1. Introduction

在本次實驗報告中會講述如何設計DDPM的model，並且如何加入condition的label到DDPM中，如何使用noise schedule和如何做training的部分，也會稍微帶到DDPM的原理和最後inference的結果，在實驗數據中也有使用不同的embedding dimension做測試，去比較不同class embedding dimension中的關係。

2. Implementation Details

**A. Describe how you implement your model, including your choice of DDPM, noise schedule.**

在實作上都採用Diffusers所提供的所有套件，包括UNet2DModel、noise schedule function、optimizer scheduler (get\_cosine\_schedule\_with\_warnmup) ，在這段中會介紹如何設計一個Condition DDPM，還有他的基底ClassConditionedUnet。

先來介紹ClassConditionedUnet的實作方式，下condition的方式都跟diffusers教學內容一致，會先將要下的class condition給轉成一個embedding domain，在進入UNet之前會先將24個one-hot encoding的label(classes)轉成embedding size 大小的embedding domain，然後才將這個embedding的資料連接到UNet去做訓練。

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

在第15行會先將輸入的class label轉成class\_emb\_size大小的embedding domain，第18行道41行則是定義整個網路架構，有嘗試過很多種設計方式，最後發現直接以6層的UNet效果最好，在6層的dowmsample後的大小會是1\*1\*512，因為受到輸入圖片大小(sample\_size)的限制，因此設定6層為上限，而初始的就以128為初始，為了更好的學習深層的feature和condition之間的關係，會針對深層的網路增加注意力機制(AttnDownBlock2D)，且沒有快速的加倍channel數量，直接以128、128、256、256、512、512為最終的網路模型，其中可以看到下Condition的方式是直接以UNet的輸入加上class\_emb\_size的數量，因此在後續的實驗中會依照不同的class\_emb\_size做探討，如果挑選過大的class\_emb\_size會不會造成訓練不起來等情況。

在forward的設計上，輸入的部分會有原圖x、noise 的timestep時間點 t、24個one-hot class的label (class\_labels)，會先將label轉成embedding domain，然後再跟原圖合併(將condition的資訊加在原圖x上)，然後再給專門為Diffusion設計的UNet的模型(UNet2DModel)當作輸入，最後再從這個給定的timestep t去sample最後的結果。

在train ddpm時設計了一個TrainDDPM的class，在建構子時會先定義上述所設計的model (ClassConditionedUNet) 還有training的dataset和validation的dataset，並且在dataset的設計上可以去調整一次training的partial和val的比例(後續都改成 partial=1 與 val\_split = 0，也就是使用權部的資料及，不分割測試資料集)，圖片再經過dataset時會先做一次的transform，如第二張圖的217行，會先將圖片轉成64\*64的大小，然後轉成PyTorch的Tensor，圖的像素就會從[0,255]轉道[0,1]的範圍，並且對圖片進行標準化，針對每個RGB channel進行標準化，使平均值為0.5，標準差為0.5，這樣能將圖片的像素從[0,1]的範圍轉換為[-1,1]，利於網路中的訓練。

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

Training所使用的Optimizer為AdamW，learning rate的Scheduler則是使用Diffuser所提供的get\_cosine\_schedule\_with\_warmup，noise\_scheduler也是使用Diffuser所提供的DDPMScheduler，在DDPM論文中，在每一個timestep中都會加入少量的高斯噪音noise，而Diffuser所提供的套件可以透過timestep的步長去設定噪音量，如下方公式所表示的：



給定某個timestep 可以得到下一個加噪音的版本 ，從公式中可以看出將按照做縮放，並加入按的噪音，也就是說是根據特定的timestep來定義的，並且決定每個timestep中添加的noise數量，但是想要直接從原圖去預測在t的時間點的噪音，假設今天要求，為了避免直接執行500次，可以使用另一個公式



其中且由下方的圖中可以看出藍色曲線表示的是原始圖片在每一個timestep中保留的部分，隨著timestep越大(t越大)，會越來越小，相反的則是噪音的占比，最終會形成全部都是噪音的圖。

一張含有 文字, 圖表, 行, 繪圖 的圖片

自動產生的描述

一張含有 文字, 螢幕擷取畫面, 軟體, 多媒體軟體 的圖片

自動產生的描述

在training\_one\_epoch的程式中會隨著noise\_scheduler和挑選到的timestep去對原圖片增加噪音，然後給model去做訓練，使用MSE針對噪音去求loss是為了能更精確的去學習如何去除噪音。

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

自動產生的描述

最後在valid時候直接以完全noise的圖(第122行)進行timestep次數的sample，最後得到一張完整的圖，但是前面因為有做過transform，因此還要將圖片轉換回原本的[0,1](第137行)。

3. Results and discussion

**A. Show your synthetic image grids and a denoising process image**

|  |  |
| --- | --- |
| Timestep = 0 |  |
| Timestep = 100 |  |
| Timestep = 200 |  |
| Timestep = 300 |  |
| Timestep = 400 |  |
| Timestep = 500 |  |
| Timestep = 600 |  |
| Timestep = 700 |  |
| Timestep = 800 |  |
| Timestep = 900 |  |
| Timestep = 1000 |  |

一張含有 文字, 螢幕擷取畫面, 字型, 黑與白 的圖片

自動產生的描述

此圖示以test.json作為test的目標，最後上傳的model以下方Experiment Results為準。

從timestep的過程中可以看到大概從t=500時就可以看出物品的輪廓了，接著到t=1000時最為明顯。

**B. Discussion of your extra implementations or experiments**

在介紹class\_emb\_size時候有提到，本實驗使用了多個不同的數據，有128、256、512、1024等，由於時間限制都訓練在300個epoch上下，由此圖可以很明顯地看的出來彼此之間的差異。

一張含有 文字, 行, 圖表, 繪圖 的圖片

自動產生的描述

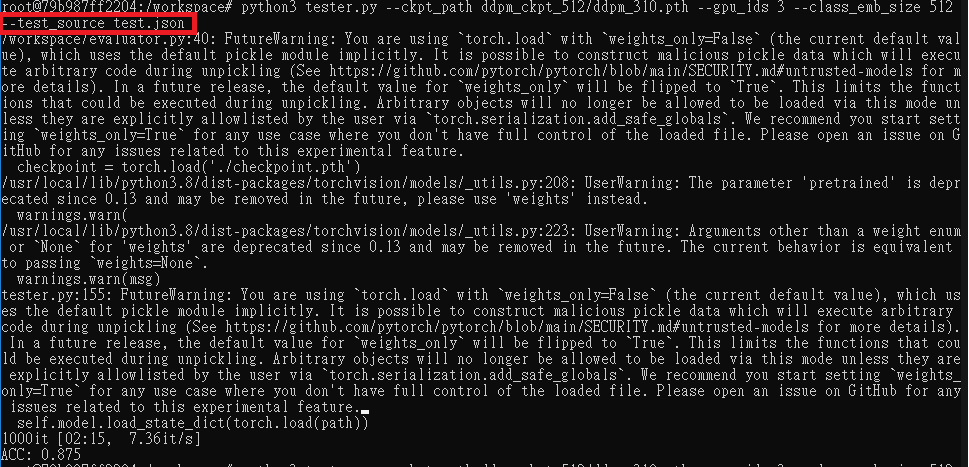
每個測試的種子碼都是固定的(seed=48763)，會發現隨著epoch的增長accuracy會越來越高，但是還是沒有辦法很穩定的去產生圖片，此圖也是依據(test.json)去做比較的圖，只取100至300個epoch之間的accuracy差異。

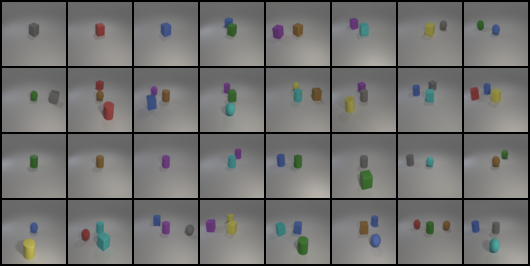
4. Experiment Results

**A. Classification accuracy on test.json and new test.json**

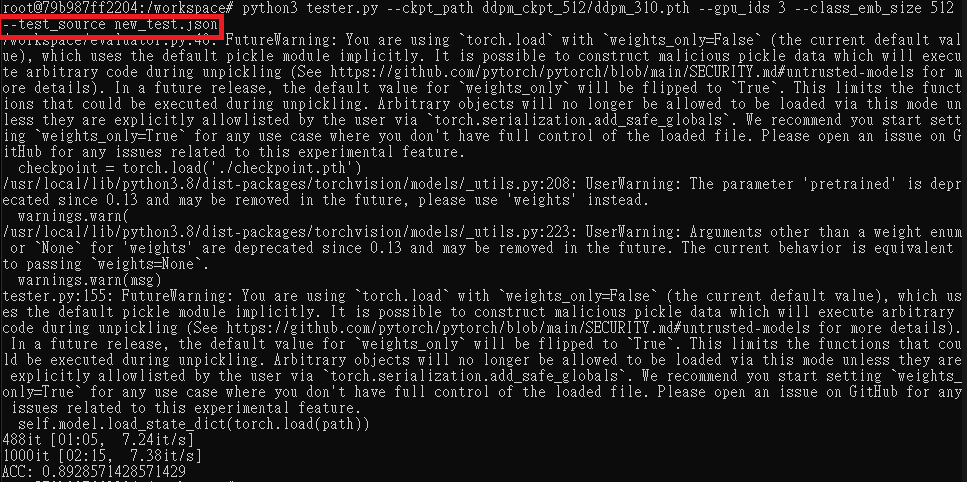
**Show your accuracy screenshots**

Test.json (0.875 (87.5%))





New\_test.json (0.892 (89.2%))





**The command for inference process for both testing data**

在inference(testing)時可以使用此指令：

|  |
| --- |
| python3 tester.py --gpu\_ids 0 --ckpt\_path {pth\_model\_path} --class\_emb\_size 512 --test\_source {path\_to\_test.json|path\_to\_new\_test.json} |

對其做詳細的講解為

* --gpu\_ids 設定gpu的ID
* --ckpt\_path 設定要載入的 checkpoint.pth位置
* --class\_emb\_size 設定對應大小 (因為上傳的model是512因此固定512)
* --test\_source 看是要選擇使用 test.json 或者 new\_test.json 資料

當然得在下載之前可能需要下載所需套件，於程式碼中有提供requirements.txt (如果因為套件原因無法執行的話)

5. Reference

https://github.com/huggingface/diffusers