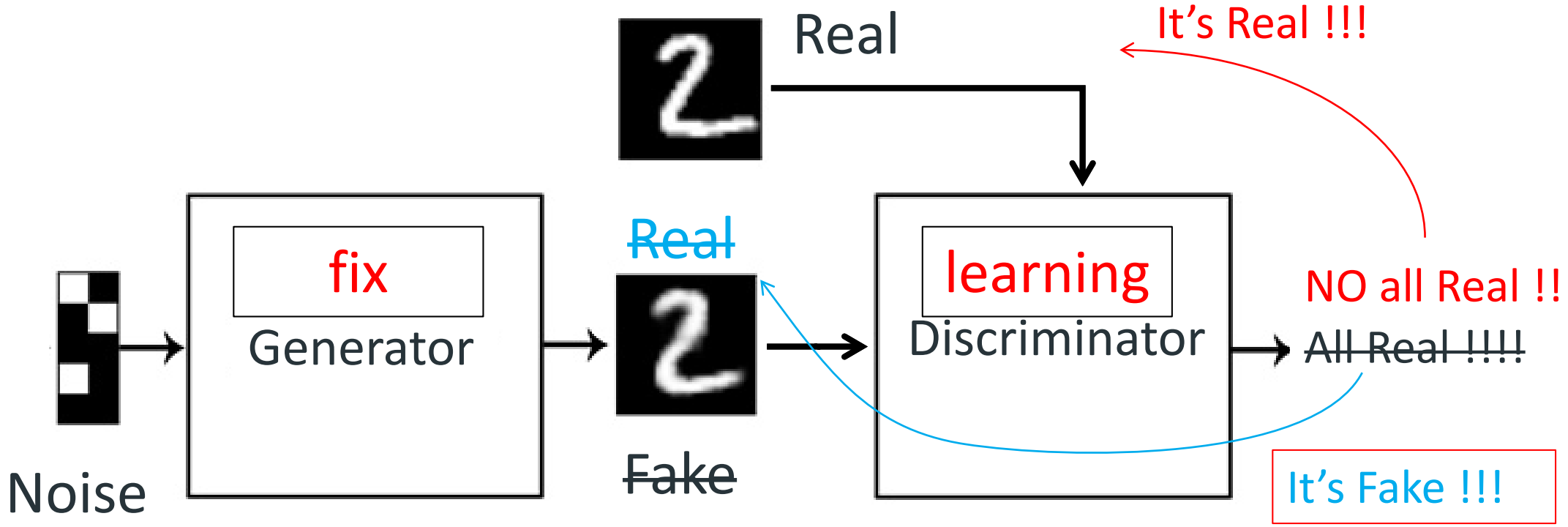
GAN – Generator 生成器

- 訓練 Generator 生成器
  - 與判別器一起訓練，但是不更新判別器，只更新生成器。
  - 輸入：低維度的高斯雜訊。 (※通常)
  - 輸出：影像等等...。(※通常與真實data的維度一樣。)



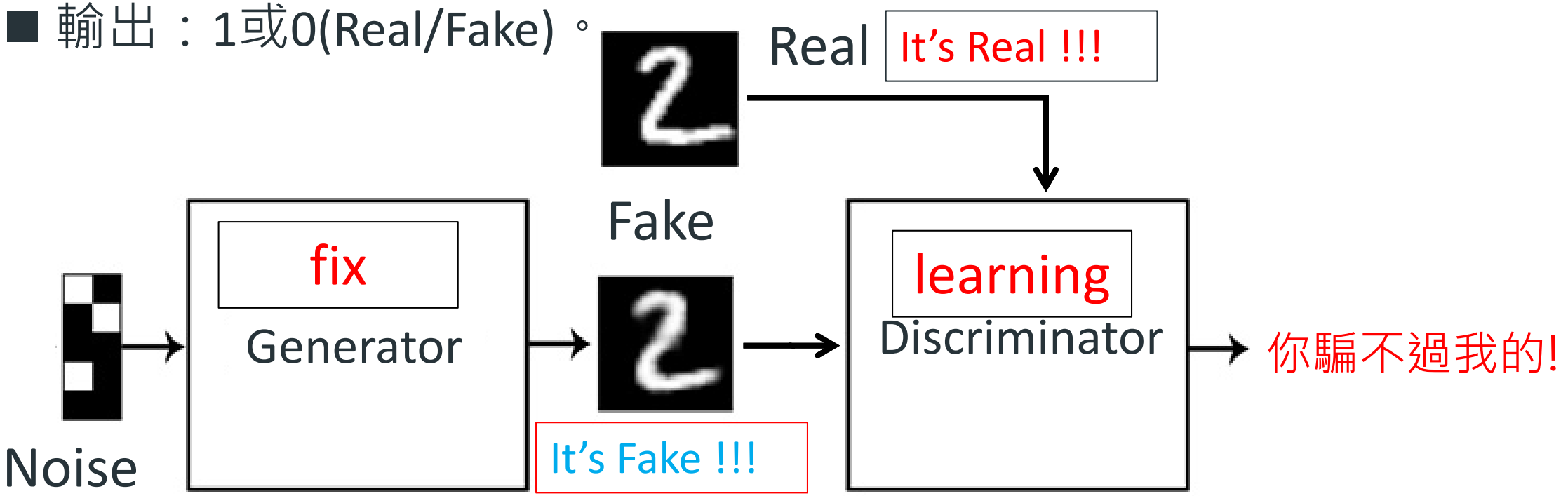
GAN - Discriminator 判別器

- Discriminator 判別器
  - 目標：能夠正確判斷，生成的圖是假的，真實的圖是真的。



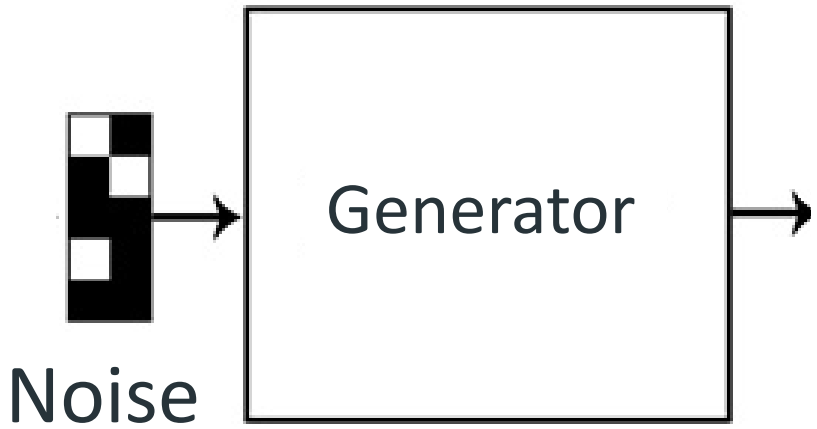
GAN - Discriminator 判別器

- 訓練 Discriminator 判別器
  - 不考慮Generator，只更新判別器。
  - 輸入：影像等等…。
  - 輸出：1或0(Real/Fake)。



GAN - 生成對抗網路

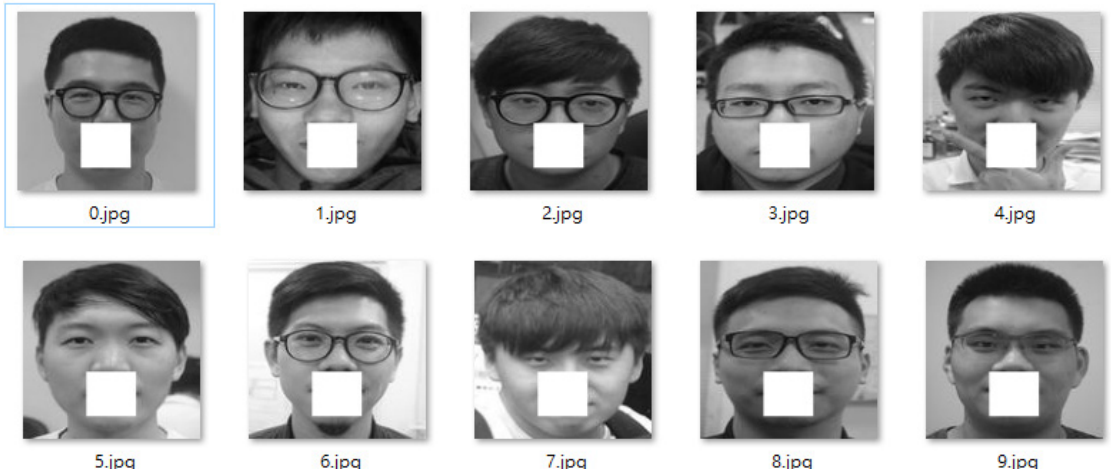
- GAN就是藉由兩個網路『一直互相對抗』，來進行訓練。
- 通常最終目的：Generator網路。
  ※因為Generator到最後會非常逼近真實圖像。



隨著對抗，會產生愈好的圖。

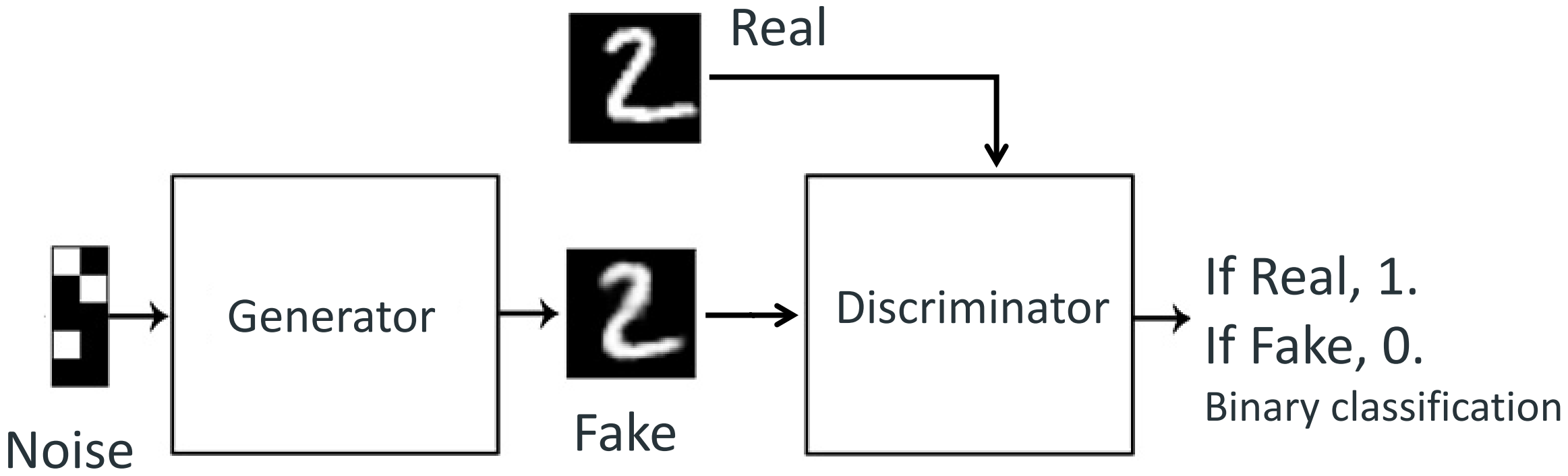
GAN – Homework -【圖像還原 –『嘴巴生成』】

- 實作一個Conditional GAN ➔ 將抹白的部分，生成嘴巴。
- Dataset : NLP Student Collection .
- Generator
  - 輸入：抹白圖。
  - 架構：建議 U-NET。
  - 輸出：真實圖。
- Discriminator
  - 輸入：Real Pair & Fake Pair。(輸入兩張圖)
  - 架構：Patch-GAN　　/　隨意。
  - 輸出：n*n的Label map /　1-dimention Label map 。



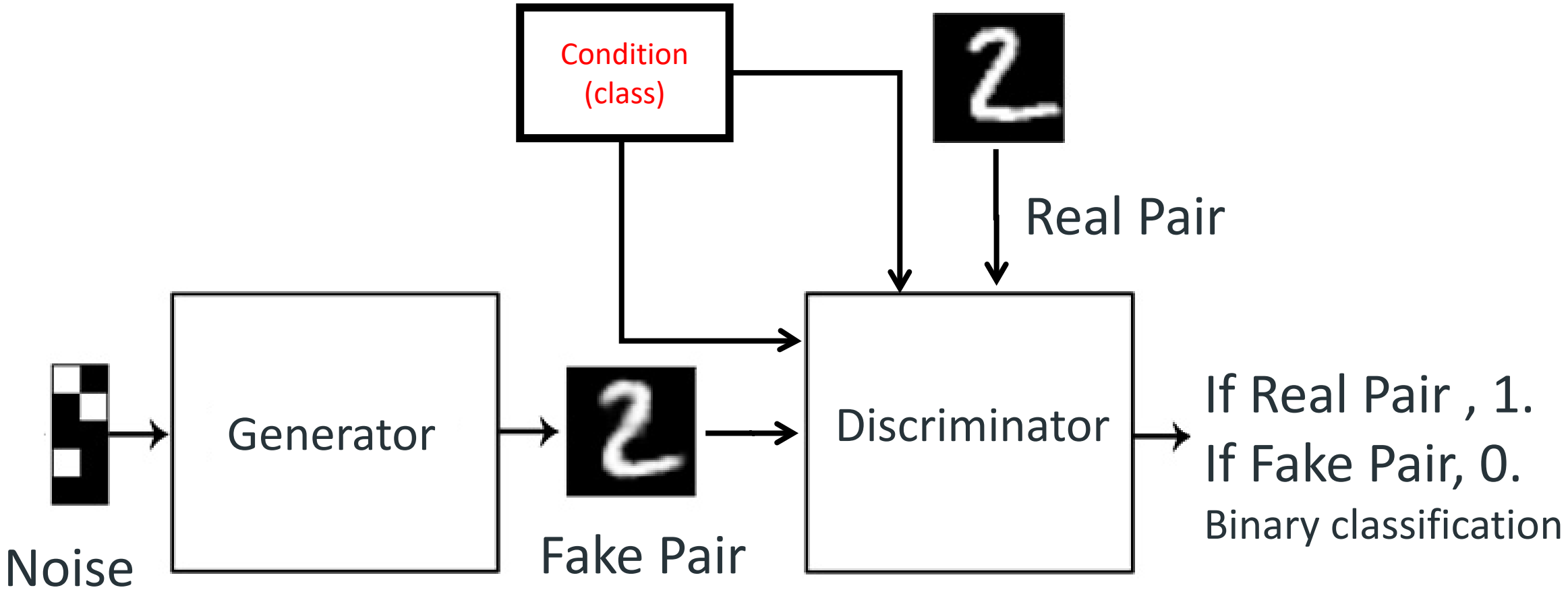
GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- GAN



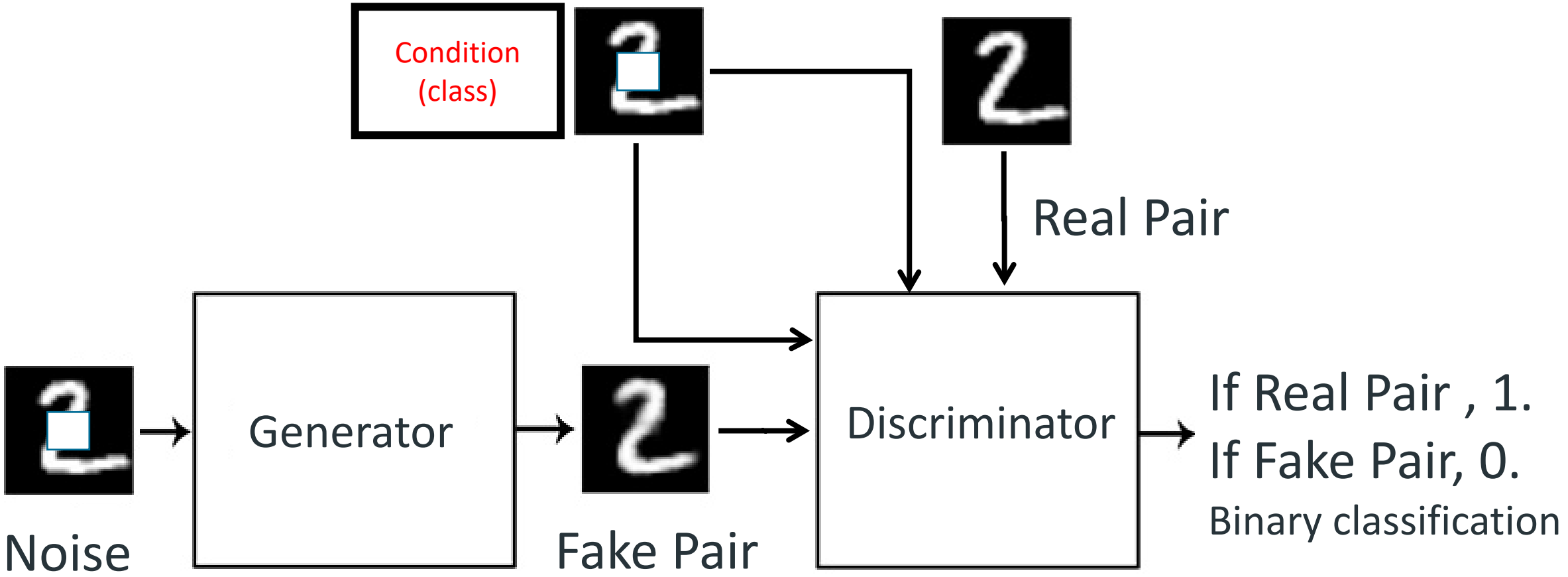
GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Conditional - GAN



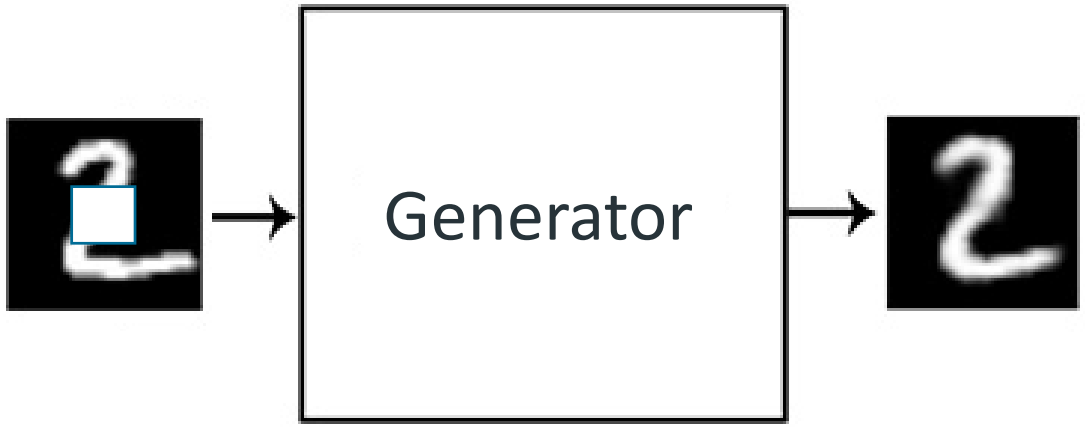
GAN – Homework - 【圖像還原 – 『嘴巴生成』】

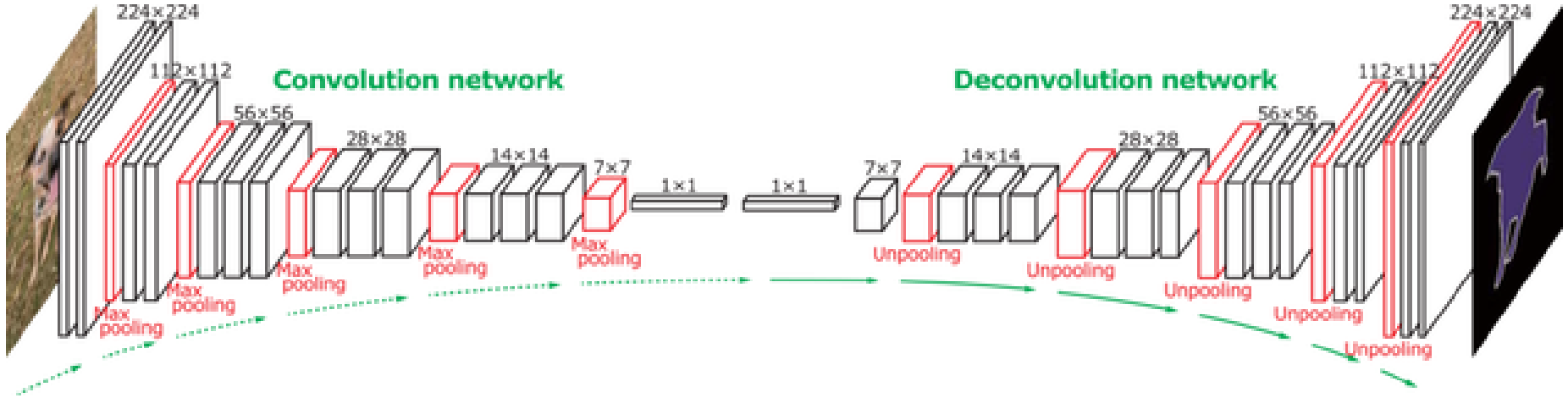
- Pixel2Pixel - GAN由兩個網路所組成 ➔ U-Net(FCN) 、Patch-GAN。



GAN – Homework - 【圖像還原 – 『嘴巴生成』】

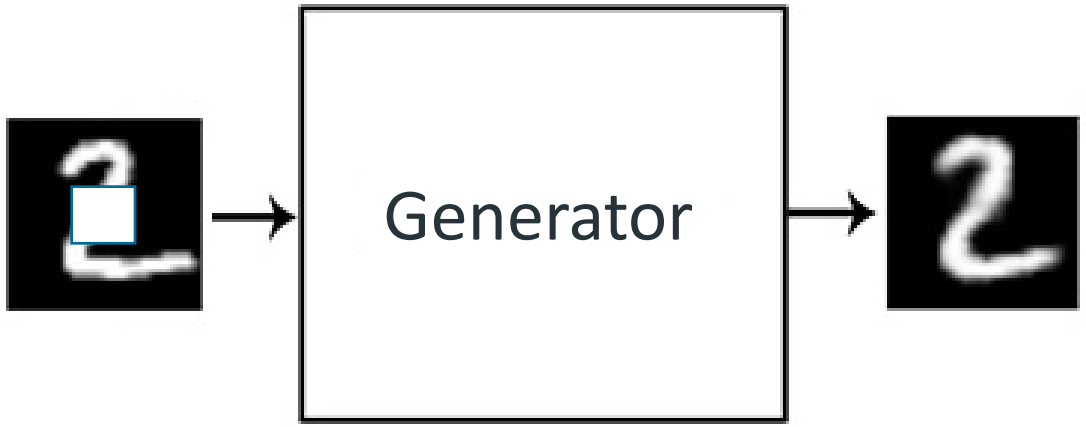
- Generator 設定建議
  - Fully Convolutional Neural Network.
  - 輸入與輸出同大小。
  - 由Encoder & Decoder所組成。

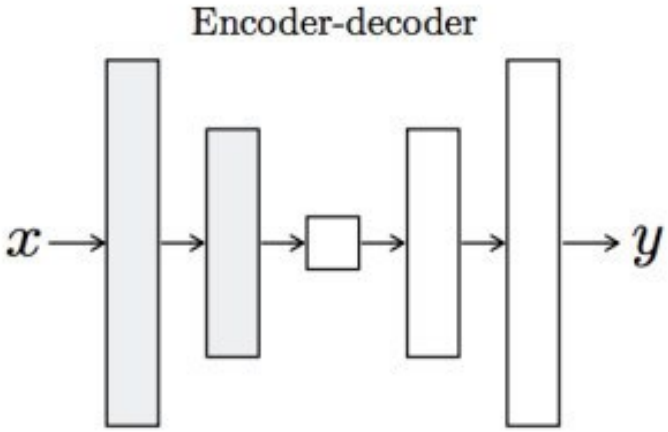


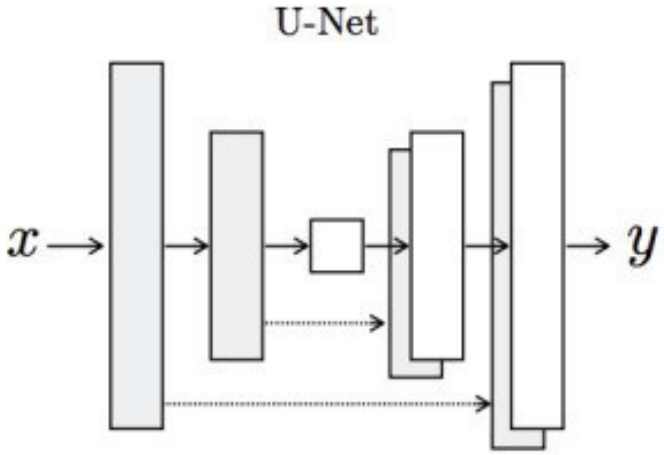


GAN – Homework - 【圖像還原 – 『嘴巴生成』】

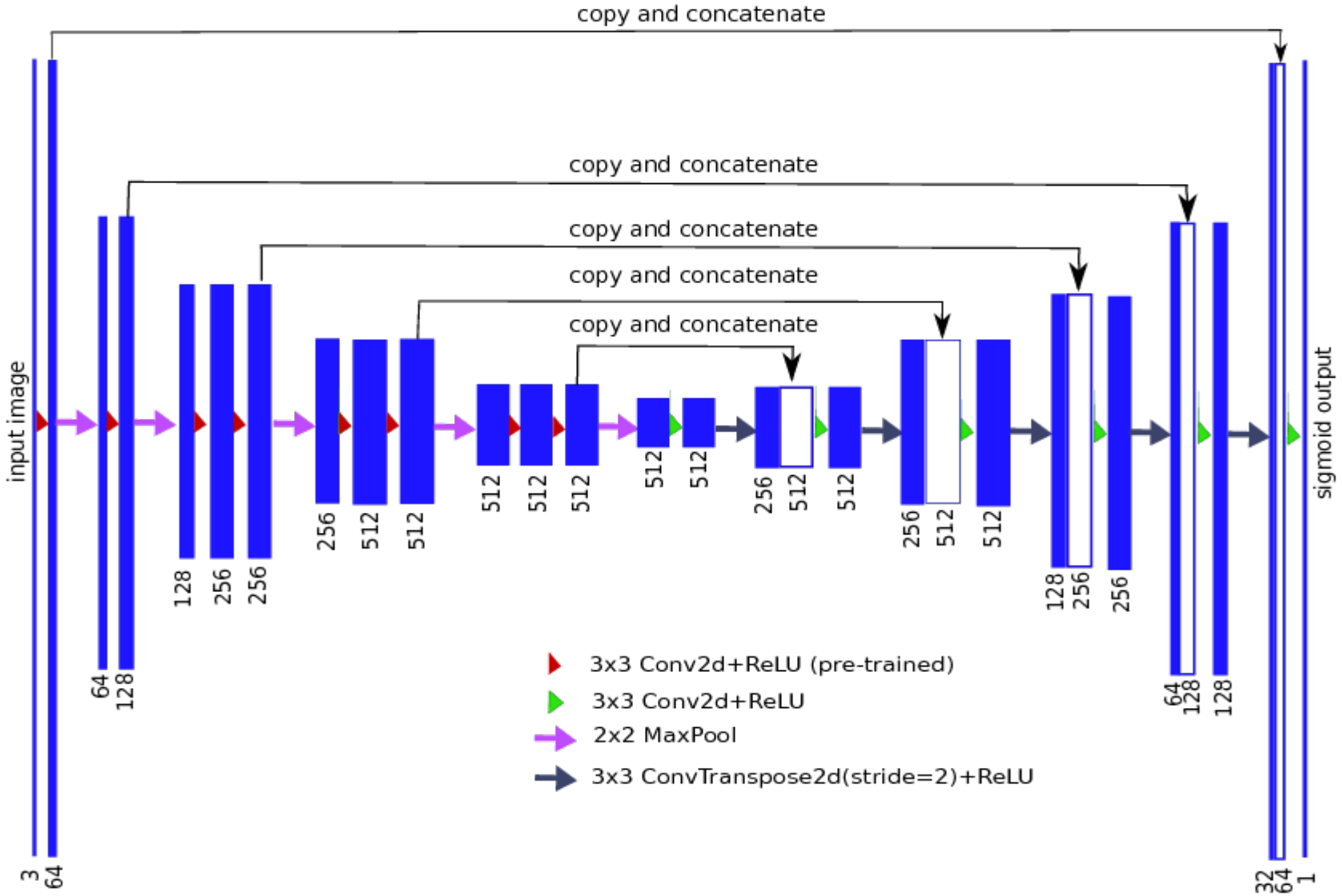
- Generator 設定建議
  - U-Net.
  - 輸入與輸出同大小。
  - 由Encoder & Decoder所組成。







GAN – Homework - 【圖像還原 – 『嘴巴生成』】



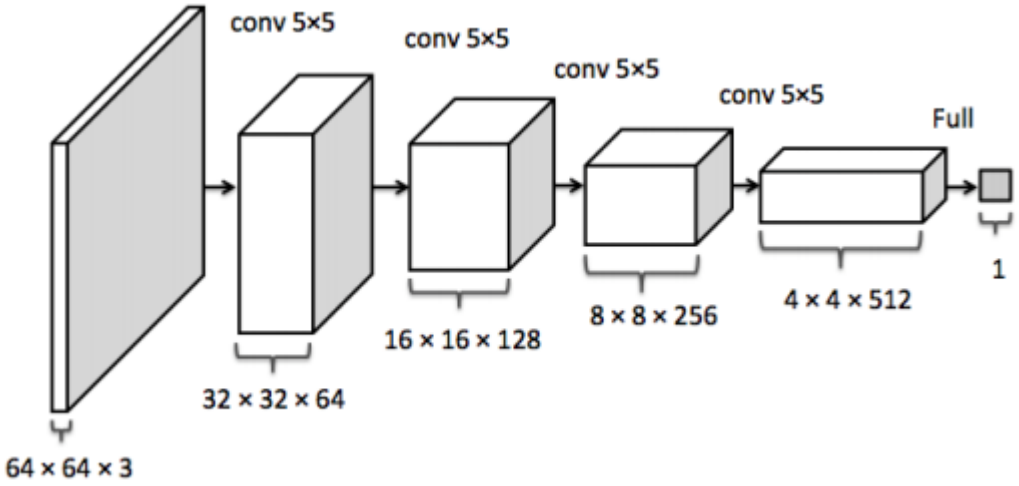
GAN – Homework -【圖像還原 – 『嘴巴生成』】

- Discriminator 設定建議
  - Patch-GAN or Dense or Conv2D + Dense。
  - 通常是1維的輸出。
  - 如果是Patch-GAN➔是n*n的維度輸出。
  - 激活函數通常用Sigmoid。
  - 可以一次訓練多次。

GAN – Homework -【圖像還原 –『嘴巴生成』】

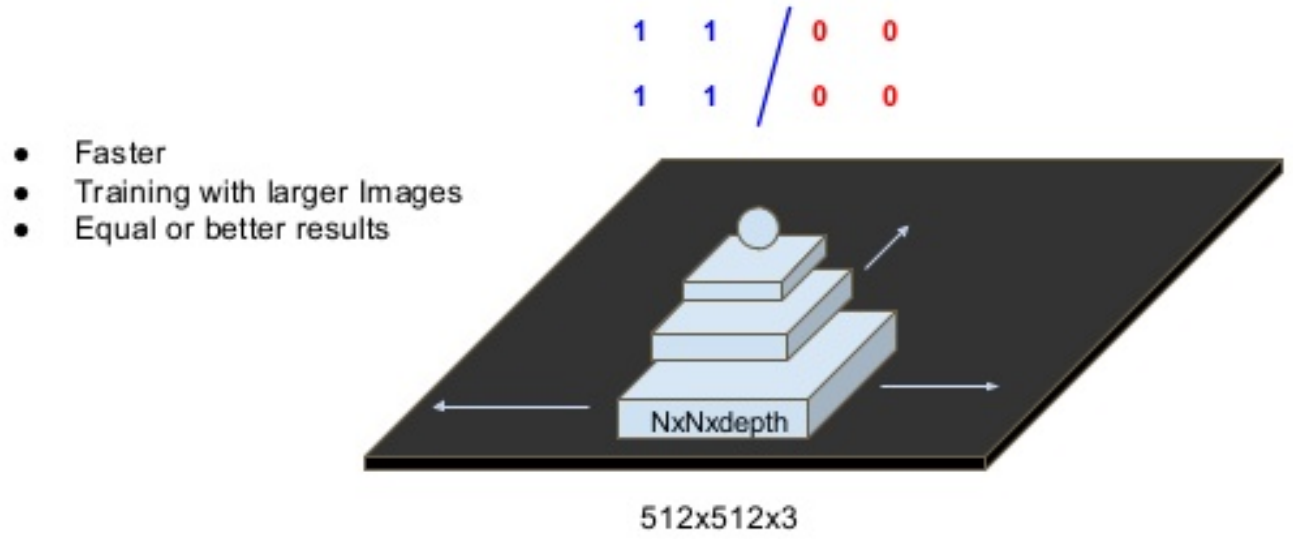
- Discriminator 設定建議
  - Dense or Conv2D + Dense。
  - 通常是1維的輸出。
  - 激活函數通常用Sigmoid。
  - 可以一次訓練多次。

※ 右圖為Conv2D + Dense示意圖。
輸入是兩張Concat成一張，
而非輸入單張。



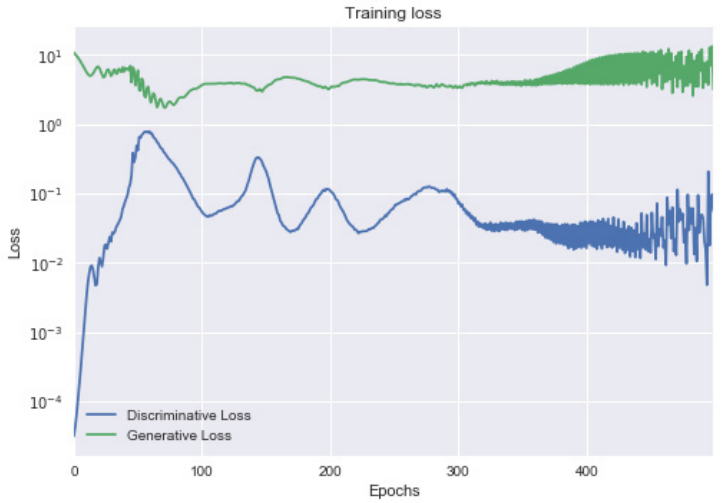
GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Discriminator 設定建議
  - Patch-GAN
  - 就是一連串的Conv2D做提取，並且輸出是n*n的輸出。
  - 可視為將一份影像，Patch成很多塊，分別做一次Discriminator。
  - 所以假設有n*n個Patch ➔ 就有n*n個輸出。



GAN – Homework -【圖像還原 –『嘴巴生成』】

- 網路統一設定建議
  - 通常用Binary Cross Entropy 來計算loss。
  - 在Pix2Pix是用MSE或MAE來計算loss。
  - 盡量使用Batch Normalization、Dropout。
  - Leaky ReLU通常會比ReLU還好。
- 網路訓練的提醒
  - 兩網路的Loss會一直跳動(對抗)。
  - 精確度➔ Discriminator會在0.5跳動。
  - **訓練Discriminator時， Real Pair data label = 1 , Fake Pair data label = 0.**
  - **訓練Generator時， Fake Pair data label = 1.**

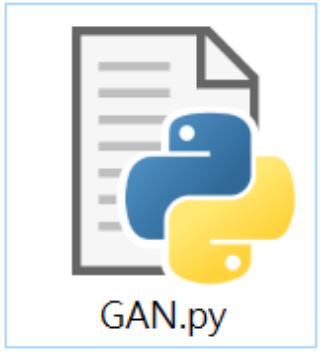


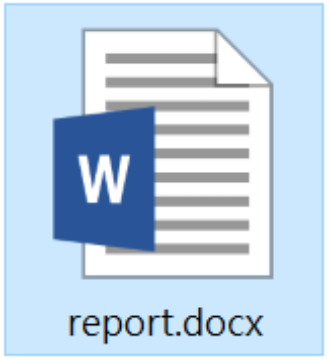
GAN – Homework -【圖像還原 –『嘴巴生成』】

- 整個作業流程大致分為以下部分：
  - 創建Generator與Discriminator網路。
  - 創建訓練Generator的網路。　(訓練Generator時，需要包含Discriminator。)
  - 讀取Data資料。
  - 定義Training Label的答案。　(Real Pair Label = 1 , Fake Pair Label = 0)
  - Step1. 隨機挑選N個Real & White Image。　(N=batch size)
  - Step2. 將White Image給Generator產生Fake Image。
  - Step3. 拿(Fake Pair & Real Pair)訓練Discriminator網路。　(可訓練多次)
  - Step4. 拿(Fake Pair)訓練Generator網路。
  - Step5. 重複Step.1至Step4. 的動作，直到訓練收斂or其他訓練條件滿足。

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Score : 15%
  - 程式碼。0% (沒給扣5分)
  - 文書檔。5%
  - Kaggle。 10%





kaggle

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - Link： https://ppt.cc/fKywMx
  - Deadline：2018/05/29 23:59 .
  - Upload limit 20 times/day .

Team Name	Score
Strong_Baseline	0.03568
Baseline	0.14744

- Final Score：
  - <1> ≦ baseline ➔ 0 score
  - <2> ≧ baseline ➔ $3 + 5\frac{baseline - your}{baseline - strong_{baseline}}$
  - <3> ≧ strong baseline ➔ $8 + 2\frac{strong_{baseline} - your}{strong_{baseline} - highest}$

GAN – Homework -【圖像還原 –『嘴巴生成』】

- Kaggle 10%
  - Data：NLP Student Collection。
  - Training Set： 700組 真實-抹白 照片 。
  - Test Set：10張 抹白照片。
  - Metric：RMSE。
  - Baseline Setting ➔ 直接丟抹白圖與答案圖做RMSE。
  - Strong Baseline ➔ 使用pix2pix GAN超簡化版。

    ※ pix2pix原本參數量：44,584,214 ➔ 77,766。

GAN – Homework -【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - NN Structure：

    <1> Generator → Conv2D(f=4,k=(4,4),s=(2,2))+LeakyRelu(0.2)
    + Conv2D(f=8,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + Conv2D(f=16,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + Conv2D(f=32,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + Conv2D(f=32,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + UpSampling(s=(2,2))+Conv2D(f=32,k=(4,4),s=(1,1))+Relu(0.2)+BN+skip_input
    + UpSampling(s=(2,2))+Conv2D(f=16,k=(4,4),s=(1,1))+Relu(0.2)+BN+skip_input
    + UpSampling(s=(2,2))+Conv2D(f=8,k=(4,4),s=(1,1))+Relu(0.2)+BN+skip_input
    + UpSampling(s=(2,2))+Conv2D(f=4,k=(4,4),s=(1,1))+Relu(0.2)+BN+skip_input
    + UpSampling(s=(2,2))+Conv2D(f=1,k=(4,4),s=(1,1))+tanh

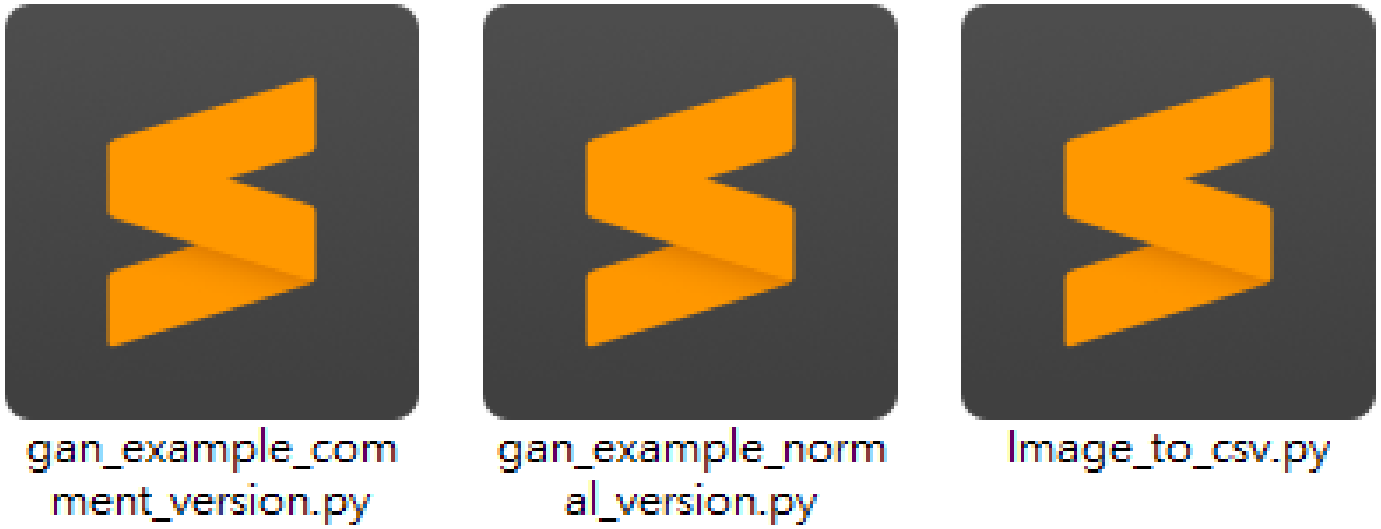
GAN – Homework -【圖像還原 –『嘴巴生成』】

- Kaggle 10%
  - NN Structure：

    <2> Discriminator ➔ Conv2D(f=4,k=(4,4),s=(2,2))+LeakyRelu(0.2)
    + Conv2D(f=8,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + Conv2D(f=16,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + Conv2D(f=32,k=(4,4),s=(2,2))+LeakyRelu(0.2)+BN
    + Conv2D(f=1,k=(4,4),s=(1,1))
  - Optimizer：Adam(lr=0.0002,beta_1=0.5) G & D same.
  - loss ：G ➔『MSE,MAE』，D ➔『MSE』。
  - Epoch：3000

GAN – Homework -【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - 提供三檔案： <1> gan_example_comment_version.py #有註解
    <2> gan_example_normal_version.py #純Code版
    <3> image_to_csv.py #繳交至Kaggle



GAN – Homework -【圖像還原 –『嘴巴生成』】

- Kaggle 10%
  - 包含：1.『Training Data Loading 』
    2.『Define D and G network structure and parameter setting』
    3.『Define Training D and G model』
    4.『Define Image Generator』
    5.『Training Phase』
    6.『Display your predict』


gan_example_comment_version.py


gan_example_normal_version.py

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - 包含：1.『Training Data Loading 』

```
##############################################################
# Training Data List Creat
##############################################################
train_real_data_dir = r'.\NLP_data\Training\Real\*'
train_white_data_dir = r'.\NLP_data\Training\White\*'

real_list = glob.glob(train_real_data_dir)
train_real_data_list = []
train_real_data_list.extend(real_list)

white_list = glob.glob(train_white_data_dir)
train_white_data_list = []
train_white_data_list.extend(white_list)
```

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - 包含：2.『Define D and G network structure and parameter setting』

```
#########################################################################
# Define D and G and parameter
#########################################################################
img_row = img_col = 128 #先規定成128*128 不過由於Pixel2Pixel架構的關係 最好是2的次方數 (2^n)
channels = 1    # 1 is gray
                # 3 is RGB
img_shape=(channels,img_row,img_col) # or (img_row,img_col,channels)
#2018.05.03補充==>這邊Img_shape在Keras中 預設是『img_row,img_col,channels』==>Channels是放在後面的
#                 所以這邊看你們怎麼設定img_shape
#                 如果你用我這種寫法『channels,img_row,img_col』你後面在Call Conv2D等等的東西 需要加入『data_format='channe
#                 不過我記得有可以在剛開始一次設定是channels_first還是channels_last, 就請各位去查查囉。
#
def dis(input_shape):...

def gen(input_shape,** kwargs):...
```

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - 包含：3.『Define Training D and G model』

```
#########################################################################
# 定義訓練D的模型
#########################################################################
#定義你的優化器巴 SGD Adam Adamax Adadelta rmsprop etc... 都可以 隨便
#只是學習率通常別調太高 ...(ps:容易壞掉QQ)   我自己是用0.0002 Adam
D_optimizer = <your creativity>
#如果你是用Patch GAN conv2D做結尾的 建議用MSE
#如果你是用Dense直接變成1維 則沒限制
D.compile(loss=<your creativity>, optimizer=D_optimizer,metrics=['accuracy'])
#Loss 跟投影片一樣 MSE 或 Cross Entropy 隨便 都可以試
D.summary()
```

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

- Kaggle 10%
  - 包含：3.『Define Training D and G model』

```
#############################################################
# 定義訓練G的模型
#############################################################
#沒錯上課說過
#再訓練G的時候 需要把D引入近來 並且Fix它 寫法就在下面
#當然我們還是先定義優化器 ==>記住 通常是跟D的優化器設定一樣 不過沒人說這樣一定好 隨便你的選擇
#這邊我們將這種網路叫做AM（對抗模型）
AM_optimizer = <your creativity>
img_A = Input(input_shape)              #真實圖
img_B = Input(input_shape)              #助教塗白的真實圖
fake_A = G(img_B)                       #先將塗白的塗去預測出fake image ==>期望生成真實圖
D.trainable=False                       #讓D在AM模型Fix
valid = D([fake_A,img_B])               #將fake img 跟抹白的圖給D去判別 得出判決結果valid
AM = Model([img_A,img_B],[valid,fake_A]) # 定義model 我們這邊的輸出 有包含D的結果以及
                                        # G收到抹白的圖所產生的fake圖
AM.compile(loss=[<your creativity>,<your creativity>],loss_weights=[1,1],optimizer=optimizer)
                                        # 這邊的Loss通常是MSE或MAE,其他也可以
                                        # 第一個Loss 是 Dis的輸出Loss
                                        # 第二個Loss 你就想像是AutoEncoder的訓練方式 輸入抹白 輸
                                        # loss_weights是為了加快訓練效果 如果不想調整 都為1即可

AM.summary()
```

GAN – Homework -【圖像還原 –『嘴巴生成』】

- Kaggle 10%
  - 包含： <1> 讀取每個Test Image
    <2> 將每個Test Image 丟給生成器預測。
    <3> 將每張預測圖串接並flatten成kaggle可接受的格式。

※ 每張預測圖 Size ➔(M,N)
※ 串接預測圖 Size ➔(10,M,N)
※ 攤平預測圖 Size ➔(10,M*N)並轉置➔(M*N,10)
※ 利用Pandas存成csv檔案。


Image_to_csv.py

※ 如果你要自己寫 就需要以上四步驟來完成，不然結果會有問題。

GAN – Homework - 【圖像還原 – 『嘴巴生成』】

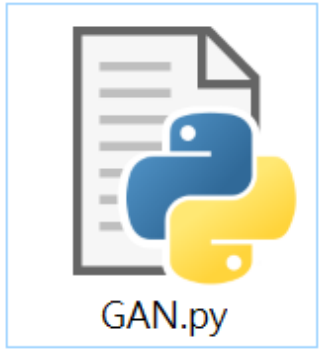
- Kaggle 10%

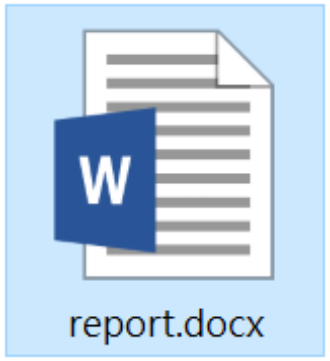
```
output_img_col = output_img_row=128
ori_img,white_img = generator_test_Img(list_dir=test_data_list,resize=(output_img_col,output_img_row))
#這邊的G就是Example Code的G拉
#只是怕放在最下面會搞亂大家 所以分別寫出來
image_array = G.predict(white_img).squeeze(1)      #Predict出來是 (n,1,img_row,img_col) 只是做Reshape為(n
image_array = (image_array+1)/2                    #這邊很重要 因為答案是0~1的灰階值
                                                   #然而如果你Generator輸出是tanh 會介於-1~1之間, 須把他變成
                                                   #當然如果你是Sigmoid輸出, 就不用作Rescale的動作


print(image_array.shape)                           #預期是(n,img_row,img_col) 如果不是 可能會轉換不出來, 或

numpy_to_csv(input_image=image_array,image_number=n,save_csv_name='Predict.csv')
```

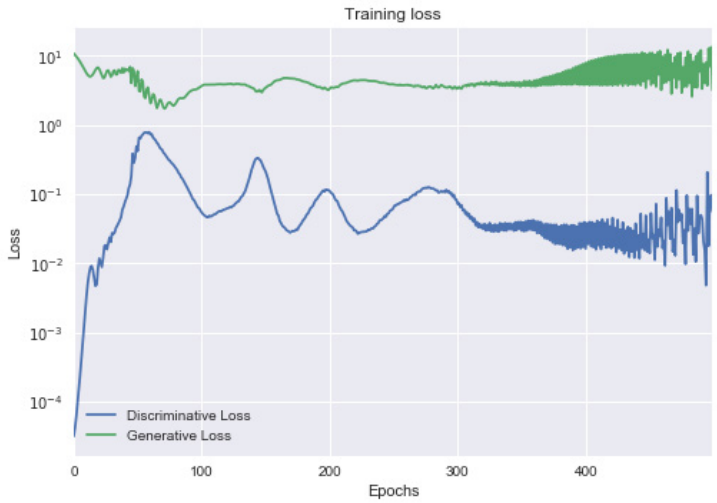
GAN – Homework -【圖像還原 –『嘴巴生成』】

- 上傳兩檔案至Moodle
  - 程式碼。

    ※ 沒限制實現語法 : keras pytorch caffe caffe2 chainer scikit-learn etc …
  - 文書檔：
    - 網路架構。 (每層的設定 等等…。)
    - 訓練Loss圖。 (包含：Generator & Discriminator。) (※ Tensorboard matlabplot etc ..)
    - 訓練方法與心得。







GAN – Homework -【圖像還原 –『嘴巴生成』】

- Score : 5%
  - 程式碼。 0% (沒給會扣5分)
  - 文書檔： 5%
    - 網路架構。 (2%)
    - 訓練Loss圖。 (1%)
    - 訓練方法與心得。(2%)

※ 隨機填值、填固定值、隨機填固定值 ➔整體分數0分。

※ 沒使用Generator去生成 ➔整體分數0分。

※ 純用AutoEncoder來預測➔整體分數打8折。

THE END

GAN – Generator 生成器 – keras example

- Generator Network
  - 輸入：100維的noise。
  - Dense➔升維➔四層Conv2D來創造圖像，最後一層用Sigmoid輸出。

```
G = Sequential()
G.add(Dense(7*7*256, input_dim=100,activation='relu'))
G.add(Reshape((256,7,7)))
G.add(UpSampling2D())
G.add(Conv2D(128, 5, padding='same',activation='relu'))
G.add(UpSampling2D())
G.add(Conv2D(64, 5, padding='same',activation='relu'))
G.add(Conv2D(32, 5, padding='same',activation='relu'))
G.add(Conv2D(1, 5, padding='same',activation='sigmoid'))
G.summary()
```

GAN – Generator 生成器 – keras example

- Generator Network
  - 聽說Conv2DTranspose有更好的效果。(給同學們自己試囉。)

```
G = Sequential()
G.add(Dense(7*7*256, input_dim=100))
G.add(Activation('relu'))
G.add(Reshape((256,7,7)))

G.add(UpSampling2D())
G.add(Conv2DTranspose(128, 5, padding='same'))
G.add(Activation('relu'))

G.add(UpSampling2D())
G.add(Conv2DTranspose(64, 5, padding='same'))
G.add(Activation('relu'))

G.add(Conv2DTranspose(32, 5, padding='same'))
G.add(Activation('relu'))

G.add(Conv2DTranspose(1, 5, padding='same'))
G.add(Activation('sigmoid'))
G.summary()
```

GAN - Discriminator 判別器 – keras example

- Discriminator Network
  - 輸入：(1,28,28)。
  - 疊4層Conv2D來完成的網路。

```
D = Sequential()
input_shape = (1,28,28)
D.add(Conv2D(64, 5, strides=2, input_shape=input_shape,padding='same',activation='relu'))
D.add(Conv2D(128, 5, strides=2, padding='same',activation='relu'))
D.add(Conv2D(256, 5, strides=2, padding='same',activation='relu'))
D.add(Conv2D(512, 5, strides=1, padding='same',activation='relu'))
D.add(Flatten())
D.add(Dense(1,activation='sigmoid'))
D.summary()
```

GAN - Discriminator 判別器 – keras example

- Discriminator Network
  - 如果把ReLU改成LeakyReLU，並加入Dropout效果會好很多。

```
D = Sequential()
input_shape = (1,28,28)
D.add(Conv2D(64, 5, strides=2, input_shape=input_shape,padding='same',activation=LeakyReLU(alpha=0.2)))
D.add(Dropout(0.4))
D.add(Conv2D(128, 5, strides=2, padding='same',activation=LeakyReLU(alpha=0.2)))
D.add(Dropout(0.4))
D.add(Conv2D(256, 5, strides=2, padding='same',activation=LeakyReLU(alpha=0.2)))
D.add(Dropout(0.4))
D.add(Conv2D(512, 5, strides=1, padding='same',activation=LeakyReLU(alpha=0.2)))
D.add(Dropout(0.4))
D.add(Flatten())
D.add(Dense(1))
D.add(Activation('sigmoid'))
D.summary()
```

GAN - keras example

- 定義訓練Discriminator & Generator網路
  - 通常會把Discriminator的學習率調高點。
  - 訓練Generator需要考慮到Discriminator，所以創建AM(adversarial Model)。

```
optimizer = RMSprop(lr=0.0002, clipvalue=1.0, decay=6e-8)
D.compile(loss='binary_crossentropy', optimizer=optimizer,metrics=['accuracy'])
D.summary()
```

```
optimizer = RMSprop(lr=0.0001, decay=3e-8)
AM = Sequential()
AM.add(G)
AM.add(D)
AM.compile(loss='binary_crossentropy', optimizer=optimizer,metrics=['accuracy'])
AM.summary()
```

THE END