Deep Learning Assignment 1, 2026

- **"What I cannot create, I do not understand."**
  - Richard Feynman, written on his blackboard in Caltech, 1988

- In this assignment, you will implement a deep neural network from first principles
  - This will greatly deepen your understanding of neural nets!
  - Multi-part assignment, drawing on material covered each week
  - You are expected to start working on it immediately; we will continue to cover relevant material in Topics 2-4
- You must use a Jupyter Notebook:
  - Python recommended but R is OK too
  - You can use low-level functionality of numpy and scikit-learn, but **NOT** their implementations of anything core (for example, not their optimisers, gradient descent implementations, etc.)
  - You will submit the notebook, which must include code, outputs, and text for all parts of the assignment
  - You are strongly recommended to create a **private** GitHub repo and commit updates as you work on them, and give me access
- The deadline will be posted on Canvas.

1-Person or 2-Person Assignment: You Choose

- You can do the assignment alone or you can join with 1 other person
  - Groups larger than 2 are not allowed
  - I recommend doing it with another person rather than alone: two heads are helpful when trying to figure out algorithms and debug code

- If in a group of 2, you must inform the lecturer by email
  - Send an email that CCs both members and includes both names and student IDs (do this when requesting your pair of classes for Task 4 – see later slides)
  - You must send that email at least 1 week before the due date
  - You must work together on all parts: you cannot divide the parts up between the group members
  - Just 1 group member should submit the assignment, with both names and IDs – see later slides.

Academic Integrity

- You are required to act with academic integrity in carrying out your assignments.
- It is **your** responsibility to ensure that you understand the School and University's rules: please refer to Computer Science Academic Integrity as a starting point.
- It's fine to discuss the assignment generally with other students, but you cannot share code with or otherwise collaborate with anyone (except your group partner, if you have notified me that you are doing your assignment in a 2-person group).
- Both group members will be held fully responsible for any academic integrity failings.
- You are **NOT** permitted to use generative AI in this assignment for any purpose, including creating any parts of your code or helping you write part of the text. Nor can you submit code from the web, or get another person to do some/all of the assignment for you.
- Formal academic integrity processes will be adhered to, with a recommendation that group members get 0 for the full assignment, if there is misconduct in **any part** of an assignment. There will be no option for resubmission, as this could be considered to be an unfair time extension not given to other students.

Interviews

- After submission, some students/groups will be interviewed to check on their understanding of their work.
- An inability to explain your work may result in a grade penalty, a zero grade, and/or an academic integrity report.
- Students who discuss their genuine work-in-progress with lecturer or TAs in labs before submission are less likely to be selected for interview.
- Students who submit code that suggests a lack of understanding are more likely to be selected for interview. Submissions which appear to be unrelated to class materials, discussions, code and terminology as used in class and labs, are more likely to be selected for interview.
- The interview time slots will be posted on Canvas. When they are posted, will be requested to book them in your calendar, as when individuals are called to interview there will not be an opportunity to reschedule. Failure to attend interview will result in a zero grade for the assignment.

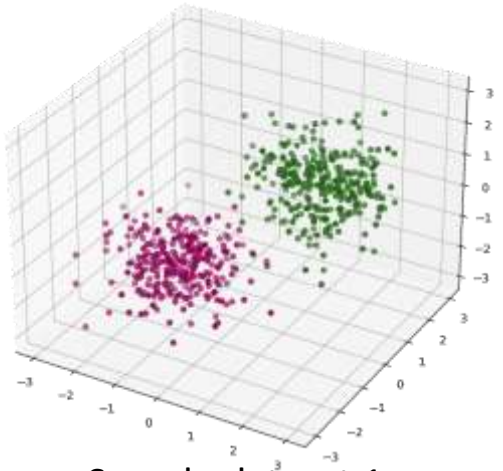
Task 1: Implement Logistic Regression [4 marks]

- Implement Logistic Regression (Topic 2):
  - In your Jupyter Notebook, implement a neural network approach to logistic regression **(no hidden layers, one output node)**
  - Your code should follow the lectures notes to implement the algorithm from scratch
  - Your notebook should include a brief description of the algorithm, relating it to your code, and citations to any external sources that you used in your work
  - Your code must handle **different numbers of input features per training sample and different number of input training samples**, but you don't have to support more than one binary output node

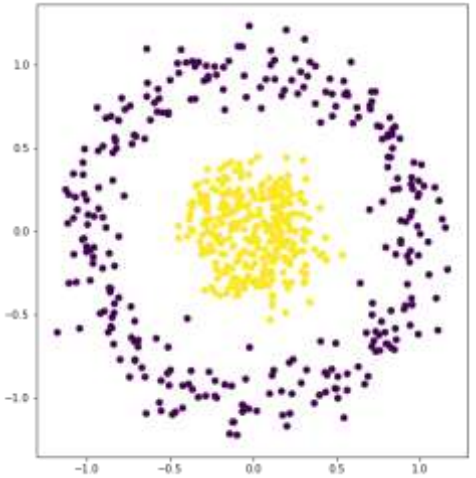
- Post questions on discussion forum if you are unclear about anything, and/or ask questions in the weekly sessions.

Task 2: Testing on Datasets [4 marks]

- I will supply two fairly small datasets:
  - One will be linearly separable (almost or fully), the other will not
  - One will have 2 input attributes, the other will have 3
  - I will provide sample Python code to load and plot the datasets; you **are allowed** to use this code in your own assignment
- Divide each dataset randomly into:
  - Training set (70%): use for main training
  - Validation set (15%): use for tuning, e.g. selecting learning rates
  - Test set (15%): held out set for final performance evaluation
- Train a logistic regressor using your code from Task 1 and observe how it performs on each dataset
- For each dataset, create a scatter plot of your classification results
  - Colour code the data points to represent classes.
  - Include these plots with relevant notations in your notebook.
  - Look at the sample dataset figures as an example.
- Provide notes in your notebook to summarise classification results, and your observations and conclusions.



Sample dataset 1



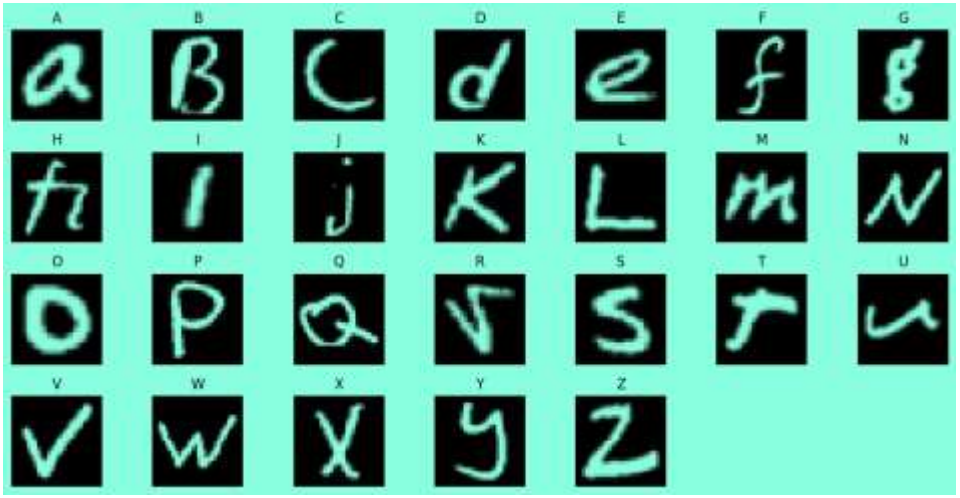
Sample dataset 2

Task 3: Implement and Test a Shallow Neural Network [4 marks]

- Implement a shallow neural network (Topic 3)
  - Start with your logistic regression code
  - Again, follow my notes of Topic 3 as closely as possible
  - As well as your code, include a brief description of the algorithm, and cite any sources you used
  - **Your code just needs to support 1 hidden layer, no more**
  - **Your code just needs to support 1 output node, no more**
  - **Your code must support different numbers of inputs and different number of nodes in the hidden layer**
- Test your implementation on the two small datasets:
  - **Use the earlier dataset division into Training, Testing, Validation.**
  - **Test with different number of nodes in the hidden layer for your datasets**
  - Summarise your observations and conclusions in the notebook – you don't have to include 3D scatter plots for this task.
  - Is it able to handle the linearly separable data? (I hope so)
  - Does it perform better than logistic regression on the second dataset?
  - Does changing the number of nodes in the hidden layer help?

Task 4: Tackle a Challenging Task [4 marks]

- You will tackle a handwritten letter recognition task
  - On Canvas, you can find a dataset of handwritten letters (EMNIST-Letters), and code to read and plot it
  - 26 classes (1 per letter, uppercase and lowercase mixed)
  - Each image is 28x28 pixels, originally 8-bit grayscale (0-255), but I have rescaled into range 0-1 for you
  - 3300 images per class: you should select the images for your task and then divide into training, validation and testing



- You must try to distinguish between **2 classes only:**
  - **Send me an email message to request your pair of classes;** don't just pick a pair of classes yourself
  - If doing it in a group, send one email and CC your collaborator
  - See my sample code for how to extract your two classes
- Use your same neural network implementation from Tasks 2 & 3
  - 1 output (probability of Class 1 of the two classes you are given)
  - Number of inputs will depend on image size, and you will definitely need a larger number of hidden nodes than Task 3
  - As part of your work, plot your training curve and report your classification results.
  - NOTE: Since this is a small dataset and one hidden layer only, your classification accuracy might not be very high.

Task 5: Deep Learning Enhancements [4 marks]

- Update your code to support an arbitrary number of layers with an arbitrary different number of nodes in each layer, and a method to avoid overfitting
  - Since this is a small dataset, include a method of your choice to avoid overfitting out of the methods discussed during lectures
  - In your notebook, include sections that briefly describe how you extend the code to support more layers, and your method to avoid overfitting, with references
- Evaluate the performance of your deep network
  - Test its performance on the dataset from Task 4
  - You can try with different numbers of layers and nodes to see which combination gives good accuracy.
  - Plot training curve(s) and report classification accuracies, comparing how your modifications regarding extra layers and (with/without) overfitting perform relative to the previous 1 hidden layer NN on the same dataset.

Assignment Submission

- You must submit your assignment in TWO formats:
  1. The Notebook itself, including all output from running it
  2. A PDF print of the Notebook (easier for us to grade)

  Note: avoid unnecessary outputs in your notebook; for example, don't print out all rows of the training data file, or every single training iteration
- Include separate sections in your notebook for each of Tasks 1-5
  - All tasks will earn you marks, so it is important that you don't delete or over-write your code from earlier tasks when you move on to later tasks.
  - Instead, copy and modify the code from earlier tasks so that we can see all tasks and grade each one
- For a 2-person group, one person must submit the assignment, which must have both group members' names and IDs on it.
- **The deadline will be posted on Canvas.**