HW4 – Machine Learning 2 – 097209

Generative Models

Avi Mish, 208470575

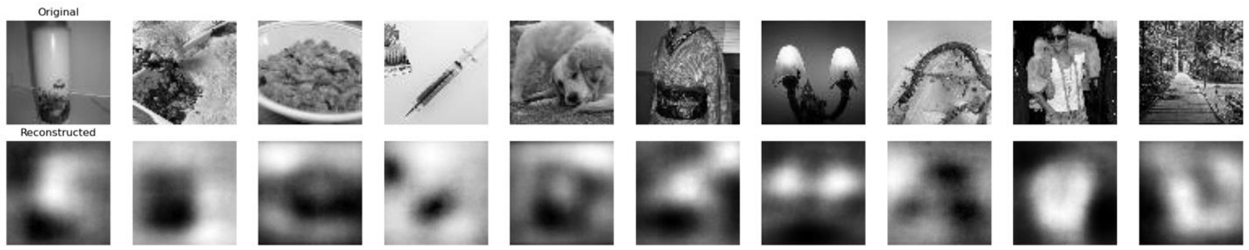
Netanel Shalev, 206897969

VAE

Attempt Number 1:

Hyper-Parameters			
Latent dimension	**Learning rate**	**Epochs**	**Batch size**
20	**0.001**	**50**	**512**

For the first attempt, we used a rather simple architecture design. We wanted to 'test the waters' so we just made something that will produce any results for grayscale, this is what we got:

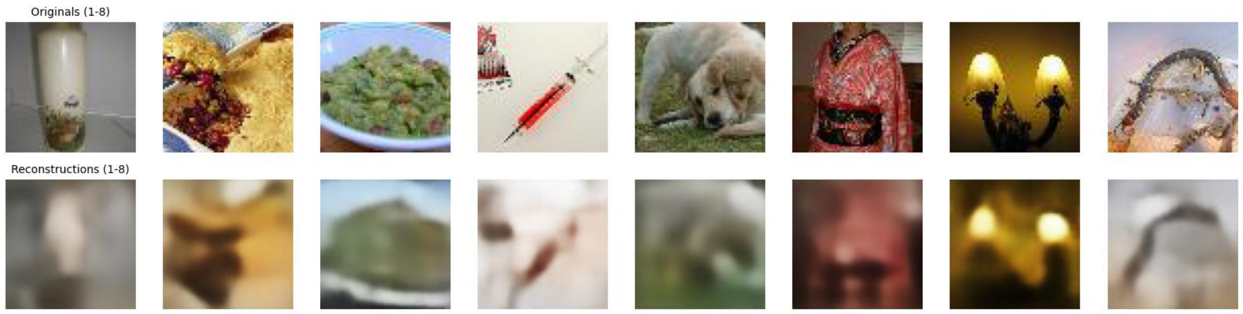


As it is possible to see, the model was able to recognize very general idea of the original images – it only was able to differentiate where the image is brighter and where is it shadier.

Attempt Number 2:

Hyper-Parameters			
Latent dimension	**Learning rate**	**Epochs**	**Batch size**
128	**0.001**	**50**	**128**

This time, we used a more complex architecture design. moreover, the images weren't transformed into grayscale. These are the results we got:

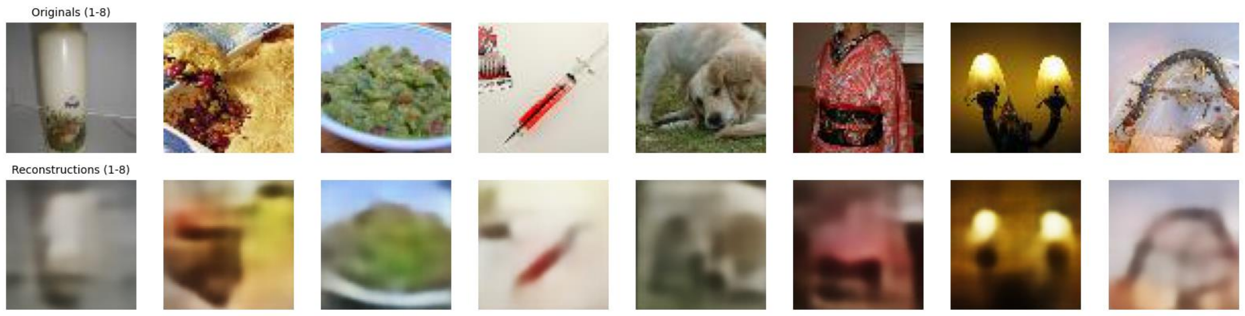


In this case, we can recognize that the model was able to differentiate colors well. The reconstructed images are very blurry, but the general features of the image are preserved.

Attempt Number 3:

Hyper-Parameters			
Latent dimension	**Learning rate**	**Epochs**	**Batch size**
128	**0.001**	**50**	**128**

In this attempt, we only changed the loss function from MSE to BCE. These are the results:

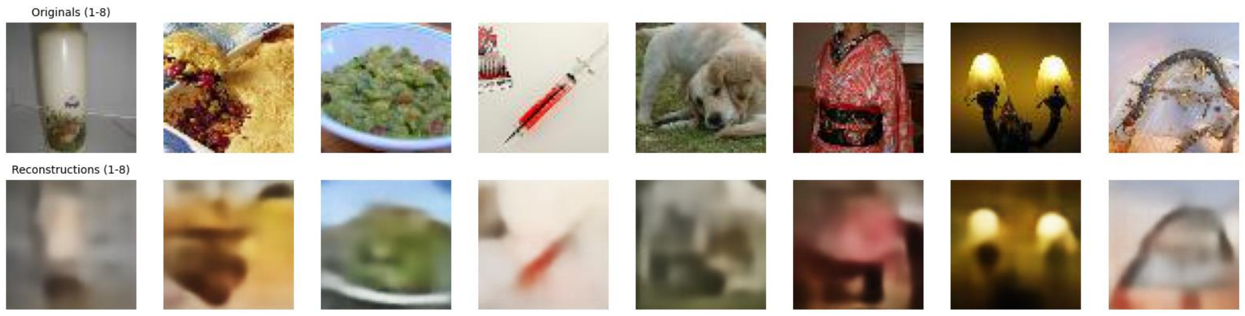


We can see that just by changing the loss function we got better results, better representing the original images, yet still blurry.

Attempt Number 4:

Hyper-Parameters			
Latent dimension	**Learning rate**	**Epochs**	**Batch size**
128	**0.0001**	**80**	**128**

For the fourth attempt, we tried a more complex architecture design to try and capture finer details of the original images. Also, we trained for 30 epochs more, and we had to tweak the learning rate for convergence. Results:

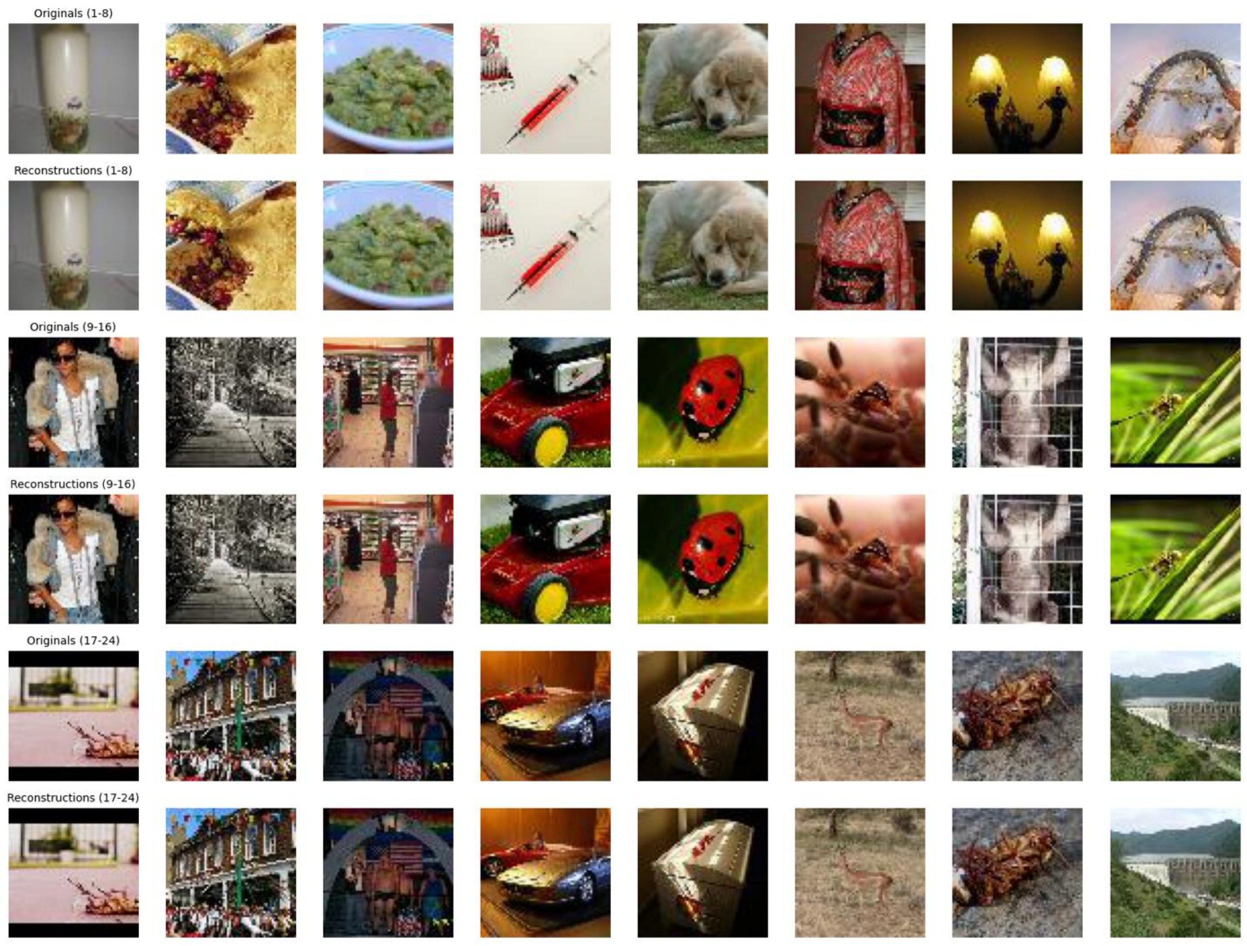


We can see that the images got sharper and that the reconstructed images were able to capture more details of the original images.

Attempt Number 5:

Hyper-Parameters			
Latent dimension	**Learning rate**	**Epochs**	**Batch size**
256	**0.0001**	**100**	**256**

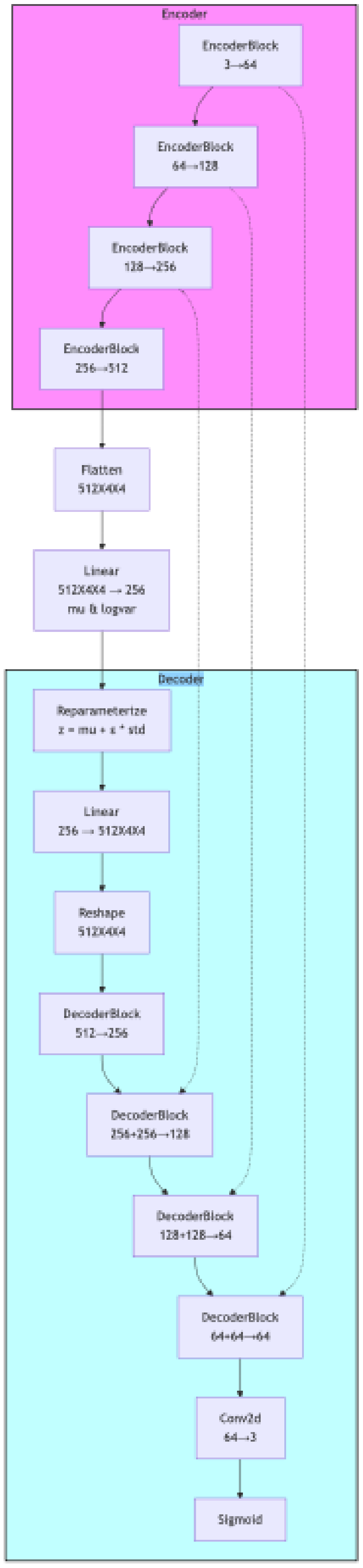
For our final attempt, we used our most complex architecture yet, U-net style which produced Latent Dimension of 256. We increased the number of epochs even more, and left the machine to train all night, and these are the results:



As we can see, the results for the complex architecture are very good. The reconstructed images are indistinguishable from the original images, and the human eye can barely recognize the differences.

The **ComplexConvVAE** is an advanced generative model that integrates Convolutional Neural Networks (CNNs) with Variational Autoencoder (VAE) principles and Residual Networks (ResNets) to effectively encode and reconstruct images. This architecture is designed in a U-Net style, utilizing skip connections to preserve spatial information and enhance reconstruction quality.

Visualization using Mermaid



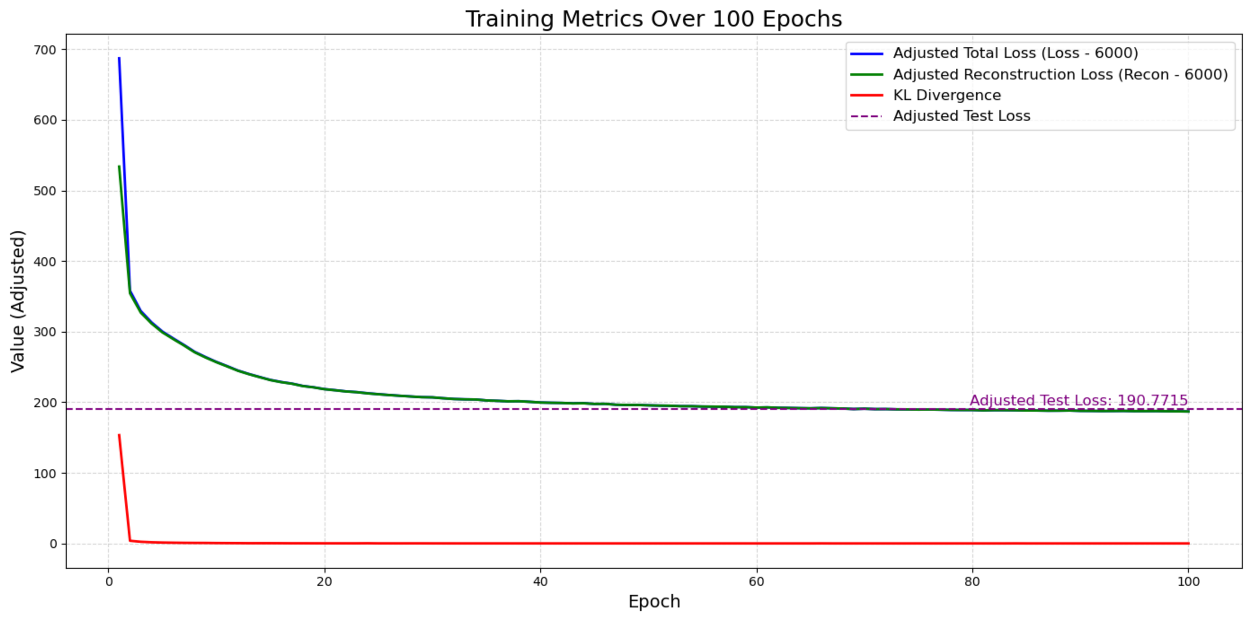
Key Components

1. **Encoder**:
   - **Structure**: Comprises four EncoderBlock modules, each reducing the spatial dimensions by half (e.g., from 64×64 to 32×32) while increasing feature channels (e.g., from 64 to 512).
   - **Residual Blocks**: Each EncoderBlock includes multiple ResidualBlock layers that facilitate deeper feature extraction and improve gradient flow through skip connections.
2. **Bottleneck**:
   - **Latent Space Representation**: The encoder's output is flattened and passed through linear layers to generate the mean (mu) and log-variance (logvar) vectors, defining the latent distribution.
   - **Reparameterization**: Utilizes the reparameterization trick (z = mu + ε * std) to sample latent vectors, enabling backpropagation through stochastic nodes.
3. **Decoder**:
   - **Structure**: Mirrors the encoder with four DecoderBlock modules, each increasing the spatial dimensions by two (e.g., from 4×4 to 8×8) while decreasing feature channels.
   - **Skip Connections**: Integrates skip connections from corresponding encoder layers to retain high-resolution features, enhancing the fidelity of reconstructed images.
   - **Output Layer**: Ends with a convolutional layer and a Sigmoid activation to produce output images with pixel values in the [0, 1] range.

Workflow Summary

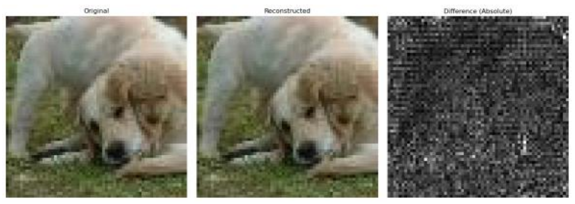
1. **Encoding**: Input images pass through the encoder, progressively reducing spatial size and increasing feature depth, resulting in a compact latent representation.
2. **Latent Sampling**: The bottleneck generates mu and logvar, from which latent vectors z are sampled using the reparameterization trick.
3. **Decoding**: Latent vectors are transformed back into high-dimensional feature maps through the decoder, utilizing skip connections to reconstruct the original image with high accuracy.

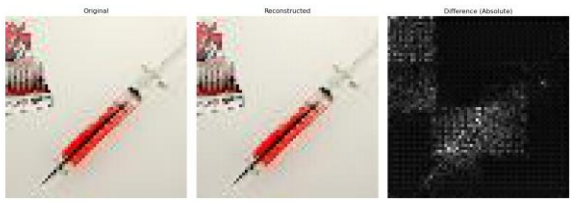
Plot loss of training over time:

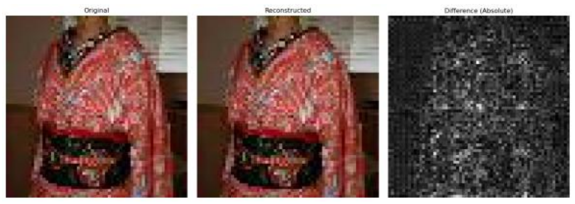


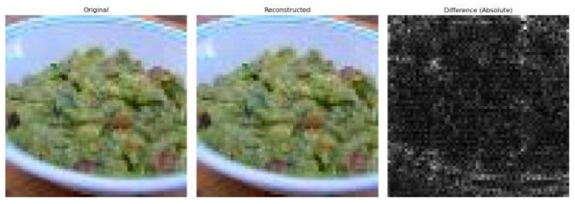
The loss here is adjusted for better presentation – the Total loss and Recon loss were subtracted by 6000. The original numbers were - Epoch 1/100, Loss: 6687.2963, Recon: 6533.8883, KL: 153.4080 to Epoch 100/100, Loss: 6186.9388, Recon: 6186.9387, KL: 0.0001

Differences between reconstructed and original images visualized:



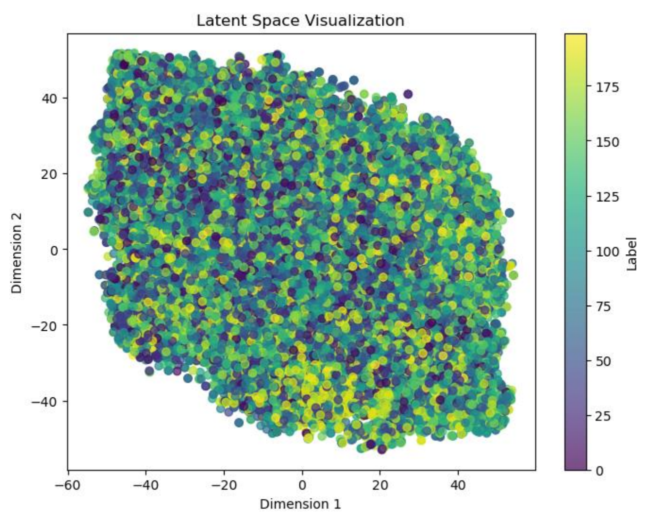


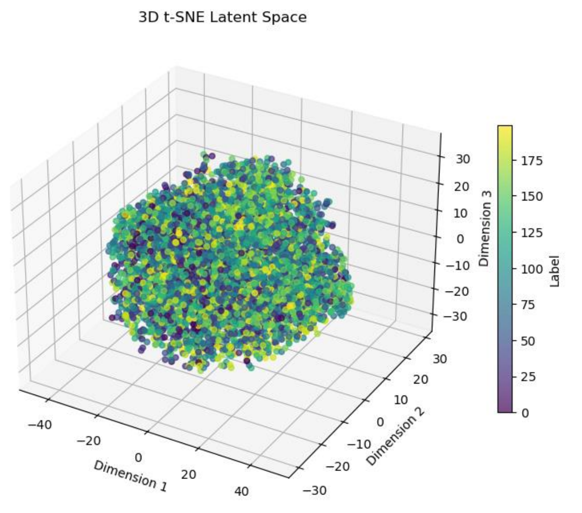




Just to make sure that the reconstructed images are actually different from the originals ☺

Latent Space visualizations:





From the Latent Space visualizations, we can’t really produce any meaningful insights. The Latent Space produced by the model is probably too complex to be dumbed-down into 2/3 dimensions that will provide actual clusters with the same label.