BME646 and ECE60146: Homework 5

Spring 2025
Due Date: Tuesday, Feb 18, 2025, 11:59pm
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via Gradescope. Post questions to Piazza. Additional instructions can be found at the end. Late submissions will be accepted with penalty: -10 points per-late-day, up to 5 days.

1 Introduction

In Homework 4, you were introduced to the COCO dataset and learned how to curate datasets. Additionally, you were introduced to Convolutional Neural Networks (CNNs) through the HW4Net architecture, gaining foundational insights into their structure and functionality. In this homework, you will explore the effects of progressively adding convolutional layers to a network. One of the fundamental questions in early deep learning was whether increasing the depth of a model — by simply adding more layers — necessarily leads to improved performance.

We will investigate this using the OOP concepts you have learnt and DLStudio. We will use the CIFAR datasets to analyze the behavior of deeper networks. These exercises are structured around the `DLStudio` framework, emphasizing the importance of code reuse and extensibility through inheritance — an essential concept for any code development.

This assignment aims to provide both practical experience and conceptual insight into deep learning architectures, motivating a deeper understanding of model depth and its implications.

2 Deeper networks

We will go back to playing with CIFAR10 dataset. In the last homework, you worked with `playing_with_cifar10.py`. We will now extend the script to create your own 'Deeper' neural network Net3 to have 8 convolutional layers. One way to do it is given below. The Net3 created is for your reference and does NOT meet the requirement of HW.

```
1     #!/usr/bin/env python
2
3 ##  playing_with_cifar10.py
```

```
4
5  """
6  This is very similar to the example script
                                    playing_with_sequential.py
7  but is based on the inner class ExperimentsWithCIFAR which uses
                                     more common
8  examples of networks for playing with the CIFAR-10 dataset.
9  """
10
11 import random
12 import numpy
13 import torch
14 import os, sys
15
16 """
17 seed = 0
18 random.seed(seed)
19 torch.manual_seed(seed)
20 torch.cuda.manual_seed(seed)
21 numpy.random.seed(seed)
22 torch.backends.cudnn.deterministic=True
23 torch.backends.cudnn.benchmarks=False
24 os.environ['PYTHONHASHSEED'] = str(seed)
25 """
26
27 ##  watch -d -n 0.5 nvidia-smi
28
29 from DLStudio import *
30 import torch.nn as nn
31 import torch.nn.functional as F
32
33
34 dls = DLStudio(
35 #                   dataroot = "/home/kak/ImageDatasets/CIFAR-10
                                    /",
36                    dataroot = "./data/CIFAR-10/",
37                    image_size = [32,32],
38                    path_saved_model = "./saved_model",
39                    momentum = 0.9,
40                    learning_rate = 1e-3,
41                    epochs = 2,
42                    batch_size = 4,
43                    classes = ('plane','car','bird','cat','deer',
                                    'dog','frog','horse','ship','
                                    truck'),
44                #   use_gpu = False,
45                )
46
47 # Inherit  DLStudio.ExperimentsWithCIFAR to extend to Net3
```

```
48
49 class CustomExperimentsWithCIFAR(DLStudio.ExperimentsWithCIFAR)
                                    :
50     class Net3(nn.Module):  # Define a new model
51         def __init__(self):
52             super(CustomExperimentsWithCIFAR.Net3, self).
                                    __init__()
53             self.conv1 = nn.Conv2d(3, 32, kernel_size=3,
                                    padding=1)
54             self.conv2 = nn.Conv2d(32, 64, kernel_size=3,
                                    padding=1)
55             self.conv3 = nn.Conv2d(64, 128, kernel_size=3,
                                    padding=1)
56             self.fc1 = nn.Linear(128 * 4 * 4, 256)
57             self.fc2 = nn.Linear(256, 10)
58             self.pool = nn.MaxPool2d(2, 2)
59             self.dropout = nn.Dropout(0.5)
60
61         def forward(self, x):
62             x = self.pool(F.relu(self.conv1(x)))
63             x = self.pool(F.relu(self.conv2(x)))
64             x = self.pool(F.relu(self.conv3(x)))
65             x = x.view(-1, 128 * 4 * 4)
66             x = F.relu(self.fc1(x))
67             x = self.dropout(x)
68             x = self.fc2(x)
69             return x
70
71
72 # Use the new class
73 # exp_cifar = DLStudio.ExperimentsWithCIFAR( dl_studio = dls )
74 exp_cifar = CustomExperimentsWithCIFAR(dl_studio=dls)
75 exp_cifar.load_cifar_10_dataset()
76
77 # Instantiate the new model
78
79 # model = exp_cifar.Net()
80 #model = exp_cifar.Net2()             ## <<< Try this also but
                                    first comment out
81                                      ##     the above line.
82
83 model = exp_cifar.Net3()
84
85 ## display network properties
86 number_of_learnable_params = sum(p.numel() for p in model.
                                    parameters() if p.requires_grad
                                    )
87 print("\n\nThe number of learnable parameters in the model: %d"
                                     % number_of_learnable_params)
```

```
88
89 exp_cifar.run_code_for_training(model, display_images=False)
90
91 exp_cifar.run_code_for_testing(model, display_images=False)
```

2.1 Experimental Tasks and Analysis

1. **Training and Evaluation of Neural Networks:** Execute Net and Net2 from ExperimentsWithCIFAR, and Net3 from CustomExperimentsWithCIFAR. For each network:

   (a) Training loss curve

   (b) Confusion Matrix

2. **Overall Classification Accuracy** Report the overall classification accuracy for each model in a tabular format (Table 1), where each row corresponds to a network and its respective accuracy score.

3. **Per class classification Accuracy** Construct a 10x3 table (Table 2) summarizing per-class classification accuracy. Each row represents a distinct class, while each column corresponds to one of the three networks, facilitating direct performance comparison.

4. Examine the overall classification accuracy across the three networks. Provide a concise discussion highlighting key findings, such as trends in model performance, the impact of network depth, and any potential trade-offs observed in classification accuracy.

The `run_code_for_testing` function already computes the necessary values for tasks 2-4 (overall classification accuracy, per-class accuracy, and confusion matrix). You have the flexibility to either **post-process its output** to extract and format the required information or modify the testing code through inheritance to generate the desired outputs directly.

By leveraging inheritance, you can customize the evaluation process within a subclass of `ExperimentsWithCIFAR` to automatically log and structure the results in a more accessible format. It is upto the student's choice.

Increasing the network size does not necessarily lead to improved performance. A primary reason for this is the vanishing gradient problem in deep networks, which can impede effective learning. Strategies for mitigating vanishing gradients will be covered in the lecture on February 18th.

Additionally, this homework is designed to enhance your object-oriented programming (OOP) and code reusage skills while interacting with DLStudio. It serves as a foundation for future assignments, which will rely extensively on the use of DLStudio.

3 Submission Instructions

Include a typed report explaining how you solved the given programming tasks. You may refer to the homework solutions posted at the class website for the previous years for examples of how to structure your report

1. **Turn in a PDF file and mark all pages on gradescope**.
2. Submit your code files(s) as zip file.
3. **Code and Output Placement:** Include the output directly next to the corresponding code block in your submission. Avoid placing the code and output in separate sections as this can make it difficult to follow.
4. **Output Requirement:** Ensure that all your code produces outputs and that these outputs are included in the submitted PDF. Submissions without outputs may not receive full credit, even if the code appears correct.
5. For this homework, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert code to `.py` and submit that as source code. **Do NOT submit .ipynb notebooks.**
6. You can resubmit a homework assignment as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.
7. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**