Advanced Deep Learning 2024
Assignment 2

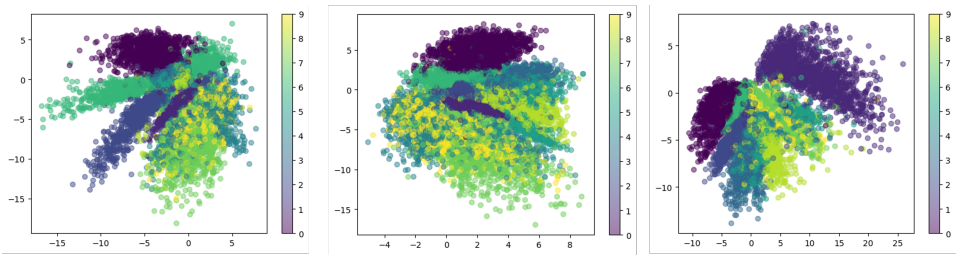
Dengke Chen

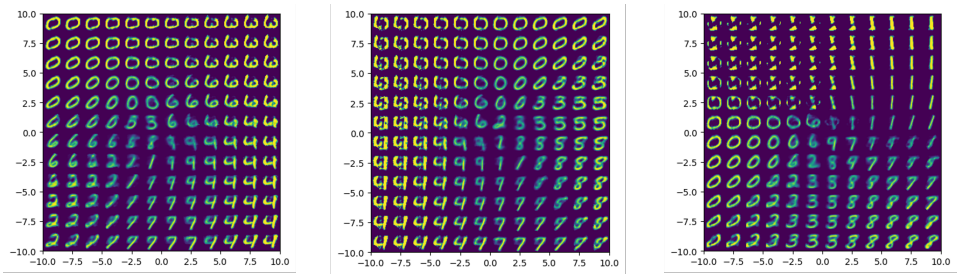
May 14, 2024

1 Autoencoders

1.1 Q1

Answer:
The result plots of running autoencoders.ipynb three times





The qualitative comparison between the resulting plots.
Latent Space Projections

1. First time (Top Left Image): The projection shows clusters with good separation but some overlap between different digit representations. The spread of colors (representing different digits) is wide, suggesting varied representation in the latent space.

2. Second time (Top Middle Image): This projection appears more compressed with increased overlap between clusters, especially in the center where multiple

colors merge. This could indicate less effective encoding leading to similar latent representations for different digits.

3. Third time (Top Right Image): This projection is more expansive with clearer separation between some clusters, particularly noticeable in the green and purple areas. The spread suggests a more effective differentiation in the latent space among some digits.

Reconstructed Images from Latent Space

1. First time (Bottom Left Image): The digits are recognizable but slightly blurred, indicating decent but not perfect reconstruction. There's visible variance in digit clarity, especially for digits like 9 and 0.

2. Second time (Bottom Middle Image): The images appear slightly more uniform than in the first run, but still show significant blurring and imperfections, reflecting consistent issues with reconstructing distinct features of each digit.

3. Third time (Bottom Right Image): Similar to the first two, the reconstructed images show blurring, with some digits like 3 and 8 being less distinct. However, the overall shape and orientation seem well captured.

Across three runs, the model consistently shows a capability to cluster and somewhat reconstruct digits, but there's notable variation in the quality and distinctiveness of both projections and reconstructions.

The varied clustering effectiveness in the latent space suggests differences in how well the encoder is capturing the underlying features of the digits in different runs. This could be due to initialization, optimization paths, or other stochastic elements of the training process.

The reconstructed images, while generally capturing the overall shapes and styles of the digits, often lack sharpness and precision, which could be improved potentially with further tuning of the model, more complex architectures, or additional training epochs.

These highlight the importance of hyperparameter tuning, training stability, and possibly exploring more complex models or additional regularization techniques to improve both the encoding and decoding stages of the autoencoder.

1.2 Q2

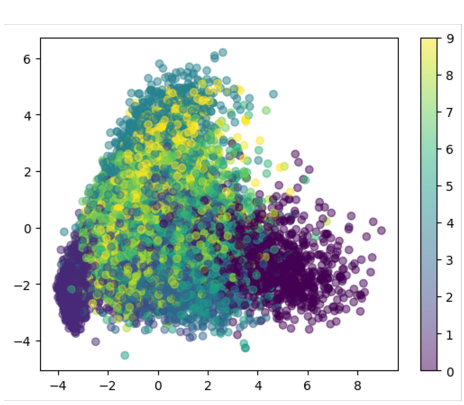
Answer:

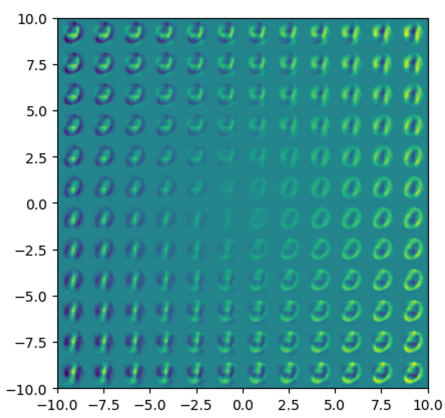
PCA module:

```
def plot_pca_latent(data_loader, pca, dim1=0, dim2=1,
    num_batches=100):
     for i, (x, y) in enumerate(data_loader):
          x_flat = x.view(x.size(0), -1).numpy()
          z = pca.transform(x_flat)
          plt.scatter(z[:, dim1], z[:, dim2], c=y, alpha=0.5)
          if i > num_batches:
               plt.colorbar()
               break
     plt.show()
```

```
def plot_pca_reconstructed(pca, w, h, r0=(-10, 10), r1=(-10,
    10), n=12):
    img = np.zeros((n*w, n*h))
    mean = pca.mean_
    components = pca.components_
    explained_variance = pca.explained_variance_

    for i, yi in enumerate(np.linspace(*r1, n)):
        for j, xi in enumerate(np.linspace(*r0, n)):
            z = np.array([xi, yi])
            x_hat = mean + np.dot(z * np.sqrt(
                explained_variance), components)
            x_hat = x_hat.reshape(w, h)
            img[(n-1-i)*w:(n-1-i+1)*w, j*w:(j+1)*w] = x_hat
    plt.imshow(img, extent=[*r0, *r1])

from sklearn.decomposition import PCA
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Lambda(lambda x: torch.flatten(x))
])

data = datasets.MNIST('./data', transform=transform, download=
    True)
n, w, h = data.data.shape
data_loader = torch.utils.data.DataLoader(data, batch_size=128,
    shuffle=True)

x_example, _ = next(iter(data_loader))
x_flat = x_example.view(x_example.size(0), -1)
pca = PCA(n_components=2)
pca.fit(x_flat)

plot_pca_latent(data_loader, pca)
plt.savefig('pca_latent.pdf')

plot_pca_reconstructed(pca, w, h)
plt.savefig('pca_reconstruction.pdf')
```

The result plots of PCA





Qualitative Comparison of PCA and Autoencoder.

Latent Space Comparison: The PCA latent space plot shows a more triangular clustering pattern, with less distinct separation between digit groups compared to the autoencoder. The clusters are more mixed, indicating that PCA captures the overall structure but not as finely as the autoencoder. The PCA scatter plot shows a different clustering pattern. The digits are less distinctly separated compared to the autoencoder's latent space. However, there is still some grouping visible, indicating PCA's ability to capture the variance in the dataset.

Reconstruction Comparison: The PCA reconstructions appear to be quite uniform but less detailed compared to the autoencoder results. The digits are recognizable, but there is a noticeable blur and loss of fine details, especially in the more complex digits. This indicates that while PCA can reduce dimensionality effectively, it does not reconstruct the data as accurately as the autoencoder.

Conclusion

Latent Space: The autoencoder provides a more nuanced and detailed latent space representation with clearer separations between different digit clusters, especially in the third time from task 1. PCA, while effective in reducing dimensionality, does not capture the finer distinctions between digits as well as the autoencoder.

Reconstructions: The autoencoder significantly outperforms PCA in reconstructing the original images. The autoencoder's reconstructions are sharper and more detailed, whereas PCA's reconstructions tend to be blurrier and lack finer details.

Consistency of Results

• Autoencoder: Running the autoencoder multiple times can result in slightly different latent space representations and reconstructions due to variations in training.

• PCA: The results of PCA are deterministic, meaning that running PCA multiple times on the same dataset will yield identical results, as PCA does not involve random initialization or iterative optimization. **Run my PCA program twice, the result don't change.**

The autoencoder is more powerful for encoding and reconstructing high-dimensional data, providing more detailed and accurate reconstructions and a more informative latent space. PCA is a simpler and deterministic method, useful for understanding overall data structure but less effective in capturing detailed variations within the data.

1.3 Q3

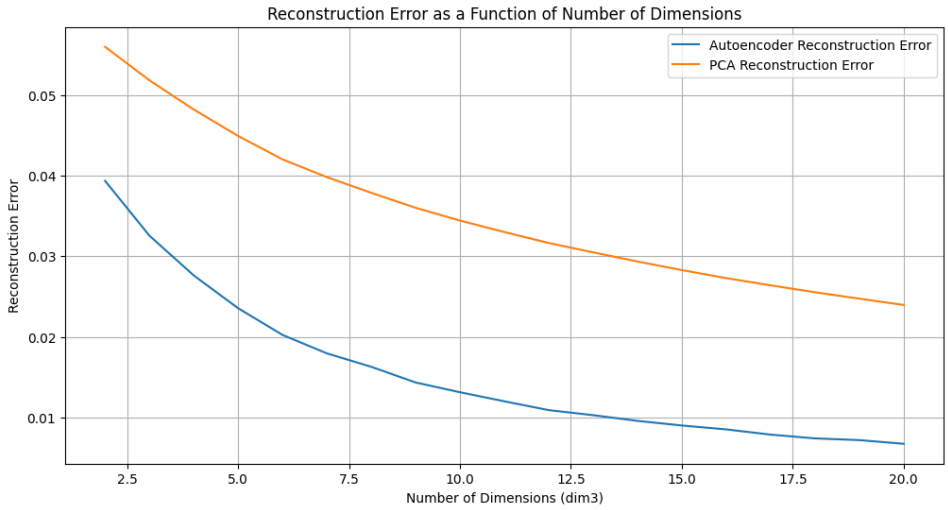
Answer:

Main Code:

```
def compute_error(model, data_loader):
    model.eval()
    total_loss = 0
    criterion = nn.MSELoss()
    with torch.no_grad():
```

```
            for data in data_loader:
                inputs, labels = data
                inputs = inputs.to(device)
                outputs = model(inputs)
                loss = criterion(outputs, inputs)
                total_loss += loss.item()
        return total_loss / len(data_loader)

    dimensions = range(2, 21)
    autoencoder_errors = []
    pca_errors = []

    for dim in dimensions:
        autoencoder = Autoencoder(784, 512, dim)
        optimizer = optim.Adam(autoencoder.parameters(), lr=0.001)
        loss_function = nn.MSELoss()
        trained_autoencoder, _ = train(data_loader, autoencoder,
            optimizer, loss_function, epochs=20)

        ae_error = compute_error(trained_autoencoder, data_loader)
        autoencoder_errors.append(ae_error)

        pca = PCA(n_components=dim)
        x_flat = np.vstack([x.view(x.size(0), -1).numpy() for x, _
            in data_loader])
        pca.fit(x_flat)
        x_pca = pca.transform(x_flat)
        x_reconstructed = pca.inverse_transform(x_pca)
        pca_error = np.mean(np.square(x_flat - x_reconstructed))
        pca_errors.append(pca_error)
```

The result plots of PCA



Reconstruction Error Trend:

PCA: The reconstruction error decreases as the number of dimensions increases, following a smooth curve. However, the error remains higher compared to the autoencoder for all values of dim3.

Autoencoder: The reconstruction error also decreases as the number of dimensions increases. The error decreases more rapidly compared to PCA and reaches lower values, indicating better reconstruction performance.

Rate of Decrease:

PCA: The error reduction rate slows down as the number of dimensions increases. This suggests diminishing returns for adding more dimensions beyond a certain point.

Autoencoder: The autoencoder shows a more significant reduction in reconstruction error with increasing dimensions, especially in the lower dimension range. This indicates that the autoencoder is more efficient in capturing the essential features of the data.

Comparison of PCA and Autoencoder

Number of Dimensions: Both methods benefit from increased dimensions, but the autoencoder consistently outperforms PCA across all values of dim3.

Performance: The autoencoder achieves lower reconstruction errors with fewer dimensions compared to PCA, demonstrating its superior ability to encode and reconstruct high-dimensional data.

Determining the Optimal Value of dim3

Elbow Method: Plotting the explained variance against the number of dimensions and looking for an "elbow" point where the explained variance starts to level off can also help in selecting the optimal number of dimensions. This involves looking for a point on the error curve where the rate of improvement slows down significantly, forming an "elbow."

For both PCA and the autoencoder, this point seems to be around dim3 = 10, where the decrease in reconstruction error starts to flatten.

Observe this plot closely, I get:

Autoencoder: The optimal number of latent variables (dim3) can be determined by looking for the "elbow" point in the plot, where the decrease in reconstruction error begins to level off. This point represents a balance between model complexity and reconstruction accuracy. From the plot, this appears to be around 10 latent dimensions.

PCA: Similarly, the optimal number of eigenvectors can be chosen by identifying the elbow point. For PCA, the reduction in error is more gradual, so the optimal number may be slightly higher, around 15 dimensions.

Overall, autoencoders are more efficient in encoding high-dimensional data into a lower-dimensional latent space while preserving the data's essential features. it provides a more efficient and accurate method for dimensionality reduction and data reconstruction.

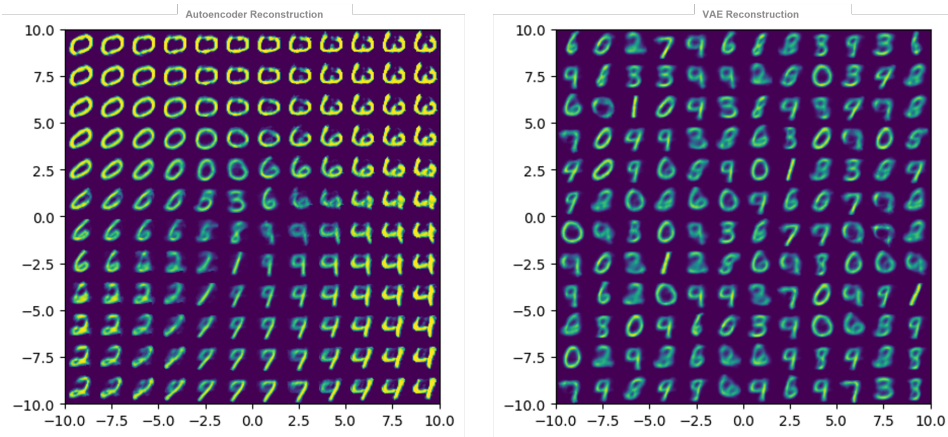
1.4 Q4

Answer:

The Code that Generate new reconstructed images of Variational autoencoder:

```
def plot_vae_reconstructed(decoder, w, h, latent_dim, r0=(-10,
    10), r1=(-10, 10), n=12):
    img = np.zeros((n*w, n*h))
    for i, y in enumerate(np.linspace(*r1, n)):
        for j, x in enumerate(np.linspace(*r0, n)):
            z = torch.randn(1, latent_dim).to(device)
            x_hat = decoder(z)
            x_hat = x_hat.reshape(w, h).to('cpu').detach().
                numpy()
            img[(n-1-i)*w:(n-1-i+1)*w, j*w:(j+1)*w] = x_hat
    plt.imshow(img, extent=[*r0, *r1])

latent_dim = trained_model.fc21.out_features

plot_vae_reconstructed(trained_model.decode, w, h, latent_dim)
plt.show()
```

The result reconstructed images of autoencoder and variational autoencoder



Qualitative Comparison of Autoencoder and VAE Reconstructions

Autoencoder Reconstruction

• The autoencoder reconstructions (left image) exhibit clear and well-defined digit shapes, especially in the outer regions of the latent space.

• There is a noticeable repetition and consistency in the shapes of the digits across different parts of the latent space.

• The transition between different digit classes is less smooth, with clear boundaries between different types of digits.

Variational Autoencoder Reconstruction

• The VAE reconstructions (right image) show more diversity in the generated digits, with a smoother variation across the latent space.

• The digits are less repetitive and more varied compared to the autoencoder reconstructions, showcasing the VAE's ability to explore the latent space more thoroughly.

• The transitions between different digit classes are smoother, with more gradual changes in the shapes of the digits as you move across the latent space.

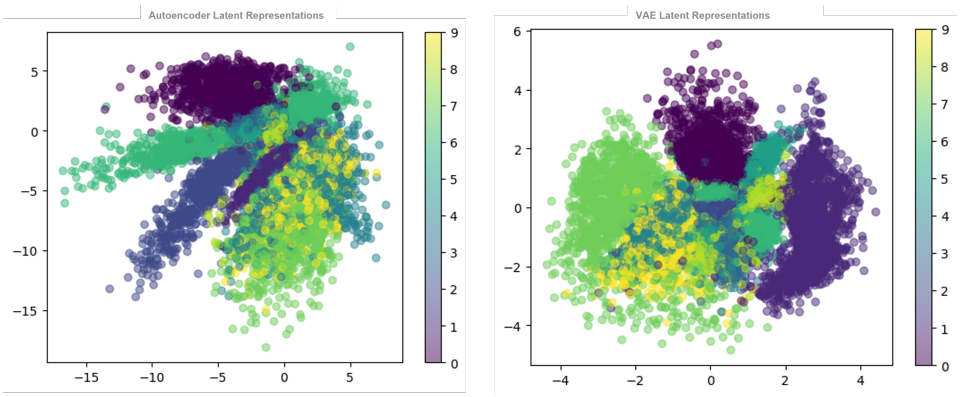
Summary

• Autoencoder: Produces clearer, more consistent digit shapes, suggesting that it learns a more direct mapping from the latent space to the output space, but lacks smooth transitions between different digit classes.

• VAE: Generates a wider variety of digits with smoother transitions, reflecting its probabilistic nature and better latent space exploration. Useful for applications needing more robust generalization from the latent space, capable of generating a wider range of variations, even if some outputs are less sharp.

The VAE's ability to produce a more diverse set of digits and smoother transitions makes it more suitable for tasks requiring generative models with better representation learning. However, the autoencoder's reconstructions are sharper and more consistent for individual digit classes. The VAE's probabilistic approach provides better coverage of the latent space, making it useful for generating diverse and novel outputs, while the autoencoder excels in producing clear and consistent reconstructions.

The latent representations of autoencoder and variational autoencoder



Qualitative Comparison of Autoencoder and VAE Latent Representations

Autoencoder Latent Representations (Left Image)

• The latent space appears more scattered and less structured.

• There are clear clusters, but they are more dispersed and not as well-separated.

• Some overlap between different digit classes is noticeable, indicating that the autoencoder has a harder time distinctly separating different digits in the latent space.

VAE Latent Representations (Right Image)

• The latent space is more compact and structured.

• Different digit classes form more distinct clusters, although there is still some overlap.

• The clusters are more uniformly distributed, which is a characteristic of the VAE due to the KL divergence regularization that encourages the latent space to follow a standard normal distribution, suggesting the VAE captures the underlying data distribution more effectively.

Analysis

• Autoencoder: The latent space is less structured, with a broader spread and more overlap between digit classes. This indicates that while the autoencoder learns to encode the data, it may not be as effective in distinguishing between different classes in a compact manner.

• VAE: The VAE's latent space is more structured, with better-separated and more compact clusters. This reflects the VAE's ability to model the data distribution and impose a more organized latent structure through the KL divergence term.

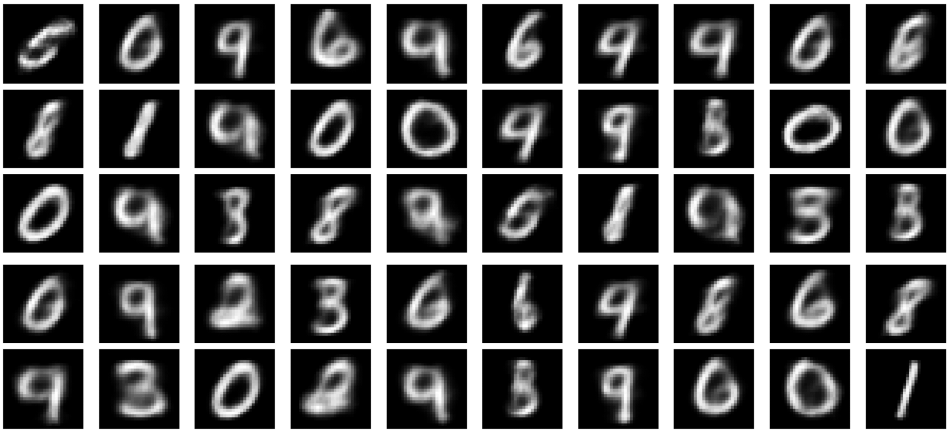
Conclusion

• The autoencoder latent space, while capturing the variance in the data, results in more overlap and less distinct separation between different classes.

• The VAE latent space shows a more organized and compact representation of the data, with clearer class distinctions. This is beneficial for generating diverse yet coherent samples from the latent space.

Overall, the VAE demonstrates superior performance in structuring the latent space compared to the standard autoencoder, aligning with its objective of modeling data distribution more effectively.

Run plot_generated_samples five times to plot the generated samples



By observing the plots generated from the VAE model by running the plot_generated_samples function, I can get several observations:

Each set of generated samples displays recognizable digit shapes, although the exact digits and their appearances vary between runs. This indicates that the VAE consistently produces outputs within the distribution of digit-like shapes, adhering to the learned data distribution.

There is noticeable variation between different runs. The digits are not identical, and some digits appear more frequently in certain runs than in others. This variation is expected because the latent vectors are sampled from a standard normal distribution, leading to different generated images each time.

The quality of generated digits is relatively stable across different runs. Most digits are identifiable, though some may appear slightly distorted or blurred. The generated digits maintain their overall structure, indicating the VAE's robustness in generating coherent samples.

Each run showcases a mix of different digit types, demonstrating the VAE's ability to capture the diversity of the MNIST dataset. The digits' representation in the latent space allows for generating a variety of outputs, highlighting the model's capability to explore different regions of the latent space.

Conclusion

Running the function multiple times results in different sets of generated images due to the stochastic nature of sampling from the latent space. While individual digits vary between runs, the overall quality and diversity remain consistent. This variability is expected and desirable in a VAE, showcasing its ability to sample from a continuous latent space and generate diverse outputs.

The VAE's ability to generate diverse and recognizable digits from random samples of the latent space demonstrates its effectiveness in capturing the underlying data distribution. The observed variations confirm that the VAE does not produce deterministic outputs but rather a range of plausible samples, which is advantageous for generative tasks.

By running the plot_generated_samples function multiple times, can confirm that the VAE reliably produces varied and coherent samples, reflecting its strength in generative modeling.