CS 4476: Computer Vision, Spring 2022
PS5

Instructor: Judy Hoffman

Due: Tuesday, April 19th, 11:59 pm ET

Setup

Note that we will be using a new conda environment for this project! If you run into import module errors, try `pip install -e .` again, and if that still doesn't work, you may have to create a fresh environment.

1. Download and extract the data from the link data.zip. Place the data folder at the root of the project folder.
2. Install Miniconda. It doesn't matter whether you use Python 2 or 3 because we will create our own environment that uses 3 anyways.
3. Open the terminal
   (a) On Windows: open the installed **Conda prompt** to run the command.
   (b) On MacOS: open a terminal window to run the command
   (c) On Linux: open a terminal window to run the command
4. Navigate to the folder where you have the project
5. Create the conda environment for this project.
   (a) On Windows: `conda env create -f proj5_env_win.yml`
   (b) On MacOS: `conda env create -f proj5_env_mac.yml`
   (c) On Linux: `conda env create -f proj5_env_linux.yml`
6. Activate the newly created environment
   (a) On Windows: use the command `conda activate proj5`
   (b) On MacOS: use the command `source activate proj5`
   (c) On Linux: use the command `source activate proj5`
7. Install the project files as a module in this conda environment using `pip install -e .` (Do not forget the `.`).
8. If you run into issues with Pytorch on Windows, please check if you have the 64 bit version installed.

Run the notebook using `jupyter notebook ./proj5_code/proj5.ipynb`.

At this point, you should see the jupyter notebook in your web browser. Follow all the instructions in the notebook for both the code + report portions of this project.

Dataset Details

The dataset used in this project is the same as PS4. The dataset is 15 scene dataset, with natural scenes from 15 different scenarios. It was first introduced by Lazebnik et al, 2006. Typically the image size is 256 by 256 pixels. The data folder will have two subfolders: train and test. We want our model to predict the class of the image from the RGB image. A sample collection of images are presented below:

Figure 1: Dataset

Assignment Overview

The goal of this project is to get familiar with convolutional neural networks to classify the scenes into different categories. The basic learning objectives of this project are:

- Construct the fundamental pipeline for performing deep learning using PyTorch
- Understand the concepts behind different layers, optimizers
- Experiment with different models and observe the performance

In order to train a model in Pytorch, following four components:

- Dataset: an object which can load the data and labels given an index (Part 1)
- Model - an object that contains the network architecture definition
- Loss function - a function that measures how far the network output is from the ground truth label
- Optimizer - an object that optimizes the network parameters to reduce the loss value

1 Dataset Loaders and Data preprocessing

In this part we will setup the workflow to load the dataset and pre-processing. In DL we generally tend to perform some preprocessing like normalization and transformation. The transformations will aid to make the inputs compatible to the model input dimensions or help in data-augmentation to improve performance.

1.1 Compute mean and std. deviation of the dataset

- Implement in compute_mean_and_std() in stats_helper.py
- Convert the image to grayscale and scale to [0,1] before computing mean and standard deviation
- Use StandardScaler function in the sklearn.preprocessing library

1.2 Data Loaders

We will be using the Pytorch's ImageFolder dataloader here. The images in data folder are organized such that each class name is the folder with the images belonging to that class inside the folder. We'll use the ImageFolder to make two loaders, for train and test split respectively.
Note: You don't have to implement anything in this subpart, go through the documentation of ImageFolder for better understanding.

1.3 Data Transforms

Implement the function get_fundamental_transforms() in data_transforms.py. In this part we will be constructing some fundamental transforms to convert the RGB image to torch tensors:

- Resize: Resize the image in accordance with input dimensions of the architecture
- Grayscale: Convert the loaded image to grayscale (Although the colors of the image are gray, the ImageFolder will read the images as RGB)
- Convert it to tensor
- Normalize: Normalize the images based on mean and std. deviation computed in part 1.1
- Use torchvision's transforms

2 Model Architecture and Forward Pass

In this part of the project, you will be working on implementing a simple 2-layer deep learning architecture in Pytorch.

2.1 SimpleNet Model

You need to add 2 convolutional layers, with supporting utilities like ReLU, Max Pooling, and Fully Connected layers with proper parameters(kernel size, padding etc), to make the output compatible with input for next layer. You can use the following image for a sample network architecture. In the image, the layers in use are written at the bottom with the kernel size, and the blocks show the shape of features after each layer.

To do: Implement the following in the simple_net.py

- Initialize self.conv_layers
- Initialize self.fc_layers
- Write the forward function

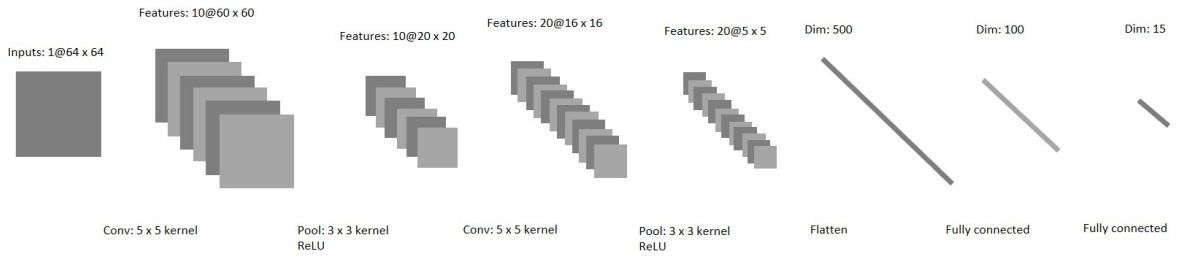

Figure 2: SimpleNet Architecture

2.2 Output Prediction

In this section, you will be looking at how the model generates output and how is it related to classification. The model has been initialized with random weights; hence the output will not reflect the model performance. We will be using a data-point from the ImageFolder object defined in Part1.2. Passing the gray-scale image through the SimpleNet model, a 15-dimensional tensor is returned as output. This 15-dimensional tensor can be converted to probability of the image belonging to each of the class with Softmax activation. To save computation, you will be directly taking the index where the value is maximum in the 15-dimensional tensor returned by the model. Think about how the argmax operation on the 15 dimenasional tensor is same as the argmax operation on the softmax output.

- Implement the function predict_labels() in dl_utils.py

3 Loss Function

In the previous section, you saw how the output from the model relates to classification. Since the model was initialized randomly the output doesnot depict the model performance. Technically should produce a probability of 1 (or close to 1) for the target sample_label and 0 for the other classes. In order to train the model to do so, you will be implementing a loss function to penalize the deviation between the desired probability and the model's predicted output. You will be using KL divergence or cross-entropy loss. Refer to this stackexchange post for a complete explanation.
Implement the following:

- Assign a loss function to self.loss_criterion in simple_net.py. Refer to Pytorch loss documentation.
- Compute the compute_loss in dl_utils.py to use the model's loss criterion and compute loss.

4 Optimizer

In this section, you will be working on minimizing the loss to optimize the weights of the network.

4.1 Manual gradient descent using Pytorch's autograd

Pytorch uses the autograd feature to usew gradient optimization and escape manual computations.

- Complete compute_quadratic_loss in optimizer.py : Define objective function L(w)

$$L(w) = w^2 - 10w + 25 \tag{1}$$

- Implement a single step of gradient descent in function gradient_descent_step in optimizer.py

4.2 Optimization with Pytorch's gradient descent optimizer

You will be using torch.optim. You can experiment with different features of the vanilla gradient descent.

4.3 Setting up the optimizer for SimpleNet

- Initialize and adjust the values of learning rate and weight decay to improve the model performance.
- Implement the get_optimizer.py function in optimizer.py

sectionTraining SimpleNet In this section, you will be training the SimpleNet architecture. Pass in the model architecture, optimizer and the transforms to the Trainer function.

- Adjust the value of your parameters, to obtain 45% validation accuracy to get full credits

5 EC1: Solving Overfitting

Observing the train and validation losses, you will observe that the training accuracy is over 90% while the validation accuracy does not go past 45%. This is an indication of overfitting; the model fits too well with the training data, but the generalization capabilities of the model are limited.

5.1 Jitter, Random Flip, and Normalization

Since we don't have a huge amount of data, we will be adding augmentations to the data, so that the model sees "different" images during every iteration.

- finish the get_data_augmentation_transforms() function in data_transforms.py

5.2 Dropout

Dropout tries to solve the problem of co-adaption. It randomly turns off the connection betweehn neurons in order to make the neurons independent of each other.

- finish the simple_net_dropout.py with your previous SimpleNet model, plus the dropout layer.
- Achieve 52% validation accuracy for full credits for this part.

6 EC2: Pretrained AlexNet

From the previous section, you will realise that the model performance improved in terms of testing accuracy. There is a drop in training accuracy, but its more inline with the final model performance. In order to further improve the testing accuracy, you will be deeper networks with more layers and weights.

- Switch to my_alexnet.py, and copy the network architecture and weights of all but the last fc layers from the pretrained network.
- Freeze the convolutional layers and first 2 linear layers so we don't update the weights of them
- you are required to pass a threshold of 85% for this part.

7 Tips and tricks

- in this project, we will be using the test set as the validation set (i.e. using it to guide our decisions about models and hyperparamters while training). In actual practise, you would not interact with the test set until reporting the final results.
- your CPU should be sufficient to handle the training process for all networks in this project, and the following training cells will take less than 5 minutes; you may also want to decrease the value for num_epochs and quickly experiment with your parameters. The default value of 30 is good enough to get you around the threshold for Part 1, and you are free to increase it a bit and adjust other parameters in this part.
- If the loss decreases very slowly, try increasing the value of the lr (learning rate).
- If the loss decreases very slowly, try increasing the value of the lr (learning rate).
- Try to first adjust lr in multiples of 3 initially. When you are close to reasonable performance, do a more granular adjustment. Do try to keep lr values less than 0.1.
- If you want to increase the validation accuracy by a little bit, try increasing the weight_decay to prevent overfitting. Do not use tricks from Section 6 just yet.
- Try jittering the colors, and flipping the image horizontally. These two operations wont change the scene contents. You can also try resizing and cropping.

Submission Instructions

Recheck you pass all local unit tests by staying in the root directory and running the command `pytest proj5_code/proj5_unit_tests`. This command will run all the unit tests once more, and you need to add a screenshot to the report. Ensure that the conda environment `proj5` is being used.

- Submit the code as zip on Gradescope at PS5 - Code.
- Submit the report as PDF on Gradescope at PS5 - Report.

There is no submission to be done on Canvas.

Rubric

This problem set has 100 points of mandatory credits and 20 extra credits.

Code: The score for each part is provided below. Please refer to the submission results on Gradescope for a detailed breakdown.

Part 1: Dataset	10
Part 2: Model Architecture and Forward Pass	45
Part 3: Loss function	10
Part 4: Optimization	10
Part 5: Training SimpleNet	10
EC 1.1: Data Augmentation	3
EC 1.2: SimpleNetDropout	3
EC 2: MyAlexNet	6
Total	*85 (+12)*

Report: The report is worth 20 points. Please refer to the pptx template where we have detailed the points associated with each question.

Part 1: Dataset	4
Part 3: Loss function	4
Part 5: Training SimpleNet	5
Conclusion	2
EC 1: Overfitting	4
EC 2: MyAlexNet	4
Total	*15 (+8)*

Instructions

1. The assignment must be done in Python3. No other programming languages are allowed.
2. Fill your answers in the answer sheet PPT provided and submit the file under the name: First-Name_LastName_PS5.pdf on Gradescope. Please **do not modify the layout** of the boxes provided in the answer sheet and fit your answers within the space provided. You will be penalized for changing the template.
3. Please enter your code in the designated areas of the template Python files. Please do not add additional functions/imports to the files. Points will be deducted for any changes to code/file names, use of static paths and anything else that needs manual intervention to fix.
4. Please submit your code and output files in a zipped format, using the helper script `zip_submission.py` with your GT username as a command line argument (using `--gt_username`), to Gradescope. Please do not create subdirectories within the main directory. The `.zip_dir_list.yml` file contains the required deliverable files, and `zip_submission.py` will fail if all the deliverables are not present in the root directory. Feel free to comment and uncomment them as you complete your solutions.
5. For the implementation questions, make sure your code is bug-free and works out of the box. Please be sure to submit all main and helper functions. Be sure to not include absolute paths. Points will be deducted if your code does not run out of the box.
6. If plots are required, you must include them in your Gradescope report and your code must display them when run. Points will be deducted for not following this protocol.
7. Ensure that you follow the instructions very carefully.

Deliverables

The code deliverable will be uploaded as a zip file on Gradescope.

Do not create this zip manually. You are supposed to use the command `python zip_submission.py -gt_username <username>` for this.

The second thing to upload is the PDF export of the report on gradescope.

This iteration of the assignment is developed by Deepanshi and Judy Hoffman.

This assignment was developed and maintained by Ayush Baid, Cusuh Ham, Jonathan Balloch, Haoxin Ma, Jing Wu, Shenhao Jiang, Frank Dellaert, James Hays, Judy Hoffman and Deepanshi.