

# Homework #4

Deep Learning for Computer Vision

NTU, Fall 2020

109/12/8

109/12/29 (Tue.) 02:00 AM due

# Outline

- Problems & Grading
- Model Evaluation
- Provided Script & Training Tips
- Dataset
- Homework policy

# Outline

- Problems & Grading
  - Model Evaluation
  - Provided Script & Training Tips
  - Dataset
  - Homework policy

# Problem Description - Few-Shot Classification

- Problem 1: Prototypical Network (50%) [Mini-ImageNet]
- Problem 2: Data Hallucination for Few-shot Learning (50%) [Mini-ImageNet]
- Problem 3: Improved Data Hallucination Model (20%) [Mini-ImageNet]

\* Please refer to “Dataset” section for more details about Mini-ImageNet dataset.

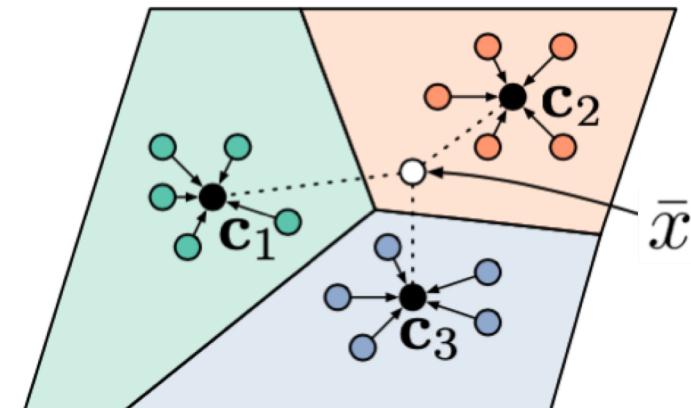
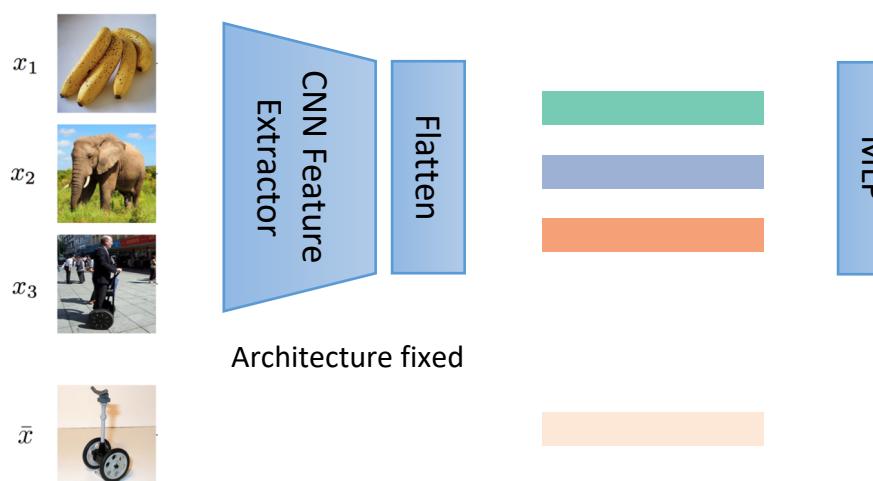
# Problem 1: Prototypical Network (50%)

In this problem, you will have to implement the prototypical network. Prototypical Networks learn a metric space in which classification can be performed by computing distances to prototype representations of each class.

- Grading:
  - Report (35%)
  - Model Performance (15%)

## Hint

- (1) For the model architecture, the CNN Feature Extractor is provided in P.26. You only need to design the architecture of the MLP.
- (2) For the objective function, you can only use the cross entropy loss.



# Problem 1: Prototypical Network (50%) (cont'd)

## Report (35%)

In Problem 1-1, you will have to implement the prototypical network as your baseline model to perform 5-way 1-shot classification.

1. (11%) Describe the architecture & implementation details of your model. (**Include** but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot setting for meta-train and meta-test phase)

**Please report the accuracy on validation set under 5-way 1-shot setting (during inference).**

- The accuracy should be the same as your model performance in P.8.
- TAs will run your code to verify the performance.

### Hint

The N-way K-shot setting in the meta-train phase may be different from the one in the meta-test phase. Your model may be trained better in larger ways (e.g., 10-way 1-shot setting in the meta-train phase and 5-way 1-shot setting in the meta-test phase).

# Problem 1: Prototypical Network (50%) (cont'd)

## Report (35%)

For Problem 1-2 and 1-3, you will do some experiments about different distance function and different K shot settings.

2. (12%) When meta-train and meta-test under the same 5-way 1-shot setting, please report and discuss the accuracy of the prototypical network using 3 different distance function (*i.e., Euclidean distance, cosine similarity and parametric function*). You should also describe how you design your parametric function.
  - Parametric: use a learnable model to measure distance between two features
3. (12%) When meta-train and meta-test under the same 5-way K-shot setting, please report and compare the accuracy with different shots. (K=1, 5, 10)

**[Hint] In order to reduce the training time, you may use a model simpler than that in Problem 1-1 for these two subproblems. For example, you can train the model with less training epochs.**

# Problem 1: Prototypical Network (50%) (cont'd)

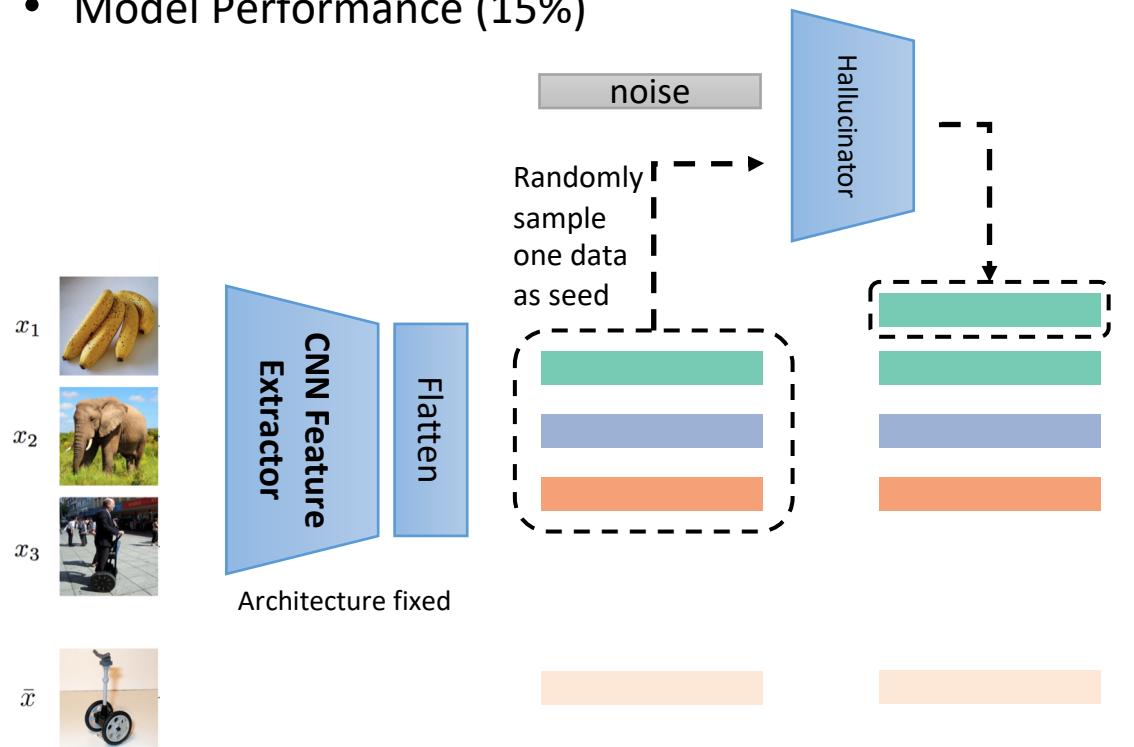
## Model Performance (15%)

- The mean accuracy of your model under **5-way 1-shot** setting (during inference) (Problem 1-1) should be above the baseline score.
  - On the validation set (10%): **0.44 [should be reported in Problem 1-1]**
  - On the test set (5%): **0.42**
- You **have to** use the test case provided by the TAs to calculate your mean accuracy on the validation set.
- TAs will execute your code to check if you pass the baseline.
- Only TAs have the test data.

# Problem 2: Data Hallucination for Few-shot Learning (50%)

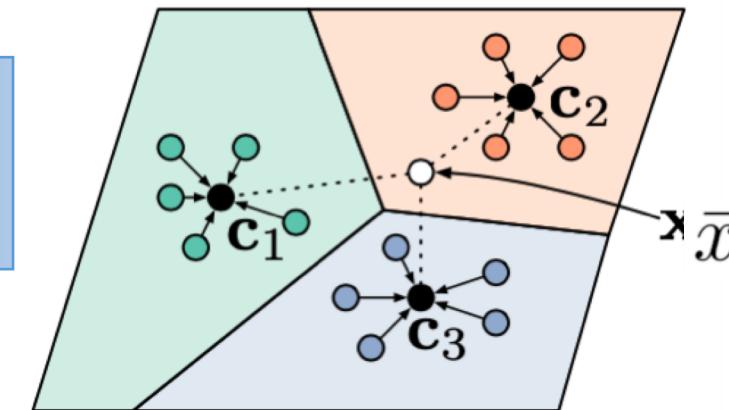
In this problem, you will have to improve the prototypical network by data hallucination. Your hallucinator will take a seed example  $x$  and a noise vector  $z$  as input, and produce a hallucinated example as output.

- Grading:
  - Report (35%)
  - Model Performance (15%)



## Hint

- (1) For the model architecture, the CNN Feature Extractor is provided in P.26. You only need to design the architecture of the MLP and Hallucinator.
- (2) For the objective function, you can only use the cross entropy loss.
- (3) When calculating the prototypes of each class, please average all the real data and hallucinated data. (Do not use weighted mean)



# Problem 2: Data Hallucination for Few-shot Learning (50%) (cont'd)

## Report (35%)

In Problem 2-1 and 2-2, you will have to improve the prototypical network by data hallucination to perform 5-way 1-shot classification and analyze the quality of your hallucinated data by t-SNE.

1. (10%) Describe the architecture & implementation details of your model. (**Include** but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot M-augmentation setting for meta-train and meta-test phase)

**Please report the accuracy on validation set under 5-way 1-shot setting (during inference).**

- You should select  **$1 \leq M \leq 200$** .
- The accuracy should be the same as your model performance in P.13.
- TAs will run your code to verify the performance.

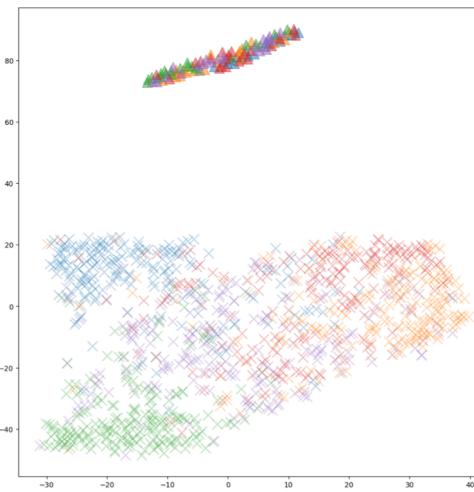
### Hint

M-augmentation: each class hallucinate M more data.

# Problem 2: Data Hallucination for Few-shot Learning (50%) (cont'd)

## Report (35%)

2. (10%) To analyze the quality of your hallucinated data, please visualize the real and hallucinated (training) data in the latent space (the space where you calculate the prototypes of each class, e.g., the output of the MLP layer) by mapping the features to 2D space (with t-SNE). Briefly explain your result of t-SNE visualization. Example for fig2-3.png is shown below.
  - In the t-SNE visualization, you should have 5 classes with different colors, and each class has at least 200 real data and 20 hallucinated data. Real data are shown as crosses and hallucinated data are shown as triangles.



Example for fig2-2.png

# Problem 2: Data Hallucination for Few-shot Learning (50%) (cont'd)

## Report (35%)

For Problem 2-3, you will do some experiments on different M-augmentation settings.

3. (10%) When meta-train and meta-test under the same 5-way 1-shot **M-augmentation** setting, please report and compare the accuracy with different number of hallucinated data. ( $M=10, 50, 100$ )
4. (5%) Discuss what you've observed and learned from implementing the data hallucination model.

[Hint] In order to reduce the training time, you may use a model simpler than that in Problem 2-1 for Problem 2-3. For example, you can train the model with less training epochs.

# Problem 2: Data Hallucination for Few-shot Learning (50%) (cont'd)

## Model Performance (15%)

- Mean accuracy under **5-way 1-shot** setting (Problem 2-1) should be above the baseline score to get points
  - On the validation set (10%): **0.46 [should be reported in Problem 2-1]**
  - On the test set (5%): **0.44**
- You **have to** use the test case provided by the TAs to calculate your mean accuracy on the validation set.
- TAs will execute your code to check if you pass the baseline.
- Only TAs have the test data.

# Problem 3: Improved Data Hallucination Model (20%)

In this problem, you need to implement an **improved** data hallucination model to perform few-shot classification. The performance of this model should be better than your model in Problem 2.

- Grading:
  - Report (15%)
  - Model Performance (5%)

For example:

Design **additional loss** as guidance for the hallucinator (e.g., adversarial loss).

**Note that only modifying training configs and MLP architecture is not allowed.**

You may select any one of the papers below, or search for other related papers, or design a method by yourself to implement your improved model.

## Related Few-Shot Learning papers

- [Adversarial Feature Hallucination Networks for Few-Shot Learning](#). CVPR 2020
- [Diversity Transfer Network for Few-Shot Learning](#). AAAI 2020
- [Image Deformation Meta-Networks for One-Shot Learning](#). CVPR 2019
- [Multi-level Semantic Feature Augmentation for One-shot Learning](#). TIP 2019

# Problem 3: Improved Data Hallucination Model (20%) (cont'd)

## Report (15%)

In Problem 3-1~3-3, you need to implement an **improved** data hallucination model to perform 5-way 1-shot classification and analyze the quality of your hallucinated data by t-SNE.

1. (9%) Describe the architecture & implementation details of your model. (**Include** but not limited to the number of training episodes, distance function, learning rate schedule, data augmentation, optimizer, and N-way K-shot M-augmentation setting for meta-train and meta-test phase)

**Please report the accuracy on validation set under 5-way 1-shot setting (during inference).**

- The accuracy should be the same as your model performance in P.17.
- TAs will run your code to verify the performance (only valid set).

# Problem 3: Improved Data Hallucination Model (20%) (cont'd)

## Report (15%)

2. (2%) To analyze the quality of your hallucinated data, please visualize the real and hallucinated (training) data in the latent space (the space where you calculate the prototypes of each class, e.g., the output of the MLP layer) by mapping the features to 2D space (with t-SNE).
  - In the t-SNE visualization, you should have 5 classes with different colors, and each class has at least 200 real data and 20 hallucinated data. Real data are shown as crosses and hallucinated data are shown as triangles.
3. (4%) Try to explain why the improved model performs better than the one in Problem 2 (e.g., discuss the difference between your visualization of latent space in Problem 2 and Problem 3).

# Problem 3: Improved Data Hallucination Model (20%) (cont'd)

## Model Performance (5%)

- Mean accuracy under **5-way 1-shot** setting (Problem 3-1) should be above your accuracy in Problem 2-1 to get points
  - On the validation set (5%) **[should be reported in Problem 3-1]**
  - ~~On the test set~~
- You **have to** use the test case provided by the TAs to calculate your mean accuracy on the validation set.
- TAs will execute your code to check if your accuracy **outperforms** the one in Problem 2-1.
- Only TAs have the test data.

# Reminder

- Please start working on this homework earlier. The training may take a few hours on a GPU or days on CPUs.
- Please follow the homework rules shown in “Homework policy” section.

# Outline

- Problems & Grading
- Model Evaluation
- Provided Script & Training Tips
- Dataset
- Homework policy

# Model Evaluation

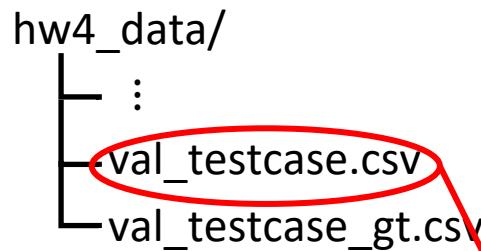
- You should report the 95% confidence interval of the accuracy over 600 episodes. In each episode, the accuracy is calculated over totally N \* 15 query data where each way has 15 query data (N-way K-shot setting).

$$\bar{x} \pm 1.96 \times \frac{\sigma}{\sqrt{600}}$$

( $\bar{x}$  is the mean,  $\sigma$  is the standard deviation)

# Model Evaluation – Test Case

- For the model performance part, we offer the validation test case which contains 600 episodes (5 way 1 shot with 75 query data) randomly sampled from the validation set. You **have to** report your **mean accuracy** on the validation set on this test case. TAs will test your model in the test set with the test case file **in the same format**. (The classes in train, val, and test set are **disjoint**.)
- For other parts of your report, you can calculate the accuracy by randomly sampled 600 episodes on your own rather than using this test case.



# validation test case file  
# ground truth for validation test case

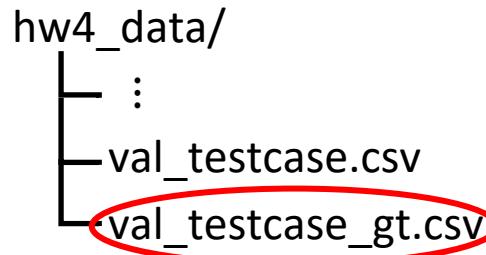
This file is for model evaluation with 600 episodes under 5-way 1-shot setting. Each row corresponds to each episode. In each episode, there are 5 support data (total 5 classes, each class contains 1 support data) and 75 query data. Each data is represented by its id that appears in val.csv .

Header in first row: <episode\_id>, <class 0 support data>, ..., <class 4 support data>, <query data 0>, ..., <query data 74>

```
episode_id,class0_support0,class1_support0,class2_support0,class3_support0,class4_support0,query0,query1,query2,query3,query4,query5,  
0,4520,6599,2606,320,3165,6209,6096,374,4556,4434,3225,2688,42,4758,2970,2592,77,367,2913,3084,6136,2817,3009,6588,4488,3378,2999,438  
1,4348,6684,1892,5488,44,1800,5478,6721,182,2164,1888,4528,5624,5926,5720,111,4592,5620,7008,521,7095,1827,4255,2178,6879,6696,43,596  
2,510,6888,4957,2133,3275,5307,1823,451,398,3375,7007,407,2264,5298,354,5191,190,434,6777,1901,2018,3243,69,2321,6647,4878,5051,3493,  
3,5189,708,1956,1454,9484,9425,9155,1064,9171,1543,984,1498,1468,9284,5254,1265,1127,5209,5139,1852,2129,746,5091,5316,1432,5337,2317
```

# Model Evaluation – Test Case

- For the model performance part, we offer the validation test case which contains 600 episodes (5 way 1 shot with 75 query data) randomly sampled from the validation set. You should report your **mean accuracy** on the validation set on this test case. TAs will test your model in the test set with the test case file in the same format. (The classes in train, val, and test set are **disjoint**.)
- For other parts of your report, you can calculate the accuracy by randomly sampled 600 episodes on your own rather than using this test case.



# validation test case file  
# ground truth for validation test case

This file contains the ground truth label for query data in `val_testcase.csv`. The label ranges from 0 to 4 because the test case is under the 5-way 1-shot setting.

Header in first row: <episode\_id>, <label for query data 0>, ..., <label for query data 74>

```
episode_id,query0,query1,query2,query3,query4,query5,query6,query7,query8,query9,query10,query11,query12,query13,query14,query15
0,1,1,3,0,0,4,2,3,0,2,2,3,3,2,4,1,2,4,1,0,4,2,3,1,3,2,3,3,1,4,3,1,4,4,1,2,2,3,1,0,4,0,4,0,4,1,3,4,3,4,1,2,0,3,1,0,0,1,2,4,2,1,0,
1,2,3,1,4,2,2,0,3,3,3,4,0,3,1,4,1,2,0,2,1,1,4,3,2,2,1,2,0,4,3,4,2,2,0,0,4,1,1,1,1,1,4,0,3,0,0,4,4,3,4,1,3,0,4,2,2,0,4,4,3,2,1,2,0,
2,2,3,0,0,4,1,0,3,2,0,2,0,0,1,3,3,4,0,3,1,2,2,4,2,0,4,4,4,0,1,1,0,2,1,3,2,2,1,4,3,4,3,1,3,3,3,1,1,0,2,1,0,4,0,1,1,4,3,4,2,4,2,1,
3,4,4,1,4,3,1,3,3,4,0,3,1,0,0,2,2,1,0,0,3,0,2,0,0,2,3,4,1,0,3,2,4,2,4,3,1,0,0,3,4,0,1,1,1,2,3,1,1,4,3,0,2,2,1,2,3,2,3,4,3,0,2,2,
```

# Outline

- Problems & Grading
- Model Evaluation
- **Provided Script & Training Tips**
- Dataset
- Homework policy

# Provided Script

- Template of testing code:
  - We provide a partially completed testing code, with a dataloader (reads the test case file and generates one task/episode in each batch) finished, and some “TODOs” unfinished.
  - After you finish all the TODOs in `test_testcase.py`, you can output your prediction result to a csv file.
  - Usage:

```
python3 test_testcase.py [--N-way N_WAY] [--N-shot N_SHOT]
                         [--N-query N_QUERY] [--load LOAD] [--test_csv TEST_CSV]
                         [--test_data_dir TEST_DATA_DIR] [--testcase_csv TESTCASE_CSV]
                         [--output_csv OUTPUT_CSV]
```

```
optional arguments:
  -h, --help            show this help message and exit
  --N-way N_WAY          N_way (default: 5)
  --N-shot N_SHOT        N_shot (default: 1)
  --N-query N_QUERY      N_query (default: 15)
  --load LOAD            Model checkpoint path
  --test_csv TEST_CSV    Testing images csv file
  --test_data_dir TEST_DATA_DIR
                        Testing images directory
  --testcase_csv TESTCASE_CSV
                        Test case csv
  --output_csv OUTPUT_CSV
                        Output filename
```

[Optional] If you have no idea about building the dataloader for training, you may also refer to `test_testcase.py`.

# Provided Script

- Evaluation:
  - We provide the code to evaluate your model performance.
  - We use the output **mean** value to check if you pass the baseline.
  - Usage:

```
python3 eval.py <path_to_the_predict_csv> <path_to_the_ground_truth_csv>
```

# CNN Feature Extractor – Conv-4

- **[IMPORTANT]** For simplicity, you can **only** use Conv-4 as a feature extractor. Your Conv-4 should follow the following definition.
- The definition of Conv-4: a model that is composed of four convolutional blocks. Each block comprises a 64-filter  $3 \times 3$  convolution, batch normalization layer, a ReLU nonlinearity and a  $2 \times 2$  max-pooling layer.
- Note: For an  $84 \times 84$  RGB image, the output space is 1600.
- You **cannot** use pre-trained weight, but you can initialize the model weights with Gaussian noise.

```
class Convnet(nn.Module):
    def __init__(self, in_channels=3, hid_channels=64, out_channels=64):
        super().__init__()
        self.encoder = nn.Sequential(
            conv_block(in_channels, hid_channels),
            conv_block(hid_channels, hid_channels),
            conv_block(hid_channels, hid_channels),
            conv_block(hid_channels, out_channels),
        )

    def forward(self, x):
        x = self.encoder(x)
        return x.view(x.size(0), -1)

def conv_block(in_channels, out_channels):
    bn = nn.BatchNorm2d(out_channels)
    return nn.Sequential(
        nn.Conv2d(in_channels, out_channels, 3, padding=1),
        bn,
        nn.ReLU(),
        nn.MaxPool2d(2)
    )
```

We provide the sample code of Conv-4 network for you :)

# Training Tips

- There are many ways to sample a task, including using a custom sampler in dataloader, or sampling one task inside `__getitem__()` method in your custom dataset class, etc.
- **If you still don't know how to write a few shot learning code, you may find `test_testcase.py` helpful.**
- If you use any random function in `__getitem__()` method with `num_workers>0` in your dataloader, you should set different seeds for different workers by setting `worker_init_fn` argument in dataloader. Otherwise, all the workers will generate the same sampling sequence.
- **[IMPORTANT] For each training epoch, you should sample different tasks/episodes. Otherwise, the performance of your model might be unpleasant.**

# Outline

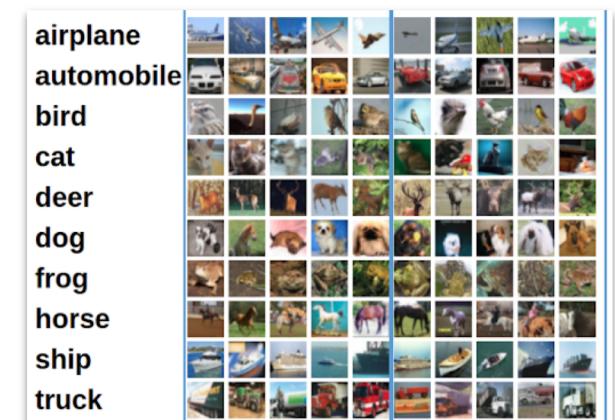
- Problems & Grading
- Model Evaluation
- Provided Script & Training Tips
- **Dataset**
- Homework policy

# Mini-ImageNet Dataset

- The dataset consists of 48,000 84x84 RGB images in **80** classes.
- In the dataset, you will get

```
hw4_data/
  └── train/
  └── train.csv                                # training images directory (64 class, each class has 600 images)
  └── val/
  └── val.csv                                  # validation images directory (16 class, each class has 600 images)
  └── val_testcase.csv                         # validation image csv file
  └── val_testcase_gt.csv                      # validation test case file
                                              # ground truth for validation test case
```

- You **CANNOT** use validation data for training purposes.
- The classes in train and val set are **disjoint**.
- All the testing set files are in the same format as the validation set files.  
To be more specific, there are ***test/***, ***test.csv*** and ***test\_testcase.csv***.



# Mini-ImageNet Dataset

- The dataset consists of 48,000 84x84 RGB images in **80** classes.
- In the dataset, you will get

```
hw4_data/
  └── train/
    └── train.csv
  └── val/
    └── val.csv
  └── val_testcase.csv
  └── val_testcase_gt.csv
```

```
# training images directory (64 class, each class has 600 images)
# training image csv file
# validation images directory (16 class, each class has 600 images)
# validation image csv file
# validation test case file
# ground truth for validation test case
```

Header in first row: <image\_id>, <filename>, <label>

```
id,filename,label
0,n0153282900000005.jpg,n01532829
1,n0153282900000006.jpg,n01532829
2,n0153282900000007.jpg,n01532829
3,n0153282900000010.jpg,n01532829
```

# Outline

- Problems & Grading
- Model Evaluation
- Provided Script & Training Tips
- Dataset
- Homework policy

# Deadline and Academic Honesty

- Report and source code deadline: **109/12/29 (Tue.) 02:00 AM (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.

# Deadline and Academic Honesty

- Searching for online materials or discussing with fellow classmates are highly encouraged. However, you must provide the code or solution by yourself.
- Please specify, if any, the references for any parts of your HW solution in your report (e.g., the name and student ID of your collaborators and/or the Internet URL you consult with). If you complete the assignment all by yourself, you must also specify “no collaborators”.

# Submission

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

<https://classroom.github.com/a/5rgIMjMw>

- 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

- In your GitHub repository, you should make sure that your submission includes at least the following files in the **root directory**
  - hw4\_<studentID>.pdf
  - **hw4\_download.sh**
    - Download **all** the models needed for Problem 1~3. We will execute this script first.
  - hw4\_1.sh
  - hw4\_2.sh
  - hw4\_3.sh
  - your python files (e.g., training code & testing code)
  - your model files (can be loaded by your python file)
- **Don't upload the dataset.**
- If any of the file format is wrong, you will get zero point.

# Trained Model

- If your model is larger than GitHub's maximum capacity (100MB), you can upload your model to another cloud service (e.g., Dropbox). However, your script file should be able to download the model **automatically**.
  - (Dropbox tutorial: <https://drive.google.com/file/d/1XOz69Mgxo67IZNQWnRSjT2eZAtpUAgZ/view> )
- Do not delete your trained model before the TAs disclose your homework score and before you make sure that your score is correct.
- Use the **wget** command in your script to download your model files. Do not use the curl command.
- Note that you **should NOT hard code any path** in your file or script except for the path of your trained model.

# Bash Script

- TA will run your code as shown below:
  - `bash hw4_download.sh`
  - `bash hw4_1.sh $1 $2 $3 $4`
  - `bash hw4_2.sh $1 $2 $3 $4`
    - \$1: testing images csv file (e.g., *hw4\_data/val.csv*)
    - \$2: testing images directory (e.g., *hw4\_data/val*)
    - \$3: path of test case on test set (e.g., *hw4\_data/val\_testcase.csv*)
    - \$4: path of output csv file (predicted labels) (e.g., *output/val\_pred.csv*)
- The output csv file **MUST** have the same format as ‘*val\_testcase\_gt.csv*’
- Note that you should **NOT** hard code any path in your file or script
- Your testing code have to be finished in **10 mins.**

# Bash Script

- TA will run your code as shown below:
  - `bash hw4_3.sh $1 $2 $3 $4 $5 $6`
    - \$1: testing images csv file (e.g., *hw4\_data/val.csv*)
    - \$2: testing images directory (e.g., *hw4\_data/val*)
    - \$3: path of test case on test set (e.g., *hw4\_data/val\_testcase.csv*)
    - \$4: path of output csv file (predicted labels) (e.g., *output/val\_pred.csv*)
    - \$5: training images csv file (e.g., *hw4\_data/train.csv*)
    - \$6: training images directory (e.g., *hw4\_data/train*)
- You may or may not use training images (\$5, \$6), depending on your hallucination method.  
**However, your bash script must include them.**
- The output csv file **MUST** have the same format as ‘*val\_testcase\_gt.csv*’.
- Note that you should **NOT** hard code any path in your file or script.
- Your testing code have to be finished in **10 mins.**

# Bash Script

- 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
- 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.

# Packages

- python: 3.6
- numpy: 1.18.1
- **torch >= 1.4.0**
- **torchvision >= 0.5.0**
- cudatoolkit: 10.1
- scikit-learn: 0.21.3
- pandas: 1.1.3
- and other standard python packages
- **E-mail or ask TA first if you want to import other packages.**
- matplotlib: 3.3.2
- pillow: 8.0.0
- imageio: 2.9.0
- scipy: 1.2.1
- tqdm: 4.54.0

# 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.**
- If you train on GPU ids other than 0, remember to deal with the “map location” issue when you load model.

# Penalty

- If we can not reproduce your accuracy on the validation set or you use the feature extractor other than Conv-4, you will get 0 points in model performance.
- If we can not execute your code, we will give you only **one** chance to make minor modifications to your code. After you modify your code,
  - If we can execute your code and reproduce your results on the validation set, you will still receive a 30% penalty in your model performance score.
  - If we still cannot execute your code or **cannot reproduce your accuracy on the validation set**, you will still get 0 in this problem.

# How to find help

- Google !
- Use TA hours (please check [course website](#) for time/location)
- Post your question under hw4 FAQ section in FB group
- 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.