DEEP LEARNING

CSL 4020

Assignment **6** Report

Autoencoder

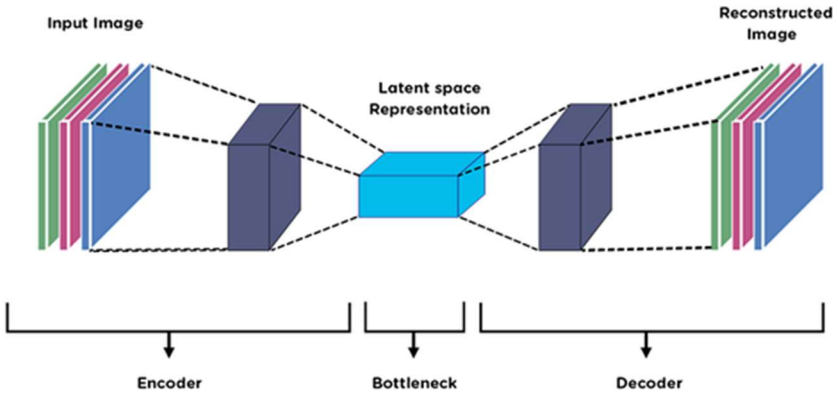
By - PRANAV PANT

B21CS088

Autoencoders

1. Introduction

Autoencoders (AEs) have been widely used for unsupervised learning tasks, such as data compression, feature learning, and denoising. The ability of Autoencoders to learn an efficient representation of the input data in the form of a latent space makes them valuable for various applications. This report focuses on building different types of Autoencoders—simple Autoencoder (AE), Autoencoder with regularization (Dropout), and the goal is to train these models on noisy, colored MNIST data and visualize the latent space using t-SNE.



Auto encoder model architecture

2. Methodology

The primary goal of this project is to build Autoencoders to **denoise noisy images** and visualize the latent space using **t-SNE**. This includes the following:

1. **Data Preparation**: Create noisy data by adding Gaussian noise to the Colored MNIST dataset.
2. **Model Implementation**: Implement a 4-layer Encoder and Decoder for Autoencoder models.
3. **Evaluation**: Pass noisy images through the Autoencoders to produce denoised images and calculate evaluation metrics like **SNR** (Signal-to-Noise Ratio) and **MSE**.
4. **t-SNE Visualization**: Visualize the latent space representation using t-SNE to interpret the performance intuitively.

5. **Regularization**: Apply regularization techniques to improve model generalization.

3. Data Preprocessing

3.1 Colored MNIST Dataset

The Colored MNIST dataset is a variant of the MNIST dataset where each digit is assigned some random colors. It contains grayscale digits, but the color is assigned randomly based on the classes of the digit.

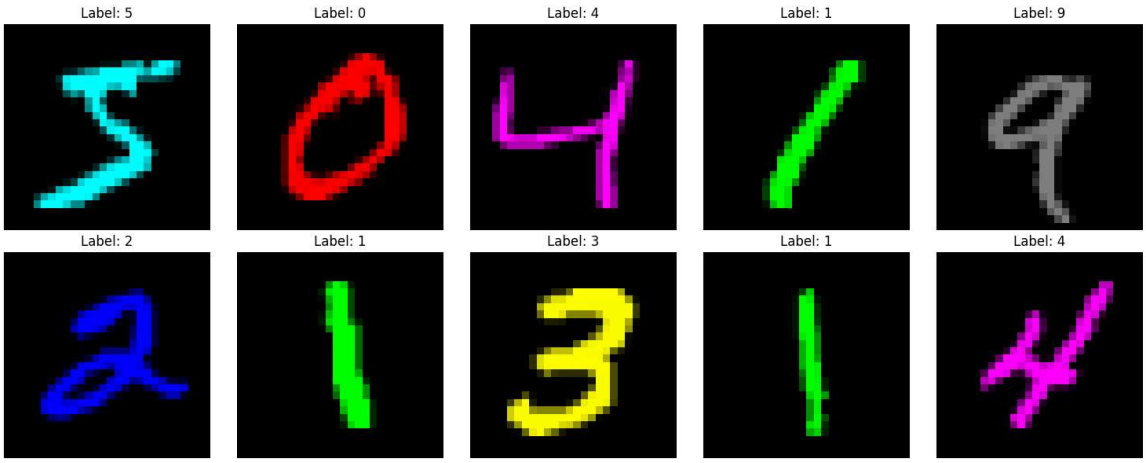


Figure 1: Colored MNIST Dataset

3.2 Adding Gaussian Noise

Gaussian noise is added to the images to simulate a noisy environment and create the denoising task for the Autoencoder.

Sample Images

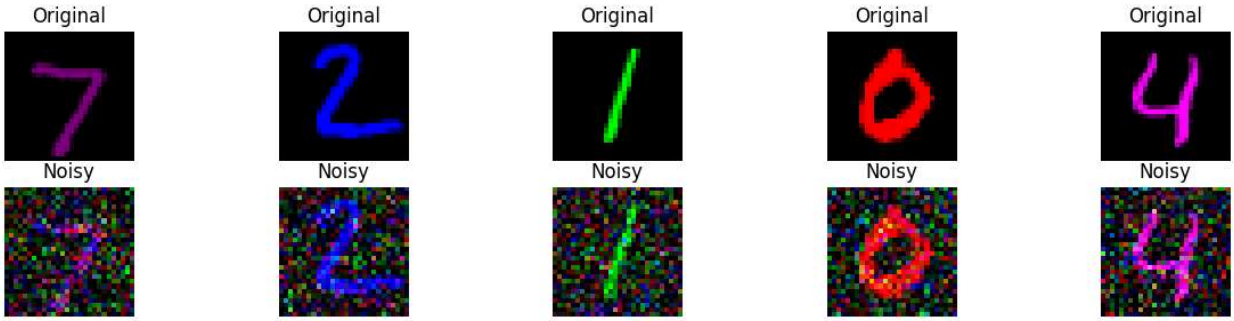


Figure 2: Sample images from the training set (original image) and with gaussian noise.

4. Architecture Design

The architecture for all the Autoencoders follows a symmetrical design with an encoder-decoder structure. The encoder compresses the input data into a latent representation, while the decoder reconstructs the data from this latent space. The key design aspects include the number of layers, types of operations, and dimensional changes. Below are the architectures for each model:

4.1 Simple Autoencoder (AE)

Encoder:

- **4 Convolutional Layers**:
    - The encoder consists of 4 convolutional layers, each progressively reducing the spatial dimensions of the input image.
    - **Filter sizes**: The filters are applied in the following sequence: 128, 64, 32, and 16.
    - **Stride and Padding**: Strides are set to 2, reducing the input image size by half after each layer. Padding is used to maintain the spatial dimensions after convolution.
    - **Activation Function**: Each convolutional layer is followed by a **ReLU activation** to introduce non-linearity, allowing the model to learn complex patterns in the data.
    - **Output of Encoder**: The final output of the encoder is a compressed representation (latent space) with fewer channels and smaller spatial dimensions, typically reducing the image to a **2x2 size** with 16 channels.

Decoder:

- **4 Transpose Convolutional Layers**:
    - The decoder mirrors the encoder, using transpose convolutions to upsample the latent representation back to the original image size.
    - **Filter sizes**: The filter sizes are the same as the encoder, but in reverse order: 16, 32, 64, and 128.
    - **Stride and Padding**: Similar to the encoder, strides are used to increase the spatial dimensions of the image after each layer, and padding ensures the correct output size.
    - **Activation Function**: ReLU is used after each transpose convolutional layer, except the last layer, where a **Sigmoid** activation function is used to ensure that the output pixel values are between 0 and 1.

Purpose:

- The simple AE is designed to learn an efficient and compact latent representation of the input, which can be used to reconstruct the original (or denoised) image.

4.2 Autoencoder with Dropout

This model is an extension of the simple Autoencoder with the addition of Dropout to regularize the model and prevent overfitting. The overall structure is similar, but Dropout is applied after each convolutional and transpose convolutional layer.

4.3 Variational Autoencoder (VAE)

The Variational Autoencoder (VAE) differs significantly from the previous models because it learns a probabilistic distribution in the latent space, rather than a deterministic latent representation. The VAE learns to model the latent space as a multivariate Gaussian distribution characterized by its mean (mu) and variance (log_var). This allows the model to generate new samples by sampling from the learned distribution.

The VAE introduces uncertainty in the latent space, which allows for generating diverse reconstructions of the input data. It's particularly useful for tasks like image generation and unsupervised learning where learning a flexible latent distribution is advantageous.

---

5. Training and Evaluation

5.1 Training Setup

All models were trained with the following setup:

- **Loss Function**: Mean Squared Error (MSE) for simple AE and AE with Dropout; VAE uses a combined **MSE loss + KL divergence**.
- **Optimizer**: Adam optimizer with a learning rate of 1e-3 and weight decay of 1e-5 (for regularization).
- **Batch Size**: 128
- **Epochs**: 20

5.2 Evaluation Metrics

Signal-to-Noise Ratio (SNR)

The **SNR** is used to measure the denoising performance:

$$SNR = 10 \cdot \log_{10} \left(\frac{\text{signal power}}{\text{noise power}} \right)$$

Mean Squared Error (MSE)

The **MSE** is the primary loss function, calculated as:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

5.3 Results

Denoising Performance:

- **Simple Autoencoder**: Achieved moderate denoising with noticeable noise reduction but some loss in sharpness.

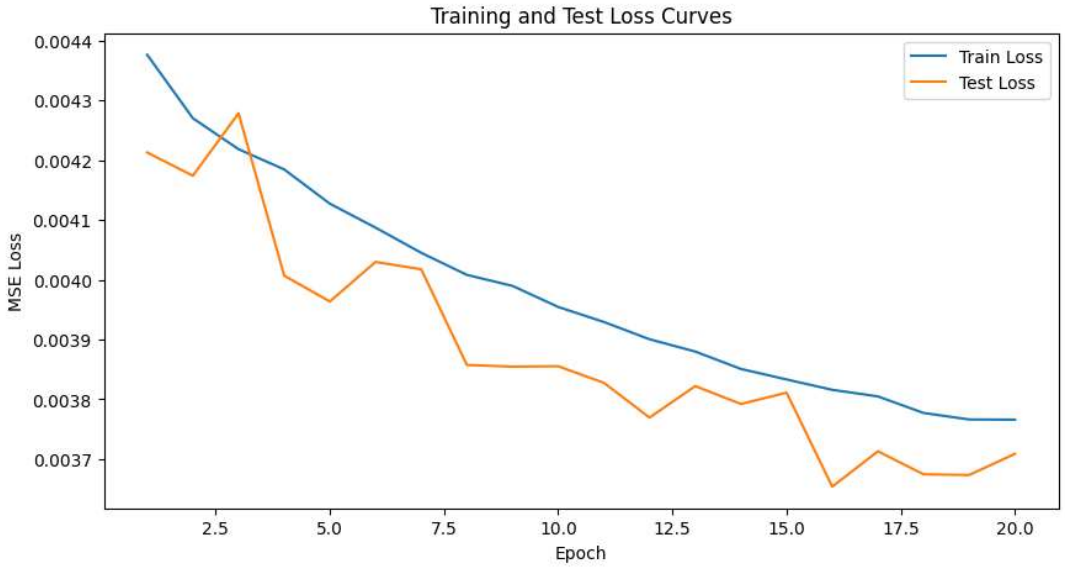


Figure: Training and validation loss over epochs for Autoencoder.

Average SNR on Test Set: 10.27 dB

Average MSE on Test Set: 0.003709

- **Autoencoder with Dropout**: Slightly better generalization and slightly sharper results compared to the simple AE.

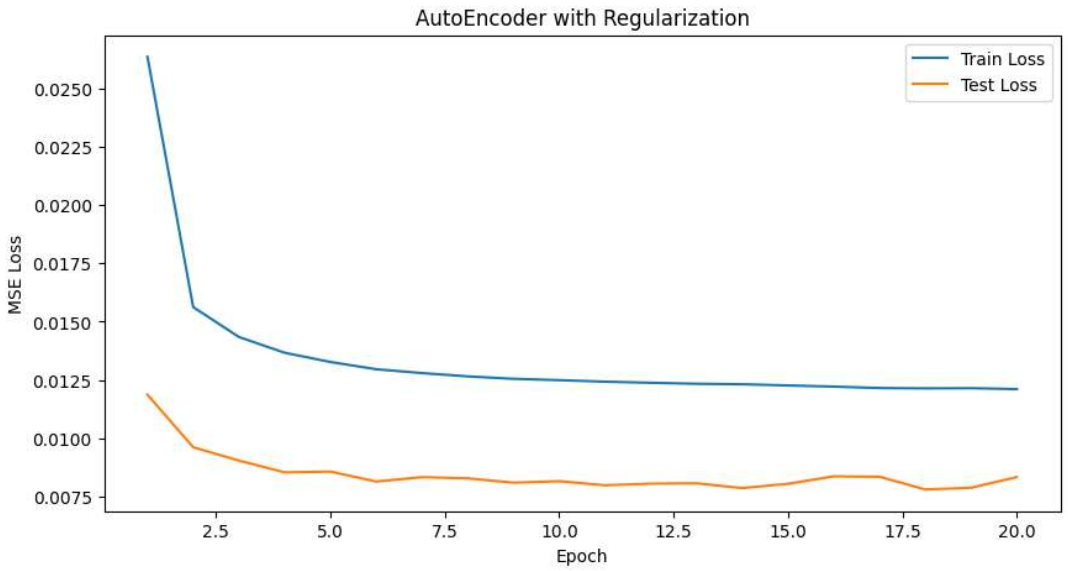


Figure: Training and validation loss over epochs for Autoencoder with Regularization.

Average SNR on Test Set: 7.08 dB

Average MSE on Test Set: 0.008330

- **Variational Autoencoder**: Produced smoother images but sometimes less accurate reconstruction due to the probabilistic nature of VAEs.

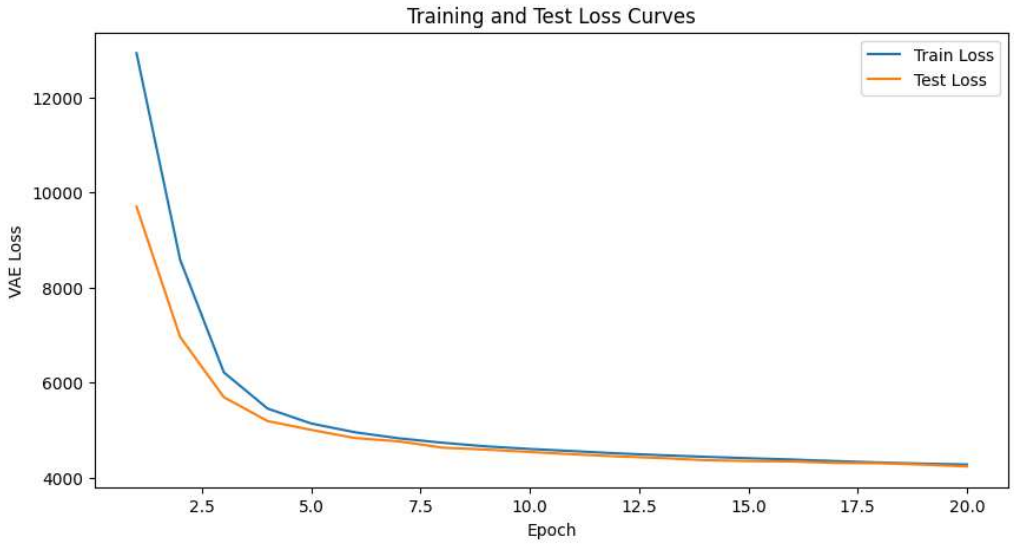


Figure: Training and validation accuracy over epochs for VAE.

5.4 t-SNE Latent Space Visualization

The latent space of the Autoencoder models was visualized using **t-SNE**:

Simple Autoencoder:

- The t-SNE plot shows distinct clusters for different digits, indicating that the Autoencoder learned meaningful features in the latent space.

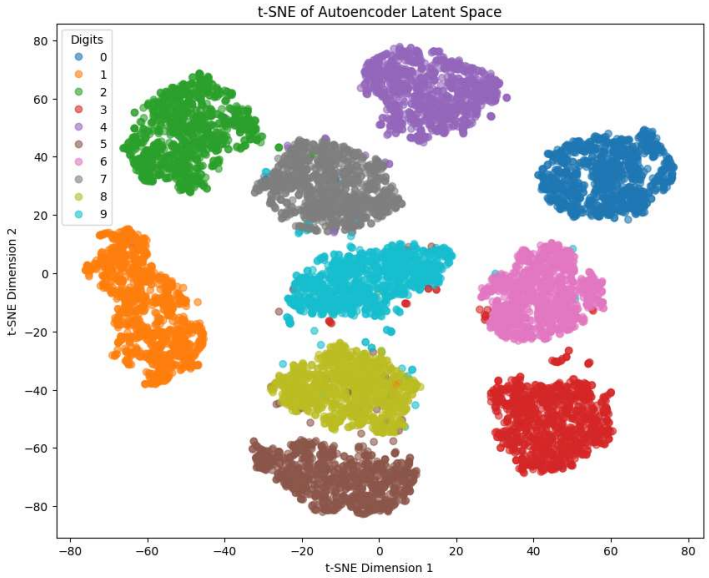


Figure: 2D t-SNE representation of Latent space of Autoencoder.

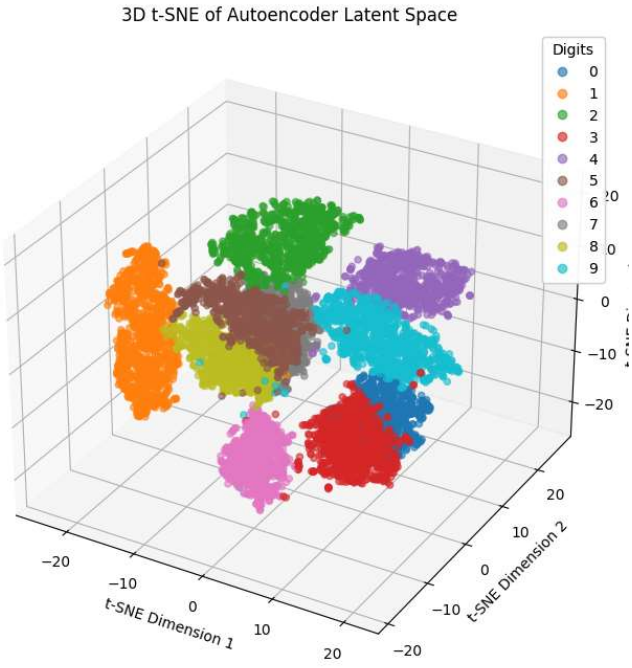


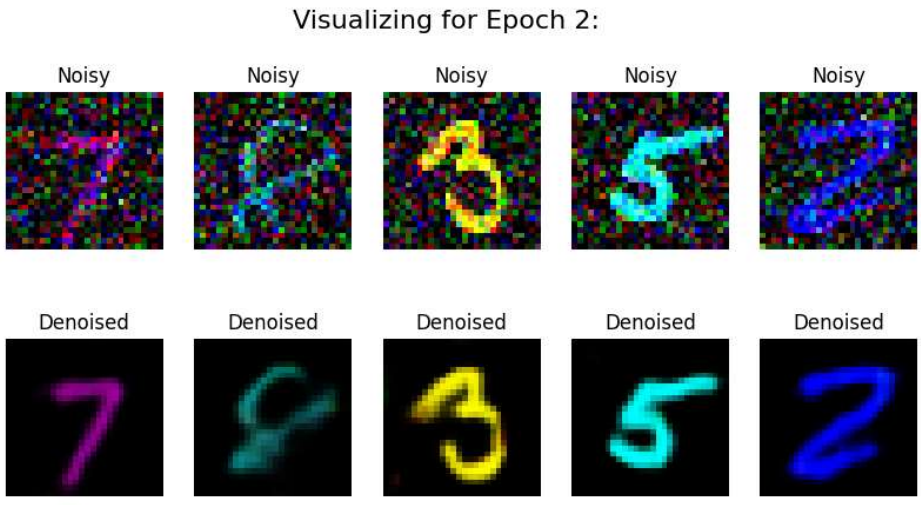
Figure: 3D t-SNE representation of Latent space of Autoencoder.

** Link to hosted interactive plot*: https://assasin1202.github.io/Autoencoder/tsne_3d_plot.html

Visualizing the outputs with training:

While training, I try to get the denoised image after each few epochs to see does the image changes as the training proceeds.

We can see that with more epochs, the outputs get more clearer as the decoder learns to get good outputs mimicking the original image.



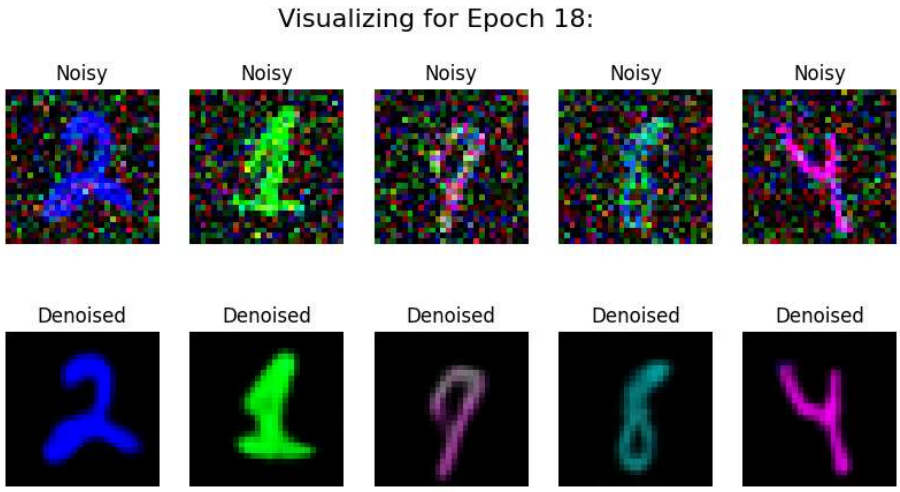


Figure: Visualizing the outputs after every few epochs.

Interpolating Latent space to see prediction changes:

For this, I take 2 image samples of different classes and get their representations in the latent space. Now, in the latent space, we can take points on the plane joining the two original points and get their decoded representations to see how actually they change in predictions.

We can see exactly how the output changes as we move from one class to another. In the middle we have the images which represent some info of both the classes.

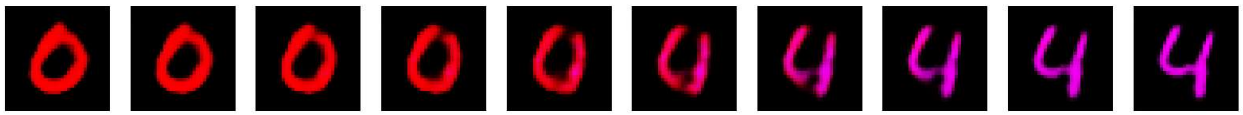


Figure: Interpolating Latent Space and plotting their decoded outputs.

Autoencoder with Dropout:

- The clusters are more spread out but still form distinct groups, showing the effect of dropout on the latent representation.
- We use Dropout and Batch Normalization with our simple Autoencoder and observe the effects.
- Looking at the Latent Space representation, we can see that the model output is closer to 0 because of use of batch normalization techniques.

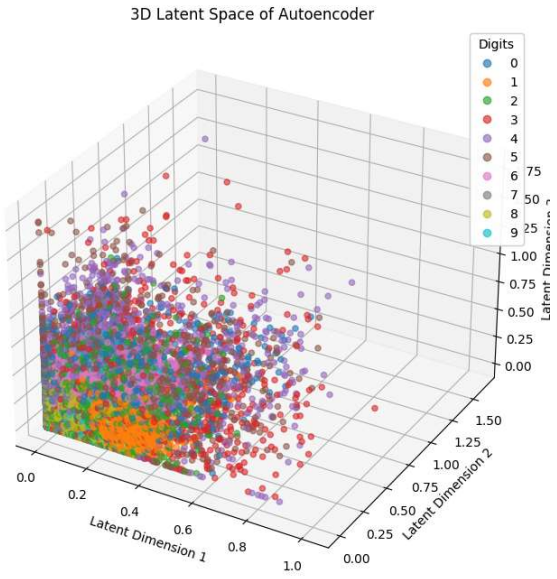


Figure: 3D Latent space of Autoencoder.

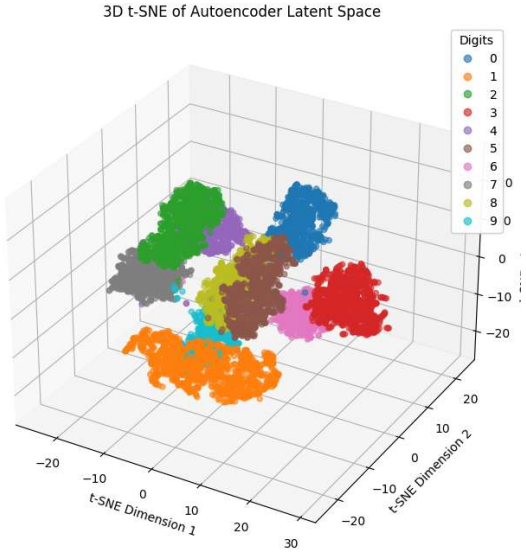


Figure: 3D t-SNE representation of Latent space of Autoencoder.

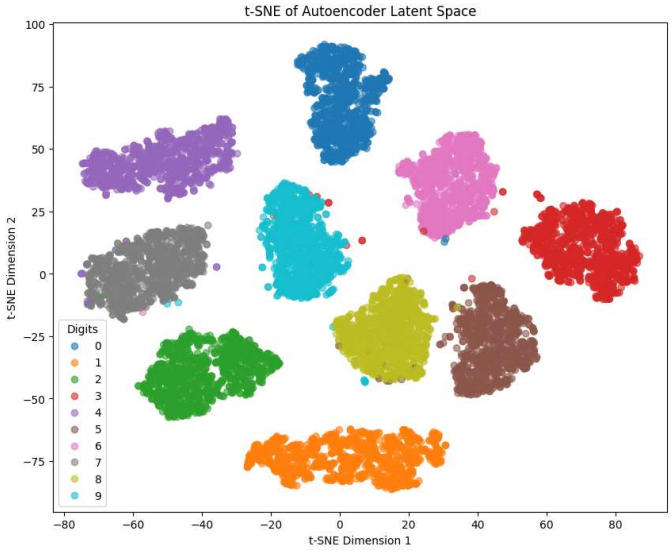
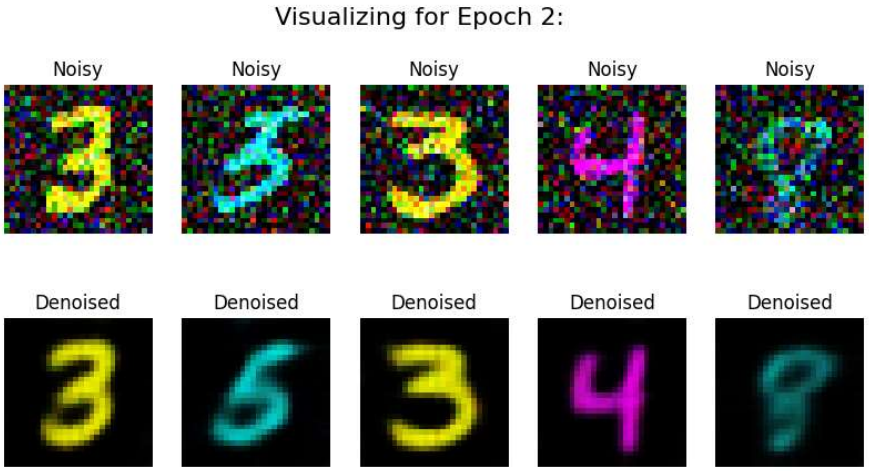


Figure: 2D t-SNE representation of Latent space of Autoencoder.

Visualizing while training:

We can see that the model output is a bit blurrier in case of dropout use as the model becomes more robust to changes. Hence, it takes more time in training due to introduction of dropout layers.



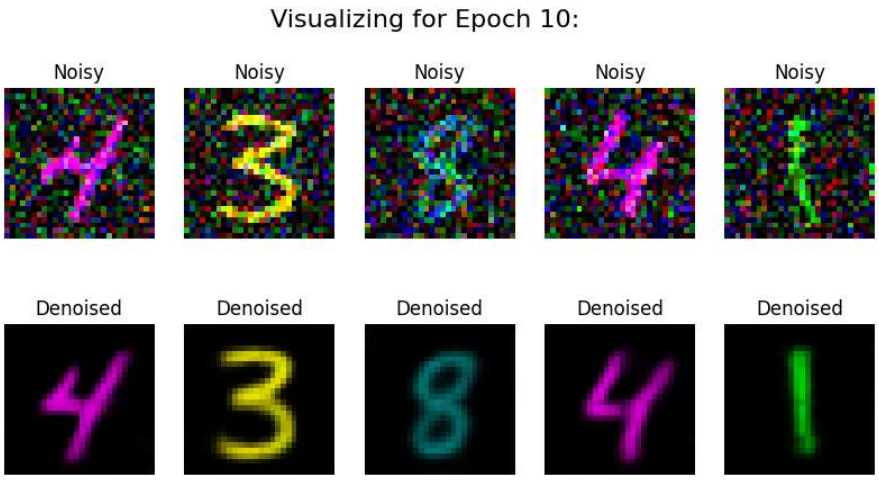


Figure: Visualizing the outputs after every few epochs.

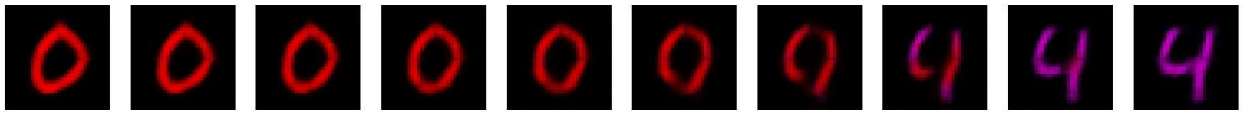
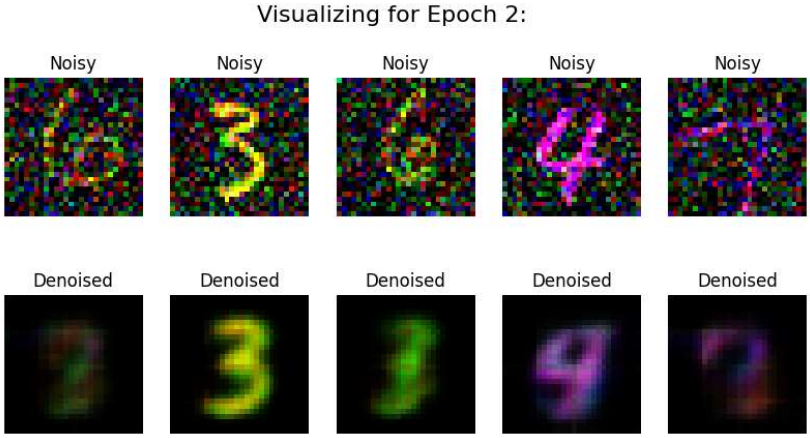


Figure: Interpolating Latent Space and plotting their decoded outputs.

Variational Autoencoder

- The VAE shows more dispersed clusters, indicating a continuous latent space, as expected for VAEs.



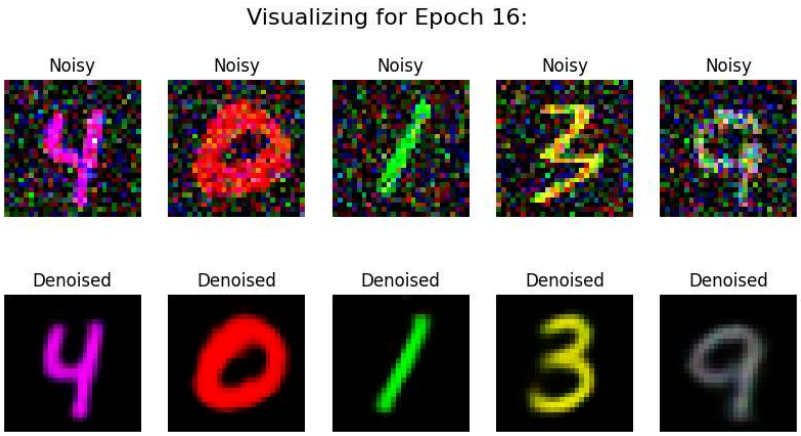


Figure: Visualizing the outputs after every few epochs.

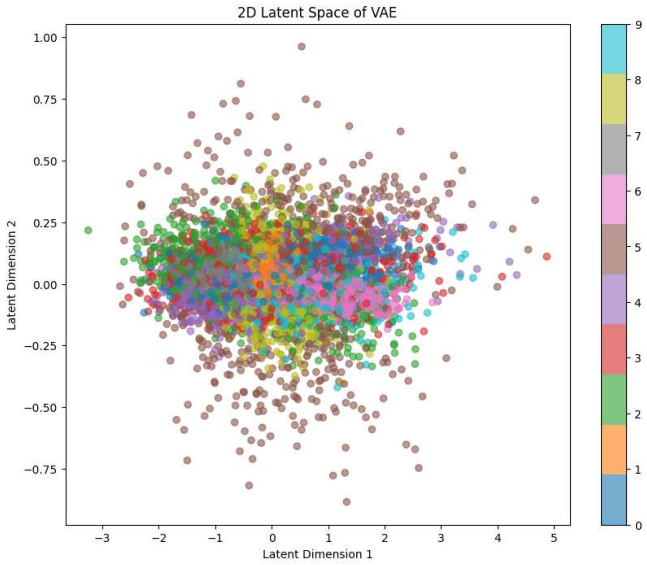


Figure: 2D Latent space of Autoencoder.

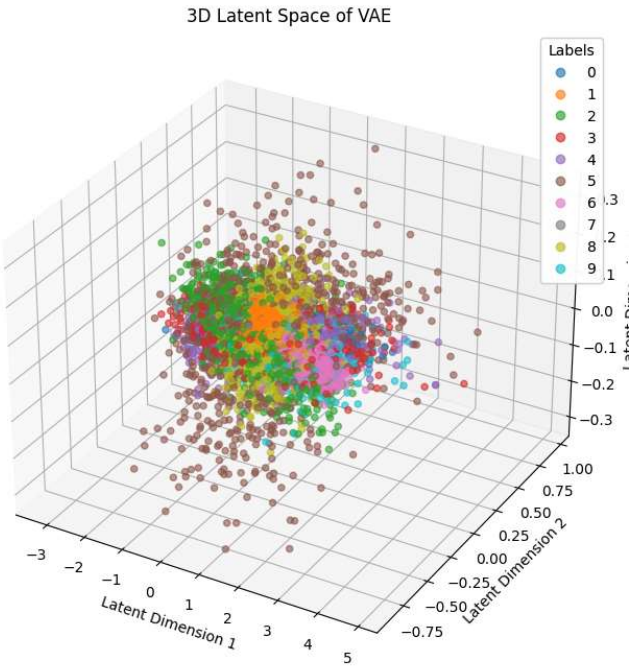


Figure: 3D Latent space of Autoencoder.

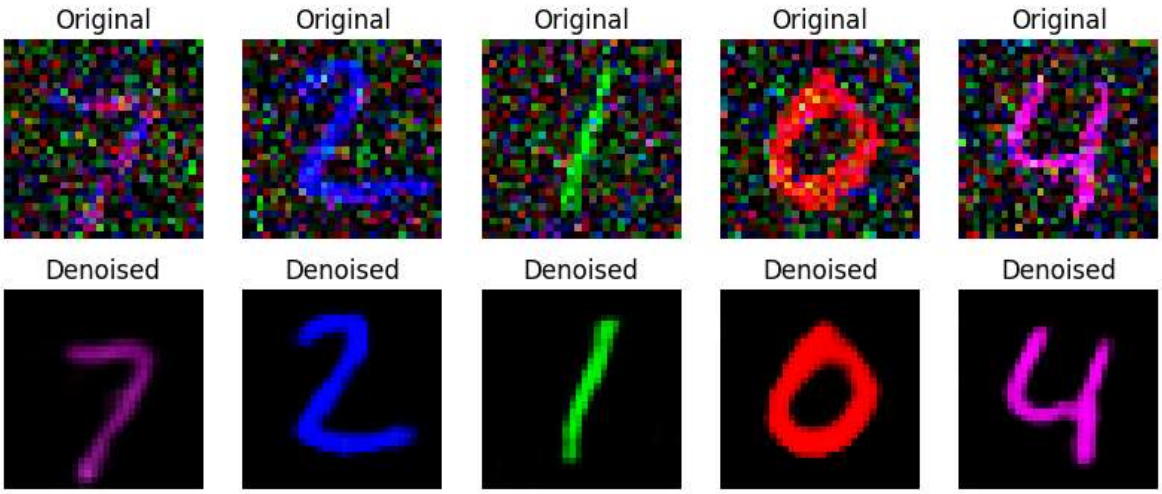


Figure: Testing some images to get denoised Images.

---

8. Conclusion

This report presents the implementation of three different Autoencoder models for denoising colored MNIST data. The models were able to successfully denoise the images and provide meaningful latent space representations. The Autoencoder with regularization (Dropout)

provided better generalization, while the VAE offered a more probabilistic latent space. The t-SNE visualizations helped in understanding the quality of the learned representations.

---------- *Report Ends* ----------