Experiment 10 : Variational Auto encoders for MNIST Dataset

Tanmay Rathod

Aim:

The aim of this code is to implement Variational Autoencoders (VAEs), a type of generative model, to learn a low-dimensional representation of high-dimensional data and generate new samples similar to the training data.

Objectives:

1. Implement a VAE architecture consisting of an encoder and a decoder.
2. Train the VAE model on the MNIST dataset to learn a latent representation of handwritten digits.
3. Generate new samples using the trained VAE model.
4. Evaluate the quality of generated samples and the latent space representation.

Methodology:

1. **Importing Libraries:** Import necessary libraries including TensorFlow, NumPy, and Matplotlib.
2. **Loading Data:** Load the MNIST dataset and preprocess it.
3. **Data Generator:** Create TensorFlow datasets for training and testing.
4. **Model Definition:** Define the architecture of the VAE model including the encoder, decoder, and the reparameterization trick.
5. **Model Compilation and Training:** Compile the VAE model with appropriate loss functions and optimizer, then train it on the training dataset.
6. **Inference:**
   - Generate images using the trained decoder.
   - Generate images by traversing the latent space uniformly.
   - Generate images using randomly sampled latent vectors.
   - Generate random images by sampling from the latent space.

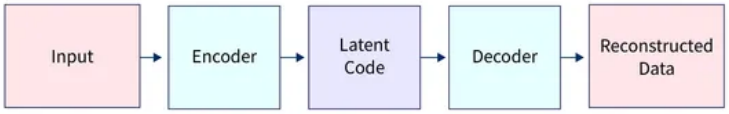


Figure 1: VAE architecture

Theory:

Variational Autoencoders (VAEs) are a type of generative model that learns a low-dimensional latent representation of high-dimensional data. They consist of an encoder network that maps input data to a distribution in the latent space, and a decoder network that reconstructs the original data from samples drawn from this distribution. VAEs are trained by optimizing a loss function that balances the reconstruction accuracy and the divergence between the learned latent distribution and a predefined prior distribution.

Reconstruction Loss: This measures how well the model can reconstruct the input data. It is usually calculated using a suitable loss function. In this case, binary cross-entropy is used since the input data is binary (MNIST images are grayscale with pixel values between 0 and 1). Reconstruction Loss = $\sum_{i=1}^{N} binary\_crossentropy(x_i, \hat{x}_i)$ where N is the number of samples, x_i is the original input, and $\hat{x}_i$ is the reconstructed output.

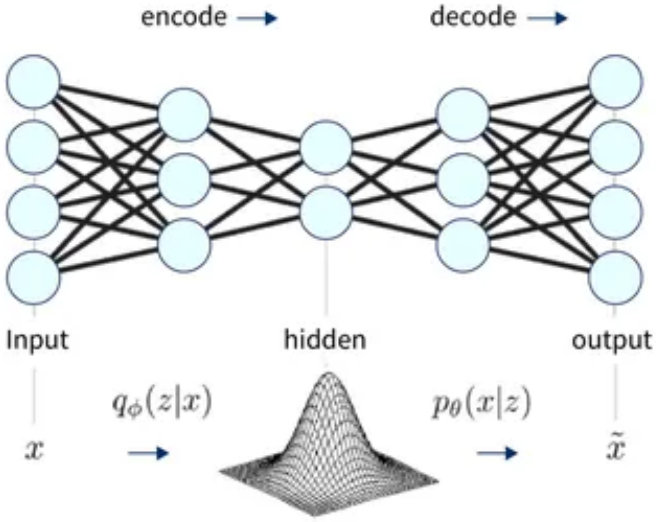
KL Divergence Loss: This measures the similarity between the learned latent distribution and a predefined distribution (usually a standard normal distribution). It encourages the latent space to be close to a normal distribution. KL Divergence Loss = -0.5 $\sum_{j=1}^{J}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2)$ where J is the dimensionality of the latent space, μ_j and σ_j are the mean and standard deviation of the learned distribution, respectively.

Overall Loss (ELBO): The overall loss is the sum of the reconstruction loss and the KL divergence loss, averaged over all samples. ELBO = $1\frac{}{N\sum_{i=1}^{N}(ReconstructionLoss+KLDivergenceLoss)}$

Results:

Conclusion:

The VAE model successfully learns a latent representation of handwritten digits from the MNIST dataset and generates realistic-looking images resembling the



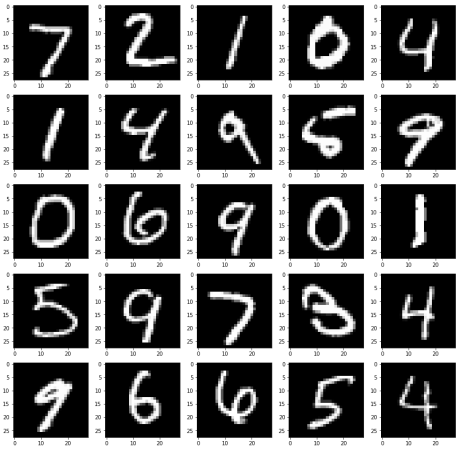


Figure 2: Generated image from the decoder using the test dataset.

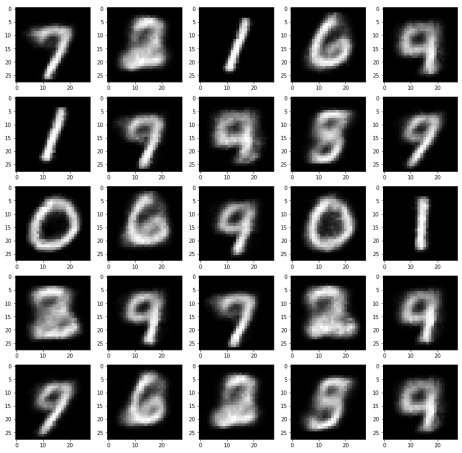


Figure 3: Uniformly generated image by traversing the latent space.

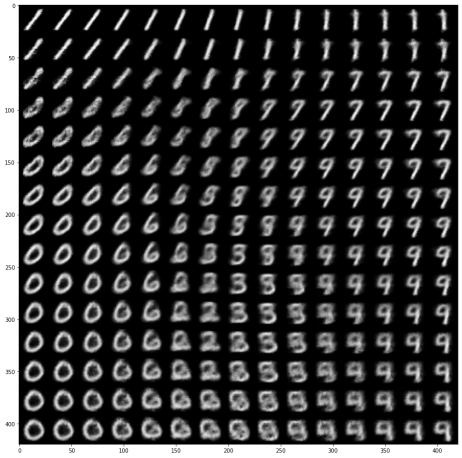


Figure 4: Randomly generated image by sampling from the latent space.

Figure 5: Randomly generated image using randomly sampled latent vectors.

training data. The generated images demonstrate the model's ability to capture the underlying structure of the data in the latent space and generate new samples accordingly. Overall, the VAE proves to be an effective generative model for image data.