R12521504 吳玉文

Problem 1

1. *Describe your implementation details and the difficulties you encountered.*

按照 Pseudo code 實現 training 的過程：

```python
def forward(self, x, c):
    """
    this method is used in training, so samples t and noise randomly
    """
    # t ~ Uniform(0, n_T)
    _ts = torch.randint(1, self.n_T+1, (x.shape[0],)).to(self.device)
    # eps ~ N(0, 1)
    noise = torch.randn_like(x)
    # This is the x_t, which is sqrt(alphabar) x_0 + sqrt(1-alphabar) * eps
    # We should predict the "error term" from this x_t. Loss is what we return.
    x_t = (
        self.sqrtab[_ts, None, None, None] * x
        + self.sqrtmab[_ts, None, None, None] * noise
    )
    # dropout context with some probability
    context_mask = torch.bernoulli(torch.zeros_like(c)+self.drop_prob).to(self.device)
    # return MSE between added noise, and our predicted noise
    return self.loss_mse(noise, self.nn_model(x_t, c, _ts / self.n_T, context_mask))
```

按照 Pseudo code 實現 Sampling 的過程：

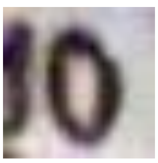
```python
# The reverse generation (denoising) process
for i in range(self.n_T, 0, -1):
    print(f'sampling timestep {i}',end='\r')
    t_is = torch.tensor([i / self.n_T]).to(device)
    t_is = t_is.repeat(n_sample,1,1,1)
    # double batch
    x_i = x_i.repeat(2,1,1,1)
    t_is = t_is.repeat(2,1,1,1)

    z = torch.randn(n_sample, *size).to(device) if i > 1 else 0

    # split predictions and compute weighting
    eps = self.nn_model(x_i, c_i, t_is, context_mask)
    eps1 = eps[:n_sample] # with condition
    eps2 = eps[n_sample:] # without condition
    eps = (1+guide_w)*eps1 - guide_w*eps2
    x_i = x_i[:n_sample]
    x_i = (
        self.oneover_sqrta[i] * (x_i - eps * self.mab_over_sqrtmab[i])
        + self.sqrt_beta_t[i] * z
    )
    if i%80==0 or i==1:
        x_i_store.append(x_i.detach().cpu().numpy())

x_i_store = np.array(x_i_store)
return x_i, x_i_store
```

因為有找到可以參考的程式碼，所以在 implementation 上沒有遇到什麼大困難。反而是因為運算資源不太夠所以 train 很久(最後還是屈服了，買了 Colab Pro)。

2. *Please show 10 generated images for each digit (0-9) from both MNIST-M & SVHN dataset in your report. You can put all 100 outputs in one image with columns indicating different noise inputs and rows indicating different digits. [Visualize BOTH MNIST-M & SVHN]*

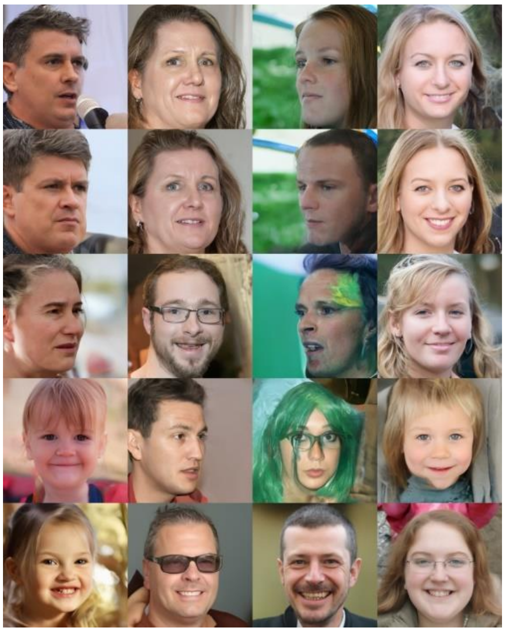
MNIST-M	SVHN
	

3. *Visualize a total of six images from both MNIST-M & SVHN datasets in the reverse process of the first "0" in your outputs in (2) and with different time steps. [Visualize BOTH MNIST-M & SVHN]*

MNIST-M					
t = 0	t = 80	t = 160	t = 240	t = 320	t = 400
					
SVHN					
t = 300	t = 320	t = 340	t = 360	t = 380	t = 400
					

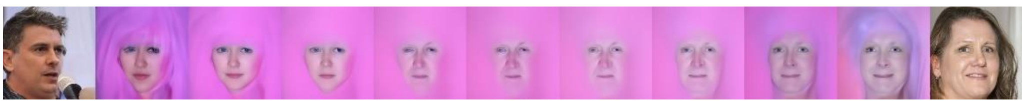
Problem 2

1. *Please generate face images of noise 00.pt ~ 03.pt with different eta in one grid. Report and explain your observation in this experiment.*

<table>
<tr><td>eta = 0.00</td><td rowspan="5"></td></tr>
<tr><td>eta = 0.25</td></tr>
<tr><td>eta = 0.50</td></tr>
<tr><td>eta = 0.75</td></tr>
<tr><td>eta = 1.00</td></tr>
</table>

當 eta 越大的時候，加入的隨機 noise 就會對原本的 noise 影響越大。造成結果與 eta=0 的時候相差越遠，而每次 eta=0 所產生的的照片，因為沒有隨機性的關係，結果都會一樣。

2. *Please generate the face images of the interpolation of noise 00.pt ~ 01.pt. The interpolation formula is spherical linear interpolation, which is also known as slerp. What will happen if we simply use linear interpolation? Explain and report your observation.*

<table>
<tr><td>slerp</td></tr>
<tr><td></td></tr>
<tr><td>linear</td></tr>
<tr><td></td></tr>
</table>

比起使用 slerp，linear 因為插值的過程中並沒有經過平滑，導致結果看起來會產生許多不自然的混和。而 slerp 則很明顯的正常許多。

Problem 3

1. *Conduct the CLIP-based zero shot classification on the hw2_data/clip_zeroshot/val, explain how CLIP do this, report the accuracy and 5 successful/failed cases.*

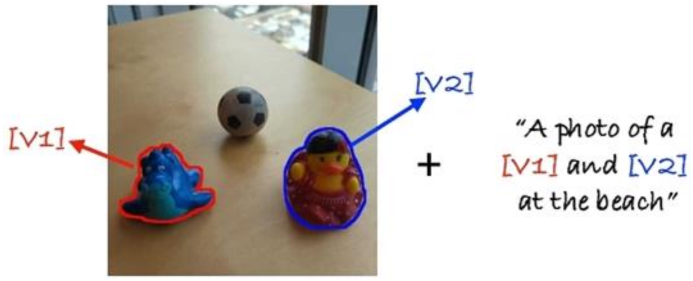
CLIP 如何做到 CLIP-based zero shot classification：

要對新的圖像進行分類時，CLIP 不需要針對目標類別重新訓練。而是可以直接為每個目標類別撰寫簡單的描述，像是「a photo of a cat」、「a photo of a dog」等等。然後，CLIP 將這些描述和待分類的圖像一起輸入模型，將它們都轉換成向量，並計算圖像向量和每個描述向量的相似度。模型最終會選擇相似度最高的描述，作為圖像的預測類別。

2. *What will happen if you simply generate an image containing multiple concepts (e.g., a <new1> next to a <new2>)? You can use your own objects or the provided cat images in the dataset. Share your findings and survey a related paper that works on multiple concepts personalization, and share their method.*

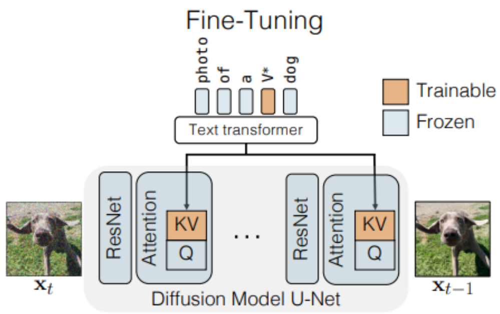
"Photo of S_{clock} in the style of S_{style}" "Photo of S_{cat} in the style of S_{style}" "Photo of S_{craft} in the style of S_{style}" "Photo of S_{clock} in the style of S_{cat}" "Photo of S_{clock} in the style of S_{craft}" "Photo of S_{cat} in the style of S_{craft}"

原 paper 有提到多個 concepts 分開放置的話是 OK 的，但如果 side-by-side 的放置的則不行。我有找到其他 paper 有呈現可能會產生一些奇怪的融合的情形，像下圖所展示的這樣。

<table>
<tr><td colspan="2">Input 的圖片及 prompt</td></tr>
<tr><td colspan="2">
</td></tr>
<tr><td>產生奇怪的融合</td><td>符合預期的情況</td></tr>
<tr><td></td><td></td></tr>
</table>

Related paper that works on multiple concepts personalization：

我找到的是 Multi-Concept Customization of Text-to-Image Diffusion 這篇，這篇只對目標文本以及 Diffusion Model U-Net 的 cross-attention 層的 key 和 value 進行權重更新（如下圖所示）。



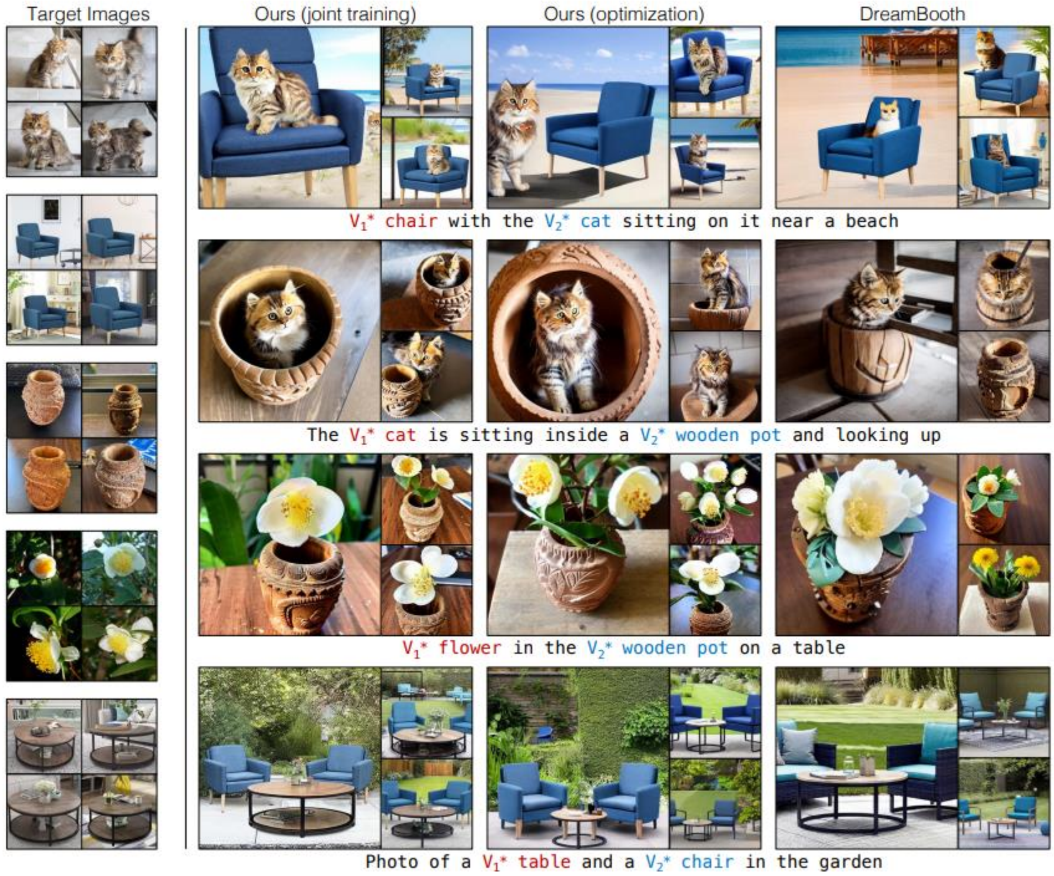
作法是將每一個 concept 分開訓練後再將各個概念的 cross-attention 投影矩陣（key 和 value）進行合併，合併的方式是使用約束最小平方問題（如下列式子，N是 concept 的數量，c 是攤平後的文本特徵）來讓不同的 concept 可以在互相影響最小的情況下合併，確保合併後仍能保有特徵。

$$\hat{W} = \arg\min_{W} ||WC_{\text{reg}}^{\top} - W_0 C_{\text{reg}}^{\top}||_F$$

$$\text{s.t.}\ \ WC^{\top} = V,\ \text{where}\ C = [\mathbf{c}_1 \cdots \mathbf{c}_N]^{\top}$$

$$\text{and}\ V = [W_1 \mathbf{c}_1^{\top} \cdots W_N \mathbf{c}_N^{\top}]^{\top}.$$

最後呈現的結果如下：



Reference

Problem 1：https://github.com/TeaPearce/Conditional_Diffusion_MNIST

Problem 2：

https://github.com/xiaohu2015/nngen/blob/main/models/diffusion_models/ddim_mnist.ipynb