Lab 3 結報

姓名：賴昱凱　　學號：111511141

1. 請**詳述 GAN** 的流程與作用

GAN 的全名為 Generative Adversarial Network，生成對抗網路。由名稱中的對抗二字可以知道，這個神經網路是由兩個模型組成，並巧妙的利用互相競爭的方式讓模型表現越來越好。

GAN 是由 Generator 與 Discriminator 構成，利用他們的互相對抗進行訓練，運作流程如下：

Step 1：初始化模型

建立 Generator：負責產生以假亂真的資料。

建立 Discriminator：負責判斷資料的真假。

Step 2：Generator 生成資料

給 Generator 雜訊來生成假資料。

Step 3：訓練 Discriminator

Generator 不做訓練，並利用 noise 與真實資料混合，藉由 Discriminator 做判斷，並依照其判斷結果以及正確答案做 loss 的計算以及參數更新。

其中，Discriminator 判斷的結果會是一個 0 到 1 的數字，代表的就是 Discriminator 認為這個資料是真資料的機率，接近 1 代表 Discriminator 認為為真，接近 0 代表 Discriminator 認為為假。

因此，我們可以利用 Discriminator 的輸出結果利用以下公式計算 loss：（real：真資料結果、fake：假資料結果）

$$loss = -(real - fake)$$

我們希望 real 越高越好、fake 越低越好，因此這個式子剛好可以讓 real = 1、fake = 0 時為最小值（表現最好）；real = 0、fake = 1 時為最大值（表現最差），方便後續做 Gradient Descent。

Step 4：訓練 Generator

Discriminator 不做訓練，利用上一步已經訓練好的 Discriminator（可以判斷真假）來訓練 Generator。我們將 Generator 生成的假資料交給 Discriminator 做判斷，他會輸出一個 0 到 1 之間的數字，接近 0 代表他覺得是假的、接近 1 代表他認為是真的，因此我們可以利用這個輸出值來判斷 Generator 表現的好不好，0 就是不好、1 就是好，因此 loss function 可以藉由該值加上負號構成：（fake：Generator 經由 Discriminator 判斷的結果）

$$loss = -fake$$

讓後續方便做 Gradient descent。

Step 5：重複 Step 2 ~ 4

當訓練達到平衡時：

Generator 生成的資料無法判別真假

Discriminator 無法準確判斷真假

可能出現的問題：

(1) Generator 只會生成一種類型的資料，Discriminator 都會判斷該假資料為真。

解法：Discriminator 要更強，學會判斷該種類型的真假

(2) Discriminator 過擬合，導致太會判斷訓練資料的真假，Generator 怎麼做都是錯的就學不到東西。

解法：加上

w_clip = ClipConstrain(clipping_value)

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

防止 Discriminator 過擬合

為什麼這樣可以防止過擬合？

kernel_constraint 可以在每次參數更新時自動將權重值限制在一定範圍內，讓 Discriminator 不要學的那麼好。

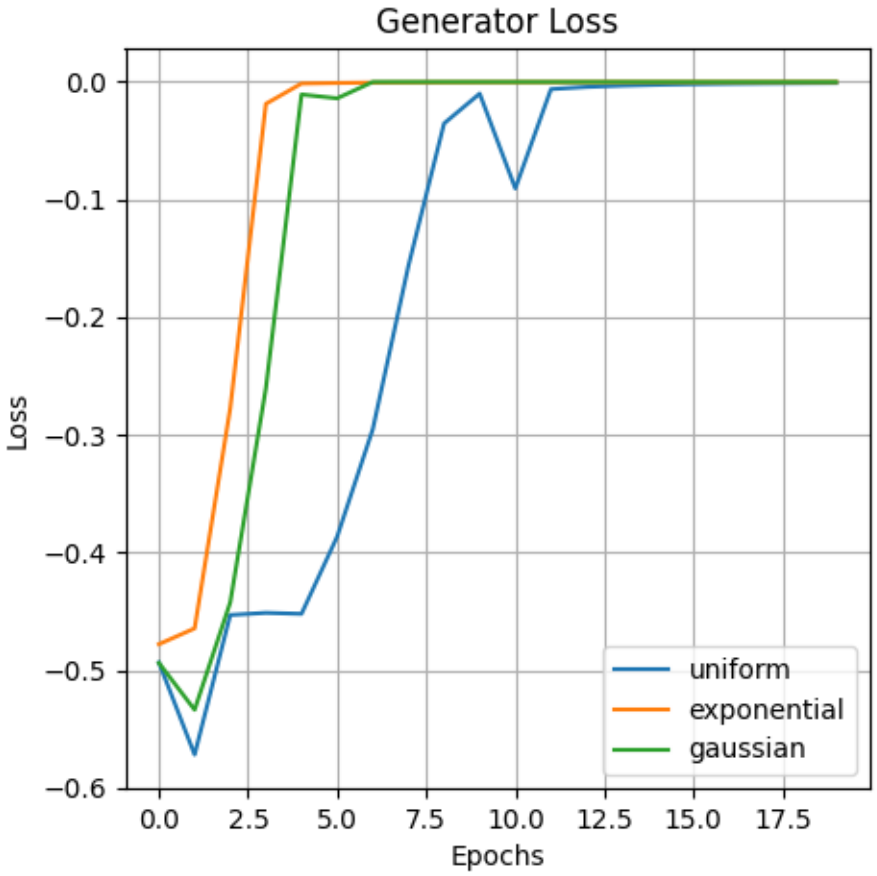
2. 心得

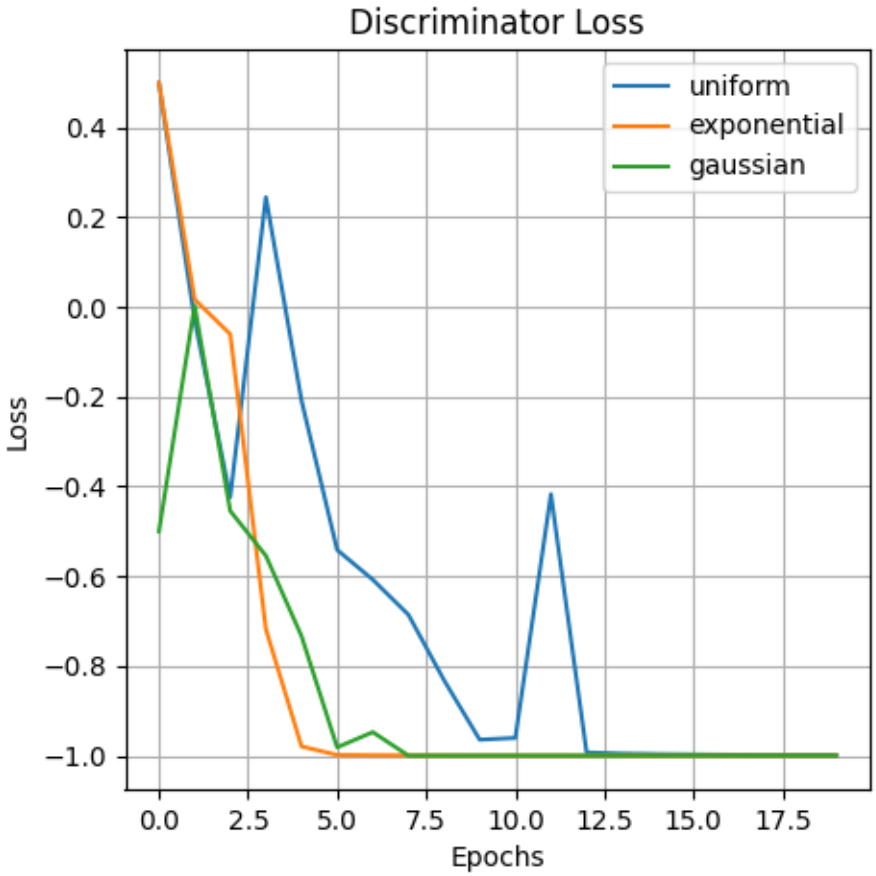
本週的實驗是比較特別的內容，一般其他的課程或是其他 module 都沒有實際接觸到對抗生成網路的實作，我覺得很興奮，有不一樣的學習內容，也讓我更熟悉對抗生成網路內部的運作原理。

實驗中有觀察到一個比較特別的問題，每次訓練都有點靠運氣，常常會訓練不起來，我推測是因為雜訊是利用隨機生成的方式，若雜訊太大可能造成 Generator 無法有效生成以假亂真的資料，我後來將雜訊的範圍設在大約-0.5 到 +0.5 之間，就比較穩定可以有很不錯的訓練結果。

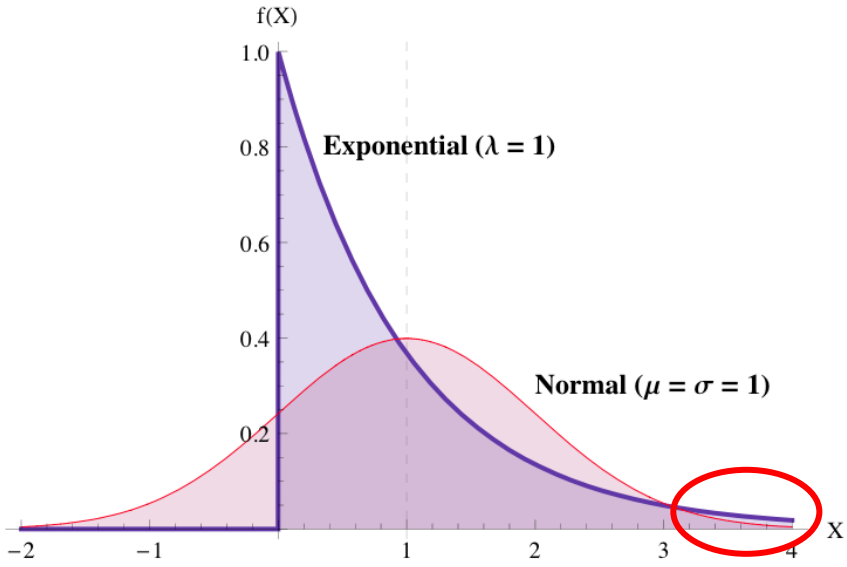
本週的加分題是探討不同雜訊下，對抗生成網路訓練過程的差異，我做出的結果以及相關參數如下圖：

```
if noise_type == 'uniform':
    noise[i,j,k] = random.uniform(-0.5, 0.5)
elif noise_type == 'exponential':
    noise[i,j,k] = np.random.exponential(0.2)
elif noise_type == 'gaussian':
    noise[i,j,k] = np.random.normal(0, 0.5)
```





可以發現無論在 Generator 還是 Discriminator 的 loss 都是 exponential 收斂的最快，Gaussian 次之，uniform 最慢。但是我也發現在訓練的穩定度上，反而是相反的關係：uniform 最佳、Gaussian 次之而 exponential 最差，這應該是因為不同雜訊的數據性質不同。



極端值 Exponential 多於 Normal

Uniform 最穩定，是因為各個資料點的雜訊不會有很明顯的差距，就像是一張白紙很容易就可以生成出很好的資料。但 Exponential 大部分的數據都集中在靠近 0 的位置，並且會有極端值的出現，就像是一張白紙上有一些的隨機黑點，導致訓練不穩定。Gaussian 也是類似的道理，但他因為是常態分佈，極端值出現的又沒有 Exponential 那麼頻繁，因此穩定性位於中間。