EX 3 – VAE, GAN, Transformer

Due date: 10/16/24 at 11:59 pm.

Please read the submission guidelines before you start.

Theory [50 pts]

1. [40pts] Consider the following 3 distance measures between the distributions p and q:

   - KL (Kullback–Leibler Divergence):

     $$D_{KL}(p||q) = \int_{x \in X} p(x) log \frac{p(x)}{q(x)} dx$$

     Note that it is always either positive, or 0 iff $\forall x \in X, p(x) = q(x)$.

     One of its disadvantages is that it is asymmetric.

   - JS (Jensen-Shannon Divergence) – a symmetric (and smoother) version of the KL divergence:

     $$D_{JS}(p||q) = \frac{1}{2} D_{KL}\left(p||\frac{p+q}{2}\right) + \frac{1}{2} D_{KL}\left(q||\frac{p+q}{2}\right)$$

   - Wasserstein Distance (also known as "Earth Mover distance), defined for continuous domain:

     $$W(p,q) = \inf_{\gamma \sim \prod(p,q)} \sum_{x,y} \gamma(x,y) \|x - y\| = \inf_{\gamma \sim \prod(p,q)} E_{(x,y) \sim \gamma}[||x - y||]$$

     where $\prod(p,q)$ is the set of all possible joint distributions of p and q, γ describes a specific one, and x and y are values drawn from p and q respectively. Intuitively, it measures the minimal cost of "moving" p to become q.

   Suppose we have two 2-dimensional probability distributions, P and Q:

   - $\forall (x,y) \in P, x = 0 \ and \ y \sim U(0,1)$
   - $\forall (x,y) \in Q, x = \theta, 0 \leq \theta \leq 1, and \ y \sim U(0,1)$

   θ is a given constant.

   Obviously, when $\theta \neq 0$, there is no overlap between P and Q.

   a. [20pts] For the case where $\theta \neq 0$, calculate the distance between P and Q by each of the three measurements.

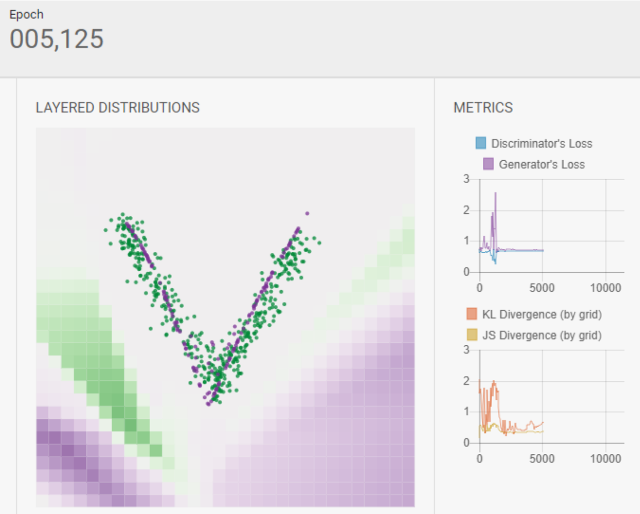
   b. [10pts] Repeat question a for the case where $\theta = 0$.

   c. [10pts] Following your answers, what is the advantage of the Wasserstein Distance over the previous two?

2. [10pts] Assume that we have access to an LSTM and a Transformer model. What is the computational complexity of generating a sequence of length N for both models? Explain your reasoning.

Practical [50 pts]

3. [5pts] To gain Intuition (and have fun), go to https://poloclub.github.io/ganlab/. Read the instructions, and then train a GAN by yourself. Draw a distribution and train the GAN on it. Submit in the PDF a screen shot of the result along with the epoch number and the graphs on the right. For example:



Comments:

- The model should be trained until convergence, and at least 5,000 epochs. Convergence can be seen visually and by the graphs.
- Do not remove the gradients in the picture. In the example, they just nullified after convergence.

4. [20pts] Here we address semi-supervised learning via a variational autoencoder. You will be implementing a part of the paper "Semi-supervised Learning with Deep Generative Models", by Kingsma et al.

Read the paper, and implement the M1 scheme, as described in Algorithm1, and detailed throughout the paper. It is based on a VAE for feature extraction, and then a (transductive) SVM for classification.

Implement the network suggested for MNIST, and apply it on the Fashion MNIST data set. Present the results for 100, 600, 1000 and 3000 labels, as they are presented in Table 1 in the paper. For simplicity, you may use a regular SVM (with a kernel of your choice). Take the latent representation of the labeled data as training, and then test it on the test set.

Comments:

- Please mention in the pdf which kernel did you use.
- Describe in the readme file how to train and test your model.
- Make sure to save the SVM model as well as the NN weights.
- For all labels, make sure you have an equal number of examples from each class. In addition, use a fixed seed value so the results would be consistent.

5. [25pts] Here you will get to play a bit with GANs and WGANs, by implementing a part of the paper "Improved Training of Wasserstein GANs", by Gulrajani et al.

   Read the paper, and implement DCGAN, WGAN (weight clipping), and WGAN-GP (gradient clipping). Choose two architectures from Table 1. Make appropriate modifications and apply it on the Fashion MNIST data set. Training should be supervised by the class label.

   a. Plot the loss functions. There should be two plots, one for each architecture. Each plot should have three loss curves (one for each method). No need for validation / testing loss curves, as there is no “ground-truth.”
   b. From the better performing architecture, select two generated images from each method for two labels (e.g. six images for T-shirt and six images for Dress). Also include two real images from the corresponding label.

   Comments:

   - Describe in the readme file how to train the model, and how to generate (with the trained weights) a new picture from a trained model for the three methods.
   - Write the number of failed convergences of your implementation before having a successful one (for all three methods).
   - Feel free to implement the architecture which involves residual layers (denoted WGAN-GP ResNet (ours) in the paper) but this is not required.
   - Make sure the code and the readme file support easy generation of new images.

Good Luck!