深度學習 作業四

學號：312831002　姓名：廖健棚

VAE

Sample 數個 latent

```python
def generate_samples(model, num_samples):
    model.eval()
    with torch.no_grad():
        z = torch.randn(num_samples, latent_dim)
        samples = model.decode(z)
    return samples.reshape(-1, 28, 28), z
```

Decoder 生成圖片

```python
def decode_latents(model, zs):
    images = []
    model.eval()
    with torch.no_grad():
        for z in zs:
            images.append(model.decode(z))
    return images
```

繪製 8*8 張 latent grid

```python
def interpolate_four_corners(corners, width_count, height_count):
    top_left, top_right, bottom_left, bottom_right = corners
    sections = []
    for i in range(height_count):
        for j in range(width_count):
            alpha = i / height_count
            beta = j / width_count

            top = alpha * top_left + (1 - alpha) * top_right
            bottom = alpha * bottom_left + (1 - alpha) * bottom_right

            interpolated_vector = beta * top + (1 - beta) * bottom
            sections.append(interpolated_vector)
    return sections
```

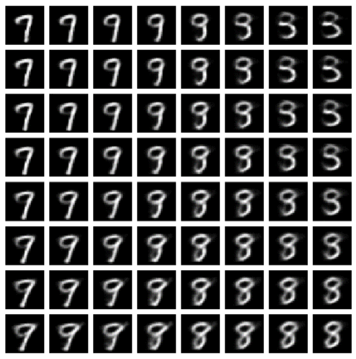
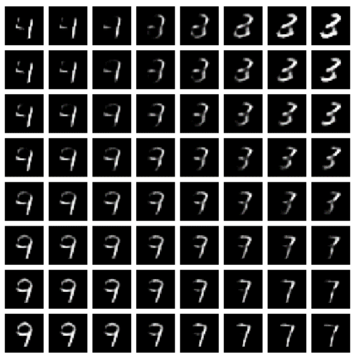
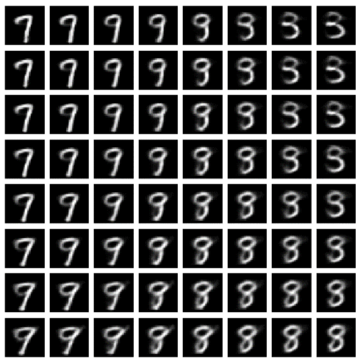
Latent interpolation: 先做每列左右的插值，再做一次上下查值就可以得到這些分布

VAE Parameter 調整

Latent: 20, 60

image transform 有做 normalization

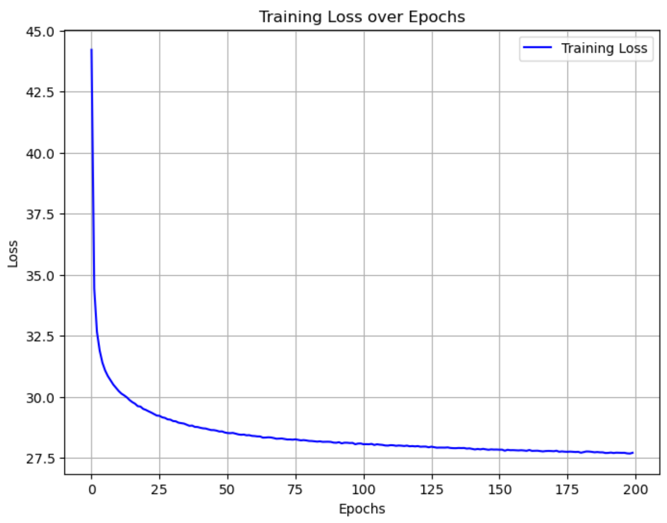
```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])
```

Change	Latent:20	Normalized	Latent:60
Graph			

討論:

同樣是 latent 20 的條件下，且都訓練 200 epochs，有經過 normalization 的訓練圖片所生成的圖片會較不完整，但圖片比較銳利

Loss:訓練 VAE 的 loss 都有穩定下降



VAE loss

GAN

Conditional GAN(CGAN)

超參數設定:

```
BATCH_SIZE = 256
Z_DIM = 100
LABEL_EMBED_SIZE = 10
NUM_CLASSES = 10
IMGS_TO_DISPLAY_PER_CLASS = 10
CHANNELS = 3
EPOCHS = 200
```

優化器:

```
# Define Optimizers
g_opt = optim.Adam(gen.parameters(), lr=0.0002, betas=(0.5, 0.999), weight_decay=2e-5)
d_opt = optim.Adam(dis.parameters(), lr=0.0002, betas=(0.5, 0.999), weight_decay=2e-5)
```

CGAN 設定: 在 Generator 及 Discriminator 部分必須嵌入 label 的 Embedding

```
def forward(self, x, label):
        x = x.reshape([x.shape[0], -1, 1, 1])
        label_embed = self.label_embedding(label)
        label_embed = label_embed.reshape([label_embed.shape[0], -1, 1, 1])
        x = torch.cat((x, label_embed), dim=1)
```

訓練 GAN 的 loss 設定

```
# Train Discriminator
fake_out = dis(x_fake.detach(), x_label)
real_out = dis(x_real.detach(), x_label)
d_loss = (loss_fn(fake_out, fake_label) + loss_fn(real_out, real_label)) / 2
```

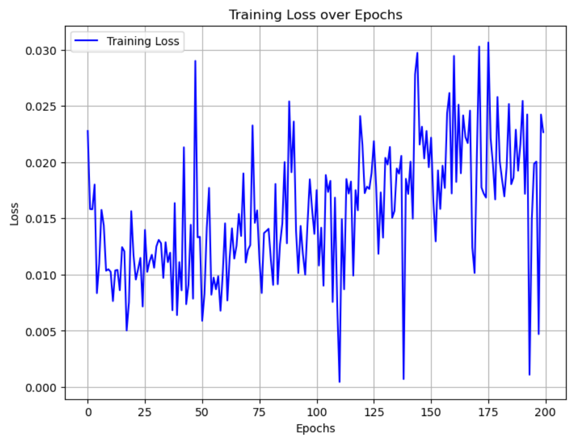
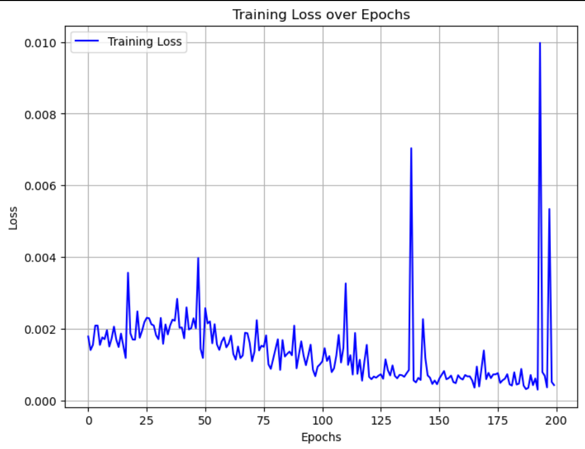
Discriminator 的 loss 是將真的圖片與生成圖片的 loss 相加除以 2

主要目的是讓 Discriminator 要能夠分辨出真實圖片與生成圖片

```
# Train Generator
fake_out = dis(x_fake, x_label)
g_loss = loss_fn(fake_out, real_label)
```

Generator 的 loss 是在經過 Discriminator 後是否可以騙過其生成的圖片被辨識為真，所以生成越真實，loss 就會越低

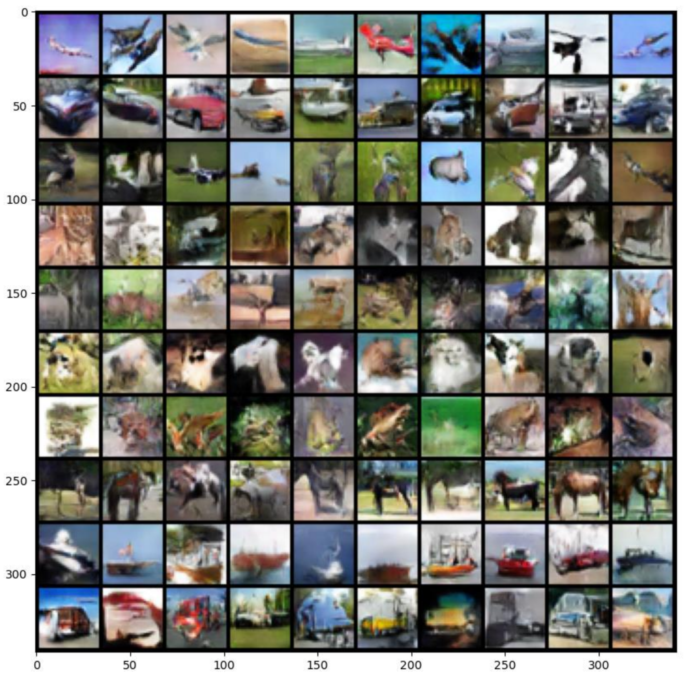
訓練時 Generator & Discriminator Loss 變化

Generator loss	Discriminator loss
	

Generator loss 在訓練過程中，loss 有漸漸上升，但不會上升太多

Discriminator loss 在訓練過程中，loss 有漸漸下降，但還是會有突然的高峰

以整體訓練下來，loss 的變化沒有過於起伏，這會是必較好的訓練過程。



CGAN 生成 10x10 的圖片，每一列分別為類別，每列共十個 samples

以 CGAN 訓練結果可以控制所生成的類別，有效的生成可分辨的圖片，所生成出來的效果也會相較於 un-condtional 的訓練來的清楚。

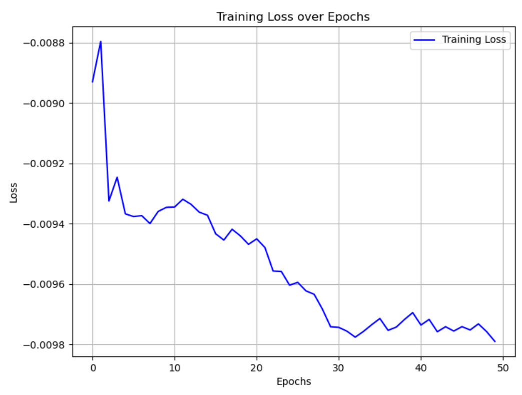
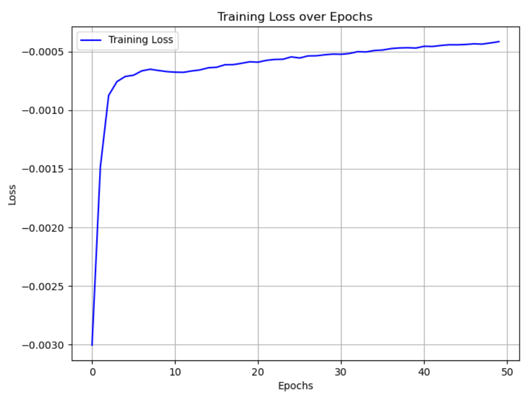
WGAM-CP

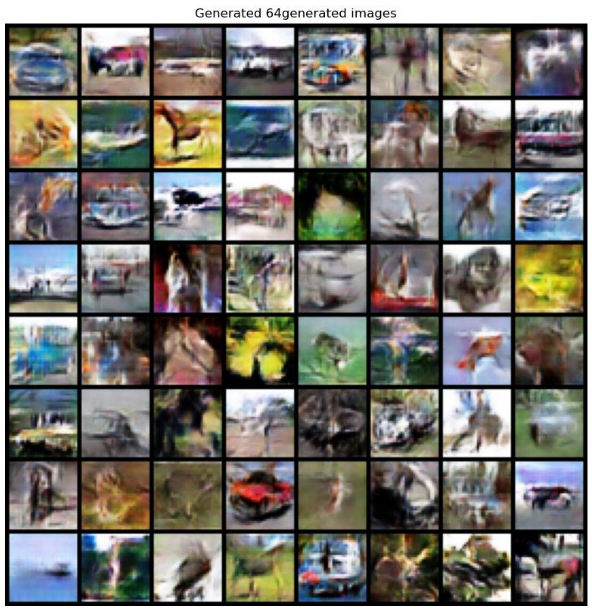
WGAN 引入了一種新的損失函數，稱為 Wasserstein 距離或 Earth-Mover（EM）距離，來衡量生成器和真實數據分佈之間的差異。並不使用 Sigmoid 層及 BCELoss

```
# Measure discriminator's ability to classify real from generated samples
real_pred = discriminator(real_imgs)
fake_pred = discriminator(fake_imgs.detach())
real_loss = real_pred.mean()
fake_loss = fake_pred.mean()
d_loss = -real_loss + fake_loss
```

loss 為正或者負均可，表示的真數據和假數據的相對分佈位置，收斂到 0 為目標

訓練時 Generator & Discriminator Loss 變化

Generator loss	Discriminator loss
	



WGAN 生成 8x8 隨機圖片

以上我的訓練成果，可以看到所生成的圖片有明顯油畫感，且不能夠輕易辨識其類別，比較像是混和體，相對於 CGAN 鎖定類別生成來說，感覺上有明顯差距。

參考資料

i. S-Chh. (n.d.). S-CHH/pytorch-cgan-conditional-gan: Pytorch implementation of conditional generative Adversarial Network (cgan) using DCGAN architecture for generating 32x32 images of mnist, SVHN, FashionMNIST, and USPS datasets. GitHub. https://github.com/s-chh/Pytorch-cGAN-conditional-GAN/tree/main

ii. Eriklindernoren. (n.d.). Eriklindernoren/Pytorch-Gan: Pytorch implementations of generative adversarial networks. GitHub. https://github.com/eriklindernoren/PyTorch-GAN/tree/master

iii. OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model]. https://chat.openai.com/chat