Updates

1. 第三題時間限制 10 mins -> 15 mins
2. Slide P38 typo: hw2_download.sh -> hw2_download_ckpt.sh
3. Github README typo:
   stable-diffusion/models/ldm/stable-diffusion-v1/model.ckpt ->
   stable-diffusion/ldm/models/stable-diffusion-v1/model.ckpt
4. Slide P37 新增一個argument輸入model weight
5. For HW2_3, our A4500 20GB allows setting n_sample to 5. However, be aware that setting n_sample to 1 or 5 will result in different generated images even with the same random seed, due to the random sampling process. (Just a reminder: generating an image with `n_sample * n_iter` set to `1 * 25` or `5 * 5` will yield different results)

Homework #2

Deep Learning for Computer Vision
NTU, Fall 2024

Problems – Overview

- **Diffusion Models**
  - Problem 1: Conditional Diffusion Models (30%) [digit dataset - MNIST-M & SVHN]
  - Problem 2: DDIM (35%) [face dataset]
  - Problem 3: Personalization (35%) [personalization dataset]

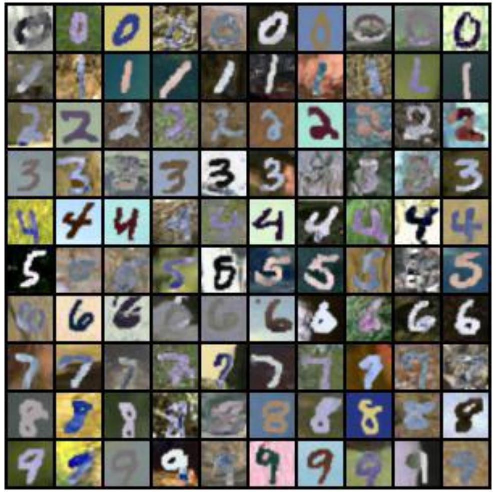
Please refer to “Dataset” section for more details about all datasets.

Outline

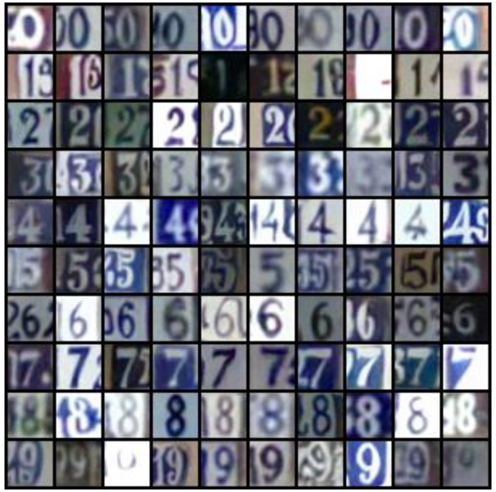
- Problems & Grading
- Dataset
- Submission & Rules

Problem 1: Diffusion Models (30%)

In this problem, you need to implement a **conditional** diffusion model from scratch and train it on **MNIST-M & SVHN datasets** (inside the digit dataset folder). Given the conditional labels 0-9 & dataset labels (e.g. “0” -> Mnist-M, “1” -> SVHN), your model needs to generate the corresponding digit images from a specified dataset as the following example.



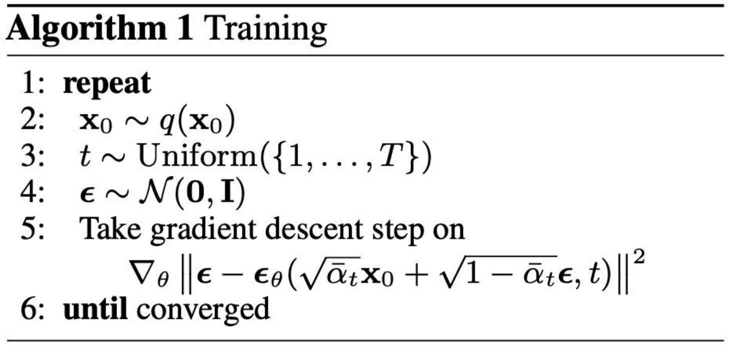
MNIST-M



SVHN

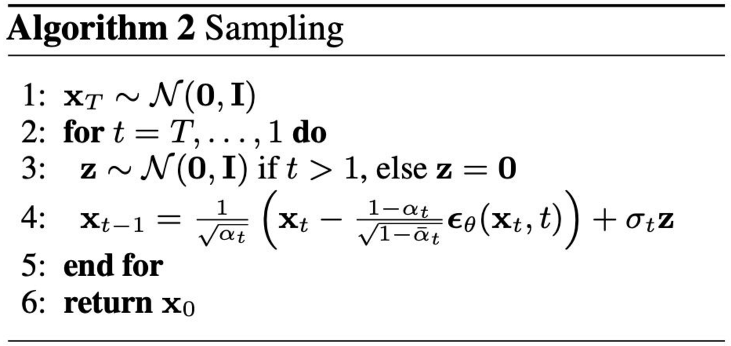
Problem 1: Diffusion Models (30%)

- For simplicity, you are encouraged to implement the training/sampling algorithm introduced in the pioneering paper **DDPM** (Denoising Diffusion Probabilistic Models) as follows:



Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4:   $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5:   Take gradient descent step on
         $\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$
6: until converged
```



Algorithm 2 Sampling

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

- While we do not specify particular ways to implement two-condition diffusion model, we do encourage you to refer to Classifier-free guidance for simplicity.

Reference : Jonathan Ho, Ajay Jain, Pieter Abbeel, “Denoising Diffusion Probabilistic Models” [link]

Problem 1: Diffusion Models (30%)

- Sample random noise from normal distribution to generate **50** conditional images **for each digit (0-9) on MNIST-M & SVHN datasets**. Your script should save total **1000** outputs in the assigned folder for further evaluation.
  - You should name your output digit images as the following format:
    (The first character of each filename indicates the corresponding digit label)

```
Output_folder/
  mnistm/
      0_001.png
      ...
      0_050.png
      1_001.png
      ...
      9_050.png
  svhn/
      0_001.png
      ...
      0_050.png
      1_001.png
      ...
      9_050.png
```

⚠️ You should **fix the random seed** in your program such that the generated images are always the same.

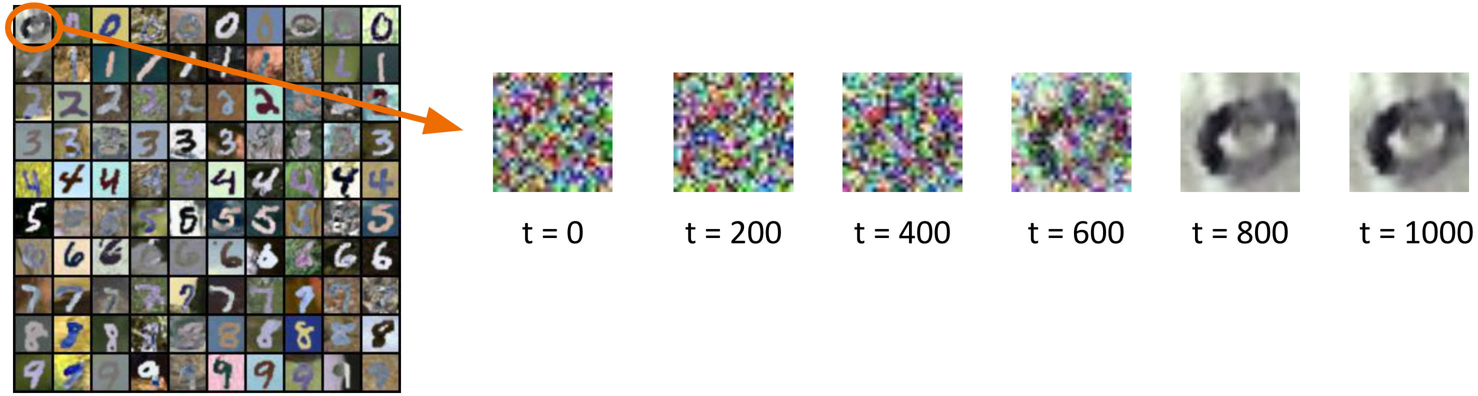
Problem 1: Evaluation (15%)

- For each dataset, we will use **a digit classifier** to evaluate your generated images by **classification accuracy, and average of two accuracies will be your final score**.
  - For evaluating MNIST-M dataset, the evaluation model weight **(Classifier.pth)** is provided in the GitHub template. For SVHN dataset, we leverage timm's model, which will be downloaded by the source code. [notice timm model need to install detectors library]
  - The source code **(digit_classifer.py)** is also provided in the template, which will calculate your accuracy of each dataset and the average accuracy.
    - **Usage:** python3 digit_classifier.py --folder <path_to_output_folder> --checkpoint <path_to _Classifier.pth>
    - Please follow the saving format in the previous page so that the command can run successfully
- (15%) Baseline:

Metric	Simple Baseline (10%)	Strong Baseline (5%)
Accuracy	90.00 %	95.00 %

Problem 1: Report (15%)

1. (5%) Describe your implementation details and the difficulties you encountered.
2. (5%) Please show 10 generated images **for each digit (0-9) from both MNIST-M & SVHN dataset** in your report. You can put all 100 outputs in one image with columns indicating different noise inputs and rows indicating different digits. [see the below MNIST-M example, you should visualize **BOTH** MNIST-M & SVHN]
3. (5%) Visualize a total of six images from **both MNIST-M & SVHN datasets** in the reverse process of the **first "0"** in your outputs in (2) and with **different time steps**. [see the MNIST-M example below, but you need to visualize **BOTH** MNIST-M & SVHN]



Problem 2: DDIM (35%)

In this problem, you need to implement the DDIM algorithm to generate face images. In addition, you will need to further analyze the properties of DDIM.

- Please implement DDIM algorithm by modifying the sampling formulas of DDPM. We will provide pre-trained model weights so you do **NOT** need to modify model weight.
  - A. Implement sampling formulas of DDIM.
  - B. Define beta with the given function in **utils.py** in the GitHub template.
  - C. Implement **uniform time-step scheduler** in this task.
    - Each time-step should has an equal interval.
    - Totally **50 time-step** in this task. [notice that the timesteps is from T=1000 to T=1]
- According to the algorithm mentioned in the class, the output images of each provided noise should be the same as the ground truth **when setting η=0**.

Reference : Jiaming Song, Chenlin Meng, Stefano Ermon, “Denoising Diffusion Implicit Models” [link]

Problem 2: Evaluation (20%)

- Generate 10 face images with each provided noise as input **(by your script)** and evaluate the MSE score between your output and ground truth.
  - **You should keep η=0 in this section.**
  - using **torchvision.utils.save_image** to save image
  - The source code **(UNet.py)** is provided in the GitHub template and the model weight **(UNet.pt)** would be downloaded with **get_dataset.sh**.
  - You should name your output face images as the following format:

    (The number of the filename indicated the corresponding number of input noise.)

```
Output_folder/
    00.png
    01.png
    …
    08.png
    09.png
```

Problem 2: Evaluation (20%)

- The **MSE** between the images you generated and the ground truth should be less than the baseline in order to receive points.
- (20%) Baseline:
    - (10%) Public baseline

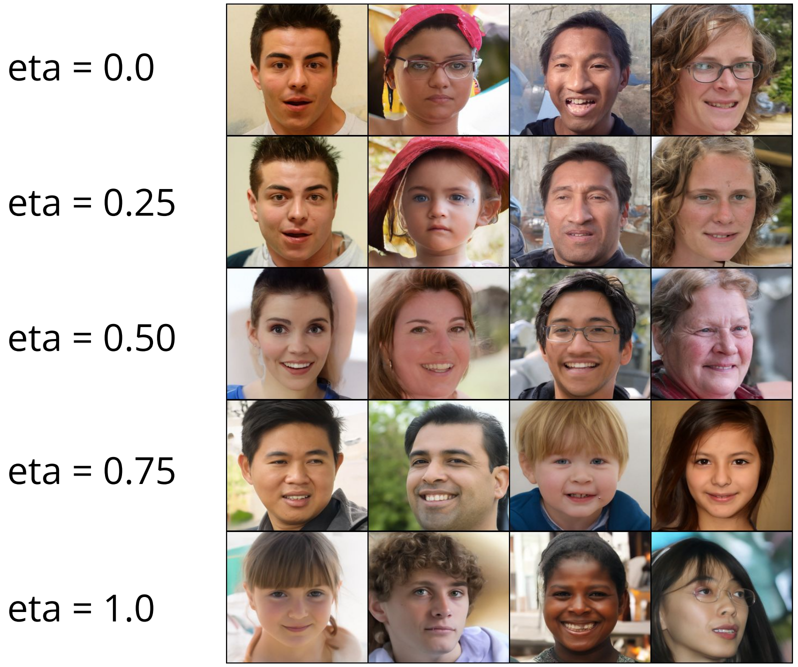
Metric	Baseline
MSE ↓	20

    - (10%) Private baseline - TBD

      We will generate the other 10 face images with private noise by your script and calculate the MSE.

Problem 2: Report (15%)

1. (7.5%) Please generate face images of noise **00.pt ~ 03.pt with different eta** in one grid. Report and explain your observation in this experiment. (This following image is just for illustration.)



Problem 2: Report (15%)

2. (7.5%) Please generate the face images of the interpolation of noise **00.pt ~ 01.pt.** The interpolation formula is **spherical linear interpolation**, which is also known as **slerp**.

$$\boldsymbol{x}_T^{(\alpha)} = \frac{\sin((1-\alpha)\theta)}{\sin(\theta)} \boldsymbol{x}_T^{(0)} + \frac{\sin(\alpha\theta)}{\sin(\theta)} \boldsymbol{x}_T^{(1)}$$

where $\theta = \arccos\left(\frac{(\boldsymbol{x}_T^{(0)})^\top \boldsymbol{x}_T^{(1)}}{\|\boldsymbol{x}_T^{(0)}\|\|\boldsymbol{x}_T^{(1)}\|}\right)$. These values are used to produce DDIM samples.

in this case, α = {0.0, 0.1, 0.2, …, 1.0}.

What will happen if we simply use linear interpolation? Explain and report your observation. (There should be **two images** in your report, one for spherical linear and the other for linear)

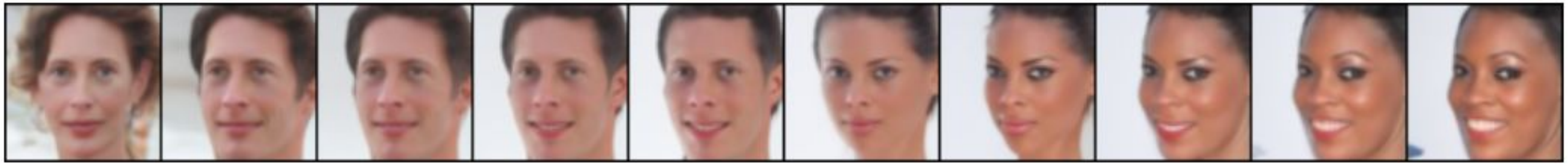


Image source: DDIM Fig.6

Problem 3: Personalization (35%)

In this problem, you need to implement **Textual Inversion** (paper link) on **different concepts.**

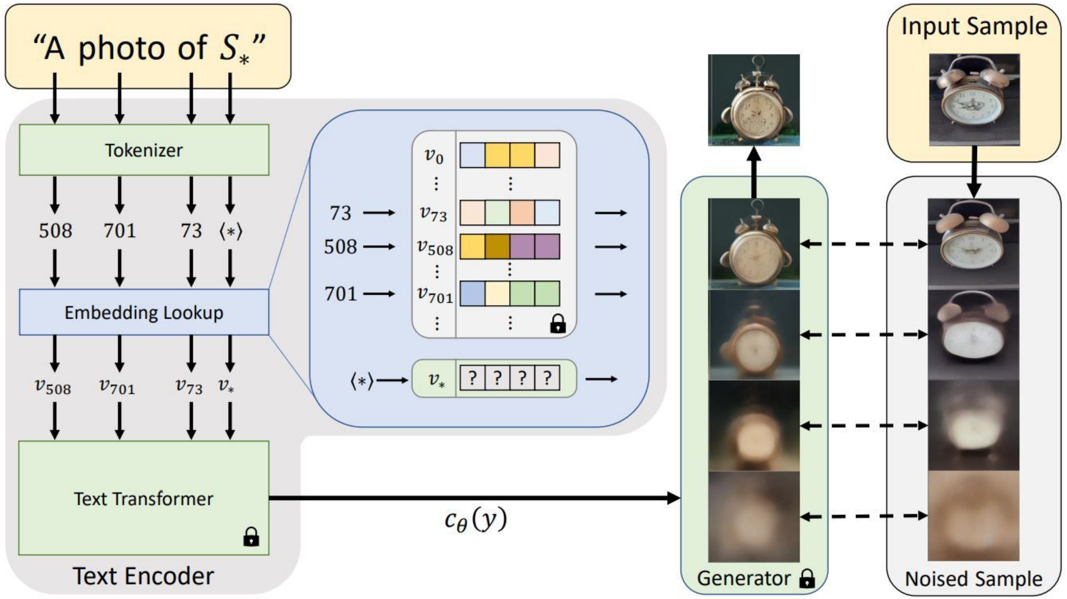


Image source: textual inversion project page

Problem 3: Personalization (35%)

In this problem, you will need to implement **Textual Inversion** to generate images of specific concepts. In addition, you will need to further analyze the properties of Textual Inversion.

We will provide a text-to-image latent diffusion model codebase (stable-diffusion/). You only need to implement Textual Inversion from this codebase. Please read README in GitHub template for details.

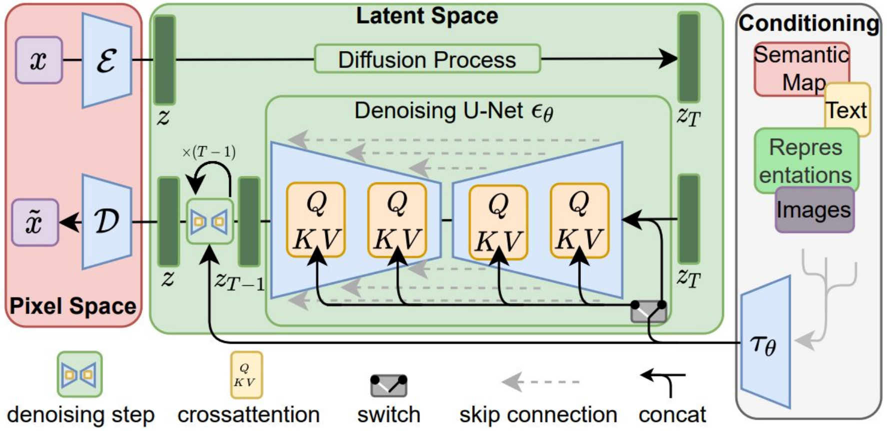
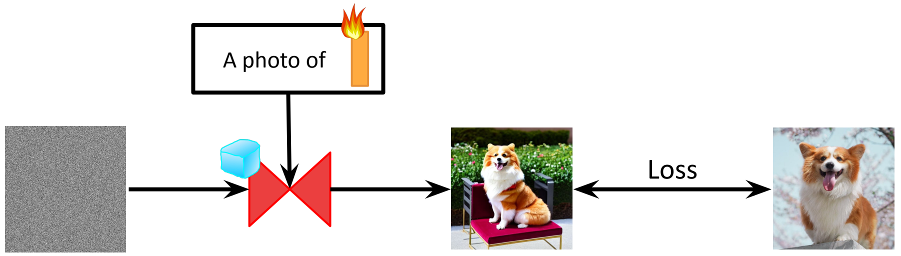


Image source: https://arxiv.org/abs/2112.10752

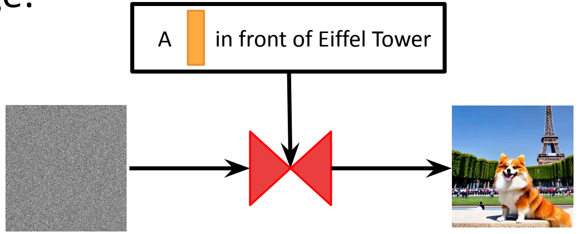
Problem 3: Evaluation (20%)

For each concept, you need to **learn a special token** with Textual Inversion, and generate images with the given prompts containing the special token.

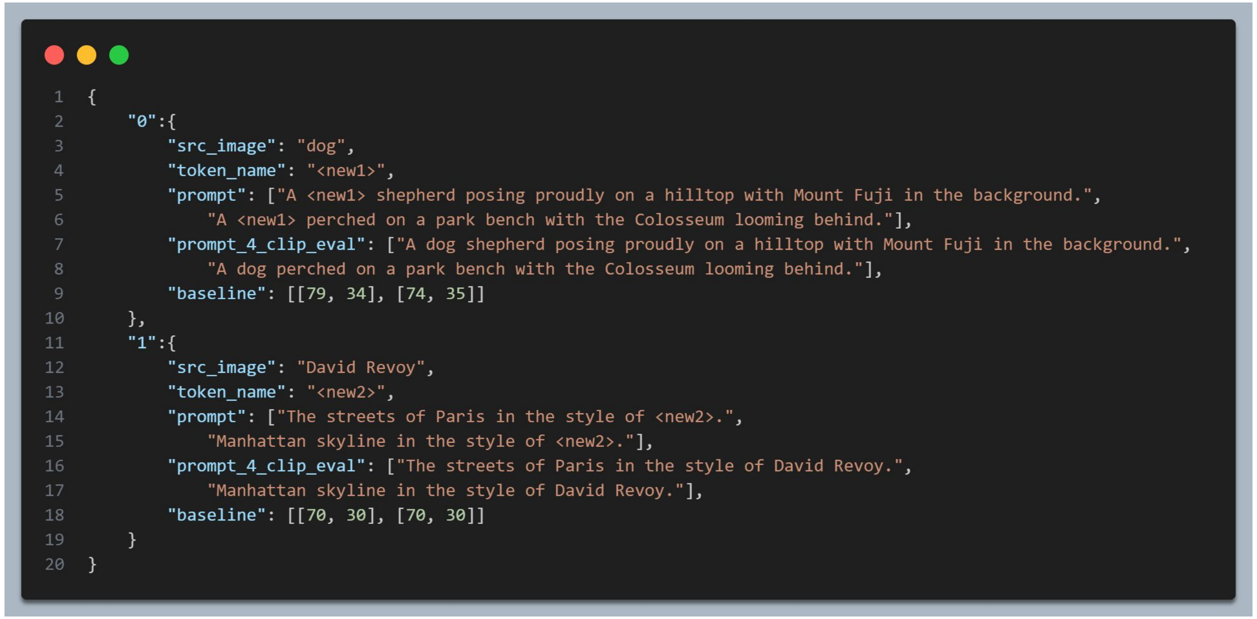
- Training Stage:



- Inference Stage:



Problem 3: Evaluation (20%)



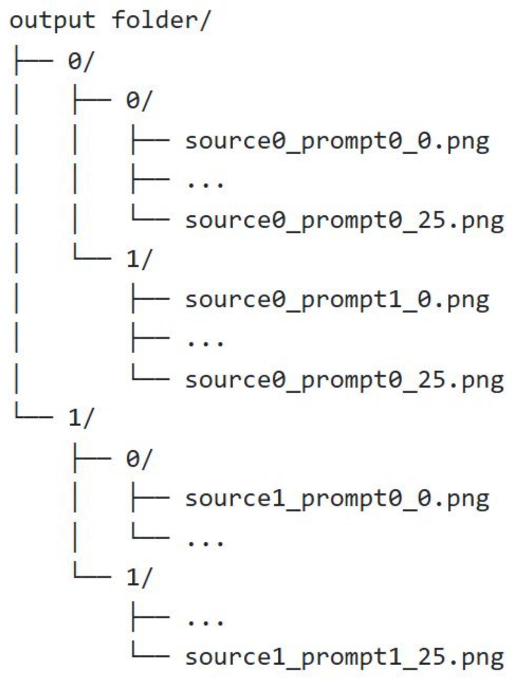
```
{
    "0":{
        "src_image": "dog",
        "token_name": "<new1>",
        "prompt": ["A <new1> shepherd posing proudly on a hilltop with Mount Fuji in the background.",
            "A <new1> perched on a park bench with the Colosseum looming behind."],
        "prompt_4_clip_eval": ["A dog shepherd posing proudly on a hilltop with Mount Fuji in the background.",
            "A dog perched on a park bench with the Colosseum looming behind."],
        "baseline": [[79, 34], [74, 35]]
    },
    "1":{
        "src_image": "David Revoy",
        "token_name": "<new2>",
        "prompt": ["The streets of Paris in the style of <new2>.",
            "Manhattan skyline in the style of <new2>."],
        "prompt_4_clip_eval": ["The streets of Paris in the style of David Revoy.",
            "Manhattan skyline in the style of David Revoy."],
        "baseline": [[70, 30], [70, 30]]
    }
}
```

Generate image based on the prompt containing the special token in the json file in the dataset. **The prompts will be changed in private set**, please use token_name to map the trained embedding.

Problem 3: Evaluation (20%)

- You must follow the output structure for the given evaluation code. The first level consists of the indices of image sources, and the second level consists of the indices of prompts.

- The index of prompts is the order specified in "prompt": [prompt0, prompt1] in input.json. Note that in the private set, the number of prompts may be different, so make sure your program can parse and output for a different number of prompts.



```
output folder/
├── 0/
│   ├── 0/
│   │   ├── source0_prompt0_0.png
│   │   ├── ...
│   │   └── source0_prompt0_25.png
│   └── 1/
│       ├── source0_prompt1_0.png
│       ├── ...
│       └── source0_prompt0_25.png
└── 1/
    ├── 0/
    │   ├── source1_prompt0_0.png
    │   └── ...
    └── 1/
        ├── ...
        └── source1_prompt1_25.png
```

⚠️ You should **fix the random seed** in your program such that the generated images are always the same.

Problem 3: Evaluation (20%)

We will calculate the image-image similarity and text-image similarity with CLIP for each prompt. You need to ***generate 25 images for each prompts.*** The top-5 images will be selected and considered based on the score = 1 * CLIP image score + 2.5 * CLIP text score.

- The source code **(evaluation/grade_hw2_3.py)** is provided in the GitHub template.
  - **Usage:** python3 –json_path <path to input json> –input_dir <path to src images> –output_dir <path to the root of output folder>
  - Please follow the saving format in the previous page so that the command can run successfully
- (10%) Public baseline

Metric (2.5% per prompt; both need to be passed for each prompt)	Baseline
Avg. of top-5 images' Image-image similarity ↑	***specified in input.json***
Avg. of top-5 images' Text-image similarity ↑	

- (10%) Private baseline - TBD

  We will change the prompts in input.json

Problem 3: Report (15%)

1. (7.5%) Conduct the CLIP-based zero shot classification on the hw2_data/clip_zeroshot/val, explain how CLIP do this, report the accuracy and 5 successful/failed cases.

- ❑ Use the prompt "A photo of {object}."
- ❑ The naming rules of Images are {class_id}_{image_id}.png.
- ❑ The id-to-label mapping is provided in hw2_data/clip_zeroshot/id2label.json
- ❑ You should be able to import clip after installing the package from environment.yaml.
- ❑ You can follow the sample code in CLIP repo.

Problem 3: Report (15%)

2. (7.5%) What will happen if you simply generate an image containing multiple concepts (e.g., a <new1> next to a <new2>)? You can use your own objects or the provided cat images in the dataset. Share your findings and survey a related paper that works on multiple concepts personalization, and share their method.

Outline

- Problems & Grading
- Dataset
- Submission & Rules

Tools for Dataset

- **Download the dataset**
  - (Option 1) Manually download the dataset here

    [2024_hw2_data.zip]

  - (Option 2) Run the bash script provided in the hw2 repository

    bash get_dataset.sh

Dataset – Digits

Format

```
hw2_data/
├── digits/
│   ├── mnistm/
│   │   ├── data/       #images for training (*.png)
│   │   └── train.csv   #label data
│   └── svhn/
│       └── ...
├── face/
│   └── ...
├── textual_inversion/
│   └── ...
└── clip_zeroshot/
    └── ...
```

Dataset – Digits

- MNIST-M Dataset
  - # of data: 56000 (training/validation)
  - # of classes: **10** (0~9)
  - Generated from MNIST
  - A subset of MNIST - The digit images are normalized (and centered) in size **28 * 28 * 3** pixels

Dataset – Digits

- SVHN Dataset
  - # of data: 79431 (training/validation)
  - # of classes: **10** (0~9)
  - Real-world image dataset for machine learning development
  - MNIST-like (size: **28 * 28 * 3**) images centered around a single character

Dataset – Face

Format

```
hw2_data/
├── digits/
│   └── ...
├── face/
│   ├── noise/   #10 noise images as public dataset (00.pt ~ 09.pt)
│   ├── GT/      #10 face images for validation (00.pn ~ 09.png)
│   └── UNet.pt  #pre-trained weight of UNet
├── textual_inversion/
│   └── ...
└── clip_zeroshot/
    └── ...
```

Dataset – Textual Inversion

Format

```
hw2_data/
├── digits/
│   └── ...
├── face/
│   └── ...
├── textual_inversion/
│   ├── 0/  #customization image source (*.jpg)
│   ├── 1/  #customization image source (*.jpg)
│   └── input.json  #prompts for evaluation, baselines
└── clip_zeroshot/
    └── ...
```

Dataset – Textual Inversion

- Artwork of David Revoy

DAVID REVOY

Image credits: @David Revoy

License: Deed - Attribution 4.0 International - Creative Commons

Dataset – CLIP Zeroshot Classification

Format

```
hw2_data/
├── digits/
│    └── ...
├── face/
│    └── ...
├── textual_inversion/
│    └── ...
└── clip_zeroshot/
     ├── val/  #image for zeroshot classification (*.png)
     └── id2label.json
```

Outline

- Problems & Grading
- Dataset
- Submission & Rules

Submission

- **Deadline:** **113/10/29 (Tue.) 23:59 (GMT+8)**
- Click the following link to get your submission repository with your GitHub account:

  https://classroom.github.com/a/3NjwmDJu

  - You should connect your Github account to the classroom with your **student ID**
  - If you cannot find your student ID in the list, please contact us (ntudlcv@gmail.com)
- By default, we will grade your last submission (commit) before the deadline (NOT your last submission). Please e-mail the TAs if you'd like to submit another version of your repository and let us know which commit to grade.
- We will clone the **main** branch of your repository.

Submission

- Your GitHub repository **DLCV-Fall-2024/hw2-{GitHub_ID}** should include the following files:
    - hw2_<studentID>.pdf (report)
    - hw2_1.sh (for Problem 1)
    - hw2_2.sh (for Problem 2)
    - hw2_3.sh (for Problem 3)
    - your python files (e.g., training code & inference code)
    - your model files (can be loaded by your python file)
- **Don’t push the dataset to your repo.**
- If any of the file format is wrong, you will get zero point.

Shell Script (Problem 1) – hw2_1.sh

- Please provide a **script** to the specified directory with your model, and save the 1000 (500 images for each dataset) generated images into the specified directory.
- TAs will run your script as shown below:
  - bash hw2_1.sh $1
    - $1: path to the directory for your 1000 generated images (e.g. "~/hw2/DDPM/output_images")
- This section must be finished in **15 mins**, otherwise would be considered as a failed run.

⚠️ You should **follow the filename format** for different digit images as described in Problem 1

Shell Script (Problem 2) – hw2_2.sh

- Please provide a **script** to the specified directory with your model, and save the 10 generated images into the specified directory.
- TAs will run your script as shown below:
  - bash hw2_2.sh $1 $2 $3
    - $1: path to the directory of predefined noises (e.g. "~/hw2/DDIM/input_noise")
    - $2: path to the directory for your 10 generated images (e.g. "~/hw2/DDIM/output_images")
    - $3: path to the pretrained model weight(e.g. "~/hw2/DDIM/UNet.pt")
- This section must be finished in **5 mins**, otherwise would be considered as a failed run.

⚠️ You should **follow the filename format** for different face images as described in Problem 2

Shell Script (Problem 3) – hw2_3.sh

- Please provide a **script** for the specified input JSON file and save the generated results in the specified folder, following the specified structure.
- TAs will run your script as shown below:
  - bash hw2_3.sh $1 $2 $3
    - $1: path to the json file containing the testing prompt (e.g. "~/hw2_data/textual_inversion/input.json")
    - $2: path to your output folder (e.g. "~/output_folder")
    - $3: path to the pretrained model weight (e.g. "~/hw2/personalization/model.ckpt")
- This section must be finished in **15 mins**, otherwise would be considered as a failed run.

⚠️ **The directory structure of output_folder should be the same as output_folder_example/ provided in the repository.**

Rules – Submission

- If your model checkpoints are larger than GitHub's maximum capacity (50 MB), you could download and preprocess (e.g. unzip, tar zxf, etc.) them in hw2_download.sh.
    - TAs will run `bash hw2_download_ckpt.sh` prior to any inference if the download script exists, i.e. it is **NOT** necessary to create a blank `hw2_download_ckpt.sh` file.
- Do **NOT** delete your model checkpoints before the TAs release your score and before you have ensured that your score is correct.

Rules – Submission

- [Recommend] Please use **wget** to download the model checkpoints from cloud drive (e.g. Dropbox/Onedrive) or your working station.
    - You should use **-O argument** to specify the filename of the downloaded checkpoint.
    - Please refer to this Dropbox Guide for a detailed tutorial.
- Google Drive is a widely used cloud drive, so it is allowed to use **gdown** to download your checkpoints from your drive.
    - It is also recommended to use -O argument to specify the filename.
    - **Remember to set the permission visible to public, otherwise TAs are unable to grade your submission, resulting in zero point.**
    - If we could not download your model on Google Drive due to Google policy, you will need to provide the evidence that you had **set the permission visible to public BEFORE deadline** for us, then we will manually download your models and run your scripts.

Rules – Environment

- Ubuntu 22.04.4 LTS
- NVIDIA RTX A4500(20 GB)
- GNU bash, version 5.1.16(1)-release
- Python 3.8.5

Rules – Environment

- Ensure your code can be executed successfully on **Linux** system before your submission.
- Use only **Python3** and **Bash** script conforming to our environment, do not use other languages (e.g. CUDA) and other shell (e.g. zsh, fish) during inference.
  - Use the command **“python3”** to execute your testing python files.
- You must **NOT** use commands such as **sudo, CUDA_VISIBLE_DEVICES** or other commands to interfere with the environment; **any malicious attempt against the environment will lead to zero point in this assignment**.
- You shall **NOT hardcode any path** in your python files or scripts, while the dataset given would be the absolute path to the directory.

Rules – Packages

- numpy: 1.24.4
- torch: 1.11.0
- torchvision: 0.12.0
- transformers: 4.19.2
- timm: 0.8.19.dev0
- pandas: 2.0.3
- and other standard python packages
- matplotlib: 3.7.0
- Pillow: 10.4.0
- imageio: 2.9.0
- scipy: 1.9.1
- scikit-image: 0.20.0
- tqdm, gdown, glob, yaml, detectors, argparser, etc.
- please make sure your environment can run stable-diffusion/scripts/txt2img.py

- **E-mail or ask TA first if you want to import other packages.**

Rules – Packages

- Do not use **imshow()** or **show()** in your code or your code will crash.
- Use **os.path.join** to deal with path as often as possible.

Rules – Policy

- **Late policy:** We provide a total of **three free late days** for all four homework submissions this semester. After that, late homework will be deducted by **30%** each day.
- Students are encouraged to discuss the assignment, but you must complete the assignment by yourself. TA will compare the similarity between everyone's assignment. Any form of cheating or plagiarism will not be tolerated, which will also result in F for students with such misconduct.
- Please specify, if any, the **references** for any parts of your HW solution in your report (e.g., your collaborators or the GitHub source code).
- **Using external dataset is forbidden for this homework.**

Rules – Code Modification

- If your code cannot be executed, you have a chance to make minor modifications to your code. After modifying your code,
  - If we can execute your code, you will receive a **30% penalty** in your model performance score.
  - If we still cannot execute your code, no points will be given.
- TAs will release the log of execution after grading, please check.
  - Email the TAs if something goes wrong in your submission.

How to Find Help

- Google!
- Use TA hours (please check course website for time/location)
    - Please seek help from **the TAs in charge of this assignment as possible as you can**.
    - Thu. 14:20～15:10 in MK-514
- Post your question to NTU COOL
- Contact TAs by e-mail: ntudlcv@gmail.com

DOs and DON’Ts for the TAs (& Instructor)

- Do NOT send private messages to TAs.
    - TAs are happy to help, but they are not your tutors 24/7.
- TAs will NOT debug for you, including addressing coding, environmental, library dependency problems.
- TAs do NOT answer questions not related to the course.
- If you cannot attend the TA hours, please email the TAs to schedule an appointment instead of stopping by the lab directly.