深度學習TensorFlow實務

驗證碼辨識

Lab7

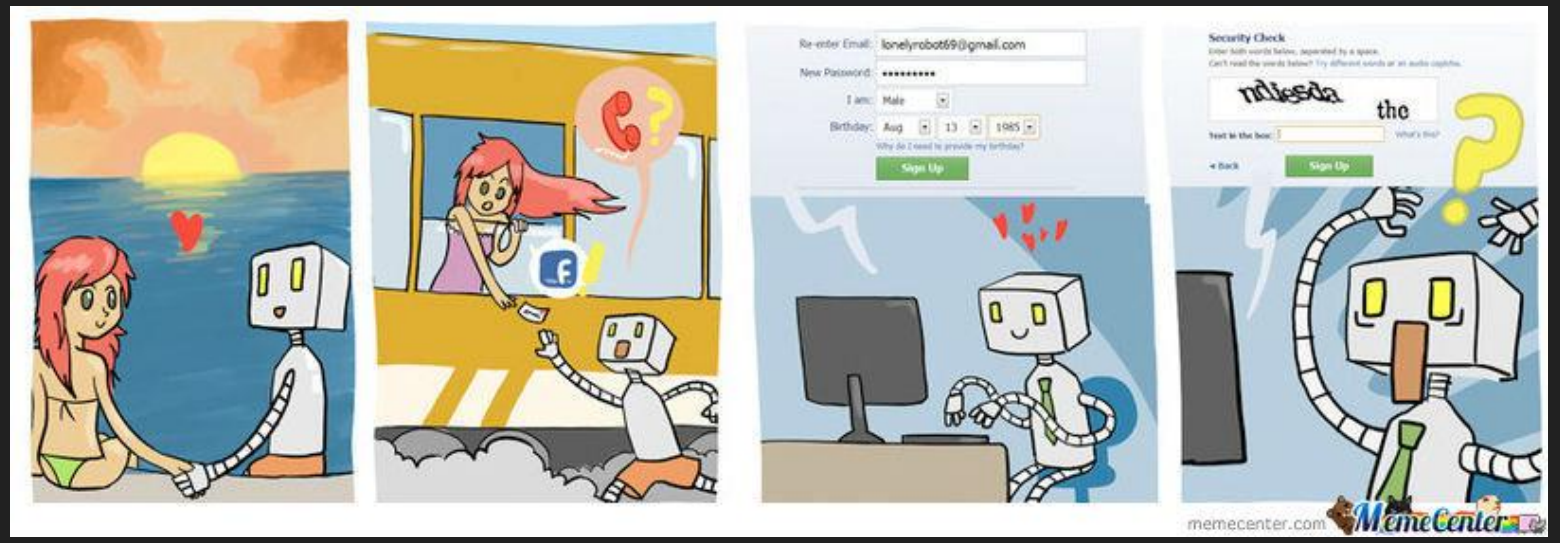
-TA-
李偉弘
廖宜健
林佑昌
蔡明諺
彭冠偉

1. 驗證碼介紹

什麼是驗證碼?

- 驗證碼(Captcha)是根據隨機字元生成一幅圖片，然後在圖片中加入干擾像素，使用者必須手動填入，防止有人利用機器人自動批量註冊、灌水、發垃圾廣告等等

驗證碼的作用是驗證用戶是真人還是機器人; 設計理念是對人友好，對機器難。



CAPTCHA 測試定義

- 全自動區分電腦和人類的公開圖靈測試 (Completely Automated Public Turing test to tell Computers and Humans Apart，CAPTCHA)
- 根據 CAPTCHA 測試的定義，產生驗證碼圖片的演算法必須公開，即使該演算法可能有專利保護。這樣做是證明想破解就需要解決一個不同的人工智慧難題，而非僅靠發現原來的秘密演算法，而後者可以用還原工程等途徑得到

驗證碼種類

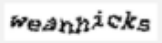
Scheme	Website(s)	Example	Security Features		Excluded Characters
			Anti-segmentation	Anti-recognition	
Wikipedia	wikipedia.org		Overlapping characters, Enligh letters	Rotation, distortion, waving	–
Microsoft	{live, bing, miscosoft}.com {office, linkedin}.com		Overlapping characters, solid background	Different font styles, varied font sizes, rotation, waving	0, 1, 5, D, G, I, Q, U
eBay	ebay.com		Overlapping characters, Only arabic numerals	Character rotating, distortion and waving	–
Baidu	{baidu, qq}.com		Occluding lines, overlapping, only Enligh letters	Varied font size, color, rotation, disortion and waving	Z
Google	google.{com,co.in,co.jp, co.uk,ru,com.br,fr com.hk,it,ca,es,com.mx} youtube.com		Overlapping characters, Enligh letters	Varied font sizes & color, rotation, disortion, waving	–
Alipay	{alipay, tmall}.com {taobao, login.tmall}.com alipayexpress.com		English letters and arabic numerals, overlapping characters	Rotation and distortion	0, 1, I, L, O
JD	jd.com		English letters and arabic numerals, overlapping characters	Rotation and distortion	0, 1, 2, 7, 9, D, G, I, J, L, O, P, Q, Z
Qihu360	360.cn		English letters and arabic numerals, overlapping characters	Varied font sizes, rotation and distortion	0, I, L, O, T, i, l, o, t, q
Sina	sina.cn		English letters and arabic numerals, overlapping characters	Rotation, distortion, waving	1, 9, 0, D, I, J, L, O, T, i, j, l, o, t, g, r
Weibo	weibo.cn		English letters and arabic numerals, overlapping characters, occluding lines	Rotation and distortion	0, 1, 5, D, G, I, Q, U
Sohu	sohu.com		Complex background, occluding lines, and overlapping	Varied font size, color and rotation	0, 1, i, l, o, z

Table 1: Text-based captcha schemes tested in our experiments.

2. 驗證碼生成

驗證碼來源?

- 或許你可以寫一支爬蟲程序，截取個 5 萬張驗證碼，再自己手動標記答案上去，但這樣有點太費時了，或許我們可以試着模仿產生一些驗證碼看看。
- 不過當然，我們產生的訓練集必須非常接近真實的驗證碼，否則最後訓練完可能用在真實的驗證碼上效果會非常的差。

驗證碼生成

- Code：7_1 生成驗證碼與tfrecord.ipynb

LAB 7	9/9	驗證碼辨識	7_1 生成驗證碼與tfrecord 7_2 多任務驗證碼辨識 7_3 驗證模型

OR

驗證碼生成

- 定義驗證碼包含 0-9 字元

In [1]:
```
from captcha.image import ImageCaptcha
import numpy as np
from PIL import Image
import random
import sys
import os
import tensorflow as tf
import matplotlib.pyplot as plt
```

In [2]:
```
number = [str(i) for i in range(10)]
print(number)
CHAR_SET = number
```

```
['0', '1', '2', '3', '4', '5', '6', '7', '8', '9']
```

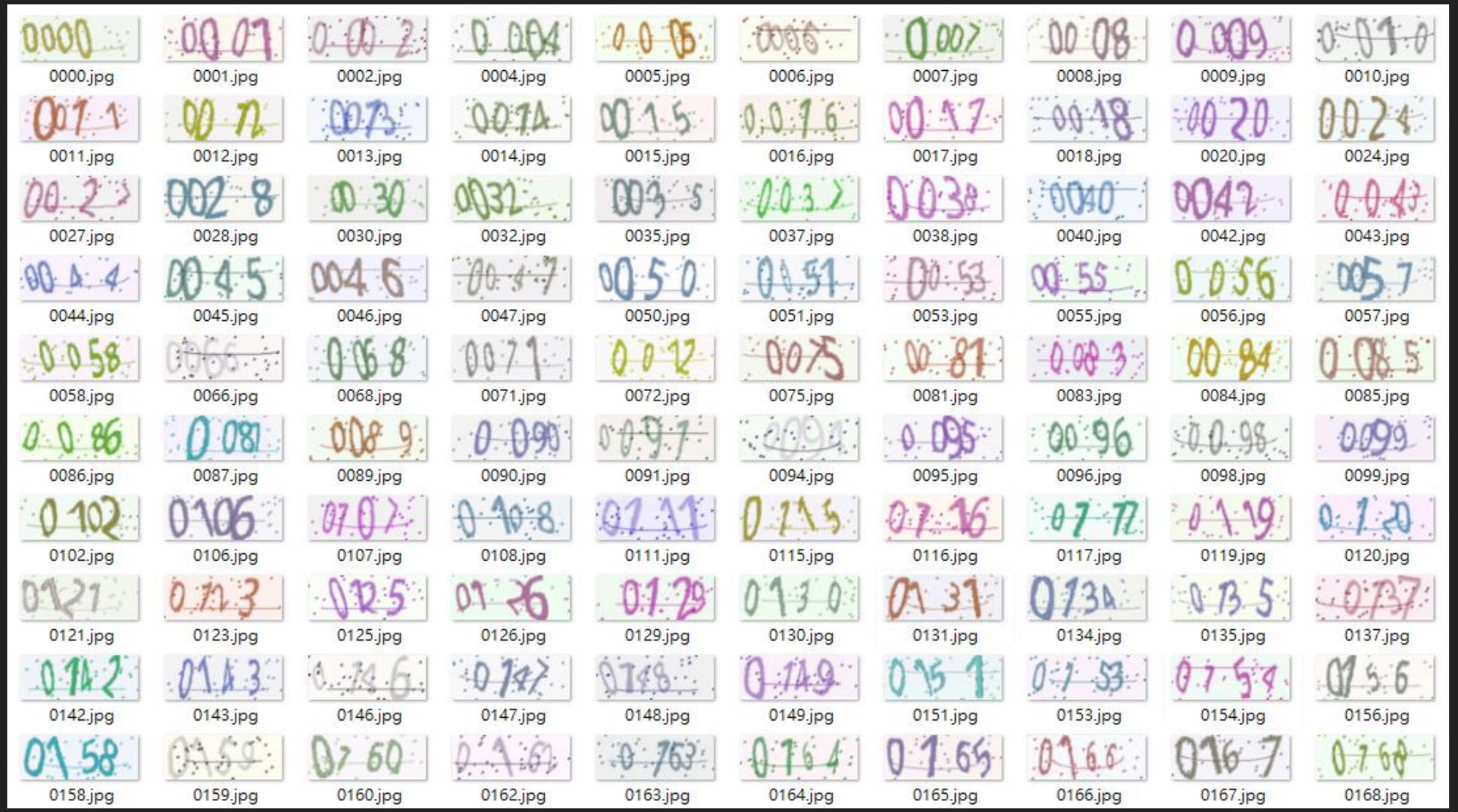
驗證碼生成

- 隨機生成4個字元，並輸出驗證碼圖片

In [3]:
```
# 隨機生成4個字元
def random_captcha_text(char_set=number,captcha_size=4):
    return ''.join(random.choices(CHAR_SET,k=4))

# 生成對應字元的驗證碼
def gen_captcha_text_and_image(path):
    image_ = ImageCaptcha()
    captcha_text = random_captcha_text()
    captcha = image_.generate(captcha_text)
    image_.write(captcha_text, path + captcha_text + '.jpg')

path = 'captcha/images/'
num = 10000

if not os.path.exists(path):
    os.makedirs(path)
for i in range(num):
    gen_captcha_text_and_image(path)
    sys.stdout.write('\r>> Creating image %d/%d' % (i+1,num))
    sys.stdout.flush()
sys.stdout.write('\n')
sys.stdout.flush()
```

```
>> Creating image 10000/10000
```

驗證碼生成


0000.jpg
0001.jpg
0002.jpg
0004.jpg
0005.jpg
0006.jpg
0007.jpg
0008.jpg
0009.jpg
0010.jpg
0011.jpg
0012.jpg
0013.jpg
0014.jpg
0015.jpg
0016.jpg
0017.jpg
0018.jpg
0020.jpg
0024.jpg
0027.jpg
0028.jpg
0030.jpg
0032.jpg
0035.jpg
0037.jpg
0038.jpg
0040.jpg
0042.jpg
0043.jpg
0044.jpg
0045.jpg
0046.jpg
0047.jpg
0050.jpg
0051.jpg
0053.jpg
0055.jpg
0056.jpg
0057.jpg
0058.jpg
0066.jpg
0068.jpg
0071.jpg
0072.jpg
0075.jpg
0081.jpg
0083.jpg
0084.jpg
0085.jpg
0086.jpg
0087.jpg
0089.jpg
0090.jpg
0091.jpg
0094.jpg
0095.jpg
0096.jpg
0098.jpg
0099.jpg
0102.jpg
0106.jpg
0107.jpg
0108.jpg
0111.jpg
0115.jpg
0116.jpg
0117.jpg
0119.jpg
0120.jpg
0121.jpg
0123.jpg
0125.jpg
0126.jpg
0129.jpg
0130.jpg
0131.jpg
0134.jpg
0135.jpg
0137.jpg
0142.jpg
0143.jpg
0146.jpg
0147.jpg
0148.jpg
0149.jpg
0151.jpg
0153.jpg
0154.jpg
0156.jpg
0158.jpg
0159.jpg
0160.jpg
0162.jpg
0163.jpg
0164.jpg
0165.jpg
0166.jpg
0167.jpg
0168.jpg

3. 驗證碼辨識方法

驗證碼辨識-方法一

- 把標籤轉為向量，向量長度為 40
- 有一個驗證碼：0782
- 它的標籤可以轉為長度為 40 的向量

1000000000 0000000100 0000000010 0010000000

0 7 8 2

- 訓練方法跟 0-9 手寫數字辨識類似

驗證碼辨識-方法二

- 將一張驗證碼圖片拆成 4 個標籤
- 有一個驗證碼：0782

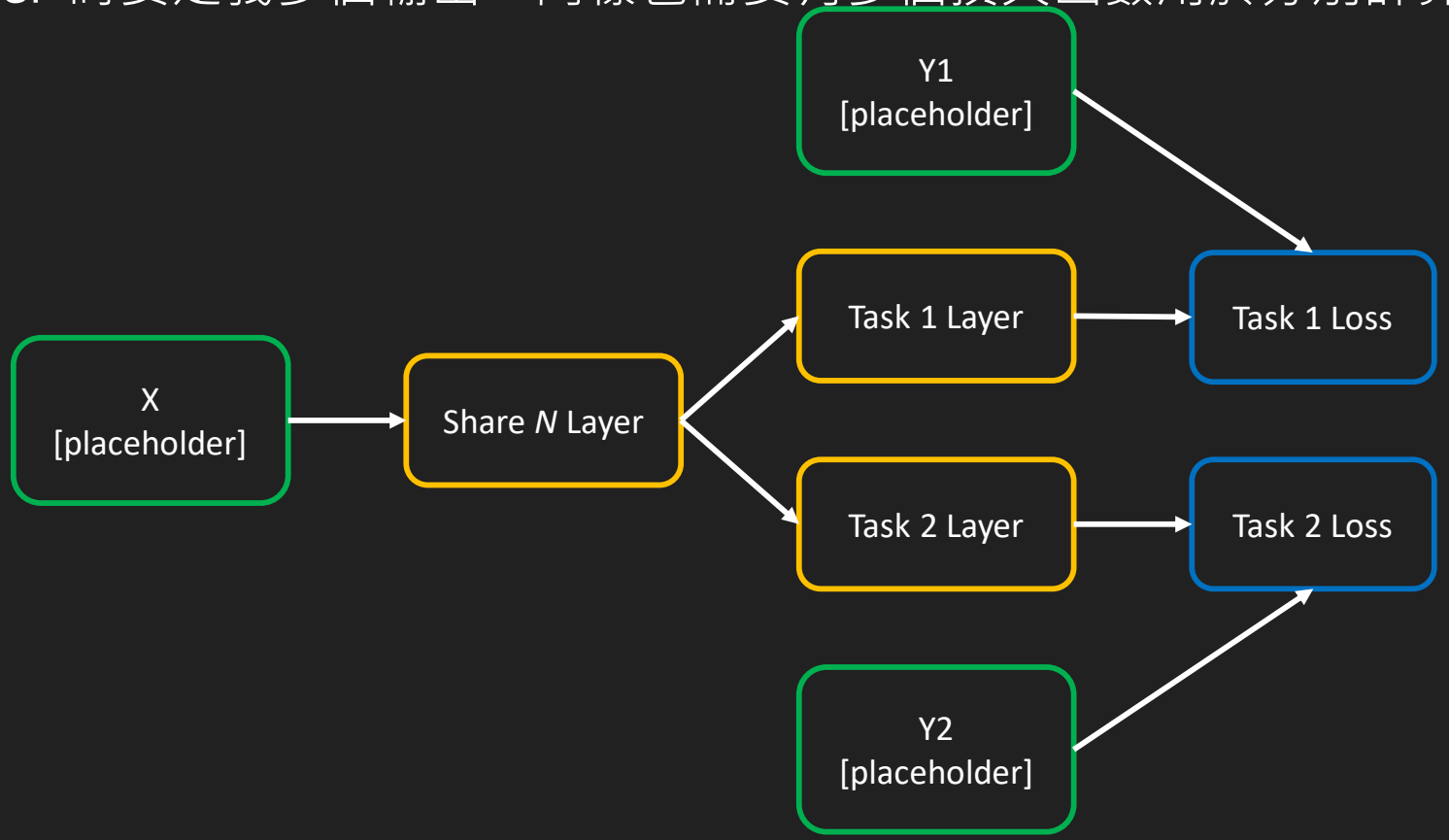
```
Label_0：1000000000
Label_1：0000000100
Label_2：0000000010
Label_3：0010000000
```

- 可以使用多任務學習(Multi-task Learning)

4. 多任務學習

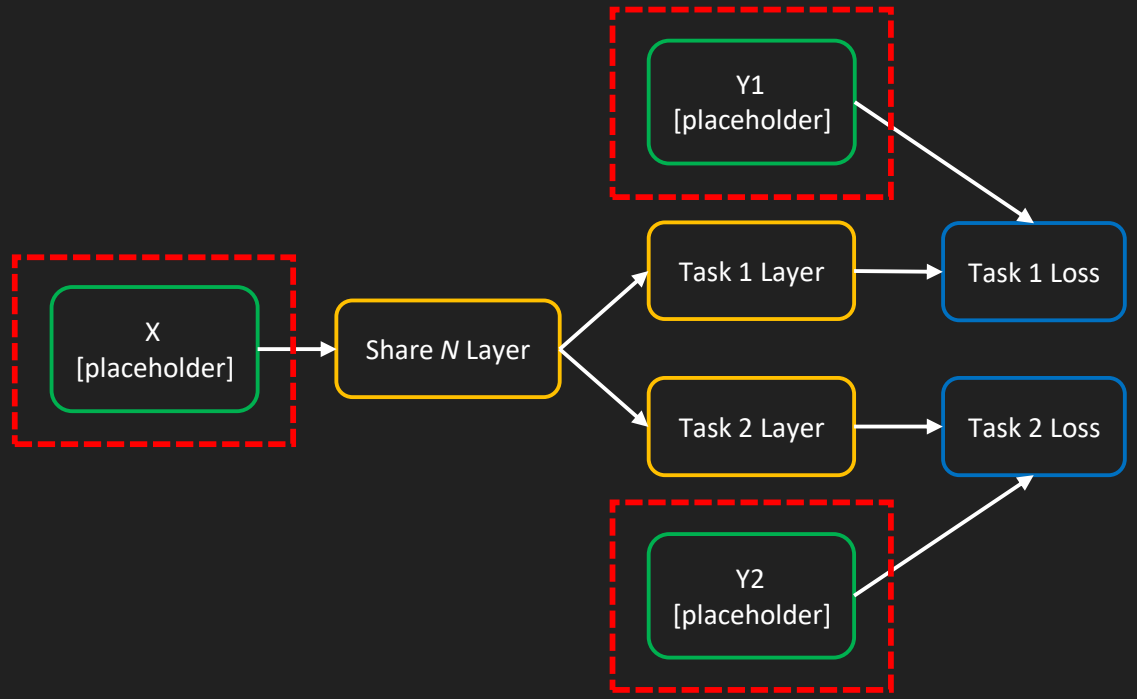
建立多任務圖

- 多任務的一個特點是，單一個張量輸入（X）以及多個輸出（Y_1, Y_2）。因此在定義 placeholder 時要定義多個輸出。同樣也需要有多個損失函數用於分別計算每個任務的損失

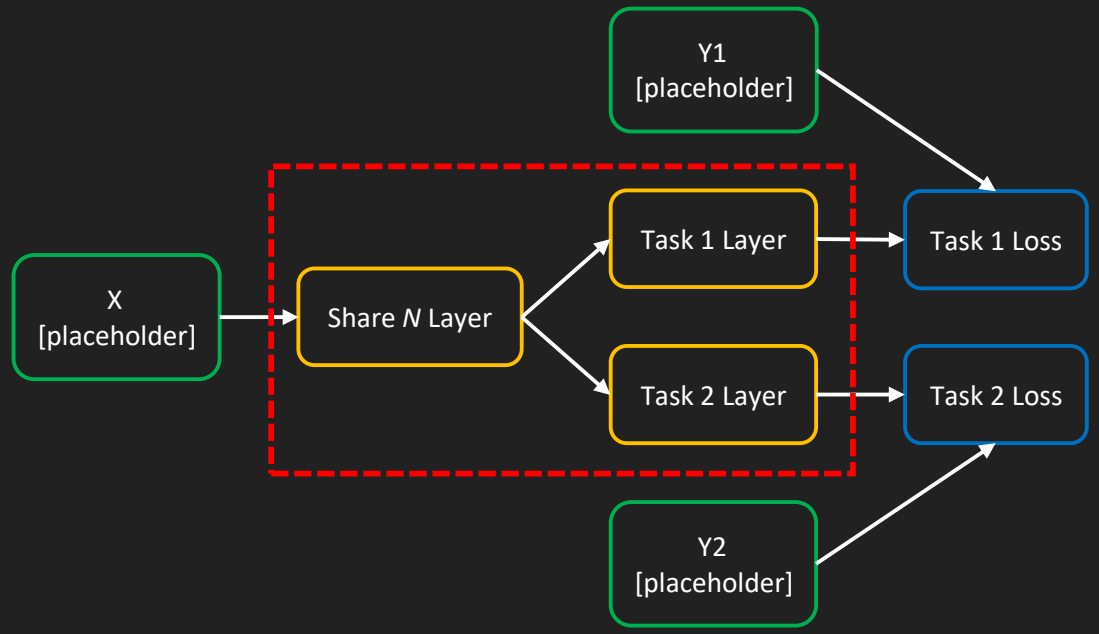


多任務圖

```
import Tensorflow as tf

# 定義佔位符
X = tf.placeholder("float", [10, 10], name="X")
Y1 = tf.placeholder("float", [10, 20], name="Y1")
Y2 = tf.placeholder("float", [10, 20], name="Y2")
```



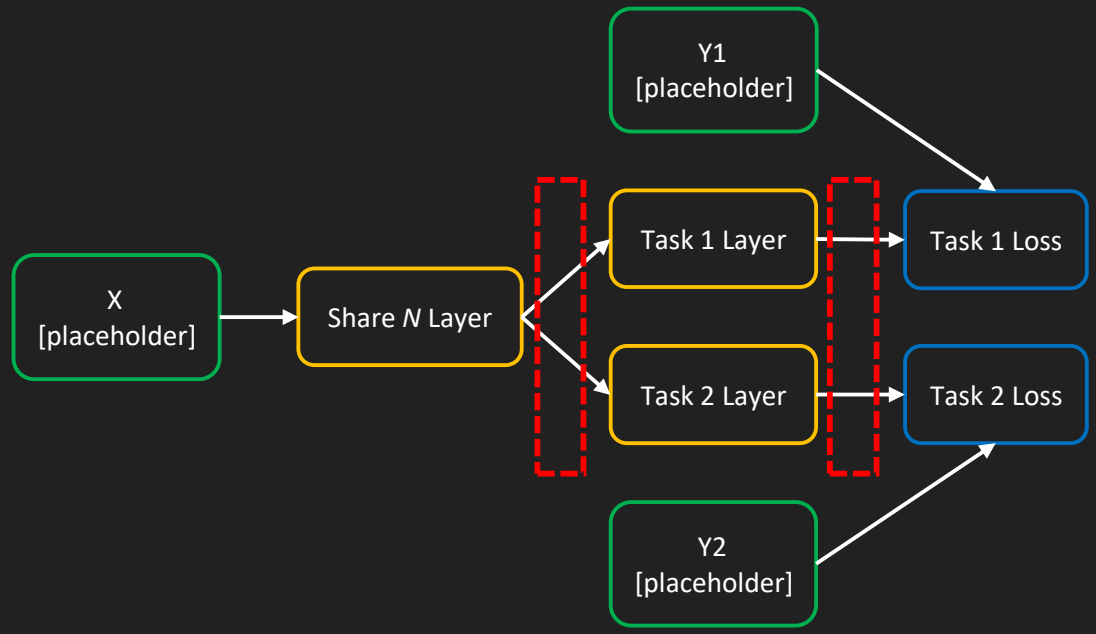
多任務圖




```
import Tensorflow as tf

# 定義佔位符
X = tf.placeholder("float", [10, 10], name="X")
Y1 = tf.placeholder("float", [10, 20], name="Y1")
Y2 = tf.placeholder("float", [10, 20], name="Y2")

# 定義權重
initial_shared_layer_weights = np.random.rand(10,20)
initial_Y1_layer_weights = np.random.rand(20,20)
initial_Y2_layer_weights = np.random.rand(20,20)

shared_layer_weights = tf.Variable(initial_shared_layer_weights, name="share_W",
dtype="float32")
Y1_layer_weights = tf.Variable(initial_Y1_layer_weights, name="share_Y1", dtype="float32")
Y2_layer_weights = tf.Variable(initial_Y2_layer_weights, name="share_Y2", dtype="float32")
```

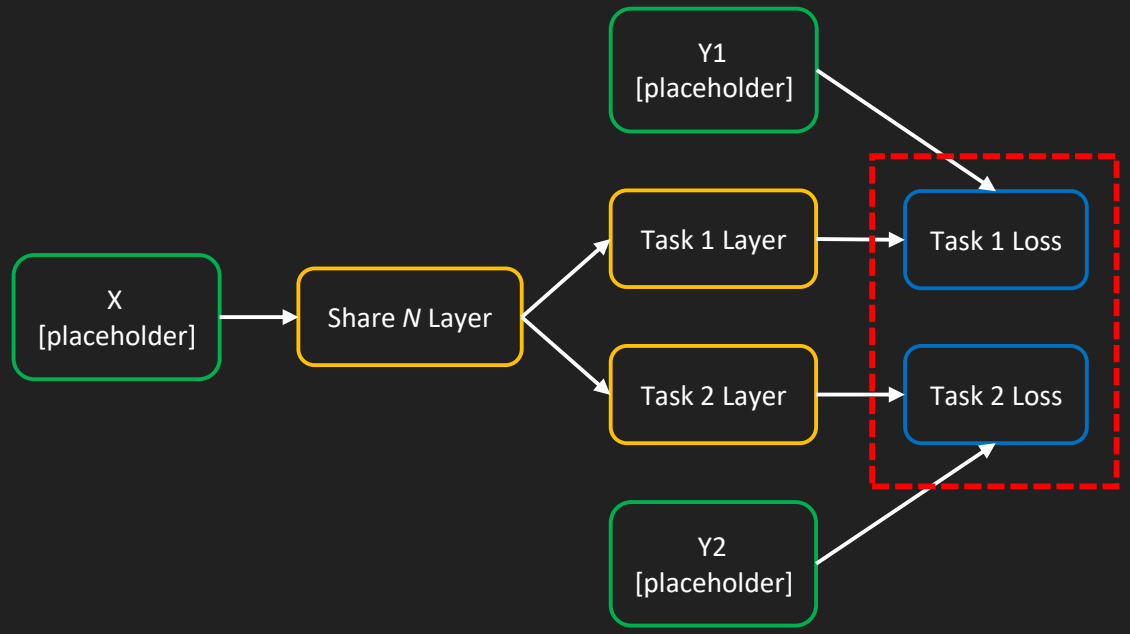
多任務圖

```
import Tensorflow as tf

# 定義佔位符
X = tf.placeholder("float", [10, 10], name="X")
Y1 = tf.placeholder("float", [10, 20], name="Y1")
Y2 = tf.placeholder("float", [10, 20], name="Y2")

# 定義權重
initial_shared_layer_weights = np.random.rand(10,20)
initial_Y1_layer_weights = np.random.rand(20,20)
initial_Y2_layer_weights = np.random.rand(20,20)

shared_layer_weights = tf.Variable(initial_shared_layer_weights, name="share_W", dtype="float32")
Y1_layer_weights = tf.Variable(initial_Y1_layer_weights, name="share_Y1", dtype="float32")
Y2_layer_weights = tf.Variable(initial_Y2_layer_weights, name="share_Y2", dtype="float32")

# 使用relu激活函數
shared_layer = tf.nn.relu(tf.matmul(X,shared_layer_weights))
Y1_layer = tf.nn.relu(tf.matmul(shared_layer,Y1_layer_weights))
Y2_layer = tf.nn.relu(tf.matmul(shared_layer,Y2_layer_weights))
```



Feed forward

多任務圖



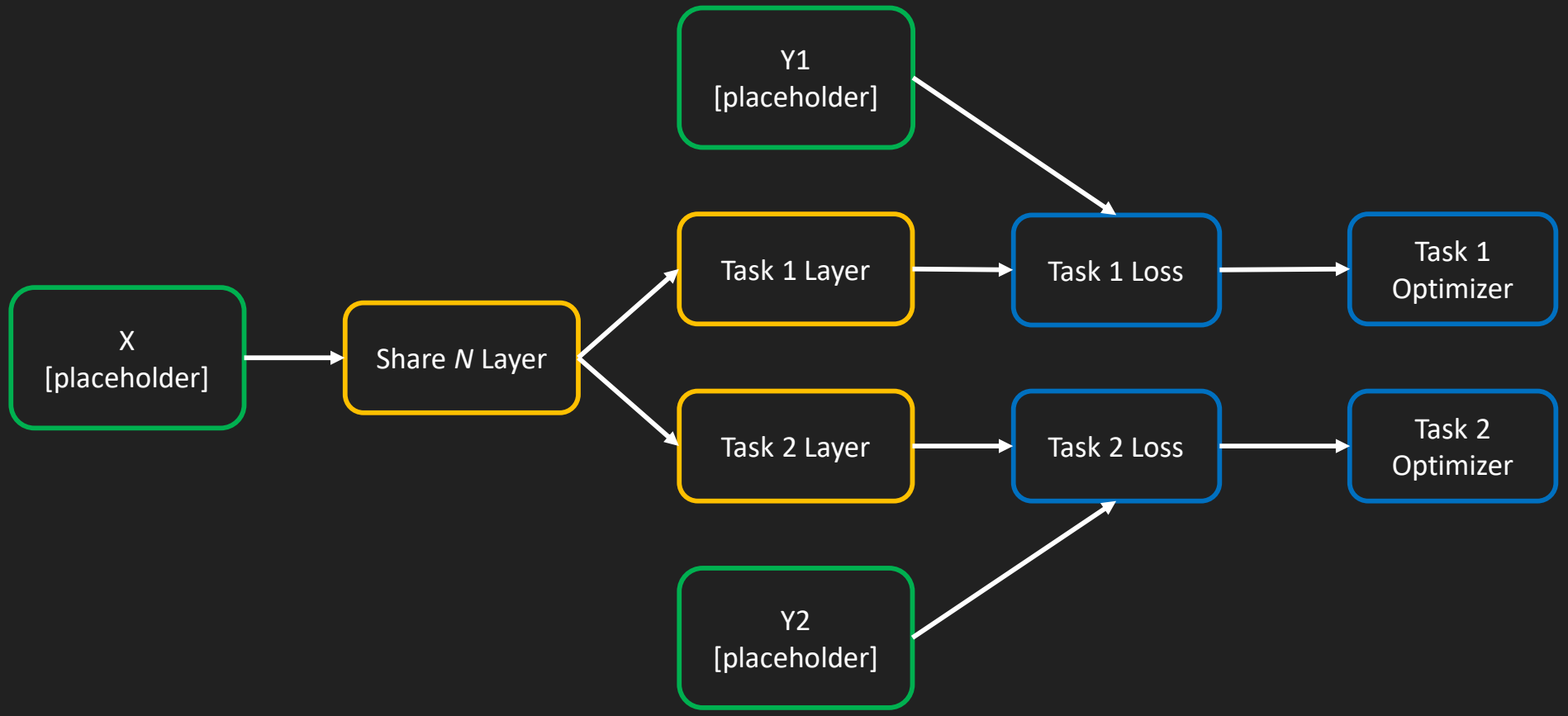

```
import Tensorflow as tf

# 定義佔位符
X = tf.placeholder("float", [10, 10], name="X")
Y1 = tf.placeholder("float", [10, 20], name="Y1")
Y2 = tf.placeholder("float", [10, 20], name="Y2")

# 定義權重
initial_shared_layer_weights = np.random.rand(10,20)
initial_Y1_layer_weights = np.random.rand(20,20)
initial_Y2_layer_weights = np.random.rand(20,20)

shared_layer_weights = tf.Variable(initial_shared_layer_weights, name="share_W", dtype="float32")
Y1_layer_weights = tf.Variable(initial_Y1_layer_weights, name="share_Y1", dtype="float32")
Y2_layer_weights = tf.Variable(initial_Y2_layer_weights, name="share_Y2", dtype="float32")

# 使用relu激活函數
shared_layer = tf.nn.relu(tf.matmul(X,shared_layer_weights))
Y1_layer = tf.nn.relu(tf.matmul(shared_layer,Y1_layer_weights))
Y2_layer = tf.nn.relu(tf.matmul(shared_layer,Y2_layer_weights))

# 計算loss
Y1_Loss = tf.nn.l2_loss(Y1-Y1_layer)
Y2_Loss = tf.nn.l2_loss(Y2-Y2_layer)
```

多任務學習

- 有了網路的建構後，接下來是訓練，有兩種方式：
  - 交替訓練
  - 聯合訓練

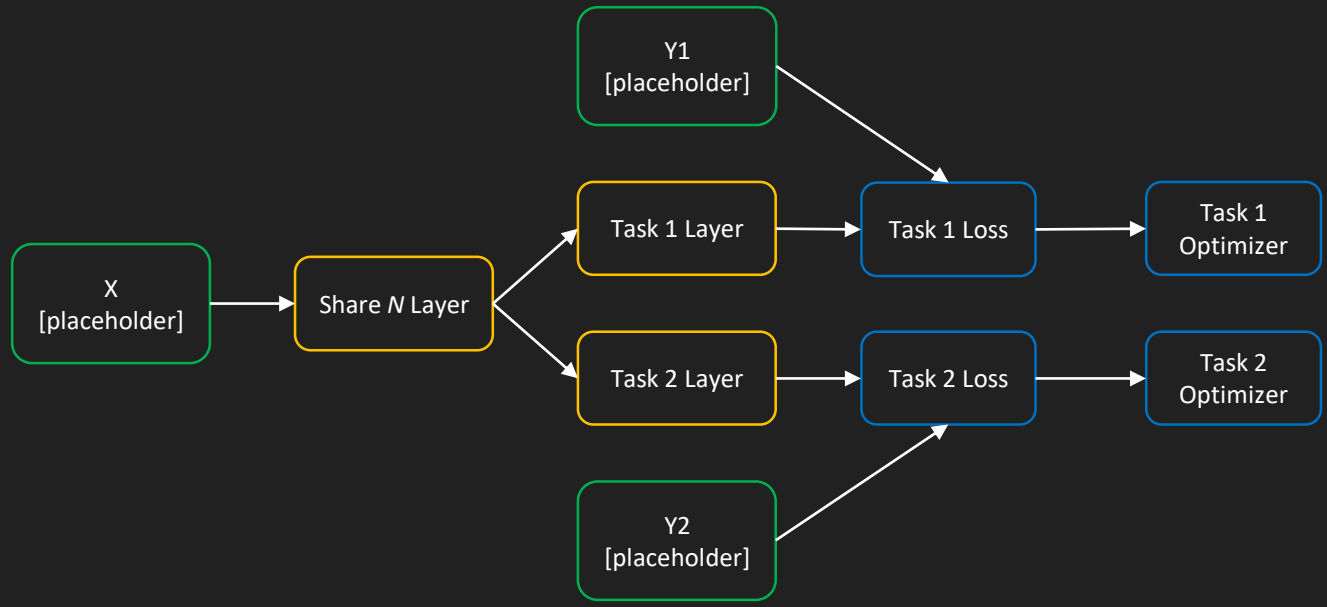
多任務學習

- 交替訓練


Y1
[placeholder]
X
[placeholder]
Share *N* Layer
Task 1 Layer
Task 1 Loss
Task 1
Optimizer
Task 2 Layer
Task 2 Loss
Task 2
Optimizer
Y2
[placeholder]

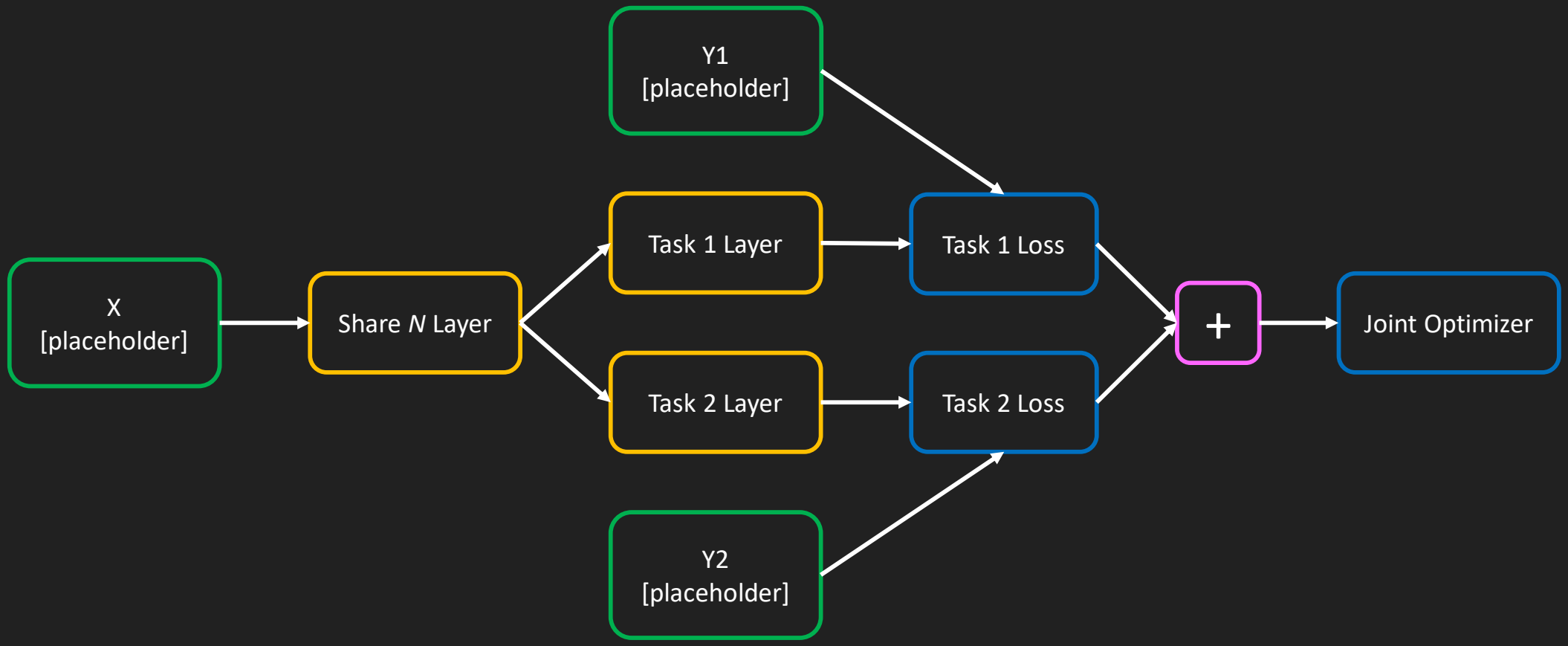
交替訓練

```
# 優化器
Y1_op = tf.train.AdamOptimizer().minimize(Y1_Loss)
Y2_op = tf.train.AdamOptimizer().minimize(Y2_Loss)

# Calculation (Session) Code
with tf.Session() as session:
    session.run(tf.initialize_all_variables())
    for iters in range(10):
        if np.random.rand() < 0.5:
            _, Y1_loss = session.run([Y1_op, Y1_Loss],
                        {
                          X: np.random.rand(10,10)*10,
                          Y1: np.random.rand(10,20)*10,
                          Y2: np.random.rand(10,20)*10
                          })
            print(Y1_loss)
        else:
            _, Y2_loss = session.run([Y2_op, Y2_Loss],
                        {
                          X: np.random.rand(10,10)*10,
                          Y1: np.random.rand(10,20)*10,
                          Y2: np.random.rand(10,20)*10
                          })
            print(Y2_loss)
```

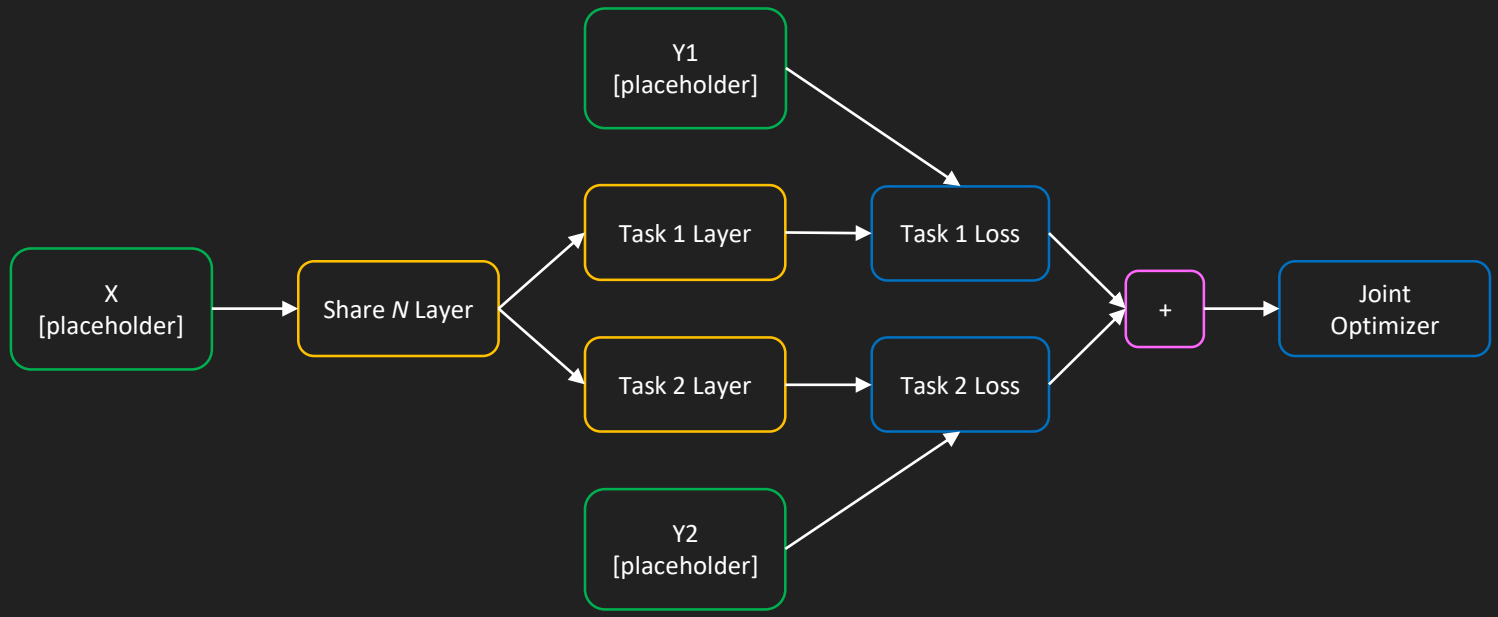


多任務學習

- 聯合訓練


Y1
[placeholder]
X
[placeholder]
Share *N* Layer
Task 1 Layer
Task 2 Layer
Task 1 Loss
Task 2 Loss
+
Joint Optimizer
Y2
[placeholder]

交替訓練




```
# 計算loss
Joint_Loss = Y1_Loss + Y2_Loss

# 優化器
Optimiser = tf.train.AdamOptimizer().minimize(Joint_Loss)

# Calculation (Session) Code
with tf.Session() as session:
    session.run(tf.initialize_all_variables())
    _, Joint_Loss = session.run([Optimiser, Joint_Loss],
                    {
                      X: np.random.rand(10,10)*10,
                      Y1: np.random.rand(10,20)*10,
                      Y2: np.random.rand(10,20)*10
                      })
    print(Joint_Loss)
```

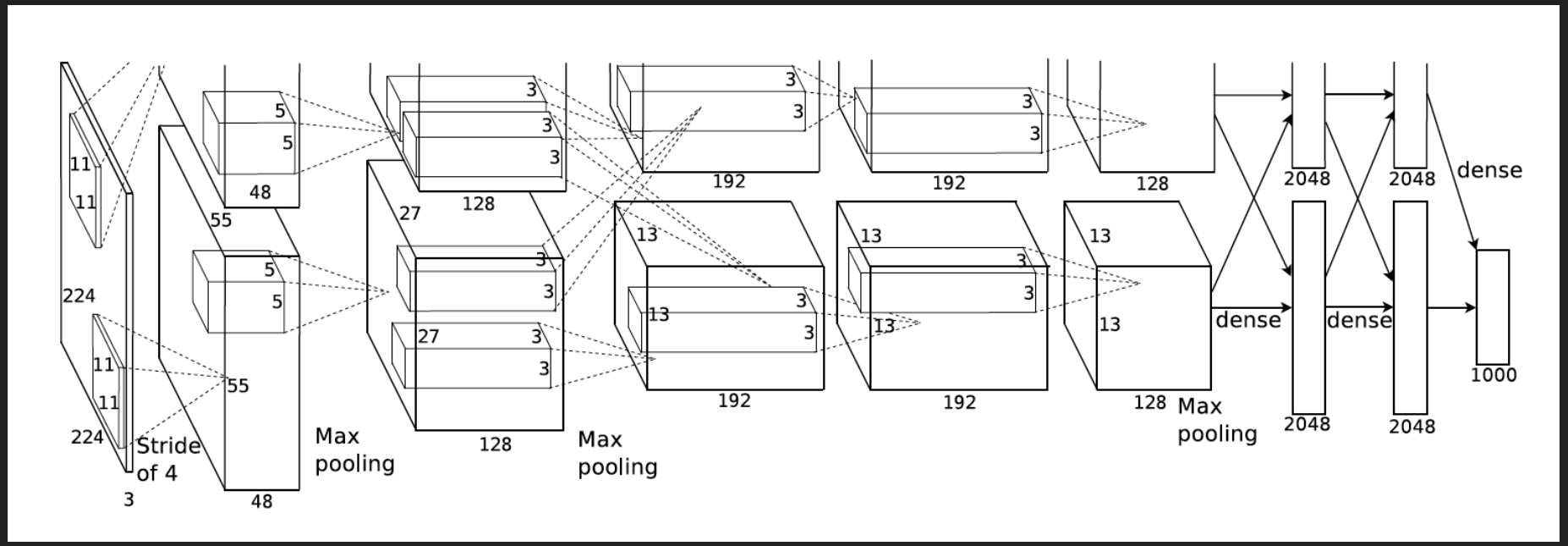
4. 程式碼

驗證碼生成

- 開啟「**7_1 生成驗證碼與tfrecord.ipynb**」
- 將驗證碼圖片與數字標籤轉成 tfrecord
  - train.tfrecord
  - test.tfrecord

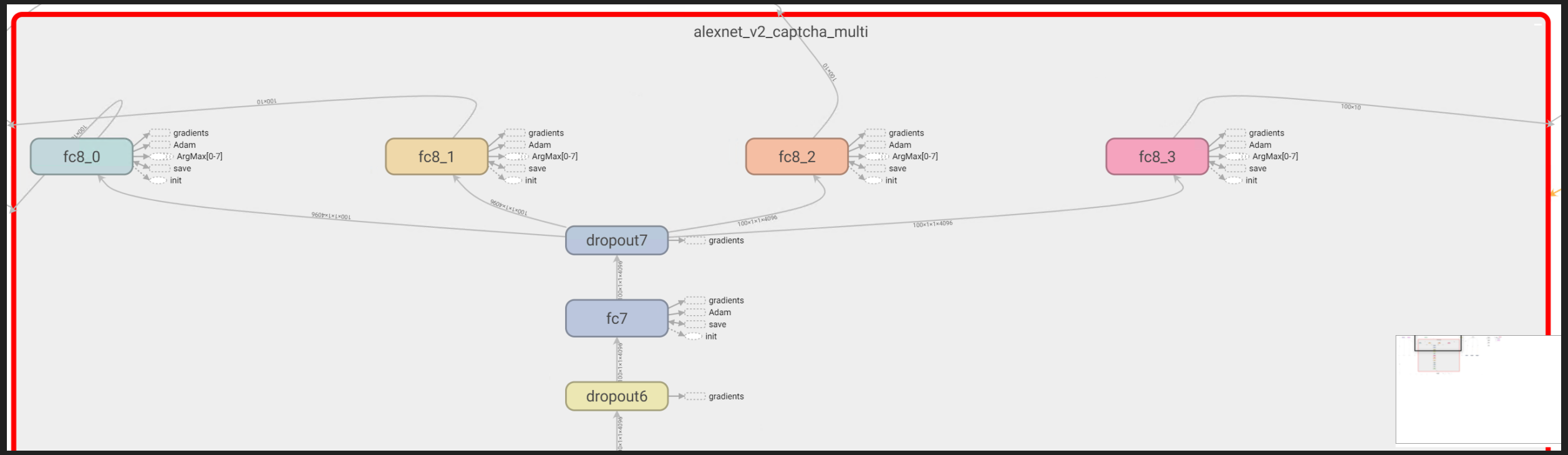
多任務學習模型訓練

- 以 AlexNetv2 為基礎網路



多任務學習模型訓練

- 將原本 AlexNetv2 輸出，由單一任務改為多任務
- 4 個網路輸出：fc8_0, fc8_1, fc8_2, fc8_3

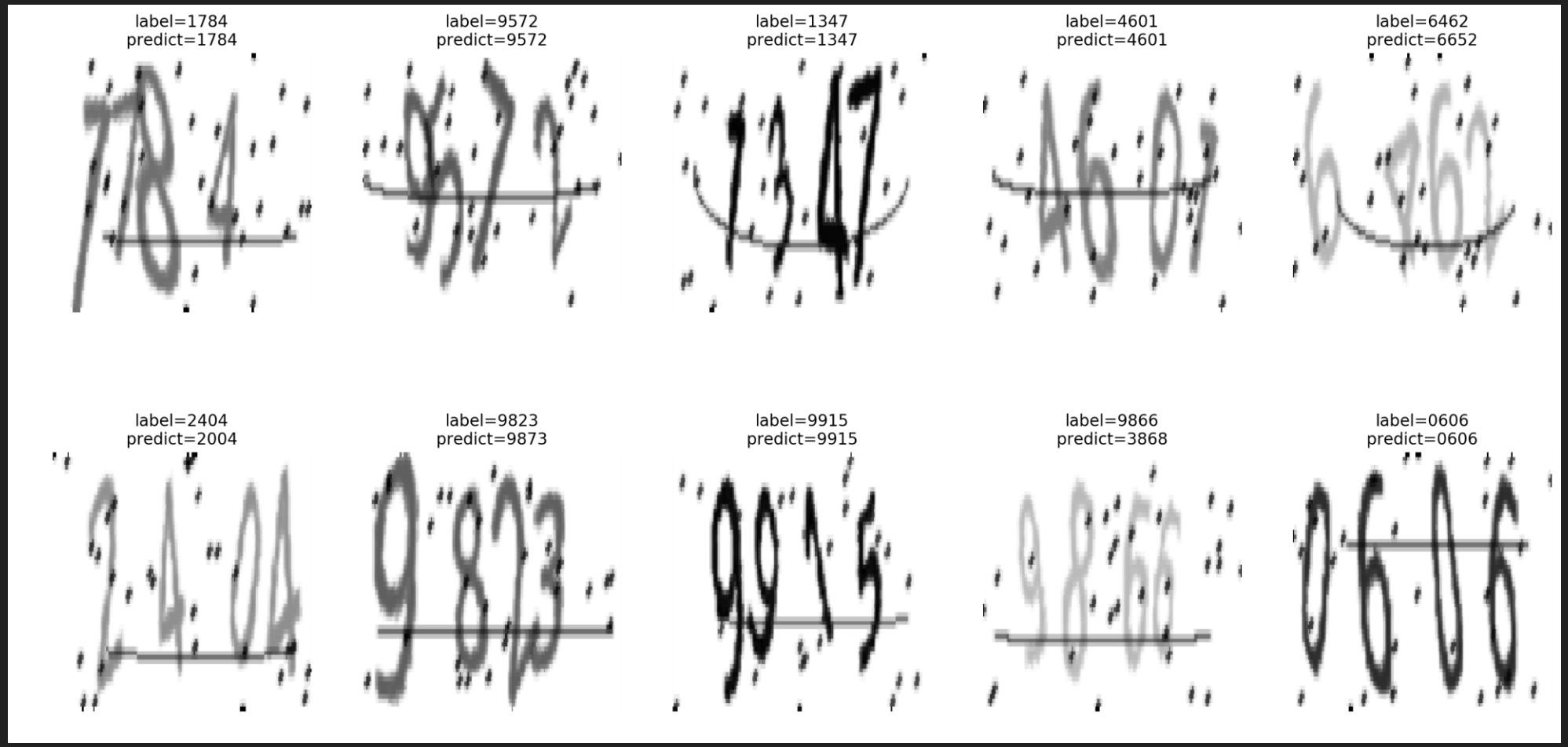


多任務學習模型訓練

```
Iter:0/1359 epoch:1,  Loss:10.469  Accuracy:0.15,0.15,0.09,0.12  Learning_rate:0.00100
Iter:20/1359 epoch:1,  Loss:2.311  Accuracy:0.10,0.06,0.07,0.11  Learning_rate:0.00100
Iter:40/1359 epoch:1,  Loss:2.302  Accuracy:0.12,0.09,0.12,0.14  Learning_rate:0.00100
Iter:60/1359 epoch:2,  Loss:2.303  Accuracy:0.14,0.07,0.06,0.04  Learning_rate:0.00100
Iter:80/1359 epoch:2,  Loss:2.299  Accuracy:0.14,0.10,0.17,0.09  Learning_rate:0.00100
Iter:100/1359 epoch:3,  Loss:2.299  Accuracy:0.13,0.11,0.11,0.12  Learning_rate:0.00100
Iter:120/1359 epoch:3,  Loss:2.303  Accuracy:0.16,0.11,0.08,0.09  Learning_rate:0.00100
Iter:140/1359 epoch:4,  Loss:2.303  Accuracy:0.11,0.09,0.11,0.09  Learning_rate:0.00100
Iter:160/1359 epoch:4,  Loss:2.301  Accuracy:0.16,0.15,0.12,0.12  Learning_rate:0.00100
Iter:180/1359 epoch:4,  Loss:2.301  Accuracy:0.08,0.14,0.09,0.16  Learning_rate:0.00100
Iter:200/1359 epoch:5,  Loss:2.302  Accuracy:0.09,0.12,0.16,0.12  Learning_rate:0.00100
Iter:220/1359 epoch:5,  Loss:2.304  Accuracy:0.12,0.14,0.11,0.09  Learning_rate:0.00100
Iter:240/1359 epoch:6,  Loss:2.303  Accuracy:0.11,0.18,0.07,0.12  Learning_rate:0.00100
Iter:260/1359 epoch:6,  Loss:2.306  Accuracy:0.09,0.09,0.09,0.10  Learning_rate:0.00100
Iter:280/1359 epoch:7,  Loss:2.307  Accuracy:0.08,0.11,0.09,0.11  Learning_rate:0.00100
Iter:300/1359 epoch:7,  Loss:2.303  Accuracy:0.06,0.12,0.12,0.12  Learning_rate:0.00100
Iter:320/1359 epoch:8,  Loss:2.305  Accuracy:0.11,0.05,0.16,0.09  Learning_rate:0.00050
Iter:340/1359 epoch:8,  Loss:2.296  Accuracy:0.06,0.12,0.16,0.09  Learning_rate:0.00050
Iter:360/1359 epoch:8,  Loss:2.248  Accuracy:0.14,0.10,0.16,0.16  Learning_rate:0.00050
Iter:380/1359 epoch:9,  Loss:2.181  Accuracy:0.18,0.17,0.16,0.20  Learning_rate:0.00050
Iter:400/1359 epoch:9,  Loss:2.080  Accuracy:0.14,0.28,0.21,0.13  Learning_rate:0.00050
Iter:420/1359 epoch:10,  Loss:2.095  Accuracy:0.21,0.23,0.23,0.19  Learning_rate:0.00050
Iter:440/1359 epoch:10,  Loss:1.957  Accuracy:0.23,0.23,0.22,0.23  Learning_rate:0.00050
Iter:460/1359 epoch:11,  Loss:1.834  Accuracy:0.32,0.37,0.29,0.25  Learning_rate:0.00050
Iter:480/1359 epoch:11,  Loss:1.814  Accuracy:0.38,0.25,0.17,0.28  Learning_rate:0.00050
Iter:500/1359 epoch:12,  Loss:1.588  Accuracy:0.43,0.41,0.29,0.46  Learning_rate:0.00050
```

驗證模型

- 7_3 驗證模型.ipynb

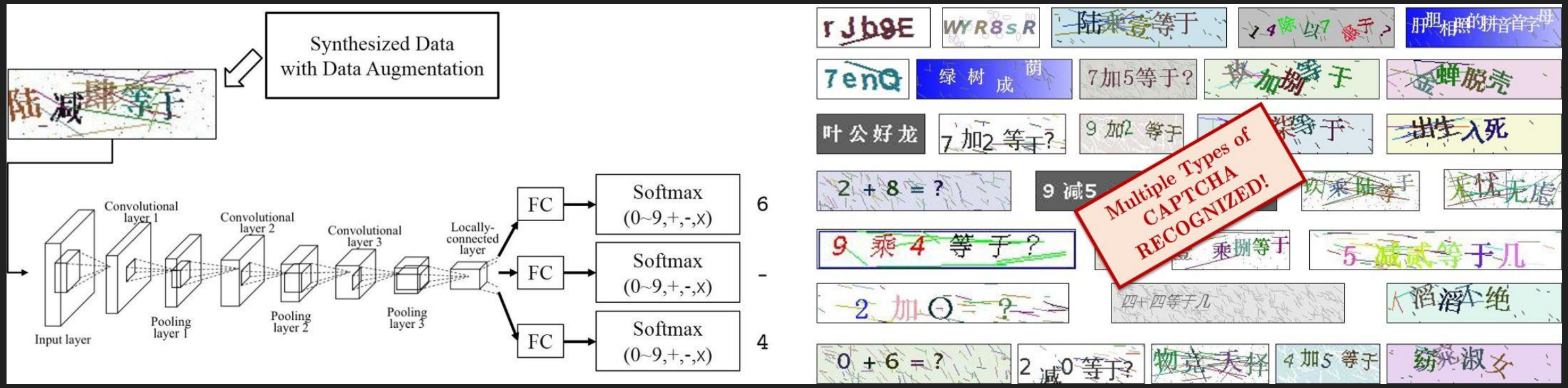


Reference

- JasonLiTW，simple-railway-captcha-solver，基於 CNN 的台鐵訂票驗證碼辨識以及驗證性高的訓練集產生器，GitHub [Link]
- Parker-Lyu，TensorFLow-Learning，B站上煉數成金的公開課筆記，GitHub [Link]
- Jonathan Godwin，Multi-Task Learning in Tensorflow [Link]

Good Project about CAPTCHA

- CAPTCHA Cracking: A CNN Based OCR Module for Web Crawler
- http://wangchuan.github.io/archive/projects/captcha-crack/



-END-

驗證碼生成

- 安裝...

```
> pip install captcha
```