Homework #2

Deep Learning for Computer Vision
NTU, Fall 2023

Problems – Overview

- **Diffusion Models**
  - Problem 1: Conditional Diffusion models (35%) [digit dataset - MNIST-M]
  - Problem 2: DDIM (35%) [face dataset]
- **Unsupervised Domain Adaptation (UDA)**
  - Problem 3: DANN (35%) [digit dataset - MNIST-M, SVHN and USPS]

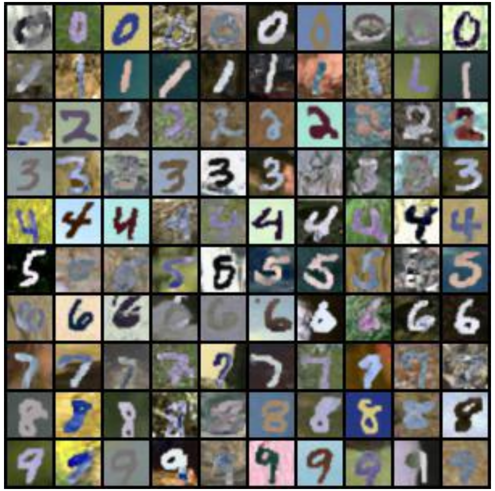
Please refer to “Dataset” section for more details about face and digit datasets.

Outline

- Problems & Grading
- Dataset
- Submission & Rules

Problem 1: Diffusion models (35%)

In this problem, you will implement **conditional** diffusion model from scratch and train it on the **MNIST-M dataset (inside the digit dataset)**. Given conditional labels 0-9, your model need to generate the corresponding digit images as the following example.



Problem 1: Diffusion models (35%)

- 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: $\quad \mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $\quad t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: $\quad$ 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: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ 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** $\mathbf{x}_0$

- The concept/implementation of **conditional** diffusion models is similar to conditional GANs (e.g., ACGAN)

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

Problem 1: Evaluation (15%)

- Sample random noise from normal distribution to generate **100** conditional images **for each digit (0-9)**. 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/
    0_001.png
    0_002.png
    ...
    0_100.png
    1_001.png
    ...
    9_100.png
```

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

Problem 1: Evaluation (15%)

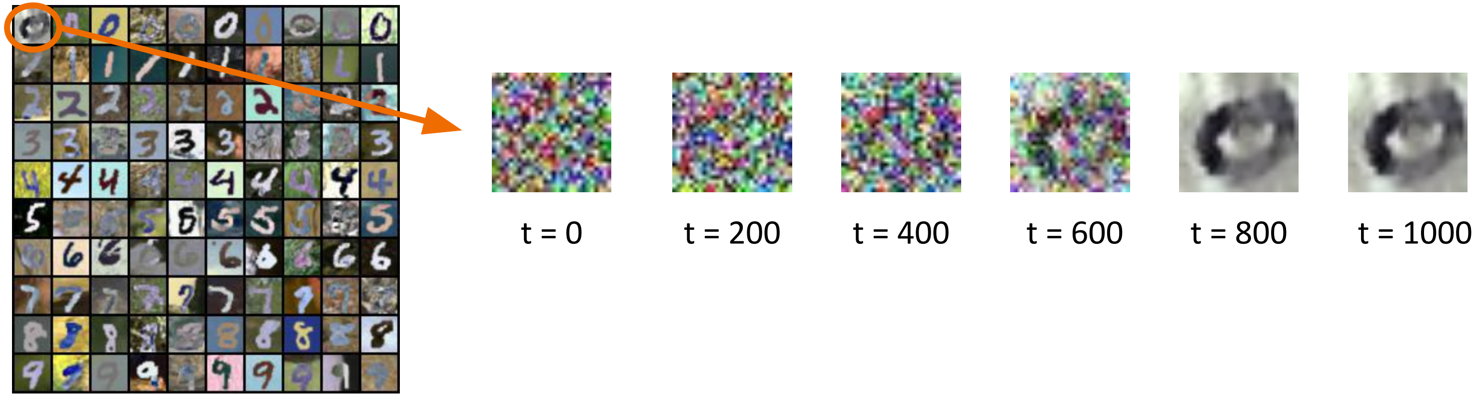
- We will use a **digit classifier** to evaluate your generated images by **classification accuracy**.
    - The source code **(digit_classifer.py)** and the model weight **(Classifier.pth)** is provided in the GitHub template.
        - **Usage:** python3 digit_classifier.py --folder <path_to_output_folder>
        - 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 (20%)

1. (5%) Follow the Github Example to draw your model architecture and describe your implementation details.
2. (5%) Please show 10 generated images **for each digit (0-9)** 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 example]
3. (5%) Visualize total six images in the reverse process of the **first “0”** in your grid in (2) **with different time steps**. [see the below example]
4. (5%) Please discuss what you’ve observed and learned from implementing conditional diffusion model.



Problem 2: DDIM (35%)

In this problem, you will 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.
- 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.**
  - 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

02.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.)

eta = 0.0

eta = 0.25

eta = 0.50

eta = 0.75

eta = 1.0

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.

(This following image is just for illustration.)

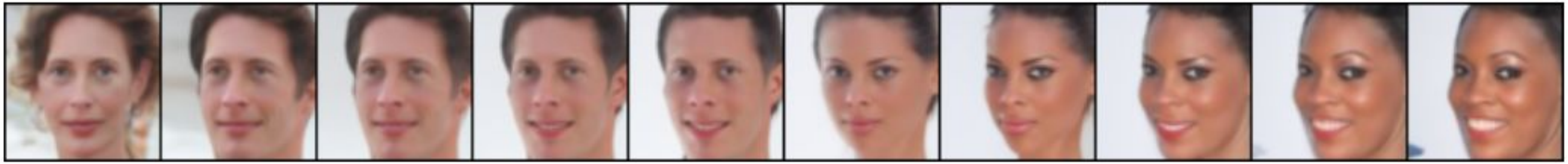


Image source: DDIM Fig.6

Problem 3: DANN (35%)

For unsupervised domain adaptation, you will need to implement **DANN** (paper link) for image classification on the **digit datasets**, and consider the following 2 scenarios

(a) MNIST-M → SVHN (b) MNIST-M → USPS (source domain → target domain)

Conduct the following experiments to confirm the effectiveness of your method:

1. **(Lower bound)** Compute the accuracy on **target** domain, while the model is trained on **source** domain.
   - Please use **source** images and labels in "train.csv" for training, **target** images and labels in "val.csv" to evaluate
2. **(DANN)** Compute the accuracy on **target** domain, while the model is trained with DANN.
   - You can utilize **both** *images and labels* in the source domain, but **only** *images* in the target domain.
   - Please use **source** *images and labels* in "train.csv" + **target** *images* in "train.csv" for training, **target** *images and labels* in "val.csv" to evaluate
3. **(Upper bound)** Compute the accuracy on **target** domain, while the model is trained on **target** domain.
   - Please use **target** *images and labels* in "train.csv" for training, **target** *images and labels* in "val.csv" to evaluate

Problem 3: Evaluation (12%)

- Baseline (classification accuracy on the **target** domain):
  - (6%) Public baseline:

	MNIST-M $\rightarrow$ SVHN (3%)	MNIST-M $\rightarrow$ USPS (3%)
Adaptation (DANN)	40%	76%

  - (6%) Private baseline - TBD

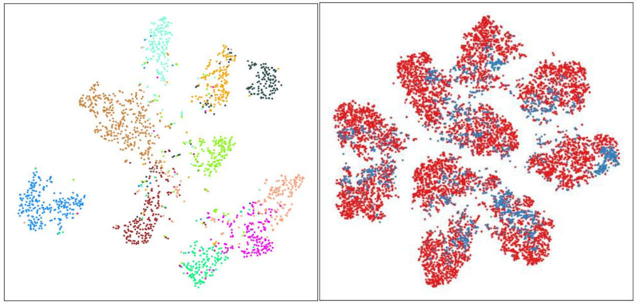
Problem 3: Report (23%)

1. (10%) Please create and fill the table with the following format **in your report**:

	MNIST-M → SVHN	MNIST-M → USPS
Trained on source		
Adaptation (DANN)		
Trained on target		

Problem 3: Report (23%) (cont'd)

2. (10%) Please visualize the latent space **(output of CNN layers)** of DANN by mapping the ***validation*** images to 2D space with t-SNE. For each scenario, you need to plot two figures which are colored **by digit class (0-9)** and **by domain**, respectively.
   - Note that you need to plot the figures of both **2 scenarios**, so **4 figures** in total.



(a) by class **(b) by domain**

3. (3%) Please describe the implementation details of your model and discuss what you've observed and learned from implementing DANN.

Outline

- Problems & Grading
- Dataset
- Submission & Rules
- Training Tips

Tools for Dataset

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

    2023_hw2_data.zip
  - (Option 2) Run the bash script provided in the hw2 repository

    bash get_dataset.sh

Dataset – Face

Format

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

Dataset – Digits

Format

```
hw2_data/
└── digits/
    ├── usps/
    │   ├── data/      # images for training and validation (*.png)
    │   ├── train.csv  # labels for training (0, 1, 2, …, 9)
    │   ├── val.csv    # labels for validation (0, 1, 2, …, 9)
    │   └── test.csv   # sample format
    ├── svhn/
    │       ...
    └── mnistm/
            …
└── face/
```

Dataset – Digits

- USPS Dataset
  - # of data: 5,950 / 1,488 (training/validation)
  - # of classes: **10** (0~9)
  - Image size: **28 * 28 * 1**

- MNIST-M Dataset
  - # of data: 44,800 / 11,200 (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: 63,544 / 15,887 (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

⚠️You need to deal with the channel difference between datasets by yourself.

Outline

- Problems & Grading
- Dataset
- Submission & Rules
- Training Tips

Submission

- **Deadline: 112/11/7 (Tue.) 23:59 (GMT+8)**
- Click the following link to get your submission repository with your GitHub account:

  https://classroom.github.com/a/pA8ESRLP

  - 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-2023/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 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** to the specified directory with your model, and save the classification results in the specified csv file.
- TAs will run your script as shown below:
  - bash hw2_3.sh $1 $2
    - $1: path to testing images in the target domain
      (e.g. "~/hw2_data/digits/svhn/test" for MNIST-M→SVHN
      and "~/hw2_data/digits/usps/test" for MNISTM→USPS)
    - $2: path to your output prediction file (e.g. "~/test_pred.csv")
- This section must be finished in **10 mins**, otherwise would be considered as a failed run.

⚠️ **The format of test_pred.csv should be the same as test.csv provided in the dataset.**
(detailed in next page)

Sample CSV Format (Problem 3)

- Predict class labels for all images
  - Output format: csv file
  - The first row must be: 'image_name,label'
  - The format should be the same as test.csv

image_name	label
00000.png	0
00001.png	0
00002.png	0
00003.png	0
00004.png	0
00005.png	0
00006.png	0
00007.png	0
00008.png	0
00009.png	0

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.sh` prior to any inference if the download script exists, i.e. it is **NOT** necessary to create a blank `hw2_download.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 20.04.1 LTS
- NVIDIA GeForce RTX 2080 Ti (11 GB)
- GNU bash, version 5.0.17(1)-release
- Python 3.10

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: 1.23.1
- torch: 2.0.1
- torchvision: 0.15.2
- scikit-learn: 1.2.1
- timm: 0.6.7
- transformers: 4.35.0
- and other standard python packages
- matplotlib: 3.7.0
- Pillow: 9.4.0
- imageio: 2.26.0
- scipy: 1.10.0
- scikit-image: 0.19.3
- pandas: 1.5.3
- tqdm, gdown, glob, yaml

- **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**.
  - Fri. 11:20～12:10 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.