Machine Learning Practical 2025/26: Coursework 1

Released: Monday 13 October 2025

Submission due: 12:00 Friday 24 October 2025

This coursework is formative (not for mark), and contains exactly the same questions as Coursework 2 will. Working on Coursework 1 is effectively working for Coursework 2, and is intended as a way of spreading the work and receiving early feedback on your ongoing work for Coursework 2.

Parts of the work are addressed later in the course Labs and Lectures. Specifically:

- You will work on L1 and L2 regularization in Lab 5, and Dropout in Lab 6 (with solutions for these mlp layers released with the next Lab, Labs 6 and 7 respectively, as normal). It is recommended that you work on these before checking the Lab solutions.
- Question 9 on the Custom Activation Function also requires knowledge from Lecture 6 to be addressed fully.

The recommended workload is for 5 hours for Coursework 1, and 10 hours for Coursework 2.

1 Introduction

The aim of this coursework is to study the classification of images of handwritten digits using neural networks. The first part of this coursework will concern the identification and discussion of a *fundamental problem* in machine learning, as shown in Figure 1. Following this preliminary discussion, you will further investigate this problem in *wider* and *deeper* neural networks, study it in terms of network width and depth. The second part involves implementing different methods to combat the problem identified in Task 1 and then comparing these methods empirically and theoretically. In the final part, you will present your conclusions and contextualise them in the literature.

The coursework will use an extended version of the MNIST database, the EMNIST Balanced dataset, described in Section 2. Section 3 describes the additional code provided for the coursework (in branch `mlp2025-26/coursework1` of the MLP github), and Section 4 describes how the coursework is structured into three tasks. The main deliverable of this coursework is a report, discussed in section 8, using a template that is available on the github. Section 9 discusses the details of carrying out and submitting the coursework, and the marking scheme is discussed in Section 10.

You will need to submit your completed report as a PDF file and your local version of the `mlp` code including any changes you made to the provided (`.py` files). The detailed submission instructions are given in Section 9.2 – please follow these instructions carefully.

2 EMNIST dataset

In this coursework we shall use the EMNIST (Extended MNIST) Balanced dataset [Cohen et al., 2017], https://www.nist.gov/itl/iad/image-group/emnist-dataset. EMNIST extends the well-known MNIST by including images of handwritten letters (upper and lower case) as well as handwritten digits. Both EMNIST and MNIST are extracted from the same underlying dataset, referred to as NIST Special Database 19. Both use the same conversion process resulting in centred images of dimension 28×28.

There are 62 potential classes for EMNIST (10 digits, 26 lower case letters, and 26 upper case letters). However, we shall use a reduced label set of 47 different labels. This is because (following the data conversion process) there are 15 letters for which it is confusing to discriminate between upper-case and lower-case versions. In the

47 label set, upper- and lower-case labels are merged for the following letters:
`C, I, J, K, L, M, O, P, S, U, V, W, X, Y, Z`.

The training set for Balanced EMNIST has about twice the number of examples as the MNIST training set, thus you should expect the run-time of your experiments to be about twice as long. The expected accuracy rates are lower for EMNIST than for MNIST (as EMNIST has more classes, and more confusable examples), and differences in accuracy between different systems should be larger. Cohen et al. [2017] present some baseline results for EMNIST.

You do *not* need to directly download the EMNIST database from the nist.gov website, as it is part of the `coursework1` branch in the `mlpractical` Github repository, discussed in Section 3 below.

3 Github branch `mlp2025-26/coursework1`

You should run all of the experiments for the coursework inside the (mini-)Conda environment you set up for the labs. The code for the coursework is available on the course Github repository on a branch `mlp2025-26/coursework1`. To create a local working copy of this branch in your local repository you need to do the following.

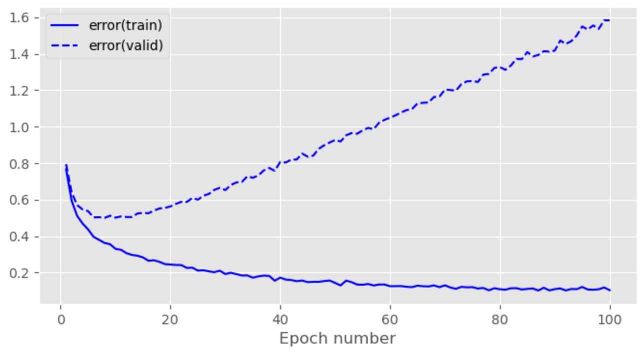
1. Make sure all modified files on the branch you are currently have been committed (see notes/getting-started-in-a-lab.md if you are unsure how to do this).
2. Fetch changes to the upstream `origin` repository by running
   `git fetch origin`
3. Checkout a new local branch from the fetched branch using
   `git checkout -b coursework1 origin/mlp2025-26/coursework1`

You will now have a new branch in your local repository with all the code necessary for the coursework in it.

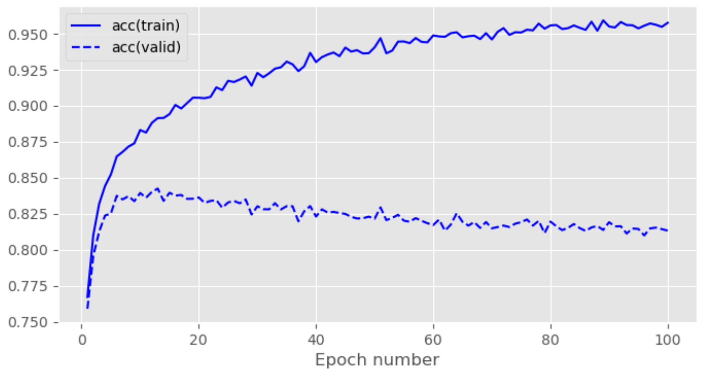
This branch includes the following additions to your setup:

- A new `EMNISTDataProvider` class in the `mlp.data_providers` module. This class makes some changes to the `MNISTDataProvider` class, linking to the `EMNIST Balanced` data, and setting the number of classes to 47.
- Training, validation, and test sets for the `EMNIST Balanced` dataset that you will use in this coursework.
- In order to implement label smoothing, fill in `label_smoothing` function in `EMNISTDataProvider` class and identify the input argument to call this function.
- In order to further improve performance and mitigate the problem identified in neural networks, you will also need to implement a new class in the `mlp.layers` module:
  `DropoutLayer`
  and also two weight penalty techniques in the `mlp.penalties` module:
  `L1Penalty and L2Penalty`.
- In order to further explore regularisation techniques, you are asked to propose and implement your own combination of L1 and L2 regularisation. For this, you will also need to to implement a new weight penalty tecnique in the `mlp.penalties` module:
  `L1L2MixPenalty`.
- The `Coursework_1.ipynb` file with experiment setup, including a cell with "Problematic Training", meant to be run for Question 9, in the exploration of a proposed `CustomActivationLayer` which itself can be located in the `mlp.layers` module.
- `DropoutandPenalty_tests.ipynb` Jupyter notebook
  to be used for testing the implementations of `DropoutLayer`, `L1Penalty and L2Penalty` classes. The tests serve as a safeguard to prevent experimentation with faulty code which might lead to wrong conclusions. Tests in general are a *vital* ingredient for good software development, and especially important for building correct and efficient deep learning systems.

  Please note that passing these preliminary tests *does not* necessarily mean your classes are absolutely bug-free. If you get unexpected curves during model training, re-check your implementation of the classes. There is no test for the label smoothing function.
- A directory called `report` which contains the LaTeX template and style files for your report. You should copy all these files into the directory which will contain your report.



(a) Error curve on the training and validation set of EMNIST dataset.



(b) Accuracy curve on the training and validation set of EMNIST dataset.

Figure 1: Error and Accuracy curves for a baseline model on EMNIST Dataset.

4 Tasks

The coursework is structured into 3 tasks, the first two are supported by experiments on the `EMNIST` dataset.

1. Identification of a fundamental problem in machine learning as shown in Fig 1 and setting up a **baseline** system on `EMNIST` by a valid hyper-parameter search.
2. A research investigation and analysis into whether using Dropout and/or Weight Penalty (`L1Penalty` and `L2Penalty`) or label smoothing addresses the problem found in training machine learning models (Fig 1), including your proposed combination of L1 and L2. How do these approaches improve/degrade the model's performance? Furthermore, this task includes an evaluation of a model using a Custom Activation Function and whether its performance problems relate to the same problem addressed with the previous regularisation methods, or something else.
3. Summarise and conclude the report, relating your conclusions to the overall literature.

5 Task 1: Problem identification

Figure 1 shows the training and validation error curves in Figure 1a and also training and validation accuracies in Figure 1b for a model with 2 hidden layers[1] with ReLU trained on the EMNIST dataset by using cross-entropy error function. This curve can be re-produced by running the model settings defined in the `Coursework1.ipynb` notebook in the github repository. We first identify and discuss the problem shown by the curves in Figure 1 as overfitting, and briefly discuss potential solutions in this section for overcoming this problem.

[1]*All layers are hidden layer except the output one.*

Varying number of hidden units. Initially you will train various 1-hidden layer networks by using either 32, 64 and 128 ReLU hidden units per layer on `EMNIST`. Note that 1-hidden layer network contains two layers, one mapping input units to hidden units and another one mapping hidden units to output units. 2 and 3-hidden layer networks would contain 3 and 4 layers respectively. Make sure you use Adam optimiser with the hyperparameters provided in the template and train each network for 100 epochs. Visualise and discuss how increasing number of hidden units affects the validation performance and whether it worsens or mitigates the overfitting problem.

Varying number of layers. Here you will train various neural networks by using either 1, 2, 3 hidden layers with 128 ReLU hidden units per layer on `EMNIST`. Make sure that you use Adam optimiser with the hyperparameters provided in the template and train each network for 100 epochs. Visualise and discuss how increasing number of layers affects the validation performance and whether it worsens or mitigates the overfitting problem.

The questions in (`mlp-cw1-questions.tex`) that you must answer and count for this task are:

- Question 1;
- Question Table 1;
- Question Figure 2;
- Question 2;
- Question 3;
- Question Table 2;
- Question Figure 3;
- Question 4; and
- Question 5.

(20 Marks)

6 Task 2: Mitigating the problem with regularization

Definition and Motivation. We provide the analysis and explanation for Dropout, L1Penalty, L2Penalty, and label smoothing in the template document, along with relevant citations. Alongside the code, you will also need to provide and present an approach for combining L1 and L2 regularisation in one loss function.

Implementing Dropout, Weight Penalty, Label Smoothing. Here you will implement `DropoutLayer`, `L1Penalty` and `L2Penalty` and test their correctness. You will also implement your proposed approach that combines the L1 and L2 penalties into one loss function. Here are the steps to follow:

1. Implement the Dropout class in the `DropoutLayer` of the `mlp.layers` module. You need to implement `fprop` and `bprop` methods for this class. Please note that the solution uses the original dropout formulation (i.e. scale the hidden unit activations by inclusion probability p in the final network for compensating missing units). The sample distribution to be used for Dropout implementation is numpy's uniform distribution, U(0,1) to pass the unit tests.

2. Implement the L1Penalty and L2Penalty class in the `L1Penalty` and `L2Penalty` of the `mlp.penalties` module. You need to implement `__call__` and `grad` methods for this class. After defining these functions, they can be provided as a parameter, `weights_penalty, biases_penalty` in the `AffineLayer` class while creating the multi-layer neural network.

3. Verify the correctness of your implementation using the supplied unit tests in `DropoutandPenalty_tests.ipynb`

4. Automatically create test outputs `xxxxxx_regularization_test_pack.npy`, by running the provided program `scripts/generate_regularization_layer_test_outputs.py` which uses your code for the previously mentioned layers to run your fprop, bprop, `__call__` and `grad` methods where necessary for each layer on a unique test vector generated using your student ID number.

   To do this part simply go to the scripts folder `scripts/` and then run `python generate_regularization_layer_test_outputs.py --student_id Sxxxxxx` replacing the 'student id' with your student number. A file called `xxxxxxx_regularization_test_pack.npy` will be generated under data which you need to submit with your report.

5. To implement the label smoothing function, fill in `label_smoothing` function in `EMNISTDataProvider` class of `data_providers.py` and identify when this function is called. This function must replace the hard target 0 labels and 1 labels with $\frac{\alpha}{K-1}$ and $1-\alpha$ respectively. α must be set to 0.1.

$$\begin{cases} \frac{\alpha}{K-1}, & \text{if} \quad t_k = 0 \\ 1-\alpha, & \text{if} \quad t_k = 1 \end{cases}$$

6. Implement your proposed L1 and L2 combined weight penalty, as the `L1L2MixPenalty` in the `mlp.penalties` module.

Alongside the code, the question in (`mlp-cw1-questions.tex`) that you must answer and counts for this task is:

- Question 6.

(30 Marks)

EMNIST Experiments. In this section you should modify your baseline network to one that uses `DropoutLayer`, `L1Penalty`, or `L2Penalty` and train a model for each case. You will also run one experiment with the baseline setup plus label smoothing, using no other regularization techniques. For the experiments, your baseline network should contain 3 hidden layers and 128 hidden units with ReLU activation function. You should use the Adam optimiser with a learning rate of 10^{-4} as specified in the template.

Your main aim is to i) investigate whether/how each of these functions addresses the above mentioned problem, ii) study the generalisation performance of your network when used with one of these functions, iii) discover the best possible network configuration, when the only available options to choose from are Dropout and Weight Penalty functions and the hyper-parameters (Dropout inclusion probability and penalty coefficient for the Weight Penalty functions). You should use weight penalty on both weights and biases of your layers. Otherwise, unless explicitly specified, you can leave classes' arguments to their default values.

The Dropout inclusion probability is a float value in the range (0,1), *e.g.* 0.5, chosen manually. Penalty coefficient is also a manually selected float value, *e.g.* 0.001, usually in the range of 0.1 – 0.00001 . **For model selection, you should use validation performance to pick the best model and finally report test performance of the best model.**

Ensure that you thoroughly describe how these functions affect performance when used with different hyperparameters in your report, ideally both at the theoretical and empirical level. When running such experiments, the expected amount of work is not a brute-force exploration of all possible variations of network configurations and hyperparameters, but a carefully designed set of experiments that provides meaningful analysis and insights. We have prespecified for what hyperparameter values you should run each individual experiment for L1/L2 regularisation and Dropout on Table 3 of the template. You should not rerun the experiments for which we provide results, but you will have to run a new experiment to get test results for the best performing model.

You will have to identify and argue for a set of 4 different hyperparameter combinations for which you will run the combined L1 and L2 experiments (for your implementation). (The number 4 was not picked because there are for example 4 obvious combinations to pick or because one could not arguably run more. Rather, this is to simulate computational limitations, constraining your options and limiting the amount of time put into this. There are many valid combinations of experiments to try, but you should motivate your specific selection.)

You should run the label smoothing experiment without other regularization techniques.

You will finally investigate and explain the performance problems with a provided model setup and experiment run, that uses a custom activation layer.

The questions in (`mlp-cw1-questions.tex`) that you must answer and count for this task are:

- Question Table 3;
- Question Figure 4;
- Question Table 4;
- Question Figure 5;
- Question Figure 6;
- Question 7;
- Question 8; and
- Question 9.

(35 Marks)

7 Task 3: Conclusions

In this section, you will summarise your work, draw conclusions, and relate them to the overall literature.

The question in (`mlp-cw1-questions.tex`) that you must answer and counts for this task is:

- Question 10.

(15 Marks)

8 Report

Your coursework will be primarily assessed based on your submitted report, as well as the code.

The report template is divided into sections, though questions for each task might spread to more than one such section, as described in the tasks above. Please read the template before starting to answer the questions to get a sense of how it all fits together. This understanding will provide context (and in some cases example structure) for your answers.

The directory `coursework1/report` contains the file (`mlp-cw1-questions.tex`) where you will add the answers to the questions, and a template for your report (`mlp-cw1-template.tex`) which you should not edit; the generated pdf file (`mlp-cw1-template.pdf`) is also provided, and you should read this file carefully as it contains some useful information about the required structure and content. The template is written in LaTeX, and you should not edit it. Instead, you will input your solutions by editing the file (`mlp-cw1-questions.tex`).

You should copy the files in the `report` directory to the directory containing the LaTeX file of your report, as `pdflatex` will need to access these files when building the pdf document from the LaTeX source file.

While inputting your answers in (mlp-cw1-questions.tex), the first thing you should do is add your Student Number in place of (SXXXXXX) at the start of the file. Then, answer each question, being careful to only edit the text that appears in the brackets of the commands (\youranswer).

The questions ask you to replace the text in **red**, fill in the tables provided in the template, and replace the figures specified with ones you created from your experiments. There is no specific word-count limit for any question, and you are responsible for identifying the correct level of detail based on the question itself (e.g. "discussion" implies an extensive analysis) and context (document section and surrounding text).

There are **10** TEXT QUESTIONS. Replace the text inside the brackets of the command (\youranswer) with your answer to the question.

There are also **6** "questions" to replace some placeholder FIGURES with your own or to add Figures, and **4** "questions" asking you to either fill in the missing entries in the TABLES provided, or to add a Table.

Note that questions are ordered by the order of appearance of their answers in the text. You should fill in the TABLES and FIGURES before discussing the results presented there.

Also note that, if for some reason you do not manage to produce results for some FIGURES and TABLES, then you can get partial marks by discussing your expectations of the results in the relevant TEXT QUESTIONS (for example Question 3 makes use of Table 1 and Figure 2; you can still discuss your expectations of the results even if you did not manage to produce the results).

Ideally, all figures should be included in your report file as vector graphics files rather than raster files as this will make sure all detail in the plot is visible. Matplotlib supports saving high quality figures in a wide range of common image formats using the savefig function. **You should use savefig rather than copying the screen-resolution raster images outputted in the notebook.** An example of using savefig to save a figure as a PDF file (which can be included as graphics in LaTeX compiled with pdflatex is given below.

```python
import matplotlib.pyplot as plt
import numpy as np
# Generate some example data to plot
x = np.linspace(0., 1., 100)
y1 = np.sin(2. * np.pi * x)
y2 = np.cos(2. * np.pi * x)
fig_size = (6, 3)  # Set figure size in inches (width, height)
fig = plt.figure(figsize=fig_size)  # Create a new figure object
ax = fig.add_subplot(1, 1, 1)  # Add a single axes to the figure
# Plot lines giving each a label for the legend and setting line width to 2
ax.plot(x, y1, linewidth=2, label='$y = \sin(2\pi x)$')
ax.plot(x, y2, linewidth=2, label='$y = \cos(2\pi x)$')
# Set the axes labels. Can use LaTeX in labels within $...$ delimiters.
ax.set_xlabel('$x$', fontsize=12)
ax.set_ylabel('$y$', fontsize=12)
ax.grid('on')  # Turn axes grid on
ax.legend(loc='best', fontsize=11)  # Add a legend
fig.tight_layout()  # This minimises whitespace around the axes.
fig.savefig('file-name.pdf') # Save figure to current directory in PDF format
```

NOTE! If you make use of any any books, articles, web pages or other resources you should appropriately cite these in your report. Add the bibtex files for your references to the ref.bib file that comes with your coursework1/report directory.

Tips: When you are mentioning a reference directly, such as that of [Cohen et al., 2017], add it where mentioned. When using references as evidence to support a statement, add it at the end of that statement and before any punctuation. For example: EMNIST contains more samples and classes than MNIST [Cohen et al., 2017].

To create a pdf file `mlp-cw1-template.pdf` from a LaTeX source file (`mlp-cw1-template.tex`), you can run the following in a terminal:

```
pdflatex mlp-cw1-template
bibtex mlp-cw1-template
pdflatex mlp-cw1-template
pdflatex mlp-cw1-template
```

(Yes, you have to run pdflatex multiple times, in order for latex to construct the internal document references.)

An alternative, simpler approach uses the `latexmk` program:

```
latexmk -pdf mlp-cw1-template
```

Another alternative is to use an online LaTeX authoring environment such as https://overleaf.com – note that all staff and students have free access to Overleaf Pro - see https://www.ed.ac.uk/information-services/computing/desktop-personal/software/main-software-deals/other-software/overleaf.

It is worth learning how to use LaTeX effectively, as it is particularly powerful for mathematical and academic writing. There are many tutorials on the web.

9 Mechanics

Marks: This assignment will be assessed out of 100 marks and is **NOT** used for the final grade for the course.

(Coursework 2 will have the same exact Questions and Mark assignments, and that will form 50% of your final grade).

Marking: Coursework 1 will not receive individual feedback. A detailed rubric will be provided for checking your progress after submission, and feedback will be provided across the cohort based on general patterns and issues observed. Piazza, and Office Hours are meant to be used where there there are questions remaining.

Academic conduct: Assessed work is subject to University regulations on academic conduct:
http://web.inf.ed.ac.uk/infweb/admin/policies/academic-misconduct

Submission: You can submit more than once up until the submission deadline. All submissions are timestamped automatically. We will mark the latest submission that comes in before the deadline.

If you submit anything before the deadline, you may not resubmit after the deadline. (This policy allows us to begin marking submissions immediately after the deadline, without having to worry that some may need to be re-marked).

If you do not submit anything before the deadline, you may submit *exactly once* after the deadline up to a maximum of 7 calendar days.

Warning: Unfortunately the submission system on Learn will technically allow you to submit late even if you submitted before the deadline (i.e. it does not enforce the above policy). Don't do this! We will mark the version that we retrieve just after the deadline.

Extension requests: The deadline and late policy for this assignment are specified on Learn in the "Coursework Planner". Guidance on late submissions is at https://informatics.ed.ac.uk/taught-students/all-students/your-studies/late-coursework-extension-requests **Do not email any course staff directly about extension requests**; you must follow the instructions on the web page.

9.1 Backing up your work

It is **strongly recommended** you use some method for backing up your work. Those working in their AFS homespace on DICE will have their work automatically backed up as part of the routine backup of all user homespaces. If you are working on a personal computer you should have your own backup method in place (e.g. saving additional copies to an external drive, syncing to a cloud service or pushing commits to your local Git repository to a private repository on Github). **Loss of work through failure to back up does not constitute a good reason for late submission**.

You may *additionally* wish to keep your coursework under version control in your **local** Git repository **(DO NOT share public instances of your Git branch!)** on the `coursework1` branch.

If you make regular commits of your work on the coursework this will allow you to better keep track of the changes you have made and if necessary revert to previous versions of files and/or restore accidentally deleted work. This is not however required and you should note that keeping your work under version control is a distinct issue from backing up to guard against hard drive failure. If you are working on a personal computer you should still keep an additional back up of your work as described above.

9.2 Submission

Your coursework submission should be done online on the Learn course webpage.

Your submission should include one zip file `Sxxxxxx.zip` that should contain

- Your test outputs `xxxxxx_regularization_test_pack.npy`. which can be generated by implementing the previously mentioned classes, going into `scripts/` and running `python generate_regularization_layer_test_outputs.py --student_id Sxxxxxx` replacing the student id with your student number (NOT your exam id). A file called `xxxxxx_regularization_test_pack.npy` will be generated under data which you need to submit with your report and the code.
- your completed report as a PDF file renamed as `Sxxxxxx_report.pdf`, using the provided template
- your local version of the `mlp` code including any changes you made to the modules (`.py` files) and the `Coursework_1.ipynb` notebook.

Please do not submit anything else (e.g. log files, dataset files).

You can use this command on Linux machines to zip all the files together –

```
zip -r Sxxxxxx.zip mlp/ Coursework_1.ipynb Sxxxxxx_report.pdf
xxxxxx_regularization_test_pack.npy
```

Replace `Sxxxxxx` with your student number.

Please check whether these files are included in the zip file before moving to the next step:

```
unzip -l Sxxxxxx.zip
```

Note that this file must not have your model weights and its size should be few Mbs only.

Once you have successfully created the .zip file and checked its content, you need to login to your Learn `Machine Learning Practical (2025-2026)[YR]` webpage and submit the file.

- From the main Learn page, find the item named **Assessment** and click on it.
- Click on **Submit Coursework 1 Here**.
- A page will appear where you will need to browse and upload your .zip file that you created previously in `Attach Files` (click on the paperclip icon) and then click **Submit**.

You can amend an existing submission by attaching a different .zip file using the Attach Files option and then Submit again.

Note that we will only mark the last uploaded coursework in case you amend your files. Thus it is your responsibility to make sure that correct files are uploaded. Please check that your zip file is not empty or missing files.

10 Marking Guidelines

A note (see also Section 9 on the type of feedback received) that **individual marking will not be provided for Coursework 1**. We provide the below for your reference in relation to Coursework 2, and for use with the rubric that will be released after your Coursework 1 submission.

This document (Section 4 in particular) and the template report (`mlp-cw1-template.pdf`) provide a description of what you are expected to do in this assignment, and how the report should be written and structured.

Assignments will be marked using the scale defined by the **University Common Marking Scheme**:

Numeric mark	Equivalent letter grade	Approximate meaning
< 40	F	fail
40-49	D	poor
50-59	C	acceptable
60-69	B	good
70-79	A3	very good/distinction
80-100	A1, A2	excellent/outstanding/high distinction

Please note the University specifications for marks above 70:

A1 90-100 Often faultless. The work is well beyond what is expected for the level of study.

A2 80-89 A truly professional piece of scholarship, often with an absence of errors.
As ‘A3’ but shows (depending upon the item of assessment): significant personal insight / creativity / originality and / or extra depth and academic maturity in the elements of assessment.

A3 70-79
Knowledge: Comprehensive range of up-to-date material handled in a professional way.
Understanding/handling of key concepts: Shows a command of the subject and current theory.
Focus on the subject: Clear and analytical; fully explores the subject.
Critical analysis and discussion: Shows evidence of serious thought in critically evaluating and integrating the evidenced and ideas. Deals confidently with the complexities and subtleties of the arguments. Shows elements of personal insight / creativity / originality.
Structure: Clear and coherent showing logical, ordered thought.
Presentation: Clear and professional with few, relatively minor flaws. Accurate referencing. Figures and tables well constructed and accurate. Good standard of spelling and grammar.

References

Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017. URL https://arxiv.org/abs/1702.05373.