CMPE 597 Sp. Tp. Deep Learning: Assignment 2

Group 10: Tarik Can Ozden – Ahmet Firat Gamsiz

2022400327 - 2020400180

Source code link: https://drive.google.com/file/d/1bAaNjjQ6i5mC1srB1_iNSpZY_YWoLG9I

I. Autoencoders

1. **Consider a gated recurrent neural network as the encoder and decoder. Please treat the grayscale images as multivariate time series. You may try Long Short-Term Memory (LSTM) or Gated Recurrent Unit (GRU). You need to determine the number of layers and the number of neurons in the units.**

   We used LSTM for both the encoder and decoder, with a hidden size of 128, two hidden layers, and a latent dimension of 64, after conducting a hyperparameter search. Similarly, the decoder maps vectors from the 64-dimensional latent space to the image space. We used Adam optimizer with a learning rate of 0.005, and trained the autoencoder for 60 epochs.

2. **Consider convolutional encoder and decoder. You must determine the number of layers, kernel sizes, number of feature maps, and the activation functions.**

   For both the encoder and decoder, we used three convolutional layers with ReLU activation functions, a kernel size of 3, a stride of 2, and padding of either 0 or 1. After the convolutional layers, we used an FC layer to project into the latent space. Similarly, in the decoder, we started with an FC layer and then applied convolutional layers to reconstruct the image space. To ensure equal-dimensional latent spaces, we also set the latent dimensionality to 64 in the convolutional autoencoder. We used Adam optimizer with a learning rate of 0.0001, and trained the convolutional autoencoder for 30 epochs.

3. Please report the change in the MSE loss during training.

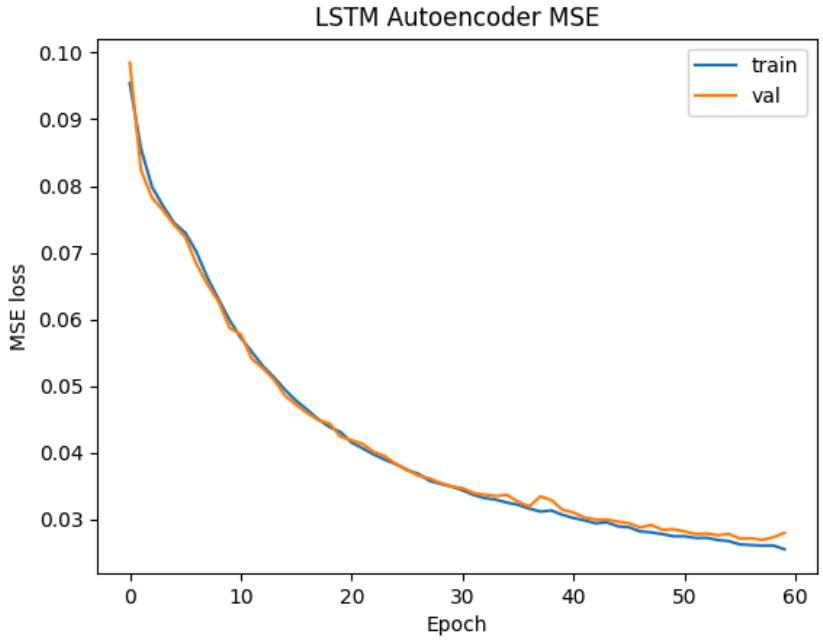


Figure 1. LSTM Autoencoder Loss Curves

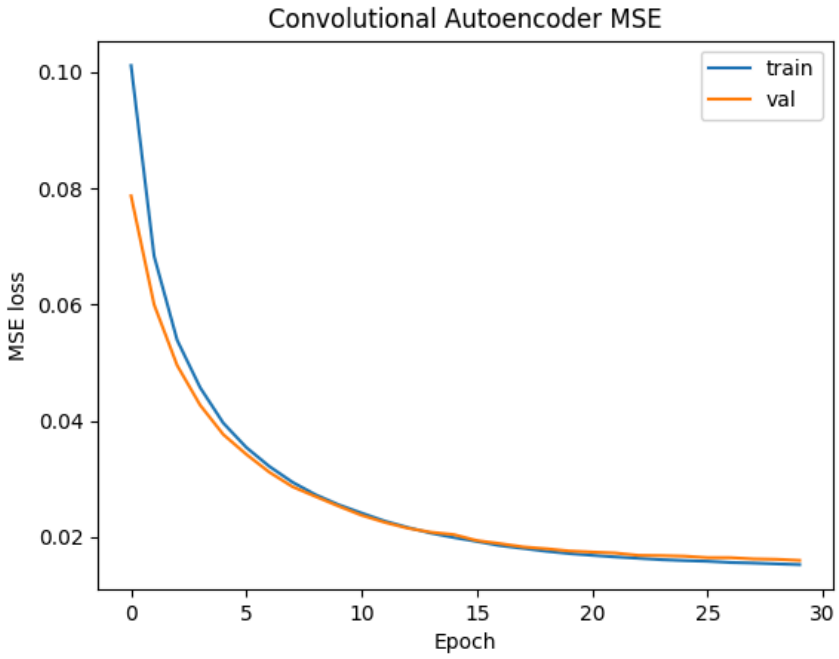


Figure 2. Convolutional Autoencoder Loss Curves

4. **Plot the embeddings obtained from the encoder in 2D using t-SNE. Comment on the discriminability of the embeddings learned by the autoencoders above.**

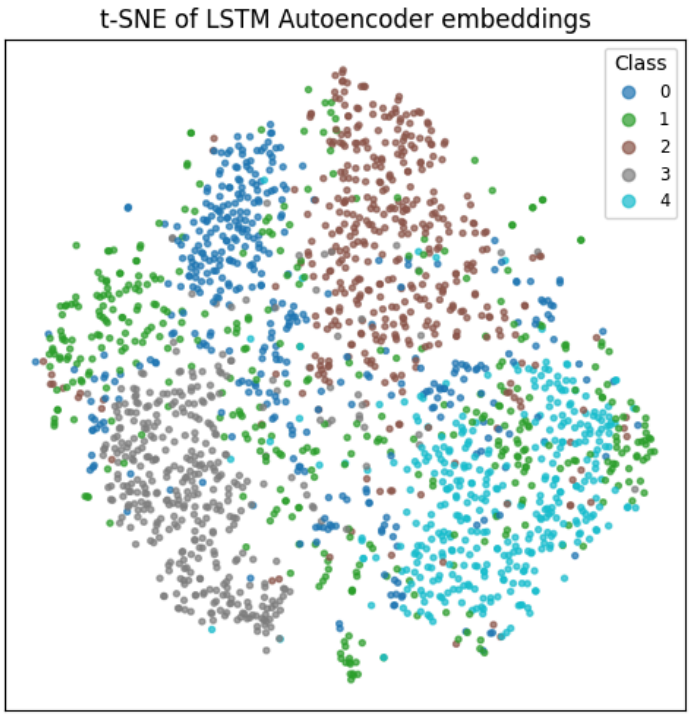


Figure 3. Embeddings of the LSTM Autoencoder

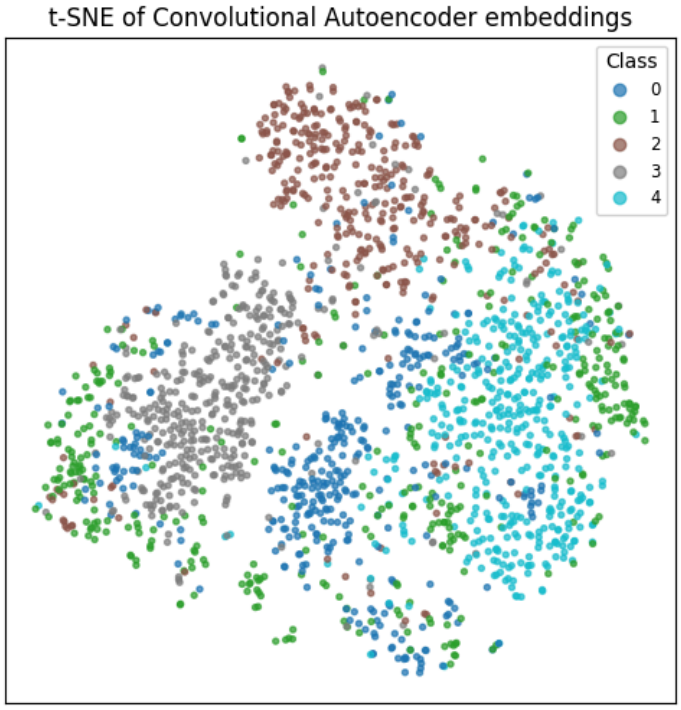


Figure 4. Embeddings of the Convolutional Autoencoder

As seen in Fig. 3 and Fig. 4, both autoencoders successfully discriminate most of the classes. The clusters corresponding to classes 2, 3, and 4 (hand, snowman, motorbike) are clearly separated from the others, indicating that the encoder can easily recognize their differences. However, neither model forms a distinct cluster for classes 0 and 1 (rabbit and yoga). For class 0, (rabbit), there appears to be a region where most of the samples fall, however, there are many samples that are scattered around the plot. Additionally, the convolutional autoencoder generates several small clusters within rabbit and yoga classes, suggesting intra-class differences that lead to scattered sub-clusters. In contrast, the LSTM autoencoder fails to form a clear cluster for class 1 (yoga), which appears to be scattered throughout the plot. This suggests that the convolutional autoencoder performs better in terms of class discriminability.

II. Variational Autoencoders

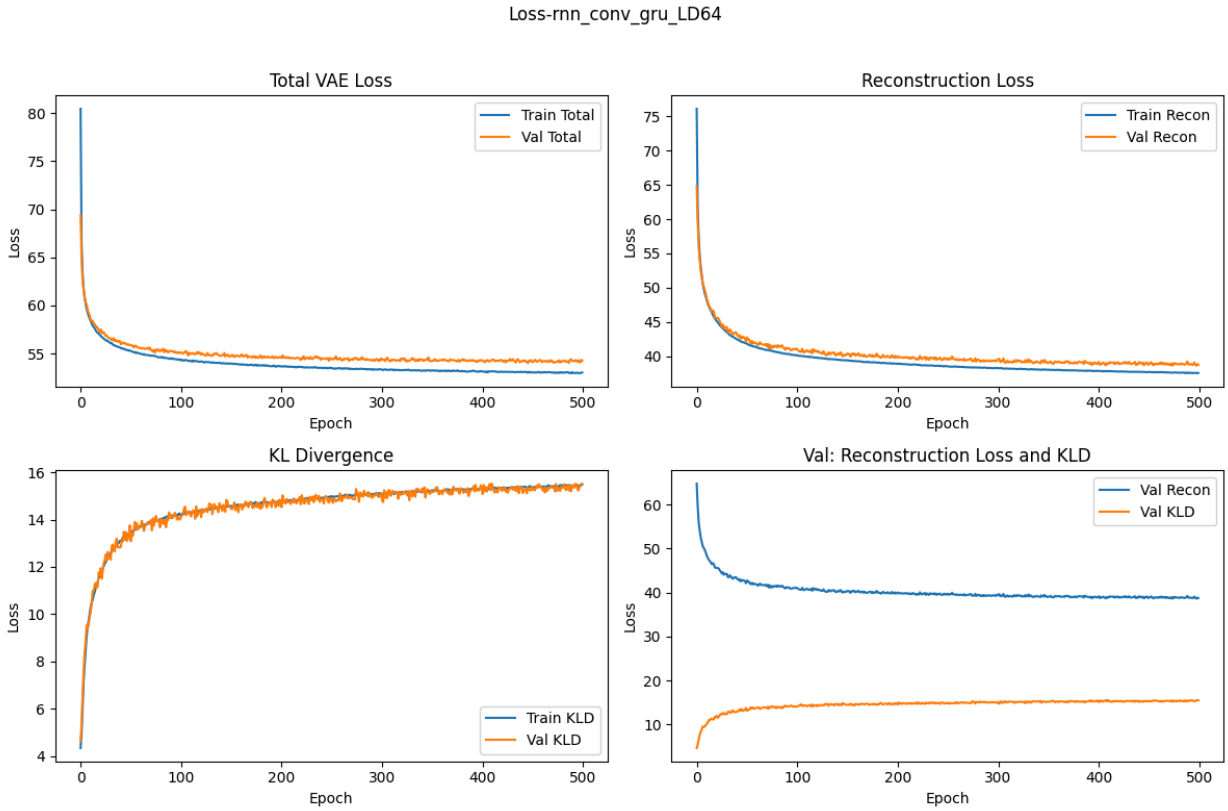
1. **Consider a VAE with the gated RNN encoder and a convolutional decoder. Train the VAE and plot the changes in its objective during training. Show the change in both KL divergence and reconstruction terms on the same plot.**

We implemented an autoencoder, using RNN as the encoder, a convolutional network as the decoder, and a projection matrix to connect the RNN to the convolutional network. After conducting a hyperparameter search, we set RNN structure and hyperparameters as follows:

```
"rnn_type": "gru",
"latent_dim": 64,
"beta": 1.0,
"lr": 0.001,
"hidden_dim": 128,
"num_layers": 3,
"encoder_variant": "last_hidden",
"bidirectional": true,
"batch_size": 64
```

We tested out both using RNN's last hidden state and full output, and grid search showed the hidden state performs better, as can be seen in encoder_variant. We extensively trained the models until 500 epochs and took the results from best best-performing epoch. For the convolutional network, we used the same network from previous section.

Our model had a 54.00 validation loss.



Reconstruction samples from training, validation, and test sets, respectively (top orig, bot recon):

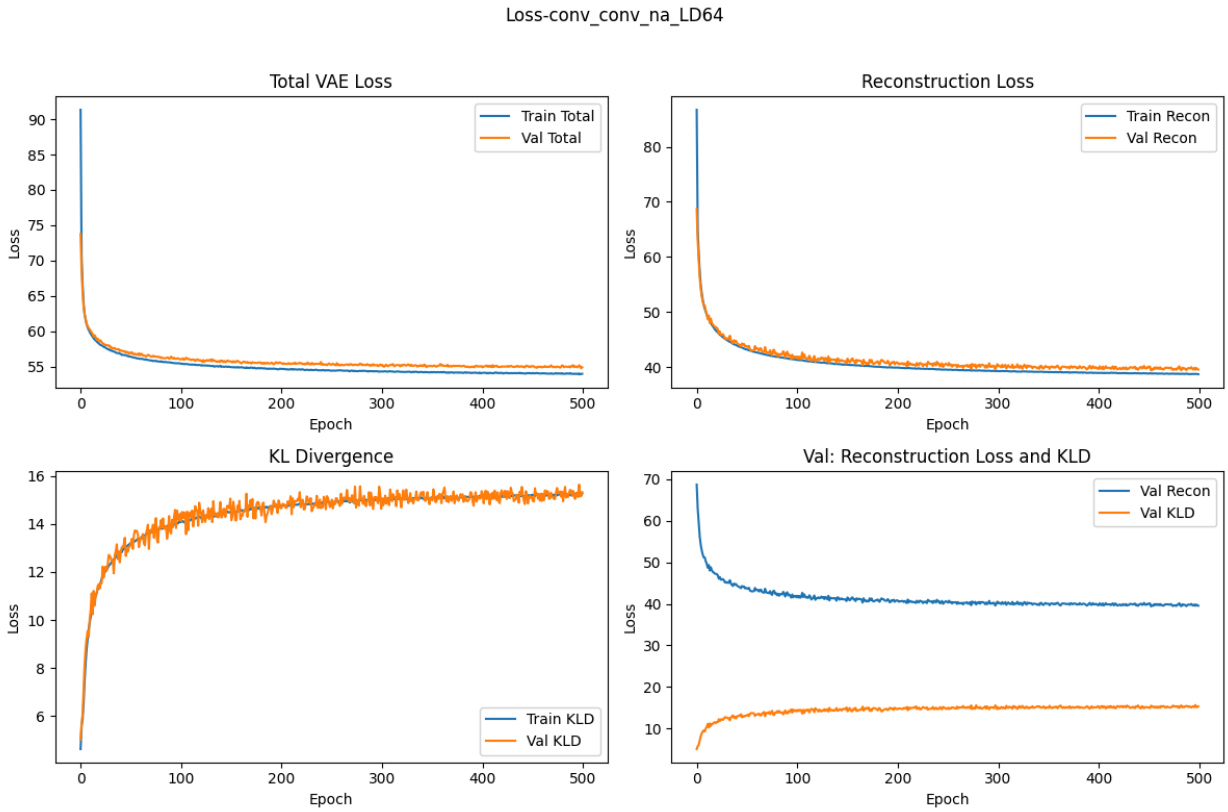
2. **Now, consider a VAE with a convolutional encoder and decoder. Compare its convergence with the one with the RNN encoder.**

For this part, we implemented the same network architecture from section 1 with VAE loss. Our hyperparameter search results are as follows:

```
        "latent_dim": 64,
```

```
"lr": 0.0005,
"batch_size": 64,
"beta": 1.0
```

Our model had a 54.76 validation loss.

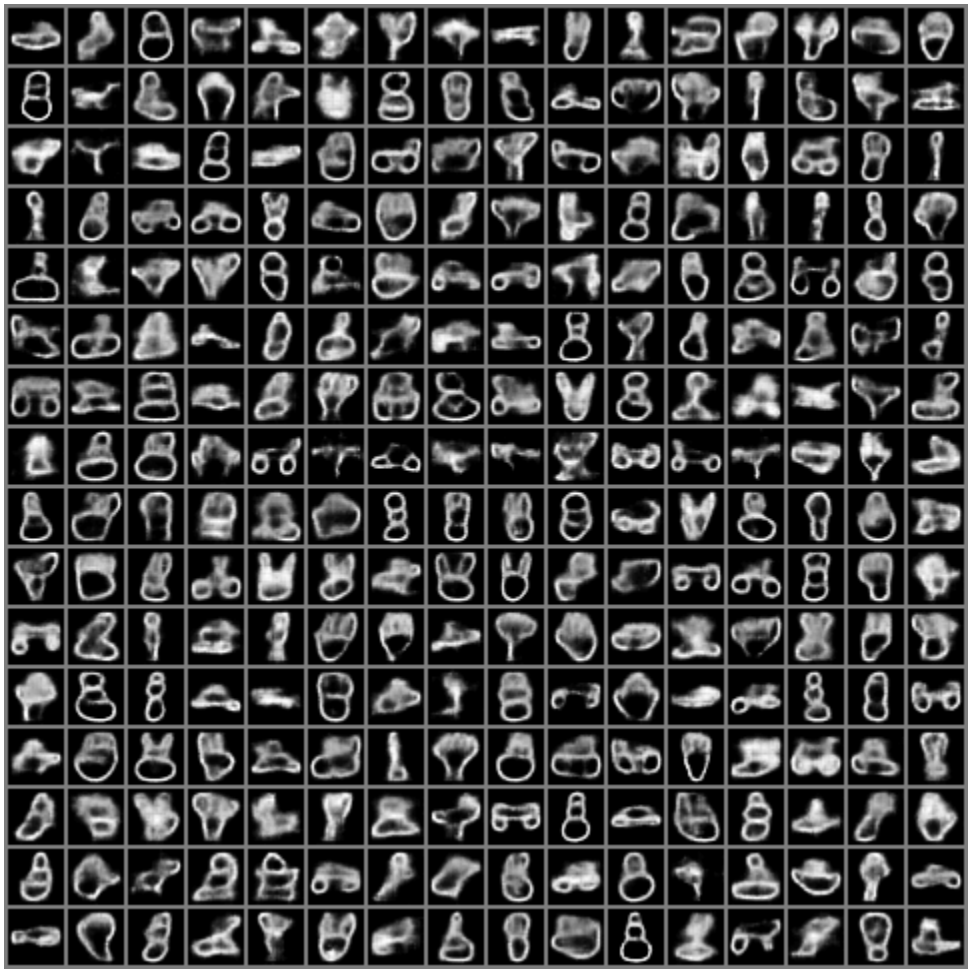


We consider RNN architecture converges to a better point faster and more stably.

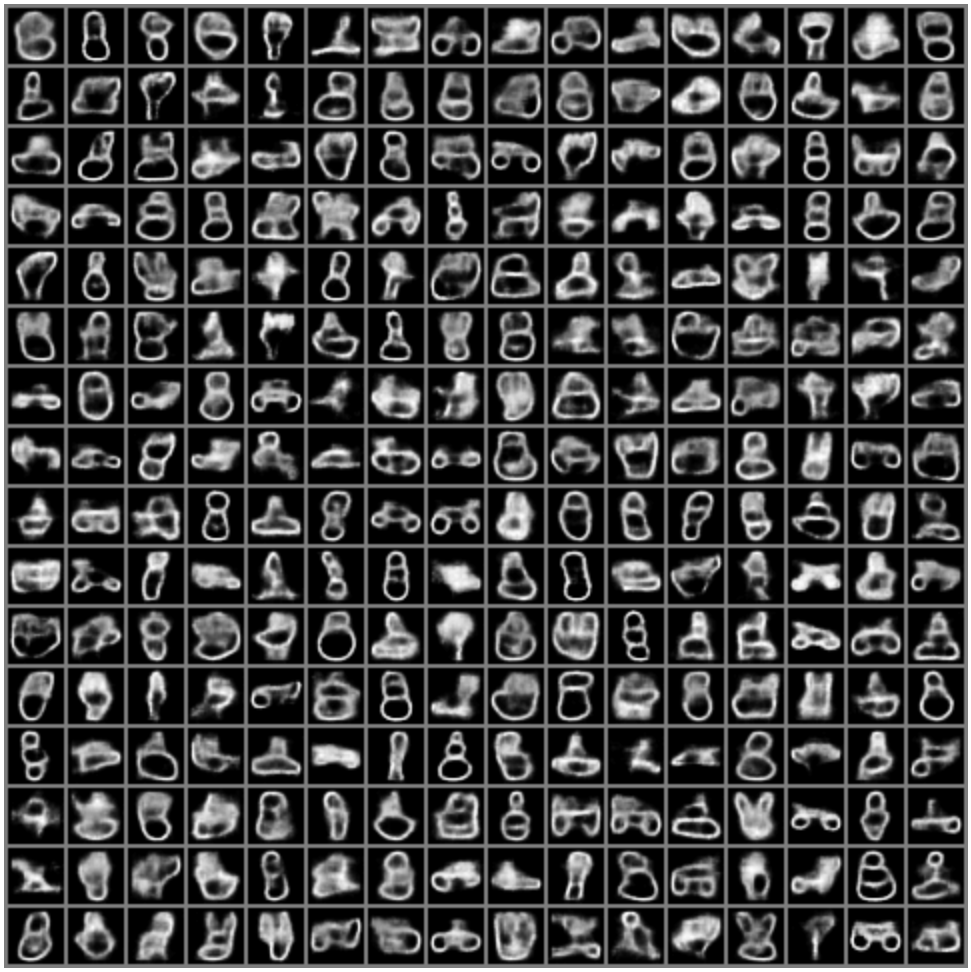
Reconstruction samples from training, validation, and test sets, respectively (top orig, bot recon):

3. **Generate new samples using the decoders in question 1 and 2. Compare their visual quality and diversity. You may use metrics such as Inception Score (IS) and Frechet Inception Distance (FID) to evaluate the visual quality.**

   Random new samples generated with RNN architecture:



   Random new samples generated with a convolutional network architecture:



In our manual inspections, we generally observed RNN architecture provides slightly less blurry images that are more faithful to images from dataset. We also used Inception Score and Frechet Inception Distance for evaluation:

	Inception Score	Frechet Inception Distance
RNN	2.46 ± 0.28	153.04
Convolutional Network	2.31 ± 0.30	140.31

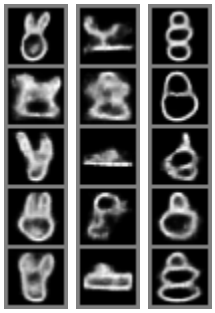
Lower FID scores imply more diverse samples, and higher IS scores generally yield higher quality in individual images [1].

4. **Take the best-performing VAE architecture and implement a conditional VAE. Generate and visualize five new samples from each rabbit, yoga, and snowman category. Using the classifier you trained in Assignment 1, predict the labels of the generated samples. Comment on the confidence of the classifier.**

We did this section for both the RNN and the convolutional network. For this section, we also implemented beta-VAE and experimented with different beta values (1, 2, 4). We see that generation quality declined with increasing beta value. For conditioning, we used an embedding layer of Pytorch with an embedding dimension of 16. We concatenated this vector to the output of the encoder before providing it to the linear layers that produce the mean and standard deviation vectors. For the classifier, we directly took our best-forming architecture and weights from the first assignment (please check the code for details).

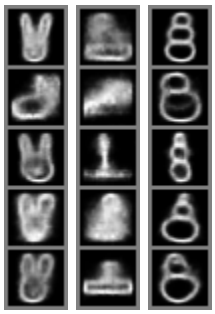
RNN results are as follows, respectively, for rabbit, snowman, and yoga labels:

Beta = 1



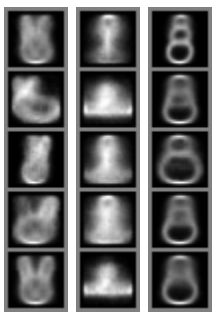
Classifier predictions for generated 'rabbit':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 0.992
Sample 2: Predicted as 'Unknown' (index 2), Confidence: 0.568
Sample 3: Predicted as 'rabbit' (index 0), Confidence: 1.000
Sample 4: Predicted as 'rabbit' (index 0), Confidence: 0.993
Sample 5: Predicted as 'rabbit' (index 0), Confidence: 1.000
Classifier predictions for generated 'yoga':
Sample 1: Predicted as 'yoga' (index 1), Confidence: 0.996
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.913
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.996
Sample 4: Predicted as 'yoga' (index 1), Confidence: 0.704
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.887
Classifier predictions for generated 'snowman':
Sample 1: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 2: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.412
Sample 4: Predicted as 'snowman' (index 3), Confidence: 0.439
Sample 5: Predicted as 'snowman' (index 3), Confidence: 0.999

Beta = 2



Classifier predictions for generated 'rabbit':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 1.000
Sample 2: Predicted as 'rabbit' (index 0), Confidence: 1.000
Sample 3: Predicted as 'rabbit' (index 0), Confidence: 1.000
Sample 4: Predicted as 'rabbit' (index 0), Confidence: 0.998
Sample 5: Predicted as 'rabbit' (index 0), Confidence: 0.999
Classifier predictions for generated 'yoga':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 0.408
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.510
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.999
Sample 4: Predicted as 'yoga' (index 1), Confidence: 0.971
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.997
Classifier predictions for generated 'snowman':
Sample 1: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 2: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 3: Predicted as 'snowman' (index 3), Confidence: 0.995
Sample 4: Predicted as 'snowman' (index 3), Confidence: 0.999
Sample 5: Predicted as 'snowman' (index 3), Confidence: 0.993

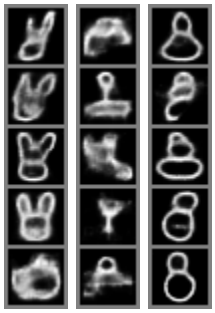
Beta = 4



Classifier predictions for generated 'rabbit':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 0.999
Sample 2: Predicted as 'rabbit' (index 0), Confidence: 0.998
Sample 3: Predicted as 'rabbit' (index 0), Confidence: 0.999
Sample 4: Predicted as 'rabbit' (index 0), Confidence: 0.994
Sample 5: Predicted as 'rabbit' (index 0), Confidence: 1.000
Classifier predictions for generated 'yoga':
Sample 1: Predicted as 'yoga' (index 1), Confidence: 0.971
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.935
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.976
Sample 4: Predicted as 'yoga' (index 1), Confidence: 0.932
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.963
Classifier predictions for generated 'snowman':
Sample 1: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 2: Predicted as 'snowman' (index 3), Confidence: 0.999
Sample 3: Predicted as 'snowman' (index 3), Confidence: 0.998
Sample 4: Predicted as 'snowman' (index 3), Confidence: 0.998
Sample 5: Predicted as 'snowman' (index 3), Confidence: 0.998

Convolution results are as follows, respectively, for rabbit, snowman, and yoga labels:

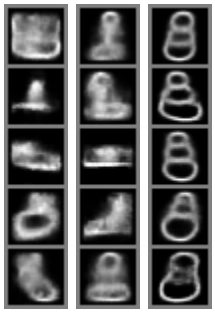
Beta = 1



Classifier predictions for generated 'rabbit':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 0.999
Sample 2: Predicted as 'rabbit' (index 0), Confidence: 0.996
Sample 3: Predicted as 'rabbit' (index 0), Confidence: 1.000
Sample 4: Predicted as 'rabbit' (index 0), Confidence: 1.000
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.623
Classifier predictions for generated 'yoga':
Sample 1: Predicted as 'Unknown' (index 2), Confidence: 0.756
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.999

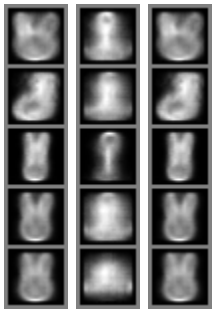
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.604
Sample 4: Predicted as 'yoga' (index 1), Confidence: 0.995
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.966
Classifier predictions for generated 'snowman':
Sample 1: Predicted as 'snowman' (index 3), Confidence: 0.985
Sample 2: Predicted as 'snowman' (index 3), Confidence: 0.786
Sample 3: Predicted as 'snowman' (index 3), Confidence: 0.997
Sample 4: Predicted as 'snowman' (index 3), Confidence: 0.993
Sample 5: Predicted as 'snowman' (index 3), Confidence: 1.000

Beta = 2



Classifier predictions for generated 'rabbit':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 0.935
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.560
Sample 3: Predicted as 'rabbit' (index 0), Confidence: 0.971
Sample 4: Predicted as 'rabbit' (index 0), Confidence: 0.997
Sample 5: Predicted as 'rabbit' (index 0), Confidence: 1.000
Classifier predictions for generated 'yoga':
Sample 1: Predicted as 'yoga' (index 1), Confidence: 0.998
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.571
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.982
Sample 4: Predicted as 'yoga' (index 1), Confidence: 0.551
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.996
Classifier predictions for generated 'snowman':
Sample 1: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 2: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 3: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 4: Predicted as 'snowman' (index 3), Confidence: 0.992
Sample 5: Predicted as 'snowman' (index 3), Confidence: 0.999

Beta = 4



Classifier predictions for generated 'rabbit':
Sample 1: Predicted as 'rabbit' (index 0), Confidence: 0.996
Sample 2: Predicted as 'rabbit' (index 0), Confidence: 0.988
Sample 3: Predicted as 'rabbit' (index 0), Confidence: 0.999
Sample 4: Predicted as 'rabbit' (index 0), Confidence: 0.999
Sample 5: Predicted as 'rabbit' (index 0), Confidence: 0.998
Classifier predictions for generated 'yoga':
Sample 1: Predicted as 'yoga' (index 1), Confidence: 0.950
Sample 2: Predicted as 'yoga' (index 1), Confidence: 0.944
Sample 3: Predicted as 'yoga' (index 1), Confidence: 0.988
Sample 4: Predicted as 'yoga' (index 1), Confidence: 0.827
Sample 5: Predicted as 'yoga' (index 1), Confidence: 0.866
Classifier predictions for generated 'snowman':
Sample 1: Predicted as 'snowman' (index 3), Confidence: 0.998
Sample 2: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 3: Predicted as 'snowman' (index 3), Confidence: 0.998
Sample 4: Predicted as 'snowman' (index 3), Confidence: 1.000
Sample 5: Predicted as 'snowman' (index 3), Confidence: 1.000

The classifier mostly has a high confidence when it has the slightest understanding of the overall image (as can be seen from blurry images of beta 4). Moreover, when it mispredicts, it produces a small confidence value (between 0.4 and 0.6), which can be helpful to detect the images that were not successfully generated by the model.

REFERENCES

1) Ho, J., & Salimans, T. Classifier-Free Diffusion Guidance. In NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications.