Computer Vision 2025
(CSE344/ CSE544/ ECE344/ ECE544)
Assignment-4

Max Marks: 45 + (5 Bonus) **Due Date**: 20-April-2025, 11:59 PM

Instructions

- Keep collaborations at high-level discussions. Copying/plagiarism will be dealt with strictly.
- Your submission should be a single zip file **HW[n]_Roll_Number.zip**. Include only the **relevant files** arranged with proper names. A single **.pdf report** explaining your codes with relevant graphs, visualization and solution to theory questions.
- Remember to **turn in** after uploading on Google Classroom. No justifications would be taken regarding this after the deadline.
- Start the assignment early. Resolve all your doubts from TAs during their office hours **two days before the deadline.**
- Kindly **document** your code. Don't forget to include all the necessary plots and images in your report.
- All [**BONUS**] questions, if any, are optional for all the students. As the name suggests, BONUS marks will be awarded to all the students who solve these questions.
- Please ensure that your submission includes all the code used to solve the questions. You must submit a separate Jupyter Notebook (.ipynb) for each question. For example, name the file for **Question 2 as Q2.ipynb** and for **Question 3 as Q3.ipynb**. *Bonus questions can be included in the Q1 notebook file.*
- For each of the programming questions **you must set the seed with your roll number for reproducibility**. Use `torch.manual_seed(seed: int)` to set the seed value, where *seed = #RollNumber*. If you roll number starts with a letter such as `MT` or `PhD`, you must use the ASCII values of the characters. Submitted codes without this will incur a **penalty**.

1. (15 points) **Theory**

   1. Consider a Generative Adversarial Network (GAN) defined by the following minimax objective:

   $$\min_G \max_D \; \mathbb{E}_{x\sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z\sim p_z}[\log(1 - D(G(z)))]$$

   which, after a change of variable and noting that samples from G induce a distribution $p_G(x)$, can be written as:

   $$\min_G \max_D \int_x \left(p_{\text{data}}(x) \log D(x) + p_G(x) \log(1 - D(x))\right) dx.$$

(a) (3 points) **Optimal Discriminator:**

(a) Show that for any fixed generator G (with induced density p_G), the function Conclude that the optimal discriminator is:

$$D_G^*(x) = \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}.$$

(b) (4 points) **Reduction to Jensen-Shannon Divergence:**

(b) By substituting the optimal discriminator $D_G^*(x)$ back into the original objective, show that the minimax game reduces to:

$$\min_G \left(2 \cdot JSD(p_{\text{data}}, p_G) - \log 4\right),$$

where the Jensen-Shannon divergence is defined by:

$$JSD(p_{\text{data}}, p_G) = \frac{1}{2} KL\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_G}{2}\right) + \frac{1}{2} KL\left(p_G \,\middle\|\, \frac{p_{\text{data}} + p_G}{2}\right),$$

and $KL(p\|q) = \mathbb{E}_{x\sim p} \log \frac{p(x)}{q(x)}$.

(c) (3 points) **Global Optimality:**

(c) Prove that the above expression is minimized if and only if $p_G(x) = p_{\text{data}}(x)$, and in this case, show that the optimal discriminator becomes:

$$D_G^*(x) = \frac{1}{2} \quad \text{for all } x.$$

Provide a detailed, step-by-step mathematical proof for each part, utilizing techniques from calculus, properties of logarithms, and divergence measures.

2. (5 points) Show that minimizing the KL divergence

$$D_{KL}(p_D \,\|\, p_\theta) = \mathbb{E}_{x\sim p_D}\left[\log p_D(x) - \log p_\theta(x)\right]$$

with respect to the model parameters θ is equivalent to maximizing the likelihood of the data under the model $p_\theta(x)$. In your proof, explain why the term $\mathbb{E}_{x\sim p_D}[\log p_D(x)]$ can be ignored in the optimization.

2. (10 points) **DC-GAN** (Deep Convolutional Generative Adversarial Network) is a type of Generative Adversarial Network (GAN) that uses deep convolutional layers to generate realistic images from random noise. Introduced in their paper Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, DCGANs consist of two neural networks a Generator and a Discriminator trained adversarially. The Generator learns to produce synthetic images that resemble real data, while the Discriminator learns to distinguish between real and generated images. You will build a DCGAN to generate MNIST handwritten digits, implementing both the Generator and Discriminator networks according to the architectural guidelines in the paper. Your implementation should follow the code structure provided and adhere to the specific architectural constraints that make DCGANs stable to train.

1. (4 points) Implement the Discriminator class following DCGAN architectural guidelines: use strided convolutions (no pooling layers), apply LeakyReLU activation (alpha=0.2) to all layers, use BatchNorm for all layers except the first convolutional layer, and end with a Sigmoid activation. The network should progressively downsample the input image to a single probability output.
2. (4 point) Implement the Generator class following DCGAN architectural guidelines: use transposed convolutions for upsampling (no pooling layers), apply ReLU activation to all layers except the output layer which uses Tanh, use BatchNorm for all layers except the output layer, and structure the network to progressively upsample from a random noise vector to a full image.
3. (2 points) Implement the weight initialization function that initializes all model weights from a normal distribution with mean 0.0 and standard deviation 0.02, as specified in the DCGAN paper for stable training.

3. (20 points) **StyleGAN3** is a state-of-the-art generative adversarial network developed by NVIDIA that excels at synthesizing high-quality, photorealistic images. It builds upon previous versions by addressing aliasing artifacts and offering improved stability, allowing for more precise control over the generated outputs. With its advanced latent space manipulation and style mixing capabilities.

   1. (3 points) **Image Generation from a Pretrained Model:** Implement the function that loads a pretrained StyleGAN3 network from a given URL and generates images based on a list of random seeds. In your implementation, modify the code to overlay a caption on each image—using PIL's drawing utilities—that clearly displays the seed number before saving the image. Use the following URL to load the pretrained model: Pretrained StyleGAN3 model URL.
   2. (5 points) **Interpolation Between Latent:** Implement function that not only generates images from two different seeds but also creates a smooth transition between them through latent space interpolation. Begin by loading the StyleGAN3 network and generating latent vectors for the two seeds, then use a linear interpolation method (e.g., torch.lerp) to compute intermediate latent vectors. Extend your implementation to generate a series of images (for example, 10 equally spaced steps) that show the gradual transition from the first seed to the second. Finally, display the sequence of images in a single figure using matplotlib subplots and save the resulting figure.
   3. (5 points) **Style Mixing:** Implement a function to perform style mixing using a pretrained StyleGAN3 model. Your task is to generate style-mixed images by first computing the latent vectors for a set of source seeds (row seeds) and a set of style seeds (column seeds), then mapping these latent vectors into the intermediate latent space W. For each source image, replace specific layers (as determined by an adjustable parameter) with corresponding layers from a style seed to generate a new mixed image. Experiment by modifying the set of layers used for style mixing and document the visual impact on the generated images. Your submission should include the modified code, visualizations of the output (using matplotlib), and a

detailed explanation in comments regarding how the changes in the latent spaces and layer selection influence the final image appearance.

   4. (7 points) **Dataset Preparation and Model Fine-Tuning:** Implement a complete pipeline for preparing a custom dataset Anime Faces and fine-tuning a pre-trained StyleGAN3 model on this dataset. Your submission should include the modified code, visualizations of the output after finetuning (using matplotlib).

4. (5 points) **BONUS** Implement the complete training loop for your DCGAN including: proper optimizer configuration (Adam with learning rate 2e-4 and betas=(0.5, 0.999)), correct discriminator and generator loss functions, appropriate training steps alternating between discriminator and generator updates, and visualization of generated samples throughout training.