Middle East Technical University | Department of Computer Engineering

CENG 499

Introduction to Machine Learning

Fall 2021-2022

Homework 1 - Artificial Neural Networks

version 1.0

Due date: 30 November 2021, 23:59

1 Introduction

In this assignment, you have a chance to get hands-on experience with ANNs (Artificial Neural Networks). You are going to design various image classifier ANNs and optimize their hyperparameters. Your implementation should be in Python, using PyTorch library. You will also comment on the hyperparameter optimization process in **report.pdf**, whose format is up to you.

2 Dataset

You will be working on CIFAR-10 dataset. The dataset consists of 32 x 32 pixel RGB images, each of which belongs to one of 10 categories. However, we turn those images into grayscale. The following code snippet can be used to download and manipulate the dataset.

```
from torchvision.datasets import CIFAR10
import torchvision.transforms as T

train_transform = T.Compose([
    # can add additional transforms on images
    T.ToTensor(),  # convert images to PyTorch tensors
    T.Grayscale(), # RGB to grayscale
    T.Normalize(mean=(0.5,), std=(0.5,)) # normalization
                                # speeds up the convergence
                                # and improves the accuracy
])

val_transform = test_transform = T.Compose([
    T.ToTensor(),
    T.Grayscale(),
    T.Normalize(mean=(0.5,), std=(0.5,))
])
```

```
train_set = CIFAR10(root='CIFAR10', train=True,
                    transform=train_transform, download=True)

test_set  = CIFAR10(root='CIFAR10', train=False,
                    transform=test_transform, download=True)
```

3 Hyperparameter Optimization

You are going to design and train models with different hyperparamaters. You should pay attention to the following points:

- Although it is not a requirement for this homework, to get consistent results, you may want to check the reproducibility page of PyTorch.
- You are expected to use fully-connected (linear) layers in your models. You can try other type of layers as long as you provide similar hyperparameter optimization procedures in your code/report.
- You are working on a multiclass classification task. Activation function at the output layer should be **softmax** function, and you should use **cross entropy** as your loss function. In PyTorch, there are more than one way to implement softmax + cross entropy combination. For example, **CrossEntropyLoss** includes both softmax and cross entropy loss, which means **you should not put an extra softmax layer before that**.
- Do **NOT** put an activation function after the last fully-connected layer before the softmax operation.
- Since the output shape of the network should match the number of classes, the number of units in the final layer should be 10.
- You are going to do a sanity check before training the network. Compute the accuracy and loss you expect on the test set. Then, compute the real accuracy and loss on the test with an untrained ANN. Are they similar or not? Write the results you got and comment on them in your report.
- You can choose a suitable optimizer. You don't have to spend too much time on it though. Adam optimizer can be a good initial choice.
- Create your validation set from the training set. You can use any method for that purpose such as **torch.utils.data.random_split()**. Describe how you created the validation set in the report.
- You should perform grid search on the following hyperparameters, draw tables for them and select the best ones (you can also experiment on other hyperparameters as well but these are the required ones):

  - Try different number of layers. You should at least try 1-layer, 2-layer, and 3-layer networks. You should also try different number of neurons in each layer; however, you don't have to spend too much time on it.
  - Try different activation functions at the end of each layer.
  - Try different learning rates.

- **Do grid search on the hyperparameters above** and create tables in your report. The values in those tables should be validation accuracy and validation loss. You should report the results of at least 30 different hyperparameter configurations. From all the k-layer networks, select the best performing hyperparameter configurations and draw their training and validation losses on the same graph and comment on them. What countermeasures did you take against overfitting? How may one understand when a network starts to overfit during training? What method did you use to decide where to end the training procedure? Additionally, write the test accuracies in your report for those 3 hyperparameter configurations.

- The hyperparameters you tried should be visible in your codes. Hyperparameter optimization is an important part of this homework; so, **do not** change the code and try values by hand. Write a loop instead.

- You should be able to achieve accuracy around 0.3 with the 1-layer network and 0.5 with the 3-layer network on the test set easily. If your results are far below from these, you may want to check your code again.

- Discuss whether the accuracy is a suitable metric to measure the performance of a network for this specific dataset. Explain your reasons (You don't have to use another metric in your experiments if you can achieve the minimum accuracy required for this homework. Just discuss how using another metric might have changed hyperparameter optimization results).

- Lastly, answer the following questions in your report.

  1. What are the advantages/disadvantages of using a small learning rate?
  2. What are the advantages/disadvantages of using a big learning rate?
  3. What are the advantages/disadvantages of using a small batch size?
  4. What are the advantages/disadvantages of using a big batch size?

4 Where can I train my network?

You can always use your local computer's CPU/GPU to train the models. Be aware that hyperparameter optimization may take too much time even if you

use a GPU since you go over multiple training procedures. It would be wise to start the homework as soon as possible.

You can use lab computers in the department (i.e., inek machines) as they have PyTorch - CUDA. However, please check if there are any processes already running on CPU/GPU of the same computer before running your program so that you will not slow down each other. To check tasks on the GPU, you can use **nvidia-smi** command. Another option could be Google Colab even though it imposes some restrictions on the free users.

5 Specifications

- Your code must be in Python and use PyTorch.
- Include a README file explaining the source code file(s) and how to run the training and testing phases.
- The test set is used only for computing the test accuracy. You cannot use it during the training or hyperparameter optimization.
- Falsifying results, changing the composition of training and test data are strictly forbidden, and you will receive 0 if this is the case. Your programs will be examined to see if you have actually reached the results and if it is working correctly.
- Using any piece of code that is not your own is strictly forbidden and constitutes as cheating. This includes friends, previous homeworks, or the internet. However, example code snippets shared in PyTorch's website can be used. The violators will be punished according to the department regulations.
- Follow the course page on ODTUClass for any updates and clarifications. Please ask your questions in the discussion forum instead of e-mailing.
- You have total of 3 late days for **all** homeworks. The late submission penalty will be calculated using $5d^2$, that is, 1 day late submission will cost you 5 points, 2 days will cost you 20 points, and 3 days will cost you 45 points. No late submission is accepted after reaching a total of 3 late days.

6 Submission

Submission will be done via ODTUClass. You will submit a zip file called **hw1.zip** that contains your **source code**, the **README file**, and **report.pdf**.

7 Resources

Homework1's recitation notebook can be accessed through the link.