Diffusion Models and Generative AI – Assignment 4

00960236

TA in Charge: Avishag Nevo
(avishag.nevo@campus.technion.ac.il)

Abstract

In this assignment, we will extend our previous assignments and implement two important techniques in diffusion models: Denoising Diffusion Implicit Models (DDIM) and Classifier-Free Diffusion Guidance (CFG). We highly recommend you to check out the details in the papers and understand the mathematics before you start the assignment. The list of recommended resources is as follows:

1. Paper Denoising Diffusion Implicit Models (DDIM)
2. Paper Classifier-Free Diffusion Guidance (CFG)
3. The relevant tutorial slides

Set up the project environment

- activate the environment *conda activate azureml_py38*
- run *pip install dotmap==1.3.30*
- run *pip install pytorch_lightning==2.5.1.post0*

Throught this HW you are only allowed to edit the parts marked by TODO.

1 Task 1: Denoising Diffusion Implicit Models (DDIM)

Recall that the algorithm for DDPM sampling is nothing but a sequence of noise predictions and denoising steps.

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Figure 1: DDPM

To speed up sampling, we keep the pre-trained DDPM and only change the relevant parts within the loop.

$$\boldsymbol{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\epsilon_\theta^{(t)}(\boldsymbol{x}_t)}{\sqrt{\alpha_t}} \right)}_{\text{“ predicted } \boldsymbol{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \epsilon_\theta^{(t)}(\boldsymbol{x}_t)}_{\text{“direction pointing to } \boldsymbol{x}_t\text{”}} + \underbrace{\sigma_t \epsilon_t}_{\text{random noise}}$$

Figure 2: DDIM

TODO In this assignment, we will primarily reuse the relevant code from the previous assignments. Before proceeding, please ensure that your previous results are successfully reproduced.

1.1 Implement the DDIM Sampling Algorithm

Complete the definitions of the functions ‘ddim_p_sample()‘ and ‘ddim_p_sample_loop()‘ in ‘2d_plot_diffusion_todo/ddpm.py‘.

1.2 Testing with 2D Examples

Execute every cell in the notebook ‘2d_plot_diffusion_todo/ddpm_tutorial.ipynb‘. - the code will attempt DDIM sampling during training; - the Chamfer Distance between the particles generated via the DDIM sampling and the ground truth will be reported in the notebook. For sanity checks, you may replace the invocation of ‘ddim_p_sample_loop‘ with ‘p_sample_loop‘.

Take screenshots of:

1. the Chamfer Distance reported after executing the cell ‘DDIM Sampling‘
2. the visualization of the sampled particles after executing the cell ‘DDIM Sampling‘

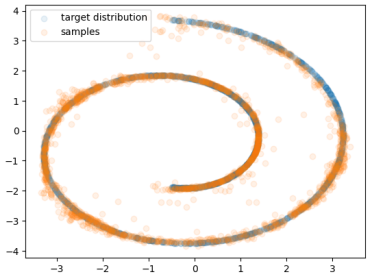

Figure 3: Example for (2)

2 Task 2: Classifier-Free Guidance (CFG)

In this task, we will explore the most representative technique for generating class conditional data —the Classifier-Free Guidance (CFG).

Given a dataset consisting of pairs $(\mathbf{x}, \mathbf{c})$ of data $\mathbf{x}$ and its corresponding class label $\mathbf{c}$, CFG can be used to improve the quality of class-conditional samples by randomly omitting class labels during the training process with some probability p_{uncond}, to train both the data classes $c \in C$ and with the empty class $\varnothing$.

After training the model, we can generate class-conditional samples by using a slightly modified version of the reverse diffusion process. The variable w, known as the guidance strength, enables us to balance the trade-off between sample quality and diversity, just like we saw in the tutorials.

In this assignment, we will use the AFHQ dataset, consisting of animal images of three categories (cat, dog, wildlife), along with their class labels. We will train a DDPM model on the AFHQ dataset and generate class-conditional samples using CFG.

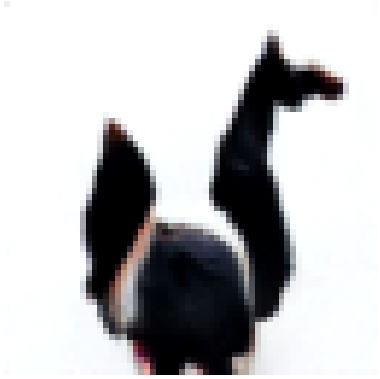

Figure 4: Example image after 20,000 epochs. Not good enough for final submission, use this as a sanity check after 20,000 epochs.

Note : After ~20,000 epochs you should already see some (awful) results, not just pure noise 4

2.1 Implement Class Conditioning Mechanism

Implement a simple class conditioning mechanism in the method ‘forward‘ of the class ‘UNet‘ defined in the file ‘image_diffusion_todo/network.py‘. The optional argument ‘class_label‘ is assumed to be a PyTorch tensor holding the integer class labels. Apply the one-hot encoding to the class labels. You may use the predefined linear layer ‘self.class_embedding‘. For now, the main backbone of the network is conditioned only on the diffusion timestep ‘temb‘. Before proceeding to the next steps, we highly recommend you to check whether class-conditioned training runs properly.

Use the script *‘image_diffusion_todo/train.py‘ with the flag ‘–use_cfg‘* to train the model with CFG enabled.

2.2 Implement CFG Training

Complete the ‘TODO‘ block in the method ‘forward‘ of the class ‘UNet‘ defined in the file ‘image_diffusion_todo/network.py‘. It should be suffice to write a few lines of code.

2.3 Implement CFG Sampling

Complete the ‘if do_classifier_free_guidance‘ blocks in the method ‘sample‘ in the file ‘image_diffusion_todo/model.py‘. In our implementation, the CFG is enabled by setting ‘cfg_scale‘ greater than 0.0.

You can test your implementation by running the script ‘image_diffusion_todo/sampling.py‘. Specifically, run *python image_diffusion_todo/sampling.py - -ckpt_path @CKPT_PATH - -save_dir @DIR_TO_SAVE_IMGS - -use_cfg*

by providing the path to your model’s checkpoint trained *with* CFG training enabled.

You would eventually need to sample 500 images (for each scale)

2.4 Test and Evaluate

In this part you will evaluate and report the FID measured on the samples generated using CFG with the scale of 0.0, and 7.5 (default) on your best model, with **500 generated images** for each scale. Do the following:

Do NOT forget to execute 'dataset.py' before measuring FID score. Otherwise, the output will be incorrect due to the discrepancy between the image resolutions.
python dataset.py # to constuct eval directory.

After processing the data, use the script 'image_diffusion_todo/fid/measure_fid.py' to measure the FID score **in the notebook** *fid_todo.ipynb*. The pre-trained inception model for FID is provided in the file 'image_diffusion_todo/fid/afhq_inception.ckpt'.

In the notebook *fid_todo.ipynb*, execute something like
python image_diffusion_todo/fid/measure_fid.py GT_IMG_DIR GEN_IMG_DIR

Use the directory containing the processed data (e.g., 'data/afhq/eval') as @GT_IMG_DIR. The script will automatically search and load the images. The path DIR_TO_SAVE_IMGS should be the same as the one you provided when running the script 'sampling.py', for each scale you should use 500 samples.

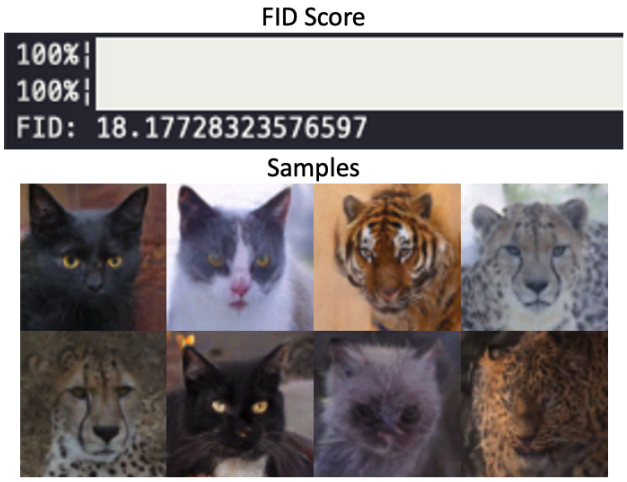

Figure 5: Task 2 example

3 What to Submit

3.1 HW4_{id}.pdf or HW4_{id1}_{id2}.pdf

- Task 1
  - Screenshot of Chamfer distance result of DDIM sampling
  - Visualization of DDIM sampling
- Task 2
  - Screenshot of FID computed using the samples generated using the CFG scale of 0.0 and 7.5
  - images generated using the CFG scale of 0.0 (8 total) and 7.5 (8 total, 2 for each category)

In a single PDF file. Write your name and student/s ID, and include submission items listed above. Refer to more detailed instructions written in each task section about what to submit.

3.2 HW4_{id}.zip or HW4_{id1}_{id2}.zip

When creating your zip file, exclude data (e.g., files in AFHQ dataset) and any model checkpoints, including the provided pre-trained classifier checkpoint when compressing the files. Include in the ZIP file the following (unzipped) directories:

3.2.1 HW4_{id}_code or HW4_{id1}_{id2}_code directory

The files structure should be exactly like the original one, with all the notebooks **executed** containing your real results.

3.2.2 HW4_{id}_images or HW4_{id1}_{id2}_images directory

64 images generated using the CFG scale of 0.0 and 7.5 (32 each).

4 Grading

You will receive a zero score if:

- you do not submit,
- your code is not executable in the Python environment we provided, or
- you don't report your real results in the .ipynb notebooks, or
- you modify anycode outside of the section marked with 'TODO' or use different hyperparameters that are supposed to be fixed as given.

Plagiarism in any form will also result in a zero score and will be reported to the university.

Otherwise, you will receive up to 100 points from this assignment

Other then implementation correctness, points is given by the following:
*Task 1

- higher points: Achieve CD lower than **60** from DDIM sampling.
- lower points: Achieve CD greater, or equal to **60** and less than **80** from DDIM sampling.
- no point: otherwise.

*Task 2

- higher points: Achieve FID less than **30** in both CFG scales: 0.0 and 7.5.
- lower points: Achieve FID between **30** and **50** in one of the two CFG scales: 0.0 and 7.5.
- no point: otherwise.