Assignment (3)

24789 - Special Topics: Deep Learning for Engineers (Spring 2022)

Out Date: 2022/3/16 (Wed)
Due Date: 2022/3/30 (Wed) @ 11:59 pm EST

All exercises should be submitted to Gradescope. There are 2 assignment sections to submit in Gradescope. The first is **HW3**, where you submit a pdf file containing your answers to all theory exercises. The second is **HW3-programming**, where you submit your codes, saved model as well as a brief report for the programming exercise. For **HW3-programming**, you should submit a zip file as the following structure:

Submission file structure

/andrewID-HW3
/p1
***.py** (all the Python script files)
p1_vae_model.pth
p1_gan_model.pth
p1_report.pdf

You can refer to Python3 tutorial, Numpy documentation and PyTorch documentation while working on this assignment. Any deviations from the submission structure shown below would attract penalty to the assignment score. Please use Piazza for any questions on the assignment.

Theory Exercises (48 points)

PROBLEM 1

Variational Autoencoder (VAE) (24 points)

VAE contains two components: an encoder and a decoder. The former encodes high-dimensional sample x into a low-dimensional latent representation z. While the decoder takes as input the latent vector z and upsamples from representation domain to the original data domain $\tilde{x}$. The encoder and decoder is given as

$$z \sim \text{Enc}(x) = q(z|x),\ \tilde{x} \sim \text{Dec}(z) = p(\tilde{x}|z)$$

To regulate the encoder, VAE takes prior of latent distribution $p(z)$ into consideration. It assumes that $p(z) \sim \mathcal{N}(0, \mathbf{I})$ which follows an isotropic Gaussian distribution. And loss function of VAE is a combination of KL loss and mean square error given by

$$\begin{aligned}\mathcal{L}_{VAE} &= \mathcal{L}_{recon} + \alpha\mathcal{L}_{prior} \\ \mathcal{L}_{recon} &= ||\tilde{x} - x||_2 \\ \mathcal{L}_{prior} &= D_{KL}(q(z|x)||p(z))\end{aligned}$$

where α is the weight assigned to $\mathcal{L}_{prior}$.

$\mathcal{L}_{recon}$ measures how well the reconstructed data $\tilde{x}$ is comparing the original x by mean square error and $\mathcal{L}_{prior}$ is the KL divergence which measures the difference between encoded representation vectors and prior Gaussian distribution.

And KL divergence is given as

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

a) Let's assume the dimension of z is 1, meaning $p(z) \sim \mathcal{N}(0, 1)$. And your encoder predict the mean μ and standard deviation σ of $q(z|x)$. Derive the close form of $D_{KL}(q(z|x)||p(z))$ with μ, σ. [8 points]

b) If α is too large, what would you expect for the reconstructed results $\tilde{x}$ from a well-trained VAE model? (hint: 1. With large α, all input x will have similar encoding z; 2. you can also test your assumption by playing with α in your coding exercise.) [8 points]

c) Name 2 aspects where VAE is different from Principal Component Analysis (PCA)? [8 points]

PROBLEM 2

Generative Adversarial Network (GAN) (24 points)

Fig.1 illustrates loss function of the generator against the output of the discriminator when given a generated image $G(z)$. Concerning the discriminator's output, we consider that 0 means that the discriminator thinks the input "is generated by G", and 1 means that the discriminator thinks the input "comes from the real data".

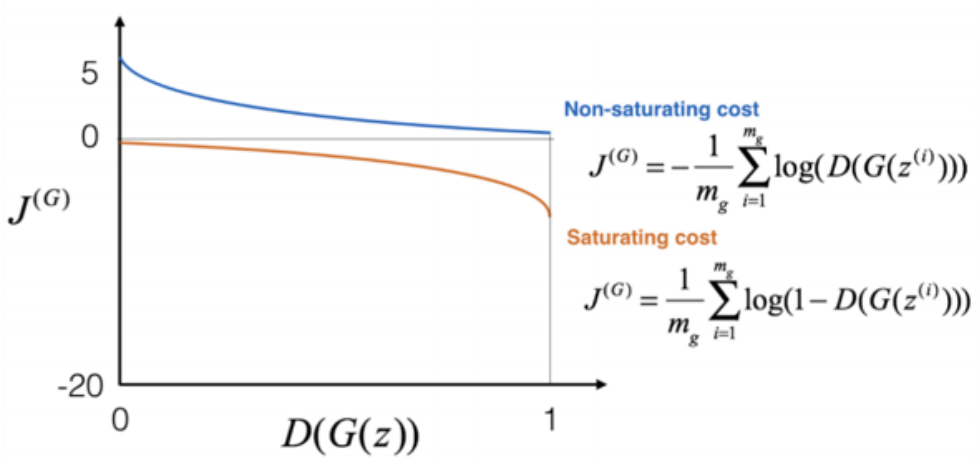


Fig. 1: Generator loss vs. Discriminator output

a) (True/False) A common method to accelerate the training of Generative Adversarial Networks (GANs) is to update the Generator $k(\geq 1)$ times for every 1 time you update the Discriminator. Briefly explain your answer. [6 points]

Consider the graph in Fig. 1 representing the training procedure of a GAN to answer the following questions:

b) Early in the training, is the value of $D(G(z))$ closer to 0 or closer to 1? Explain. [6 points]

c) Two cost functions are presented in Fig. 1, which one would you choose to train your GAN? Briefly explain your answer. (hint: you may want your model to learn fast at the early stage during training) [6 points]

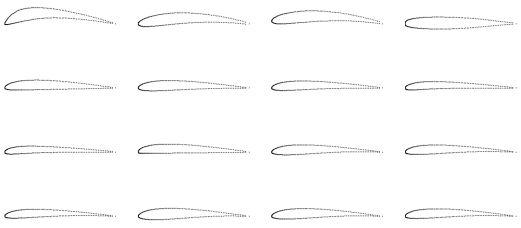
d) (True/False) You know that your GAN is trained when $D(G(z))$ is close to 1. Briefly explain your answer. [6 points]

Programming Exercises (52 points)

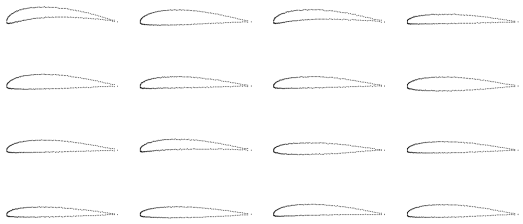
PROBLEM 1

Airfoil Generation via VAE & GAN (52 points)

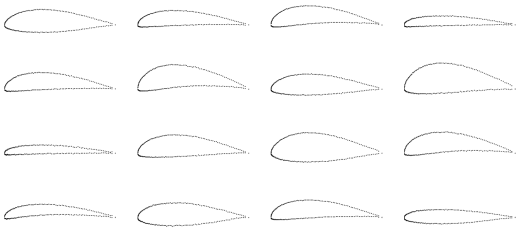
UIUC Airfoil Coordinates Database includes coordinates for nearly 1,600 airfoils. Since number of points in each sample differ, we first process all airfoils to have 200 points and share the same x coordinates via spline interpolation. Also, all y coordinates are rescaled to [-1,1]. The processed airfoils are shown in Fig. 2a. Therefore, only y coordinates of each airfoil is used to train and test generative models.



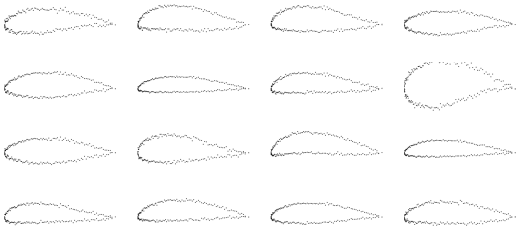
(a) Real airfoils



(b) Reconstructed airfoils from VAE



(c) Synthesized airfoils from VAE



(d) Synthesized airfoils from GAN

Fig. 2: Visualization for Coding Problem 2

In this problem, you are asked to build, train and test a VAE model as well as a GAN model via PyTorch. You are encouraged to implement from the starter code. However, you are not strictly restricted to that as long as you submit all your code and complete report. Specifically, in vae.py file:

- Implement Encoder() class which takes y coordinates of airfoils as input and encodes that into a low-dimensional vector.
- Implement Decoder() class which takes the encoding vector as input and output the y coordinates of airfoils.
- Implement VAE() class which contains the encoder and decoder.

Also, you need to modify train_vae.py, which is used to train and test your VAE model:

- Define you loss function and calculate loss at each training iteration
- Tune hyperparameters both in the model and optimizer to improve the performance

Similarly, in gan.py file:

- Implement Discriminator() class which takes y coordinates of airfoils as input and predict whether airfoils are real or fake.

- Implement Generator() class which takes the normal distributed noise as input and synthesizes the y coordinates of airfoils.

Also, you need to modify train_gan.py, which is used to train and test your GAN model:

- Define you loss function and calculate loss at each training iteration
- Tune hyperparameters both in the model and optimizer to improve the performance

Also, you can write helper functions in utils.py, like generate random noises and generate labels for discriminator in GAN. Besides, since GAN loss is not directly implemented in PyTorch, you may also want to customize a GAN loss function or class.

In your report, you should include:

- Structure of your VAE and GAN (eg. number of layers, number of neurons, etc.)
- Hyper-parameters (eg.learning rate, optimizer, learning rate decay, etc.)
- Visualization of training loss vs. epoch (both VAE and GAN)
- For VAE, visualization of reconstructed airfoils and corresponding real airfoils as shown in Fig. 2b and Fig. 2a. Also, visualization of synthesized airfoils from random noise should be included too, as shown in Fig. 2c (starter code already has this visualization script)
- For GAN, visualization of synthesized airfoils from random noise as shown in Fig. 2d (starter code already has this visualization script)
- Compare the synthesized airfoils from VAE and GAN, describe your observation and give a brief explanation.

You can refer to this Github repo for implementation of VAE: `https://github.com/pytorch/examples/tree/master/vae`, and `https://github.com/eriklindernoren/PyTorch-GAN/blob/master/implementations/gan/gan.py` for implementation of GAN.
