

# Assignment 2:

# Recognition/Classification

Computer Vision  
NTU, Spring 2025

Announced: 2025/03/21

**Deadline: 2025/04/11**

# Outline

Part 1: Bag-of-Words Scene Recognition

Part 2: CNN Image Classification

# Before you start...

- Environment setup (see README.md)
  - **Python 3.8**
  - **Python 3.6**
- Download the dataset
  - hw2\_data.zip

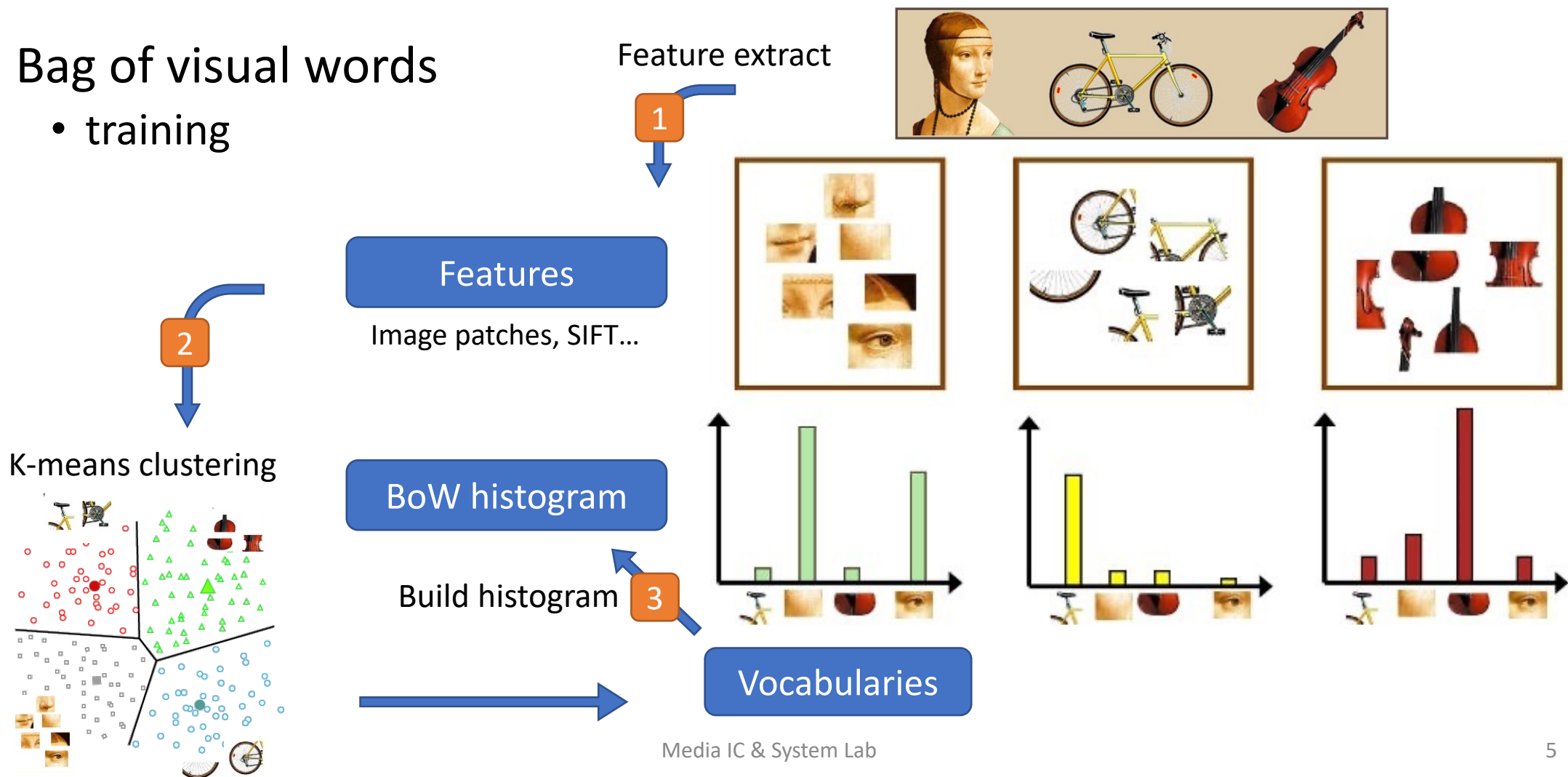
```
./hw2
├── hw2_data
├── p1
├── p2
├── README.md
├── report_template.docx
├── requirements_py36.txt
└── requirements_py38.txt
```

# Part 1 :

## Bag-of-Words Scene Recognition

# BoW Scene Recognition

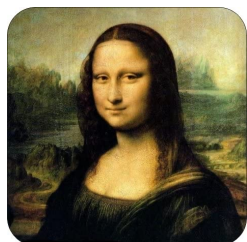
- Bag of visual words
  - training



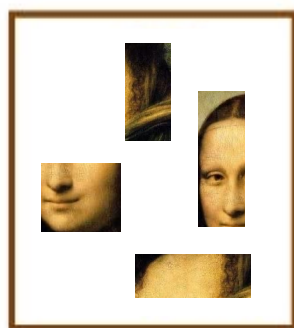
# BoW Scene Recognition

- Bag of visual words
  - testing

Test image



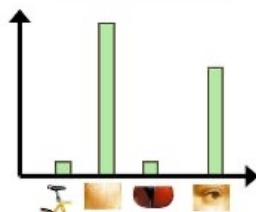
1  
Feature extract



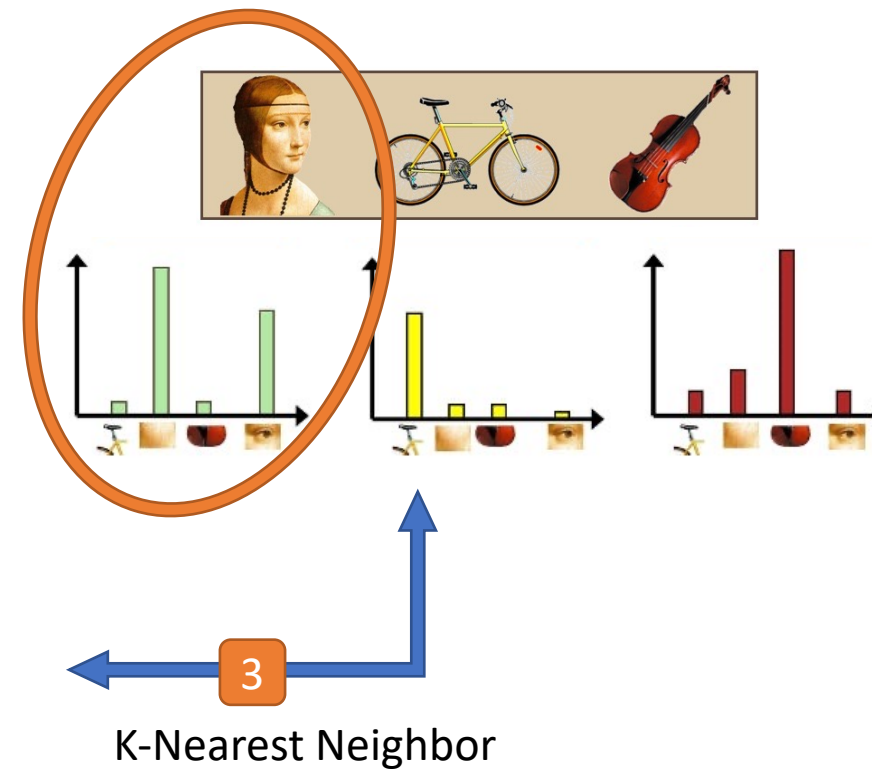
Features

Image patches, SIFT...

2  
Build histogram



BoW histogram



# BoW Scene Recognition

- You will have to implement two kinds of feature representations
  1. Tiny image
    - `get_tiny_images()`
  2. Bag of SIFT (`cv2feat.sift` is allowed)
    - `build_vocabulary()`
    - `get_bags_of_sifts()`
- Then apply hand-craft classifier
  1. K-Nearest Neighbor (`sklearn.neighbors.KNeighborsClassifier` is **NOT** allowed)
    - `nearest_neighbor_classify()`

# Dataset

- hw2\_data/p1\_data/train
  - 100 grayscale images per category
- hw2\_data/p1\_data/test
  - 100+ grayscale images per category



```
./hw2_data/p1_data
├── test
│   ├── Bedroom
│   ├── Coast
│   ├── Forest
│   ├── Highway
│   ├── Industrial
│   ├── InsideCity
│   ├── Kitchen
│   ├── LivingRoom
│   ├── Mountain
│   ├── Office
│   ├── OpenCountry
│   ├── Store
│   ├── Street
│   ├── Suburb
│   └── TallBuilding
└── train
    ├── Bedroom
    ├── Coast
    ├── Forest
    ├── Highway
    ├── Industrial
    ├── InsideCity
    ├── Kitchen
    ├── LivingRoom
    ├── Mountain
    ├── Office
    ├── OpenCountry
    ├── Store
    ├── Street
    ├── Suburb
    └── TallBuilding
```



# Assignment Description

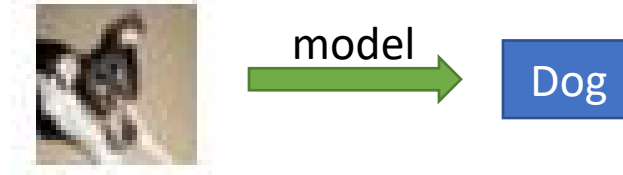
- p1/p1.py
  - Read data, feature extract, classification, compute accuracy, plot confusion matrix.
  - TA will run this code to evaluate your result.
- p1/utils.py
  - get\_tiny\_images() **## TODO ##**
    - Build tiny image features.
  - build\_vocabulary() **## TODO ##**
    - Sample SIFT descriptors from training images, cluster them with kmeans and return centroid.
  - get\_bags\_of\_sifts() **## TODO ##**
    - Construct SIFT and build a histogram indicating how many times each centroid was used.
  - nearest\_neighbor\_classify() **## TODO ##**
    - Predict the category for each test image.
    - **CAN NOT USE sklearn.neighbors.KNeighborsClassifier**
- p1/p1\_run.sh
  - example for code execution.

# Part 2 :

## CNN Image Classification

# CNN Image Classification

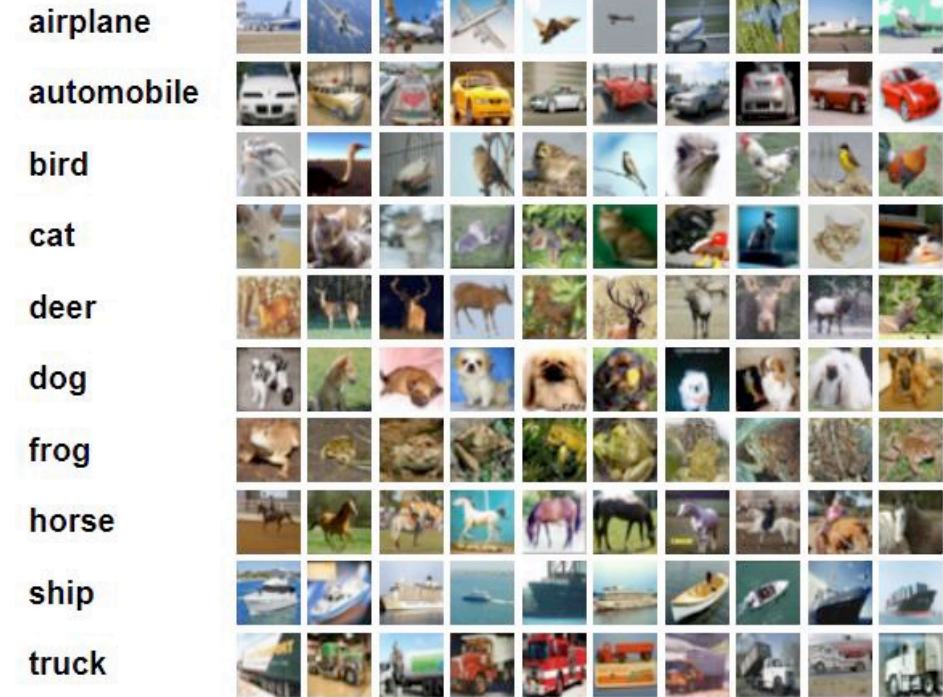
- Image classification – predict a label for each image
  - Input : RGB image
  - Output : classification label



- You need to perform image classification with the following methods:
  - A. Build and train a CNN model from scratch
    - torchvision.models, pretrained weights is **NOT** allowed.
  - B. Try ResNet18 model
    - You can use torchvision.models.resnet18(weights="<pretrained-model>").
    - You may have to modify the structure to pass the strong baseline.

# Dataset

- Original CIFAR-10 dataset (32x32 RGB images)
  - 50000 training images, 5000 each class
  - 10000 test images
- Provided dataset (**You can only use this**)
  - p2\_data/train
    - 20000 images, 1 annotation file (w/ labels)
  - p2\_data/val
    - 5000 images, 1 annotation file (w/ labels)
    - You **cannot** use the validation labels to train your model in a fully supervised manner.
  - p2\_data/unlabel
    - 30000 images, 1 annotation file (**w/o labels**)
    - For semi-supervised (optional)



```
./hw2_data/p2_data
├── train
├── unlabel
└── val
```

# Assignment Description

- p2/p2\_train.py **## TODO ##**
  - Start training, write log files, plot learning curves, etc.
- p2/p2\_inference.py **## TODO ##**
  - Inferencing, generate predicted output to csv file.
  - TA will run this code to generate your result.
- p2/p2\_eval.py **## DO NOT MODIFY ##**
  - Evaluate accuracy between predicted and ground truth labels.
  - TA will run this code to evaluate your result.

```
def plot_learning_curve(logfile_dir, result_lists):  
    #####  
    # TODO:  
    # Plot and save the learning curves under logfile_dir, you can  
    # use plt.plot() and plt.savefig().  
    #####
```

```
##### VALIDATION #####  
model.eval()  
with torch.no_grad():  
    val_start_time = time.time()  
    val_loss = 0.0  
    val_correct = 0.0  
    #####  
    # TODO:  
    # Finish forward part in validation, you can refer to the  
    # training part.  
    #####
```

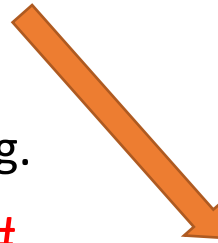
```
##### INFERENCE #####  
predictions = []  
model.eval()  
with torch.no_grad():  
    test_start_time = time.time()  
    #####  
    # TODO:  
    # Finish forward part in inference process, similar to  
    # validation, and append predicted label to 'predictions'  
    # list.  
    #####
```

```
1  filename,label  
2  10000.jpg,3  
3  10001.jpg,1  
4  10002.jpg,8  
5  10003.jpg,0  
6  10004.jpg,6
```

output csv file format

# Assignment Description

- p2/model.py **## TODO ##**
  - Define your own models.
- p2/dataset.py **## TODO ##**
  - Define your customized dataset.
- p2/config.py
  - Hyperparameters setting for training.
- p2/utils.py **## DO NOT MODIFY ##**
  - Some helper functions.
- p2/p2\_run\_train.sh, p2\_run\_test.sh
  - example for code execution.



```
class MyNet(nn.Module):
    def __init__(self):
        super(MyNet, self).__init__()

        #####
        # TODO:
        # Define your CNN model architecture. Note that the first
        # input channel is 3, and the output dimension is 10 (class).
        #####

        pass

    def forward(self, x):

class ResNet18(nn.Module):
    def __init__(self):
        super(ResNet18, self).__init__()

        #####
        # NOTE:
        # Pretrain weights on ResNet18 is allowed.
        #####

        # (batch_size, 3, 32, 32)
        self.resnet = models.resnet18(pretrained=True)
        # (batch_size, 512)
        self.resnet.fc = nn.Linear(self.resnet.fc.in_features, 10)
        # (batch_size, 10)
```

```
transform = transforms.Compose([
    transforms.Resize((32,32)),
    ##### TODO: Data Augmentation Begin #####

    ##### TODO: Data Augmentation End #####
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])

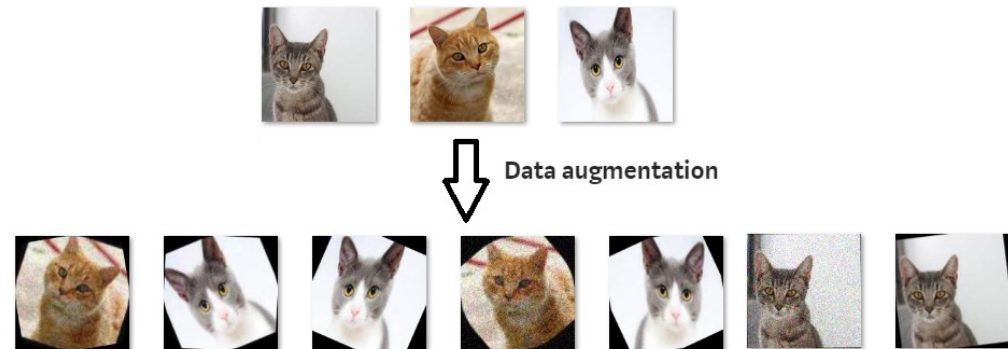
class CIFAR10Dataset(Dataset):

    def __getitem__(self, index):

        #####
        # TODO:
        #####
```

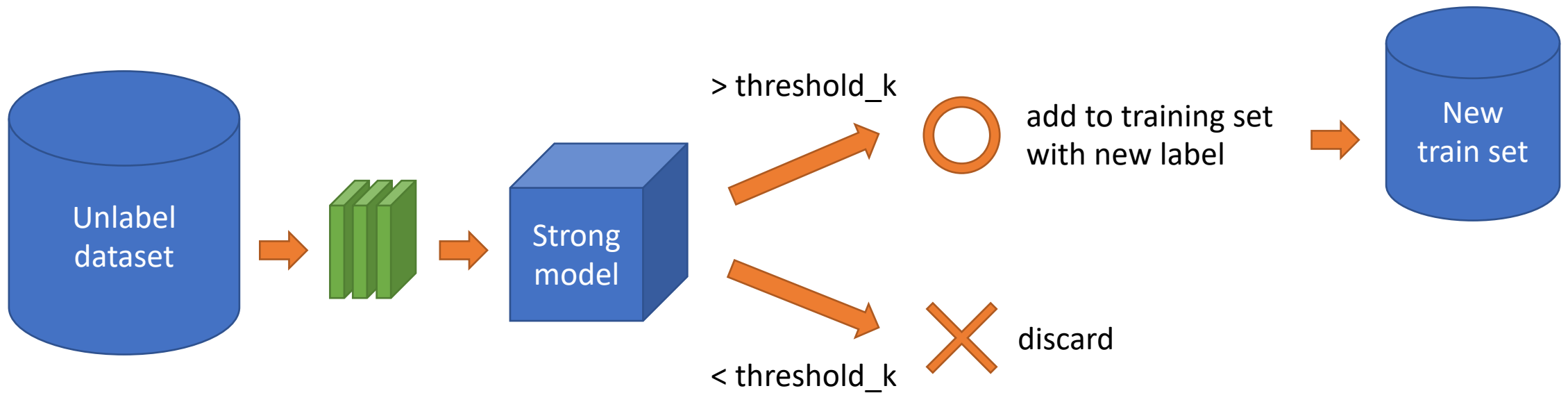
# Data Augmentation

- CNNs are not rotationally invariant! We can apply a serial of data augmentation on training images for a robust model.
- You can simply apply a built-in function in pytorch
  - <https://pytorch.org/vision/stable/transforms.html>



# Semi-supervised Learning (optional)

- Apply a well-trained model to verify if the images need to be preserved.





# Execution

- TAs will execute your code in the following manner, each of them must be finished in **10 mins**
- Part 1.
  - `python3 p1.py`
    - `--feature < tiny_image, bag_of_sift >`
    - `--classifier <nearest_neighbor>`
    - `--dataset_dir <path/to/hw2_data/p1_data>`
- Part 2.
  - `python3 p2_inference.py`
    - `--test_datadir <path/to/hw2_data/p2_data>`
    - `--model_type <mynet, resnet18>`
    - `--output_path <path/to/prediction/csv/file>`
  - `python3 p2_eval.py $1 $2` => this should **reproduce the accuracy** in your report.
    - `--csv_path <path/to/prediction/csv/file>`
    - `--annos_path <path/to/annotation/file>`

# Performance (65%)

- Part 1 : 25%

	Tiny Image (10%)	
	Accuracy	Grade
baseline	0.2	10%

	Bag of SIFT (15%)	
	Accuracy	Grade
strong baseline	0.6	15%
simple baseline	0.55	10%

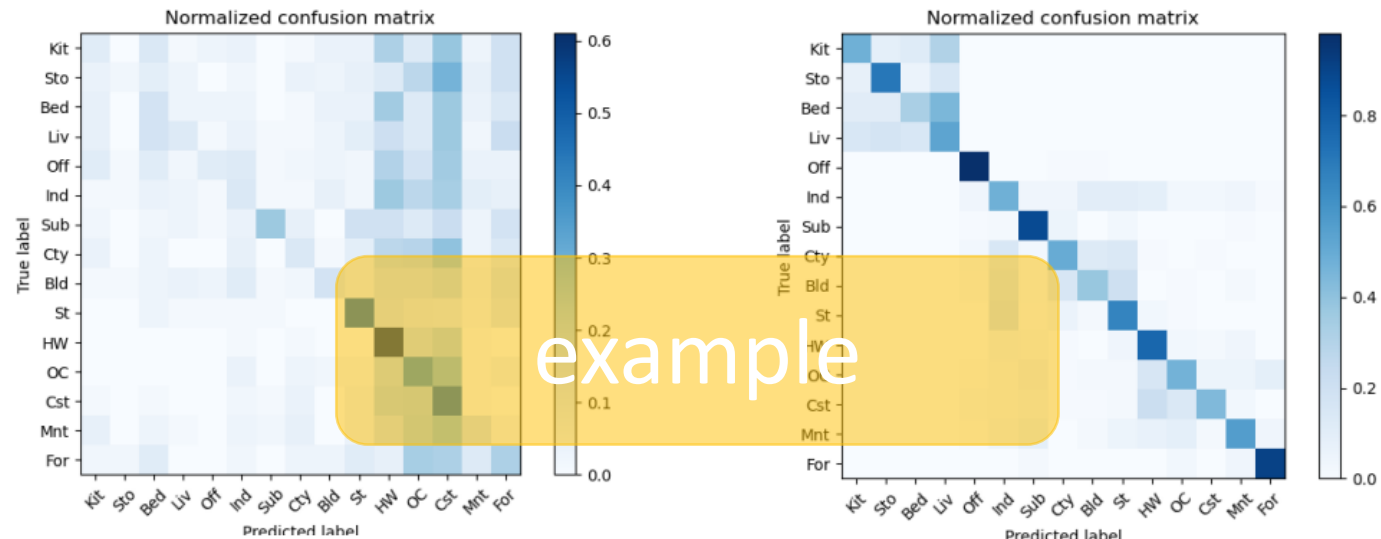
- Part 2 : 40% (The better result of your two models will be used here)

	Public Validation Set (20%)	
	Accuracy	Grade
strong baseline	0.88	20%
medium baseline	0.85	15%
simple baseline	0.75	10%

	Private Test Set (20%)	
	Accuracy	Grade
strong baseline	0.88	20%
medium baseline	0.85	15%
simple baseline	0.75	10%

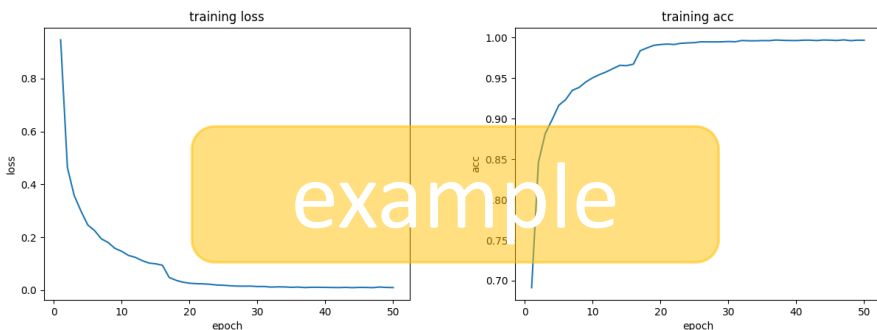
# Report (35%)

- Follow the template we provide and submit **pdf** file
- Part 1 : BoW Scene Recognition (10%)
  - (5%) Plot confusion matrix of two settings.
  - (5%) Compare the results / accuracy of both settings and explain the result.



# Report (35%)

- Part 2 : CNN Image Classification (25%)
  - (2%) Report accuracy of both models (both A & B) on the validation (public) set.
  - (5%) Print the network architectures & number of parameters of both models. What is the main difference between ResNet and other CNN architectures?
  - (8%) Plot four learning curves (loss & accuracy) of the training process (train/validation) for both models. Total 8 plots.
  - (10%) Briefly describe what method do you apply on your best model? (e.g. data augmentation, model architecture, loss function, etc)



```
ResNet(  
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)  
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
  (relu): ReLU(inplace=True)  
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)  
  (layer1): Sequential(  
    (0): BasicBlock(  
      (conv1): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)  
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)  
      (relu): ReLU(inplace=True)
```

# Submission

- **Deadline: 2025/04/11**
- **Fill in the form** [CV 2025 HW2 environment](#)
- **R12345678\_hw2/**
  - p1/
    - p1.py, utils.py
    - vocab.pkl, test\_image\_feats.pkl, train\_image\_feats.pkl
  - p2/
    - p2\_train.py, p2\_inference.py
    - config.py, model.py, dataset.py
    - download.sh (optional)
    - checkpoint/
      - mynet\_best.pth, resnet18\_best.pth
- Submit **report.pdf** & **StudentID\_hw2.zip** (e.g. **R01234567\_hw2.zip**) to NTU COOL
  - **IMPORTANT: There must be a directory named StudentID\_hw2/ after unzipped**



# Submission

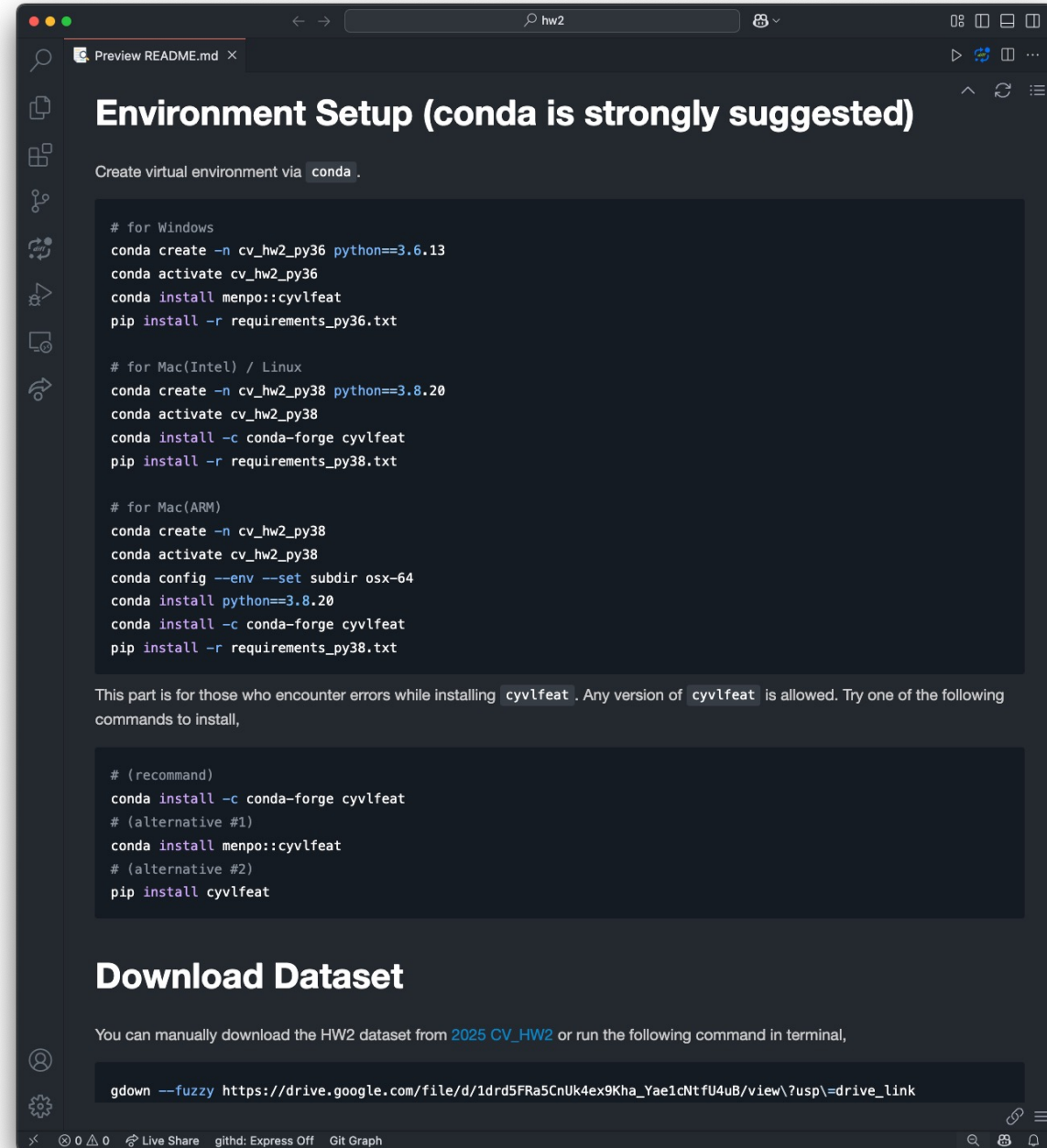
- DO NOT upload the dataset
- Late policy (refer to hw1)
- **Do not delete your trained model** before the TAs disclose your homework score and before you make sure that your score is correct.
- **Plagiarism is forbidden!**

# Submission

- For part2,
  - For each model, the size should be **less than 80MB**.
  - If you cannot match the validation accuracy in your report...
    - TA will send you an e-mail to run another chance only for one time.
  - If your models are too large to submit on NTU COOL, **provide download.sh** under p2/ to download them from cloud drive (Dropbox, Google Drive...) to checkpoint/ (use **-O** argument)
    - TA will run `bash download.sh` first if this script exists.
    - You can use **wget, gdown, unzip**, etc. command.
    - Remember to set permission to **public**.

# Packages

- You should follow README.md to install environment.
- **Python==3.8.20**
  - cyvlfeat=0.7.1
  - torch==2.2.1
  - torchvision==0.17.1
  - torchaudio==2.2.1
  - matplotlib==3.7.5
  - numpy==1.24.0
  - Pillow==10.4.0
  - scikit-learn==1.3.2
  - scipy==1.8.1
  - tqdm==4.67.1
  - gdown==5.2.0
- **Python==3.6.13**
  - cyvlfeat=0.7.0
  - torch==1.10.1
  - torchaudio==0.10.1
  - torchvision==0.11.2
  - matplotlib==3.3.4
  - numpy==1.19.5
  - Pillow==8.4.0
  - scikit-learn==0.24.2
  - scipy==1.5.4
  - tqdm==4.64.1
  - gdown==4.6.4



```
Environment Setup (conda is strongly suggested)

Create virtual environment via conda.

# for Windows
conda create -n cv_hw2_py36 python==3.6.13
conda activate cv_hw2_py36
conda install menpo::cyvlfeat
pip install -r requirements_py36.txt

# for Mac(Intel) / Linux
conda create -n cv_hw2_py38 python==3.8.20
conda activate cv_hw2_py38
conda install -c conda-forge cyvlfeat
pip install -r requirements_py38.txt

# for Mac(ARM)
conda create -n cv_hw2_py38
conda activate cv_hw2_py38
conda config --env --set subdir osx-64
conda install python==3.8.20
conda install -c conda-forge cyvlfeat
pip install -r requirements_py38.txt

This part is for those who encounter errors while installing cyvlfeat. Any version of cyvlfeat is allowed. Try one of the following commands to install,

# (recommand)
conda install -c conda-forge cyvlfeat
# (alternative #1)
conda install menpo::cyvlfeat
# (alternative #2)
pip install cyvlfeat

Download Dataset

You can manually download the HW2 dataset from 2025 CV\_HW2 or run the following command in terminal,

gdown --fuzzy https://drive.google.com/file/d/1drd5FRa5CnUk4ex9Kha_Yae1cNtfU4uB/view?usp=drive_link
```



# If You Still Have Problems...

1. **Use NTU COOL** (strongly recommended)
  - TA will answer the questions ASAP and record some common problems you may face.
2. E-mail to TA or use office hours
3. Google

# TA Information

- Tang-I Wang (王瑭毅)
  - E-mail: [tiwang@media.ee.ntu.edu.tw](mailto:tiwang@media.ee.ntu.edu.tw)
  - TA hour: Mon. 14:20-16:20
  - Location: 博理 421
- Yu-Ching Fan (范宇清)
  - E-mail: [jackmafan@media.ee.ntu.edu.tw](mailto:jackmafan@media.ee.ntu.edu.tw)
  - Location: 博理 421

# Supplementary

# Tips for Achieving the Baselines

- In Part 1.
  - Consider different metrics to evaluate the distance between features.
  - Ref: [scipy.spatial.distance.cdist -- SciPy Manual](#)
- In Part 2.
  - In addition to data augmentation and semi-supervised techniques, you can consider modifying ResNet18 architecture since the first convolution layer's kernel size of ResNet18 is 7x7 which is relatively large to 32x32 images and may loss some information.