COMP0197: Applied Deep Learning

Assessed Component 1 (Individual Coursework) 2024-25

Submission on Moodle

Introduction

This is the first of two assessed coursework. This coursework accounts for 50% of the module with two independent tasks, and for each task, a *task script* needs to be submitted with other supporting files and data. No separate written report is required.

There are hyperlinks in the document for further reference. Throughout this document, various parts of the text are highlighted, for example:

Class names are highlighted for those mandatory classes that should be found in your submitted code.
Function names are highlighted for those mandatory functions that should be found in your submitted code.
Printed messages on terminal when running the task scripts.
Visualisation saved into PNG files with task scripts.
[5]: square brackets indicate marks, with total marks being 100, for 50% of the module assessment.
"filepath.ext": quotation marks indicate the names of files or folders.
commands: commands run on bash or Python terminals, given context.

The aim of the coursework is to develop and assess your ability *a)* to understand the technical and scientific concepts behind deep learning theory and applications, *b)* to research the relevant methodology and implementation details of the topic, and *c)* to develop the numerical algorithms in Python and one of the deep learning libraries TensorFlow and PyTorch. Although the assessment does not place emphasis on coding skills and advanced software development techniques, basic programming knowledge will be taken into account, such as the correct use of NumPy arrays, tensors – as opposed to, for example, unnecessary for-loops, sufficient commenting and consistent code format. Up to [20%] of the relevant marks may be deducted for substandard programming practice.

Generative AI (GenAI) tools can be used in an assistive role in this coursework, as defined in the UCL regulations. It is your responsibility to ensure the code runs in the specified environment, with correct output. If you decide to use these tools, you need to write a statement at the beginning of each submitted Python files, detailing what these tools are and how they are used in assisting completing the coursework. Further UCL guidance can also be found from the above link.

Do NOT use this document for any other purposes or share with others. The coursework remains UCL property as teaching materials. You may be risking breaching intellectual property regulations and/or academic misconduct, if you publish the details of the coursework or distribute this further.

Conda environment and Python packages

No external code (open-source or not) should be used for the purpose of this coursework. No other packages should be used, unless specified and installed within the conda environment below. This will be assessed by running the submitted code on the markers' computers, within a conda environment created

as follows, for either TensorFlow or PyTorch. Make sure your OS is up-to-date to minimise potential compatibility issues.

```
conda create -n comp0197-cw1-pt python=3.12 pip && conda activate comp0197-cw1-pt && pip install torch==2.5.0 torchvision --index-url https://download.pytorch.org/whl/cpu
```

```
conda create -n comp0197-cw1-tf python=3.12 pillow pip && conda activate comp0197-cw1-tf && pip install tensorflow==2.18.0
```

Use one of the two for your coursework and indicate with your submitted folder name, "cw1-tf" or "cw1-pt". Use the command `conda list -n comp0197-cw1-xx` to see the available libraries for this coursework ("xx" is either "tf" or "pt"). You can choose to use either TensorFlow or PyTorch, but NOT both of them in this coursework, as it is designed to have a balanced difficulties from different tasks. [100%] of the relevant marks may be deducted for using external code.

Working directory and task script

Each task should have a task folder, named as "task1" and "task2". A Python task script should be a file named as "task.py", such that the script can be executed on a bash terminal when the task folder is used as the current/working directory, within the conda environment described above:

```
python task.py
```

It is the individual's responsibility to make sure the submitted task scripts can run in the above-specified conda environment. If using data/code available in module tutorials or elsewhere such as open-source repositories, copies or otherwise automated links need to be provided to ensure a standalone executability of the submitted code. Care needs to be taken in correct use of relative paths, as it was found to be one of the most common issues in the past. Jupyter Notebook files are NOT allowed. Up to [100%] of the relevant marks may be deducted if no runnable task script is found.

Printing and visualisation

Summarising and communicating your implementation and quantitative results is being assessed as part of the module learning outcome. Each task specifies relevant information and messages to be printed on terminal, which may contain description, quantitative summary and brief remarks. The printed messages are expected to be concise, accurate and clear.

When the task requires visualising results (usually in the form of image), the code should save the results into a PNG file in the respective working directory. These PNG files should be submitted with the code, although they can be generated by the code as well. Please see examples in the module repository using Pillow. Please note that matplotlib cannot be used in the task scripts but may be a good tool during development. Up to [50%] of the relevant marks maybe deducted if this is not followed.

Design your code

The functions/classes/files/messages highlighted (see Introduction) are expected to be found in your submitted code, along with the task scripts. If not specifically required, you have freedom in designing your own code, for example, data type, variables, additional functions/modules/classes, their input and output, and/or extra results for discussion. These will be assessed for correctness in complementing your work but not for design aspects.

The checklist

This is a list of things that help you to check before submission.

- ✓ The coursework will be submitted as a single “cw1-xx” folder, compressed as a single zip file.
- ✓ Under your “cw1-xx” folder, you should have two subfolders, “task1” and “task2”.
- ✓ The task scripts run without needing any additional files, data or customised paths.
- ✓ All the classes and functions colour-coded in this document can be found in the exact names.
- ✓ Check all the functions/classes have a docstring indicating a brief description of its purpose, together with data type, size and what-it-is, for each input argument and output.
- ✓ Check all Python files with appropriate GenAI usage statement and description.

Task 1 Optimising logistic binary regression models

- Implement a specific logistic model function logistic_fun, that takes input arguments, a weight vector $\mathbf{w}$, a hyperparameter representing the polynomial order M, and an input vector representing D-dimensional variable $\mathbf{x}$, and returns the function value representing a positive class probability y.

$$y = \sigma\left(f^M(\mathbf{x}; \mathbf{w})\right)$$

where $\sigma(\cdot)$ is the sigmoid function and $f^M(\mathbf{x}; \mathbf{w})$ is an M-order polynomial function of the input vector $\mathbf{x}$, a linear function of the weight vector $\mathbf{x}$. This function should include all the possible first- and higher-order polynomial terms including interaction terms, for example, when $M = 2$ and $D = 3$:

$$f^M(\mathbf{x}; \mathbf{w}) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_1^2 + w_5x_2^2 + w_6x_3^2 + w_7x_1x_2 + w_8x_1x_3 + w_9x_2x_3$$

where $\mathbf{x} = [x_1, x_2, x_3]^{\mathrm{T}}$ is and $\mathbf{w} = [w_0, \cdots w_9]^{\mathrm{T}}$. In general, the total number of polynomial terms is $p = \sum_{m=0}^{M} \binom{D+m-1}{m}$. For the purpose of the coursework, you can define the order of these p polynomial terms, as long as it is consistent throughout all the code submitted.

- Implement a loss class MyCrossEntropy, to compute the cross-entropy between predicted class probabilities and targets (ground-truth labels).
- Implement a loss class MyRootMeanSquare, to compute the root-mean-square error between predicted class probabilities and targets.
- Using only TensorFlow/PyTorch, implement a stochastic minibatch gradient descent algorithm for optimising the logistic functions, fit_logistic_sgd, which takes N pairs of $\mathbf{x}$ and target values t as input, with additional input arguments, options to choose and minimise one of the above loss functions, learning rate and minibatch size. This function returns the optimum weight vector $\widehat{\mathbf{w}}$. During training, the function should report the loss periodically throughout optimisation using printed messages (minimum 10 times including beginning and end).
- Implement a task script "task.py", under folder "task1", performing the following: [35]
    - Use logistic_fun to generate a training set (200 input-target pairs) and a test set (100 pairs), with the following underlying weights (model parameters):

      $M = 2, D = 5$ and $\mathbf{w} = \left[(-1)^{\boldsymbol{p}} \cdot \frac{\sqrt{p}}{p}, (-1)^{\boldsymbol{p-1}} \cdot \frac{\sqrt{p-1}}{p}, (-1)^{\boldsymbol{p-2}} \cdot \frac{\sqrt{p-2}}{p}, \ldots, -\frac{\sqrt{3}}{p}, \frac{\sqrt{2}}{p}, -\frac{1}{p}\right]^{\mathrm{T}}$.

      Each training and test input is uniformly sampled $\mathbf{x} = [x_1, \cdots x_D]^{\mathrm{T}} \epsilon [-5.0, 5.0]^D$; corresponding to each input $\mathbf{x}$, the observed binary target $t\epsilon\{0,1\}$ are obtained by first 1) adding a random Gaussian noise (with a zero mean and a standard deviation of 1.0) to the function-generated y, then 2) thresholding with a cutoff of 0.5, such that $t = 1$ if the noise-corrupted $y \geq 0.5$, $t = 0$ otherwise.

    - Use fit_logistic_sgt to compute the optimum weight vector $\widehat{\mathbf{w}}$ using the training set, by minimising the cross-entropy loss. Considering three hyperparameter values, $M\epsilon\{1,2,3\}$, for each M, c compute the predicted target values $\hat{y}$ for all $\mathbf{x}$ in both the training and test sets.
    - Repeat the above experiments, by minimising the root-mean-square error as the loss.
    - Consider what a metric (other than the losses) is appropriate for this classification problem. Justify briefly your choice of metric in a printed comment (no more than 50 words).
    - For each loss, report the metric using printed messages a) on the model prediction and b) on observed training data, both with respect to the underlying "true" classes as ground-truth.

Comment briefly on the difference between the two in a printed message (no more than 100 words).

- Implement a task script "task1a.py", under folder "task1". [15]
  - Experiment how to make M a learnable model parameter and using SGD to optimise this more flexible model.
  - Report, using printed messages, the optimised M value and the associated metric values, on the model prediction.

Task 2 Regularising extreme learning machines

This task uses the same data set in the Image Classification tutorial, and to experiment a different type of neural networks, the extreme learning machines (ELMs). For the purpose of the coursework, the dataset is only split into two, training and test sets, as deemed appropriate.

- Implement a specific ELM, with one hidden convolutional layer with pre-assigned, non-trainable *fixed weights* and one fully-connected layer with *trainable weights*, in a class MyExtremeLearningMachine. The fixed weights of the convolutional layer should be initialised (and thus fixed during training) in its __init__ class function, which should have a hyperparameter to indicate the size of the hidden convolutional layer, in terms of the number of convolutional-layer-generated feature maps. The fully-connected layer projects these feature maps to produce the multiclass class probability vector suitable for the image classification application.
- Implement a class function initialise_fixed_layers in MyExtremeLearningMachine. This function should initialise all the fixed weights in the convolution kernels, by random sampling from a Gaussian distribution with zero mean and a standard deviation. This function should be used in the __init__ function with this standard deviation being an additional hyperparameter.
- Implement a function fit_elm_sgd for optimising the implemented ELMs, using a variant of stochastic (minibatch) gradient descent of your choice.
- Implement the following regularisation methods, for training the ELMs.
  - A data augmentation class MyMixUp, using the mixup algorithm, such that:
    - Inherited from the relevant classes in TensorFlow/PyTorch is recommended but not assessed. You can use open-source code and built-in functions, but appropriate reference is required.
    - The algorithm should be seeded for reproducible results.
    - The MyMixUp algorithm can be applied to images and labels in each training iteration.
    - You can design and configure all the other hyperparameters.
    - Visualise your implementation, by saving to a PNG file "mixup.png", a montage of 16 images with randomly augmented images that are about to be fed into network training.
  - A model ensemble class MyEnsembleELM, combining multiple trained instances of MyExtremeLearningMachine, with individually initialised the fixed weights.
    - The algorithm should be seeded for reproducible results.
    - Consider how to combine the multiple predictions from the trained ELMs.
    - Consider what appropriate ranges for the above two hyperparameters, with warning raised if the provided hyperparameters out of the specified range.
    - You can design and configure all the other hyperparameters.

- Implement a task script "task.py", under folder "task2", completing the following: [35]
  - Consider what is considered a "random guess" in a multiclass classification. Explain briefly how such a random guess is defined and can be tested (no more than 100 words).
  - Experiment the ELM implementation, without the regularisation methods, e.g. by changing the hyperparameters, such that a trained ELM predicts better than random guessing.
  - Experiment the regularisation methods, configure and train three additional models, 1) using MyMixUp, 2) using MyEnsembleELM, and 3) using both MyMixUp and MyEnsembleELM.
  - For marking purpose, save the above trained models and submit your trained models within the task folder[1]. The testing code producing the following results should load these saved models, without requiring running the training again.
  - Consider two metrics (other than the losses) that are appropriate for this image classification application. Justify briefly your metrics in a printed comment (no more than 50 words).
  - For each of the four models, report a summary of the test set performance in terms of the two metrics versus the sampled epochs.
  - Visualise your results from the best model based on the metrics, by saving to a PNG file "result.png", a montage of 36 test images with a printed message clearly indicating the ground-truth and the predicted classes for each, in the same order of the images.
- Implement a task script "task2a.py", under folder "task2". [15]
  - Experiment how to use direct least-square solver, instead of fit_elm_sgd, to optimise the ELM models. Implement a class fit_elm_ls for MyEextremLearningMachine.
  - Compare the speed and performance metrics, between fit_elm_sgd and fit_elm_ls, using the best model configurations (regularisation and hyperparameter values) in the above task.
  - Implement a random hyperparameter search strategy with fit_elm_ls, to find a better performant ELM model over the above "best" model.
  - Report a summary of the test set performance for this new model.
  - Visualise the new results from this model by saving to a PNG file "new_result.png", as in the previous task.

[1] It is expected the model weight checkpoints should not be excessively big to exceed the file upload limit on Moodle. However, if it is the case in your submission, these models can be stored on UCL OneDrive. In this case, you should provide a link and code in the submitted Python script, such that the model can be accessed and downloaded automatically.

For reference only

```
(base) yipenghu@prec-7750:~$ conda list -n comp0197-cw1-pt
# packages in environment at /home/yipenghu/miniconda3/envs/comp0197-cw1-pt:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                        main
_openmp_mutex             5.1                       1_gnu
bzip2                     1.0.8                h5eee18b_6
ca-certificates           2025.2.25            h06a4308_0
expat                     2.6.4                h6a678d5_0
filelock                  3.13.1                   pypi_0    pypi
fsspec                    2024.6.1                 pypi_0    pypi
jinja2                    3.1.4                    pypi_0    pypi
ld_impl_linux-64          2.40                 h12ee557_0
libffi                    3.4.4                h6a678d5_1
libgcc-ng                 11.2.0               h1234567_1
libgomp                   11.2.0               h1234567_1
libstdcxx-ng              11.2.0               h1234567_1
libuuid                   1.41.5               h5eee18b_0
markupsafe                2.1.5                    pypi_0    pypi
mpmath                    1.3.0                    pypi_0    pypi
ncurses                   6.4                  h6a678d5_0
networkx                  3.3                      pypi_0    pypi
numpy                     2.1.2                    pypi_0    pypi
openssl                   3.0.16               h5eee18b_0
pillow                    11.0.0                   pypi_0    pypi
pip                       25.0            py312h06a4308_0
python                    3.12.9               h5148396_0
readline                  8.2                  h5eee18b_0
setuptools                75.8.0          py312h06a4308_0
sqlite                    3.45.3               h5eee18b_0
sympy                     1.13.1                   pypi_0    pypi
tk                        8.6.14               h39e8969_0
torch                     2.5.0+cpu                pypi_0    pypi
torchvision               0.20.0+cpu               pypi_0    pypi
typing-extensions         4.12.2                   pypi_0    pypi
tzdata                    2025a                h04d1e81_0
wheel                     0.45.1          py312h06a4308_0
xz                        5.6.4                h5eee18b_1
zlib                      1.2.13               h5eee18b_1
```

```
(base) yipenghu@prec-7750:~$ conda list -n comp0197-cw1-tf
# packages in environment at /home/yipenghu/miniconda3/envs/comp0197-cw1-tf:
#
# Name                    Version                   Build  Channel
_libgcc_mutex             0.1                         main
_openmp_mutex             5.1                       1_gnu
absl-py                   2.1.0                     pypi_0    pypi
astunparse                1.6.3                     pypi_0    pypi
bzip2                     1.0.8                h5eee18b_6
ca-certificates           2025.2.25            h06a4308_0
certifi                   2025.1.31                 pypi_0    pypi
charset-normalizer        3.4.1                     pypi_0    pypi
expat                     2.6.4                h6a678d5_0
flatbuffers               25.2.10                   pypi_0    pypi
freetype                  2.12.1               h4a9f257_0
gast                      0.6.0                     pypi_0    pypi
google-pasta              0.2.0                     pypi_0    pypi
grpcio                    1.70.0                    pypi_0    pypi
h5py                      3.13.0                    pypi_0    pypi
idna                      3.10                      pypi_0    pypi
jpeg                      9e                   h5eee18b_3
keras                     3.8.0                     pypi_0    pypi
lcms2                     2.16                 hb9589c4_0
ld_impl_linux-64          2.40                 h12ee557_0
lerc                      4.0.0                h6a678d5_0
libclang                  18.1.1                    pypi_0    pypi
libdeflate                1.22                 h5eee18b_0
libffi                    3.4.4                h6a678d5_1
libgcc-ng                 11.2.0               h1234567_1
libgomp                   11.2.0               h1234567_1
libpng                    1.6.39               h5eee18b_0
libstdcxx-ng              11.2.0               h1234567_1
libtiff                   4.5.1                hffd6297_1
libuuid                   1.41.5               h5eee18b_0
libwebp-base              1.3.2                h5eee18b_1
lz4-c                     1.9.4                h6a678d5_1
markdown                  3.7                       pypi_0    pypi
markdown-it-py            3.0.0                     pypi_0    pypi
markupsafe                3.0.2                     pypi_0    pypi
mdurl                     0.1.2                     pypi_0    pypi
ml-dtypes                 0.4.1                     pypi_0    pypi
namex                     0.0.8                     pypi_0    pypi
ncurses                   6.4                  h6a678d5_0
numpy                     2.0.2                     pypi_0    pypi
openjpeg                  2.5.2                he7f1fd0_0
openssl                   3.0.16               h5eee18b_0
opt-einsum                3.4.0                     pypi_0    pypi
optree                    0.14.1                    pypi_0    pypi
packaging                 24.2                      pypi_0    pypi
pillow                    11.1.0           py312hcea889d_0
pip                       25.0             py312h06a4308_0
protobuf                  5.29.3                    pypi_0    pypi
pygments                  2.19.1                    pypi_0    pypi
python                    3.12.9               h5148396_0
readline                  8.2                  h5eee18b_0
requests                  2.32.3                    pypi_0    pypi
rich                      13.9.4                    pypi_0    pypi
setuptools                75.8.0           py312h06a4308_0
six                       1.17.0                    pypi_0    pypi
sqlite                    3.45.3               h5eee18b_0
tensorboard               2.18.0                    pypi_0    pypi
tensorboard-data-server   0.7.2                     pypi_0    pypi
tensorflow                2.18.0                    pypi_0    pypi
termcolor                 2.5.0                     pypi_0    pypi
tk                        8.6.14               h39e8969_0
typing-extensions         4.12.2                    pypi_0    pypi
tzdata                    2025a                h04d1e81_0
urllib3                   2.3.0                     pypi_0    pypi
werkzeug                  3.1.3                     pypi_0    pypi
wheel                     0.45.1           py312h06a4308_0
wrapt                     1.17.2                    pypi_0    pypi
xz                        5.6.4                h5eee18b_1
zlib                      1.2.13               h5eee18b_1
```