AI 無線通訊系統實驗

Lab 3 GAN

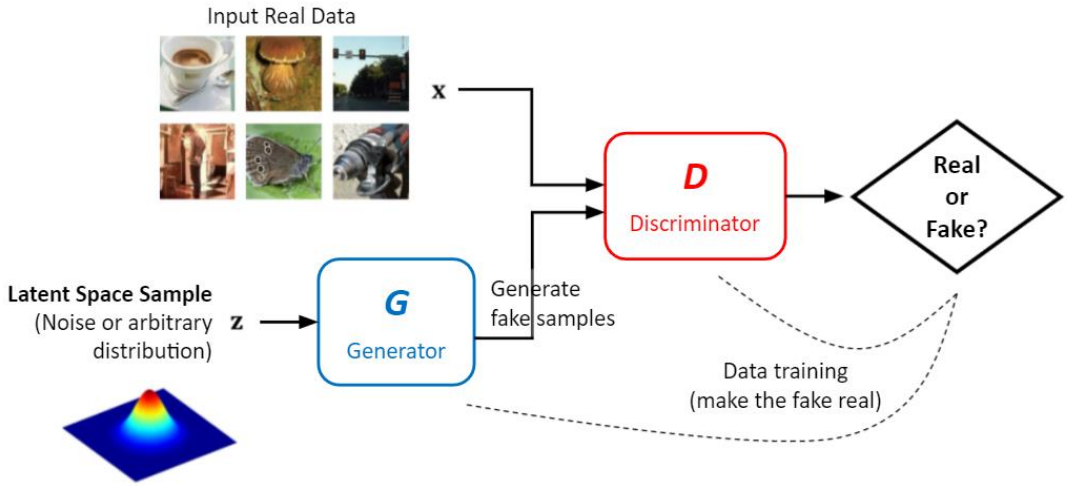
Author: 蕭安紘 助教

實驗目的

建立一個生成對抗網路，利用雜訊生成之資料來提升負責判斷之網路的效能，這次實驗將由 Lab2 所衍生，我們將判斷測試資料是否為同一人在同一個位置上

實驗介紹

How to Build a GAN Model



GAN 中會包含 Generator 與 Discriminator

Generator

利用隨機(可以給定分布)生成之雜訊資料，經過 Generator 產生類似於真實資料(Real data)的假資料(Fake data)，此資料會交由 discriminator 進行判斷，並根據由 discriminator 的 feedback 來優化模型

Discriminator

判斷當下的資料為真實資料或假資料，並根據判斷的結果來優化自身的模型，並給予 generator 優化模型所需的 feedback

Loss Function

Generator loss

在 generator 中，將由 discriminator 判斷出來的機率

```
fake = model.discriminator(fake_data, training=False)
```

作為 loss 的值

且因為要我們的 optimizier 將會設計為尋找 loss function 的最小值，而我們想要讓判斷 fake 的值趨近於 1，代表我們生產的資料越接近真實資料，因此在設計 loss function 時可以藉由加上負號，使其在做 gradient descent 時可以往期望的方向走

我們將利用 `tf.reduce_mean()` 來將 output 機率進行平均，方便 loss 的計算

範例:

```
tf.reduce_mean(fake)
```

Discriminator loss

在 discriminator 中，將由 discriminator 判斷 real data 與 fake data 出來的機率作為 loss

```
fake = model.discriminator(fake_data, training=True)
real = model.discriminator(real_data, training=True)
```

discriminator 的目標為使 fake 的值越低越好，而 real 的值越高越好，所以 loss 的設計思路可以想成 real−fake。同時，如 generator loss 一樣，因為我們的 optimizier 在會去尋找最小值，因此 loss function 也需要加上負號，也將利用 `tf.reduce_mean()` 來將 output 機率進行平均，方便 loss 的計算

Train Step

在每次迭代中，通常會先進行 discriminator 的訓練，再進行 generator 的訓練

有時候會利用一些技巧來幫助 GAN 的訓練，避免無法收斂的狀況發生，如:

- 先預先訓練好一個 discriminator，再開始 GAN 的訓練
- 在每次迭代中，進行多次 discriminator 的訓練後再進行 generator 的訓練

實驗步驟

1. 讀取資料
2. 產生雜訊資料
3. 建立 GAN 網路

   Generator:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 56, 4)]           0
_________________________________________________________________
flatten (Flatten)            (None, 224)               0
_________________________________________________________________
dense (Dense)                (None, 16384)             3686400
_________________________________________________________________
reshape (Reshape)            (None, 4, 4096)           0
_________________________________________________________________
conv1d_transpose (Conv1DTran (None, 8, 4)              81924
_________________________________________________________________
flatten_1 (Flatten)          (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 224)               7392
_________________________________________________________________
reshape_1 (Reshape)          (None, 56, 4)             0
=================================================================
Total params: 3,775,716
Trainable params: 3,775,716
Non-trainable params: 0
```

Discriminator (需要自己實作)

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 56, 4)]           0
_________________________________________________________________
conv1d (Conv1D)              (None, 56, 32)            672
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 56, 64)            10304
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 56, 128)           41088
_________________________________________________________________
conv1d_3 (Conv1D)            (None, 56, 128)           82048
_________________________________________________________________
flatten_2 (Flatten)          (None, 7168)              0
_________________________________________________________________
dense_2 (Dense)              (None, 256)               1835264
_________________________________________________________________
dense_3 (Dense)              (None, 128)               32896
_________________________________________________________________
dense_4 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_6 (Dense)              (None, 4)                 132
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 5
=================================================================
Total params: 2,012,745
Trainable params: 2,012,745
Non-trainable params: 0
```

我們在 discriminator 中加入 constraints，避免其過度擬合。

範例:

```
w_clip = ClipConstrain(clipping_value)
c1 = tf.keras.layers.Conv1D(32, 5, activation=tf.nn.leaky_relu,
kernel_constraint=w_clip, padding='same')(inputs)
```

4. 設定好 loss function 與訓練策略
5. 進行神經網路模型的訓練
    - 這裡與前幾次的訓練方式不一樣，不再使用 fit 涵式，而改用 gradients descent 的方式手動進行優化
6. 印出最後一個 epoch 所產生之模型的 loss

```
epochs: 20
dis_loss: -0.9996980428695679
gen_loss: -0.0002622601459734142
```

7. 畫出 confusion matrix

```
Confusion Matrix:
[[200   0]
 [  0 600]]
```

基礎題

完成以上實驗步驟並找助教 Demo

加分題(先完成基礎題才可以 demo)

1. 比較不同分布產生之雜訊資料，對於訓練之影響
   例如: uniform, exponential, 高斯