Lab Project Part 2
CNNs for Image Classification

Computer Vision 1
University of Amsterdam

Due 11:59pm, October 19, 2020 (Amsterdam time)

General Guideline

- **Aim:**
  - Able to understand the Image Classification/Recognition pipeline using a data-driven approach (train/predict stages).
  - Able to implement and test a simple CNN image classify.
- **Prerequisite:**
  - Familiar with Python and relevant packages.
  - Known the basic knowledge of Convolutional Neural Networks.
- **Guidelines:** Students should work on the assignments in a group of **three person** for two weeks. Some minor additions and changes might be done during these three weeks. Students will be informed for these changes via Canvas. Any questions regarding the assignment content can be discussed on Canvas Discussions. Students are expected to do this assignment in Python and Pytorch, however students are free to choose other tools (like Tensorflow). Your source code and report must be handed in together in a zip file (`ID1_ID2_ID3.zip`) before the deadline. Make sure your report follows these guidelines:
  - The maximum number of pages is 10 (single-column, including tables and figures). Please express your thoughts concisely.
  - Follow the given script and answer all given questions (in green boxes). Briefly describe what you implemented. Blue boxes are there to give you hints to answer questions.
  - Analyze your results and discuss them, e.g. why algorithm A works better than algorithm B in a certain problem.
  - Tables and figures must be accompanied by a brief description. Do not forget to add a number, a title, and if applicable name and unit of variables in a table, name and unit of axes and legends in a figure.

  Late submissions are not allowed. Assignments that are submitted after the strict deadline will not be graded. In case of submission conflicts, TAs' system clock is taken as reference. We strongly recommend submitting well in advance, to avoid last minute system failure issues.

  Plagiarism note: Keep in mind that plagiarism (submitted materials which are not your work) is a serious crime and any misconduct shall be punished with the university regulations.

PyTorch Tutorial

This tutorial aims to make you familiar with the programming environment that will be used throughout the course. If you have experience with PyTorch or other frameworks (TensorFlow, MXNet etc.), you can skip the tutorial exercises; otherwise, we suggest that you complete them all, as they are helpful for getting hands-on experience.

Anaconda Environment We recommend installing *anaconda* for configuring *python* package dependencies, whereas it's also fine to use other environment manager as you like. The installation of anaconda can be found in `https://docs.anaconda.com/anaconda/install/`.

Installation The installation is available at `https://pytorch.org/get-started/locally/` depending on your device and system.

Getting start The 60-minute blitz can be found at `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` and and examples at `https://pytorch.org/tutorials/beginner/pytorch_with_examples.html`

Documents Considering there might be potential unknown functions or class, you shall look through the official documents website(`https://pytorch.org/docs/stable/index.html`)and figure out by yourself. (***Think*:** What's the difference between $torch.nn.Conv2d$ and $torch.nn.functional.conv2d$?)

1 Introduction

This part of the assignment makes use of Convolutional Neural Networks (CNN). The previous part makes use of hand-crafted features like SIFT to represent images, then trains a classifier on top of them. In this way, learning is a two-step procedure with image representation and learning. The method used here instead *learns* the features jointly with the classification. Training CNNs roughly consists of three parts: (i) Defining the network architecture, (ii) Reprocessing the data, (iii) Feeding the data to the network, and updating the parameters. Follow the instruction and finish the below tasks.

2 Session 1: Image Classification on CIFAR-10

2.1 Installation

First of all, you need to install Pytorch and relevant packages. In this session, we will use CIFAR-10 as the training and testing dataset.

CIFAR-10 (3-*pts*)

The relevant codes is provided in *Lab_project_part2.pynb*. What you need to do is to run and modify the given code and show the example images of CIFAR-10, describe the classes and image of CIFAR-10.

2.2 Architecture understanding

In this section, we provide two wrapped class of architectures defined by *nn.Module*. One is an ordinary two-layer network with fully connected layers and ReLu, and the other is Convolutional Network utilizing the structure of LeNet-5[2].

Architectures (5-*pts*)

1. Complement the architecture of *TwolayerNet* class and complement the architecture of *ConvNet* class using the structure of LeNet-5[2]. Run the training code of these two architecture. Show the learning curve (3-*pts*)
2. Compare the differences and explain why. (2-*pts*)

2.3 Preparation of training

In above section, we use the *CIFAR10* dataset class from *torchvision.utils* provided by PyTorch. Whereas in most cases, you need to prepare the dataset yourself. One of the ways is to create a *dataset* class yourself and then use the *DataLoader* to make it iterable. After preparing the training and testing data, you also need to define the transform function for data augmentation and optimizer for parameter updating.

Preparation of training (5-*pts*)

1. Complement the *CIFAR10_loader*(2-*pts*)
2. Complement *transform* function and Optimizer (2-*pts*)
3. Train the ConvNet with *CIFAR10_loader*, *transform* and *optimizer* you implemented and show the learning curve. (1-*pts*)

2.4 Setting up the hyperparameters

There are different parameters that must be set properly before training CNNs.These parameters shape the training procedure. They determine how many images to be processed at each step, how much the weights of the network will be updated,how many iterations will the network run until convergence. These parameters are called hyperparameters in the machine learning literature.

Hyperparameter Optimization and Evaluation (10-*pts*)

1. Play with ConvNet and TwolayerNet yourself, set up the hyperparameters and reach the accuracy as high as you can. You can modify the *train*, *Dataloader*, *transform* and *Optimizer* function as you like. You can also modify the architecture of these two Nets. Show the final results and described what you've done for improving the results.
2. Show the results, describe the influence of hyperparameters among TwolayerNet and ConvNet and explain why.

Hint

You can adjust the following parameters and other parameters not listed here as you like: *Learning rate, Batch size, Number of epochs, optimizer, transform function, Weight decay etc.*

3 Session 2: Fine-tuning the ConvNet

The above-implemented network(ConvNet) is trained on a different dataset named CIFAR-10, which contains the images of 10 different object categories, each of which has $32 \times 32 \times 3$ dimensions. The dataset we use in this session is a subset of STL-10 with larger sizes and different object classes. Consequently, there is a discrepancy between the dataset we use to train (CIFAR-10) and the test dataset (STL-10). One solution would be to train the whole network from scratch. However, the number of parameters is too large to be trained properly with such few numbers of images provided. One solution is to shift the learned weights in a way to perform well on the test set, while preserving as much information as necessary from the training class. This procedure is called transfer learning and has been widely used in the literature. This is also called fine-tuning, where the weights of the pre-trained network change gradually. One of the ways of fine-tuning is to use the same architectures in all layers except the output layer, as the number of output classes changes (from 10 to 5).

3.1 STL-10 Dataset

STL-10 Dataset (2-*pts*)

Download STL-10 from provided link. In this session, we only need 5 classes from STL-10. The labels of images can be defined as $\{1 : airplanes, 2 : birds, 3 : ships, 4 : cats, 5 : dogs\}$

- Extract mentioned 5 classes of images from STL-10. Complement *stl10_data* class and match each class with the label accordingly.

Hint

You can use the *functions* from *stl10_input.py* to help to complement *stl10_data* class.

3.2 Fine-tuning ConvNet

In this case, you need to modify the output layer of pre-trained ConvNet module from 10 to 5. In this way, you can either load the pre-trained parameters and then modify the output layer or change the output layer firstly and then load the matched pre-trained parameters. You can find the examples from `https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html` and `https://pytorch.org/tutorials/beginner/transfer_learning_tutorial.html`.

Finetuning ConvNet (5-*pts*)

1. Complement the architecture by fine-tuning from ConvNet and load the pre-trained parameters. (2-*pts*)
2. Train the model and show the results. (3-*pts*)

3.3 Feature Space Visualization

Once the network is trained, it is a good practice to understand the feature space by visualization techniques. There are several techniques to visualize the feature space. In this part, you are expected to use **t-sne**, which is a dimensionality reduction method. **t-sne** takes as input features and labels, then reduces their dimensions to 2, where the structure of the feature and label space is preserved as much as possible.

Visualization (10-*pts*)

In this step, you need to provide the following visualizations:

1. Features of the pre-trained network on the provided dataset
2. Features of the fine-tuned network on the provided dataset

It is necessary to comment on the differences and similarities between the visualization of different settings described above.

Hint

You are able to search the examples online. Specify the source link used in your report.

3.4 Bonus

Bonus (5-*pts*)

- Play with the code and try to get a higher accuracy on the test dataset (5 class from STL-10) as high as you can. The higher accuracy among all teams can get extra points. Specifically, 1st: 5-*pts*, 2nd and 3rd: 4-*pts*, 4th and 5th : 3-*pts*, 6th and 7th: 2-*pts*. 8th-10th: 1-*pts*. Your strategies should be described and explained in your report.

Hint

You can try the following methods but are not restricted to these methods:

- **Data augmentation.**
- **Grid Search.**
- **Freezing early layers.**
- **Modifying Architecture.** you can use a deeper network but you can not use a pre-trained one directly. The only data you can use is from CIFAR-10 and SLT-10.
- **Modifying hyperparameters**
- **Other advice**: `https://cs231n.github.io/transfer-learning/` and Razavian *et. al.* 2014 [3]

References

[1] LeCun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

[2] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (1998): 2278-2324.

[3] Sharif Razavian, Ali, et al. "CNN features off-the-shelf: an astounding baseline for recognition." Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2014.