
Homework #2

Deep Learning for Computer Vision
NTU, Fall 2022

Problems – Overview

- **Image Generation**

- Problem 1: GAN (35%) [face dataset - CelebA]
- Problem 2: Diffusion models (35%) [digit dataset - MNIST-M]

- **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
- Training Tips

Problem 1: GAN (35%)

In this problem, you will need to implement GANs and train them on the **face dataset** (from scratch). In addition, you will need to compute some scores to analyze the performance of your models.

- Please build the following GAN models and **train from scratch**:
 - A. DCGAN ([paper link](#))
 - B. **Improve your GAN with any method** (modify the model architecture or learning objectives, implement other methods of recent papers, etc.)

❌ **Style-based GANs (e.g., StyleGAN, StyleGAN2) are prohibited**

❌ **Pre-trained model weights are prohibited**



Problem 1: Evaluation (20%)

- Generate 1000 face images (by your script) and evaluate them with the following two metrics:
 - (1) Fréchet inception distance (FID)
 - We will use this package for evaluation: <https://github.com/mseitzer/pytorch-fid>
 - (2) Face Recognition
 - We employ **HOG** algorithm for feature extraction
 - Please refer to the provided script “face_recog.py”
 - **Usage:** `python3 face_recog.py --image_dir <path_to_output_folder>`



You should **fix the random seed** in your program such that the generated images are always the same. (for grading)

Problem 1: Evaluation (20%)

- (20%) Baseline:

- (10%) Public baseline

Metric	Simple Baseline	Strong Baseline
FID ↓	30.00 (2.5%)	27.00 (2.5%)
Face Recognition ↑	85.00 % (2.5%)	90.00 % (2.5%)

FID will be computed with your generated images and the 2,025 face images in the validation set.
You are NOT allowed to train your models with validation set.

- (10%) Private baseline - TBD (Only FID will be computed in this part)

FID will be computed with your generated images and the other 2,132 face images (private set)

- You only need to submit **one** model (either A or B) for the above public / private evaluation.



Note that your generated images will be evaluated with both validation set and private set.

Problem 1: Report (15%)

1. (5%) Please print the model architecture of method A and B.
 - You can use “print(model)” directly.
2. (5%) Please show the first 32 generated images of both method A and B then discuss the difference between method A and B.

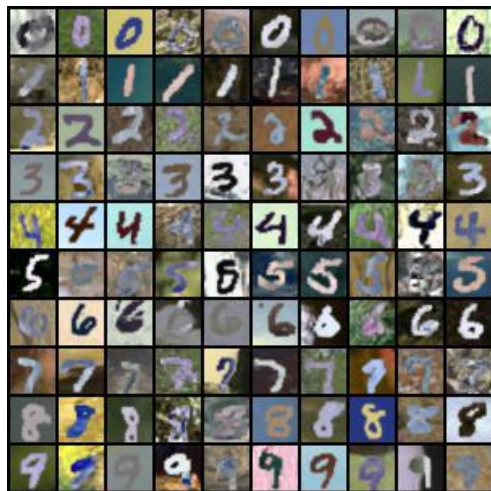


Example for the first 32 generated images

3. (5%) Please discuss what you've observed and learned from implementing GAN.
 - You can compare different architectures or describe some difficulties during training, etc.

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

Algorithm 2 Sampling

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

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

Problem 2: 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.

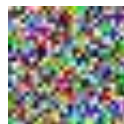
Problem 2: Evaluation (15%)

- We will use a **digit classifier** to evaluate your generated images by **classification accuracy**.
 - The source code (**digit_classifier.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:

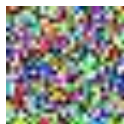
Metric	Simple Baseline (10%)	Strong Baseline (5%)
Accuracy	80.00 %	88.00 %

Problem 2: Report (20%)

1. (5%) Please print 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.



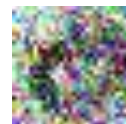
t = 0



t = 200



t = 400



t = 600



t = 800



t = 1000

Problem 3: DANN (35%)

For unsupervised domain adaptation, you 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

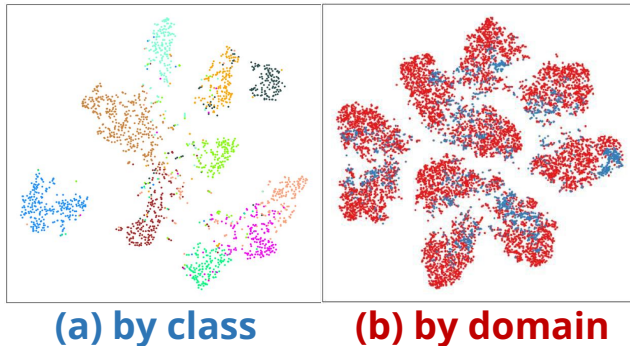
Problem 3: Report (23%)

1. (5%) Please create and fill the table with the following format **in your report**:

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

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

2. (8%) Please visualize the latent space 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.



3. (10%) 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

<https://drive.google.com/file/d/1YxkObGDIqZM0-9Zq-QMjk7q1vND4UJl3/view?usp=sharing>

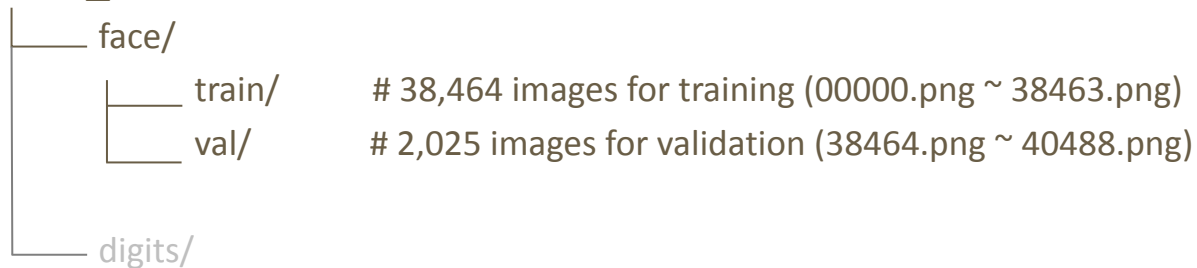
- (Option 2) Run the bash script provided in the hw2 repository

`bash get_dataset.sh`

Dataset – Face

Format

hw2_data/



Dataset – Face

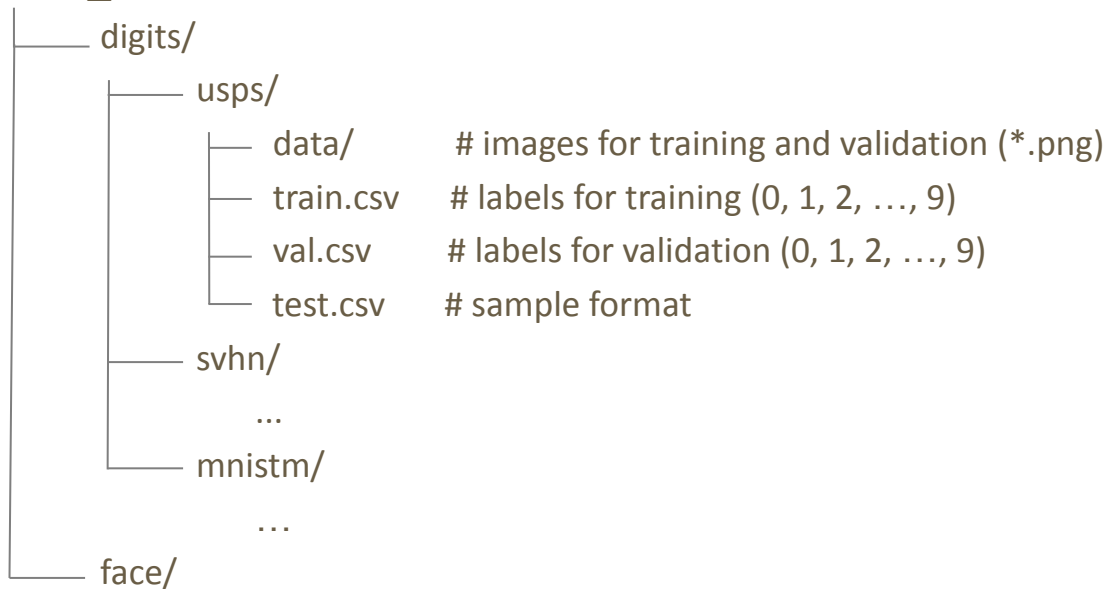
A subset of human face dataset CelebA

- Images are cropped and downscaled to 64 X 64
- 38,464 training samples (about 20% of complete CelebA)

Dataset – Digits

Format

hw2_data/



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: 111/10/31 (Mon.) 23:59 (GMT+8)**
- Click the following link to get your submission repository with your GitHub account:
<https://classroom.github.com/a/hlUQFicD>
 - 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-2022/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/GAN/output_images”)
- This section must be finished in **10 mins**, otherwise would be considered as a failed run.

Shell Script (Problem 2) – hw2_2.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_2.sh $1`
 - \$1: path to the directory for your 1000 generated images (e.g. “~/hw2/Diffusion/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 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

- Please use **wget** to download the model checkpoints from cloud drive (e.g. Dropbox) 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 you have set the permission correspondingly but failed to download with **gdown** because of Google's policy, TAs will manually download them, no worries!!

Rules – Environment

- Ubuntu 20.04.1 LTS
- NVIDIA GeForce RTX 2080 Ti (11 GB)
- GNU bash, version 5.0.17(1)-release
- Python 3.8

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: 1.12.1
 - torchvision: 0.13.1
 - scikit-learn: 1.1.2
 - timm: 0.6.7
 - transformers: 4.21.3
 - and other standard python packages
 - matplotlib: 3.5.3
 - pillow: 9.2.0
 - imageio: 2.21.2
 - scipy: 1.9.1
 - scikit-image: 0.18.1
 - pandas: 1.5.0
 - face-recognition: 1.3.0
 - 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)
- 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 via Facebook.
 - 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.

Outline

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

Training Tips – GAN

Well known [tips and tricks](#) published on GitHub.

It is suggested that you take a look before you start training.

TA's experience and hints

- Surveying related papers and using similar architectures may help.
- Trace the accuracy of discriminator network to see if Gnet and Dnet performance matches.
- Improved GAN algorithm is harder to implement but easier to train (e.g. WGAN, WGAN-GP)

GAN is often difficult to train and tune, starting this part early may help a lot.