2. Neural Computing (NECO) individual project (70 marks) with the submission of a research paper, Python code, short video presentation and supplementary materials (as detailed below).

On completing this assessment you should be able to:

1. Understand, describe and specify a Neural Network or SVM solution to a data analysis problem;
2. Adjust model parameters and explain how the model seeks to solve the problem;
3. Apply, compare and contrast, and critically evaluate **two** INM427 NECO models.

The subject of the above learning outcomes follows the lectures and associated materials, lab-based tutorials and exercises. The models to be developed by you should be based on the INM427 NECO methods discussed in the lecture notes, which includes Neural Networks and SVMs.

This assessment is worth 100% of the module mark and it is a combination of group and individual work. To pass the module you need to obtain 50 marks or more.

General Guidelines

Our objective is to give you some experience carrying out a piece of research in neural computing. Seek to describe your work clearly and relate it to existing work. You will need to write or change code, run it on your choice of data, read a few background papers, compare and contrast results, and write a paper describing your work, the algorithms you used and the results you've obtained with the use of suitable figures and tables. Check prior to submission that your models run successfully in the City, University of London, lab build machines.

Specific Requirements

Select the data and NECO algorithms wisely to make your task interesting and yet feasible within the time available.

Individual Project: You are asked to evaluate critically, compare and contrast **two** NECO methods applied to your choice of data set using Python and Jupyter notebook. NumPy, PyTorch, Scikit Learn and Skorch can be used. **Tensorflow and Keras are not allowed**.

Teamwork is not allowed in this part of the assessment. **The use of Matlab is not allowed in this part of the assessment**.

You are encouraged to continue to use the same data set from the group work. If you decide to use a new data set, check that it is a suitable choice.

The results from the group work can be used as a baseline for the individual project, but you should not copy large parts of the group work submission into the individual project.

If you choose to use a multilayer perceptron and backpropagation as one of your methods, you are expected to consider now the use of multiple hidden layers, momentum and other regularization techniques such as weight decay.

Popular choices of NECO method include the simple perceptron, multilayer perceptron, convolutional neural networks, SVMs and recurrent neural networks trained with backpropagation through time.

Submit on Moodle:

1. All the code and test data set (as a single zip file) required to run your two best-trained models (one for each INM427 NECO method) and produce at least two of the figures appearing in your paper (item 2 below). Include in the zip file a readme.txt file with any instructions on how your

models should be run, including library or directory dependencies and required software versions (see **code reproducibility** below). If the test set is too large to be included in the zip file, add a link in the readme.txt file stating where the test set can be downloaded from.

2. A paper (in pdf format, single column, font Arial 11, maximum 6 pages including all figures and references) containing a description of the experiments and comparative evaluations (an example paper has been provided on Moodle), plus any appendices (maximum 2 pages) containing any supplementary materials, including a glossary of the main terms used in the paper and any relevant intermediate results or implementation details not worth including in the main body of the paper. This may include other graphs you have produced during the project, other model architectures and parametrizations that you have considered, relevant implementation choices, issues or errors that prompted you to make changes leading up to the main results. The paper must include at least two figures which graphically illustrate quantitative aspects of your results, such as training/testing error curves, learned parameters, algorithm outputs. The paper may be a comparison of two existing algorithms, or it may propose a new algorithm in which case you still must compare it to one other existing algorithm.

3. A 5-minute video of yourself showcasing your paper and code (as a single .mp4 file). Do not exceed 5 minutes! Start a video call on MS Teams with just yourself and record the meeting. Share your screen to show your paper and to demo your code. Add automatic captioning to the compressed video recording that will be produced at the end of the meeting. Your video should focus on explaining the implementation work: comment on the main blocks of your code, including the training of the models; demonstrate a run of your code on the test set (which we will seek to reproduce). Conclude by summarising your main findings and lessons learned: what went wrong and how you addressed the challenges that you've encountered, the results that you're most proud of, and what you would have done differently.

4. A print-out (as a single pdf file) of your entire code. Make sure to add comments to the entire code describing briefly the purpose of each function. The print-out will be used to check the code for plagiarism. Make sure to indicate clearly any code that is not your own, providing a clear reference to where it was taken from with permission. The use of ChatGPT or other automated code generation is not allowed.

Code reproducibility

We should be able to run your two best trained models (one from each NECO method) on your test set without having to re-train the models. In case we are not able to run the code due to version issues or missing packages you may lose marks.

We expect most submissions to be in the form of a Jupyter notebook (ipynb file). Submit also a HTML version of the .ipynb file showing the saved outputs. If you submit any system python (.py) files, you will need to save the outputs too. If utilising .py files for training, you can create a Jupyter notebook along with its HTML version just for testing the two best models. You should still submit all files and add comments to your .py files which will be marked. If re-using code, this should be shown clearly and with an acknowledgement of the source.

To make sure your test scripts run on a virtual environment with all the necessary packages with the right version numbers, you are encouraged to submit as part of your readme.txt file, a *requirements* block and a *setup_instructions* block, for example:

requirements:

torch==1.0.1.post2
torchvision==0.2.1
tornado==5.1.1
tqdm==4.31.1
...
numpy==1.16.0
opencv-contrib-python==4.0.0.21
pandas==0.24.1

setup_instructions:

1. Extract all the files from the zip file
2. Change directory (cd) to the extracted folder, and save any test data into this root folder
3. Create a new virtualenv and install packages from *requirements*:
   virtualenv Student_env
   cd Student_env
   source bin/activate
   cd -
   pip install -r requirements.txt
4. Copy and paste the following code to open the IPYNB:
      jupyter notebook test.ipynb