# Machine Learning Practical: Coursework 2

**Release date: Monday 6th November 2017**

**Due date: 16:00 Tuesday 28th November 2017**

## 1 Introduction

The aim of this coursework is to further explore the classification of images of handwritten digits using neural networks. We'll be using an extended version of the MNIST database, the EMNIST Balanced dataset, described in Section 2. Part A of the coursework will consist of building baseline deep neural networks for the EMNIST classification task, implementation and experimentation of the Adam and RMSProp learning rules, and implementation and experimentation of Batch Normalisation. Part B will concern implementation and experimentation of convolutional networks. As with the previous coursework, you will need to hand in test files generated from your code, and a report.

## 2 Dataset

In this coursework we shall use the EMNIST (Extended MNIST) Balanced dataset, https://www.nist.gov/itl/iad/image-group/emnist-dataset [Cohen et al., 2017]. EMNIST extends 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.

Although there are 62 potential classes for EMNIST (10 digits, 26 lower case letters, and 26 upper case letters) we shall use a reduced label set of 47 different labels. This is because of confusions which arise when trying to discriminate upper-case and lower-case versions of the same letter, following the data conversion process. 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 and Z.

The training set for Balanced EMNIST has about twice the number of examples as MNIST, 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. See Cohen et al. [2017] for some baseline results on EMNIST, as well as a description of the dataset.

You don't need to download the EMNIST database from the NIST website, it will be part of the `coursework_2` branch from the `mlpractical` Github repository, discussed in Section 3 below.

## 3 Code

You should run all of the experiments for the coursework inside the Conda environment you set up in the first labs. The code for the coursework is available on the course Github repository on a branch `mlp2017-8/coursework_2`. 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 details here 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 coursework_2 origin/mlp2017-8/coursework_2`

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 notebook `BatchNormalizationLayer_tests` which includes test functions to check the implementations of the BatchNorm layer `fprop`, `bprop` and `grads_wrt_params` methods. The BatchNormalizationLayer skeleton code can be found in mlp.layers. The tests use the mlp.layers implementation so be sure to reload your notebook when you update your mlp.layers code.
- A notebook `Convolutional_layer_tests` which includes test functions to check the implementations of the Convolutional layer `fprop`, `bprop` and `grads_wrt_params` methods. The ConvolutionalLayer skeleton code can be found in mlp.layers. The tests use the mlp.layers implementation so be sure to reload your notebook when you update your mlp.layers code.
- A new `ReshapeLayer` class in the `mlp.layers` module. When included in a a multiple layer model, this allows the output of the previous layer to be reshaped before being forward propagated to the next layer.
- A new `EMNISTDataProvider` class in the `mlp.data_providers` module. This class is a small change to the `MNISTDataProvider` class, linking to the Balanced EMNIST 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

There will also be a `coursework_2/report` directory which contains the LaTeX template and style files for the report. You should copy all these files into the directory which will contain your report.

## 4 Tasks

### Part A: Deep Neural Networks

In part A of the coursework you will focus on using deep neural networks on EMNIST, and you should implement the Adam and RMSProp learning rules, and Batch Normalisation.

1. Perform baseline experiments using DNNs trained on EMNIST. Obviously there are a lot things that could be explored including hidden unit activation functions, network architectures, training hyperparameters, and the use of regularisation and dropout. You cannot explore everything and is best to carefully investigate a few things in depth.

2. Implement the RMSProp [Tieleman and Hinton, 2012] and Adam [Kingma and Ba, 2015] learning rules, by defining new classes inheriting from `GradientDescendLearningRule` in the `mlp/learning_rules.py` module. The `MomentumLearningRule` class is an example of how to define a learning rules which uses an additional state variable to calculate the updates to the parameters.

3. Perform experiments to compare stochastic gradient descent, RMSProp, and Adam for deep neural network training on EMNIST, building on your earlier baseline experiments.

4. Implement batch normalisation [Ioffe and Szegedy, 2015] as a class `BatchNormLayer`. You need to implement `fprop`, `bprop` and `grads_wrt_params` methods for this class.

5. Verify the correctness of your implementation using the supplied unit tests in `Batchnorm_tests.ipynb`.

6. Automatically create a test file `sXXXXXXX_batchnorm_test.txt`, by running the provided program `generate_conv_test.py` which uses your `BatchnormLayer` class methods on a unique test vector generated using your student ID number.

7. Perform experiments on EMNIST to investigate the impact of using batch normalisation in deep neural networks, building on your earlier experiments.

In the above experiments you should use the validation set to assess accuracy. Use the test set at the end to assess the accuracy of the deep neural network architecture and training setup that you judge to be the best.

#### Part B: Convolutional Networks

In part B of the coursework you should implement convolutional and max-pooling layers, and carry out experiments using a convolutional networks with one and two convolutional layers.

1. Implement a convolutional layer as a class `ConvolutionalLayer`. You need to implement `fprop`, `bprop` and `grads_wrt_params` methods for this class.
2. Verify the correctness of your implementation using the supplied unit tests in `Convolutional_layer_tests.ipyn`
3. Automatically create a test file `sXXXXXXX_conv_test.txt`, by running the provided program `generate_conv_test.py` which uses your `ConvolutionalLayer` class methods on a unique test vector generated using your student ID number.
4. Implement a max-pooling layer. Non-overlapping pooling (which was assumed in the lecture presentation) is required. You may also implement a more generic solution with striding as well.
5. Construct and train networks containing one and two convolutional layers, and max-pooling layers, using the Balanced EMNIST data, reporting your experimental results. As a default use convolutional kernels of dimension 5x5 (stride 1) and pooling regions of 2x2 (stride 2, hence non-overlapping). As a default convolutional networks with two convolutional layers, investigate a network with two convolutional+maxpooling layers with 5 feature maps in the first convolutional layer, and 10 feature maps in the second convolutional layer.

As before you should mainly use the validation set to assess accuracy, using the test set to assess the accuracy of the convolutional network you judge to be the best.

## 5 Unit Tests

Part one of your coursework submission will be the test files generated for batch normalisation (`sXXXXXXX_batchnorm_test.txt`) and for the convolutional layer (`sXXXXXXX_conv_test.txt`), as described above. Please do not change the names of these files as they will be automatically verified.

## 6 Report

Part two of your coursework submission, worth 70 marks will be a report. The directory `coursework_2/report` contains a template for your report (`mlp-cw2-template.txt`); the generated pdf file (`mlp-cw2-template.pdf`) is also provided, and you should read this file carefully as it contains information about the required structure and experimentation. The template is written in LaTeX, and we strongly recommend that you write your own report using LaTeX, using the supplied document style `mlp2017` (as in the template).

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.

Your report should be in a 2-column format, based on the document format used for the ICML conference. The report should be a **maximum of 7 pages long**, with a further page for references. We will not read or assess any parts of the report beyond the allowed 7+1 pages.

As before, all figures should be included in your report file as vector graphics; please see the section in `coursework1.pdf` about how to do this.

If you make use of any any books, articles, web pages or other resources you should appropriately cite these in your report. You do not need to cite material from the course lecture slides or lab notebooks.

To create a pdf file `mlp-cw2-template.pdf` from a LaTeX source file (`mlp-cw2-template.tex`), you can run the following in a terminal:

```
pdflatex mlp-cw2-template
bibtex mlp-cw2-template
pdflatex mlp-cw2-template
pdflatex mlp-cw2-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-cw2-template
```

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.

## 7 Mechanics

**Marks:** This assignment will be assessed out of 100 marks and forms 25% of your final grade for the course.

**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. Identically named files will overwrite earlier submitted versions, so we will mark the latest submission that comes in before the deadline.

If you submit anything before the deadline, you may not resubmit afterward. (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, and a late penalty will be applied to this submission unless you have received an approved extension. Please be aware that late submissions may receive lower priority for marking, and marks may not be returned within the same timeframe as for on-time submissions.

*Warning:* Unfortunately the `submit` command 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, and (even worse) you may still be penalized for submitting late because the timestamp will update.

For additional information about late penalties and extension requests, see the School web page below. Do **not** email any course staff directly about extension requests; you must follow the instructions on the web page.

http://web.inf.ed.ac.uk/infweb/student-services/ito/admin/coursework-projects/late-coursework-extension-requests

**Late submission penalty:** Following the University guidelines, late coursework submitted without an authorised extension will be recorded as late and the following penalties will apply: 5 percentage points will be deducted for every calendar day or part thereof it is late, up to a maximum of 7 calendar days. After this time a mark of zero will be recorded.

## 8 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 consitute a good reason for late submission**.

You may *additionally* wish to keep your coursework under version control in your local Git repository on the `coursework_2` 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 Submission

Your coursework submission should be done electronically using the `submit` command available on DICE machines.

Your submission should include

- the unit test files generated for part 1, `sXXXXXXX_batchnorm_test.txt` and `sXXXXXXX_conv_test.txt`, where your student number replaces `sXXXXXXX`. Please do not change the names of these files.
- your completed report as a PDF file, using the provided template
- any notebook (`.ipynb`) files you used to run the experiments in
- and your local version of the `mlp` code including any changes you made to the modules (`.py` files).

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

You should copy all of the files to a single directory, `coursework2`, e.g.

```
mkdir coursework2
cp reports/coursework2.pdf sXXXXXXX_batchnorm_test.txt sXXXXXXX_conv_test.txt coursework2
```

and then submit this directory using

```
submit mlp cw2 coursework2
```

Please submit the directory, not a zip file, not a tar file.

The `submit` command will prompt you with the details of the submission including the name of the files / directories you are submitting and the name of the course and exercise you are submitting for and ask you to check if these details are correct. You should check these carefully and reply `y` to submit if you are sure the files are correct and `n` otherwise.

You can amend an existing submission by rerunning the `submit` command any time up to the deadline. It is therefore a good idea (particularly if this is your first time using the DICE submit mechanism) to do an initial run of the `submit` command early on and then rerun the command if you make any further updates to your submisison rather than leaving submission to the last minute.

## 10 Marking Scheme

- Part 1, Unit tests (30 marks).
- Part 2, Report (70 marks). The following aspects will contribute to the mark for your report:
  - Abstract - how clear is it? does it cover what is reported in the document
  - Introduction - do you clearly outline and motivate the paper, and describe the research questions investigated?
  - Methods – have you carefully described the approaches you have used?
  - Experiments – did you carry out the experiments correctly? are the results clearly presented and described?
  - Interpretation and discussion of results
  - Conclusions
  - Presentation and clarity of report

## References

G. Cohen, S. Afshar, J. Tapson, and A. van Schaik. EMNIST: an extension of MNIST to handwritten letters. *arXiv preprint arXiv:1702.05373*, 2017. URL https://arxiv.org/abs/1702.05373.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *ICML*, pages 448–456, 2015. URL http://proceedings.mlr.press/v37/ioffe15.html.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICML*, 2015. URL https://arxiv.org/abs/1412.6980.

T. Tieleman and G. E. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012. URL https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.