

CENG 499

Introduction to Machine Learning

Spring 2020-2021

Homework 1 - Artificial Neural Network version 1

Due date: 25 April 2021, Sunday, 23:59

1 Introduction

For this assignment you are going to implement an ANN (Artificial Neural Network) to classify the clothing classes in the dataset we provided. You are going to implement various architectures and optimize their hyperparameters. The implementation will be done in Python3, using PyTorch library. After getting the results, you are expected to plot the graphs of the best networks and comment on them.

2 Dataset

The dataset you will be working on is created using [Fashion-MNIST Dataset](#) and by adding some color and noise over it. However, the class labels (T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, Ankle boot) are still the same. The dataset can be downloaded through: http://user.ceng.metu.edu.tr/~artun/ceng499/hw1_data.zip.

The dataset has 2 folders: "train" and "test" in which the input images reside. These folders also contain the label files: "labels.txt". For train split, every line in labels.txt contains image name and its label. However, the labels.txt in test folder only contains image names and the labels for the test split are not given. How you are going to calculate accuracy will be discussed in the later sections. We have not specified the validation split. You are going to create your own validation split from the train split.

3 Test Accuracy Calculation

To emphasize that you cannot use test set during training or hyperparameter optimization, we did not provide the labels for test split. However, you still need to report the test accuracy of your best networks. To calculate your accuracy, you can login the website with ids and passwords in provided on ODTUCLASS:

<http://207.154.196.239/>

and submit your prediction in the following format:


```
image_name_1 predicted_label_1
image_name_2 predicted_label_2
...
```

As you can see, the format is very similar to the labels.txt in train split. For hyperparameter optimization and determining best ones, you are going to divide your training data into separate validation and training sets and calculate validation loss/accuracy. **The ranking in the leaderboard in the website does NOT affect the grade you get from this homework.**


4 Networks and Reports

You are going to implement different models and train them using different hyperparameters. 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 network in this homework. However, you can try other type of layers as well if you want to, as long as you provide similar hyperparameter optimization procedures in your code/report.
- You are going to work 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 ways to implement softmax + crossentropy combination; however, read the description of the one you selected. For example, **CrossEntropyLoss** includes both softmax and crossentropy loss which means **you shouldn't put an extra softmax layer before that**.

Another issue with that is separately calculating softmax and crossentropy may lead to numerical issues because of the logarithm. Therefore, instead of applying logarithm separately from softmax, use **log_softmax** (and **nll_loss** afterwards).


 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. After the random initialization of the weights and before the training of the model, calculate the accuracy and loss on some split (for example train split) and write it in your report. Additionally, considering the fact that the dataset is balanced, calculate the loss and accuracy you expect and also put it in your report. Are they similar?
- 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.
- Separate your training set into train and validation sets. You can employ any method here; for example, you can manually cut and paste the files or you can use [torch.utils.data.random_split](#). However, please describe how you did it in your report.

- You need to prevent overfitting. You can do it by using the validation set to calculate the validation loss after every epoch or after some amount of optimization step. Store the best validation loss and after every calculation of the validation loss, update the best validation loss and save the model if the newly calculated validation loss is better than the best validation loss until then. The code of this would be something like this:

```
best_val_loss = float('inf')
for epoch_idx in range(epoch):
    model.train()
    for sample in train_loader:
        # DO TRAINING STUFF
    model.eval()
    with torch.no_grad():
        for sample in val_loader:
            # DO VALIDATION STUFF AND ACCUMULATE VALIDATION LOSS
    if val_loss < best_val_loss:
        # SAVE MODEL
        best_val_loss = val_loss
```

- You can stop your optimization process before its specified epochs using early stopping to save time. For example, you can stop it if your validation loss hasn't improved for some constant number of epochs.
- You should perform grid search on the following hyperparameters, draw tables for them and select the best one (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(0 hidden layers), 2(1 hidden layer), and 3(2 hidden layers) layer networks. You should also try different number of neurons in each hidden layer; however, you don't have to spend too much time on it.
 - Try different activation functions at the end of each layer. You should try at least **sigmoid**, **ReLU**, and **tanh**. You can implement them yourself if you want but they are already implemented in PyTorch. You are free to use them.
 - Try different learning rates. For example, you can divide learning rate by $\sim \sqrt{10}$ and try these: 0.1, 0.03, 0.01, 0.003, 0.001, 0.0003, 0.0001, 0.00003, 0.00001 etc. When the learning rate is too small the validation loss may continue decreasing for many epochs. Therefore, you may want to limit the number of epochs since it can take too much time.
- **Do grid search on the hyperparameters above** and fill the tables in the given report.tex. The values in those tables should be validation accuracy or validation loss. From all the k-layer networks, select the best performing hyperparameters and draw its training and validation losses on the same graph and comment on it. Additionally, write your test accuracy result in your report for this best network.

 The learning rates given above are just suggestions. You may want to choose a smaller set of learning rates to try since hyperparameter optimization may take too much time. However, it is important that you filled the grid you specified.

 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 in test set easily. If your results are far below from these, you may want to check you code again.

5 Where can I train my network?

You can train your networks using your computer on CPU. The dataset is selected to be small in size so that it does not take too much time even you train it on CPU. However, be aware that hyperparameter optimization may take too much time even if you use GPU since you go over multiple training procedures. Therefore, starting your hyperparameter optimization as early as possible would be wise.

If you have access to inek machines (computer engineering department lab computers), PyTorch 1.7.1 cuda version is already installed in python3. You can train your models using GPUs or CPUs of these computers. However, please check whether there are any processes already running on the GPU/CPU of the same computer before running your code so that you won't slow down each other. To check the tasks on GPU, you can use **nvidia-smi** command.

Another option is using Google Colab; however, be aware that there are some usage limitations on it. We are going to mention 2 ways of accessing the dataset in Google Colab. The first one is putting it in your drive. The tutorial in [this link](#) can be beneficial for that. Second option is directly downloading the dataset. Here is a sample code for that:

```
!wget user.ceng.metu.edu.tr/~artun/ceng499/hw1_data.zip
!unzip hw1_data.zip
with open('data/train/labels.txt', 'r') as f:
    for line in f:
        print(line)
```

6 Specifications

- The codes must be in Python3 and must use PyTorch.
- Include a README file explaining how to run the training and testing.
- 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.
- The late policy given in the syllabus applies here. The late submission penalty will be calculated using $5n^2$, 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 6 late days.
- 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. You can use the examples that are provided in PyTorch's website. The violators will be punished according to the department regulations.
- Follow the Announcements page on ODTUCLASS for the updates and clarifications. You can ask your questions on Homework1 in Discussion section of ODTUCLASS instead of e-mailing if the question does not contain code or solution.

7 Submission

Submission will be done via ODTUCLASS. If you do not have access to ODTUCLASS send an email to "artun@ceng.metu.edu.tr" as soon as possible. You will submit a zip file called "hw1.zip" that contains all your source code, the README file, and your report in a pdf format compiled from the given latex file.