

HOMEWORK 2

16824 VISUAL LEARNING AND RECOGNITION (FALL 2023)

<https://piazza.com/class/llo9w21ejlp2f>

RELEASED: Mon, 2nd Oct 2023

DUE: Wed, 25th Oct 2023

Instructor: Jun-Yan Zhu

TAs: Zhipeng Bao, Wen-Hsuan Chu, Yeojin Jung, Rutika Moharir

START HERE: Instructions

- **Collaboration policy:** All are encouraged to work together BUT you must do your own work (code and write up). If you work with someone, please include their name in your write-up and cite any code that has been discussed. If we find highly identical write-ups or code or lack of proper accreditation of collaborators, we will take action according to strict university policies. See the [Academic Integrity Section](#) detailed in the initial lecture for more information.
- **Late Submission Policy:** There are a **total of 10** late days across all homework submissions. Submissions that use additional late days will incur a 10% penalty per late day.
- **Submitting your work:**
 - We will be using Gradescope (<https://gradescope.com/>) to submit the Problem Sets. Please use the provided template only. You do **not** need any additional packages and using them is **strongly discouraged**. Submissions must be written in LaTeX. All submissions not adhering to the template will not be graded and receive a zero.
 - **Deliverables:** Please submit all the .py files. Add all relevant plots and text answers in the boxes provided in this file. To include plots you can simply modify the already provided latex code. Submit the compiled .pdf report as well.

NOTE: Partial points will be given for implementing parts of the homework even if you don't get the mentioned accuracy as long as you include partial results in this pdf.

1 Generative Adversarial Networks (50 points)

We will be training Generative Adversarial Networks ([GAN](#)) on the [CUB 2011 Dataset](#).

- **Setup:** Run the following command to set up everything you need for the assignment:
`./setup.sh /path/to/python_env/lib/python3.8/site-packages`

- **Question:** Follow the instructions in the `README.md` file in the `gan/` folder to complete the implementation of GANs.

- **Debugging Tips:**

- GAN losses are pretty much meaningless! If you want to understand if your network is learning, visualize the samples. The FID score should generally be going down as well.
- **Do NOT change the hyper-parameters at all**, they have been carefully tuned to ensure the networks will train stably. If things aren't working its a bug in your code.
- For debugging, disable JIT using `export PYTORCH_JIT=0 python ...` and disable AMP by using the flag `--disable_amp`. However, do note that disabling JIT will cause the FID calculation to fail. So only disable JIT to make sure that your network code runs correctly, then re-enable when training. If you observe any errors involving type mismatches and tensors that have half types, it is due to AMP, you may need to explicitly cast the tensor using `.half()`.
- Here is a sample image from WGAN-GP at the end of training. The other networks may have variations but should look similar:



- **Expected results:**

- Vanilla GAN: Final FID should be less than 110.
- LS-GAN: Final FID should be less than 90.
- WGAN-GP: Final FID should be less than 70.

- **Deliverables:** The code will log plots to `gan/data_gan`, `gan/data_ls_gan`, and `gan/data_wgan_gp`. Extract plots and paste them into the appropriate section below. Note for all questions, we ask for final FID. Final FID is computed using 50K samples, at the very end of training. See the final printout for "Final FID (Full 50K)".

1. Paste your plot of the samples and latent space interpolations from Vanilla GAN as well as the *final* FID score you obtained.

Solution:

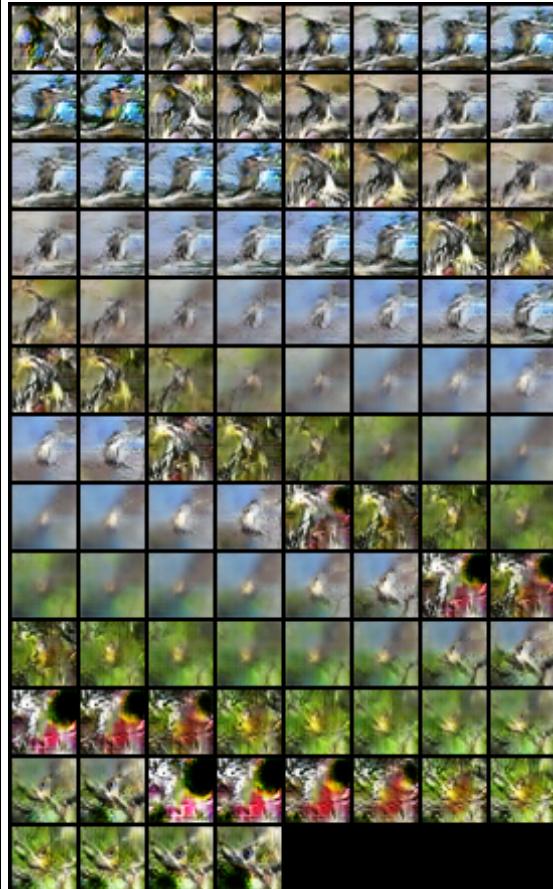
Vanilla GAN FID: **91.4784504834638**



2. Paste your plot of the samples and latent space interpolations from LS-GAN as well as the *final FID score* you obtained.

Solution:LS-GAN FID: **81.03417865616962**

(a) LS-GAN Samples



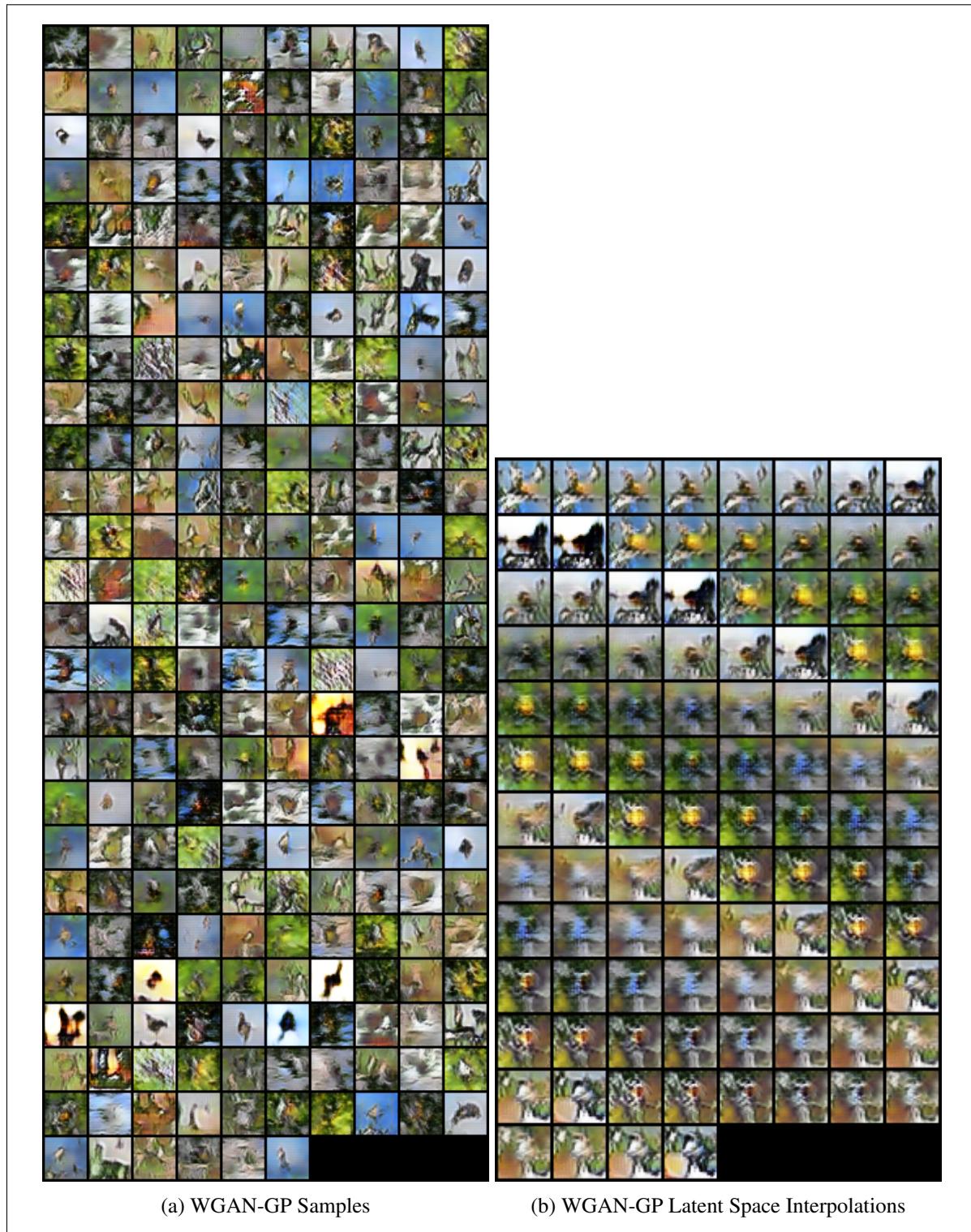
(b) LS-GAN Latent Space Interpolations

3. Paste your plot of the samples and latent space interpolations from WGAN-GP as well as the *final* FID

score you obtained.

Solution:

WGAN-GP FID: **66.5059153279584**

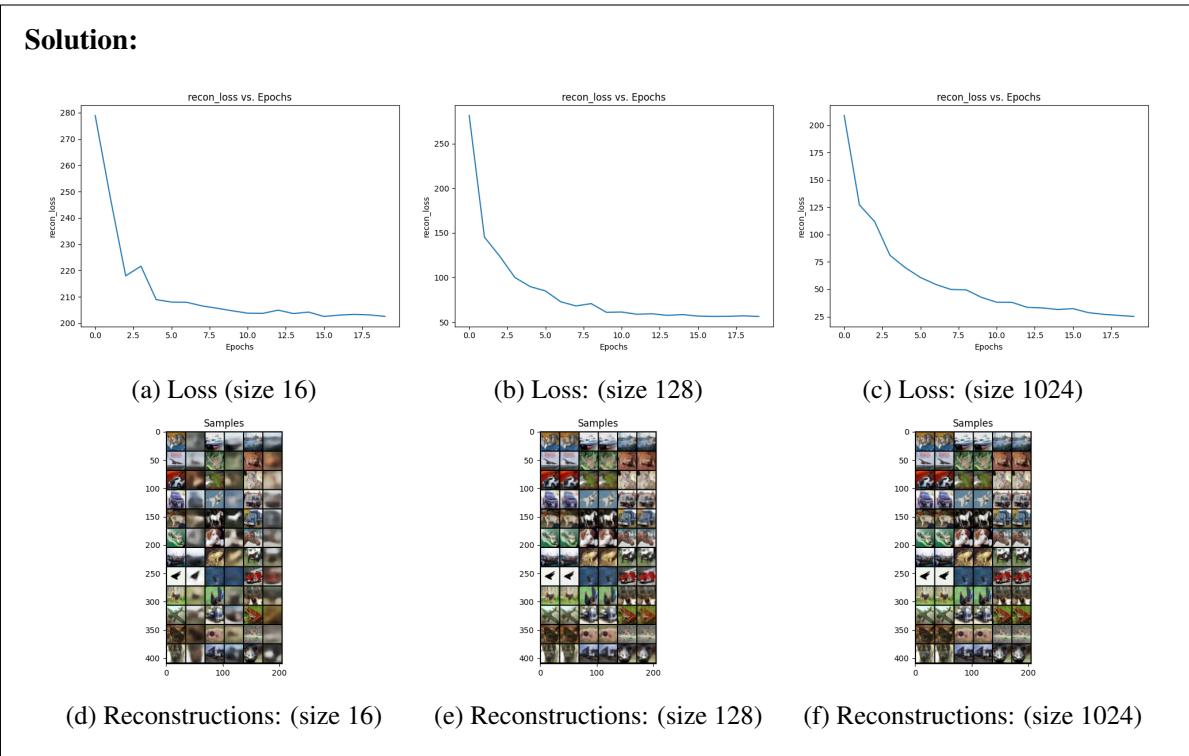


2 Variational Autoencoders (30 pts)

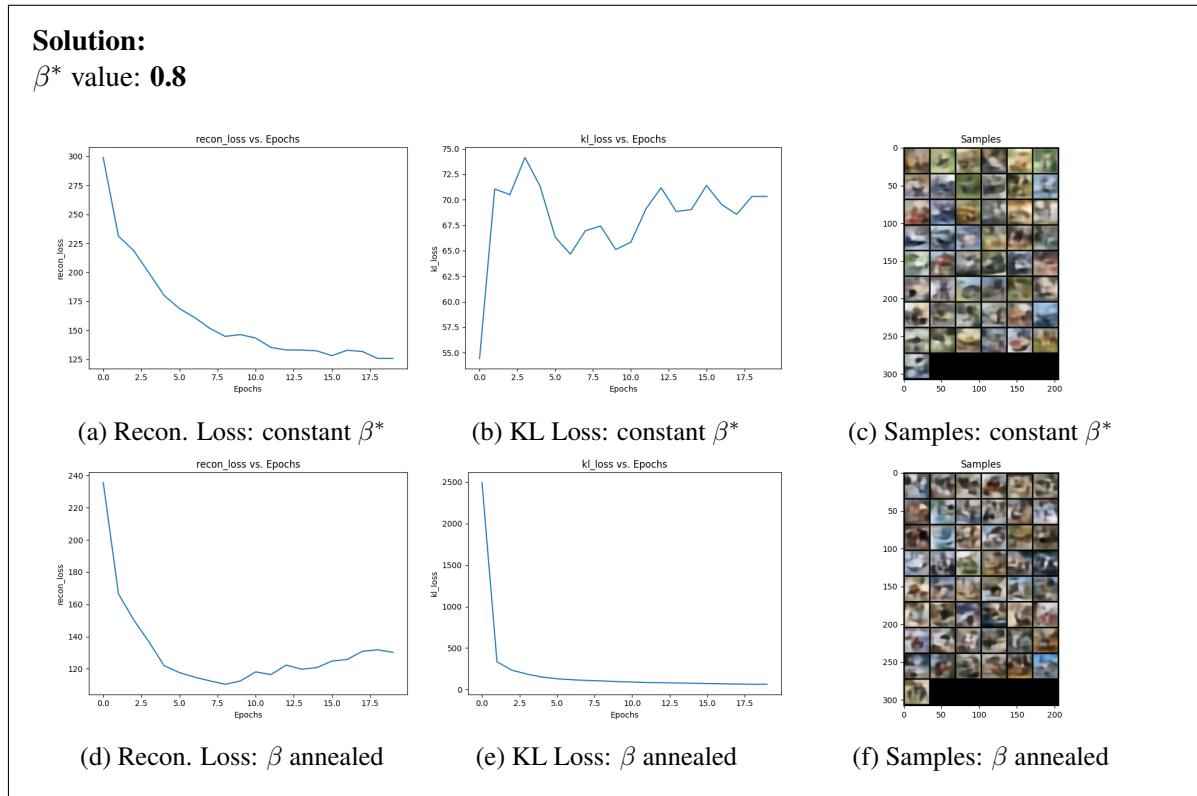
We will be training Autoencoders and Variational Auto-Encoders ([VAE](#)) on the [CIFAR10](#) dataset.

- **Question:** Follow the instructions in the `README.md` file in the `vae/` folder to complete the implementation of VAEs.
- **Debugging Tips:**
 - Make sure the auto-encoder can produce good-quality reconstructions before moving on to the VAE. While the VAE reconstructions might not be clear and the VAE samples even less so, the auto-encoder reconstructions should be very clear.
 - If you are struggling to get the VAE portion working: debug the KL loss independently of the reconstruction loss to ensure the learned distribution matches standard normal.
- **Expected results:**
 - AE: reconstruction loss should be < 40 , reconstructions should look similar to original image.
 - VAE: reconstruction loss should be < 145 ($\beta = 1$ case).
 - VAE: reconstruction loss should be < 125 when annealing β .
- **Deliverables:** The code will log plots to different folders in `vae`. Please paste the plots into the appropriate place for the questions below. Note for ALL questions, use the reconstructions and samples from the final epoch (epoch 19).

1. Autoencoder: For each latent size, paste your plot of the reconstruction loss curve and reconstructions.



2. VAE: Choose the β that results in the best sample quality, β^* . Report the β^* you used in your experiments. In addition, paste the reconstruction, KL loss curve plots, and the sample images corresponding to the VAE trained using constant β^* and the VAE trained using β annealing scheme with β^* .



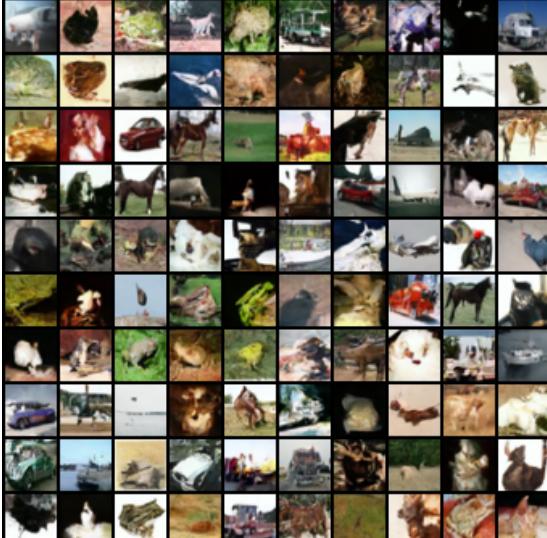
3 Diffusion Models (20 points)

We will be running inference using a pre-trained diffusion model (**DDPM**) on CIFAR-10.

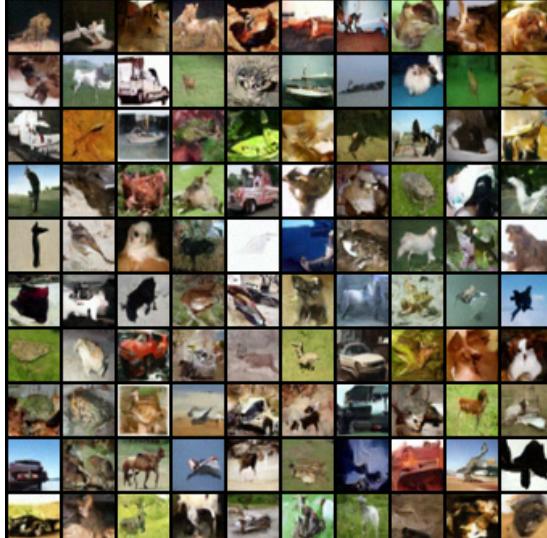
- **Setup:** Download our pre-trained checkpoint for DDPM from [here](#).
- **Question:** Follow the instructions in the `README.md` file in the `diffusion/` folder to complete the implementation of the sampling procedures for Diffusion Models.
- **Expected results:**
 - FID of less than 60 for DDPM and DDIM
 - **Deliverables:** The code will log plots to `diffusion/data_ddpm` and `diffusion/data_ddim`. Extract plots and paste them into the appropriate section below.

1. Paste your plots of the DDPM and DDIM samples.

Solution:



(a) DDPM Samples



(b) DDIM Samples

2. Paste in the FID score you obtained from running inference using DDPM and DDIM.

Solution:

- DDPM FID: **48.003751162844196** at sampling steps **100**
- DDIM FID: **47.9544994967365**

3. Compare the generated samples from your VAE and diffusion models. What do you observe? Does one generate better pictures than the other? Does the quality of the generated images match with the FID differences?

Solution:

When comparing the VAE and the Diffusion models, we see that the diffusion models produce a better/sharper result in samples. The distinction in image quality between VAEs and diffusion models can be attributed to their underlying architectures and training procedures. The nature of how these models handle image generation and the approach to learning the underlying distributions significantly impacts the quality of generated images. Diffusion models tend to better preserve details due to their autoregressive nature and training strategy, which allows them to capture finer details and complex structures in the images. However, this often comes at the cost of increased computational complexity and longer training times compared to VAEs.

I think the quality of the generated images for DDPM and DDIM look very similar. The lower FID score represents that the quality of images generated by the generator is higher and similar to the real ones. **Due to budget constraints, I replaced the sampling steps by 100 to speed up the DDPM inference as mentioned on Piazza post 139 f14** and got at FID score slightly higher than the DDIM but based on the samples, I see minute differences by the naked eye which indicate the FID scores are very close to each other. So, I would say the quality of the generated images match with the FID difference since the FID difference is very minuscule. I would say that if I ran the inference on DDPM with the default values, it would reach very close to the DDIM, possibly even surpassing it ever so slightly.

Collaboration Survey Please answer the following:

1. Did you receive any help whatsoever from anyone in solving this assignment?

Yes

No

- If you answered ‘Yes’, give full details:
- (e.g. “Jane Doe explained to me what is asked in Question 3.4”)

I received help from Steven Shi to debug my code of diffusion models for DDIM. (tau previous was not clipped earlier)

I received help from Arpit Sahni to help me get my FID score for GANs below the target thresholds, I made a mistake in the pixel unshuffle because my permute was incorrect.

2. Did you give any help whatsoever to anyone in solving this assignment?

Yes

No

- If you answered ‘Yes’, give full details:
- (e.g. “I pointed Joe Smith to section 2.3 since he didn’t know how to proceed with Question 2”)

I helped Mukul Ganwal in solving the problem of random pixels printing in VAEs, get his FID for GANs below threshold.

I helped Ritwhik Jayanth in solving shape errors in GANs - networks.

I helped Sumesh KS in understanding the diffusion calculations for calculating cumulative product at previous timestep.

3. Note that copying code or writeup even from a collaborator or anywhere on the internet violates the [Academic Integrity Code of Conduct](#).