HW2 – Scene Recognition & Image Classification Using CNN

Computer Vision

NTU, Spring 2022

Announced: 03/25 2022(Fri.)

Due: 04/15 2022(Thur.) 23:59

Outline

- Part 1. Scene Recognition
  - Use SIFT in opencv as a feature extractor.
  - Apply K-Nearest Neighbor Algorithm as a weak classifier.
- Part 2. Image Classification
  - Use convolutional neural network as a feature extractor and perform image classification.

What You Will Learn

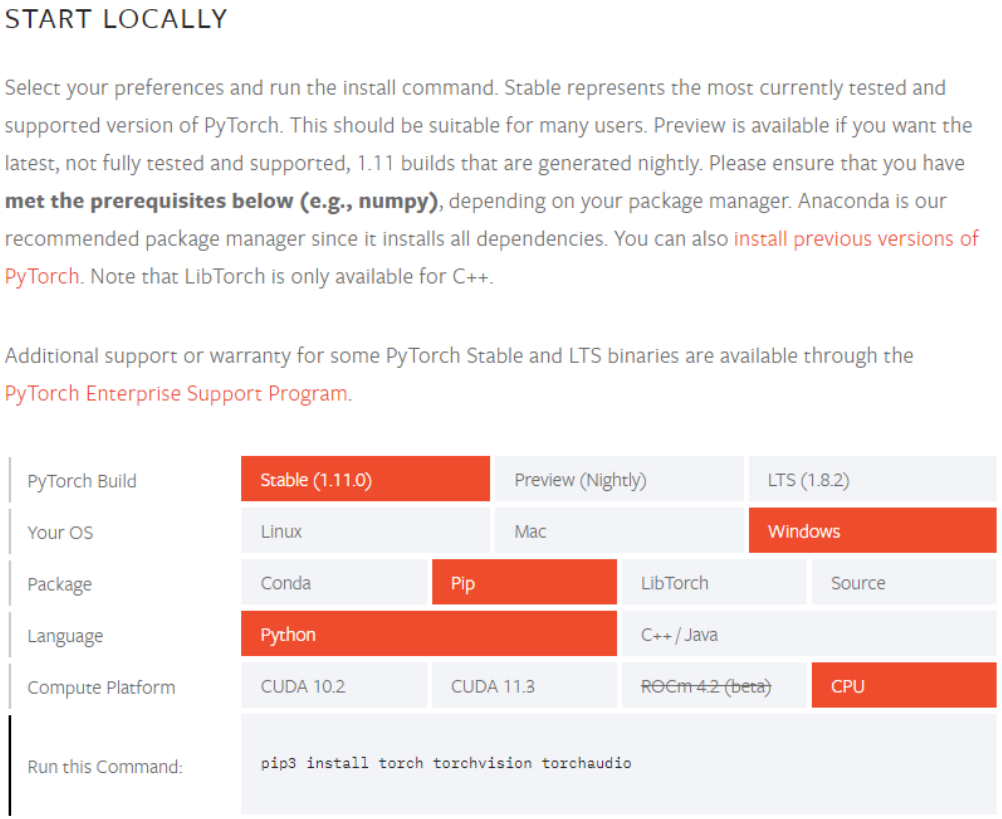
- Basics
  - What the pipeline of bag of sifts is.
  - How to build a KNN classifier.
  - Some useful function in opencv and cyvlfeat.
  - How to build convolutional layers, fully-connected layers and residual blocks in CNN-based model.
  - How to train a model under pytorch framework.

- Advanced learning (optional)
  - How to perform simple data augmentation method in pytorch to gain accuracy.
  - How to perform data cleaning method on some dirty data.
  - How to supply semi-supervised to unlabeled data.

Preparation

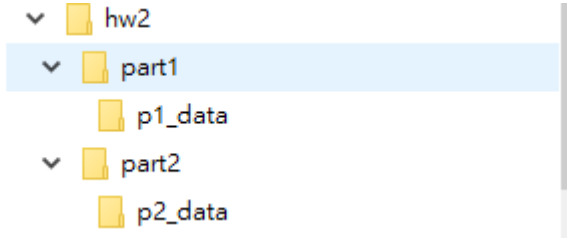
- Before getting started, you have to download the dataset in the following link,

- Install pytorch package in official website. (strongly recommended)
  - Start Locally | PyTorch [1]



Preparation

- Extract hw2_data.zip. Move ‘’p1_data’’ and ‘’p2_data’’ in your working space as shown in the following directory.

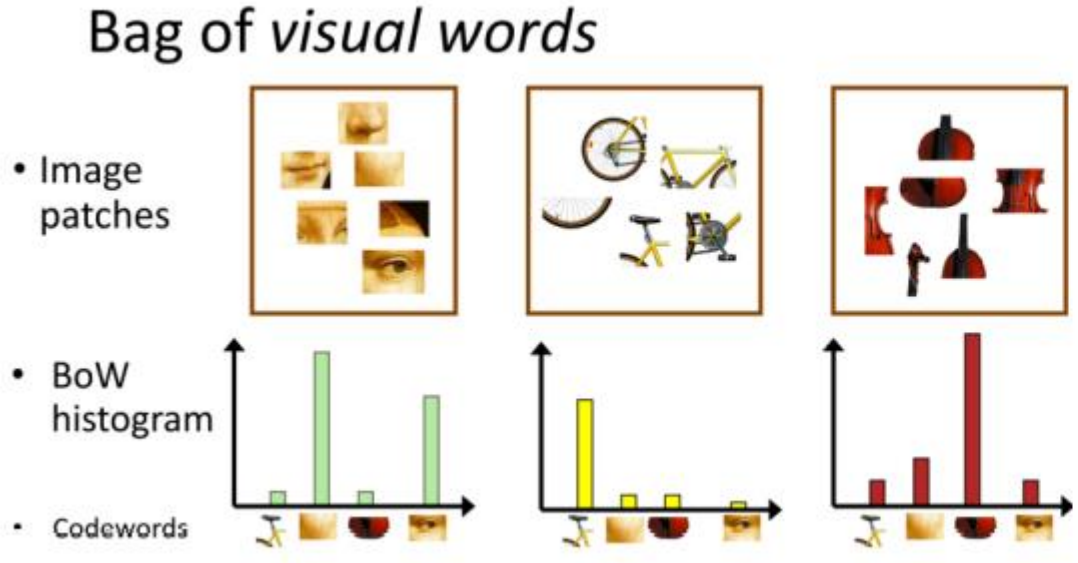


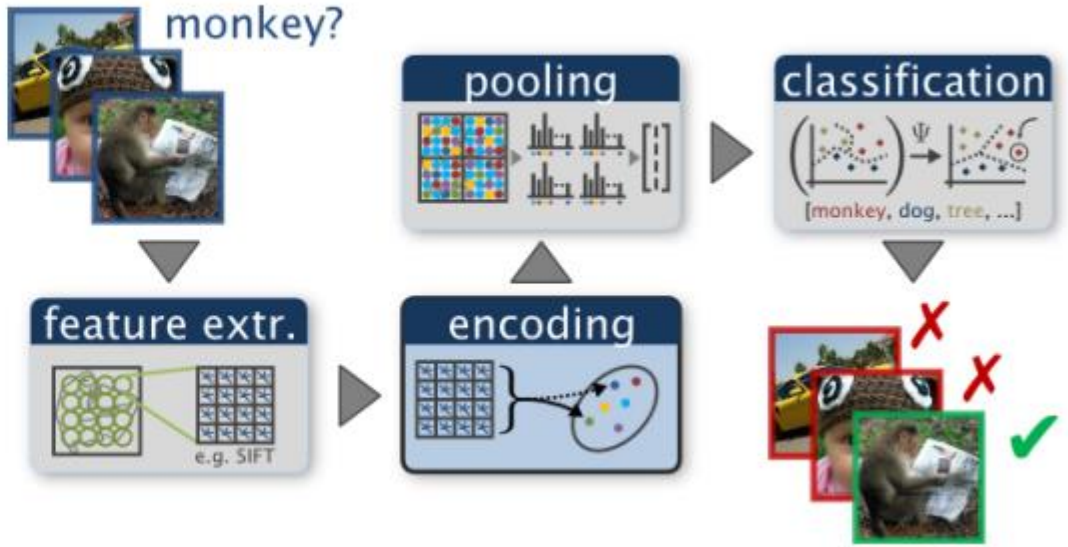
- The download link for hw2_data.zip
  - https://drive.google.com/u/1/uc?id=1Uq1_00JtfZ8ETueo8RjPvL6ANhIc1qvD&export=download

Part 1. Scene Recognition

BoW Scene Recognition

- Tiny images representation and nearest neighbor classifier.
- Bag of SIFT representation and nearest neighbor classifier.





Reference:[2] http://media.ee.ntu.edu.tw/courses/cv/21S/slides/cv2021_lec05.pdf , page 46-49

Dataset Description [3]

- hw2_data/p1/train
  - 100 images per category
- hw2_data/p1/test
  - 100+ images per category


```
p1
├── 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

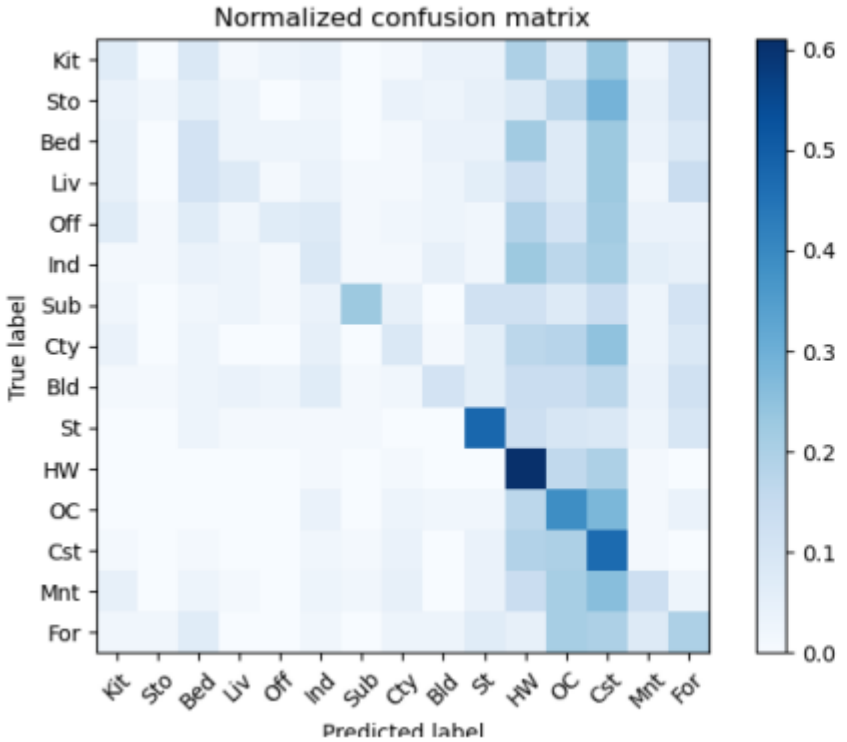
You will have the files….

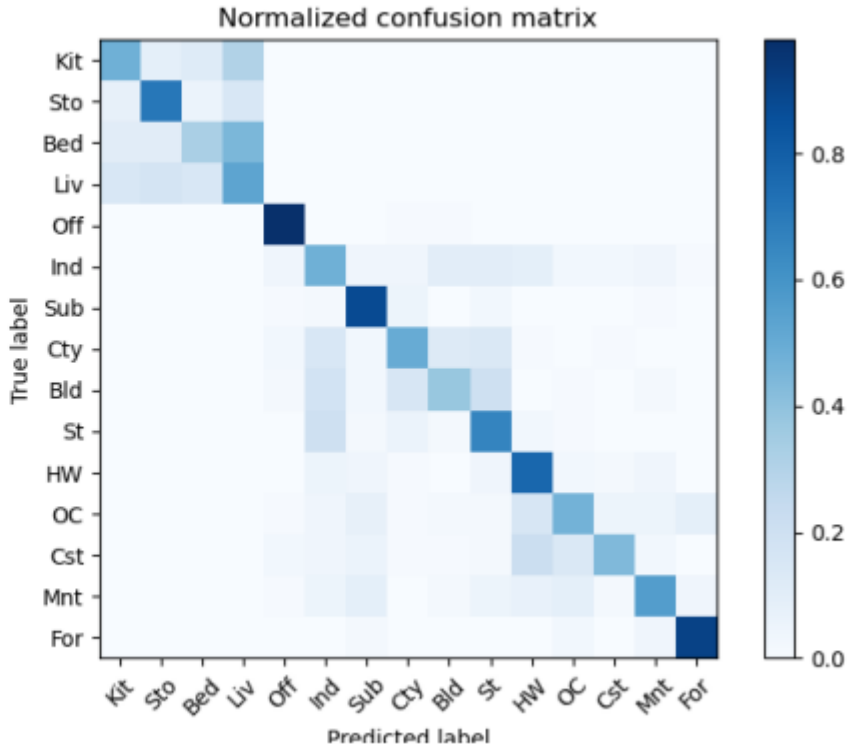
- part1/p1.py
  - Read image, construct feature representations, classify features, etc.
- part1/get_tiny_images.py ## TO DO ##
  - Build tiny images features.
- part1/build_vocabulary.py ## TO DO ##
  - Sample SIFT descriptors from training images, cluster them with k-means and return centroids.
- part1/get_bags_of_sifts.py • ## TO DO ##

  Construct SIFT and build a histogram indicating how many times each centroid was used.
- part1/nearest_neighbor_classify.py ## TO DO ##
  - Predict the category for each test image.
  - CAN NOT USE sklearn.neighbors.KNeighborsClassifier

Confusion Matrix

- **Result visualization – confusion matrix**





Part 2. Image Classification

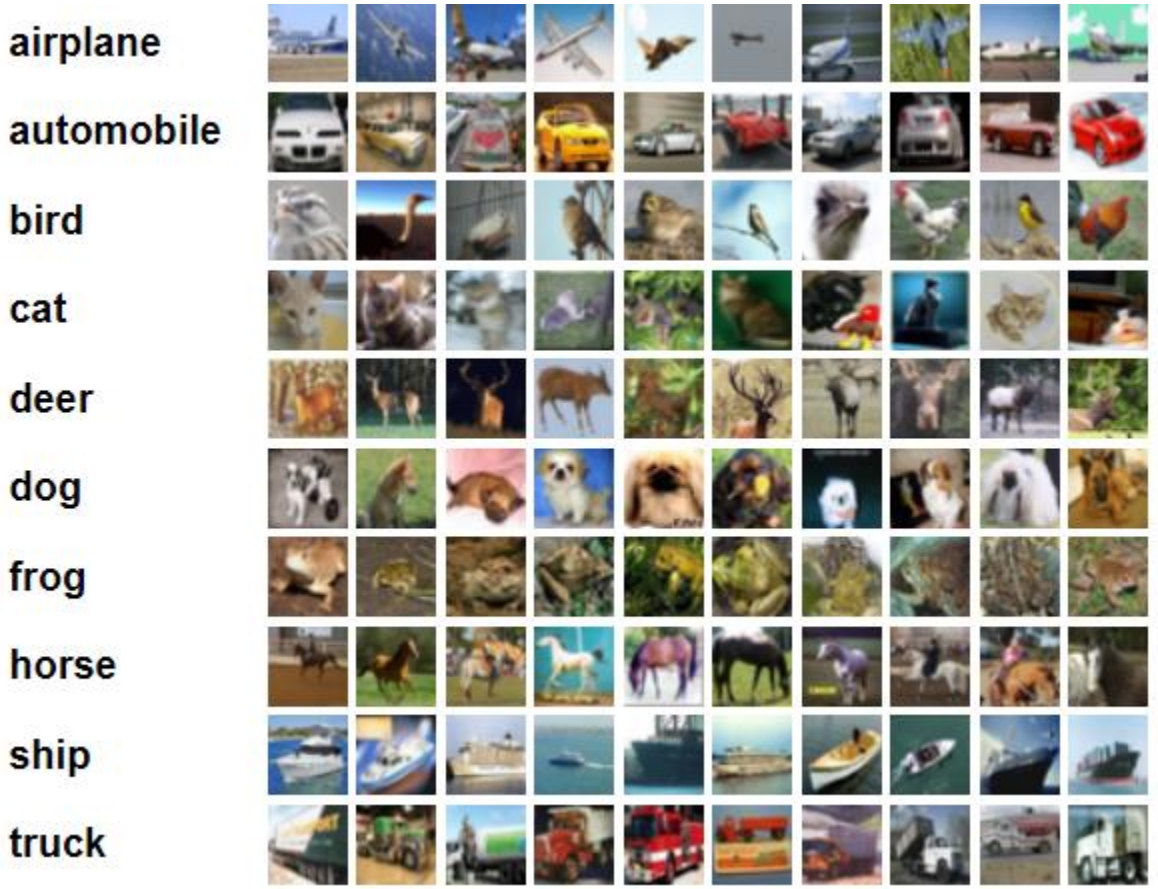
Task Description

- **In this part, you need to build a CNN-based model to predict the labels for the certain images.**

Ship, Cat, Ship, Dog, Dog,
Horse, Deer, Frog, Ship, Frog

Dataset Description

- Original Cifar-10 dataset [4] contains
  - 50,000 training images.
    - For each class it contains 5000 images
  - 10,000 test images.
- In the dataset, all images consists RGB channel with resolution equal to 32*32.



Dataset Description

- In the homework, we are using Cifar-10 mini given by TA
  - It contains only 23,000 images for training.
  - In the training set, TA mixed up to 3000 dirty images which may not belong to any of the class.
  - Besides, you will get extra 30,000 unlabeled data for semi-supervised
  - TA will give you 5000 images as the public score for testing.
  - TA will evaluate another 5000 images as the private score.

Example for dirty images

Bird, Airplane, Deer, Frog, Bird
Dog, Horse, Deer, Ship, Truck

Assignment Description

You will have the files….

- part2/main.py
  - Top. Start training and some basics settings, etc.
- part2/cfg.py
  - Some hyperparameters, seeds setting for certain mode.
- part2/myDatasets.py
  - Define your customized Datasets for training process. ## TO DO ##
- part2/tool.py ## TO DO ##
  - Functions/tools for saving/loading model parameters, Training/validation process, and some other useful function.
- part2/myModels.py • ## TO DO ##
  - Define your own modelzoo including at least myResnet and myLenet
- part2/eval.py
  - Predict the labels for public datasets. ## TO DO ##
- Note : In part2. , feel free to modify all the files as long as it's reasonable and reproducible.

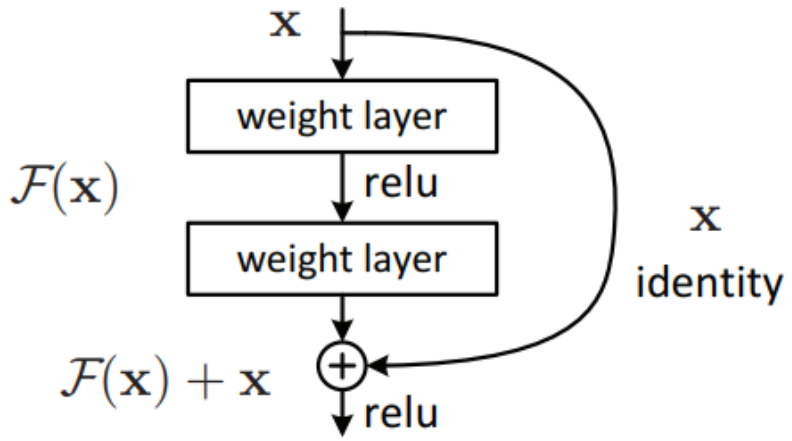
Basic Flows for Training a Model

1. Define your “cifar10_dataset” for data.
    - You can apply data augmentation in the stage.
2. Define your objective function or loss function.
3. Train the model iteratively
    - Forward → Calculate gradient → Backpropagation
4. Apply validation set in the case of overfitting.
    - Only apply forward path to check accuracy compared with the training process.
5. Apply the well-trained model on your test set.

Residual Block

- Build your own Resnet using the residual block.

- Residual block [5]
  - You can add extra layer yourself.




```python
class residual_block(nn.Module):
    def __init__(self, in_channels):
        super(residual_block, self).__init__()

        self.conv1 = nn.Sequential(nn.Conv2d(in_channels=in_channels, out_channels=in_channels, kernel_size=3, padding=1),
                                   nn.BatchNorm2d(in_channels))

        self.relu = nn.ReLU()

    def forward(self,x):
        ## TO DO ##
        # Perform residaul network.
        # You can refer to our ppt to build the block. It's ok if you want to do much more complicated one.
        # i.e. pass identity to final result before activation function
        pass


class myResnet(nn.Module):
    def __init__(self, in_channels=3, num_out=10):
        super(myResnet, self).__init__()

        self.stem_conv = nn.Conv2d(in_channels=in_channels, out_channels=64, kernel_size=3, padding=1)

        ## TO DO ##
        # Define your own residual network here.
        # Note: You need to use the residual block you design. It can help you a lot in training.
        # If you have no idea how to design a model, check myLeNet provided by TA above.

        pass

    def forward(self,x):
        ## TO DO ##
        # Define the data path yourself by using the network member you define.
        # Note : It's important to print the shape before you flatten all of your nodes into fc layers.
        # It help you to design your model a lot.
        # x = x.flatten(x)
        # print(x.shape)

        pass
```

Data Augmentation

- CNN are not rotational-invariant !! We need to apply a serial of data augmentation to train a robust model.

- You can simply apply a built-in function in pytorch
  - https://pytorch.org/vision/stable/transforms.html

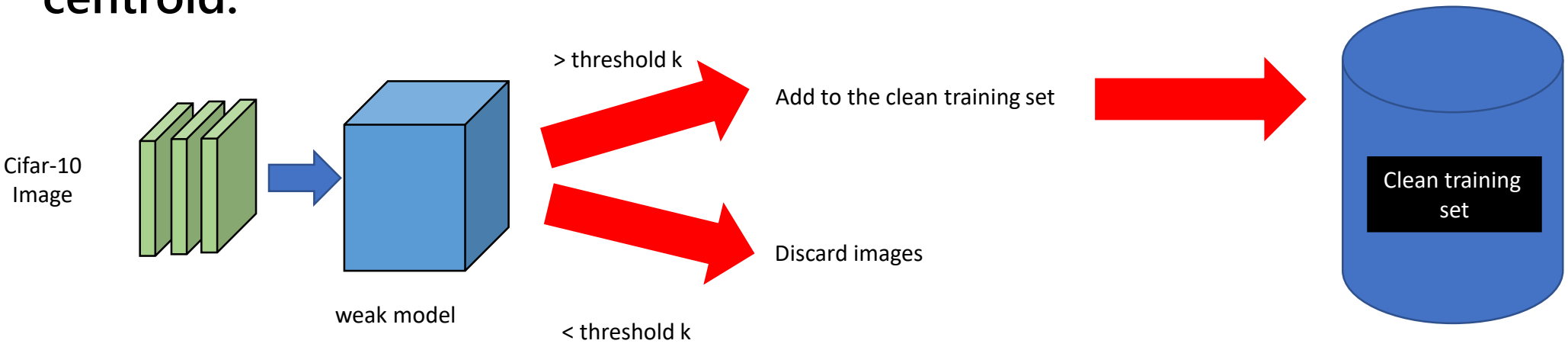
```
    ## TO DO ##
    # Define your own transform here
    # It can strongly help you to perform data augmentation and gain performance
    # ref: https://pytorch.org/vision/stable/transforms.html
    means = [0.485, 0.456, 0.406]
    stds = [0.229, 0.224, 0.225]
    train_transform = transforms.Compose([
                ## TO DO ##
                # You can add some transforms here

                transforms.ToTensor(),
                transforms.Normalize(means, stds),
            ])
```

Cleaning The Data

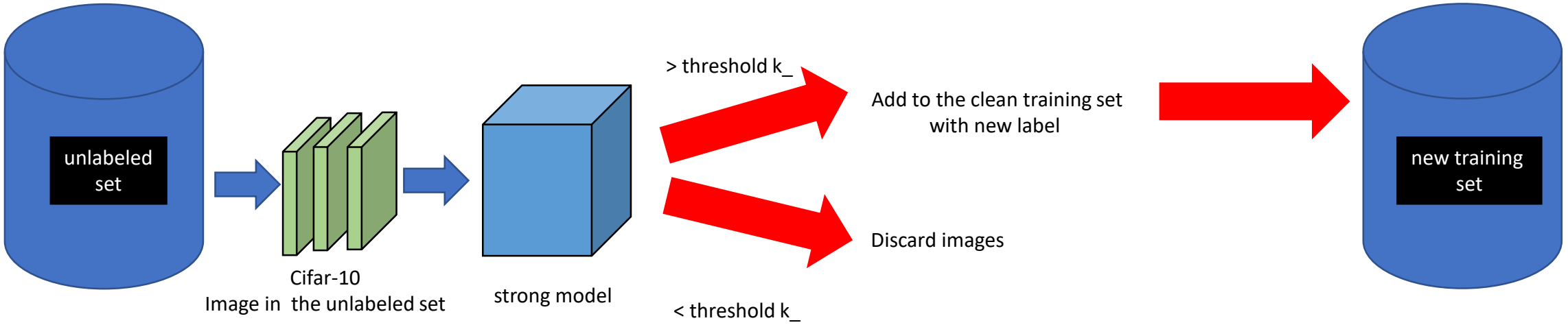
- You could apply a weak model to the original dataset.
  - Discard the image with low confidence.
  - Reserve the image with high confidence.

- You could also apply K-mean centroids algorithm.
  - Discard the images whose feature maps are far away from the centroid.



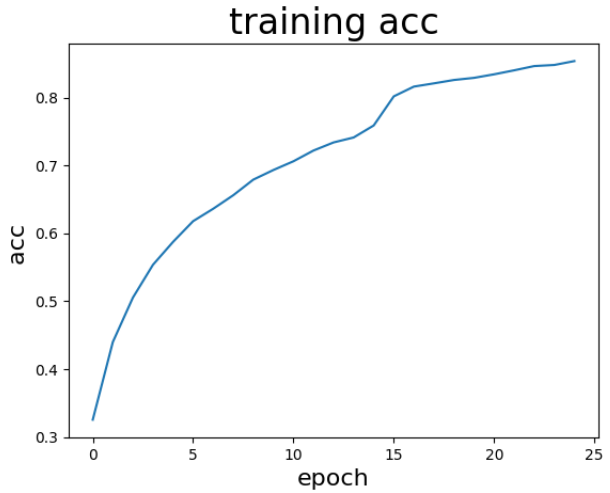
Semi-supervised Learning

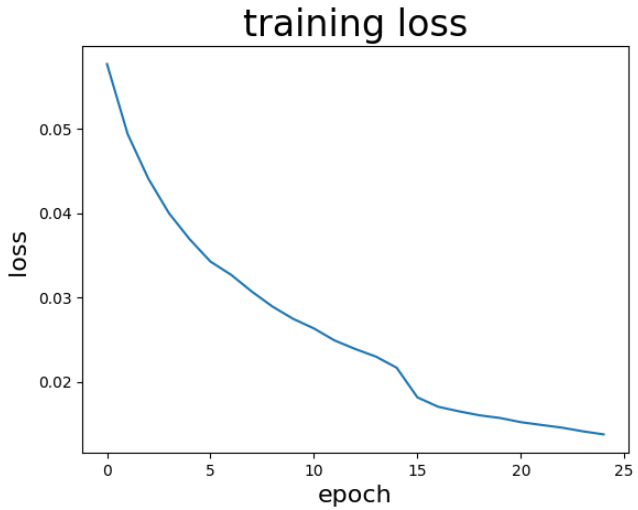
- Apply a well-trained model to verify if the images need to be preserved.



Plot Learning Curve

- Report your model's architecture and learning curve for both Lenet and resnet.






```
myResnet(
  (stem_conv): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
  (layer1): residual_block(
    (conv1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (relu): ReLU()
  )
  (cnn_layer2): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
  )
  (layer3): residual_block(
    (conv1): Sequential(
      (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (relu): ReLU()
  )
  (cnn_layer4): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (3): ReLU()
  )
  (layer5): residual_block(
    (conv1): Sequential(
      (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
      (1): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    )
    (relu): ReLU()
  )
  (fc1): Sequential(
    (0): Linear(in_features=8192, out_features=512, bias=True)
    (1): ReLU()
  )
  (fc2): Linear(in_features=512, out_features=10, bias=True)
)
```

Submission and Some Notes

Execution of hw2

- TAs will run your codes as the following manner,
- Part 1.
  - python3 p1.py $1 $2 $ 3
    - $1: type of feature representation (tiny images/SIFT)
    - $2: type of classifier(nearest neighbor)
    - $3: dataset path
  - E.g., python3 p1.py -- feature tiny_images – classifier nearest_neighbor – dataset_path ./p1_data/
- Part 2.
  - python3 main.py → It need to generate your final result (.pt) and reproduce the accuracy in your report.
  - python3 eval.py $1 $2
    - $1 the path of model's parameter
    - $2 the path of the annotation for test data
  - E.g.
  - python3 eval.py --path ./save_dir/LeNet/best_model.pt --test_anno ./data/annotations/public_test_annos.json

Submission

- Directory architecture:

  R07654321_hw2/

  -- / report.pdf

  -- /part1

  --/ p1.py, get_tiny_images.py, build_vocabulary.py, get_bags_of_sifts.py, nearest_neighbor_classify.py

  --/ vocab.pkl, train_image_feats.pkl, test_image_feats.pkl

  -- /part2

  -- / ReadMe.MD → Write a simple read me to tell TA how to run your code. Important !!

  -- / train.py → It needs to generate .pth file, which is the final parameter of the model you use in your report.

  -- / eval.py → TA could simply run, " python3 eval.py " to utilize .pth file that train.py generated to evaluate your code on public dataset

  -- / myModel.py → To clearly show the model you build yourself.

  -- / any other file you need.

- Put all above files in a directory (StudentID_hw2, e.g. R07654321_hw2) and compress the directory into zip file StudentID_hw2.zip (e.g. R07654321_hw2.zip)
  - Note : You don't need to submit your model's parameter (.pt) or raw data !!!!!!!!
- Submit to NTU COOL
- Deadline: 111/04/13 (Thur.) 23:59
  - Late policy: http://media.ee.ntu.edu.tw/courses/cv/22S/hw/delay_policy.pdf

Rules

- You can only use the dataset TA provided. You are not allowed to used original dataset online.

- Pretrained model is allowed if and only if the model is pretrained on ImageNet. No more pretrained weights are allowed.

- TA will reproduce your training in part2. If you cannot match the accuracy in your report, TA will send you an e-mail to run another chance only for one time.
  - Note: Your code needs to be finished training in 12 hours and your model size should less than 100 MB.

- Plagiarism is forbidden !

- If you violate the any of the rules above, you will get 0 point in the corresponding sections.

Grading

- Part 1 : 30 % + extra 5 %
  - Model Performance(25%)
    - Accuracy should be above the baseline score to get points
      - Tiny images + KNN(10%): 0.2
      - Bag of SIFT + KNN (15%):
        - Weak baseline (9%): 0.55
        - Strong baseline (6%): 0.6
  - TAs will execute your code to check if you pass the baseline.
  - Report(10%)
    - Plot confusion matrix of two settings. (5%)
    - Compare the results/accuracy of both settings and explain the result. (5%)

Grading

- Part 2 : 70 % + extra 5 %
  - Model Performance(40%)
  - TA will test your code on both private and pubic test sets.
    - Weak baseline : accuracy > 64.65%. Both 5 % for the private and public test sets (10%)
    - Medium baseline : accuracy > 72.30%. Both 7.5 % for the private and public test sets (15%)
    - Strong baseline : accuracy > 77.92%. Both 7.5% for the private and public test sets (15%)
  - Report(35%)
    - Compare the performance on residual networks and LeNet. Plot the learning curve (loss and accuracy) on both training and validation sets for both 2 schemes. 8 plots in total. (20%)
    - Attach basic information of the model you use including model architecture and number of the parameters. (5%)
    - Briefly describe what method do you apply ? (e.g. data augmentation, model architecture, loss function, etc.)(10%)

Packages TA used

- python: 3.6+
- numpy: 1.19.2+
- cyvlfeat: 0.5.1+
- matplotlib: 3.3.4+
- pillow: 8.1.2+
- scipy: 1.5.2+
- opencv-python: 4.5.1+
- sklearn: 0.24.1+
- pytorch: 1.5.1+ (https://pytorch.org/get-started/locally/)
- torchvision: 0.6.1+
- And other standard python packages
- E-mail or ask TA first if you want to import other packages.

If You Have Problems…

- Use NTU cool (strongly recommended)
  - TA will answer the questions every day and record some common problems you may face.
  - https://cool.ntu.edu.tw/courses/14814/discussion_topics/94504
- Attach TA
  - E-main: mjsun@media.ee.ntu.edu.tw
  - Office hour in the lab or online TA hour
- Goooooooooogle yourself.

TA information

- Ming-Jhih Sun (孫名志)

  E-mail: mjsun@media.ee.ntu.edu.tw

  TA hour: Wed. 10:00 - 12:00

  Location: 博理 421

  Online TA hour : Every Mon. & Thur. 20:00 – 21:00 during the period of hw2

  https://meet.google.com/txc-ppsa-sai

  Note: Send an e-mail including your problems if you want to attend online TA hour !!

- Chih-Ting Liu (劉致廷)

  E-mail: jackieliu@media.ee.ntu.edu.tw

  Location: 博理 421

Supplementary

Tips for Achieving the Baselines

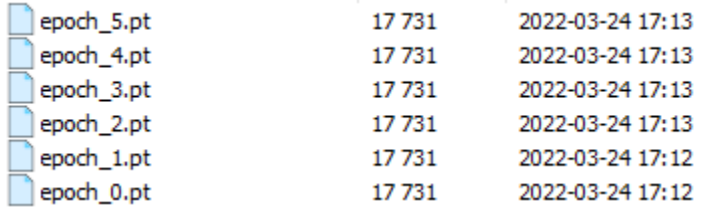
- In Part 1.
  - Consider different metrics to evaluate the distance between features.
  - Ref: scipy.spatial.distance.cdist — SciPy v1.8.0 Manual [6]

- In part 2.
  - Weak baseline – a simple residual model + little effort
  - Medium baseline – data augmentation + adjusting learning rate (+add some batchnorm and pooling layers)
  - Strong baseline – a stronger model + data augmentation + data cleaning + semi-supervised + adjusting learning rate +add some batchnorm and pooling layers

TA's Experience for Part 2.

- TA trained the model by CPU only.
  - TA's equiptments : Intel(R) Core(TM) i9-9900K CPU @ 3.60GHz + 128 GB RAM

- The model TA applied.
  - TA applied a super shallow model with only 2 residual blocks without any pretrained weight.
  - The size of the parameter is approximately 17 MB only.



- Training milestones
  - For each epoch with 23,000 images, it takes ~ 4 mins.
  - TA runs total 25 epochs to get the performance by 73.1%
  - With another 25 epochs for data cleaning, TA get the performance by 77.3%.
  - Semi-supervised is applied and TA trained 48,262 images in total.
    - For each epoch, it takes ~ 8 min. With total 25 epochs, TA get the performance by 79.54%.

For Pytorch & Colab Tutorial

- For Pytorch & colab tutorial
  - [7] Machine Learning Couse. Prof. Lee-HY Website ML 2022 Spring (ntu.edu.tw)
  - colab tutorial
    - https://www.youtube.com/watch?v=YmPF0jrWn6Y
    - https://speech.ee.ntu.edu.tw/~hylee/ml/ml2022-course-data/Colab%20Tutorial%202022.pdf
  - Pyortch tutorial 1
    - https://www.youtube.com/watch?v=85uJ9hSaXig
    - https://speech.ee.ntu.edu.tw/~hylee/ml/ml2022-course-data/Pytorch%20Tutorial%201.pdf
  - Pytorch tutorial 2
    - https://www.youtube.com/watch?v=VbqNn20FoHM
    - https://speech.ee.ntu.edu.tw/~hylee/ml/ml2022-course-data/Pytorch%20Tutorial%202.pdf

Reference

- [1] Pytoch download. https://pytorch.org/get-started/locally/

- [2] Introduction to BoW. http://media.ee.ntu.edu.tw/courses/cv/21S/slides/cv2021_lec05.pdf

- [3] Scence Recognition datasets. S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," CVPR, 2006

- [4] Cifar 10 Datasets Learning Multiple Layers of Features from Tiny Images, Alex Krizhevsky, 2009. CIFAR-10 and CIFAR-100 datasets (toronto.edu).

Reference

- [5] Deep Residual Learning for Image Recognition. K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 770-778, doi: 10.1109/CVPR.2016.90

- [6] scipy.spatial.distance.cdist — SciPy v1.8.0 Manual

- [7] Pytorch & Colab tutorial ML 2022 Spring (ntu.edu.tw)