① $D_{KL}(q(x)\|p(x)) = -\int q(x)\log\left[\frac{p(x)}{q(x)}\right]dx \geq 0$

$= -\int q(x)\log p(x) + \int q(x)\log q(x)$ — eqn ②

$q(x) = N(\mu, \sigma)$ $\quad p(x) = N(0,1)$

$D_{KL} = \int -q(x)\log(p(x))dx + \int q(x)\log(q(x))dx$ — ①

2nd term in eqn ①

$-\int q(x)\log(q(x))dx$

$= E_{\text{expectation}}[\log N(\mu,\sigma)]$

$= E\left\{\log\left[(2\pi\sigma^2)^{-1/2}\exp\left(\frac{-1}{2}\left(\frac{x-\mu}{\sigma}\right)^2\right)\right]\right.$

$= E\left[\frac{-1}{2}\log(2\pi\sigma^2)\right] + E\left[\frac{-1}{2\sigma^2}(x-\mu)^2\right]$

$\Downarrow$ Const.

$= -\frac{1}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\left[E(x-\mu)^2\right]$

$E(x-\mu)^2 = \sigma^2$

$\int q(x)\log q(x) = -\frac{1}{2}\left[1 + \log(2\pi\sigma^2)\right]$ — ②

I^st term in equation (1).

$\int q(x) \log P(x)$ $\quad \sigma_0 = 1,\ \mu_0 = 0$

$$= \int \frac{1}{\sqrt{2\pi}\sigma} \times e^{-\left(\frac{x-\mu}{\sigma}\right)^2} \times \log \times \left[\frac{1}{\sqrt{2\pi}\sigma_0} \times e^{-\left(\frac{x-\mu_0}{\sigma_0}\right)^2}\right] dx.$$

$$= \frac{1}{\sqrt{2\pi}\sigma} \int \underbrace{e^{-(x-\mu_0)^2}}_{q(x)} \times \left[-\frac{1}{2}\log\left[2\pi\sigma_0^2\right] - \left(\frac{x-\mu_0}{\sigma_0}\right)^2\right] dx.$$

$$= \int_{-\infty}^{\infty} -\frac{1}{2}\log\left[2\pi\sigma_0^2\right] q(x)\, dx. + \frac{q(x)}{2} \times \int\left[-\left(\frac{x-\mu_0}{\sigma_0}\right)^2\right]$$

$$= -\frac{1}{2}\log\left[2\pi\sigma_0^2\right] - \frac{q(x)}{\sigma_0^2}\int\left[x^2 - 2\mu_0 x + \mu_0^2\right] dx.$$

$$= -1/2 \log\left[2\pi\sigma_0^2\right] - \frac{1}{2\sigma_0^2}\left[\int q(x)x^2 - 2\mu_0 \int q(x)x\,dx + \left[q(x)\mu_0^2\right] dx.\right]$$

$$\int q(x)x^2 = \sigma^2 + \mu^2 \;;\; \int q(x)x\,dx = \mu.$$

$$= -1/2 \log\left[2\pi\sigma_0^2\right] - \frac{1}{2\sigma_0^2}\left[\sigma^2 - 2\mu_0\mu + \mu^2\right]$$

$$= -1/2\left[\log(2\pi\sigma_0^2) + 1/\sigma_0^2\left[\sigma^2 + (\mu-\mu_0)^2\right]\right.$$

$\left.\sigma_0 = 1,\ \mu_0 = 0\right]$

$$= -1/2\left[\log(2\pi) + 1\left[\sigma^2 + \mu^2\right]\right] \quad \text{—(3)}$$

Putting ② ③ back in ①

we get

$$= -\frac{1}{2}\left[\log(2\pi) + (\sigma^2 + \mu^2) - 1 - \log(2\pi\sigma^2)\right]$$

$$= -\frac{1}{2}\left[\log\frac{2\pi\sigma^2}{2\pi} + 1 - \sigma^2 - \mu^2\right]$$

$$= -\frac{1}{2}\left[\log(\sigma^2) + 1 - \sigma^2 - \mu^2\right]$$

b) When α is too large, the K.L divergence loss will give higher gradients to the network. So the K.L.D loss will try to force the 0-mean, I condition to encoder samples. So all the data sampled will have similar latent vector. So all the reconstructed data will be similar to each other. As the decoder after training is deterministic, if the sampled latent vectors are similar the reconstruction will yield similar data.

All reconstructed data will be similar.

5)

① PCA, as a linear operation, that transforms a higher dimensional data to a low dimensional data. While VAE, does the same but with non-linear between layers

② PCA always reduce data. In VAE, you can project data to even a higher dimension. This is more like a feature transformation.

③ In PCA there is no parameters learned, but VAE's can be trained to find the correct weights to encode the data.

Problem 2:-

(a) False.

We should update discriminator $k \geq 1$, times when we optimize generator once. If we do opposite as mentioned in the statement, generator becomes really better than discriminator, so every bad quality fake sample gets passed through the discriminator... So GAN's should always have a balance between discriminator & and generator which will be broken when we train according to the statement.

(b) True

$D(G(z))$ initially will be close to 0, because the generator will be very bad initially and discriminator will be good, so all the fake samples generated from Generator $G(z)$, will be close to 0, and will be classified as fake = 0, by the discriminator.

(c) Blue, non-saturating cost.

Initially as explained above the value of $D(G(z))$ will be close to 0, The figure shows that orange has a very low slope close to 0, initially while blue has a higher slope. So gradients will be higher for non-saturating cost

function blue. As gradients are sufficient the generator optimization will be faster. For the case of saturating orang cost, the optimizer will suffer from vanishing gradients

(d) **False**

If $D(G(z)) \approx 1$, it means all the fake samples are getting classified as Real by the discriminator. This can occur when discriminator is very or generator is very powerful. If discriminator is weak, even the bad data samples generated will be passed through. So we want a balance between generator and discriminator. Ideally both of them should be around 0.5. Generator & discriminator should be both powerful..

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**

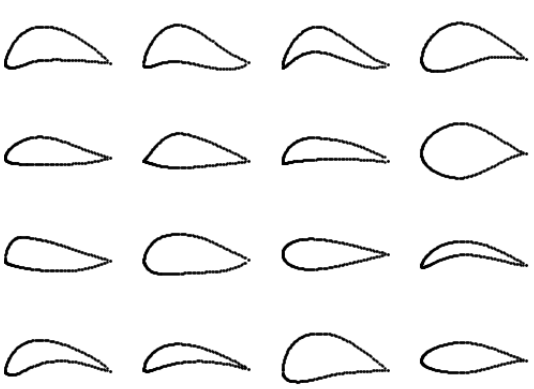
   Four Linear layers of the following input and output size
   - nn.Linear(16, 32)
   - nn.Linear(32, 64)
   - nn.Linear(64, 128)
   - nn.Linear(128, 200)

   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)

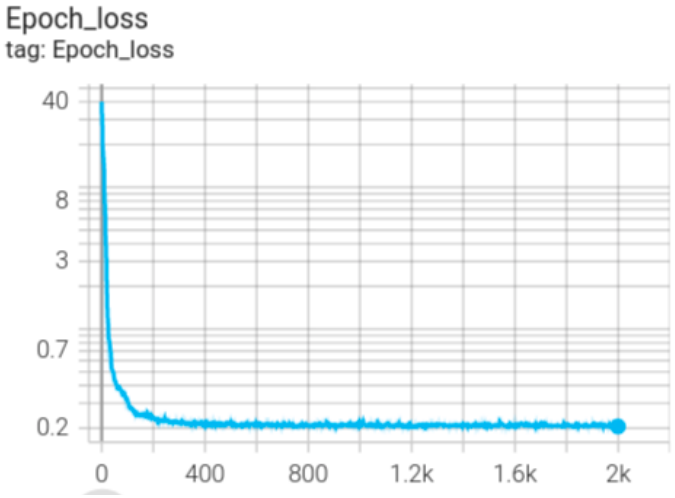
Results



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**

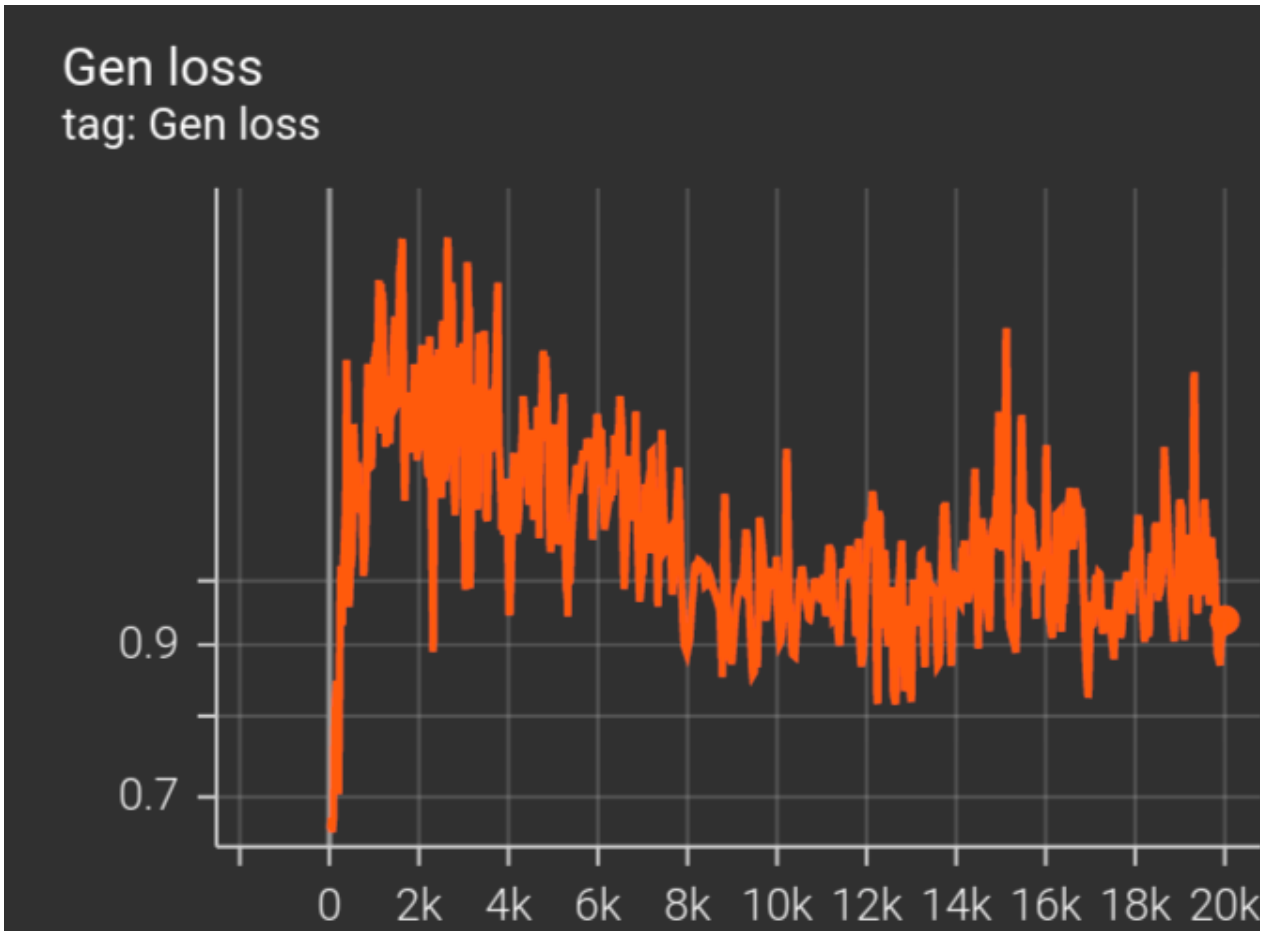
    Four Linear layers of the following input and output size

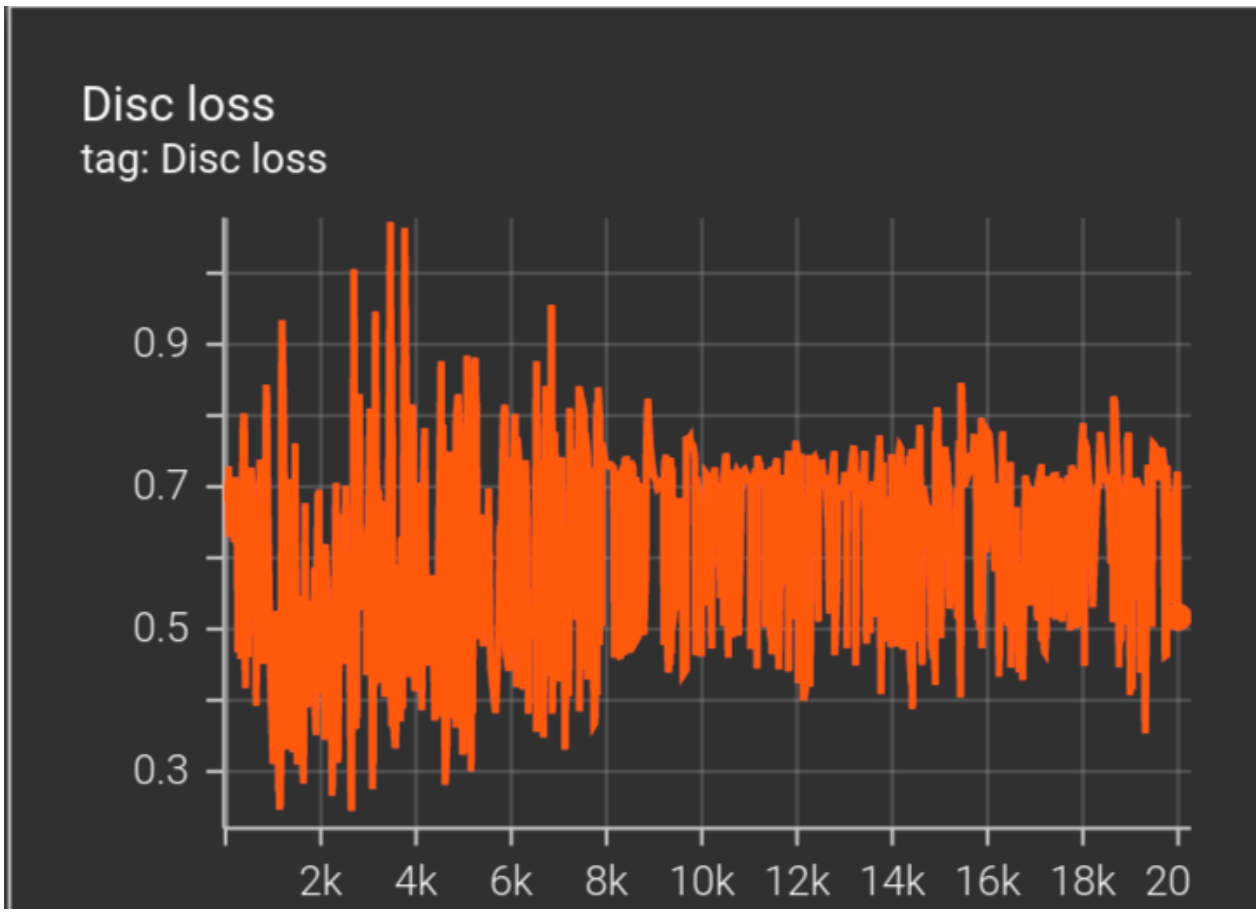
    - Linear(in_features=200, out_features=128, bias=True)
    - Linear(in_features=128, out_features=64, bias=True)
    - Linear(in_features=64, out_features=32, bias=True)
    - Linear(in_features=32, out_features=1, bias=True)

    Activation layer: ReLU,

    number of trainable parameters: 36097

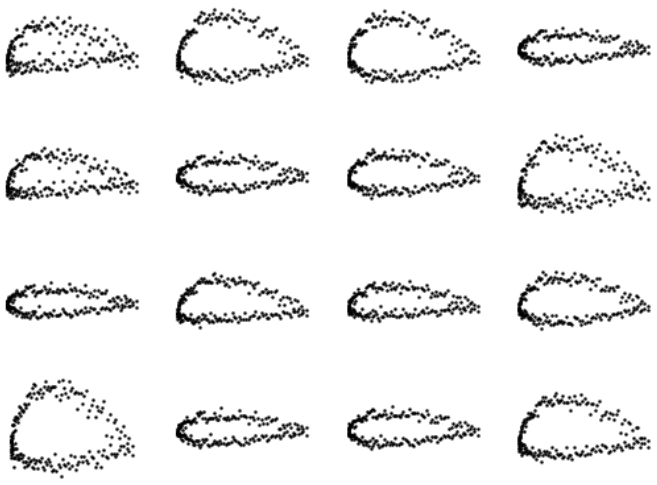
12. **Gen Learning rate:** 0.00015
13. **Disc learning rate:** 0.0002
14. **Batch size:** 512
15. **Number of epochs:** 5000
16. **Loss function** = BCEwithlogitsloss()
17. **Optimizer** = Adam() {for both gen and dis}





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.