

1. Show the model architecture, implementation detail, PSNR, and SSIM scores between original images and reconstructed images. Please describe the methods to achieve the final result in detail, e.g., epochs, batch size, learning rate, reconstruction loss, etc. At last, plot training loss curve of your best model. (Both AE and VAE.) (10%)

AE

Architecture

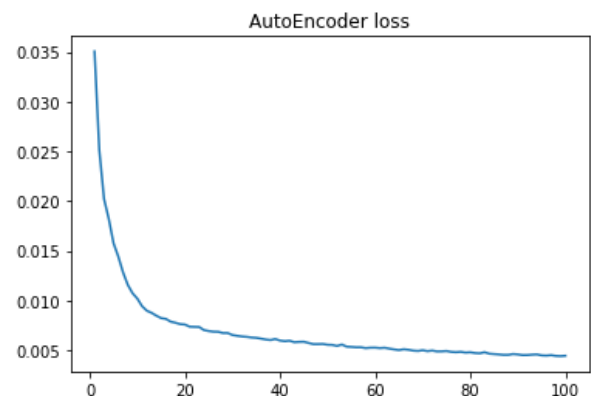
```
class ae(nn.Module):
    def __init__(self):
        super(ae, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(3*50*50, 256),
            nn.ReLU(),
            nn.Linear(256, 128),
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.Linear(128, 256),
            nn.ReLU(),
            nn.Linear(256, 3*50*50),
            nn.Sigmoid()
        )
    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

```
ae(
  (encoder): Sequential(
    (0): Linear(in_features=7500, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=128, bias=True)
    (3): ReLU()
  )
  (decoder): Sequential(
    (0): Linear(in_features=128, out_features=256, bias=True)
    (1): ReLU()
    (2): Linear(in_features=256, out_features=7500, bias=True)
    (3): Sigmoid()
  )
)
```

Loss function

```
loss_func = nn.MSELoss()

output = model(input_img).view(-1, 50, 50, 3)
loss = loss_func(output, input_img.view(-1, 50, 50, 3))
```



Settings

lr = 0.001
epoch = 100
batch = 20
loss function 採用 mse

PSNR & SSIM

average PSNR=24.043617248535156, SSIM=0.6841096557718369

在 learning rate 的部分, 在後續實驗有得出 learning rate = 0.001 會有比較好的成效。而在 epoch 的話, 由於 epoch 為 100 時已有不錯的 PSNR 及 SSIM, 也怕 epoch = 200 時會 overfitting, 所以就取100為值。在 batch 的部分, 在後續實驗有得出在 AE 這個架構下 batch = 15 時優於 batch = 20, 但也因為 batch 如果太小, 在更新 weight 時可能會以比較雜亂的方式更新(呈鋸齒狀), 因此最後取 20 作為 batch size。最後, 由於我們的目標是為了重建image, 而我們希望 output image 跟 input image 越相似越好, 因此 loss function 選用 mse, 希望 output 的每個 pixel 都能跟原 image 相近。

VAE

Architecture

```
class vae(nn.Module):
    def __init__(self):
        super(vae, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(3*50*50, 4096),
            nn.ReLU(),
            nn.Linear(4096, 1024),
            nn.ReLU(),
            nn.Linear(1024, 512),
            nn.ReLU()
        )
        self.mean = nn.Linear(512, 256)
        self.variance = nn.Linear(512, 256)
        self.decoder = nn.Sequential(
            nn.Linear(256, 512),
            nn.ReLU(),
            nn.Linear(512, 1024),
            nn.ReLU(),
            nn.Linear(1024, 4096),
            nn.ReLU(),
            nn.Linear(4096, 3*50*50),
            nn.Sigmoid()
        )

    def encode(self, x):
        x = self.encoder(x)
        m = self.mean(x)
        v = self.variance(x)
        return m, v

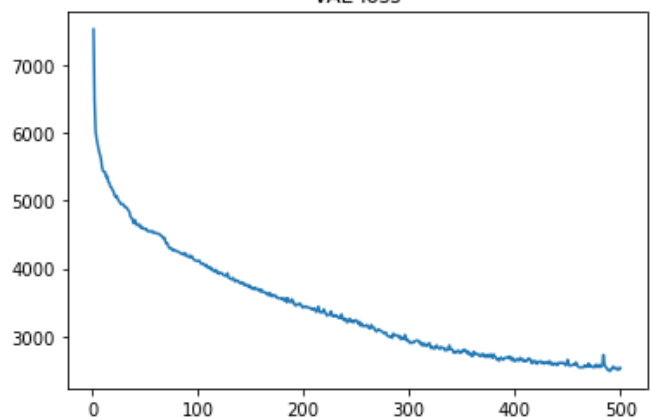
    def forward(self, x):
        m, v = self.encode(x)
        e = torch.randn_like(v)
        t = m + e * torch.exp(v)
        out = self.decoder(t)
        return m, v, t, out
```

```
vae(
    (encoder): Sequential(
      (0): Linear(in_features=7500, out_features=4096, bias=True)
      (1): ReLU()
      (2): Linear(in_features=4096, out_features=1024, bias=True)
      (3): ReLU()
      (4): Linear(in_features=1024, out_features=512, bias=True)
      (5): ReLU()
    )
    (mean): Linear(in_features=512, out_features=256, bias=True)
    (variance): Linear(in_features=512, out_features=256, bias=True)
    (decoder): Sequential(
      (0): Linear(in_features=256, out_features=512, bias=True)
      (1): ReLU()
      (2): Linear(in_features=512, out_features=1024, bias=True)
      (3): ReLU()
      (4): Linear(in_features=1024, out_features=4096, bias=True)
      (5): ReLU()
      (6): Linear(in_features=4096, out_features=7500, bias=True)
      (7): Sigmoid()
    )
)
```

Loss function

```
m, v, t, output = model2(input_img)
sigma = torch.exp(v)
kl_loss = (sigma**2 + m**2 - torch.log(sigma) - 1/2).sum()
loss = ((input_img-output)**2).sum()+kl_loss
```

VAE loss



Settings





















batch = 20
 epoch = 500
 lr = 0.0005
 loss function 採用 mse + KL divergence loss

PSNR & SSIM

PRETRAINED: average PSNR=25.30364418029785, SSIM=0.8084081480454736

在 learning rate 的部分，經多次的調整，最後決定將 learning rate 設成 0.0005 讓 model 的 weight 可以以適當的速度更新。而在 epoch 的部分，原本是取 400，但經實驗後發現 500 的效果更好，只是會需要再多花一些時間訓練，因此最後決定 epoch = 500。至於 batch size，可以在實驗中發現 20 是相對適當的值，因此取 20 為 batch size。最後在 loss function 的部分，除了希望 output 的每個 pixel 都能跟 input image 相近而採用 mse 以外，還須考慮到 mean 跟 variance 的調整，因此有加上 KL divergence loss。

2. Generate another 100 new samples from the original 20 samples by adding Gaussian noise into the latent code (Generate 5 samples from one sample) and store them into gen_data.npy and save the corresponding label to gen_label.npy. For the fairness, we specified the image id as [1-5, 226-230, 841-845, 1471-1475]. Eventually, please demonstrate the generated images of [3, 227, 841, 1475] with labels in the report. (Totally 20 images, and you only need to show the generated results of VAE in this part!) (10%)

image id	1	2	3	4	5	label
3						0
227						1
841						1
1475						1

3. In the training process, you would adjust hyperparameters (epochs, batch size, ...) to achieve higher PSNR and SSIM results. How do you choose these hyperparameters? Please conduct experiments (in Table form) to validate that these hyper-parameters are suitable to achieve your best result. (Both AE and VAE.) (10%)

AE

default settings

lr = 0.001

epoch = 100

batch = 20

latent_dim = 128

train_loss = 1 epoch的loss總和取平均

epoch	training loss	average PSNR	average SSIM
20	0.007452	21.8845	0.5755
80	0.005042	23.4440	0.6533
100	0.004594	23.7426	0.6809
200	0.003934	24.3782	0.7110

batch	training loss	average PSNR	average SSIM
15	0.004494	23.8288	0.6837
20	0.004600	23.8143	0.6756
25	0.004804	22.0016	0.6459
lr	training loss	average PSNR	average SSIM
0.1	0.195075	7.1727	0.0986
0.01	0.014990	18.6169	0.4447
0.001	0.004640	23.7831	0.6757
0.0001	0.007134	21.9108	0.5731
Description 在 learning rate 的部分, 在後續實驗有得出 learning rate = 0.001 會有比較好的成效。而在 epoch 的話, 由於 epoch 為 100 時已有不錯的 PSNR 及 SSIM, 也怕 epoch = 200 時會 overfitting, 所以就取100為值。在 batch 的部分, 在後續實驗有得出在 AE 這個架構下 batch = 15 時優於 batch = 20, 但也因為 batch 如果太小, 在更新 weight 時可能會以比較雜亂的方式更新(呈鋸齒狀), 因此最後取 20 作為 batch size。			

VAE

default settings lr = 0.0005 epoch = 400 batch = 20 latent_dim = 128 train_loss = 1 epoch的loss總和取平均			
epoch	training loss	average PSNR	average SSIM
100	4321.999023	18.6275	0.4439
200	3563.720459	20.4405	0.5608
400	2597.352783	24.7008	0.7856
500	2530.466553	25.2245	0.8064
batch	training loss	average PSNR	average SSIM
15	2254.333984	22.6511	0.6912
20	2597.352783	24.7008	0.7856

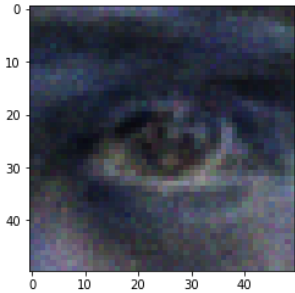
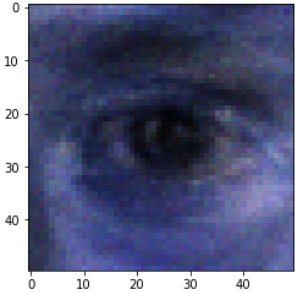
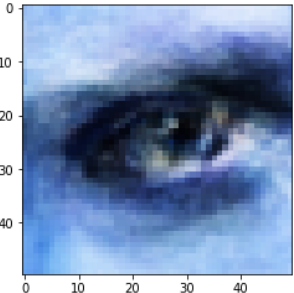
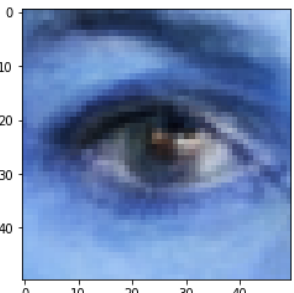
25	3564.947021	22.6437	0.6962
latent dim	training loss	average PSNR	average SSIM
2	2547.007568	18.5278	0.4499
64	1324.942139	24.3096	0.7585
256	2597.352783	24.7008	0.7856
512	4999.360352	21.7673	0.6429

Description

在 epoch 的部分，原本是取 400，但經實驗後發現 500 的效果更好，只是會需要再多花一些時間訓練，因此最後決定 epoch = 500。至於 batch size，可以在實驗中發現 20 是相對適當的值，因此取 20 為 batch size。最後在 latent_dim 的部分，可以發現等於 2 時，雖然 training loss 有比較低，但是 average PSNR & SSIM 都沒有達 spec 給的標準，而在等於 64 及 256 時，average PSNR & SSIM 都有達 spec 給的標準，因此我最後選了 average PSNR & SSIM 較高的 256。

*黃色表示有達 spec 給的標準

4. Please use the same gaussian noise to generate an image by VAE and AE. Discuss the difference between VAE and AE and generate at least two sets of images to explain. (10%)

AE	VAE
<p>(1)</p>  <p>(2)</p> 	<p>(1)</p>  <p>(2)</p> 
<h3>Description</h3>	

假設 latent_dim = N 且 input 有 M 筆 data, 那 AE 的 encoder 就是在 N 維空間產生 M 個點, 因此如果是訓練時沒有 train 到的 image, 可能 encode 到某兩點中間的位置, 在 decode 時就有可能不會得到我們想像中的圖, 但對於 VAE 來說, 從 encoder 出來後, 每個 data 都產生 N 維的 normal distribution, 這樣就類似一個高斯混合模型, 涵蓋的範圍較廣, 因此不管編碼 map 到哪個位置, 從 decoder 產出的 image 都會有較高的解析度。

5. In the reconstructed images, you might find that these images are not as sharp as the original images. Please explain the reasons for it. Second, what could we do to preserve the high-frequency information in the original images in your AE or VAE models? Please do some experiments to solve this problem. (Hint: Loss) (10%)

可能是因為 kl_loss (L2 loss) 的原因, 導致在 reconstruct image 時, 解析度沒有原圖好。因此在參考一些文獻後, 發現有一篇提到可以在 decoder 最後加兩層 layer, 並在 loss function 新增「從編碼經過 decoder 再經過兩層 layer 後的 output image 與 input image 的 mse loss」, 就可以改善解析度。因此我用 AE 做了實驗, 但我只做了在 decoder 後面新增兩層 layer, 就已經改善解析度的問題。以下為我在 AE 做的調整、它的 average PSNR & SSIM 結果及 input & output image。

```
class ae(nn.Module):
    def __init__(self):
        super(ae, self).__init__()
        self.encoder = nn.Sequential(
            nn.Linear(3*50*50, 256),
            nn.ReLU(),
            nn.Linear(256, 128),
            nn.ReLU()
        )
        self.decoder = nn.Sequential(
            nn.Linear(128, 256),
            nn.ReLU(),
            nn.Linear(256, 3*50*50),
            nn.ReLU(),
            nn.Linear(3*50*50, 3*50*50),
            nn.ReLU(),
            nn.Linear(3*50*50, 3*50*50),
            nn.Sigmoid()
        )
    def forward(self, x):
        x = self.encoder(x)
        x = self.decoder(x)
        return x
```

average PSNR=32.15559005737305, SSIM=0.9383788109146037

