CSC3022F Machine Learning
Assignment 1 - Artificial Neural Networks

Department of Computer Science
University of Cape Town

April, 2025

In this assignment, you are tasked with implementing a feedforward neural network that can classify fashion product categories from the FashionMNIST dataset. You will also be asked to write a report detailing the designs of your neural network. This must be a pdf that will be uploaded to a separate assignment on Amathuba.

ANNs for Image Classification

Details about the FashionMNIST dataset are available here: `https://github.com/zalandoresearch/fashion-mnist`. It is similar to the MNIST handwritten digit dataset, but the labels are fashion product categories (e.g. dress, sneakers) instead of digits. It consists of a training set of 60,000 examples and a test set of 10,000 examples, each of which is a 28x28 grayscale image.

The dataset is available on Amathuba `Resources`. Your code must NOT download the dataset (this will be penalised). Do not submit the dataset (the tutors will have access to it). Your code should work with the assumption that the folder `FashionMNIST` (which contains the subfolder `raw`) will be placed in the same directory as your code. This contains the train and test data in a compressed format, which you can use to train your models and evaluate their performance for the report. Assuming that the given this, you can use the following code to load the dataset:

```
from torchvision import datasets
# Load FashionMNIST from file
DATA_DIR = "."
train = datasets.FashionMNIST(DATA_DIR, train=True, download=False)
test = datasets.FashionMNIST(DATA_DIR, train=False, download=False)
```

You are free and recommended to use Pytorch and Torchvision. This includes all of the provided activation functions, loss functions and the built-in gradient descent and backpropagation functionality. The main goal of this assignment is to get you to design neural networks that can successfully (to some degree of accuracy) classify images into multiple categories.

Your ANN should be implemented in files called `classifier.py` and should be invoked as follows:

```
python classifier.py
```

When a user executes this code, it should build your network, define any necessary functions, preprocess your data (if needed), train your network and, once training has completed, allow a user to enter a path to an image from the FashionMNIST dataset:

```
> python classifier.py
Pytorch Output...
...
Done!

Please enter a filepath:
> ./path/to/dress1.jpg
Classifier: dress

Please enter a filepath:
> ./path/to/trouser.jpg
Classifier: trouser

Please enter a filepath:
> exit
Exiting...
```

IMPORTANT: This part of your program has to work with JPEG (.jpg) files, as opposed to the compressed dataset files used for training and evaluation for the report. Examples of FashionMNIST JPEG images are available on Amathuba `Resources`. More examples are available here – `https://huggingface.co/datasets/visual-layer/fashion-mnist-vl-enriched/viewer/default/train` – in the links listed in the `image_url` column.

Use available Python packages, such as `torchvision` or `PIL Image`[1], for reading in grayscale JPEG images. During training your images are represented as 28x28

[1]`https://pillow.readthedocs.io/en/stable/reference/Image.html`

grayscale matrices, so make sure you do the same when loading JPEGs, as demonstrated in the following code example:

```
# Load a grayscale image
img = torchvision.io.read_image(jpg_path, mode=torchvision.io.ImageReadMode.GRAY)
img = img.squeeze() # Convert from (1,28,28) tensor to (28,28) tensor
```

Given that FashionMNIST is a commonly solved problem in Machine Learning, a number of valid solutions exist. Many of your marks will come from your report (max 5 pages excluding references) where you will need to describe (with sufficient reasoning / motivation):

- The topology of the networks investigated (including the number of layers, activation functions and connectedness of said layers).
- Any preprocessing steps performed on the training set.
- The loss function(s) used.
- Any optimizer(s) that were used.
- How the networks are trained and validated.
- How you determined which neural network architecture and hyperparameter values performed the best and why that was the case.
- Any other miscellaneous details that you feel are relevant to your ANNs.

It is worth noting that simply saying you used a ReLU activation function or that your loss function was the Cross Entropy Loss function will not get you marks. You must motivate all of your design decisions and demonstrate that you understand your own solutions. You are allowed to take inspiration from other works such as research papers, blog posts, videos, etc. but, be aware that we will be running thorough plagiarism checks on these reports and you will receive a 0 if caught and it may come with a blotch on your academic record.

Mark allocation

Total: (30)

- Data processing (4)

- Neural network implementation (6)
- Training implementation (6)
- Evaluation implementation (4)
- Report (10)

Submission Guidelines

In a compressed archive named as your student number
e.g. GWRBRA001.zip/tar.gz/tar.xz,
place your Python scripts, makefile, Git repo, readme, and any other applicable files. **Also include a text file titled log.txt with output from a previous training run of your model, as printed out by your code when executing. This should include (at least) the training loss of the model after each epoch (see lab 2 notebook for example output) and the final accuracy of your model on the test set when you ran your python script.**

Upload the archive and your report to their respective assignments on Amathuba before 10h00 AM, 2 May, 2025.

Important Notes for All ML Assignments

1. You are not allowed to use LLMs to generate code, suggest hyperparameters or write the report. All sources should be cited in the report.
2. You must use version control from the get-go. A 10% penalty will apply should you fail to include a local repository in your submission.

   With regards to git usage, please note the following:

   -10% - usage of git is absent. This refers to both the absence of a git repo and undeniable evidence that the student used git as a last minute attempt to avoid being penalized.

   -5% - Commit messages are meaningless or lack descriptive clarity. eg: "First", "Second", "Histogram" and "fixed bug" are examples of bad commit messages. A student who is found to have violated this requirement for numerous commits will receive this penalty.

   -5% - frequency of commits. Git practices advocate for frequent commits that are small in scope. Students should ideally be committing their work after a single feature has been added, removed or modified. Tutors will look at the contents of each commit to determine whether this penalty is applicable. A student who commits seemingly unrelated work in large batches on two or more occasions will receive this penalty.

   Please note that all of the git related penalties are cumulative and are capped at -10% (ie: You may not receive more than -10% for git related penalties).

   We cannot provide a definitive number of commits that determine whether or not your git usage is appropriate. It is entirely solution dependent and needs to be accessed on an individual level. All we are looking for is that a student has actually taken the time to think about what actually constitutes a feature in the context of their solution and applied git best practices accordingly.

3. You must provide a README file explaining what each file submitted does and how it fits into the program as a whole.

4. Do not hand in any binary files. Do not add binaries (.o files and your executable) to your local repository.

5. Please ensure that your tarball works and is not corrupt (you can check this by trying to downloading your submission and extracting the contents of your tarball - make this a habit!). Corrupt or non-working tarballs will not be marked - no exceptions. Corrupt or non-working tarballs will not be marked - no exceptions.

6. **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.**

7. **DO NOT COPY**. All code submitted must be your own. Copying is punishable by 0 and can cause a blotch on your academic record. Scripts will be used to check that code submitted is unique.

8. You are NOT allowed to simply download python packages to solve your coding problems. Unless specifically stated in the Assignment brief, we expect all of your code to be your own.