1 Fashion MNIST 練習: CGAN

1.1 Fashion MNIST 資料集

上一節大家都應該有使用過 GAN 來玩進階的 MNIST 資料集，Fashion MNIST，那我們現在來試試看 CGAN，看看能不能根據我們給的標籤來產生我們需要的流行服飾，忘記的讀者可以配合 GAN with MNIST 章節對照

[程式]:
```
import numpy as np
%matplotlib inline
# 我們會使用到一些內建的資料庫, MAC 需要加入以下兩行, 才不會把對方的 ssl 憑證視為無效
import ssl
ssl._create_default_https_context = ssl._create_unverified_context
```

讀取資料集

[程式]:
```
from keras.datasets import fashion_mnist
# 回傳值: ((訓練特徵, 訓練目標), (測試特徵, 測試目標))
(x_train, y_train),(x_test, y_test) = fashion_mnist.load_data()
```

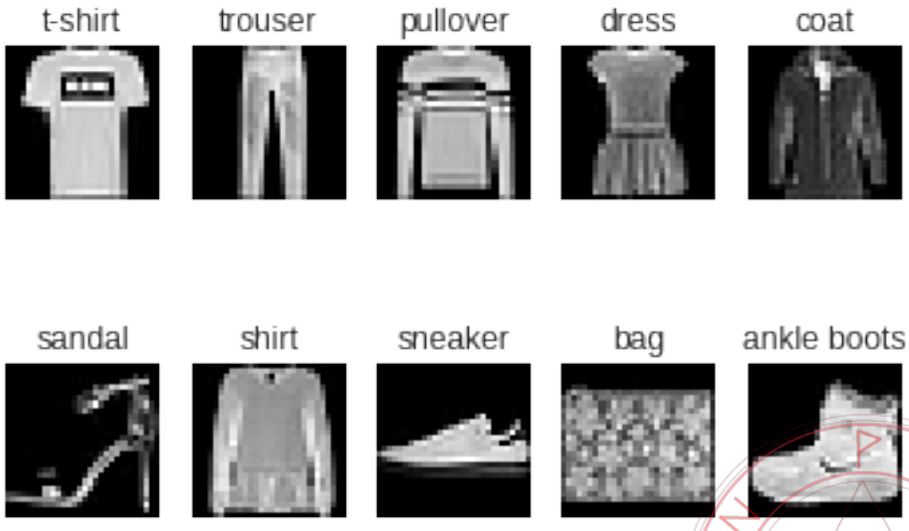
```
Using TensorFlow backend.


Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com
/train-labels-idx1-ubyte.gz
32768/29515 [================================] - 0s 3us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com
/train-images-idx3-ubyte.gz
26427392/26421880 [==============================] - 2s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-
labels-idx1-ubyte.gz
8192/5148 [===============================================] - 0s 0us/step
Downloading data from http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/t10k-
images-idx3-ubyte.gz
4423680/4422102 [==============================] - 1s 0us/step
```

觀看 shape

```
[程式]: x_train.shape

[輸出]: (60000, 28, 28)
```

十種不同的 labels 分別為

```
[程式]: labels = ['t-shirt', 'trouser',
                  'pullover', 'dress',
                  'coat', 'sandal',
                  'shirt', 'sneaker',
                  'bag', 'ankle boots']

[程式]: import matplotlib.pyplot as plt
        %matplotlib inline
        import random
        samples = {}
        for (idx, label) in enumerate(y_train):
            if not label in samples:
                samples[label] = x_train[idx]
            if len(samples) == 10:
                break
        for key in sorted(samples):
            plt.subplot(2, 5, key + 1)
            plt.title(labels[key])
            plt.axis('off')
            plt.imshow(samples[key], cmap='gray')
```



唯一的不同點就是要拿出標籤

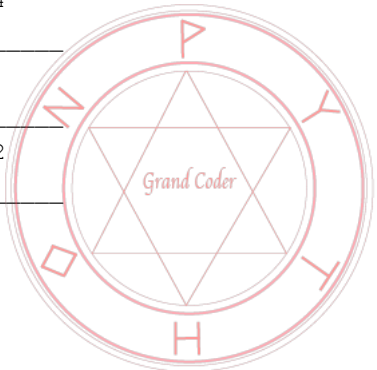
```
[程式]: from keras.utils import np_utils
        # reshape 讓他從 32 * 32 變成 784 * 1 的一維陣列
        # 讓我們標準化到-1~1 區間
        x_train_shaped = (x_train - 127.5)/127.5
        x_test_shaped = (x_test - 127.5)/127.5
        # 不一樣!! 要記得把標籤拿出來
        y_train = y_train.reshape(-1, 1)
```

```
[程式]: img_shape = (28, 28)
        random_dim = 100
```

先來個傳統 GAN 的創作家

```
[程式]: from keras.layers import Input
        from keras.models import Model, Sequential
        from keras.layers.core import Reshape, Dense, Dropout, Flatten
        from keras.layers import Embedding, multiply, BatchNormalization

        generator = Sequential()
        generator.add(Dense(256, input_dim=random_dim,
                            activation='relu'))
        generator.add(BatchNormalization())
        generator.add(Dense(512, activation='relu'))
        generator.add(BatchNormalization())
        generator.add(Dense(784, activation='tanh'))
        generator.add(Reshape(img_shape))
        generator.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 256)               25856
_________________________________________________________________
batch_normalization_3 (Batch (None, 256)               1024
_________________________________________________________________
dense_5 (Dense)              (None, 512)               131584
_________________________________________________________________
batch_normalization_4 (Batch (None, 512)               2048
_________________________________________________________________
dense_6 (Dense)              (None, 784)               402192
_________________________________________________________________
reshape_2 (Reshape)          (None, 28, 28)            0
```



```
==================================================================
Total params: 562,704
Trainable params: 561,168
Non-trainable params: 1,536
__________________________________________________________________
```

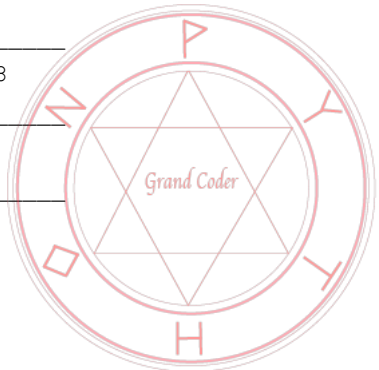
再將模型稍做修改，讓標籤可以傳入，變成我們真正需要的創作家!

```
[程式]: noise = Input(shape=(random_dim,))
        label = Input(shape=(1,), dtype='int32')
        label_embedding = Flatten()(Embedding(input_dim = 10, output_dim = random_dim)(label))

        model_input = multiply([noise, label_embedding])
        img = generator(model_input)

        cgenerator = Model([noise, label], img)

        cgenerator.compile(loss='binary_crossentropy', optimizer="adam")
        cgenerator.summary()
```

先來個傳統的鑑賞家

```
[程式]: discriminator = Sequential()
        discriminator.add(Dense(512, input_dim=784,
                                activation='relu'))
        discriminator.add(Dropout(0.25))
        discriminator.add(Dense(256, activation='relu'))
        discriminator.add(Dropout(0.25))
        discriminator.add(Dense(128, activation='relu'))
        discriminator.add(Dropout(0.25))
        discriminator.add(Dense(1, activation='sigmoid'))
        discriminator.summary()
```

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 512)               401920
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_8 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_9 (Dense)              (None, 128)               32896
```

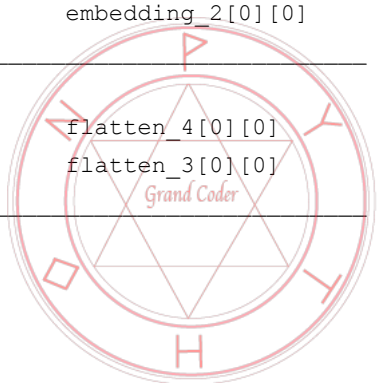


```
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_10 (Dense)             (None, 1)                 129
=================================================================
Total params: 566,273
Trainable params: 566,273
Non-trainable params: 0
_________________________________________________________________
```

加入標籤的輸入讓他成為 CGAN 所使用的鑑賞家

```
[程式]: img = Input(shape=img_shape)
        label = Input(shape=(1,), dtype='int32')
        label_embedding = Flatten()(Embedding(10, 784)(label))
        flat_img = Flatten()(img)
        model_input = multiply([flat_img, label_embedding])
        validity = discriminator(model_input)
        cdiscriminator = Model([img, label], validity)
        cdiscriminator.summary()
```

```
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_4 (InputLayer)            (None, 1)            0
__________________________________________________________________________________________________
input_3 (InputLayer)            (None, 28, 28)       0
__________________________________________________________________________________________________
embedding_2 (Embedding)         (None, 1, 784)       7840        input_4[0][0]
__________________________________________________________________________________________________
flatten_4 (Flatten)             (None, 784)          0           input_3[0][0]
__________________________________________________________________________________________________
flatten_3 (Flatten)             (None, 784)          0           embedding_2[0][0]
__________________________________________________________________________________________________
multiply_2 (Multiply)           (None, 784)          0           flatten_4[0][0]
                                                                 flatten_3[0][0]
__________________________________________________________________________________________________
```

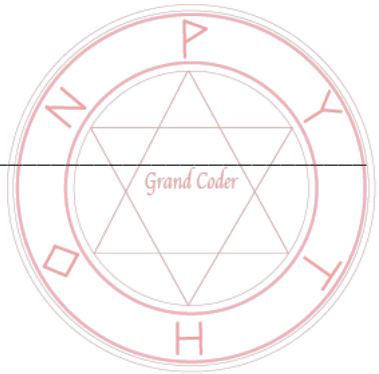


```
sequential_3 (Sequential)       (None, 1)             566273      multiply_2[0][0]
==================================================================================================
============
Total params: 574,113
Trainable params: 574,113
Non-trainable params: 0
__________________________________________________________________________________________________
____________
```

組合網路以訓練創作家

```
[程式]: cdiscriminator.trainable = False
        cgan_input = Input(shape=(random_dim,))
        cgan_label = Input(shape=(1, ))
        x = cgenerator([cgan_input, cgan_label])
        cgan_output = cdiscriminator([x, cgan_label])
        cgan = Model(inputs=[cgan_input, cgan_label], outputs=cgan_output)
        cgan.compile(loss='binary_crossentropy', optimizer="adam")
        cgan.summary()
```

```
__________________________________________________________________________________________________
____________
Layer (type)                    Output Shape          Param #     Connected to
==================================================================================================
============
input_5 (InputLayer)            (None, 100)           0
__________________________________________________________________________________________________
____________
input_6 (InputLayer)            (None, 1)             0
__________________________________________________________________________________________________
____________
model_1 (Model)                 (None, 28, 28)        563704      input_5[0][0]
                                                                  input_6[0][0]
__________________________________________________________________________________________________
____________
model_2 (Model)                 (None, 1)             574113      model_1[1][0]
                                                                  input_6[0][0]
==================================================================================================
============
Total params: 1,137,817
Trainable params: 562,168
Non-trainable params: 575,649
__________________________________________________________________________________________________
____________
```



進行訓練

```
[程式]: batch_size = 200
        epoch_count = 100

        for epoch in range(0, epoch_count):
            for batch_count in range(0, 300):
                idx = np.random.randint(0, x_train.shape[0], batch_size)
                imgs = x_train_shaped[idx]
                labels = y_train[idx]

                valid = np.ones((batch_size, 1))
                fake = np.zeros((batch_size, 1))
                # 步驟 0: 讓創作家製造出 fake image
                noise = np.random.normal(0, 1, (batch_size, random_dim))
                gen_imgs = cgenerator.predict([noise, labels])

                # 步驟 1: 讓鑑賞家鑑賞對的 image
                d_loss_real = cdiscriminator.train_on_batch([imgs, labels], valid)
                # 步驟 2: 讓鑑賞家鑑賞錯的 image
                d_loss_fake = cdiscriminator.train_on_batch([gen_imgs, labels], fake)
                d_loss = (d_loss_real + d_loss_fake) / 2

                noise = np.random.normal(0, 1, (batch_size, random_dim))
                # 步驟 3: 訓練創作家的創作能力
                g_loss = cgan.train_on_batch([noise, labels], valid)

            if (epoch + 1) % 10 == 0:
                dash = "-" * 15
                print(dash, "epoch", epoch + 1, dash)
                print("Discriminator loss:", d_loss)
                print("Generator loss:", g_loss)
```

```
--------------- epoch 10 ---------------
Discriminator loss: 0.4073936343193054
Generator loss: 1.485163
--------------- epoch 20 ---------------
Discriminator loss: 0.45828431844711304
Generator loss: 1.3055634
--------------- epoch 30 ---------------
Discriminator loss: 0.492333322763443
Generator loss: 1.3747383
--------------- epoch 40 ---------------
Discriminator loss: 0.43461155891418457
Generator loss: 1.5247307
--------------- epoch 50 ---------------
```

```
Discriminator loss: 0.4428834915161133
Generator loss: 1.3319385
--------------- epoch 60 ---------------
Discriminator loss: 0.44391652941703796
Generator loss: 1.3759596
--------------- epoch 70 ---------------
Discriminator loss: 0.4795100688934326
Generator loss: 1.403327
--------------- epoch 80 ---------------
Discriminator loss: 0.3963097333908081
Generator loss: 1.4492657
--------------- epoch 90 ---------------
Discriminator loss: 0.46034497022628784
Generator loss: 1.3583496
--------------- epoch 100 ---------------
Discriminator loss: 0.45579004287719727
Generator loss: 1.4524469
```

你可以看到我們開始可以根據不同的標籤產生衣服了!

```
[程式]: import matplotlib.pyplot as plt
        %matplotlib inline
        noise = np.random.normal(0, 1, (10, random_dim))
        sampled_labels = np.arange(0, 10).reshape(-1, 1)

        gen_imgs = cgenerator.predict([noise, sampled_labels])

        # Rescale images 0 - 1
        gen_imgs = 0.5 * gen_imgs + 0.5
        gen_imgs = gen_imgs.reshape(10, 28, 28)
        plt.figure(figsize = (14, 4))

        labels = ['t-shirt', 'trouser',
                  'pullover', 'dress',
                  'coat', 'sandal',
                  'shirt', 'sneaker',
                  'bag', 'ankle boots']

        for i in range(0, 10):
            plt.subplot(1, 10, i + 1)
            plt.axis("off")
            plt.title(labels[i])
            plt.imshow(gen_imgs[i], cmap='gray')
```