Homework 2 : Report

110024516 統研所 邱繼賢

1 Introduction

1.1 Abstract

本次作業使用 eye dataset，建構 (variational) autoencoder model，將 1476 張眼睛的圖片透過模型盡可能還原 regenerate 成新的圖片，這是一種 unsupervised learning，並且期望新舊圖片集的 PSNR & SSIM score 分別可以達到 22 & 0.64。

1.2 Data

在 eye dataset 中，總共有 1476 張圖片，每一張的維度是 (50, 50, 3)，還有包含一個變數 label : **female eyes (0)** and **male eyes (1)**，但此次作業目的僅是對圖片進行還原的 unsupervised learning 並不需要對 label 進行預測。

圖片中的每個數值 $x \in [0,1]$ 皆以除過 255，以下在建構模型前需再將數值轉換：$x \leftarrow 2x-1$ 映射到 $[-1,1]$ 區間。

2 Model

以下模型的 encoder and decoder 皆採用 convolutional neural network 的方式建構，故不須將原始 (50, 50, 3) 的 tensor data 轉換成 (7500, 1) 的 vector data。

2.1 Autoencoder

2.1.1 Architecture

模型架構如 Figure 1 所示，encoder 和 decoder 都是使用三層的 CNN 架構，因爲 output 數值都要落在 $[-1,1]$ 區間，所以 decoder output layer 的 activation function 採用 tanh function，而其餘 layers 皆採用 ReLU function。

在訓練 autoencoder model 時，loss function 是使用 original images 和 reconstructed images 之間的 MSE，optimization 的方式則是使用 Adam，Figure 2 表示的是此模型在 500 epochs 下的 training loss，可以看出差不多在 300 epoch 之後 loss 大致呈現穩定，代表模型大致上已經收斂。

```
conv_autoencoder(
  (encoder): Sequential(
    (0): Conv2d(3, 12, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(12, 24, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): Conv2d(24, 48, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): ReLU()
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(48, 24, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): ConvTranspose2d(24, 12, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): ConvTranspose2d(12, 3, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (5): Tanh()
  )
)
```

Figure 1: Architecture of AE model

2.1.2 Hyperparameters

Epoch, learning rate, batch size 等參數主要是影響 gradient loss 收斂的程度及穩定性，可藉由觀察 epochs 對 training loss 的曲線來做調整，其對模型整體的表現影響沒那麼直接，以下主要是探討在 CNN 架構中 kernel size, stride, zero padding 三個參數對我們模型表現的影響，分別用 PSNR, SSIM, MSE loss 三個數值來做衡量，實驗結果如 Table 1：

(kernel,stride,padding)	**(3,1,1)**	**(5,1,2)**	**(7,1,3)**	**(9,1,4)**	**(4,2,2)**	**(6,2,3)**
PSNR	42.53	40.56	33.8	32.4	30.26	30.19
SSIM	0.996	0.995	0.981	0.978	0.916	0.913
MSE Loss	0.0002	0.0003	0.0016	0.0020	0.0038	0.0041

Table 1: Experiment of AE model

隨著 kernel size 變大，模型在 PSNR, SSIM, MSE loss 的表現持續變差，而 stride 從 1 增加到 2，模型的表現也有所下降，故最後選擇的各項 hyperparameters 如 Table 2：

2.1.3 Performance

以 **2.1.1** 中建構的模型架構和 **2.1.2** 中設定的各項 hyperparameters 所訓練出的 best AE model 的表現如 Table 3 (可藉由執行 `performance.py` 驗證)：

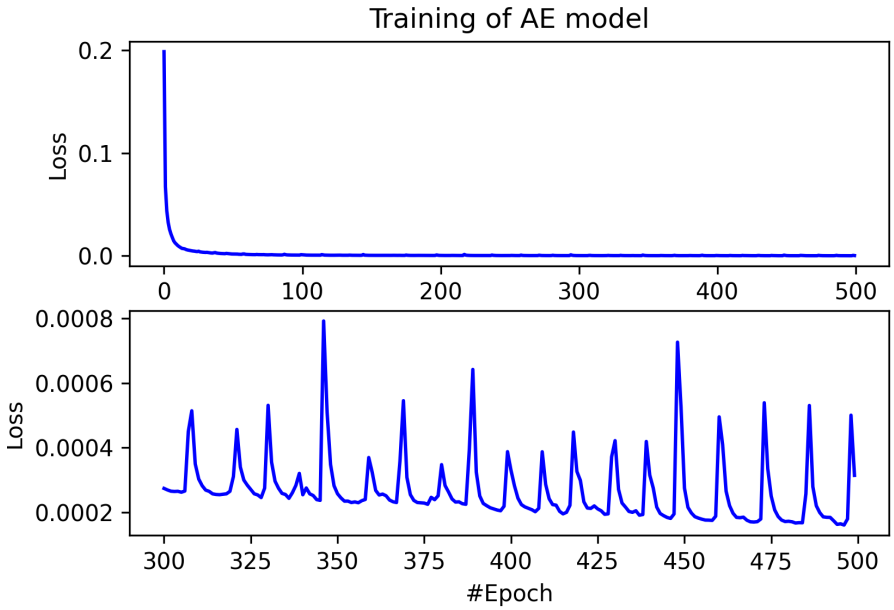

Figure 2: Loss of best AE model

number of epochs	500
batch size	164
learning rate	10^{-3}
kernel size	3
stride	1
zero padding	1

Table 2: Hyperparameters of AE model

PSNR	42.53
SSIM	0.996

Table 3: Performance of best AE model

2.2 Variational Autoencoder

2.2.1 Architecture

模型架構如 Figure 3 所示，encoder 爲兩層 CNN hidden layers 再加上兩個一層 CNN output layer，分別代表 μ 和 $\log(\sigma^2)$，再藉此以 Gaussian distribution 建構出 latent tensor，然後送進三層 CNN 架構的 decoder，一樣是只有 decoder output layer 的 activation function 採用 tanh function，其餘 layers 皆採用 ReLU function。

而在訓練 variational autoencoder model 時，因爲中間多了一個 Gaussian distribution，所以 loss function 除了 MSE 之外還要加上 KL Divergence，optimization 的方式一樣使用 Adam，Figure 4 表示的是此模型在 500 epochs 下的 training loss，可

以看出雖然在 300 epoch 後 loss 的趨勢還有在些微下降，代表模型還沒到收斂，但此時 loss 的變化已經很小了，我們可以提早停止訓練。

```
VAE(
  (encoder): Sequential(
    (0): Conv2d(3, 12, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
    (2): Conv2d(12, 24, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (3): ReLU()
    (4): Conv2d(24, 48, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): ReLU()
  )
  (enc_out_1): Sequential(
    (0): Conv2d(48, 96, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
  )
  (enc_out_2): Sequential(
    (0): Conv2d(48, 96, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
  )
  (decoder): Sequential(
    (0): ConvTranspose2d(96, 48, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): ReLU()
    (2): ConvTranspose2d(48, 24, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (3): ReLU()
    (4): ConvTranspose2d(24, 12, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (5): ReLU()
    (6): ConvTranspose2d(12, 3, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (7): Tanh()
  )
)
```

Figure 3: Architecture of VAE model

2.2.2 Hyperparameters

一樣主要是探討在 CNN 架構中 kernel size, stride, zero padding 三個參數對我們模型表現的影響，實驗結果如 Table 4：

(kernel,stride,padding)	**(3,1,1)**	**(5,1,2)**	**(7,1,3)**	**(9,1,4)**	**(4,2,2)**	**(6,2,3)**
PSNR	26.59	26.85	26.43	24.87	19.3	19.55
SSIM	0.827	0.841	0.839	0.822	0.487	0.497
MSE Loss	0.0158	0.0141	0.0144	0.0154	0.0692	0.0616

Table 4: Experiment of VAE model

可以看到隨著 kernel size 變大，模型的表現並沒有一致的變化，在 kernel size = 5 時表現最好，再加大表現只會更差，而將 stride 從 1 增至 2 表現都有所變差，故最後選擇的各項 hyperparameters 如 Table 5：

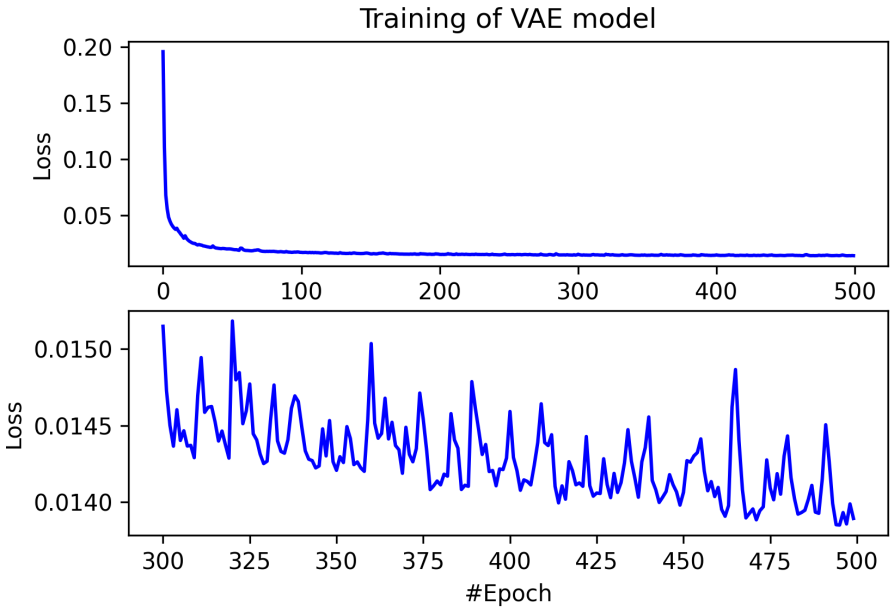

Figure 4: Loss of best VAE model

number of epochs	500
batch size	164
learning rate	10^{-3}
kernel size	5
stride	1
zero padding	2

Table 5: Hyperparameter of VAE model

2.2.3 Performance

以 **2.2.1** 中建構的模型架構和 **2.2.2** 中設定的各項 hyperparameters 所訓練出的 best VAE model 的表現如 Table 6 (可藉由執行 `performance.py` 驗證)：

PSNR	26.85
SSIM	0.841

Table 6: Performance of best VAE model

3 Some Questions

3.1 How to preserve the high-frequency information?

我們可以發現不管是 AE 還是 VAE model，在 reconstructed images 中，圖片的解析度相對於原圖都有明顯的下降，有兩個原因：

1. 圖片經過 (variational) autoencoder 轉換後，勢必會損失一些資訊，進而表現在 output 解析度較低。
2. 因爲我們在建構此類模型時重點是訓練其泛化能力 (Generalization ability) 用以處理未來新的資料，相對的就會對現有資料的擬合程度有所降低。

如果希望對現有資料的擬合程度上升以保留高頻資訊，可以試著使用不同的 loss function，以下以 VAE 模型爲例：將 loss function 改爲 L1 loss + KL Divergence，output 結果呈現如 Figure 5，可以發現圖形的解析度相對於 MSE loss + KL Divergence 來說有所提高，但依舊不如原圖，須注意此時模型的泛化能力相對於原本使用 MSE loss + KL Divergence 來說相對較差。

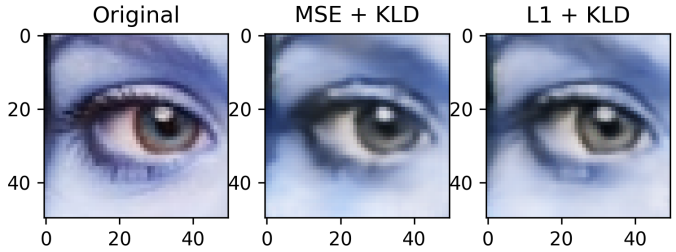

Figure 5: VAE output under different loss function

3.2 Add Gaussian noise into latent code

對 AE 和 VAE 模型的 latent tensors 加上維度一樣的 Gaussian noise tensors，然後再透過 decoder 還原圖片，結果呈現如 Figure 6 ∼ 13，可以發現 AE model generate 出的結果都有著明顯的雜訊，而 VAE model generate 出的結果則較爲平滑，這是因爲 VAE model 中間利用 μ 和 $\log(\sigma^2)$ 的 Gaussian distribution 結構有著比較好的抗噪能力，使得 output 不太會被 latent tensor 的雜訊所影響。

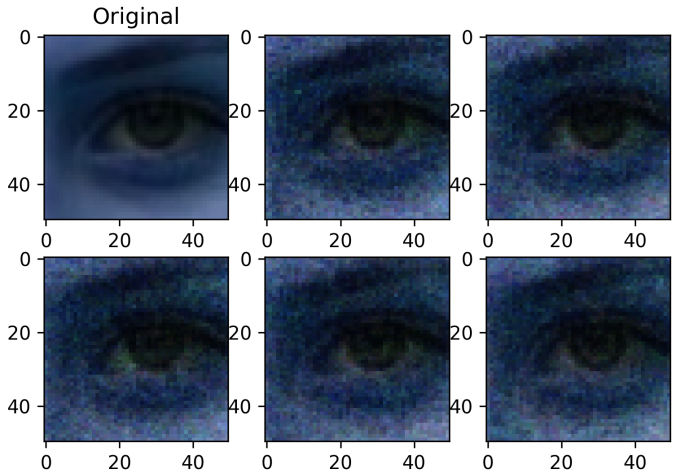

Figure 6: ID 3 - Adding Gaussian noise output (AE)

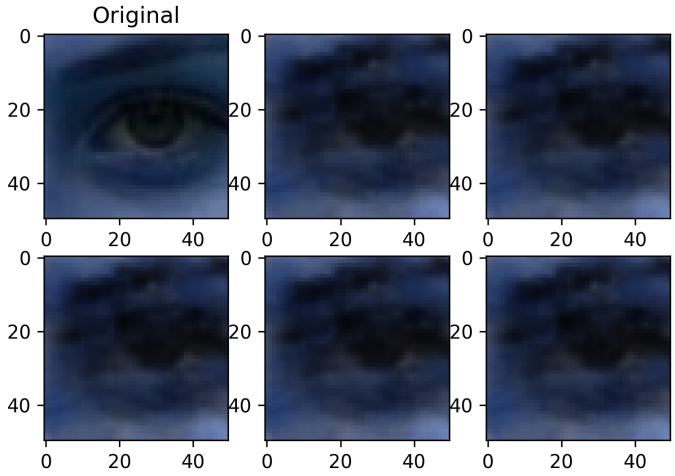

Figure 7: ID 3 - Adding Gaussian noise output (VAE)

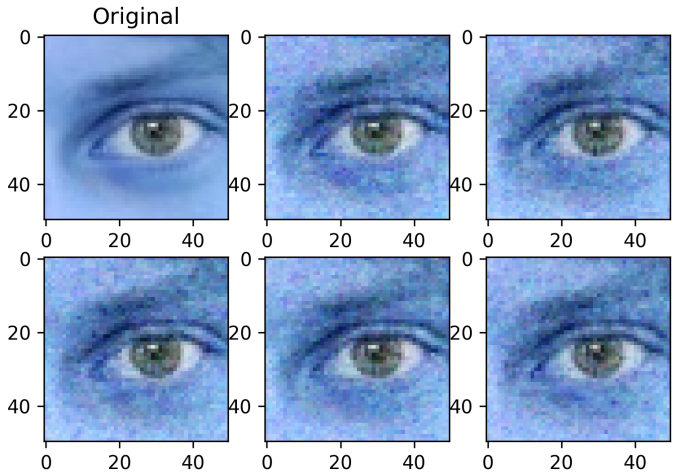

Figure 8: ID 227 - Adding Gaussian noise output (AE)

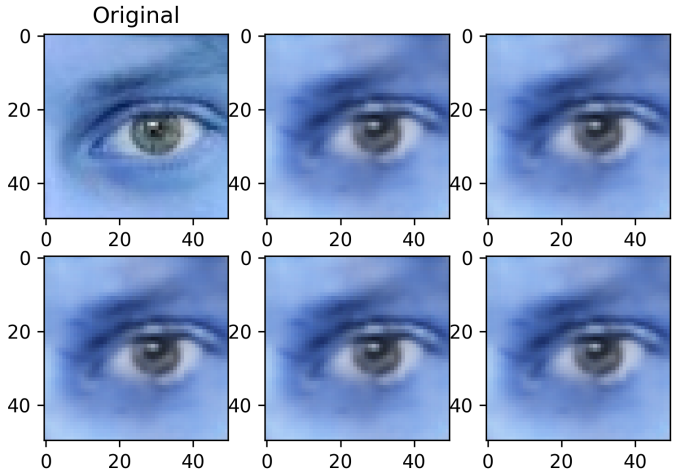

Figure 9: ID 227 - Adding Gaussian noise output (VAE)

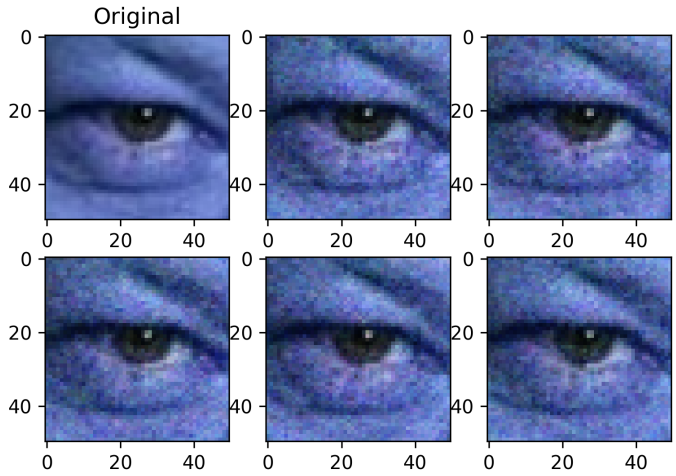

Figure 10: ID 841 - Adding Gaussian noise output (AE)

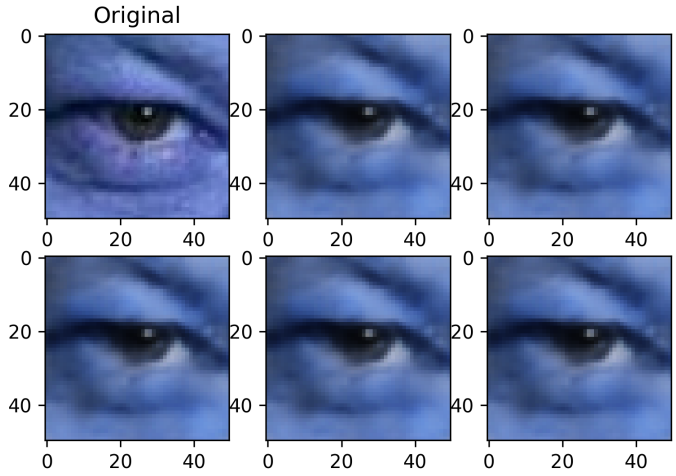

Figure 11: ID 841 - Adding Gaussian noise output (VAE)

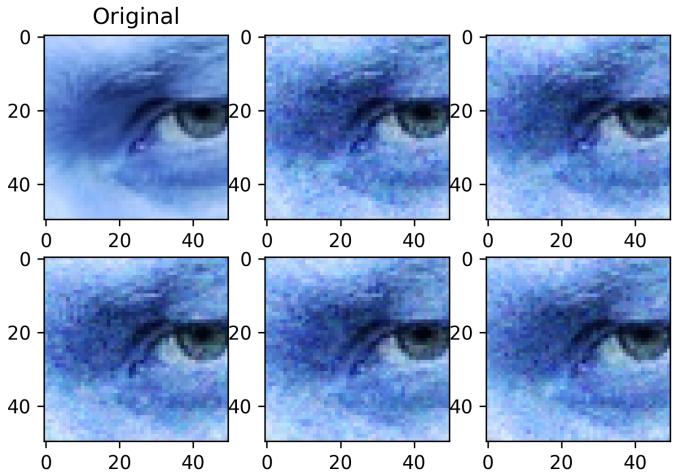

Figure 12: ID 1475 - Adding Gaussian noise output (AE)

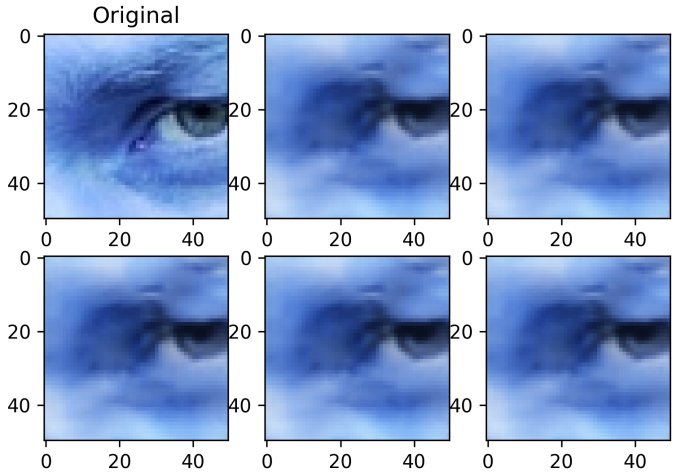

Figure 13: ID 1475 - Adding Gaussian noise output (VAE)