Assignment (2)

24789 - Special Topics: Deep Learning for Engineers (Spring 2020)

Author: Yuyang Wang, Guilin Zhang, Amir Barati Farimani
Out Date: 2020/2/24 (Mon)
Due Date: 2020/3/16 (Mon) @ 11:59 pm EST

All exercises should be submitted to Gradescope. There are 2 assignment sections to submit in Gradescope. The first is **HW2**, where you submit a pdf file containing your answers to all theory exercises. The second is **HW2-programming**, where you submit your codes, saved model as well as a brief report for 2 programming exercises. For **HW2-programming**, you should submit a zip file as the following structure:

Submission file structure

/andrewID-HW2
- **/p1**
  - ***.py** (all the Python script files)
  - **p1_model.ckpt**
  - **p1_report.pdf**
- **/p2**
  - ***.py** (all the Python script files)
  - **p2_model.ckpt**
  - **p2_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 (50 points)

PROBLEM 1

Recurrent Neural Network (RNN) [6 points]

Consider the following RNN, which has a scalar input at the first time step, makes a scalar prediction at the last time step, and uses a shifted logistic activation function

$$\phi(z) = \sigma(z) - 0.5$$
$$\text{where, } \sigma(x) = \frac{1}{1+e^{-x}}$$

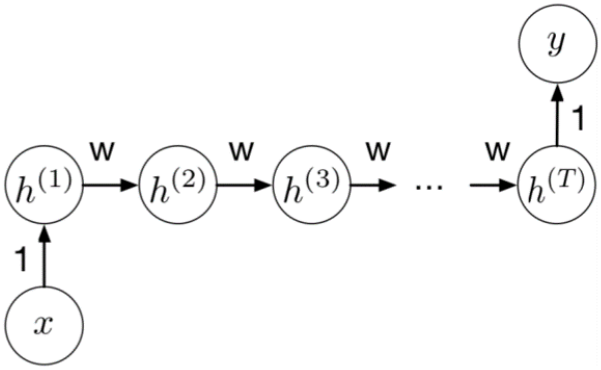


Fig. 1: RNN

a) Write the formula for the derivative ∇h_t as a function of ∇h_{t+1}, for t < T. (Use z_t to denote the input to the activation function at time t. You may write your answer in terms of σ', i.e. you don't need to explicitly write out the derivative of σ.) [3 points]

b) Suppose the input to the network is $x = 0$. Notice that $h_t = 0$ for all t. Based on your answer to part (a), determine the value α such that if $w < \alpha$, the gradient vanishes, while if $w > \alpha$, the gradient explodes. You may use the fact that $\sigma'(0) = 1/4$. [3 points]

PROBLEM 2

Long Short-term Memory (LSTM) (12 points)

Long Short Term Memory networks (LSTM) are a special kind of RNN, capable of learning long-term dependencies. They were first introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work. LSTMs are explicitly designed to solve the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn. Fig. 2 shows the structure of LSTM cell.

The symbols and expressions of each components are given below, where f_t, c_t, o_t and h_t represent forget gate, cell state, output gate and hidden state respectively. Here, $\odot$ represents element-wise multiplication.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$h_t = o_t \odot \tanh(c_t)$$

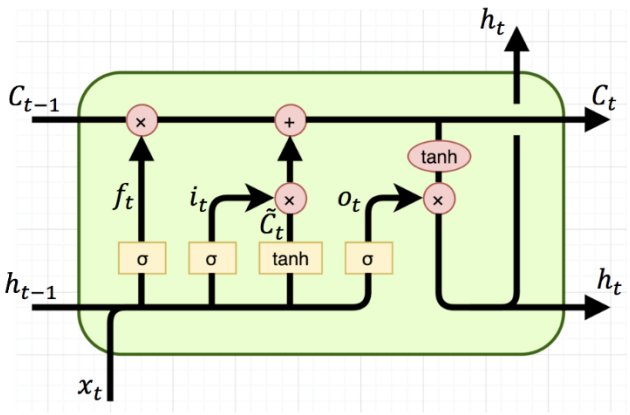


Fig. 2: LSTM

Here σ and tanh are Sigmoid and Hyperbolic Tangent functions given as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

a) (True/False) If x_t is the 0 vector, then $h_t = h_{t-1}$. Explain why. [3 points]

b) (True/False) If f_t is very small or zero, then error will not be back-propagated to earlier time steps. Explain why. [3 points]

c) (True/False) The entries of f_t, i_t, o_t are non-negative. Explain why. [3 points]

d) (True/False) f_t, i_t, o_t can be viewed as probability distributions. (i.e., their entries are non-negative and their entries sum to 1.) Explain why. [3 points]

PROBLEM 3

Calculate LSTM (9 points)

The symbols and expressions of each components in LSTM are given below, where f_t, c_t, o_t and h_t represent forget gate, cell state, output gate and hidden state respectively. Here, $\odot$ represents element-wise multiplication.

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$
$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i)$$
$$\tilde{c}_t = \tanh(W_c x_t + U_c h_{t-1} + b_c)$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o)$$
$$h_t = o_t \odot \tanh(c_t)$$

Here σ and tanh are Sigmoid and Hyperbolic Tangent functions given as:

$$\sigma(x) = \frac{1}{1+e^{-x}}$$
$$\tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Let's define the parameters as follows:

$$W_f = \begin{bmatrix} 1 & 2 \end{bmatrix}, U_f = \begin{bmatrix} 0.5 & -1 \end{bmatrix}, b_f = [0.2]$$
$$W_i = \begin{bmatrix} -1 & 0 \end{bmatrix}, U_i = \begin{bmatrix} 2 & -2 \end{bmatrix}, b_i = [-0.1]$$
$$W_c = \begin{bmatrix} 1 & 2 \end{bmatrix}, U_c = \begin{bmatrix} 1.5 & 0.5 \end{bmatrix}, b_c = [0.5]$$
$$W_o = \begin{bmatrix} 3 & 0 \end{bmatrix}, U_o = \begin{bmatrix} -1 & -0.5 \end{bmatrix}, b_o = [0.8]$$

And the input data:

$$x_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \text{ with label: } y_0 = 0.5$$
$$x_1 = \begin{bmatrix} 0.5 \\ -1 \end{bmatrix}, \text{ with label: } y_1 = 0.8$$

The structure of LSTM model in this problem is shown in Fig. 3

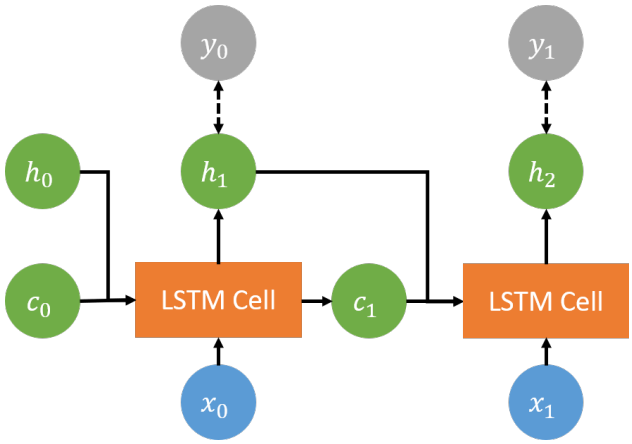


Fig. 3: 2-step LSTM

a) What is the dimension of f_t, i_t, o_t, h_t [2 points]

b) Assume the initial h_0, c_0 are both zero vectors. Calculate the output of the model h_1, h_2 [4 points]

c) Assume the problem we are dealing with is a regression problem. Calculate the MSE loss between current prediction and the ground truth label. [3 points]

PROBLEM 4

Variational Autoencoder (VAE) (11 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

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

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 $p(z|x)$. Derive the close form of $D_{KL}(q(z|x)||p(x))$ with μ, σ. [4 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.) [3 points]

c) Name 2 aspects where VAE is different from Principal Component Analysis (PCA)? [4 points]

PROBLEM 5

Generative Adversarial Network (GAN) (12 points)

Fig.4 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".

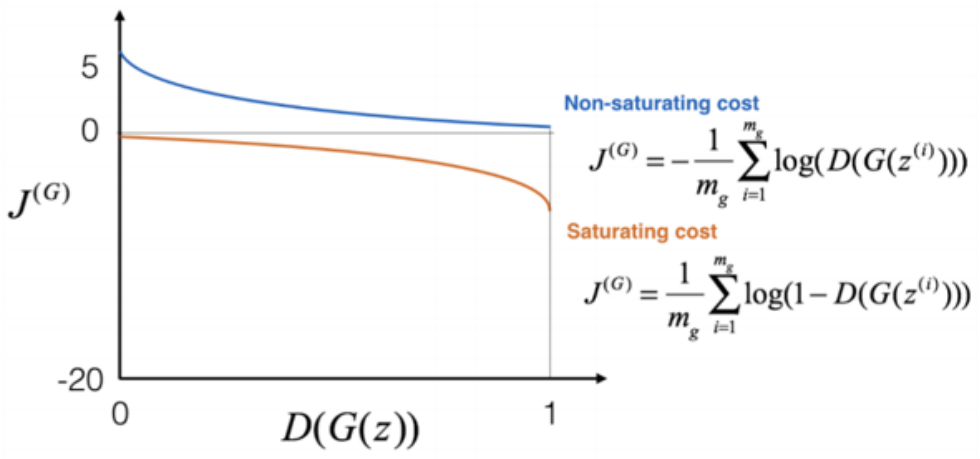


Fig. 4: Generator loss vs. Discriminator output

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. [3 points]

Consider the graph in Fig. 4 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. [3 points]

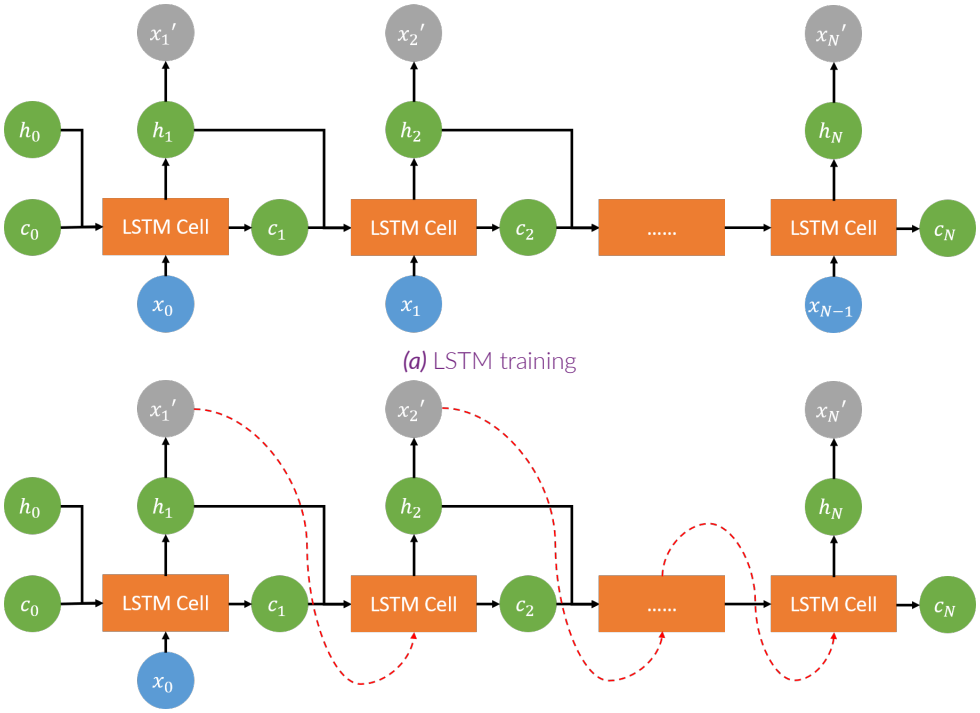
c) Two cost functions are presented in Fig. 4, which one would you choose to train your GAN? Briefly explain your answer. (hint: you may want your model to learn fast at the early stage during training) [3 points]

d) (True/False) You know that your GAN is trained when $D(G(z))$ is close to 1. Briefly explain your answer. [3 points]

Programming Exercises (50 points)

PROBLEM 1

LSTM in Fluid (25 points)



(a) LSTM training

(b) LSTM testing

Fig. 5: LSTM feed-forward pass

Hagen-Poiseuille flow, first studied experimentally by G. Hagen in 1839 and J. L. Poiseuille in 1940, is flow through a straight circular pipe. We assume that the flow is incompressible, pressure-driven, and one-dimensional along the axis of the pipe with a no-slip condition at the pipe walls. At steady-state, the velocity profile is a parabola.

If we assume that the fluid is initially at rest and then set in motion by a pressure difference at the ends of the pipe, then there is a period of time where the velocity profiles are developing until they reach the final steady-state parabolic form. These developing velocity profiles can be represented as a time series. Eq. 1 is the simplified form of Navier-Stokes for this time-dependent part of Hagen-Poiseuille flow. Here, G represents the axial pressure gradient, ρ is the fluid density, v is the fluid viscosity, and ν represents the fluid velocity along the axis of the pipe. Eq. 2 describes the solution to Eq. 1.

$$\frac{\partial u}{\partial t} = \frac{G}{\rho} + \nu\left(\frac{\partial^2 u}{\partial r^2} + \frac{1}{r}\frac{\partial u}{\partial r}\right) \tag{1}$$

$$u(r,t) = \frac{G}{4\mu}(R^2 - r^2) - \frac{2GR^2}{\mu}\sum_{n=1}^{\infty}\frac{J_0(\lambda_n \frac{r}{R})}{\lambda_n^3 J_1(\lambda_n)}\exp\left(-\lambda_n^2\frac{\nu t}{R^2}\right) \tag{2}$$

We've already generate the Hagen-Poiseuille flow dataset with different boundary conditions. We thank Nina Prakash for providing her code of generating Hagen-Poiseuille flow data. Each sample in the dataset contains 20 timesteps and velocity on 17 points. And the dataset is split into training set and testing set which contains 40500 and 10125 samples respectively. For both dataset, the first timestep represents flow conditions including diameter D axial pressure gradient G, viscosity ν and fluid density ρ. In training set, your input is of dimension (batch_size, 19, 17) and ground truth us of dimension (batch_size, 19, 17), meaning your model takes input of

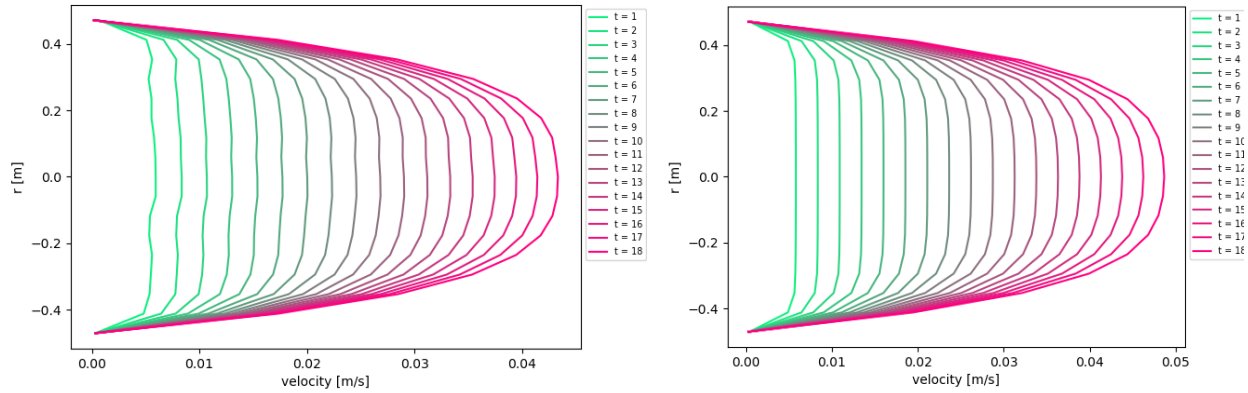
the ground truth velocity at each time step and predicts the velocity on next time step. However, for test set, the dimension of the input data is (batch_size, 17) and ground truth is (batch_size, 19, 17). As shown in Fig. 5, where x_t is the ground truth input and x'_t is the prediction. This is because during testing, your model only takes the flow conditions (first time step) as input and predict the whole process. The dataset and dataloader is already defined in the starter codes. You can tune the batch size if that helps your training.

In this problem, you are asked to build, train and test a LSTM 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 lstm.py file

- Build your LSTM model in __init__ function. Here we recommend nn.LSTMCell() instead of nn.LSTM() since the former is a lower-level implementation which is flexible,
- Define the feed forward pass in training in forward() function
- Define the feed forward pass in training in test() function

Also, you need to modify train_lstm.py, which is used to train and test your LSTM model

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



(a) Prediction using LSTM

(b) Ground trruth

Fig. 6: LSTM visualization

In your report, you should include:

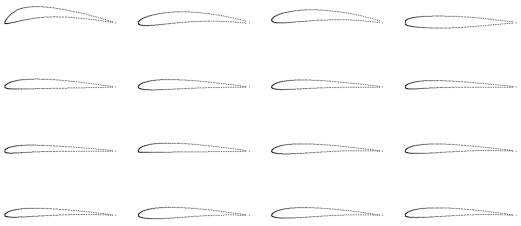
- Structure of your model (eg. number of layers, number of neurons, etc.)
- Hyper-parameters (eg.learning rate, optimizer, learning rate decay, etc.)
- Visualization of training loss vs. epoch
- Both L1 and L2 error of trained model on test set
- Visualization of prediction of flow velocity comparing to the ground truth, as shown in Fig. 6

For implementation of LSTM in PyTorch, you can refer to this repo: `https://github.com/pytorch/examples/tree/master/time_sequence_prediction`.

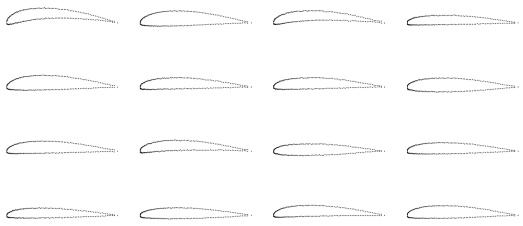
PROBLEM 2

Airfoil Generation via VAE & GAN (25 points)

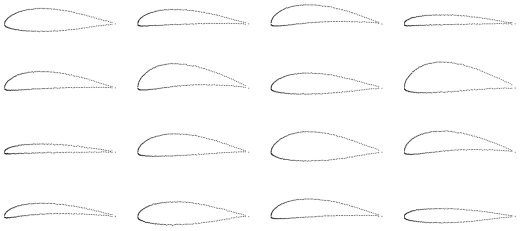
UIUC Airfoil Coordinates Database includes coordinates for nearly 1,600 airfoils. Since number of points in each sample differ, we first process all airfoils to have 200 points and share the same x coordinates via spline interpolation. Also, all y coordinates are rescaled to [-1,1]. The processed airfoils are shown in Fig. 7a. Therefore, only y coordinates of each airfoil is used to train and test generative models.



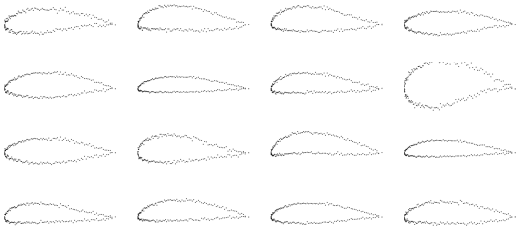
(a) Real airfoils



(b) Reconstructed airfoils from VAE



(c) Synthesized airfoils from VAE



(d) Synthesized airfoils from GAN

Fig. 7: 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. 7b and Fig. 7a. Also, visualization of synthesized airfoils from random noise should be included too, as shown in Fig. 7c (starter code already has this visualization script)
- For GAN, visualization of synthesized airfoils from random noise as shown in Fig. 7d (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.
