# Course Work for Artificial Intelligence H COMPSCI 4004 (2023-2024)

## Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

1. In respect of work submitted not more than five working days after the deadline
   (a) the work will be assessed in the usual way;
   (b) the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.
2. Work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus. Penalty for non-adherence to Submission Instructions is 2 bands You must complete an "Own Work" form via `https://studentltc.dcs.gla.ac.uk/` for all coursework

## Submission and deadline

Your submission will be:

- a report in PDF which contains the answers to all questions including discussions, algorithms, plots and figures.
- the source code of the implementation of the corrupted samples selection method

**Deadline: You must submit this on Moodle by 6.00 PM, Friday 8$^{th}$ December 2023**.

# 1 Optimally Select a Subset of Training Instances for Supervised Learning

## 1.1 Problem Statement

In this coursework, we will explore an approach to select an optimal set of training instances for supervised learning. Specifically, we are going to work with the MNIST [1] dataset, a popular dataset for hand-written digit recognition. Some examples of the images in the MNIST dataset are illustrated in Figure 1.

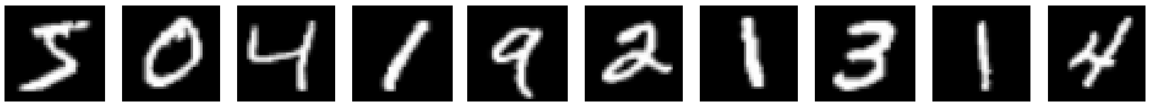

Figure 1: Examples of the images in the MNIST dataset.

Classifying the images in MNIST is no longer a big challenge since a lot of advanced deep learning models have been proposed over the years and the top performers can achieve a classification accuracy of over 99% (see `https://paperswithcode.com/sota/image-classification-on-mnist`). Having said that, most of the existing methods are based on the assumption that the annotation of the data is reliable and train the model accordingly. However, it is possible that the dataset contains inaccurate annotation due to human mistakes or even contaminated data. As a result, the performance of the trained model will be degraded.

We implemented a simulated workflow of learning with corrupted labels[1], which will act as a starting point for implementing your own solutions.

The notebook provides functions to *corrupt labels* with common types of mistakes (specific to this hand-written digit classification task), such as labelling a "3" as a "8" (and vice versa), a "1" as a "4" or "7" etc. A parameter that controls how many annotations will be corrupted in the dataset (i.e., how noisy is the data) by specifying a value for *noise_probability* ($\in [0, 1]$).

It will be instructive to look into the `corrupt_labels` and the `corrupt_label` functions in the provided notebook for the exact details.

We further provide you with some utility functions to start the supervised (parametric) training with the (noisy) labelled examples using the `trainAndEvaluateModel` function. Points to note:

1. We are not going to change the model architecture. This is considered frozen.

2. We are going to investigate if changing the dataset (seeking to leave out the noise from the training set) improves the accuracy over the test set.

[1]`https://colab.research.google.com/drive/1pWnuZ4AVZOkh2ZTTp-xtVnnoiTrXCwr6?usp=sharing`

You don't need to understand how the classification model really works (it's a one-layer 2D convolution network, which is covered in the Deep Learning and Computer Vision courses). The model's reference implementation is to be used only to evaluate the quality of your training data (the better the trained model performs on the test set, the better is the quality of the training set).

## 1.2 Your Tasks

As a part of this coursework, you need to implement a **subset selection function** that when called will return a subset of instances which will be used to train the model.

- Hypothesis: A model trained on noisy labels may not be that good. Whereas, if we can filter out the noisy data then perhaps we can improve on the test set accuracy, i.e., lead to more robust training.

To make your task easier, the workflow lets you also pass in a dictionary which contains the indexes of the instances that you would not want to use while training the model, i.e., it should contain a list of indexes that you would decide to **leave out** for training. This decision *comes from your implementation.*

We implemented a very simple and naive approach in the `baseLinePrunedSubsetMethod`. This method *randomly* selects a subset of samples to be pruned during the training (so, this essentially is a one-step random agent).

To test the oracle (ideal) situation, the program (during generating the corrupted data as a part of the simulation) keeps track of which data instances were corrupted. It turns out that if you indeed leave out those ones, you get a considerable improvement in the accuracy. This is achieved via the following line.

```
trainAndEvaluateModel(x_train, y_train_onehot,
        x_test, y_test_onehot, model, corrupted_indexes)
```

Note that you cannot use this `corrupted_indexes` dictionary as this is an unrealstic assumption. This simply gives you a sort of an upper bound on the performance. You should expect your results to be somewhere between the random baseline and the oracle.

To summarise, **your task is to implement your own version** of `myPrunedSubsetMethod` (which should take as arguments `x_train`, `y_train`, and `model`). The function should return a dictionary of indexes that are to be left out. You can then plug your function in and evaluate the results.

**Note:** We will focus on the selection of the samples to be left out in this coursework. You are not permitted to change the model architecture to improve the classification performance (i.e., you are to keep the `trainAndEvaluateModel` function frozen).

**Hints:** You can approach this as a **discrete state space optimisation** problem, where firstly you can define a "selection batch size" (this is not the same as training batch size), which decides which batch of instances you're going to leave out. For instance, if you are in a state where the training set is $X$, you may select (by some heuristics) which points you're gonna leave out (let that set be $\delta_1 \subset X$) so that a child state becomes $X' = X - \delta$.
Similarly, if you choose a different $\delta' \subset X$ you reach a different child state. You then need to train and evaluate (call the function `trainAndEvaluateModel`) to see if that child state led to an improvement or not.
A larger batch size would mean that you have a smaller number of children states to consider (convince yourself).

You are free to use any algorithm, e.g., simulated annealing, A* search, genetic algorithm etc. to implement this discrete state space optimisation.

You need to complete each of the following tasks. This is not just a coding exercise; rather, you should submit a report where you clearly describe how you have worked towards the solution of each task (preferably with examples). Your solution should be easily readable and sufficiently clear for others to understand and replicate.

### 1.2.1 Defining the states and actions [10 marks]

Clearly define the states and the actions for this problem. Clearly describe with a diagram the transition between two pairs of states.

### 1.2.2 Explain the design of your solution [40 marks]

This is the core of your methodology. Based on the states and actions explained in the previous section, you will focus on the method and algorithm to guide your subset selection program from the initial state to the goal state.

Please also explain the rationale behind your design and justify your decisions. You may discuss the other alternative solutions as well. In-line references can be included to further support your arguments.

### 1.2.3 Reporting the classification results [10 marks]

You should summarize the classification results on the baseline(s) and your solution(s) in this section. You may consider using a tabular format to show the results.

### 1.2.4 Analyse the results and suggest potential improvements [40 marks]

In this task, you will review and analyse the performance of your proposed solution based on the design and the experimental results in the previous sections. *Marks will be given to the quality of the analysis instead of the optimality of the solution.* You should also comment on potential improvement(s) to your proposed method. In-line references can be included to further support your arguments.

### 1.3 Items to Submit

1. A written report (in pdf) outlining your methodology and the observations (test set accuracy), and optionally any other details such as the parameter settings of your method (e.g., branching factor for A* search or population size for genetic algorithm).

2. Analysis of your solution (typically in terms of the effect of the parameters of your method). This is also a part of the report.

3. The source code (which will not be carrying marks directly). However, we will do some basic checks on your implementation to see if it's correct (we, most likely, would not be running your code).

## References

[1] Yann LeCun, Corinna Cortes and CJ Burges. "MNIST handwritten digit database". In: *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist* 2 (2010).