Introduction to Deep Learning for Computer Vision
Assignment 7: Convolutional Neural Networks II

Due: March 19th, 2019

Abstract

This is the final assignment! In this assignment we will extend our previous assignment and implement a resnet-like network. We will then try to see what's the best that we can do.

Please read the instructions carefully.

1 Instructions

1.1 Submission package

Within the homework assignment package, there should be a `submission-package.zip`, which contains the directory structure and empty files for you to get started. Please edit the contents within the file for code, and then create a `zip` archive with the file name `submission-package.zip`, and submit it. **Do not use other archive formats such as `rar` or `tar.gz`.** Also, submit the report in `pdf`, **as a separate file alongside** the archive with the codes.

All assignments should be submitted electronically. Hand written reports are **not** accepted. You can, however, include scanned pages in your report. For example, if you are not comfortable with writing equations, you can include a scanned copy.

1.2 Assignment Report

For this assignment, all execution results should be summarised in an assignment report. Reports should be in `pdf` format. Though not mandatory, students are encouraged to submit their reports written in LaTeX. In the assignment package, you should have been given an empty skeleton file for you to get started.

However, it is **not required** for you to explain your code in the reports. Reports are for discussing results. You **should** however, provide comments in your code, well enough to be understood by directly reading the code.

1.3 Code

All assignments should be in `Python` 3. Codes that fail to run on `Python` 3 will receive 20% deduction on the final score. In other words, do **not** use `Python` 2.7.

For this assignment, you should **not** need to create additional files. Fill in the skeleton files in the submission package. Do **not** change the name of these scripts.

It is **strongly encouraged** to follow `PEP8`. It makes your code much more readable, and less room for mistakes. There are many open source tools available to automatically do this for you.

1.4 Delayed submission

In case you think you will not meet the deadline due to network speed or any other reasons, you can send an email with the `SHA-256` hash of your `.zip` archive first, and then submit your assignment through email later on. This will **not** be considered as a delay.

Delayed submissions are subject to 20% degradation per day. For example, an assignment submitted 1 minute after the deadline will receive 80% of the entire mark, even if it was perfect.

Submission will close 24 hours after the original deadline, as the solution for the assignment will be released to be used for the next assignment.

1.5 Use of open source code

Any library under any type of open source license is allowed for use, given full attribution. This attribution should include the name of the original author, the source from which the code was obtained, and indicate terms of the license. Note that using copyrighted material without an appropriate license is not permitted. Short snippets of code on public websites such as StackOverflow may be used without an explicit license, but proper attribution should be given even in such case. This means that if you embed a snippet into your own code, you should properly cite it through the comments, and also embed the full citation in a LICENSES file. However, if you include a full, unmodified source, which already contains the license within the source file, this is unnecessary. Please note that without proper attribution, *it will be considered plagiarism.*

In addition, as the assignments are intended for you to learn, (1) if the external code implements the core objective of the task, no points will be given; (2) code from other CSC486B/CSC586B students will count as plagiarism. To be more clear on (1), with the assignment, we will release a `requirements.txt` file, that you can use with `pip` to setup your environment. On top, you are also allowed to use `OpenCV3.X`, which we will not include in `requirements.txt` as `OpenCV` installation depends on your own framework.

2 Implementing `model.py/ConvBlock`

2.1 `__init__` (30 points)

Implement the creation of convolution layers according to the comments in the code. The residual network can be invoked with the following command line:

```
python solution.py --log_dir logs/custom --save_dir saves/custom --conv custom
```

This should create the following network:

```
MyNetwork(
  (convs): Sequential(
    (conv_0_base): Conv2d(3, 8, kernel_size=(1, 1), stride=(1, 1))
    (conv_0_0): ConvBlock(
      (layers): Sequential(
        (conv_1): Sequential(
          (0): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(8, 8, kernel_size=(1, 1), stride=(1, 1))
          (2): ReLU()
        )
        (conv_2): Sequential(
          (0): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(8, 8, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (2): ReLU()
        )
        (conv_3): Sequential(
          (0): BatchNorm2d(8, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(8, 8, kernel_size=(1, 1), stride=(1, 1))
          (2): ReLU()
        )
      )
    )
    (conv_0_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_1_base): Conv2d(8, 16, kernel_size=(1, 1), stride=(1, 1))
    (conv_1_0): ConvBlock(
      (layers): Sequential(
        (conv_1): Sequential(
          (0): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(16, 16, kernel_size=(1, 1), stride=(1, 1))
          (2): ReLU()
        )
        (conv_2): Sequential(
          (0): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (2): ReLU()
        )
        (conv_3): Sequential(
          (0): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(16, 16, kernel_size=(1, 1), stride=(1, 1))
          (2): ReLU()
        )
      )
    )
    (conv_1_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (conv_2_base): Conv2d(16, 32, kernel_size=(1, 1), stride=(1, 1))
    (conv_2_0): ConvBlock(
      (layers): Sequential(
        (conv_1): Sequential(
          (0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1))
          (2): ReLU()
        )
        (conv_2): Sequential(
          (0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
          (2): ReLU()
        )
        (conv_3): Sequential(
          (0): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
          (1): Conv2d(32, 32, kernel_size=(1, 1), stride=(1, 1))
          (2): ReLU()
        )
      )
    )
    (conv_2_pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (output): Linear(in_features=32, out_features=10, bias=True)
)
```

2.2 `_conv` (5 points)

Implement the batch normalization – convolution – activation block that we will use to create our residual network block.

2.3 `forward` (5 points)

Implement the forward method.

3 Debugging (20 points)

The current assignment package has a bug that if you try to run your network trained on GPU on a CPU machine, you'll end up with a torch error message. If you do not have a GPU machine, you can also use the already provided weights, which were trained on a GPU.

Debug this error.

4 Reporting

4.1 Reproducing default architecture results – Testing (10 points)

With the provided models, you should obtain the following test performance. These performances are with all other parameters set to default, except for `conv`.

Table 1: Expected test accuracy.

Configuration	Test Accuracy (%)
`--conv torch`	49.79
`--conv custom`	62.68

4.2 Going beyond (30 points)

Use the knowledge you've acquired during this course to enhance the performance of your model. You may change your architectural choice, and/or apply any other type of method that you would like to.

Note that this would probably take some training time. Thus, you are encouraged to use your google cloud credits to do so.

For this part, we will use your final test accuracy as your grade. Your accuracy will be transformed into points linearly, where 62% accuracy will give you zero points, and 75% accuracy will give you 30 points.

Hint: It is possible to achieve this accuracy simply by changing the hyper-parameters. However, you may also do whatever you want to do instead.