# CSE472 (Machine Learning Sessional)

## Assignment 3: Function Approximation with Neural Network and Backpropagation

### Introduction

In this assignment, you will have to implement a Feed-Forward Neural Network (FNN) from scratch and apply your FNN to classify apparel.

### Basic Components of the Network

- Dense Layer: A fully connected layer, defined by the dimensions of its input and output.
- Normalization: Batch Normalization
- Activation: ReLU
- Regularization: Dropout
- Optimization: Adaptive Moment Estimation (Adam)
- Regression: Softmax for Multi-class Classification

use PyTorch

Write separate classes for each of the aforementioned building blocks. **Vectorize your code whenever possible to speed up training and inference**. Modularize your code well, set up the architecture in one place such that it is trivial to change the model architecture later on (**for possible online tasks or retraining**).

You will have to implement the backpropagation algorithm to train the model. The weights will be updated using mini-batch gradient descent with Adam optimization. **No deep learning framework will be allowed for your implementation**. Since the architecture is not fixed, you have to modularize your code in such a way that it works for any architecture that uses the aforementioned modules. To make your implementation efficient, you have to write each operation as matrix multiplication, whenever possible.

For preparing, training and testing your model, write your codes in a jupyter notebook named as <YourRollNo>.ipynb. Clearly mention the separate blocks used for data loading, cleaning, building the architecture, training, validation, testing etc.

## Library Usage

| Purpose | Allowed Packages |
|---|---|
| Read the images | opencv, pillow |
| Dataset loading | torchvision |
| Visualization | matplotlib, seaborn |
| Progress bar | tqdm |
| Data manipulation | numpy, pandas |
| Model saving and loading | Pickle |
| Performance metrics and statistics | scipy, sklearn |

## Dataset Description

We will use the FashionMNIST dataset. You can use the following code snippet to download them:

```
from torchvision import datasets, transforms

# Define transformation
transform = transforms.ToTensor()

# Load the training dataset
train_dataset = datasets.FashionMNIST(root='./data', train=True,
                                      transform=transform, download=True)

# Load test dataset separately
test_dataset = datasets.FashionMNIST(root='./data', train=False,
                                     transform=transform, download=True)
```

The key details for the dataset include:

- Number of Samples: 70,000 images
    - Training data size: 60,000
    - Testing data size: 10,000
- Image Size: 28 x 28 pixels
- Color Channel: Grayscale (1-channel)

- *Labels/Classes:*

| **Label** | **Class** |
|---|---|
| 0 | T-shirt/top |
| 1 | Trouser |
| 2 | Pullover |
| 3 | Dress |
| 4 | Coat |
| 5 | Sandal |
| 6 | Shirt |
| 7 | Sneaker |
| 8 | Bag |
| 9 | Ankle boot |

## Preservation of Your Trained Model/Model Weights

After training and finalizing your model, you must save your final model in pickle (see: https://docs.python.org/3/library/pickle.html). If saving the entire model takes a large amount of space, you may save only the weights of the model in pickle and keep the final architecture in your test file. You should write necessary codes in your testing block that can load the pickle file of your trained model and use it to predict labels for query images (i.e., the images for which classification needs to be done). The testing dataset will also contain 28x28 images and it will be provided during evaluation. Do NOT share your pickle file with others.

## Report Writing

1. You must provide clear instructions on how to run your codes in the report.
2. You have to report the training loss, validation loss, training accuracy, validation accuracy and validation macro-f1 scores for each full pass over the training set.
3. Prepare graphs for four different learning rates and for three different models.
4. Report the confusion matrix for each such model.
5. Make sure you tune the learning rate (start from 0.005 and decrease). Select the best model using validation set's macro-f1 and nicely report, perhaps with some color highlighting, the values of the above-mentioned scores. Try to have a validation macro-f1 of 0.75 or more.
6. Finally, for the best (chosen) model, report the independent test performance.

## Thrive for Good Results

You should train hard to get the best results, sky's the limit. Do not overfit, however. During online, a separate independent test set will be used to measure the performance of your model. **The top three performing models in each section will be duly recognized.**

## Marking Rubrics

| | |
|---|---|
| Implementation of the Dense Layer | 35% |
| Batch Normalization on mini-batches | 5% |
| Incorporating the Activation | 5% |
| Proper handling of Dropout in forward/backward passes | 10% |
| Estimation of moments and updates with Adam | 25% |
| Classifying with Softmax | 15% |
| Code clarity and proper submission | 5% |

## Submission Format

1905xyz
|-- 1905xyz.ipynb
|-- model_1905xyz.pickle [only for the best performing model]
|-- report_1905xyz.pdf

Zip the folder and rename it to **[Student_ID].zip**

**Deadline: 25-Oct-2024 (Friday), 10:00 PM.**

## Warning

1. Don't copy! We regularly use copy checkers.
2. First time wrongdoers (either copier or the provider) will receive **negative** marking because of dishonesty.
3. Repeated occurrences of copying will lead to severe departmental action and jeopardize your academic career. We expect fairness and honesty from you. Don't disappoint us!