Programming

Problem 1:

VAE Training Parameter Details

1. **Encoder structure**

   Four Linear layers of the following input and output size
   - nn.Linear(200, 128)
   - nn.Linear(128, 64)
   - nn.Linear(64, 32)
   - nn.Linear(32, 16)

   Activation layer: ReLU

   Batchnorm1d between linear layers

   number of trainable parameters: 24768
2. **Decoder structure**

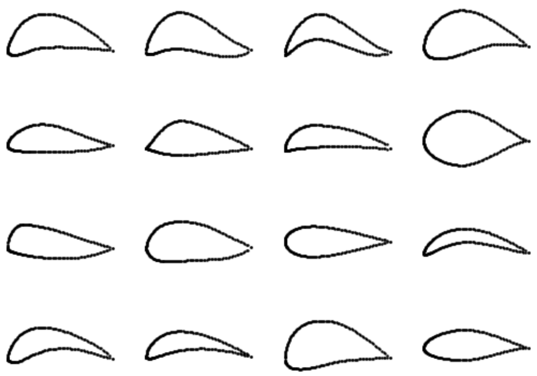
   Four Linear layers of the following input and output size
   - nn.Linear(16, 32)
   - nn.Linear(32, 64)
   - nn.Linear(64, 128)
   - nn.Linear(128, 200)

   Activation layer: ReLU, at the end of decoder tanh is used

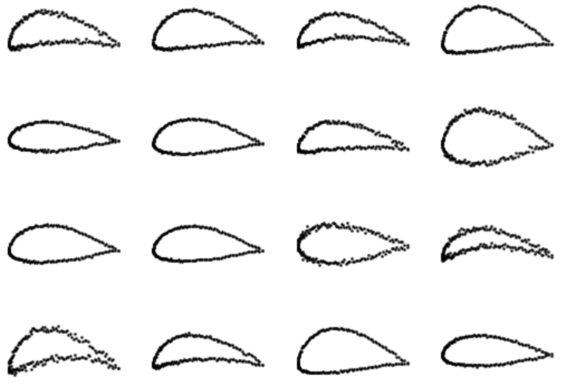
   Batchnorm1d between linear layers

   number of trainable parameters: 24324
3. **Learning rate:** 0.001
4. **Batch size:** 512
5. **Number of epochs:** 2000
6. **Alpha for kl divergence loss weightage** = 0.005
7. **Loss function** = MSELoss() + alpha*KL_divergence, {reduction used is sum}
8. **Optimizer** = Adam()
9. **Learning rate scheduler** = StepLR(step_size=100, gamma=0.4)

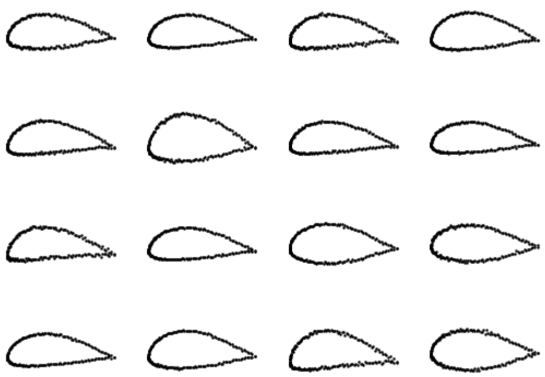
Results



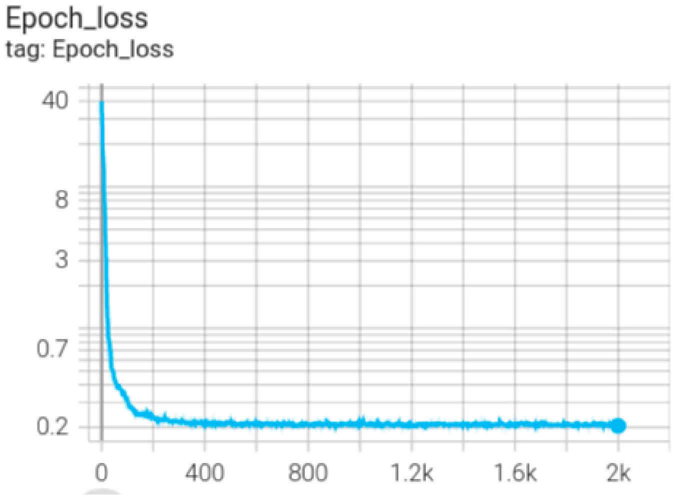
Real airfoils



Reconstructed airfoils



Synthesized airfoils



Training loss vs epoch

GAN

Problem 1:

GAN Training Parameter Details

10. **Generator structure**

    Four Linear layers of the following input and output size

    - Linear(in_features=16, out_features=32, bias=True)
    - Linear(in_features=32, out_features=64, bias=True)
    - Linear(in_features=64, out_features=128, bias=True)
    - Linear(in_features=128, out_features=200, bias=True)

    Activation layer: ReLU

    number of trainable parameters: 36776

11. **Discriminator structure**

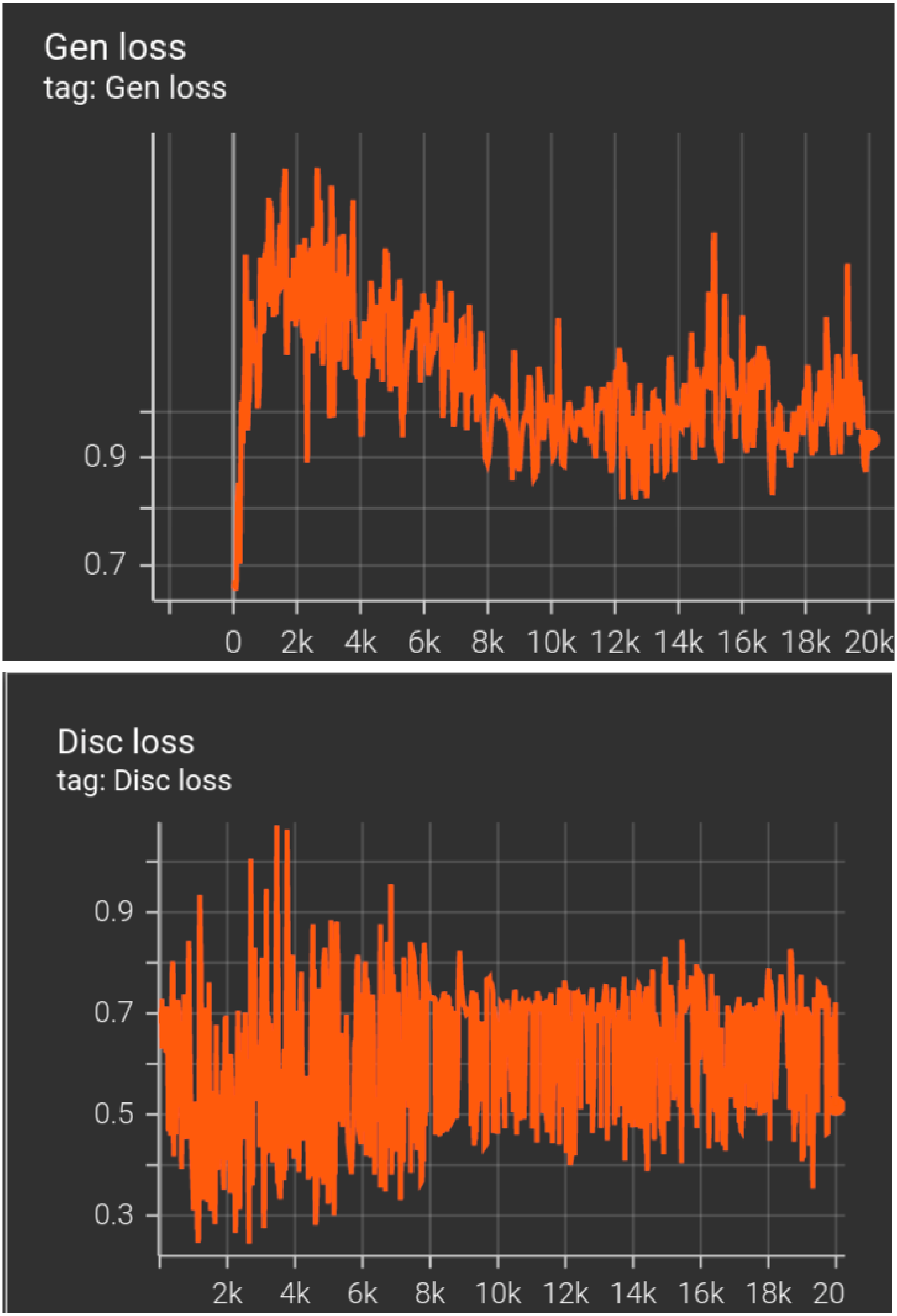
    Four Linear layers of the following input and output size

    - Linear(in_features=200, out_features=128, bias=True)
    - Linear(in_features=128, out_features=64, bias=True)
    - Linear(in_features=64, out_features=32, bias=True)
    - Linear(in_features=32, out_features=1, bias=True)

    Activation layer: ReLU,

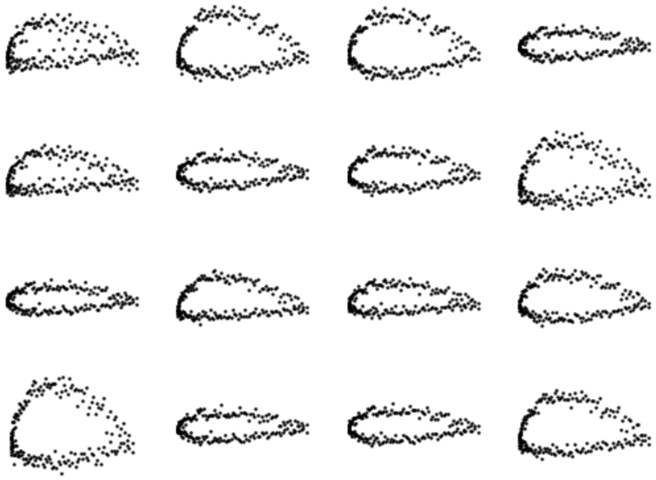
    number of trainable parameters: 36097

12. **Gen Learning rate:** 0.00015
13. **Disc learning rate:** 0.0002
14. **Batch size:** 512
15. **Number of epochs:** 5000
16. **Loss function** = BCEwithlogitsloss()
17. **Optimizer** = Adam() {for both gen and dis}



Loss vs number of iterations

Generated data from noise



Observation: The samples from gan are very noisy. The reason might be unlike VAE we do not have a reconstruction component in the loss, which compares each point generated individually in my code MSELoss. Also, VAE directly optimizes the ELBO function of MLE, while GAN is just classifying generated and real data. For gan, all we have is a classification loss which suggests if the data is fake or real. That being said the data from gans appear more diverse.