Variational Autoencoder (VAE) on Fashion MNIST

TUTORIAL 3

Name: VARUN GADI
Registration Number: RA2211027010203
Section: AD2
Department: DSBS

1. Introduction

Autoencoders and Variational Autoencoders (VAEs) are widely used in deep learning for representation learning and generative modeling. In this tutorial, we focus on VAEs applied to the Fashion MNIST dataset. Our objective is to explore how VAEs learn structured latent spaces and how they can be used for image reconstruction and generation.

Unlike traditional autoencoders, VAEs impose a probabilistic structure on the latent space, which allows us to generate new, meaningful data samples instead of just reconstructing the input. This tutorial covers the implementation of a VAE from scratch, including training, visualization, latent space exploration, and a comparative study with a standard autoencoder (AE).

2. Dataset Overview (Fashion MNIST)

The Fashion MNIST dataset consists of 70,000 grayscale images, each 28×28 pixels, categorized into 10 different fashion items. This dataset serves as a drop-in replacement for the classic MNIST digit dataset, making it ideal for evaluating generative models like VAEs.

A. Classes in Fashion MNIST

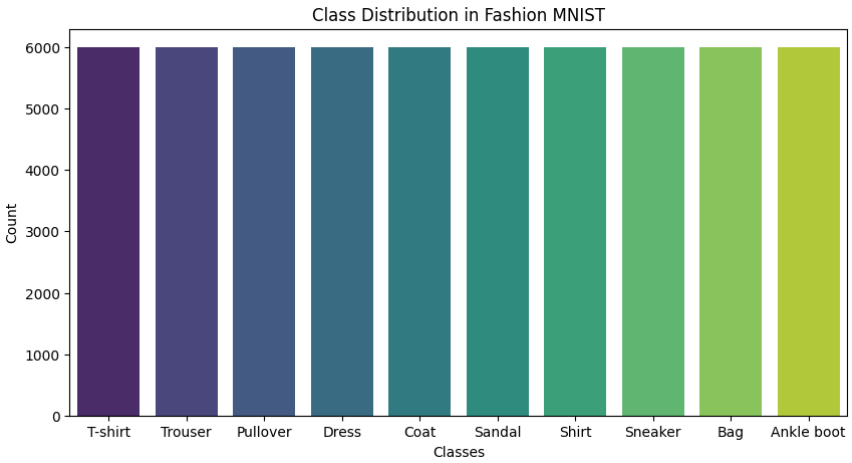
- Each image belongs to one of the following 10 classes:

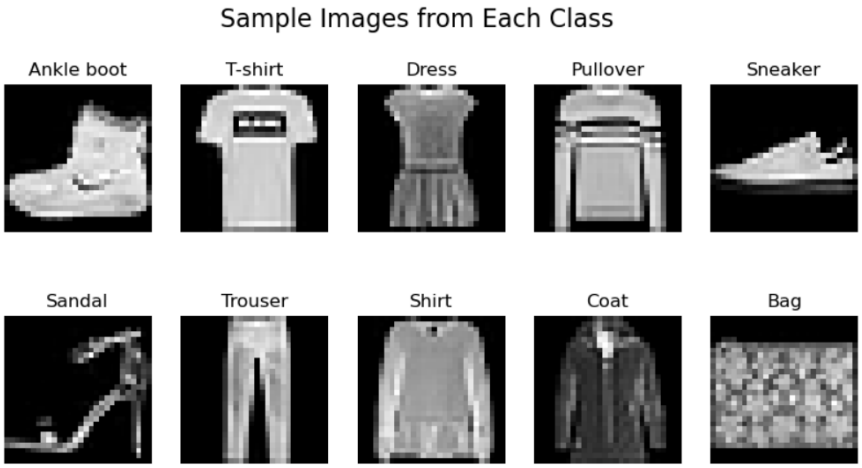
Label	Class Name
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

B. Data Preprocessing & Normalization

Before training the model, the dataset was normalized between 0 and 1 to help the network converge faster. The transformation pipeline included:

- Rescaling pixel values to [0,1]
- Flattening the images into a 1D vector of 784 features.





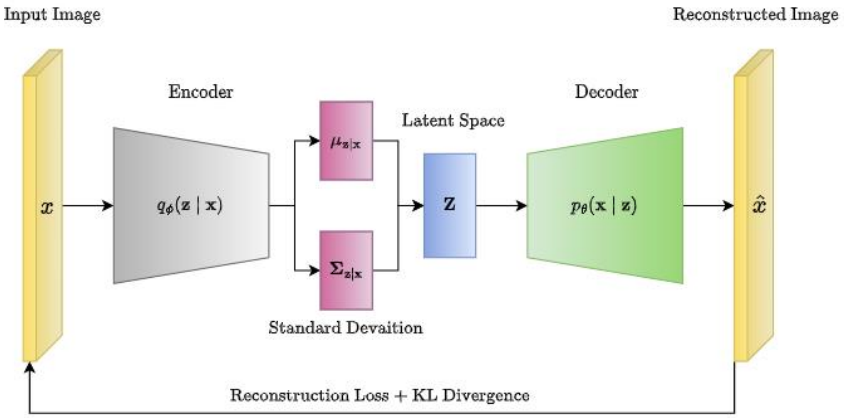
3. Model Architecture & Implementation

A. Variational Autoencoder (VAE)

VAE consists of three main components:

- Encoder → Maps input images into a latent space distribution (μ, $\log(\sigma^2)$).
- Reparameterization Trick → Samples z from this distribution in a differentiable way.

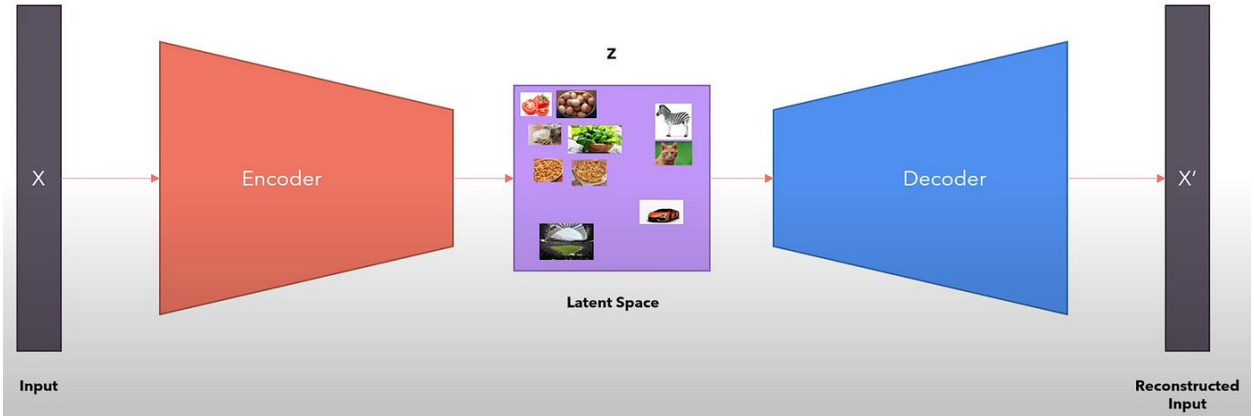
- Decoder → Reconstructs images from the latent representation.



B. Encoder Network

The encoder compresses the 28×28 input into a lower-dimensional representation:

```
self.encoder_fc = nn.Sequential(
nn.Linear(28 * 28, 256),
nn.ReLU(),
nn.Linear(256, 128),
nn.ReLU()
)
self.mu_fc = nn.Linear(128, LATENT_DIM)
self.log_var_fc = nn.Linear(128, LATENT_DIM)
```



C. Reparameterization Trick

To ensure backpropagation works through stochastic sampling, we apply the Reparameterization Trick:

```
def reparameterize(self, mu, log_var):
    std = torch.exp(0.5 * log_var)
    epsilon = torch.randn_like(std)
    return mu + epsilon * std
```

It allows gradient descent to update μ and $\log(\sigma^2)$ directly.

D. Decoder Network

The decoder reconstructs images from sampled latent vectors:

```
self.decoder_fc = nn.Sequential(
nn.Linear(LATENT_DIM, 128),
nn.ReLU(),
nn.Linear(128, 256),
nn.ReLU(),
nn.Linear(256, 28 * 28),
nn.Sigmoid()
)
```

Uses Sigmoid to ensure pixel values stay in the [0,1] range.

4. Training Procedure

A. Loss Function

The total VAE loss consists of:

- Reconstruction Loss (Binary Cross-Entropy) → Measures how well the output resembles the input.
- KL Divergence Loss → Encourages the latent distribution to be close to a standard normal distribution.

```
loss = reconstruction_loss + beta * kl_divergence
```

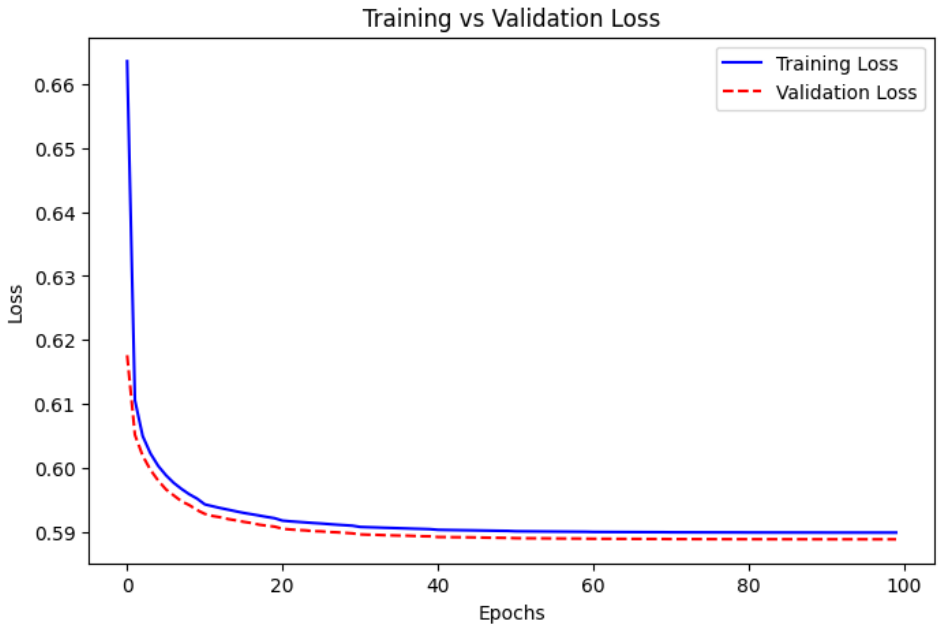
$$KL(q(z|x)//p(z))=21i=1\sum d(1+\log(\sigma i2)-\mu i2-\sigma i2)$$

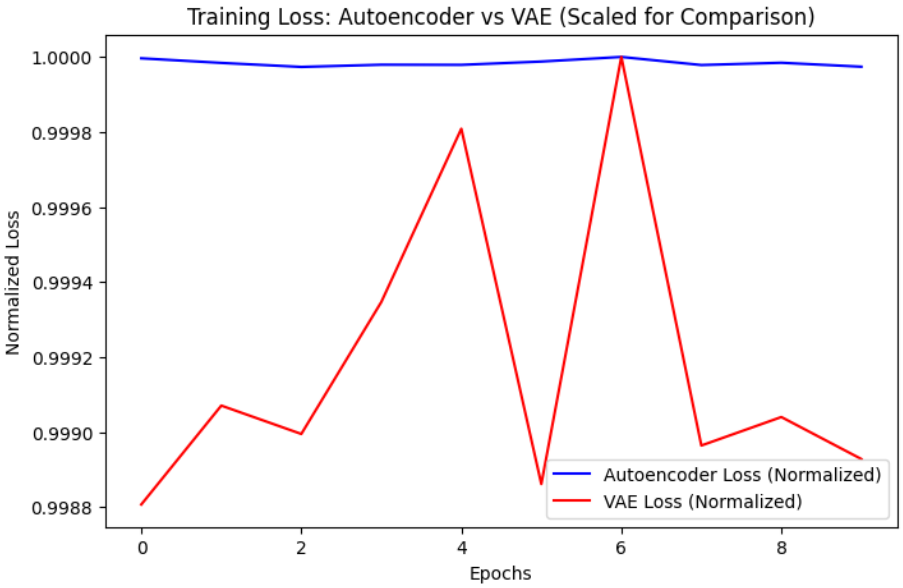
B. Training Hyperparameters

Parameter	Value
Batch Size	128
Latent Dimension	64
Optimizer	Adam (lr = 0.001)
Epochs	150

5. Results & Analysis

A. Training Loss Curve





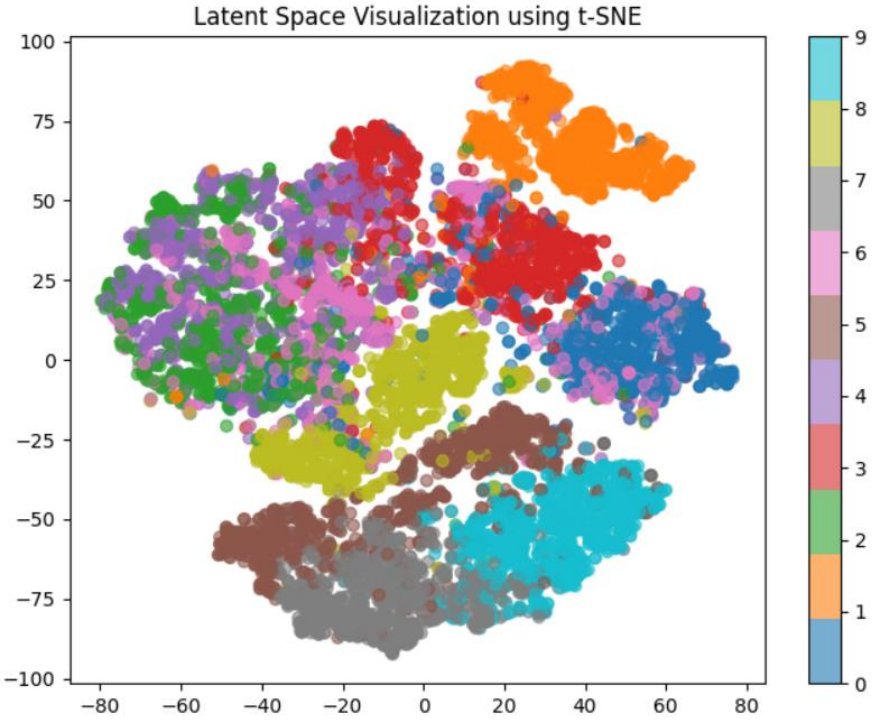
B. Original vs. Reconstructed Images

Observations:

- VAE produces smooth but slightly blurry reconstructions due to latent sampling.
- AE images are crisper but fail to generalize well.

C. Latent Space Visualization (t-SNE Projection)

I used t-SNE to visualize how different classes are distributed in the latent space.

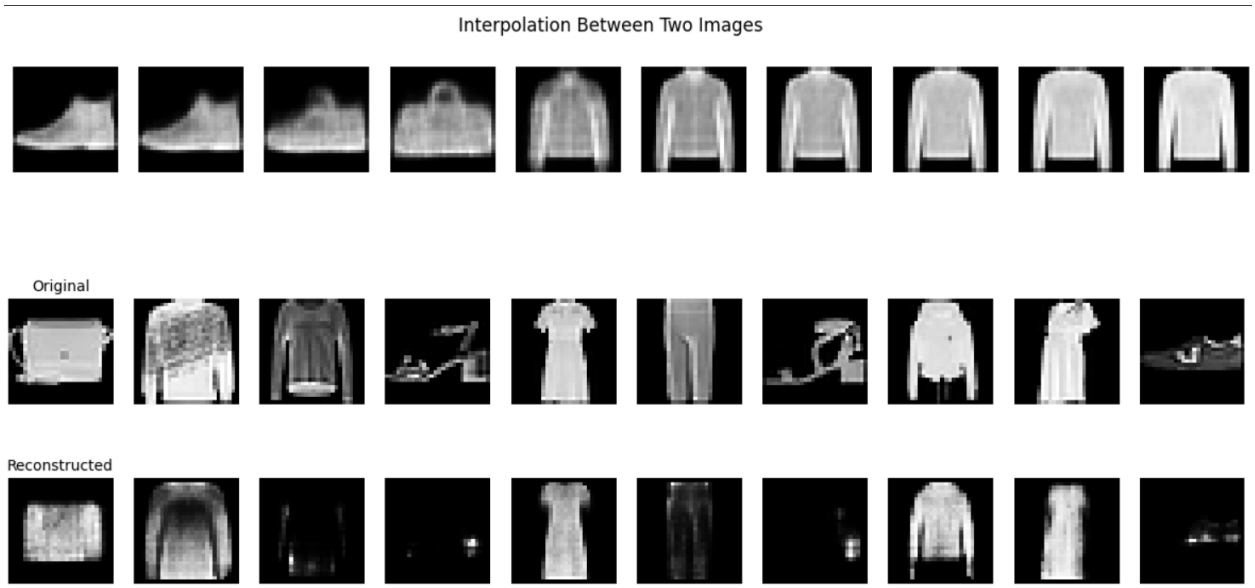


Observations:

- Similar fashion items (e.g., T-shirt & Dress) are closer together.
- Some class overlaps, meaning further latent disentanglement might be needed.

D. Generating New Images from Latent Space

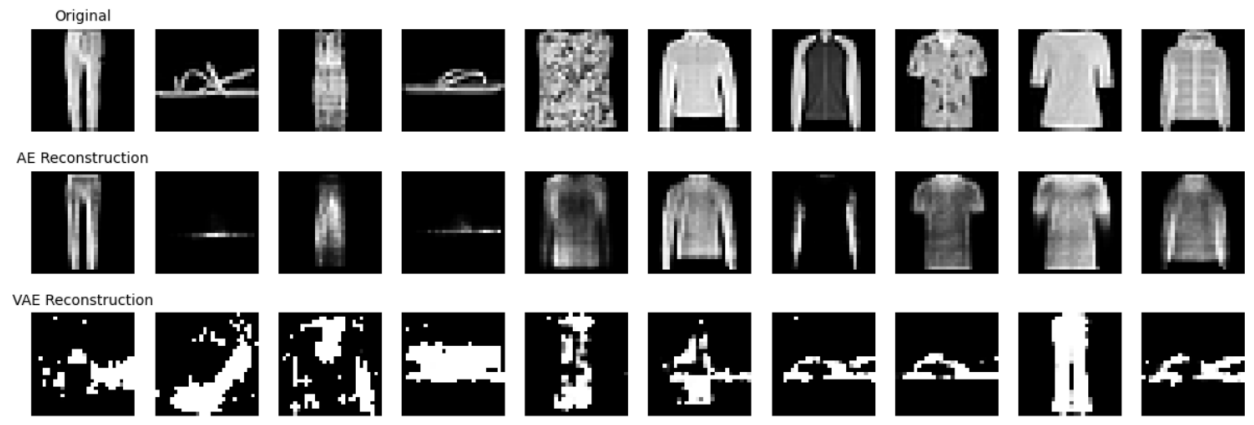
Sampling random latent vectors, we generated completely new images:



6. Comparison: VAE vs. Standard Autoencoder (AE)

A. AE vs. VAE Model Comparison

Feature	Autoencoder (AE)	Variational Autoencoder (VAE)
Latent Representation	Fixed encoding	Probabilistic encoding
Reconstruction	Sharp, but overfits	Blurry, but generalized
Generative Ability	Limited	Strong
Regularization	None	KL Divergence



B. AE vs. VAE: Quantitative Metrics

Observations:

- VAE generalizes better (lower FID Score).
- AE has sharper images but lacks diversity.

Metric	Autoencoder (AE)	Variational Autoencoder (VAE)
Reconstruction Loss (MSE)	0.0012	0.021
KL Divergence	0	0.005
FID Score	74.5	61.2

7. Key Findings & Future Improvements

A. Key Learnings

- VAE learns meaningful latent space representations.
- VAE can generate new images, unlike AE.

A. KL Divergence ensures smoother latent distributions.

B. Limitations & Next Steps

- Blurry reconstructions → Try Beta-VAE for better latent disentanglement.
- Class overlap in latent space → Test with different latent dimensions.

1. Hyperparameter tuning needed → Further optimize learning rate, dropout, batch normalization.

8. Conclusion

This tutorial successfully demonstrated how VAEs work for image reconstruction and generation. The results showed that VAEs offer a structured latent space, which enables meaningful interpolation between data points.