

# **Exploring Image Reconstruction and Colorization using Autoencoders**

**Group No: 6**

**Mentor: Sayanta Adhikari**

## **Team Members**

Sanjana Kankantee

Ch Madhav Kalyan

Nilay Nilesh Jaltare

Anuradha Muppuri

# Highlights of the presentation

Exploring Different kind of Auto encoders

- Linear Autoencoders
- Denoising Autoencoders
- Variational Autoencoders
- 

Image colorization using Auto encoders

- Convolution based Auto encoders
- Skip based Convolutional Auto encoders

# Introduction to Autoencoders(AEs)

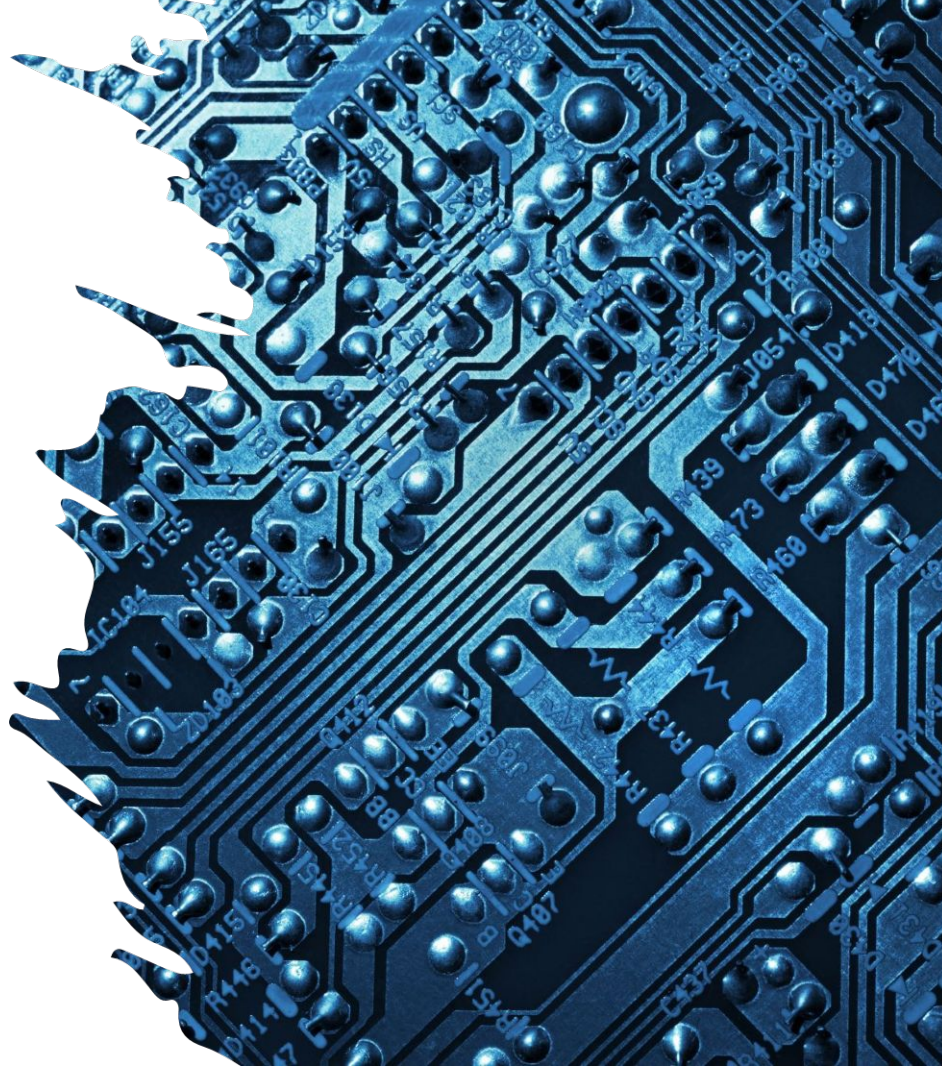
Autoencoders are a type of neural network designed to learn efficient codings of input data.

## **Architecture:**

**Encoder:** Maps input data to a latent-space representation.

**Decoder:** Reconstructs the input data from the latent representation.

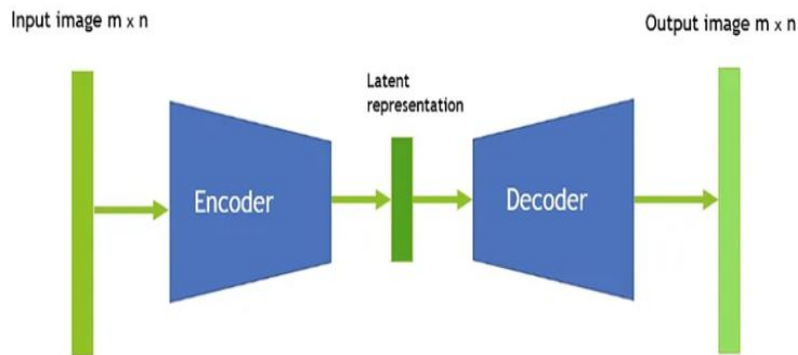
**Purpose:** Dimensionality reduction, feature learning.



# How Autoencoders Work


1. **Input Layer:** Takes in the input data.
2. **Hidden Layers (Encoder):** Compresses the data into a latent-space representation.
3. **Hidden Layers (Decoder):** Reconstructs the original data from the compressed representation.
4. **Output Layer:** Outputs the reconstructed data.


**Loss Function:** Measures the difference between the input and the reconstructed output (e.g., Mean Squared Error).




# Results:


Epoch 7 | Train Loss: 0.0296 | Val Loss: 0.0296  
Epoch: 8/10

100%  938/938


100%  157/157


Epoch 8 | Train Loss: 0.0293 | Val Loss: 0.0295  
Epoch: 9/10

100%  938/938

100%  157/157

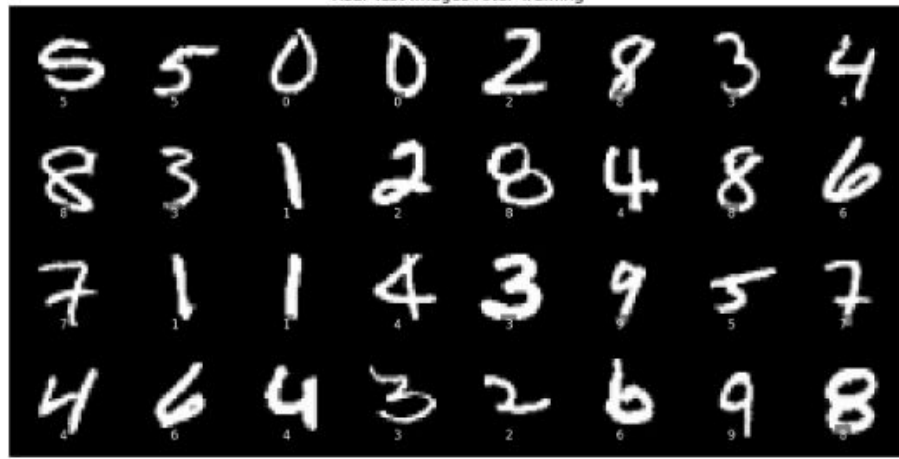
Epoch 9 | Train Loss: 0.0290 | Val Loss: 0.0292  
Epoch: 10/10

100%  938/938

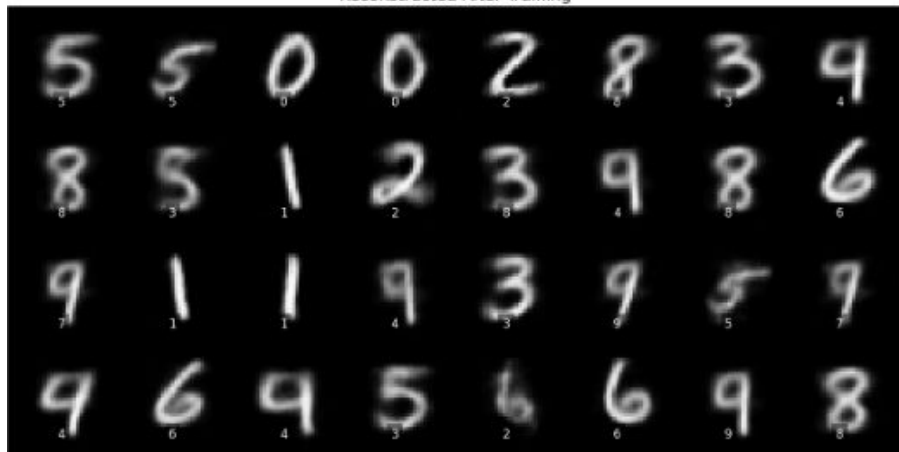
100%  157/157

Epoch 10 | Train Loss: 0.0288 | Val Loss: 0.0290  
Training finished!

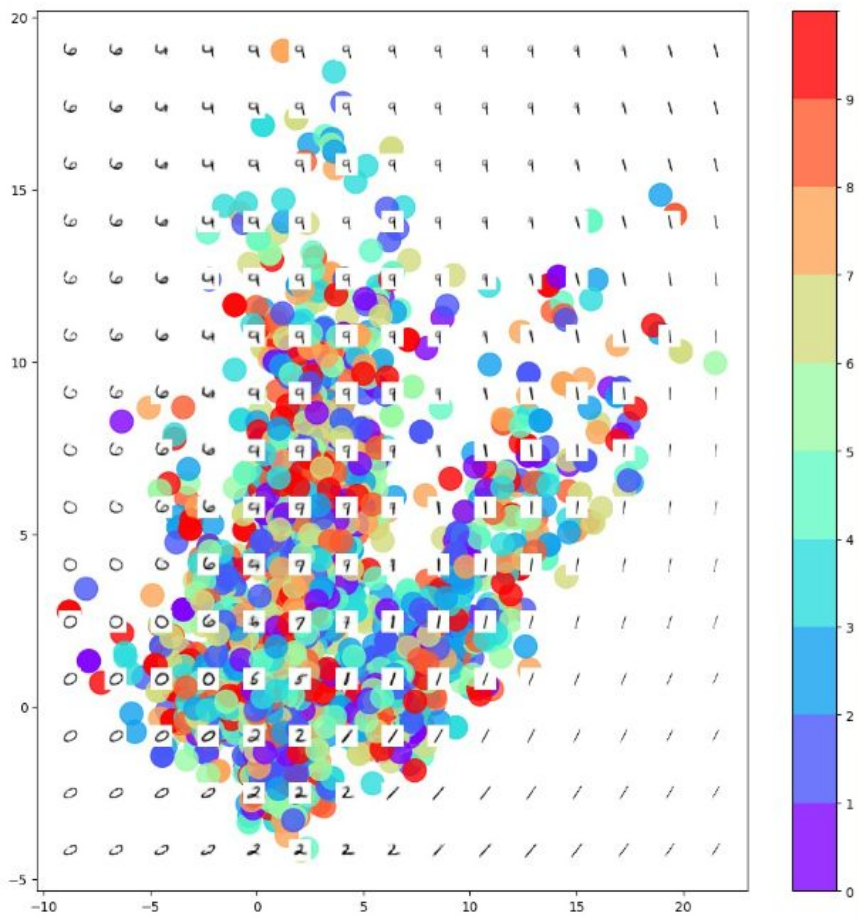
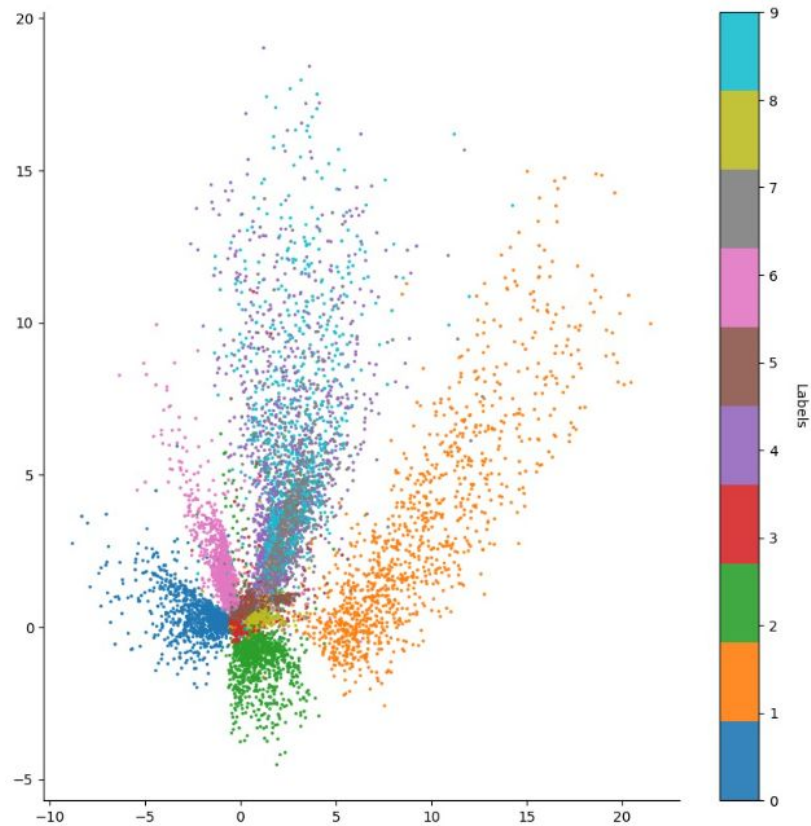
Real Test Images After Training



Reconstructed After Training



# Results:





# Introduction to Denoising Autoencoders (DAEs)

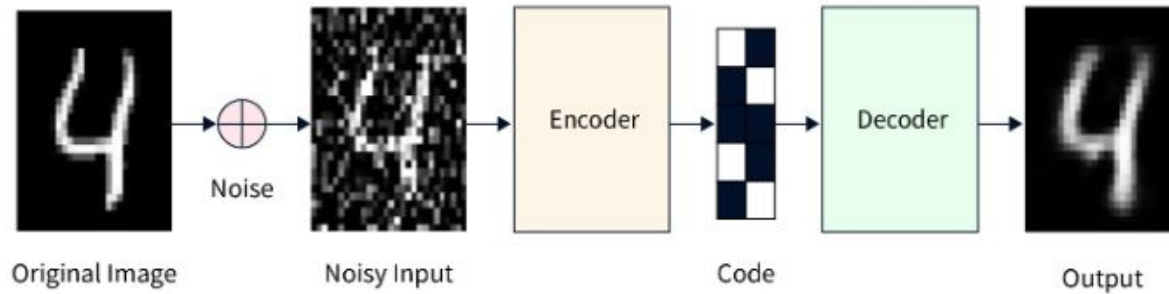
A variant of autoencoders designed to remove noise from input data.

**Key Concept:** The encoder is trained to remove noise by reconstructing the original clean data from noisy input.

**Application:** Image denoising, data preprocessing.



# How Denoising Autoencoders Work

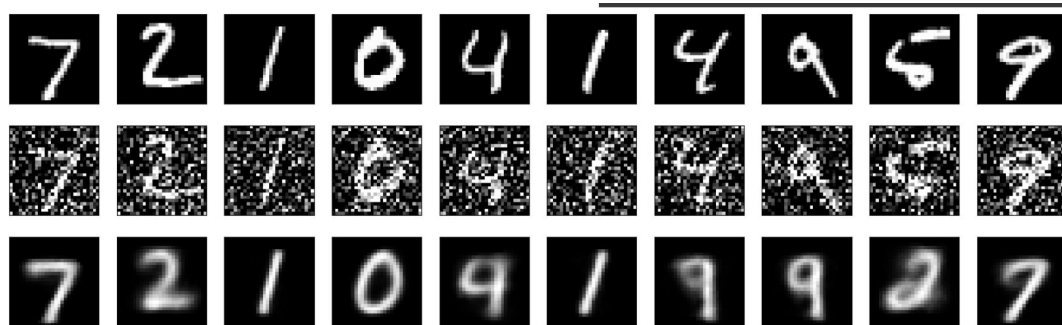


1. **Noisy Input:** Introduce noise to the input data.
2. **Encoder:** Encodes the noisy data to a latent-space representation.
3. **Decoder:** Decodes the latent representation to reconstruct the clean data.

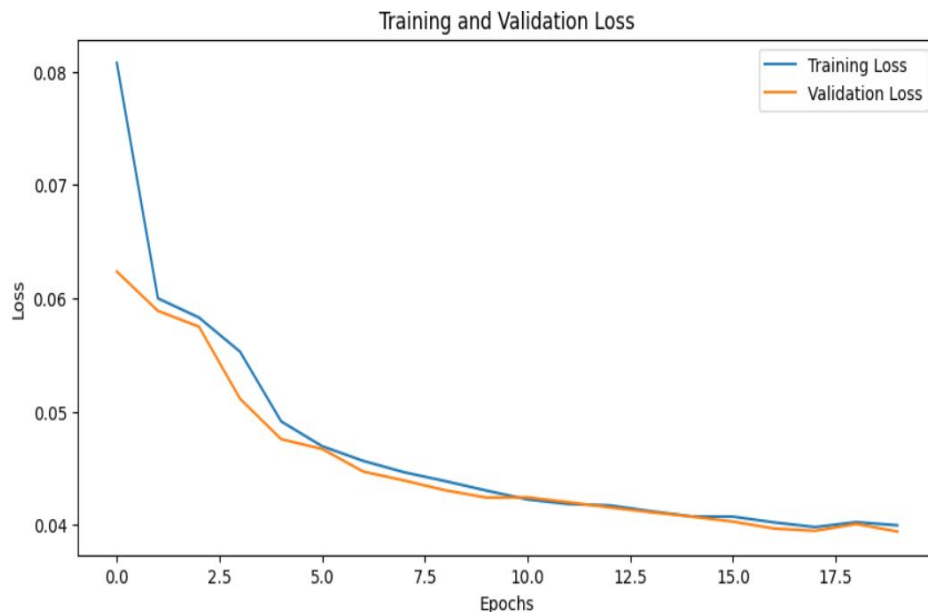
**Loss Function:** Measures the difference between the clean input data and the reconstructed output.



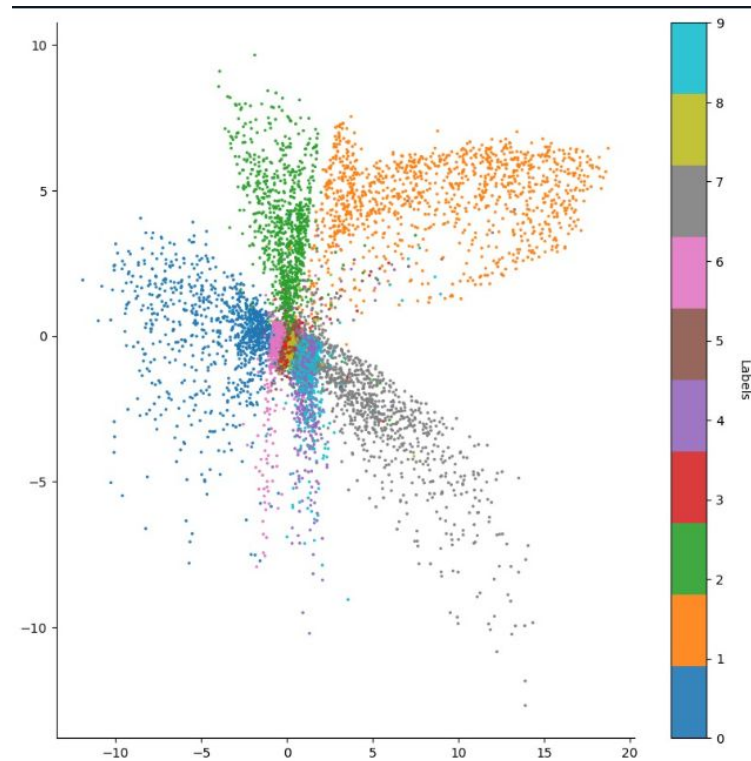
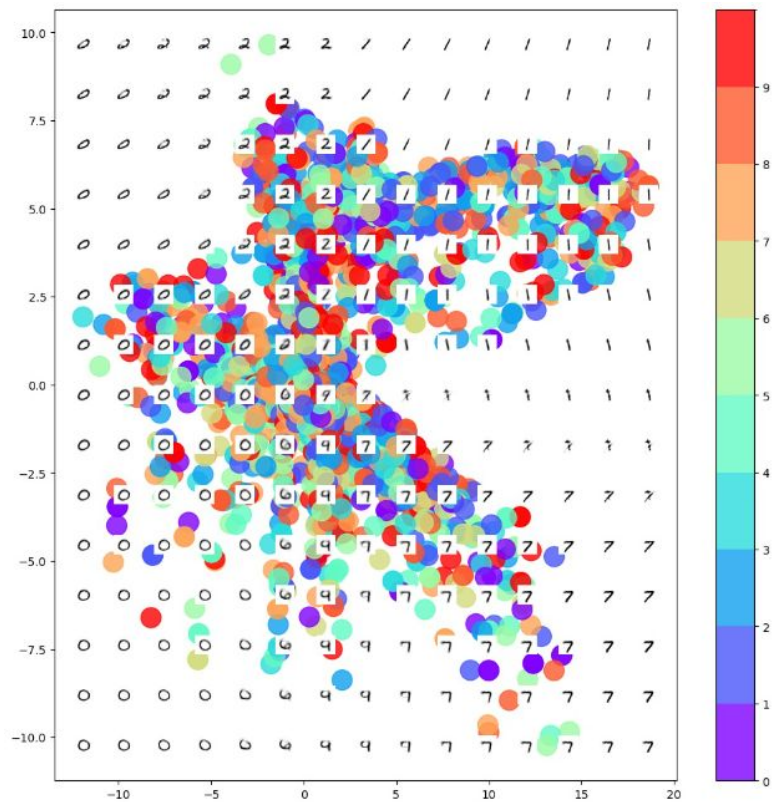
# Results:



```
Epoch 1/20, Train Loss: 0.080765, Val Loss: 0.062357
Epoch 2/20, Train Loss: 0.059998, Val Loss: 0.058893
Epoch 3/20, Train Loss: 0.058302, Val Loss: 0.057496
Epoch 4/20, Train Loss: 0.055288, Val Loss: 0.051135
Epoch 5/20, Train Loss: 0.049140, Val Loss: 0.047571
Epoch 6/20, Train Loss: 0.046947, Val Loss: 0.046704
Epoch 7/20, Train Loss: 0.045652, Val Loss: 0.044708
Epoch 8/20, Train Loss: 0.044647, Val Loss: 0.043918
Epoch 9/20, Train Loss: 0.043867, Val Loss: 0.043060
Epoch 10/20, Train Loss: 0.043031, Val Loss: 0.042414
Epoch 11/20, Train Loss: 0.042249, Val Loss: 0.042437
Epoch 12/20, Train Loss: 0.041846, Val Loss: 0.042003
Epoch 13/20, Train Loss: 0.041728, Val Loss: 0.041549
Epoch 14/20, Train Loss: 0.041214, Val Loss: 0.041126
Epoch 15/20, Train Loss: 0.040753, Val Loss: 0.040731
Epoch 16/20, Train Loss: 0.040735, Val Loss: 0.040293
Epoch 17/20, Train Loss: 0.040226, Val Loss: 0.039688
Epoch 18/20, Train Loss: 0.039816, Val Loss: 0.039485
Epoch 19/20, Train Loss: 0.040251, Val Loss: 0.040091
Epoch 20/20, Train Loss: 0.039972, Val Loss: 0.039422
```



# Results



# Variational Autoencoders (VAEs)

- Variational Autoencoders (VAEs) are a type of generative model in machine learning, particularly used for unsupervised learning. They are a variant of traditional autoencoders with a probabilistic twist, allowing them to generate new data points similar to the input data.

## Applications:

- **Image Generation:** VAEs can generate new, similar-looking images.
- **Data Imputation:** Filling in missing data points in datasets.
- **Anomaly Detection:** Identifying outliers by examining the reconstruction error.
- **Representation Learning:** Learning compact and meaningful representations of data.

# Advantages of VAEs Over Traditional Autoencoders

## 1. Enhanced Feature Learning:

- **VAE** discovers meaningful, disentangled features for better clustering and representation learning, whereas Traditional Autoencoder may learn entangled, less meaningful features.

## 2. Generative Capability:

- **VAE** can generate new, similar data points by sampling from the latent space, while Traditional Autoencoder primarily focuses on reconstructing input data and does not inherently generate new samples.

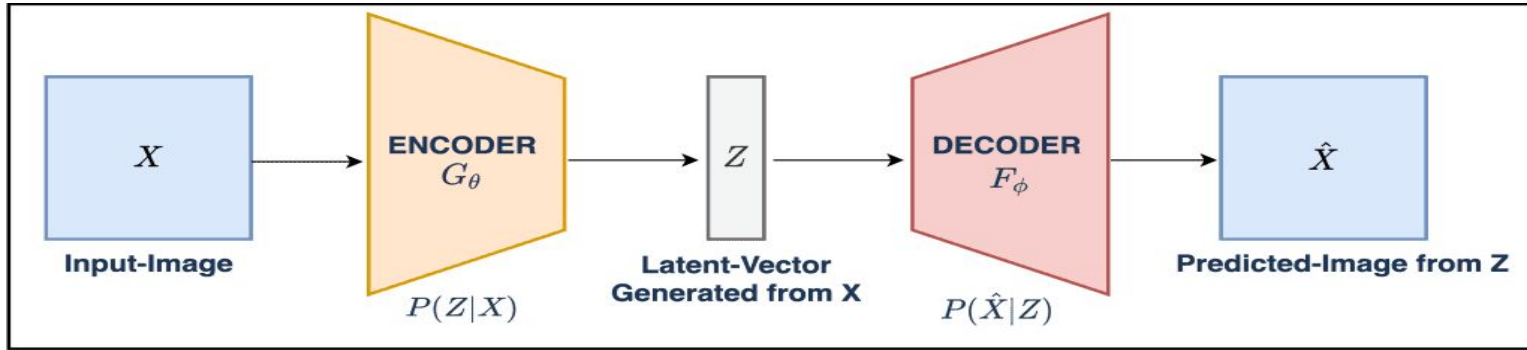
## 3. Anomaly Detection:

- **VAE** is more effective as it models the data distribution and can identify outliers based on reconstruction probability, whereas Traditional Autoencoder relies solely on reconstruction error, which may not capture all types of anomalies.

## 4. Smooth Interpolations:

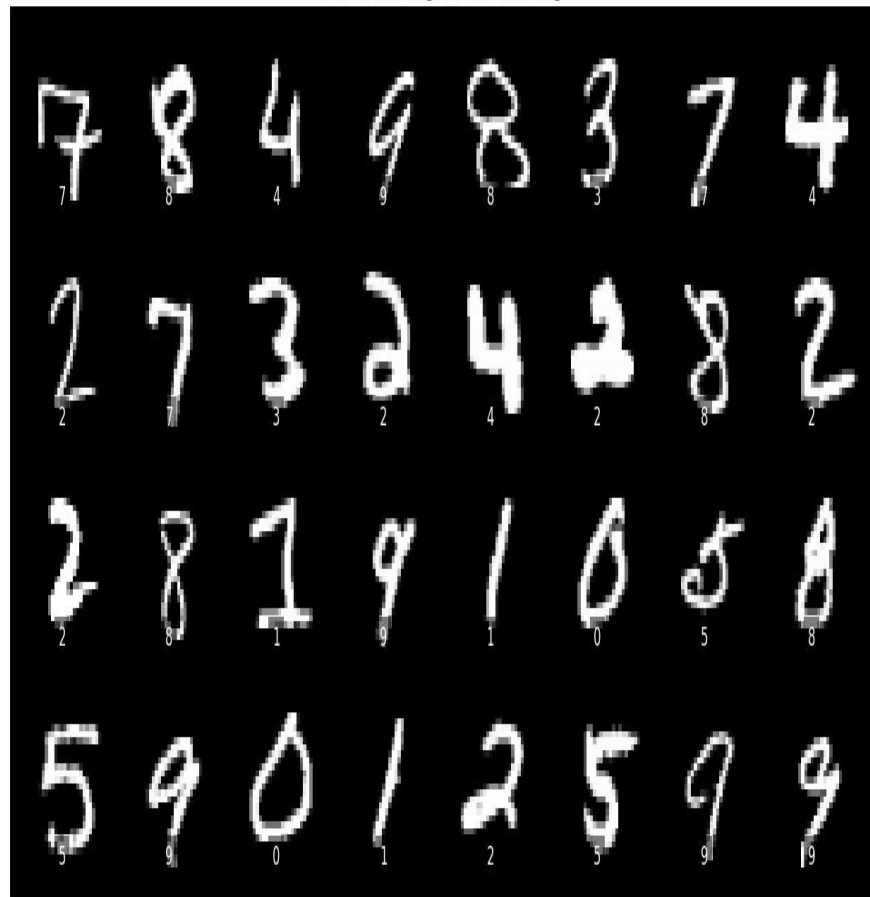
- **VAE** provides smooth interpolations between points in the latent space, enabling meaningful transitions between generated samples, while Traditional Autoencoder may produce abrupt or nonsensical interpolations due to the unstructured latent space.

# Architecture of VAEs

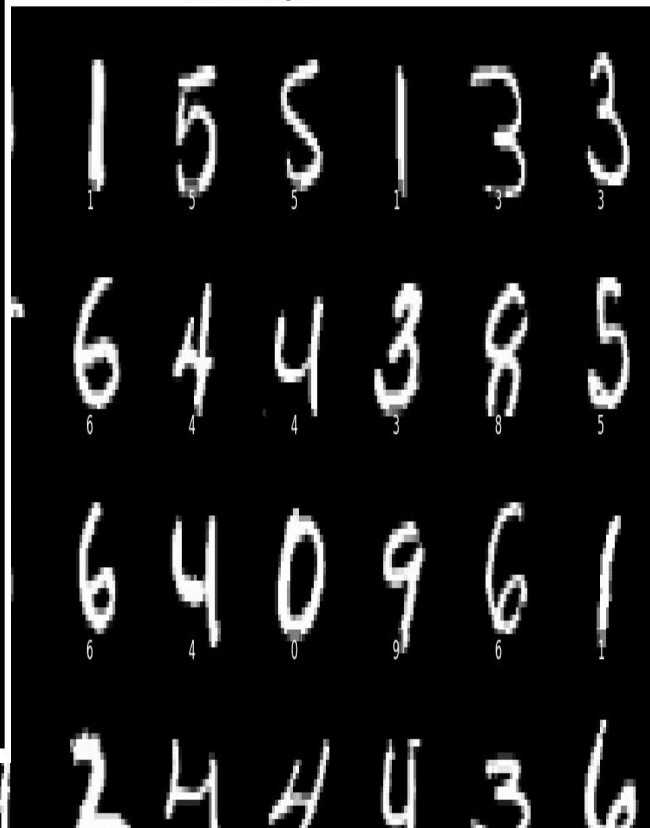


1. Input Layer: Takes the input data  $x$ .
2. Encoder Network: Maps input data  $x$  to a latent space and Produces two outputs: Mean ( $\mu$ ) of the latent variable distribution. and Log variance of the latent variable distribution.
3. Latent Space Sampling: Samples a latent variable  $z$  from the learned distribution.
4. Decoder Network: Maps the sampled latent variable  $z$  back to the data space to reconstruct the input. and Produces the reconstructed data.
5. Output: Combines reconstruction loss and KL divergence loss to train the VAE.

Real Test Images After Training



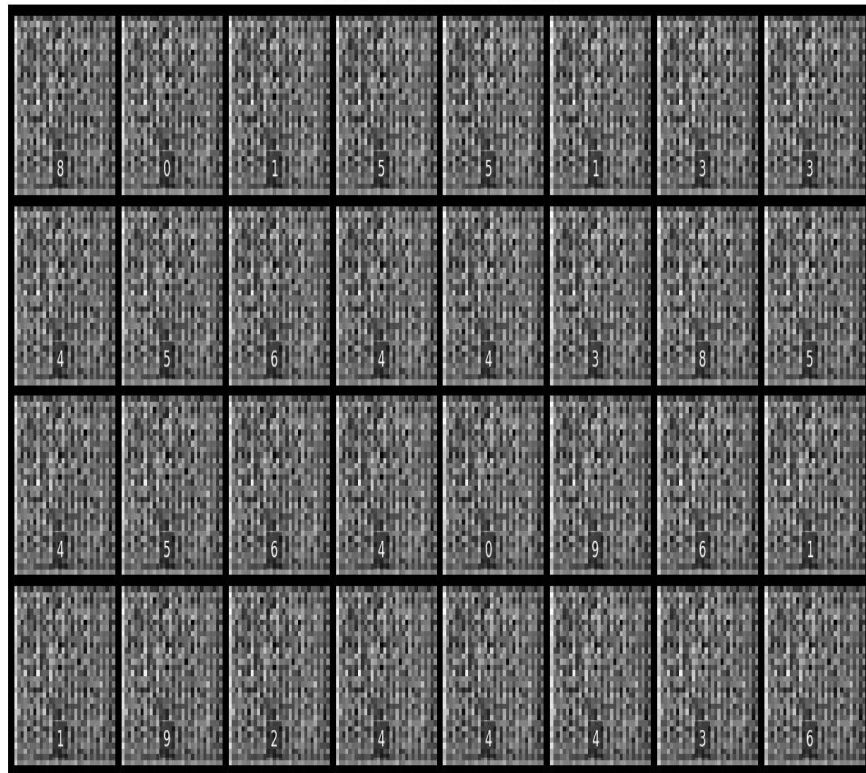
Real Test Images



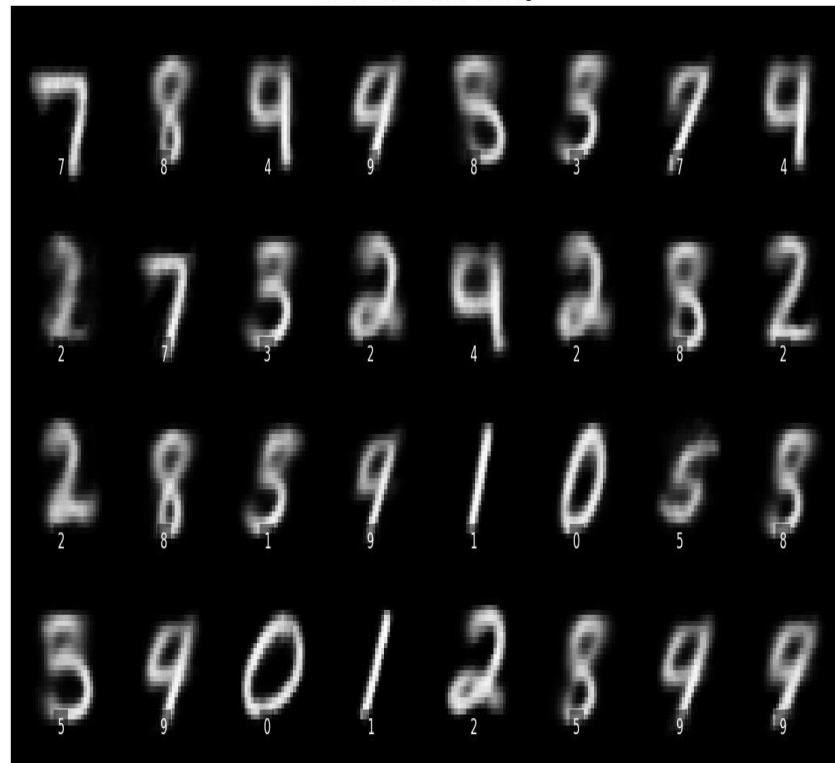


# Outcomes of Reconstructed and Generated Images

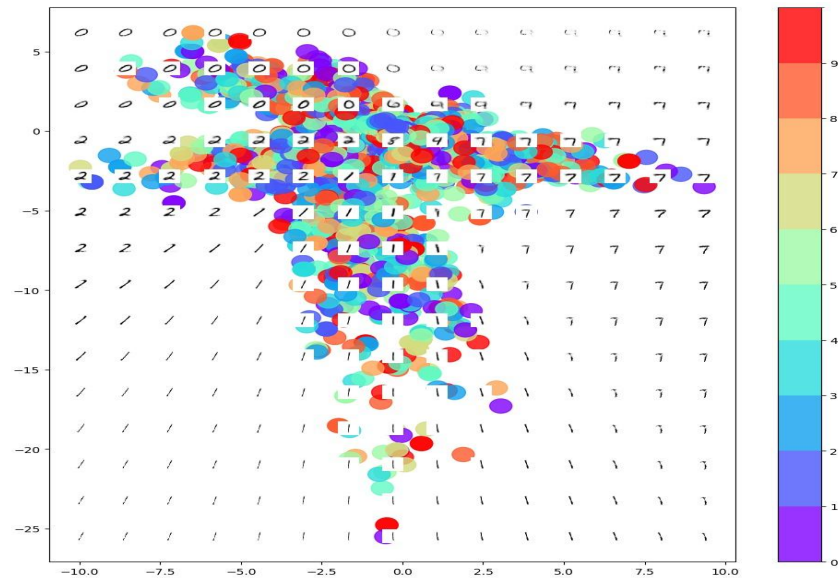
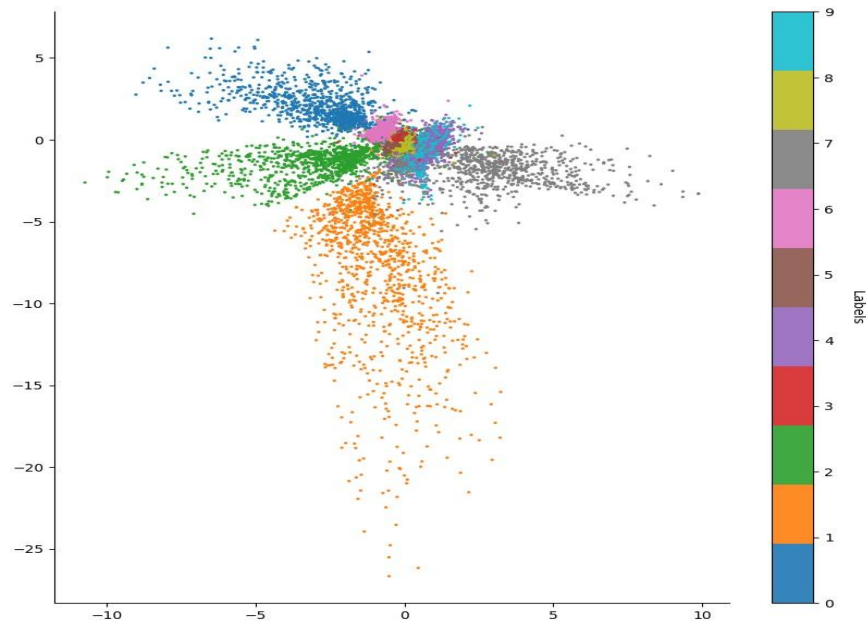
Reconstructed Before Training



Reconstructed After Training

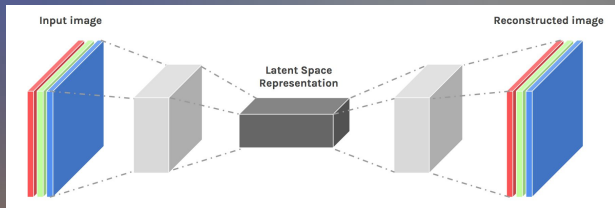


# Results:



# Convolutional Autoencoder for Image Colorization

Harnessing the power of deep learning, this presentation explores a convolutional autoencoder model that can convert grayscale images into vibrant, full-color counterparts. By leveraging the model's ability to learn rich features and patterns, we can breathe new life into monochrome photos, unlocking a world of visual possibilities.



# The Challenge of Image Colorization



1

## Challenging Task

Colorizing grayscale images is a complex task that requires understanding the underlying semantics and structures within an image.

2

## Capturing Colors

Accurately predicting the correct colors for each pixel is a significant challenge, as the model must learn intricate color relationships and patterns.

3

## Real-World Applications

Effective image colorization has numerous applications, such as photo restoration, animation, and content creation.

# Dataset for Training



## Description

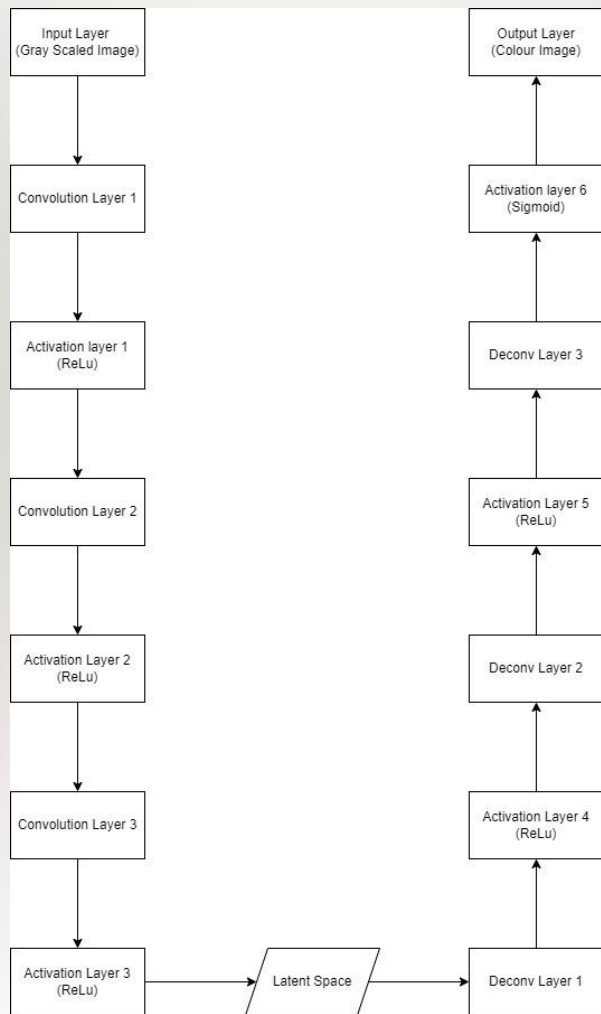
The dataset used for this project consists of 7,129 high-resolution color images, which are then converted to grayscale to serve as the training data.

## Data Splits

The dataset is divided into 80% for training, 10% for validation, and 10% for testing, ensuring a robust evaluation of the model's performance.

## Image Dimensions

All images in the dataset are resized to a consistent size of 150×150 pixels, providing a standardized input for the convolutional autoencoder.



# Convolutional Autoencoder Architecture

1

## Encoder

The encoder component of the model consists of a series of convolutional and pooling layers, which gradually reduce the spatial dimensions of the input while extracting meaningful features.

2

## Bottleneck

The encoded representation is compressed into a low-dimensional latent space, capturing the essential information required for colorization.

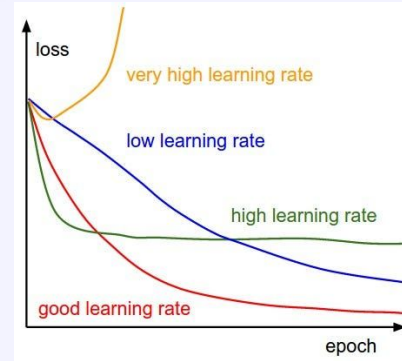
3

## Decoder

The decoder network then takes the latent representation and applies a series of transposed convolutions to gradually reconstruct the color image, restoring the original spatial dimensions.



# Hyperparameter Tuning



## Learning Rate

A learning rate of 0.001 was found to be optimal, allowing for efficient gradient updates during training.

## Batch Size

A batch size of 32 was selected, balancing computational efficiency and effective gradient estimation.

## Epochs

The model was trained for 20 epochs, a duration that provided sufficient time for convergence without overfitting.

## Optimizer

The Adam optimizer was chosen for its adaptive learning rate capabilities and proven performance in deep learning tasks.

# Loss Function: Mean Squared Error

The convolutional autoencoder is trained using the Mean Squared Error (MSE) loss function, which calculates the average squared difference between the predicted color image and the ground truth color image.

The MSE formula is:  $MSE = (1/N) * \sum (y_i - \hat{y}_i)^2$ , where  $y_i$  represents the true color values and  $\hat{y}_i$  represents the predicted color values for each pixel in the N-pixel image.

$$MSE = \frac{1}{n} \sum_{i=1}^n (Y_i - \hat{Y}_i)^2$$

**MSE** = mean squared error

$n$  = number of data points

$Y_i$  = observed values

$\hat{Y}_i$  = predicted values

# Training the Convolutional Autoencoder

## 1

## Forward Pass

The grayscale image is passed through the encoder and decoder networks to generate the predicted color image.

## 2

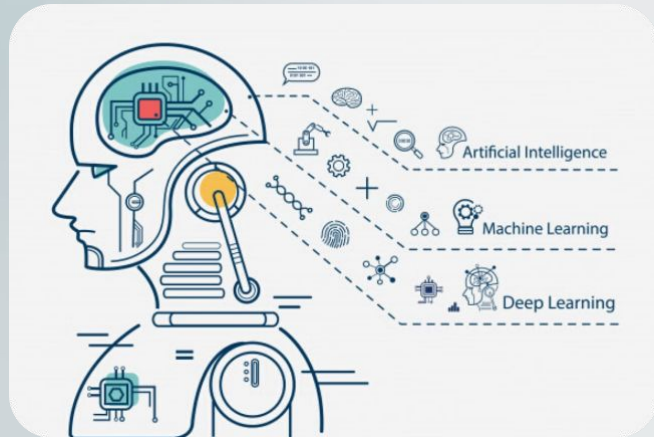
## Loss Calculation

The MSE loss is computed between the predicted color image and the ground truth color image.

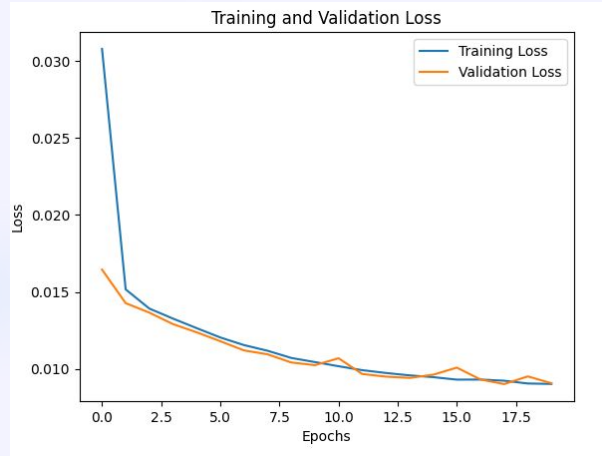
## 3

## Backpropagation

The gradients are calculated, and the model parameters are updated using the Adam optimizer.



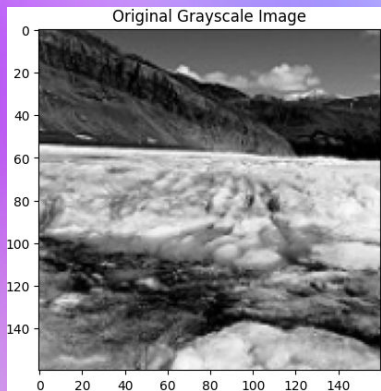
# Training and Validation Loss



## Loss Curve

The plot shows the training and validation loss over the 20 training epochs. The model demonstrates a steady decrease in loss, indicating effective learning.

The model exhibits good generalization, as the validation loss closely follows the training loss, suggesting the absence of significant overfitting.



# Colorization Results



## Original

Grayscale input image



## Prediction

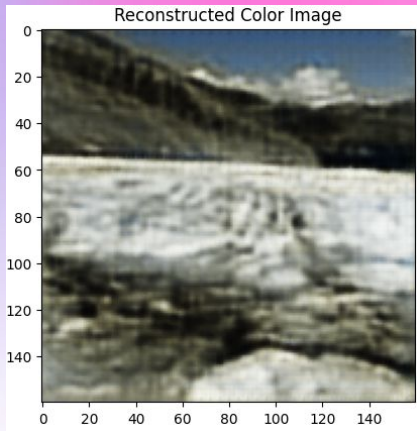
Colorized image generated by the convolutional autoencoder



## Ground Truth

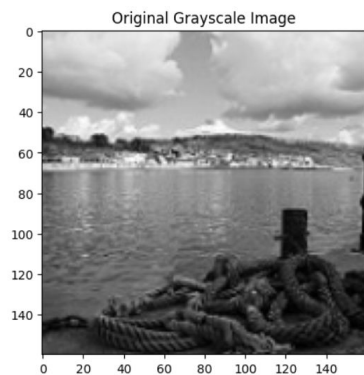
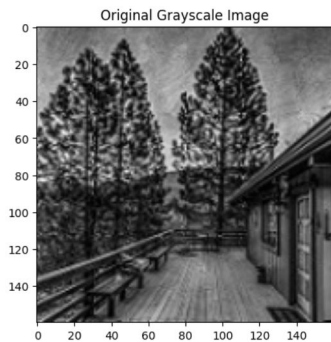
Corresponding color image used as the ground truth

The convolutional autoencoder demonstrates impressive colorization capabilities, accurately predicting vibrant colors that closely match the ground truth images.

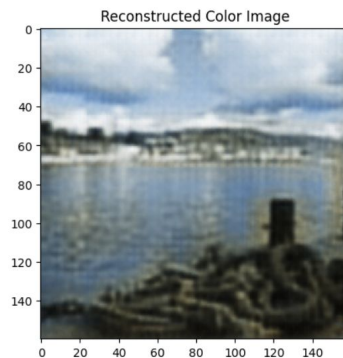
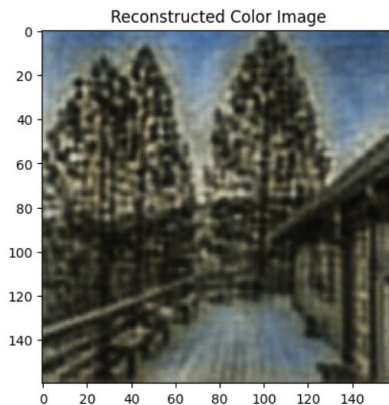


# Predicted Coloured Images using Convolutional Autoencoder

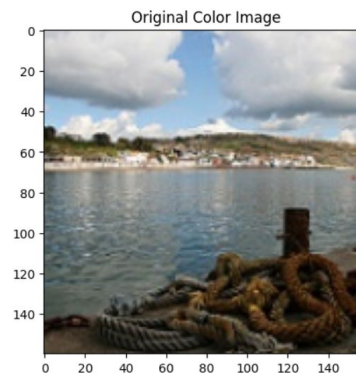
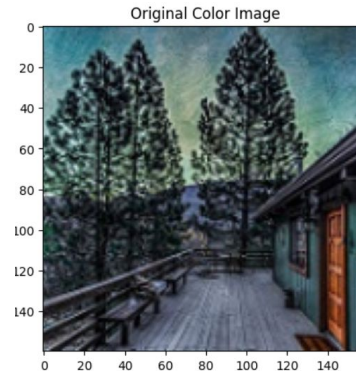
## Gray Scale Images



## Predicted Image



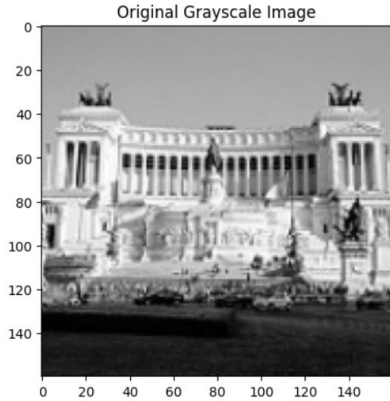
## Original Colour Image



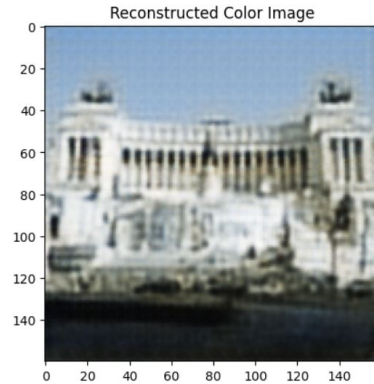


# Predicted Coloured Images using Convolutional Autoencoder

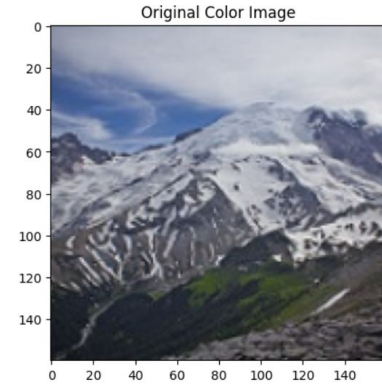
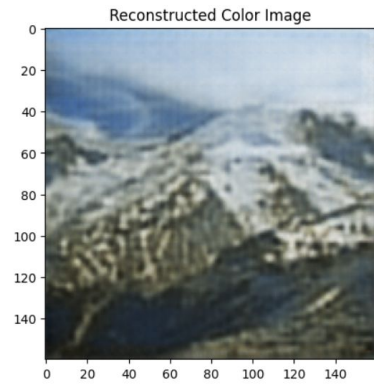
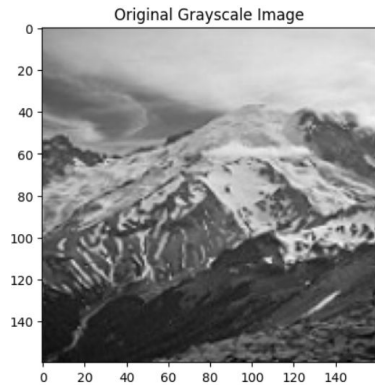
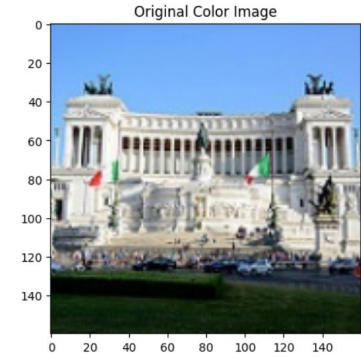
## Gray Scale Images



## Predicted Image

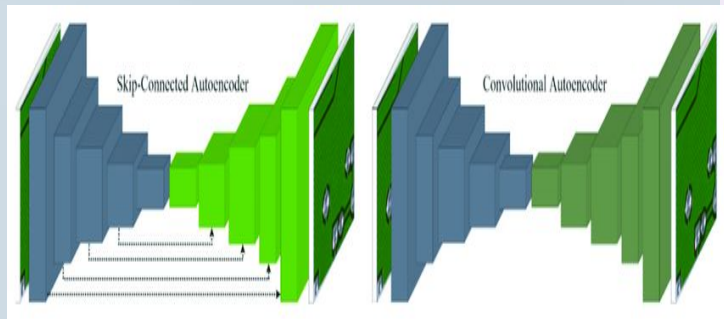


## Original Colour Image



# Skip-based Autoencoders: Enhancing Representation Learning

Skip-based autoencoders are a variant of the traditional autoencoder model, which aims to learn efficient representations of input data. By incorporating skip connections it allows the network to better preserve and leverage information from earlier layers, leading to improved reconstruction and representation learning capabilities.



# Visualization of Skip-based Autoencoders

## Encoder

The encoder network maps the input data to a lower-dimensional latent representation, capturing the essential features and patterns in the input.

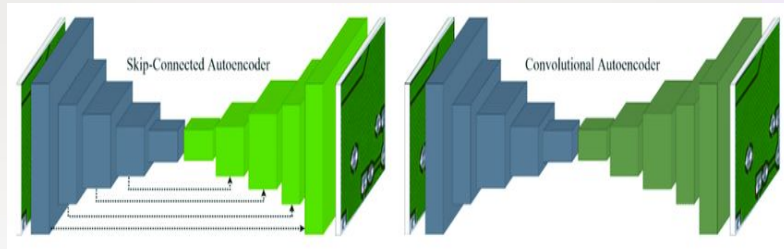
## Skip Connections

The skip connections facilitate direct information flow from earlier layers to later layers, enabling the network to effectively combine low-level and high-level features.

## Decoder

The decoder network reconstructs the original input from the compressed latent representation, leveraging the information preserved through the skip connections.

## Reconstruction



# Designed Model

```
class ColorAutoEncoder(nn.Module):
    def __init__(self):
        super().__init__()
        # Input shape: Bx1x150x150
        # Conv2d(in_channels, out_channels, kernel_size, stride, padding)

        self.down1 = nn.Conv2d(1, 64, 3, stride=2) # Bx64x74x74
        self.down2 = nn.Conv2d(64, 128, 3, stride=2, padding=1) # Bx128x37x37
        self.down3 = nn.Conv2d(128, 256, 3, stride=2, padding=1) # Bx256x19x29
        self.down4 = nn.Conv2d(256, 512, 3, stride=2, padding=1) # Bx512x10x10

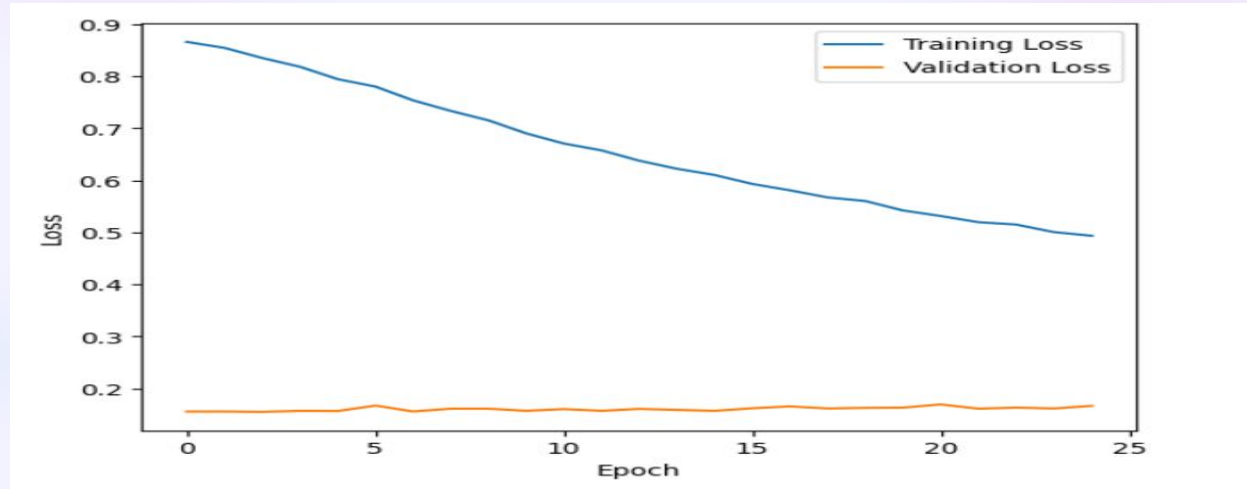
        self.up1 = nn.ConvTranspose2d(512, 256, 3, stride=2, padding=1)
        self.up2 = nn.ConvTranspose2d(512, 128, 3, stride=2, padding=1)
        self.up3 = nn.ConvTranspose2d(256, 64, 3, stride=2, padding=1, output_padding=1)
        self.up4 = nn.ConvTranspose2d(128, 3, 3, stride=2, output_padding=1)

        self.relu = nn.ReLU()
        self.sigmoid = nn.Sigmoid()

    def forward(self, x):
        # Down sample
        d1 = self.relu(self.down1(x))
        d2 = self.relu(self.down2(d1))
        d3 = self.relu(self.down3(d2))
        d4 = self.relu(self.down4(d3))
        # Upsample
        u1 = self.relu(self.up1(d4))
        u2 = self.relu(self.up2(torch.cat((u1, d3), dim=1)))
        u3 = self.relu(self.up3(torch.cat((u2, d2), dim=1)))
        u4 = self.sigmoid(self.up4(torch.cat((u3, d1), dim=1)))

        return u4
```

# Training and Validation Loss



## Loss Curve

The plot shows the training and validation loss over the 25 training epochs. The model demonstrates a steady decrease in loss, indicating effective learning.

The model exhibits good generalization, as the validation loss closely follows the training loss, suggesting the absence of significant overfitting.

# Predicted Colored images using Skip Based AutoEncoders

Gray Scale Image



Color Image



Predicted Image





Grayscale



Color



Predicted



# Conclusion

Auto encoders architectures capable of learning efficient representations of data by compressing inputs into a lower-dimensional latent space and reconstructing them back. They find wide application in tasks like data denoising, dimensionality reduction, image colorization and image generation.

Thank You