CS492D: Diffusion Models and Their Applications

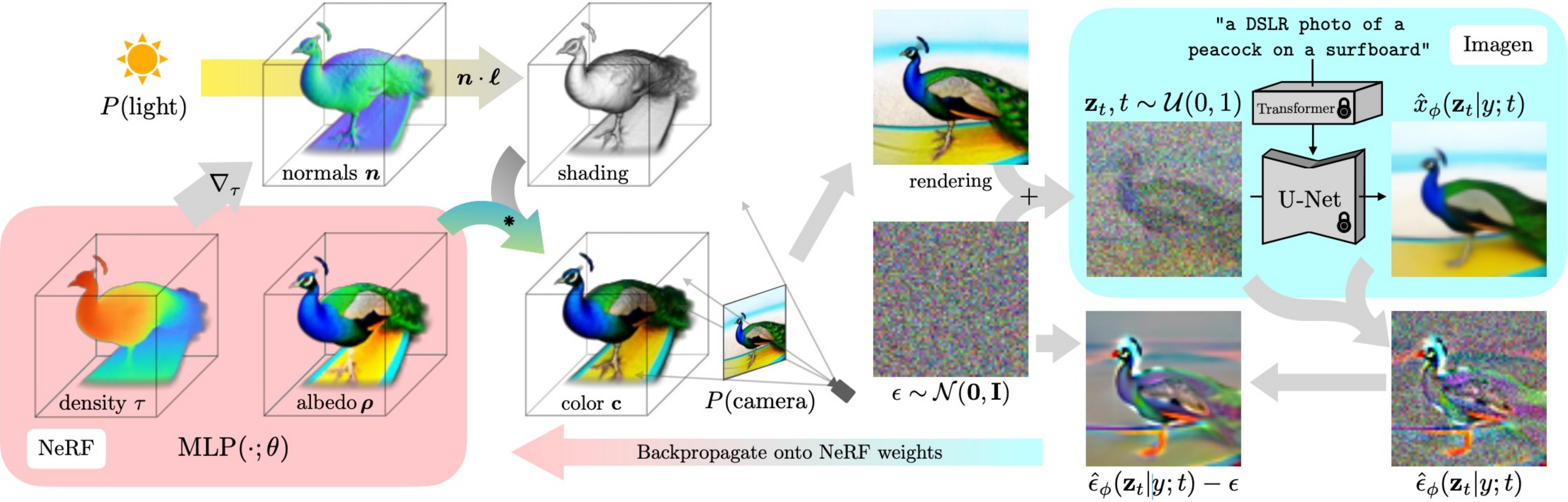
Assignment 4 Session

Jaihoon Kim

Fall 2024
KAIST

Introduction

In Assignment 4, you will implement SDS and its variant, PDS, and optionally, VSD.

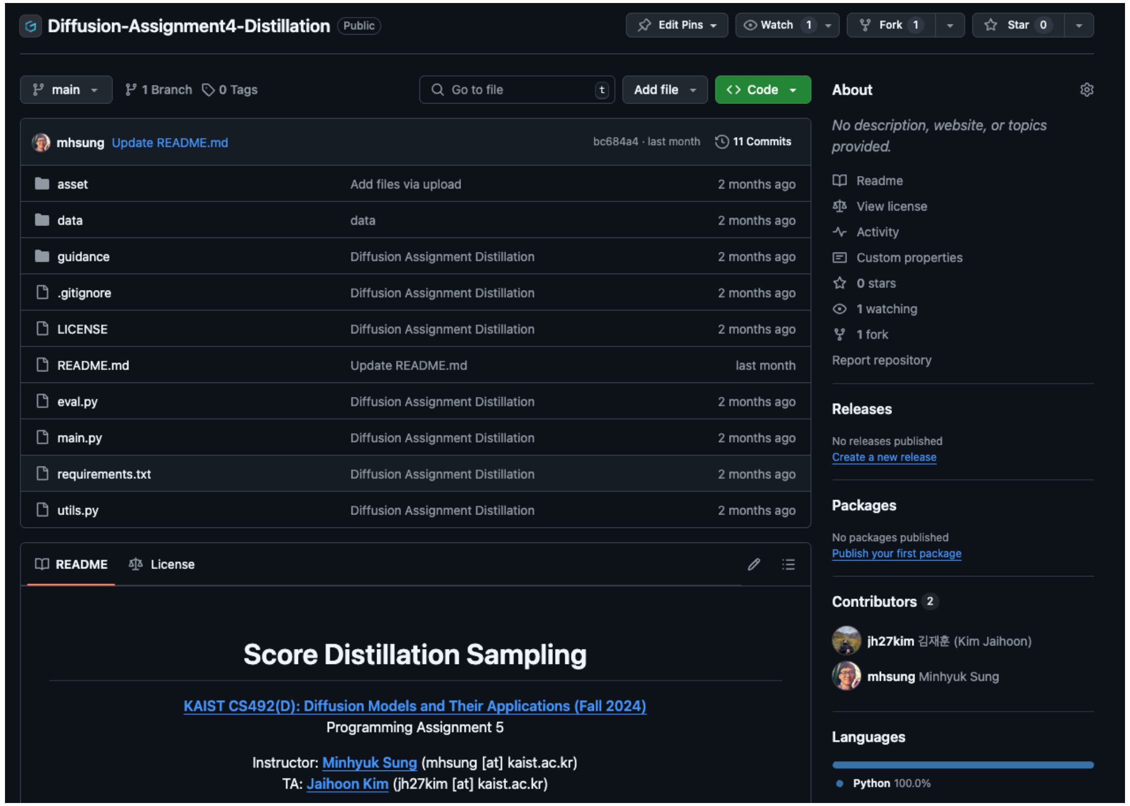


DreamFusion, Poole *et al.*, ICLR 2023

Introduction

The skeleton code and instructions are available at:

https://github.com/KAIST-Visual-AI-Group/Diffusion-Assignment4-Distillation

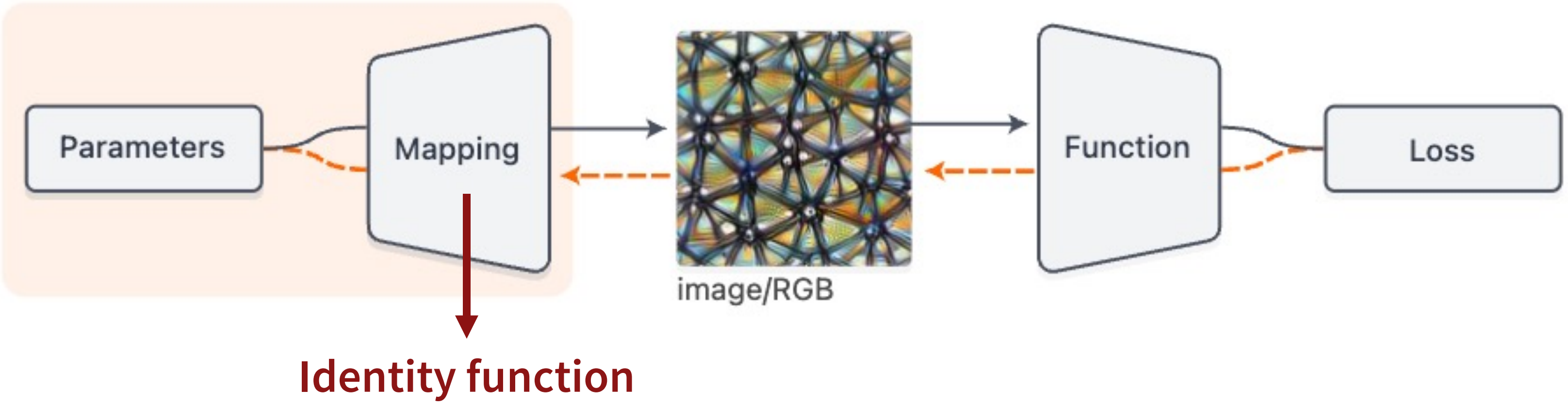


Important Notes

- All programming assignments are due **two weeks** after the assignment session.
- Late submission will incur **20% penalty** for **each** late day!
- Please carefully check the README of each assignment.
- Missing items in your submission will also incur penalties.

What to Do: Overview

We will focus on implementing the core components of SDS and PDS for image generation and editing, where $g(\boldsymbol{\theta}; c)$ is an identity function.



DIP, Mordvintsev *et al.*, Distill 2018

What to Do: Overview

Input

Output

Prompt: y
"A castle next to a river"

Target image $\boldsymbol{x_0}$
📌 Prompt **alignment**

Prompt: y
"A church beside a lake"

What to Do: Overview

Note that one can extend to generate other types of visual content by switching $g(\boldsymbol{\theta}; c)$ accordingly.

3D - $g(\boldsymbol{\theta}; c)$ = NeRF Render
DreamFusion (Poole et al.)

Vector Image - $g(\boldsymbol{\theta}; c)$ = SVG Render
VectorFusion (Jain et al.)

Mesh Texture - $g(\boldsymbol{\theta}; c)$ = Mesh Render
Paint-it (Kim et al.)

What to Do: Task 1

Recall from last previous lecture, the SDS algorithm is presented as follows:

Until convergence, repeat:

1. $t \sim \mathcal{U}(1, T)$. $c \sim \mathcal{U}(\mathcal{C})$.
2. $\boldsymbol{x}_{0|t} = g(\boldsymbol{\theta}; c)$.
3. $\boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
4. $\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{0|t} + \sqrt{1 - \bar{\alpha}_{t-1}}\boldsymbol{z}_t$.
5. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\partial}{\partial \boldsymbol{\theta}} \|\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}, y, t) - \boldsymbol{z}_t\|^2$.

📌 For ease of implementation, you may consider t as $t + 1$.

What to Do: Task 1

Since $g(\boldsymbol{\theta}; c)$ is an identity function, we can ignore some parts.

Until convergence, repeat:

1. $t \sim \mathcal{U}(1, T)$. $\cancel{c \sim \mathcal{U}(\mathcal{C})}$.
2. $\boldsymbol{x}_{0|t} = \cancel{g(\boldsymbol{\theta}; c)}\ \boldsymbol{\theta}$.
3. $\boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
4. $\boldsymbol{x}_{t-1} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{0|t} + \sqrt{1 - \bar{\alpha}_{t-1}}\boldsymbol{z}_t$.
5. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\partial}{\partial \boldsymbol{\theta}} \|\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}, y, t) - \boldsymbol{z}_t\|^2$.

📌 If you face GPU OOD issue, wrap $\hat{\boldsymbol{\varepsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}, y, t)$ with `torch.no_grad()`.

What to Do: Task 1

Initialization of pre-trained diffusion model, text embeddings and $\boldsymbol{\theta}$ is provided in `main.py`.

Implement TODOs in `guidance/sd.py`:

- `get_sds_loss()`

```python
def get_sds_loss(
    self,
    latents,
    text_embeddings,
    guidance_scale=100,
    grad_scale=1,
):

    # TODO: Implement the loss function for SDS
    raise NotImplementedError("SDS is not implemented yet.")
```

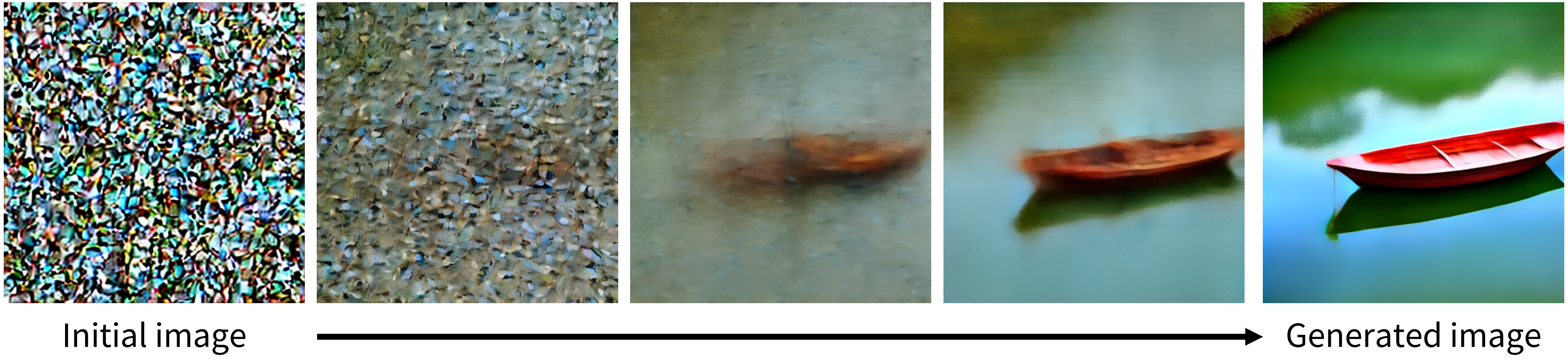
Hint: Use reparameterization trick.

What to Do: Task 1

Run the following code to sample images using SDS:

```
python main.py -prompt `{PROMPT}` --loss_type sds --guidance_scale 25 --step
500
```

An example of the evolution of the parameterized image over time.



What to Do: Task 1

For evaluation, we will measure the text alignment using CLIP score.

Use the prompts provided in `data/prompt_img_pairs.json` and gather the generated images in a directory.

!! Note that the filenames must match their corresponding text prompts.

```
./
├── a_boat_in_a_river.png
├── A_burger_on_the_table.png
├── A_cabin_surrounded_by_forests.png
├── A_car_on_the_road.png
├── A_castle_next_to_a_river.png
├── a_cat_sitting_on_a_table.png
├── A_church_beside_a_lake.png
├── A_dog_sitting_on_grass.png
├── A_red_bus_driving_on_a_desert_road.png
└── A_villa_close_to_the_pool.png

0 directories, 10 files
```

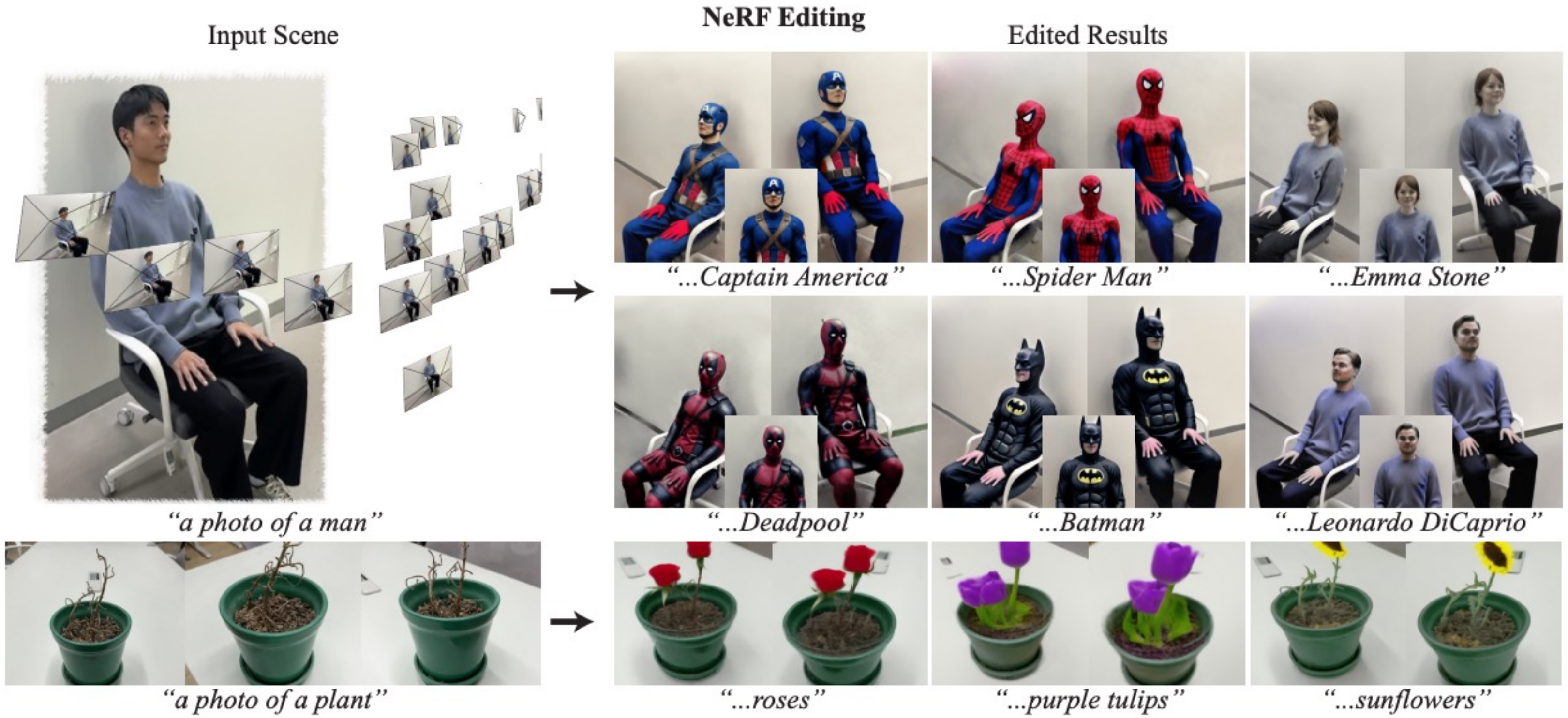
What to Do: Task 1

Run the evaluation script, which outputs an `eval.json` file.

```
Python eval.py --fdir1 {$FDIR1}
```

What to Do: Task 2

SDS can be used not only for generation but also for editing.



PDS, Koo *et al.*, CVPR 2024

What to Do: Task 2

Input

Source image $\boldsymbol{x}_0^{src}$

Source prompt y^{src} : "*deer* doll"
Target prompt y^{tgt} : "*unicorn* doll"

Output

Target edited image $\boldsymbol{x}_0^{tgt}$

📌 Source image **identity**
📌 Target prompt **alignment**

PDS, Koo *et al.*, CVPR 2024

Recall: DDIM

We have looked into cases when $\sigma_t = 0$ and $\sigma_t = \sqrt{1 - \bar{\alpha}_{t-1}}$.

Any other cases?

DDPM sets $\sigma_t = \sqrt{(1 - \bar{\alpha}_{t-1})/(1 - \bar{\alpha}_t)} \sqrt{1 - \bar{\alpha}_t/\bar{\alpha}_{t-1}}$ placing itself between fully deterministic ($\sigma_t = 0$) and maximum stochastic process ($\sigma_t = \sqrt{1 - \bar{\alpha}_{t-1}}$).

Recall: DDIM

In contrast to previous cases where $\sigma_t = 0$ or $\sigma_t = \sqrt{1 - \bar{\alpha}_{t-1}}$,

both $\sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}$ and σ_t^2 are *non-zero*.

$$q_\sigma(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_0 + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2} \cdot \frac{\boldsymbol{x}_t - \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0}{\sqrt{1 - \bar{\alpha}_t}}, \sigma_t^2 \mathbf{I}\right)$$

Recall: DDIM

To sample $\boldsymbol{x}_{t-1}$ we need both $\hat{\boldsymbol{\varepsilon}}_\theta(\boldsymbol{x}_t, y, t)$ and $\boldsymbol{z}_t$ as shown below:

$$\tilde{\mu} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{0|t} + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\,\hat{\boldsymbol{\varepsilon}}_\theta(\boldsymbol{x}_t, y, t).$$

$$\boldsymbol{x}_{t-1} = \tilde{\mu} + \sigma_t \boldsymbol{z}_t, \text{ where } \boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I}).$$

One can compute the stochastic latent $\boldsymbol{z}_t$ as follows:

$$\boldsymbol{z}_t = \frac{\boldsymbol{x}_{t-1} - \tilde{\mu}}{\sigma_t}.$$

What to Do: Task 2

PDS employs the stochastic latent $\boldsymbol{z}_t$ in the optimization.

Until convergence, repeat:

1. $t \sim \mathcal{U}(T_{min}, T_{max})$.
2. $\boldsymbol{x}_{0|t}^{tgt} = \boldsymbol{\theta}$.
3. $\boldsymbol{z}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; $\boldsymbol{z}_{t-1} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
4. $\{\boldsymbol{x}_t^{src}, \boldsymbol{x}_t^{tgt}\} = \sqrt{\bar{\alpha}_t}\boldsymbol{x}_{0|t}^{\{src, tgt\}} + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{z}_t$.
5. $\{\boldsymbol{x}_{t-1}^{src}, \boldsymbol{x}_{t-1}^{tgt}\} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{0|t}^{\{src, tgt\}} + \sqrt{1 - \bar{\alpha}_{t-1}}\boldsymbol{z}_{t-1}$.
6. $\tilde{\mu}^{\{src, tgt\}} = \sqrt{\bar{\alpha}_{t-1}}\boldsymbol{x}_{0|t}^{\{src, tgt\}} + \sqrt{1 - \bar{\alpha}_{t-1} - \sigma_t^2}\hat{\boldsymbol{\varepsilon}}_\theta(\boldsymbol{x}_t^{\{src, tgt\}}, y^{\{src, tgt\}}, t)$.
7. $\boldsymbol{z}_t^{\{src, tgt\}} = \frac{\boldsymbol{x}_{t-1}^{\{src, tgt\}} - \tilde{\mu}^{\{src, tgt\}}}{\sigma_t}$.
8. $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \frac{\partial}{\partial \boldsymbol{\theta}} \left\| \boldsymbol{z}_t^{tgt} - \boldsymbol{z}_t^{src} \right\|^2$.

What to Do: Task 2

Initialization of the source image latent, along with the source and target text embeddings, is provided.

Implement TODOs in `guidance/sd.py`:

- `get_pds_loss()`

```
def get_pds_loss(
    self, src_latents, tgt_latents,
    src_text_embedding, tgt_text_embedding,
    guidance_scale=7.5,
    grad_scale=1,
):

    # TODO: Implement the loss function for PDS
    raise NotImplementedError("PDS is not implemented yet.")
```

What to Do: Task 2

Run the following code to edit images using PDS:

```
python main.py --prompt "{$PROMPT}" --edit_prompt "{$EDIT_PROMPT}" --
src_img_path {SRC_IMG_PATH} --loss_type pds --guidance_scale 7.5 –step 200
```

An example of the evolution of the parameterized target image over time.

Source image
“A burger”

Target image
“A burger
with *lettuce*”

What to Do: Task 2

As in task 1, we will measure the text alignment using CLIP score for evaluation.

Use the source images and prompts provided in `data/prompt_img_pairs.json`.
Then place the generated images in the same directory.

!! Note that the filenames must match their corresponding text prompts.

```
./
├── a_boat_in_a_frozen_river.png
├── A_cabin_surrounded_by_snowy_forests.png
├── A_cat_sitting_on_grass.png
├── A_church_beside_a_waterfall.png
├── A_futuristic_car_wiht_neon_signs_on_the_road.png
├── A_hotdog_on_the_table.png
├── An_ancient_villa_close_to_the_pool.png
├── A_red_sportscar_driving_on_a_desert_road.png
├── a_squirrel_sitting_on_a_table.png
└── A_toy_lego_castle_close_to_the_pool.png

0 directories, 10 files
```

What to Do: Task 2

Run the evaluation script, which outputs an `eval.json` file.

```
Python eval.py --fdir1 {$FDIR1}
```

What to Submit

Include the following items into a PDF file: `{NAME}_{SID}.pdf`.

Task 1

- Generated images with their corresponding prompts.
- A screenshot of `eval.json` showing the CLIP scores for each item and the overall averaged score.

Task 2

- Edited images with their corresponding prompts and source images.
- A screenshot of `eval.json` showing the CLIP scores for each item and the overall averaged score.

What to Submit

Create a single ZIP file `{NAME}_{SID}.zip` including:

- The PDF file, formatted according to the guideline;
- Your implemented code.

Your score will be deducted by 10% for *each* missing item.

Please check carefully!

Grading

You will receive up to 20 points from this assignment.

For Task 1 and Task 2, respectively, you will receive:

- 10 points if CLIP score greater than 0.28,
- 5 points if CLIP score greater, or equal to 0.26 and less than 0.28,
- 0 point if CLIP score less than 0.22.

Grading

Additionally, you can optionally receive a maximum of 5 points for Task 3.

Task 3

- 5 points: Achieve CLIP score greater than 0.28.
- 2.5 points: Achieve CLIP score greater, or equal to 0.26 and less than 0.28.
- 0 point: CLIP score less than 0.22.

Demo

Thank You