CSC2042S

Assignment 4: Neural Networks

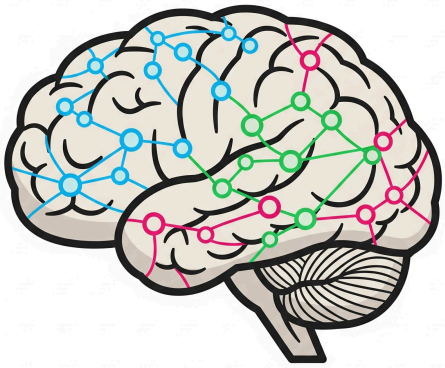
Total marks: 30

Assignment goal: Optimisation of a Neural Network for Image Classification

---

In this assignment, you will be required to build a neural network, implement the data processing and training process, including backpropagation, and experimentally discover how hyperparameters impact performance. You will then be required to analyse the results obtained.

Dataset: Fashion-MNIST - A collection of 70,000 grayscale images of clothing items, perfect for testing your models. https://www.kaggle.com/datasets/zalando-research/fashionmnist



Tasks

1. **Data processing [2 Marks]**
    - Import necessary libraries, such as `torch`, `torch.nn`, `torchvision.transforms`, `matplotlib.pyplot`, etc.

    - Load the Fashion_mNIST dataset into `DataLoader` objects for training and validation. Load the dataset and transform the dataset (convert to tensors, normalise).

2. **Building and training a Baseline model [8 Marks]**

- Before we can optimise a model, we need a baseline to compare against. We will build a simple, standard neural network and train it with default settings.

- Define the Network Architecture [3 Marks]: using a PyTorch class for your neural network that inherits from `nn.Module`. Your model should be a simple Multi-Layer Perceptron (MLP) with the following architecture:
    - Input Layer (a flattening layer).
    - Hidden Layer 1: 128 neurons, ReLU activation.
    - Hidden Layer 2: 64 neurons, ReLU activation.
    - Output Layer: 10 neurons (one for each class).

- Training Loop [2 Marks]: create a `train_model` function. You will need to fill in the essential steps of the training process within the loop. This is where you will implement the logic for the forward pass, loss calculation, backpropagation, and weight updates.

- Train the Baseline Model [3 Marks]: Create an instance of your model, define a loss function (`nn.CrossEntropyLoss`), and an optimiser (`torch.optim.Adam` with a default learning rate of 0.001). Train your model for 10 epochs and record its final accuracy on the validation set. This is your baseline accuracy.

3. **Hyperparameter Optimisation Experiment [8 Marks]**

- Remember the definition of hyperparameters - they are external configuration variables set before training begins, controlling the model's structure and the learning process itself. The goal is to find the set of values that beats the baseline accuracy.

- Design and run your experiment. Choose **three** of the following hyperparameters to investigate: [2 Marks for each hyperparameter selected]
    - Learning Process:
        - Learning rate (eg, [0.1, 0.01, 0.001]).
        - Optimise (eg, `torch.optim.Adam`, `torch.optim.SGD`, `torch.optim.RMSprop`)

    - Regularisation/Architecture:
        - Activation function (eg, Relu, Sigmoid).
        - Number of neurons in the hidden layer one.
        - Dropout Rate (Add a `nn.Dropout(p=...)` layer after each hidden layer and tune the probability `p` (eg, [0.1, 0.25, 0.5]).

- Write a script (e.g., using nested loops) [2 Marks]: that systematically trains and evaluates a new model for each combination of the hyperparameters you have chosen. Store the final validation accuracy for each run in a data structure.

4. **Part 4: Analysis and Report [12 Marks]**

In this final section, you will analyse the results of your experiment and report your findings.

- Present your experimental results in a well-formatted table (e.g., a Pandas DataFrame) showing the combination of hyperparameters and the final validation accuracy for each run. [2 Marks]
- Create at least two visualisations to help interpret your results: [4 Marks]

    - Hyperparameter Impact: Create a bar chart or heatmap that clearly compares the performance of your different hyperparameter combinations.
    - Loss Curves. Plot the training and validation loss curves (loss vs. epochs) for your baseline model and your single best-performing model on the same graph. This is crucial for comparing model convergence and overfitting.
    - Confusion Matrix. Generate and display a confusion matrix for your best model's predictions on the validation set. This will help you see which specific clothing items are being confused.
- Interpret your results. [6 Marks]
    - Answer the following questions concisely, using evidence from your table and plots to support your claims:
        - Best vs. Baseline: Which combination of hyperparameters produced the best model? How much did it improve upon the baseline accuracy?
        - Hyperparameter Influence: Which hyperparameter had the most significant impact on performance? Explain why you think certain values worked better than others.
        - Overfitting Analysis: By comparing the loss curves of your baseline and best models, analyse the degree of overfitting. Did your hyperparameter tuning help mitigate overfitting? How can you tell from the plot?
        - Error Analysis: Look at your confusion matrix. Identify two classes that the model frequently confuses. Propose a reason for this confusion by considering the visual characteristics of the items in those classes.

Submission Requirements

Submit a single compressed archive named as your student number, e.g. GWRBRA001.tar.xz. The code should be submitted as a Jupyter notebook with clear documentation. Version control with git is recommended but not required. The code should be able to run with only the provided data and standard libraries such as Pillow, numpy, scikit-learn, matplotlib, and PyTorch (check with a tutor if you are unsure about using other libraries). A README file should include setup instructions and a description of each submitted file. Do not submit any data. Also include a report of at most 5 pages (submitted as a PDF document) that describes your design decisions, reports and discusses your results, including visualisations. Marks will be given primarily

based on showing understanding of the concepts being applied and demonstrating the ability to develop appropriate models and to analyse the results. Tasks 1–4 will be marked based on both your report and the code in your notebook. Please ensure that your tarball works and is not corrupt (you can check this by trying to download your submission and extracting the contents of your tarball - make this a habit!). **Corrupt or non-working tarballs will not be marked - no exceptions. A 10% penalty will be incurred for a submission that is one day late (handed in within 24 hours after the deadline). Any submissions later than 1 day will be given a 0% mark. Academic integrity All code and analysis must be your own work. You may use standard libraries (in line with the specification of what is permitted for the different components of the assignment) and reference documentation, but must cite all sources. You may discuss your work with other students, but code sharing is prohibited. Use of AI tools must be declared and limited to debugging assistance only. AI assistance is not permitted for writing the report.**

Academic Integrity

All code and analysis must be your own work. You may use standard libraries (in line with the specification of what is permitted for the different components of the assignment) and reference documentation, but must cite all sources. You may discuss your work with other students, but code sharing is prohibited. Use of AI tools must be declared and limited to debugging assistance only. AI assistance is not permitted for writing the report.

Marking Rubric

Category	**Marks**
Data Processing	2
Building and training a baseline model: Define the Network Architecture	3
Building and training a baseline model: Training Loop	2
Building and training a baseline model: Train the Baseline Model	3

Hyperparameter Optimisation Experiment (2 Marks for each)	6
Hyperparameter Optimisation Experiment script	2
Two tables on training and validation accuracy	2
Two visualisations to interpret results	4
Result Interpretation	6
Total	**30**