Wafer Defect Generation

Part 2: Using VAE and Comparison with CycleGAN

1. Steps required to run the code:

# Import Libraries:

* + Import the required libraries and modules, including PyTorch, torchvision, PIL (Pillow), os, glob, and matplotlib.

# Define Encoder and Decoder Classes:

* + Define the Encoder and Decoder classes, which constitute the two main components of the Variational Autoencoder (VAE).

# Define VAE Class:

* + Define the VAE class, which combines the encoder and decoder. It also includes a reparameterization function for sampling from the learned distribution.

# Load Dataset Function:

* + Define the load\_dataset function to load and preprocess the dataset using torchvision's ImageFolder and apply transformations such as resizing and converting to tensor.

# VAE Loss Function:

* + Define the vae\_loss function to compute the loss for training the VAE. It includes both Mean Squared Error (MSE) loss and the Kullback-Leibler (KL) Divergence loss.

# Training Function:

* + Define the train\_vae function to train the VAE model using the specified dataloader, optimizer, number of epochs, and save interval. It prints and plots the training loss and saves the model at specified intervals.

# Set Device and Initialize Model:

* + Set the device (GPU or CPU) based on availability. Initialize the VAE model and the Adam optimizer.

# Load Training Dataset:

* + Load the training dataset using the load\_dataset function.

# Train the VAE:

* + Call the train\_vae function with the VAE model, training dataloader, optimizer, and other parameters to train the model.

# Generate Images Function:

* + Define the generate\_images function to generate new images using the trained VAE model. It loads the model, creates a folder for each model's generated images, generates random samples, and saves the images.

# Specify Output Folder and Model Folder:

* + Specify the output folder where generated images will be saved (output\_folder) and the folder containing saved VAE models (models\_folder).

# Generate Images for Each Model:

* + Iterate through the saved VAE models (based on their filenames) in the specified model folder, and generate images for each model using the generate\_images function.

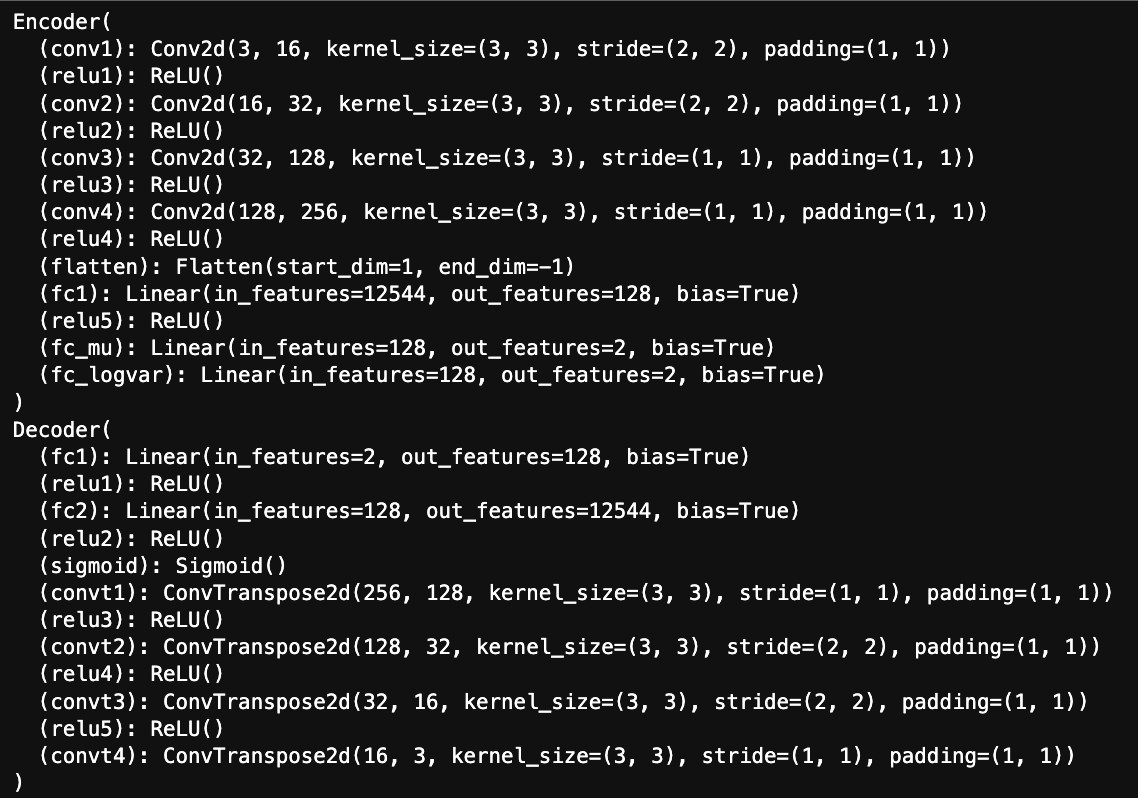
# Run the Code:

* + Run the entire script, ensuring that the required datasets are available in the specified paths, and the necessary folders for saving models and generated images exist.

# Check Output:

* + After running the code, check the output folder to find the generated images for each model and the training loss plot.

1. Screenshot of Encode and Decoder model Summary

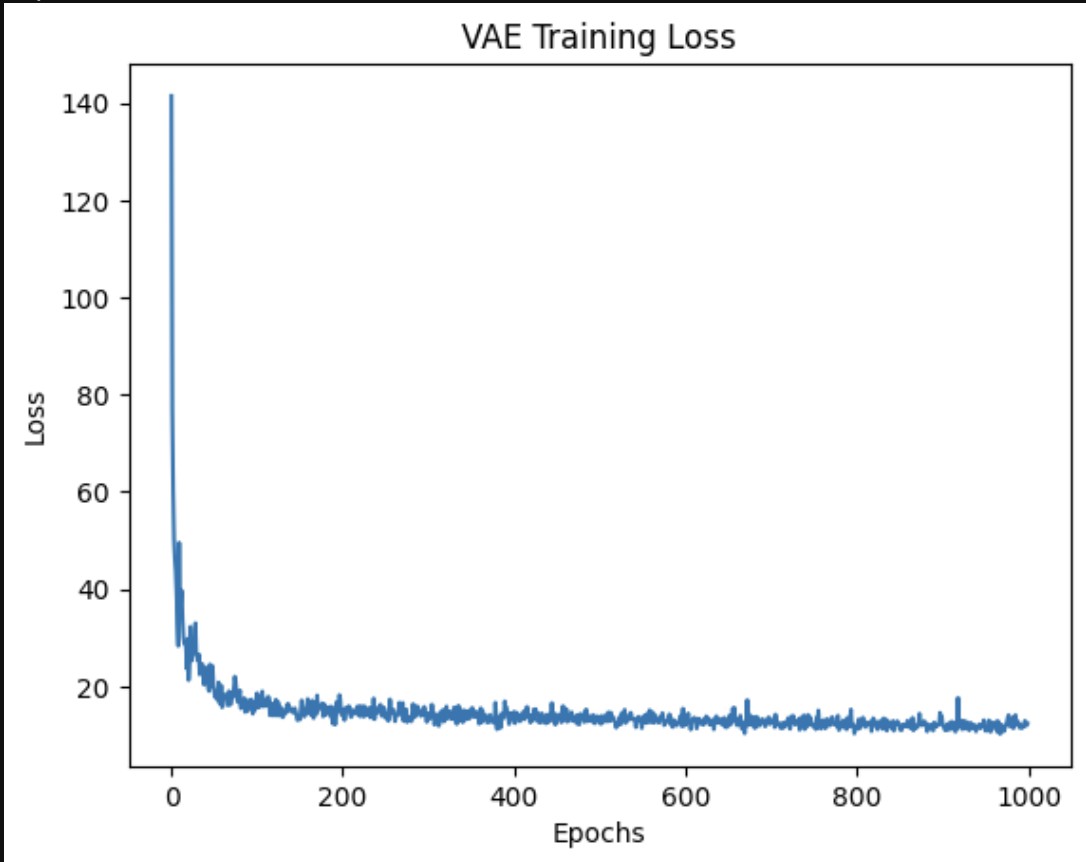


1. Generated loss graphs

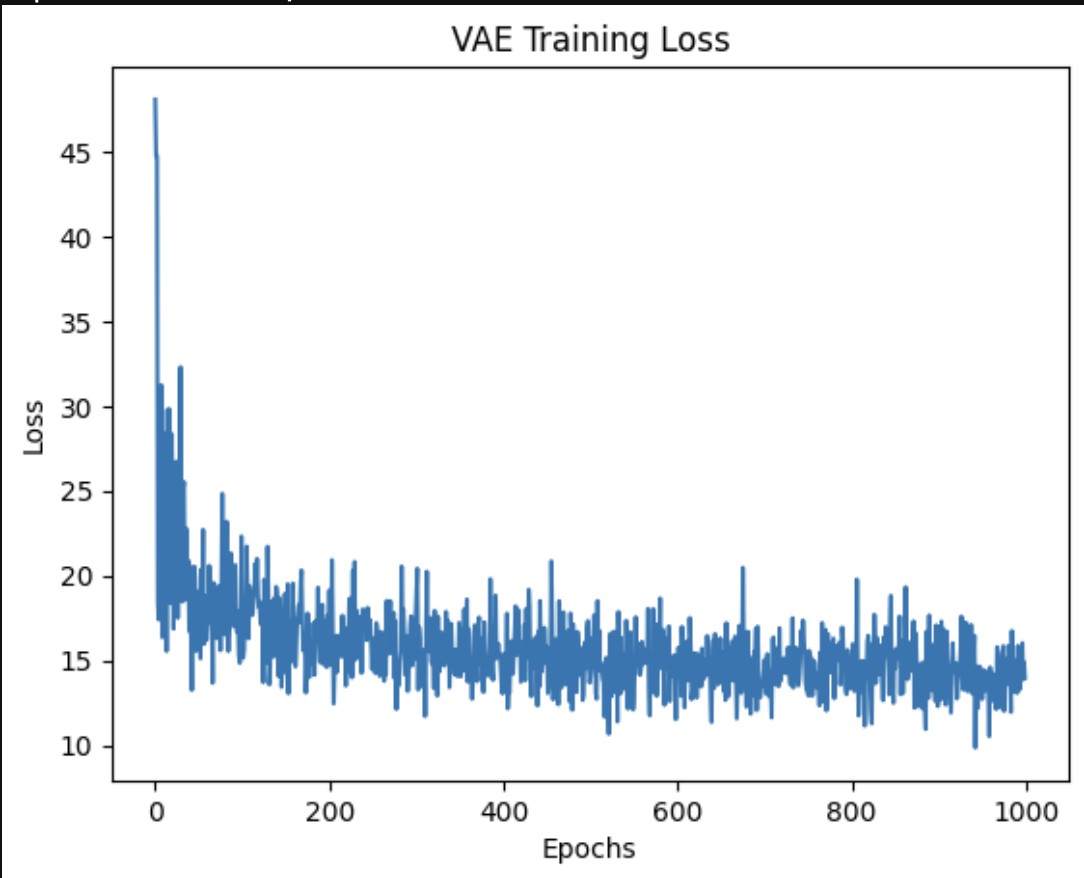
**Center**:



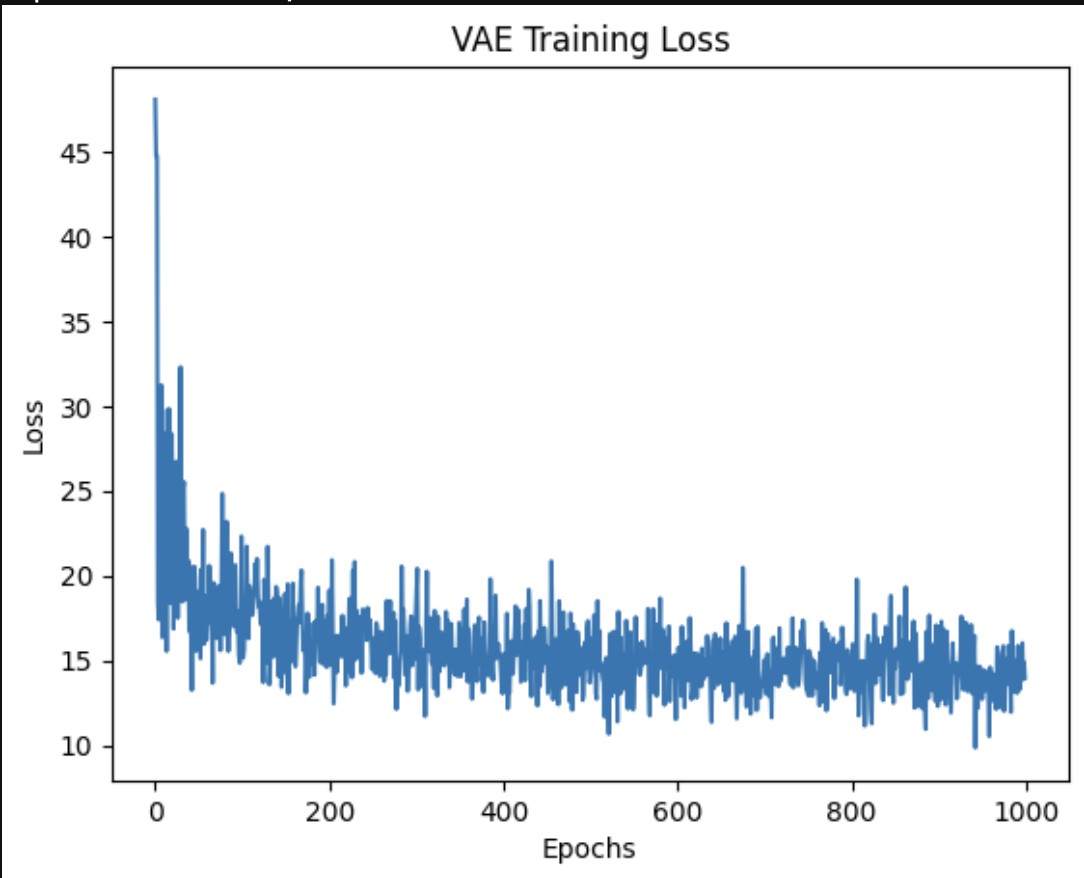
**Donut:**



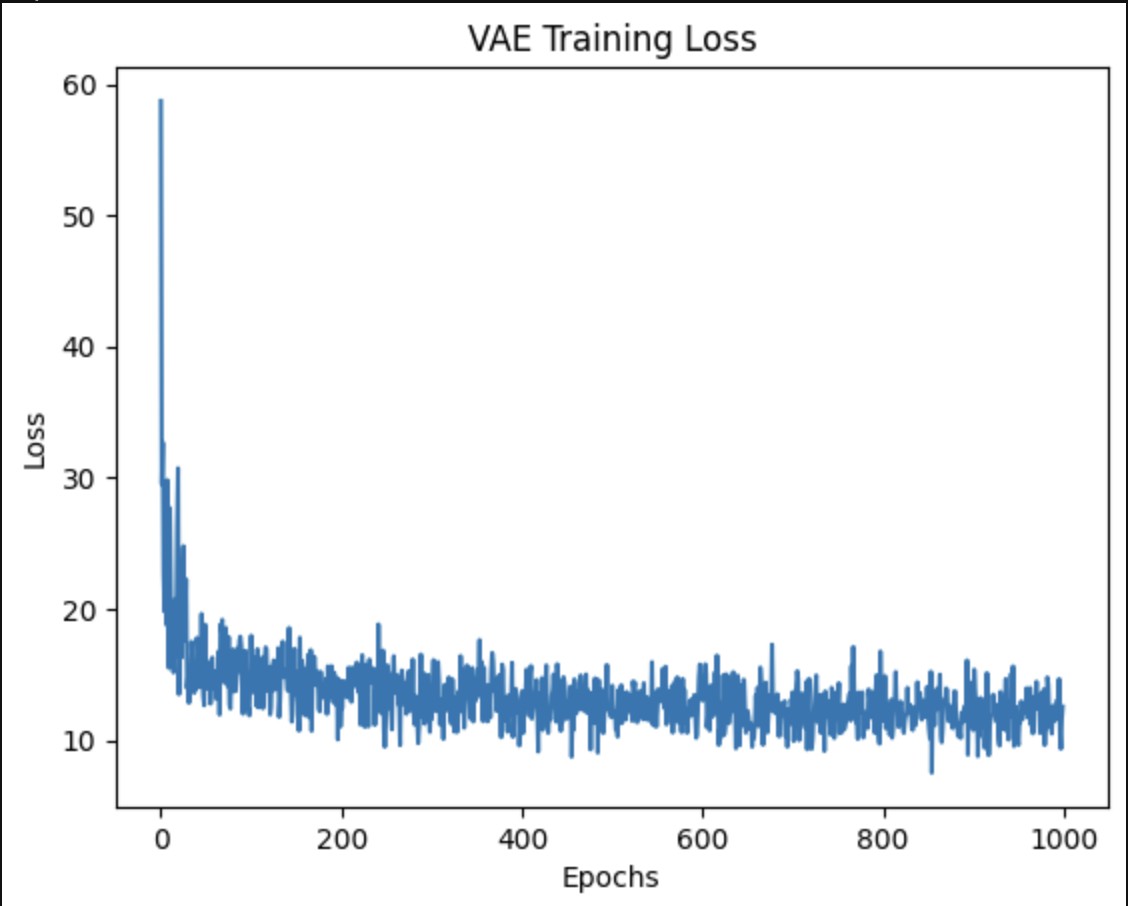
**Edge Loc:**



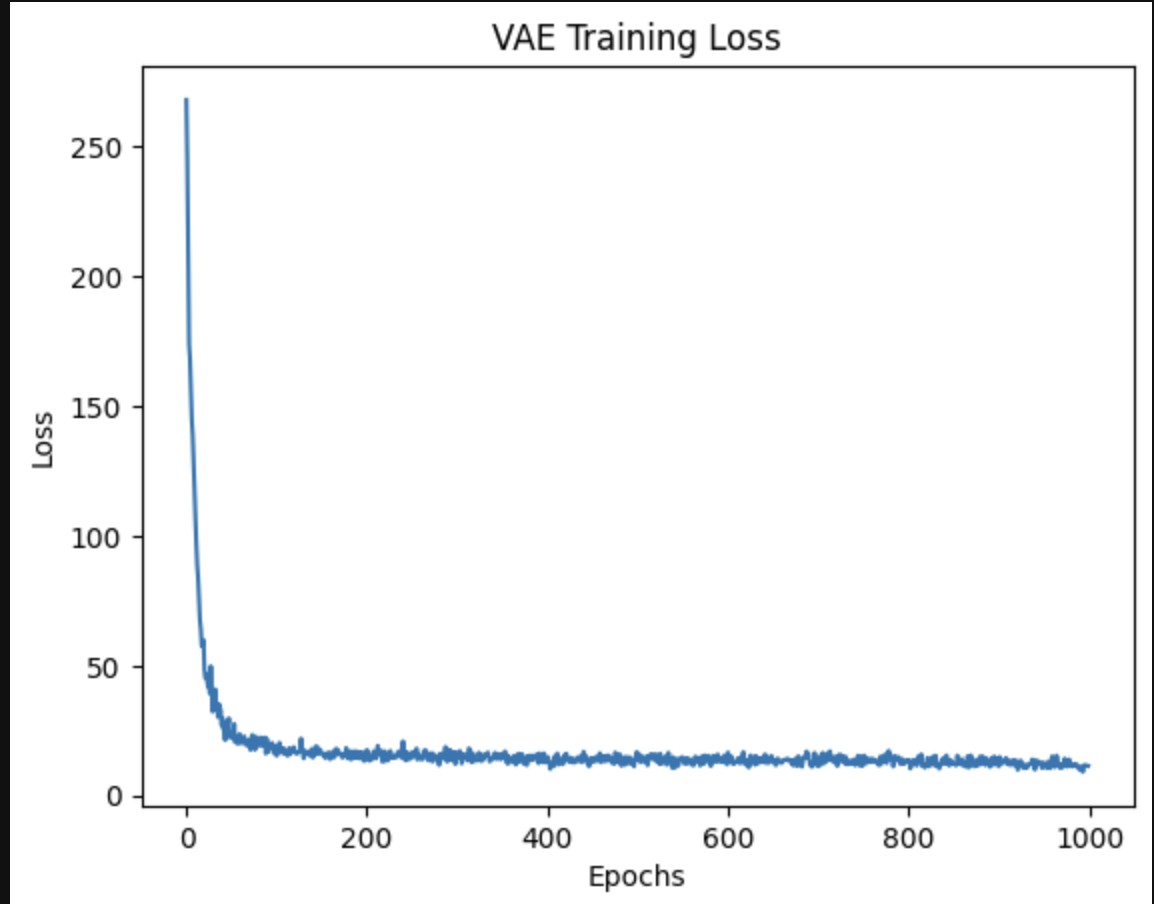
**Edge Ring:**



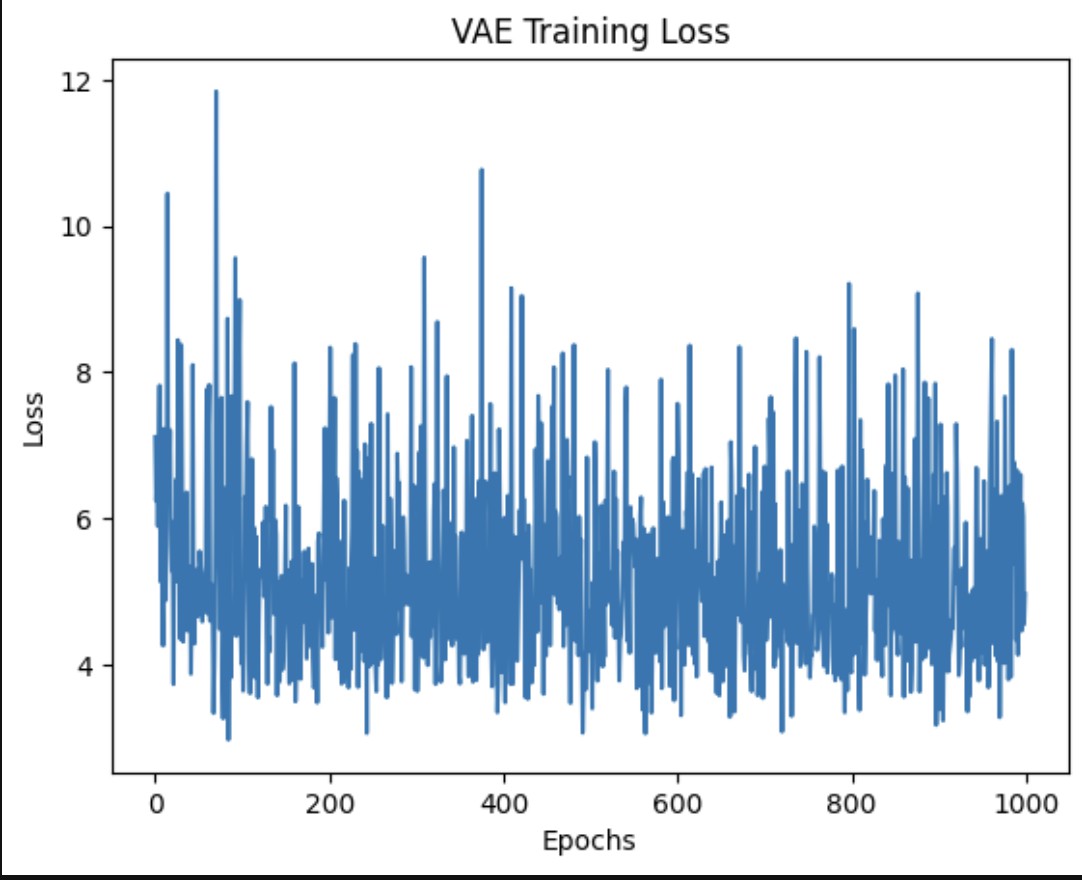
**Loc:**



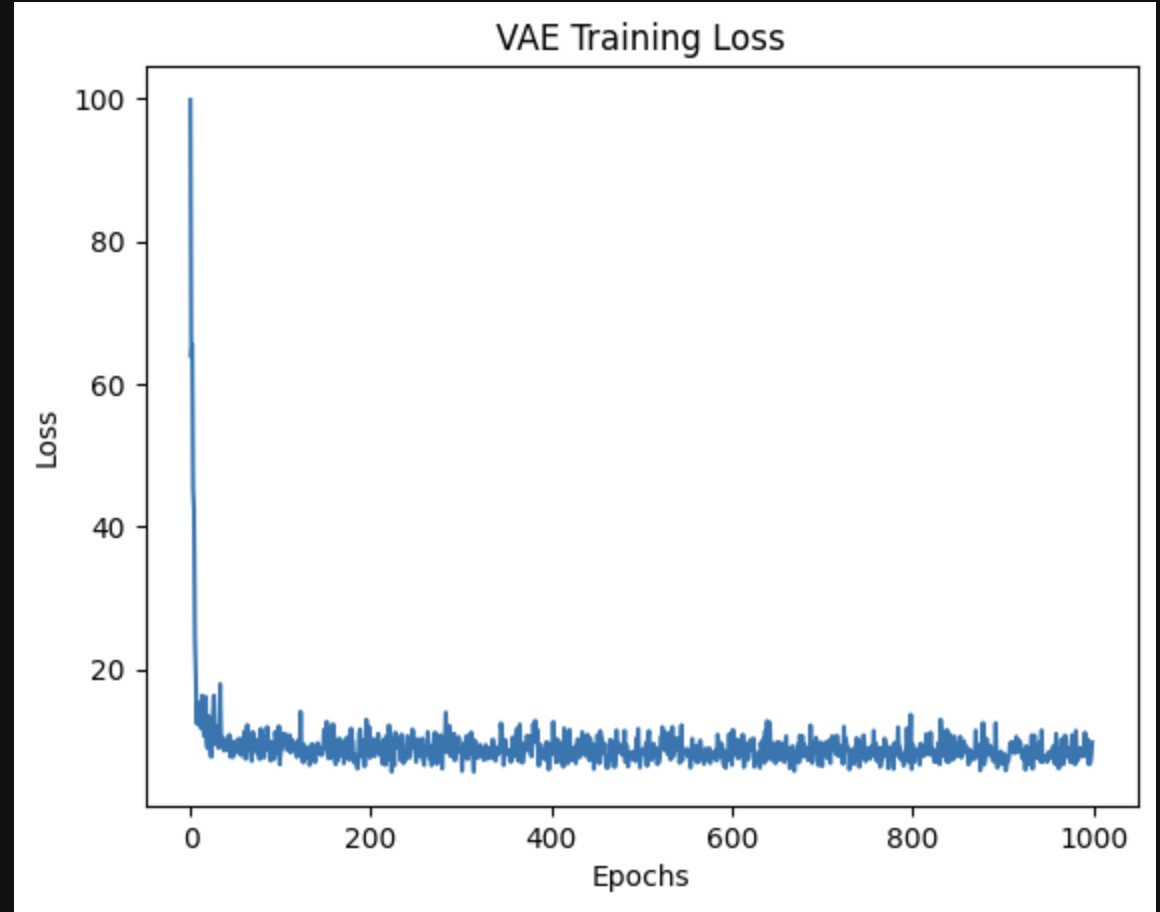
**Nearfull:**



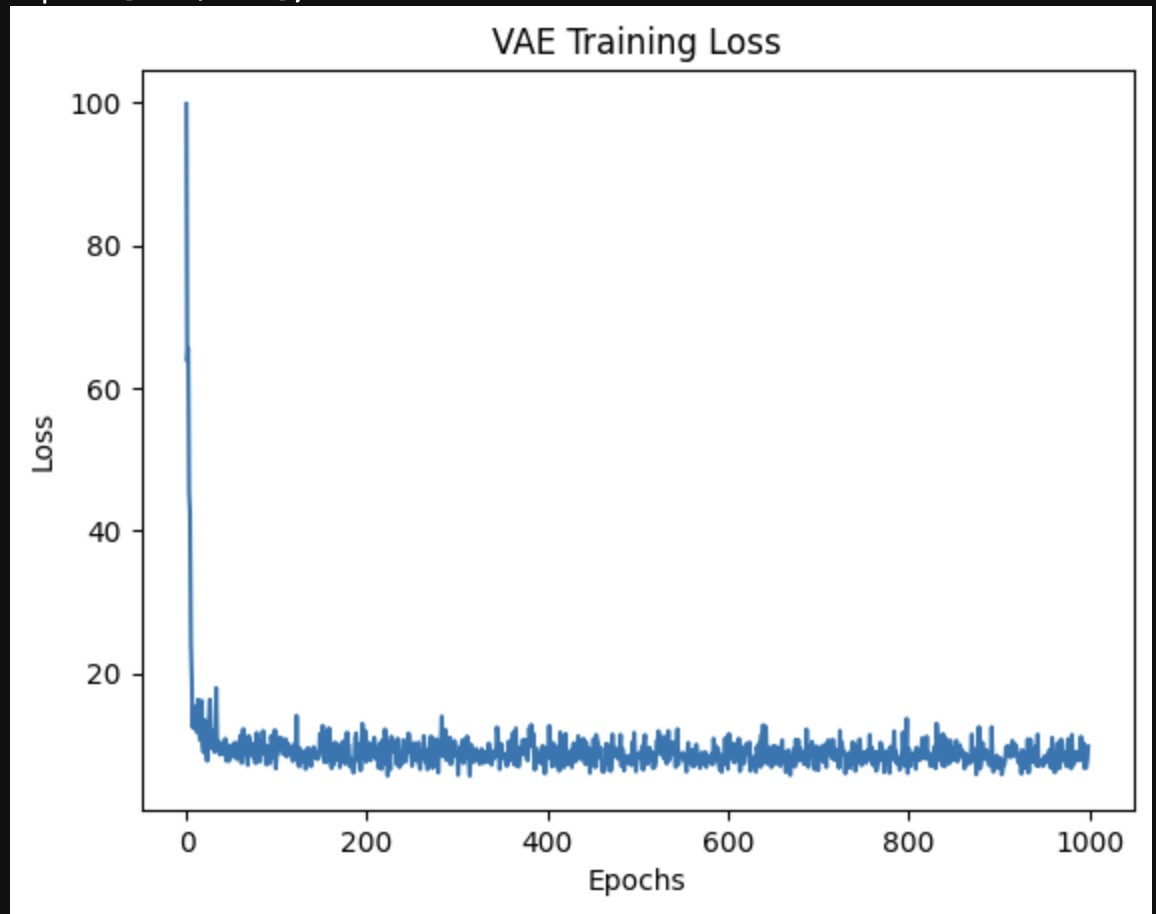
**None:**



**Random:**



**Scratch:**



1. Samples of best generated images:

Center:



Donut:



EdgeLoc:



EdgeRing:



Loc:



Nearfull:



None:



Scratch:



Random:

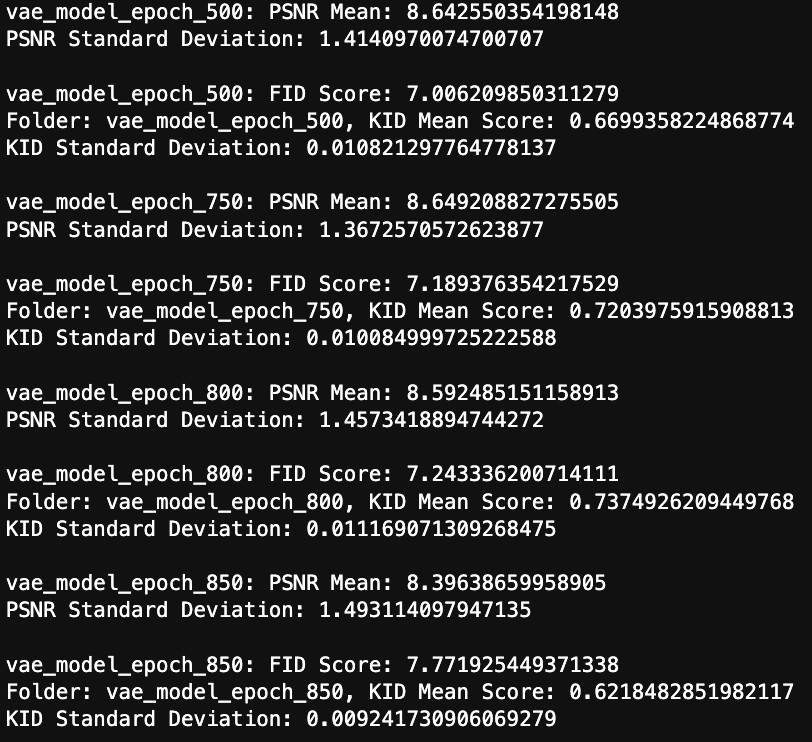


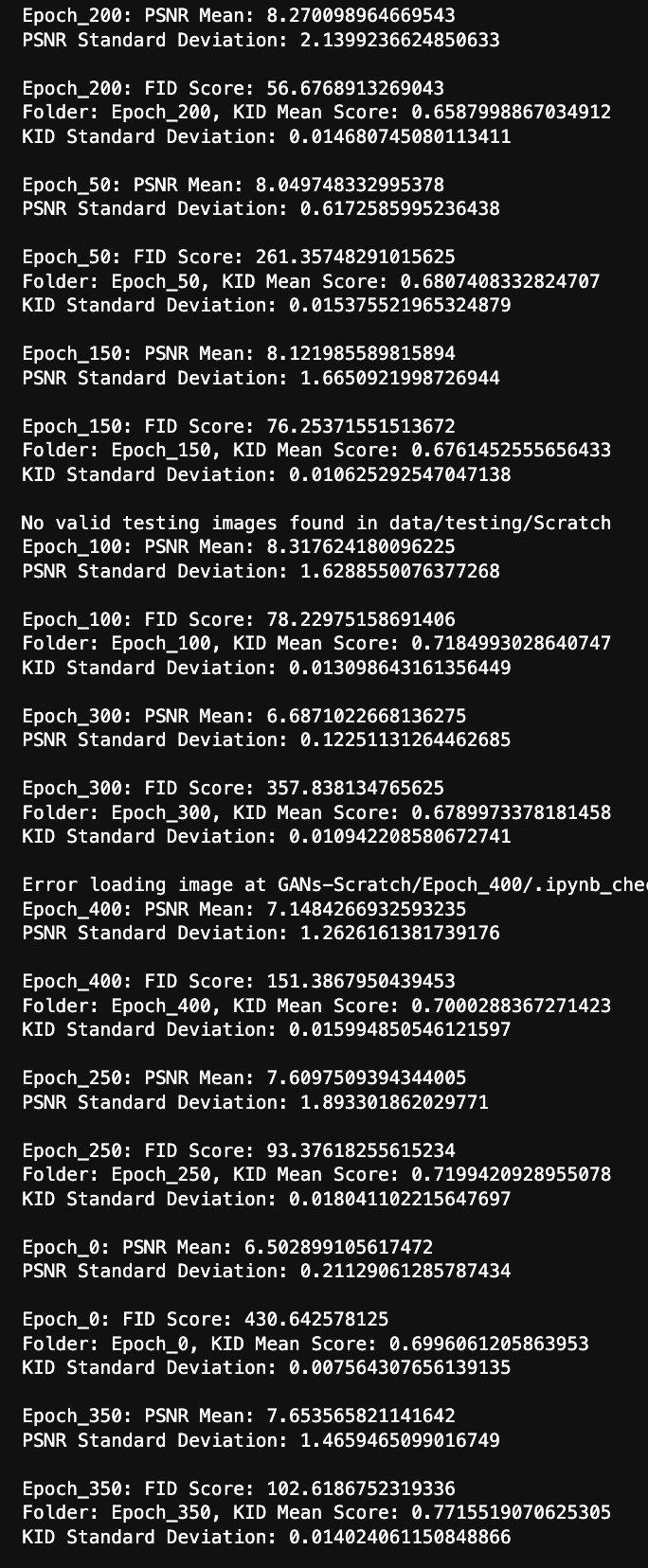


1. Evaluation metrics Results

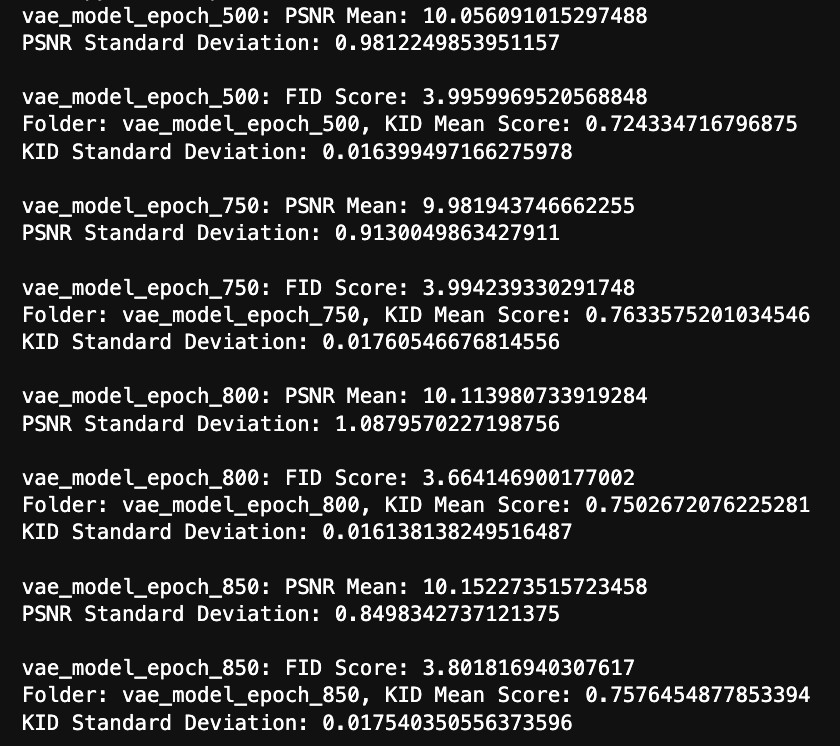
PSNR, IS, KID and FID scores:

None:

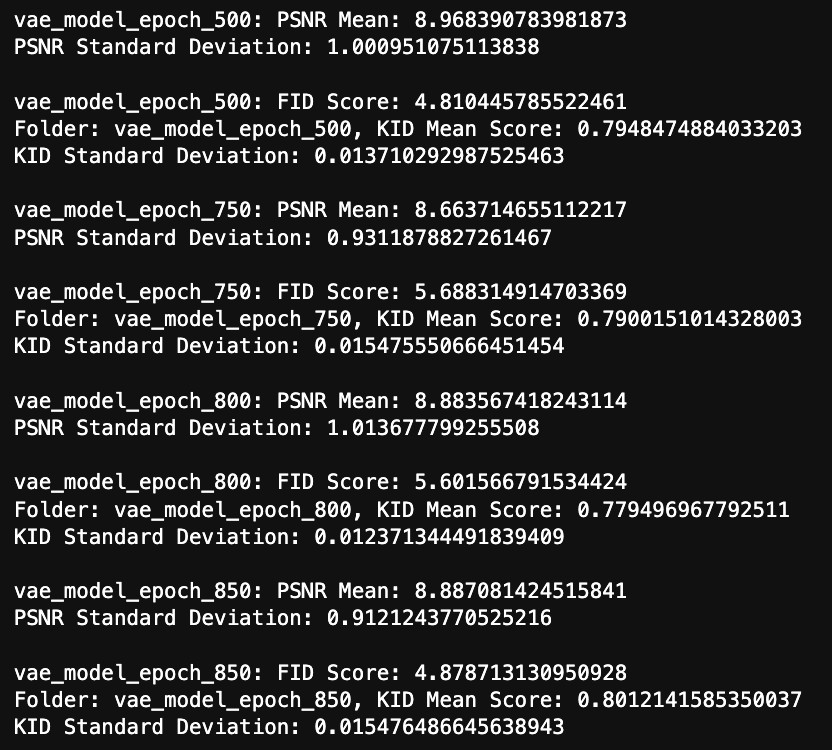




EdgeRing:



LOC:



Discussion of the results based on both evaluation metrics and loss graphs:

The results of a Variational Autoencoder (VAE) based on both evaluation metrics and loss graphs.

# Evaluation Metrics:

* **Reconstruction Error:**
  + One of the primary metrics for a VAE is the reconstruction error, which measures how well the model can reconstruct the input data. A lower reconstruction error indicates better performance in capturing the essential features of the input.

# KL Divergence:

* + The KL divergence measures the dissimilarity between the learned latent distribution and the assumed prior distribution. It reflects how well the VAE is regularizing the latent space. Lower KL divergence values suggest a more meaningful and organized latent representation.

# Generative Performance:

* + Evaluate the quality of the generated samples. This can be done subjectively by visual inspection or quantitatively using metrics like Frechet Inception Distance (FID) if dealing with image data.

# Loss Graphs:

* **Reconstruction Loss:**
  + Plot the reconstruction loss over training epochs. It should decrease over time, indicating that the model is improving in reconstructing the input data.

# KL Divergence Loss:

* + Monitor the KL divergence loss. A consistent increase in KL divergence loss may indicate that the model is struggling to maintain a balance between reconstruction accuracy and latent space regularization.

# Total Loss:

* + The total loss is the sum of the reconstruction loss and the KL divergence loss. It provides an overall measure of how well the VAE is performing. A

decreasing total loss indicates that the model is learning a good representation of the data.

# Interpretation:

* **Ideal Scenario:**
  + In an ideal scenario, you would observe a steady decrease in both the reconstruction loss and KL divergence over epochs, leading to a low and stable total loss.

# Overfitting:

* + If the reconstruction loss decreases while the KL divergence increases, it might indicate overfitting. The model is fitting the training data well but may struggle to generalize to new, unseen data.

# Underfitting:

* + If both losses remain high, the model may be underfitting. It fails to capture the underlying structure of the data, leading to poor reconstruction and a disorganized latent space.

# Adjusting Hyperparameters:

* + If the losses are not behaving as expected, it might be necessary to adjust hyperparameters such as the learning rate, latent space dimension, or the weighting of the reconstruction and KL divergence terms.

# Qualitative Evaluation:

* + Alongside quantitative metrics, visually inspect generated samples.

High-quality generated samples suggest that the VAE is learning meaningful representations.

Comparison of VAE and GAN

Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are both generative models, but they have different approaches to generating new data. Here's a brief comparison of the results and characteristics of GANs and VAEs:

# Training Approach:

* + **GANs:** GANs consist of a generator and a discriminator that are trained adversarially. The generator tries to create realistic data to fool the discriminator, while the discriminator learns to distinguish between real and generated data.
  + **VAEs:** VAEs use a probabilistic encoder-decoder framework. The encoder maps input data to a probability distribution in the latent space, and the decoder samples from this distribution to generate new data.

# Sample Quality:

* + **GANs:** GANs often produce high-quality, visually appealing samples. They are known for generating sharp and realistic images.
  + **VAEs:** VAEs may produce slightly blurrier images compared to GANs, as the objective involves maximizing the likelihood of the training data, which may result in a trade-off between sample quality and diversity.

# Latent Space:

* + **GANs:** GANs do not explicitly model the latent space. The generator produces samples directly without a clear mapping to a specific latent space.
  + **VAEs:** VAEs explicitly model a latent space where each point corresponds to a different encoding of the input data. This makes VAEs more interpretable and facilitates controlled generation by manipulating points in the latent space.

# Mode Collapse:

* + **GANs:** GANs can suffer from mode collapse, where the generator only produces a limited variety of samples, ignoring the diversity present in the training data.
  + **VAEs:** VAEs are less prone to mode collapse due to the inherent regularization in the model. The probabilistic nature of the latent space encourages diversity in generated samples.

# Training Stability:

* + **GANs:** Training GANs can be challenging and less stable. Achieving a balance between the generator and discriminator is crucial, and mode collapse and convergence issues can occur.
  + **VAEs:** VAEs are generally more stable during training. The objective function includes a reconstruction term, which helps in learning a meaningful latent representation.

# Applications:

* + **GANs:** GANs are often preferred in applications where high-quality, realistic samples are crucial, such as image synthesis, style transfer, and deepfake generation.
  + **VAEs:** VAEs are suitable for tasks where a structured latent space and interpretability are important, such as image generation with explicit control over specific features.

In general, the choice between GANs and VAEs depends on the specific requirements of the task at hand. GANs are favored for their ability to produce visually impressive samples, while VAEs offer a more interpretable latent space and stable training.

Researchers often explore hybrid models that combine the strengths of both approaches.