Homework #4

Deep Learning for Computer Vision

NTU, Fall 2022

111/11/22

111/12/19 (Mon.) 11:59 PM (GMT+8) due

Outline

- Problems & Grading
- Dataset
- Submission & Rules
- Supplementary

Problems – Overview

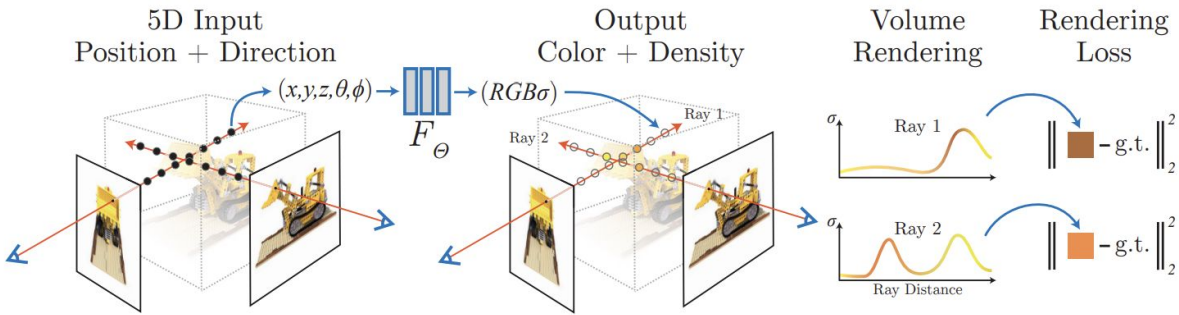
- **Problem 1:** 3D Novel View Synthesis (50%) [hw4_data/hotdog]
- **Problem 2:** Self-supervised Pre-training for Image Classification (50%) [hw4_data/mini], [hw4_data/office]

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

Problem 1: 3D Novel View Synthesis (50%)

In this problem, you will have to train your own **NeRF** model on a **one object scene(i.e., the hotdog)**. To be more specific, given a set of training images (with camera pose) of this scene, make your model fit this scene. After that, given a test set camera pose, the model should be able to synthesize these unseen novel views of this scene.

- Grading:
  - Report (30%)
  - Model Performance (20%)



Reference: [NeRF: Representing Scene as Neural Radiance Fields for View Synthesis]
[Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction]

Problem 1: 3D Novel View Synthesis (50%)

1. (5%) Please explain:

   a. the NeRF idea in your own words

   b. which part of NeRF do you think is the most important

   c. compare NeRF's pros/cons w.r.t. other novel view synthesis work

Please read through these two reference paper to get their ideas.

Reference: NeRF: Representing Scene as Neural Radiance Fields for View Synthesis
Direct Voxel Grid Optimization: Super-fast Convergence for Radiance Fields Reconstruction(DVGO)

Problem 1: 3D Novel View Synthesis (50%)

2. (10%) Describe the implementation details of Direct Voxel Grid Optimization(DVGO) for the given dataset. You need to explain DVGO's method in your own ways.
   - **You have to train a novel view synthesis model on the given dataset without loading pretrained weights.**
   - **We provide the following Github links for your reference.**
     - **NeRF - LINK1/LINK2, DVGO - LINK (Recommended, faster), etc.**
     - **NeRF training time for a scene could take almost half a day, so make sure you start earlier.**

Problem 1: 3D Novel View Synthesis (50%)

3. (15%) Given novel view camera pose from **transforms_val.json**, your model should render novel view images. Please evaluate your generated images and ground truth images with the following three metrics (mentioned in the NeRF paper). Try to use at least two different hyperparameter settings and discuss/analyze the results.
   - **Please report the PSNR/SSIM/LPIPS on the validation set. (refer to DVGO's github)**
   - **You also need to explain the meaning of these metrics.**
   - **Different configuration settings such as iteration number/number of voxel/stepsize … lead to different performance.**

Setting	PSNR	SSIM	LPIPS
Setting 1 (You need to write your setting)	*TODO*	*TODO*	*TODO*
Setting 2 (You need to write your setting)	*TODO*	*TODO*	*TODO*

Problem 1: 3D Novel View Synthesis (50%)

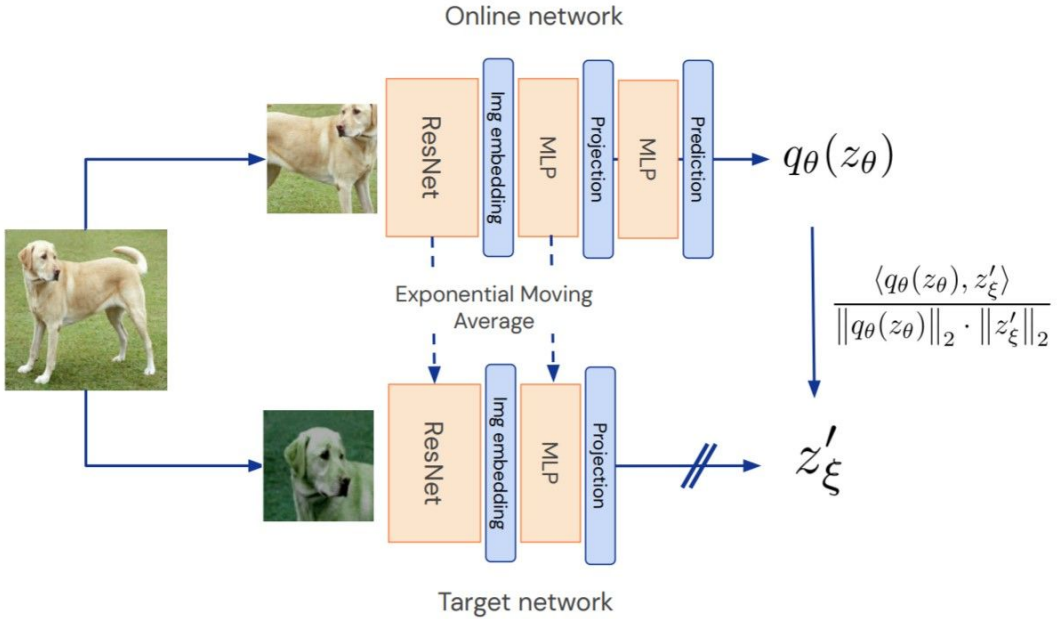
Model Performance (20%)

- PSNR and SSIM scores should **both** be above the baseline scores to get points
  - Public baseline (on the validation set):
    - Baseline (10%):
      - PSNR: **35**
      - SSIM: **0.97**
  - Private baseline (on the test set):
    - Baseline (10%):
      - PSNR: **TBD**
      - SSIM: **TBD**

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

In this problem, you will have to pre-train your own ***ResNet50*** backbone on **Mini-ImageNet** via the recently self-supervised learning methods. After that, you need to conduct image classification on **Office-Home** dataset with different settings to analyze your pre-trained backbone.

- Grading:
  - Report (30%)
  - Model Performance (20%)



Reference: [Bootstrap Your own Latent]

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

1. (10%) Describe the implementation details of your SSL method for pre-training the ResNet50 backbone. (Include but not limited to the name of the SSL method you used, data augmentation for SSL, learning rate schedule, optimizer, and batch size setting for this pre-training phase)
   - **You have to pre-train the backbone on the Mini-ImageNet without loading default pretrained weights.**
   - **We recommend you to reference the following Github. (It's easy to understand, use, and train with relatively small batch size.)**
     - **BYOL - LINK (Recommended), Barlow Twins - LINK, etc.**
   - **Since the pre-training phase may take weeks/months to finish under the general SSL setting, we fix the following training setting to reduce the training time. (You MUST follow this setting in the pre-training and fine-tuning phase for fair comparison.)**
     - **Image size: 128*128**
     - **Backbone: ResNet50 (You can choose whether to use the FC layer of ResNet50)**

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

2. (20%) Please conduct the Image classification on **Office-Home** dataset as the downstream task. Also, please complete the following Table, which contains different image classification setting, and **discuss/analyze** the results.
   - **Please report the mean classification accuracy on validation set.**
   - **TAs will run your code to verify the performance of the setting C in the Table below.**
   - **The architecture of the classifier needs to be consistent across all settings.**

Setting	Pre-training (Mini-ImageNet)	Fine-tuning (Office-Home dataset)	Validation accuracy (Office-Home dataset)
A	-	Train full model (backbone + classifier)	*TODO*
B	w/ label (TAs have provided this backbone)	Train full model (backbone + classifier)	*TODO*
C	w/o label (Your SSL pre-trained backbone)	Train full model (backbone + classifier)	*TODO*
D	w/ label (TAs have provided this backbone)	Fix the backbone. Train classifier only	*TODO*
E	w/o label (Your SSL pre-trained backbone)	Fix the backbone. Train classifier only	*TODO*

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

Model Performance (20%)

- Classification accuracy (mean) under setting C in Problem 2-2 should be above the baseline score to get points
    - Public baseline (on the validation set):
        - Simple baseline (5%): **0.36**
        - Strong baseline (5%): **0.40**
    - Private baseline (on the test set):
        - Simple baseline (5%): **TBD**
        - Strong baseline (5%): **TBD**
- TAs will execute your code to check if you pass the private baseline.
- Only TAs have the test data. (testing data is not available for students)

Problem 2: Self-Supervised Pre-training for Image Classification (50%)

Provided weights:

- We provide the supervised pre-trained weights, named "**pretrain_model_SL.pt**", of ResNet50 (You can find it in hw4_data/) for you to analyze the setting B and D. This model is pre-trained on the Mini-ImageNet with class labels as supervision and follows the training setting below.
    - backbone: ResNet50 (resnet = torchvision.models.resnet50(pretrained=False))
    - Image size: 128 * 128
    - Transformation:

```
TRANSFORM_IMG = transforms.Compose([
    transforms.Resize(128),
    transforms.CenterCrop(128),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                         std=[0.229, 0.224, 0.225] )
    ])
```

Reminder

- Please start working on this homework earlier. The training may take a few hours/**days** on a GPU or **weeks** on CPUs.
- Please follow the rules shown in “Submission & Rules” section.

Outline

- Problems & Grading
- Dataset
- Submission & Rules
- Supplementary

Problem 1 Dataset: Hotdog object

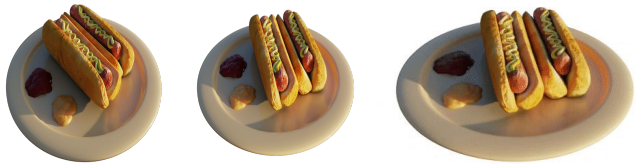
- The dataset consists of 100 RGB images of size 800 X 800 of the same object scene (hotdog).

- In the dataset, you will get

hw4_data/hotdog/

```
├── train/                  # training images directory (50 images)
├── transforms_train.json   # training image camera pose file
├── val/                    # validation images directory (50 images)
└── transforms_val.json     # validation image camera pose file
```

- You **CANNOT** use validation data for training purposes.
- Preprocess code can be found on DVGO's github

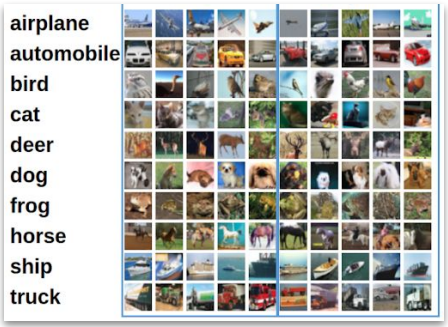


Problem 2 Dataset: Mini-ImageNet

- The dataset consists of 48,000 84x84 RGB images in 80 classes.
- In the dataset, you will get part of dataset for training your backbone.

hw4_data/mini/

```
├ train/        # training images directory (64 class, each class has 600 images)
└ train.csv     # training image csv file
```



Problem 2 Dataset: Office-Home

- The dataset consists of 3,951/406 RGB images in 65 classes for train/valid set.
- In the dataset, you will get

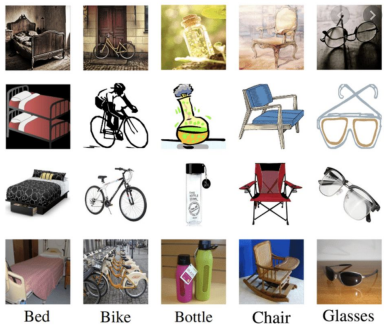
hw4_data/office/

train/	# training images directory (65 classes, total 3951 images)
val/	# validation images directory (65 classes, total 406 images)
train.csv	# train image csv file
val.csv	# validation image csv file

- The .csv files of the Office-Home and Mini-ImageNet are in the **same** format.
- You **CANNOT** use validation data for training purposes.
- All the testing set files are in the **same** format as the validation set files.

Header in first row: <image_id>, <filename>, <label>

	A	B	C
1	id	filename	label
2	0	Couch00015.jpg	Couch
3	1	Helmet00018.jpg	Helmet
4	2	Refrigerator00011.jpg	Refrigerator
5	3	Alarm_Clock00061.jpg	Alarm_Clock
6	4	Bike00088.jpg	Bike



Tools for Dataset

- Download the dataset
  - // gdrive link

    https://drive.google.com/file/d/1Tc0f28syYVE185Z6388DWUOVHGSANCmX/view?usp=share_link
  - // gdown

    gdown 'https://drive.google.com/u/0/uc?id=1Tc0f28syYVE185Z6388DWUOVHGSANCmX&export=download'

Outline

- Problems & Grading
- Dataset
- Submission & Rules
- Supplementary

Submission

- Click the following link and sign in to your GitHub account to get your submission repository TBD

  https://classroom.github.com/a/_4kiLuFw

  - 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 only grade your last submission 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 should include the following files
  - hw4_<studentID>.pdf (report)
  - hw4_1.sh (for Problem 1)
  - hw4_2.sh (for Problem 2)
  - Python files (e.g., training code & inference code & visualization code)
  - 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.

Bash Script - Problem 1

- Please provide a script to the specified json file which contain camera pose, and save the generated images into the specified directory.
- TA will run your code as shown below target
  - bash hw4_1.sh $1 $2
    - $1: path to the **transform_test.json** (e.g., */*/transform_test.json)

      which contains camera poses with the same format as in transfor_train.json, you should predict novel views base on this file.
    - $2: path of the folder to put output **image** (e.g., *.png)

      the filename should be same as stated in transform_test.json file. The image size should be the same as training set, 800x800 pixel.
- Please follow the naming rules strickly
- Note that you should **NOT** hard code any path in your file or script.
- Your testing code have to be finished in **10 mins.**
- We provide the grade.py for grading, see the supplementary page

Bash Script - Problem 2

- Please provide a script to run your model by your setting C, and generate .csv prediction file.
- TA will run your code as shown below
  - bash hw4_2.sh $1 $2 $3
    - $1: path to the images csv file (e.g., hw4_data/office/test.csv)
    - $2: path to the **folder** containing images (e.g. hw4_data/office/test/)
    - $3: path of output **csv file** (predicted labels) (e.g., hw4/output_p2/test_pred.csv)
- Please follow the order of your input .csv file when generating the prediction.
- Note that you should **NOT** hard code any path in your file or script.
- Your testing code have to be finished in **10 mins.**

	A	B	C
1	id	filename	label
2	0	000000.jpg	None
3	1	000001.jpg	None
4	2	000002.jpg	None

Example of test.csv

	A	B	C
1	id	filename	label
2	0	000000.jpg	Fork
3	1	000001.jpg	Fork
4	2	000002.jpg	Fork

Example of test_pred.csv

Bash Script (cont'd)

- You must **not** use commands such as **rm**, **sudo**, **CUDA_VISIBLE_DEVICES**, **cp**, **mv**, **mkdir**, **cd**, **pip** or other commands to change the Linux environment.
- In your submitted script, please use the command **python3** to execute your testing python files.
  - For example: python3 test.py $1 $2
- We will execute you code on **Linux** system, so try to make sure you code can be executed on Linux system before submitting your homework.

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 hw4_download.sh.
  - TAs will run `bash hw4_download.sh` prior to any inference if the download script exists, i.e. it is **NOT** necessary to create a blank `hw4_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.

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.

Packages (This is for problem 1 using DVGO)

- imageio==2.22.4
- numpy==1.23.3
- Pillow==9.2.0
- scipy==1.9.1
- opencv-python==4.6.0.66
- torch==1.12.1
- torchvision==0.13.1
- tqdm, gdown, glob, yaml
- Any dependencies of above packages, and other standard python packages
- **Basically, run:**

pip install -r requirements.txt --no-cache-dir && pip install torch-scatter -f https://data.pyg.org/whl/torch-1.12.0+cu102.html --no-cache-dir

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

Packages and Reminders

- If you use other github repository other than DVGO or BYOL, please give us your environment's requirement.txt and describe how we can reproduce your environment in your report.
- Python==3.8
- 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.
- If you train on GPU ids other than 0, remember to deal with the "**map location**" issue when you load model. (More details about this issue, please refer to https://github.com/pytorch/pytorch/issues/15541)

Deadline and Academic Honesty

- Deadline: **111/12/19 (Mon.) 11:59 PM (GMT+8)**
- Late policy : Up to 3 free late days in a semester. After that, late homework will be deducted 30% each day.
- Taking any unfair advantages over other class members (or letting anyone do so) is strictly prohibited. Violating university policy would result in F for this course.
- Students are encouraged to discuss the homework assignments, but you must complete the assignment by yourself. TA will compare the similarity of everyone's homework. Any form of cheating or plagiarism will not be tolerated, which will also result in F for students with such misconduct.

Penalty

- If we cannot execute your code, TAs will give you a chance to make minor modifications to your code. After you modify your code,
  - If we can execute your code, you will still receive a 30% penalty in your model performance score.
  - If we still cannot execute your code, no point will be given.

Reminder

- Please start working on this homework as early as possible.
- The training may take hours on a GPU or days on CPUs.
- Please read and follow the HW rules carefully.
- If not sure, please ask your TAs!

How to Find Help

- Google!
- Use TA hours (please check course website for time/location).
- Post your question under HW4 discussion section on NTU COOL.
- Contact TAs by e-mail: ntudlcv@gmail.com.

DOs and DONTs 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 make the TA hours, please email the TAs to schedule an appointment instead of stopping by the lab directly.

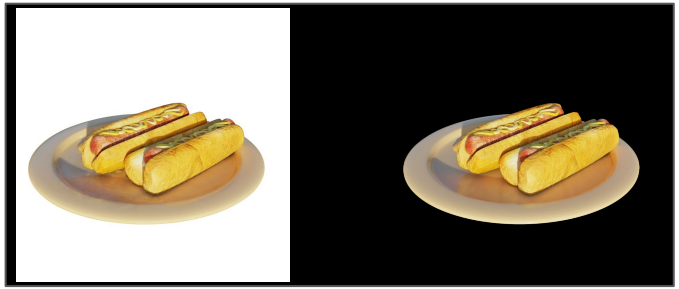
Outline

- Problems & Grading
- Dataset
- Submission & Rules
- Supplementary

Supplementary - grade.py (for Problem 1)

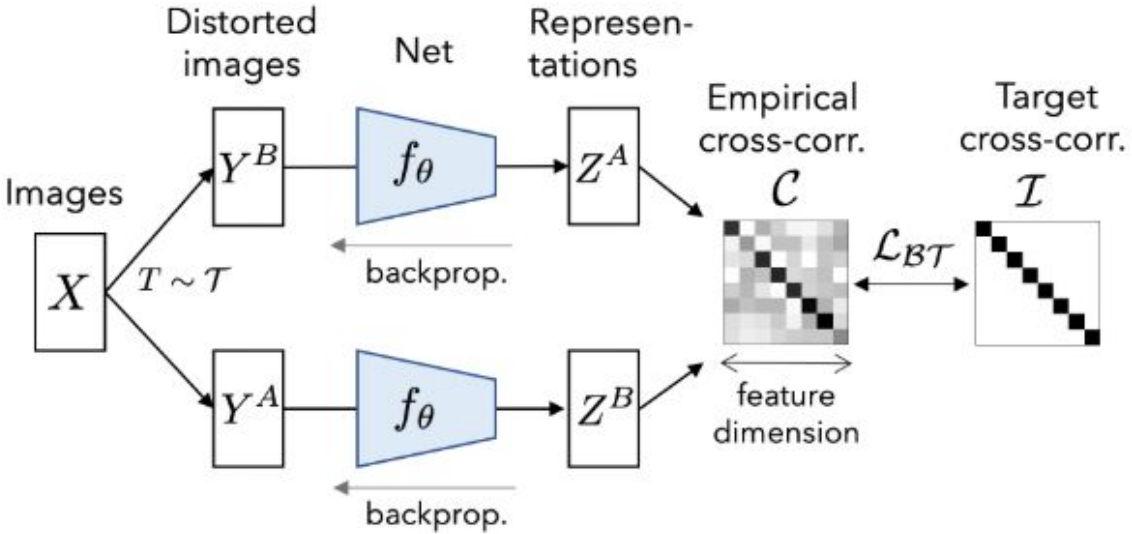
- This file is on github and it is the one we will evaluate your score after running your hw4_1.sh.
  - python grade.py $1 $2
    - $1 is the path to the folder of generated image (white background)
    - $2 is the path to the folder of gt image
  - you can also use this file to evaluate on eval dataset

```
(DVGO) timothy@vll-timothy:~/Desktop/DirectVoxGO$ bash grade.sh
100%|                                                                                | 50/50 [00:53<00:00,  1.06s/it]
Testing psnr 35.379612398147586 (avg)
Testing ssim 0.9752658490162989 (avg)
```



Supplementary - Barlow Twins (for Problem 2)

- Construct the cross-correlation matrix, and try to make this matrix close to the identity.
  - Cause the embedding vectors to be similar, while minimizing the redundancy.
  - Avoid **mode collapsed**



Barlow Twins: Self-Supervised Learning via Redundancy Reduction, ICML'21