CS 4644/7643: Deep Learning
Fall 2025
Problem Set 4

Instructor: Zsolt Kira
TAs: Wei Zhou, Aryan Sarswat, Woo Chul Shin, Elias Cho,
Haotian Xue, Sri Siddarth Chakaravarthy Prakash, Neelabh Sinha, David He,
Sriharsha Kocherla, Pratham Mehta, Ayush Patel, Cari He, Zachary Breitbart
Discussions: https://piazza.com/class/m5k29i4gzsf4ab/

Due: March 30, 2025, 11:59pm

Instructions

1. We will be using Gradescope to collect your assignments. Please read the following instructions for submitting to Gradescope carefully!

   - For the **HW4: Writeup** component on Gradescope, please upload one single PDF containing the answers to all the theory questions and both jupyter notebooks, `hw4-part1.ipynb` and `hw4-part2.ipynb`. All images and written answers should be visible in the jupyter notebook PDFs that you submit. The solution to each problem or subproblem must be on a separate page. When submitting to Gradescope, please make sure to mark the page(s) corresponding to each problem/sub-problem. Also, please make sure that your submission for the coding part only includes the files collected by the `collect_submission` scripts, else the auto-grader will result in a zero, and we won't accept regrade requests for this scenario given the size of the class Likewise, the pages of the jupyter notebook PDFs must also be marked to their corresponding subproblems.
   - For the coding problems, please use the provided `collect_submission_part1.py` script and `collect_submission_part2.py` upload the resulting files to the **HW4: Part 1 - GANs** and **HW4: Part 2 - Diffusion Models** assignments respectively (on Gradescope). Please make sure you have saved the most recent version of your code before running this script.
   - Note: This is a large class and Gradescope's assignment segmentation features are essential. Failure to follow these instructions may result in parts of your assignment not being graded. We will not entertain regrading requests for failure to follow instructions.

2. LaTeX'd solutions are strongly encouraged, but scanned handwritten copies are acceptable. Hard copies are **not** accepted.

3. We generally encourage you to collaborate with other students.

   You may talk to a friend, discuss the questions and potential directions for solving them. However, you need to write your own solutions and code separately, and *not* as a group activity. Please list the students you collaborated with.

1 Collaborators [0.5 points]

Please list your collaborators and assign this list to the corresponding section of the outline on Gradescope. If you don't have any collaborators, please write 'None' and assign it to the corresponding section of the Gradescope submission regardless.

2 Variational Auto-Encoders [7 points]

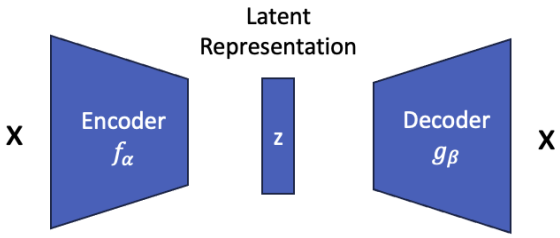

Figure 1: Standard auto-encoder that takes an input x, encodes it into a latent representation z, and then decodes back into the original x. In part 4 you will make a diagram like this, except it will be for a VAE, not a vanilla auto-encoder.

Variational Auto-Encoders (VAE) are latent variable models that can be trained to generate samples from a target distribution. VAEs are similar to standard auto-encoders, except we train them in such a way that we can sample reliably from the latent space. This allows us to take a sample, pass it through the decoder, and generate a new output that resembles the samples we trained on (hence it is a generative model). There are a number of ways to derive and motivate the training procedure for VAEs, this problem walks you through one of them.

1. **Auto-Encoder Reconstruction [1 point]**: Standard auto-encoders are neural networks that are trained to reconstruct an input sample. They are composed of an encoder that encodes the sample in a latent space, and a decoder that reconstructs the original sample from the latent representation. Lets define our encoder as f_α, our decoder as g_β, and our original sample x. What is the loss that you should use when training this network?

2. **Generative Auto-Encoders? [1 point]**: Explain why it isn't practical to generate samples from a vanilla auto-encoder. If we can't generate samples from a vanilla auto-encoder, why are they still useful?

3. **KL Divergence [2 point]**: An important component of variational auto-encoders is KL Divergence. This can be thought of as a way to measure the difference between two distributions. The loss that we use to train a VAE involves a KL divergence term. KL divergence is defined as:

$$D_{KL}(p(x)||q(x)) = E_{x\sim p(x)}[\log \frac{p(x)}{q(x)}]$$

   Conceptually, why does this measure the difference between two distributions? *Hint: how can you rewrite this equation?*

What should our KL divergence term look like if we want to generate samples by drawing from a standard normal distribution? We will use this loss to update the parameters of our model, so your answer should involve some combination of f_α and/or g_β.

4. **Re-parameterization [1 points]**: For training purposes, we want our encoder to be deterministic. But how is this possible if we need our latent representation z to resemble a Gaussian distribution? Luckily, there is an easy change to make. We can re-parameterize the model such that our encoder output is deterministic, and we add in the randomness separately before passing to the decoder. Draw the complete variational auto-encoder with this reparameterization in place (should match the style of Fig 1, but with some additional steps between the encoder and the decoder). *Hint: Your encoder should output two values*

5. **Evidence Lower BOund (ELBO)[1 point]**: Our ultimate goal is to model $p_\theta(x)$, the probability of our training dataset. If we can model $p_\theta(x)$, then we can just draw a sample to generate new data. But this is difficult when our data is complex, say an image. So we assume there is some simpler latent variable, z, that is controlling how x behaves. If we know $p_\theta(z)$ and the relationship between x and z, then we can estimate $p_\theta(x)$. In practice, we train a model to estimate the log likelihood of $p_\theta(x)$ by maximizing the $ELBO$:

$$ELBO = E_{z\sim q_\phi(z|x)}[\log p_\theta(x,z) - \log q_\phi(z|x)]$$

Show that $\log(p_\theta(x)) \geq ELBO$.

6. **Making Connections [1 point]**: The ELBO can be written in a different way that makes a bit more sense in the context of VAEs. First, manipulate your result from the previous section to show that:

$$ELBO = E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)] - E_{z\sim q_\phi(z|x)}[\log \frac{q_\phi(z|x)}{p_\theta(z)}]$$

Second, explain each term of this equation in the context of parts 1 and 3. With that in mind, why would maximizing the ELBO train an effective VAE?

3 Generative Adversarial Networks [3.5 points]

The last two parts of this question are based off of Cycle-GAN.

1. **2 Player Game [1 point]**: Explain the two player game that is played by a Generative Adversarial Network. What is the Nash Equilibrium for this game?

2. **Style Transfer [1 point]**: Generative adversarial networks train a model to generate samples that are indistinguishable from a training dataset. Now lets imagine a scenario where we have two training datasets - one of regular images, X, and one of Monet paintings, Y. If we want to generate regular images that look like Monet paintings and Monet paintings that look like regular images, we need two generators and two discriminators. What is the role of each generator and each discriminator? How many two player games are being played here? What is the Nash Equilibrium for each game?

3. **Cycle-Consistency [1.5 points]**: When training a GAN for style transfer, cycle consistency loss is an important component. For two mapping function $G : X \to Y$ and $F : Y \to X$, we enforce that $F(G(x)) = x$ and $G(F(x)) = x$. Why do we need to include this loss when training a model for style transfer? What might happen if we do not enforce cycle consistency? What are F and G in the context of the previous question?

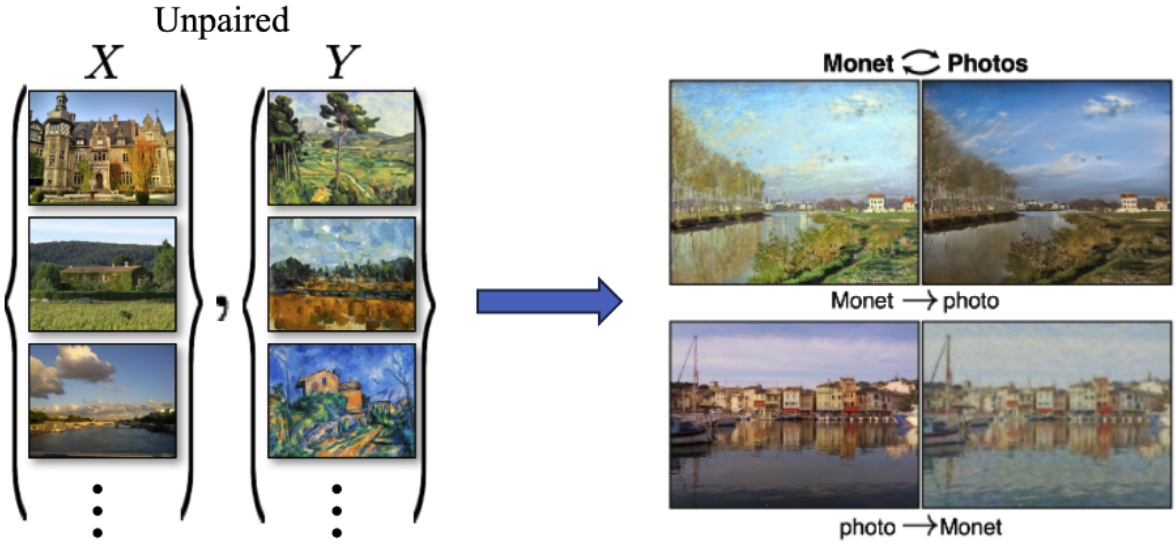

Figure 2: Style transfer between two unpaired sets of images. Images taken from Cycle-GAN

4 Paper Reviews [4 Points] [Required for both 4464 and 7643]

For this homework you will be writing two paper reviews. **The first paper is required for both 4464 and 7643**. Both of these papers are about diffusion models; we recommend reading them before you start the coding section on diffusion models. Especially the first paper, Denoising Diffusion Probabilistic Models, as you will be implementing a number of the equations and algorithms that the authors derive.

1. The following paper **Denoising Diffusion Probabilistic Models**, which was presented at NeurIPS 2020, introduces a method for high quality image generation inspired by non-equilibrium thermodynamics. Diffusion models operate through a forward noising process and a reverse denoising process (maybe explain both of these in your review). These state of the art generative models are useful in many domains including image generation and robotics.

   The paper can be viewed here.

2. **[Extra Credit for 4464]** The following paper **High-Resolution Image Synthesis with Latent Diffusion Models**, which was presented at CVPR 2022, introduces a method which generates high quality images using diffusion in a latent space. This paper combines two of the topics that you worked on in this homework: auto-encoders and diffusion models (you will work on this in the coding section). One of the main drawbacks of diffusion models is that they are quite slow to generate samples. This makes high quality image generation impractical if the model operates directly on the pixel space. Latent diffusion avoids this by operating in a learned latent space. Dalle-2 is a popular image generation tool that uses latent diffusion (although they are operating in a different latent space than what is described in this paper).

   The paper can be viewed here.

Each review will be graded using the following evaluation rubric:

3. **[1 point]** Briefly summarize the key contributions, strengths and weaknesses of this paper.

4. **[1 point]** What is your personal takeaway from this paper? This could be expressed either in terms of relating the approaches adopted in this paper to your traditional understanding of explainability techniques, or potential future directions of research in the area which the authors haven't addressed, or anything else that struck you as being noteworthy.

Guidelines: Please restrict your reviews to at least one paragraph per question, but no more than 350 words (total length for answers to both of the above questions). Please write separate answers for each question to assist in Gradescope grading. In total, you should be writing 700 words for this section. 350 words for the paper 1 review, and 350 words for the paper 2 review.

5 Coding: GANs and Diffusion Models [30 points]

The coding for this assignment is broken up into two parts - will consist of implementing a GAN and a Diffusion Model (mostly) from scratch. Get started early on the coding as this is a long assignment. To get started, download the code zip file from Canvas. Also, be sure to append the completed Jupyter notebooks for both `hw4-part1.ipynb` and `hw4-part2.ipynb` (as mentioned in the instructions) to the HW4 theory solutions and upload them as one PDF on Gradescope under HW4 writeup.

Points breakdown:

- HW4 Theory
  - Theory Answers [15 points]
- HW4 Code
  - Part 1 (GAN) Auto-grader [5 points]
  - Part 2 (Diffusion) Auto-grader [14 Points]
- HW4 Notebook
  - Part 1 (GAN) PDF [4 points]
  - Part 2 (Diffusion) PDF [7 points]

Contributers

- Matthew Bronars (Coding + Theory)
- Manav Agrawal (Coding)
- Mihir Bafna (Coding)