Machine Learning Practical 2020/21: Coursework 2

Released: Monday 16 November 2020

Submission due: 16:00 Monday 30 November 2020 (extended from 27th)

1 Introduction

The aim of this coursework is to explore the classification of images using convolutional neural networks on a different dataset, CIFAR100 (pronounced as "see far 100"). CIFAR100 consists of 60,000 32×32 colour images in 100 classes, with 600 images per class. The first part of the coursework concerns with debugging a "broken" neural network and quantitatively analyzing the problem behind this broken network. The second part involves exploring the solutions in literature for fixing this "broken" neural network, and then subsequently implementing one of these solutions to improve the performance and training of this network.

In order to support your experiments, we have acquired Google Cloud Platform credits which allow the use of the Google Compute Engine infrastructure. You will need this to run all the tasks of the coursework (which will be carried out in PyTorch). Each student enrolled on the MLP course will receive a $50 Google Cloud credit coupon which is enough to carry out the experiments required for this coursework. You will receive an email which will give you the URL you will need to access in order to request your Google Cloud Platform coupon.

As in the previous coursework, you will need to submit your python code and a report. The detailed submission instructions are given in Section 8.2 – please follow these instructions carefully.

2 Github branch `mlp2020-21/coursework_2`

The provided code and setup information for this coursework is available on the course Github repository on a branch `mlp2020-21/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 notes/getting-started-in-a-lab.md 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/mlp2020-21/coursework_2`

You will now have a new branch in your local repository with everything you need to carry out the coursework.

Before continuing, remember to run `bash install.sh` to install some additional dependencies required. This assumes that you have already installed your environment as explained in **Lab1**, under `notes/environment-set-up.md`.

This branch includes the following additions to your setup:

- For the PyTorch Framework:
  - A Jupyter notebook, `Coursework_2_Pytorch_Introduction.ipynb`, which introduces Pytorch, and provides further resources on tutorials, documentation and debugging.

  - A directory `pytorch_mlp_framework/`, which includes tooling and ready to run scripts to enable straightforward experimentation on GPU. Documentation on this is included in `notes/pytorch-experiment-framework.md`
  - A note on how to use the Google Cloud Platform (using your student credits), `notes/google_cloud_setup.md`.

- For the report:
  - A directory called `report` which contains the LaTeX template and style files for your report. You should copy all these files into the directory which will contain your report.

3 Tasks

This coursework has three tasks and objectives:

- Diagnose a deep CNN that cannot be properly trained due to optimization issues. Discuss what the problem is, why it happens in a deep CNN and quantitatively analyze the problem.
- Discuss solutions present in the literature to address this problem.
- Propose in detail and implement one of the solutions to the problem. Since the aim of this is to improve the `training` behaviour of the network, the solutions should then be evaluated in terms of convergence speed and per-epoch training accuracy/loss and gradient flows.

Carrying out larger convolutional network experiments using the MLP numpy framework is computationally inefficient because (1) it runs on CPU and not on GPU, and (2) a default implementation of a convolutional layer is unlikely to be computationally efficient. For these reasons in this coursework, which involves running convolutional network experiments we will use GPU computing (on the Google Compute Engine) and the highly efficient PyTorch framework.

4 Task 1: Identifying training problems of a deep CNN

Time budget[1]*: This section should take about 20% of the time you have allocated for your coursework.*

This part of the coursework involves debugging and tuning deep CNNs using PyTorch and the Google Compute Engine.

Identifying and resolving any optimization-related problems that might prevent your model from fitting the training set are critical skills when working with deep neural networks. Being able to work around this problem is key to building high performance deep neural networks.

4.1 Introducing our broken CNN

Note: This section's objective is to test your knowledge, understanding and research skill. It's not meant to be an implementation-oriented section. The solutions only require you to add at most 4-5 lines of code.

Using the Pytorch-based research framework that can be found in the `pytorch_mlp_framework` folder, we have built, trained and evaluated 2 deep CNNs. One consisting of a total of 7 convolutional layers + one fully connected, and another consisting of 37 convolutional layers + one fully connected layer. Figure 1 illustrates the

[1] These are simply suggested guidelines, feel free to adjust them based on your own preferences

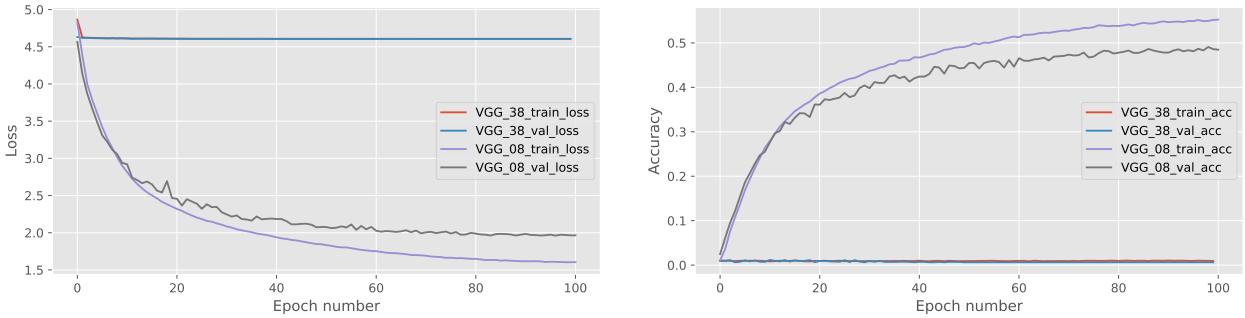

Figure 1: Training and validation plots of Healthy (VGG_08) vs Broken (VGG_38) in terms of error and accuracy.

training/val loss performance of the two models. One can clearly see that the 37 layer CNN (VGG_38) was unable to minimize its loss, unlike the healthy 7 layer CNN (VGG_08) which converges to a low error. Given that we know that extra layers means more abstraction power, parameters and capacity, one would expect the deeper model to be doing better at learning than the shallow one, however this is simply not the case.

Identifying the problem. Construct a hypothesis as to what is causing the issue in Figure 1. You can reproduce the figures by running the `notebooks/Plot_Results.ipynb` notebook. This file takes as input the collected metrics in folders `VGG_08` and `VGG_38`.

Quantitative Analysis of the problem. Support your position of the problem using arguments based on quantitative observations. For instance, Figure 2 shows how the gradient flows in the healthy network, `VGG_08` across layers. These are the *gradients with respect to the weights* of the model. Visualize and discuss how gradient flows for the broken network affects/does not affect the loss curves, training and convergence of the 37 layer CNN.

4.1.1 Implementation Guidelines

The curves shown in Figure 1 can be reproduced by running the bash scripts to train each model from scratch with the default settings given in `run_vgg_08_default.sh` and `run_vgg_38_default.sh`.

For reproducing Figure 2 and visualizing the gradient flows for the broken network, implement the function `plot_grad_flows()` within `pytorch_mlp_framework/experiment_builder.py`. This function takes as input the `model` parameters during training, accumulates the absolute mean of the gradients in `all_grads` and the layer names in `layers`. The `matplotlib` function `plt` plots gradient values for each layer and the function `plot_grad_flows()` returns this final plot.

(20 Marks)

5 Task 2: Background Literature

Time budget[1]*: This section should take about 20% of the time you have allocated for your coursework.*

There exist *at least* several methods that can improve the training performance of the broken 37 layer CNN introduced in Section 4.1. Discuss, in your own words, **any three out of the four research papers** given below.

- Batch normalization: Accelerating deep network training by reducing internal covariate shift, Ioffe and Szegedy [2015].
- Deep residual learning for image recognition, He et al. [2016].

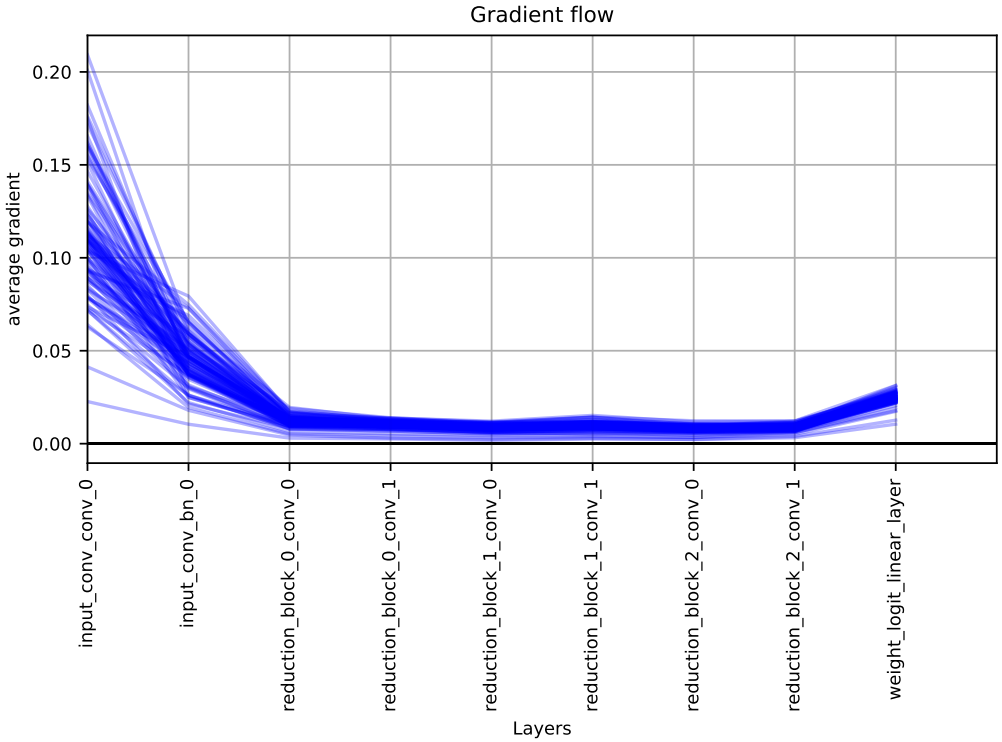

Figure 2: Gradient Flow in each layer for the VGG_08 network.

- Densely connected convolutional networks, Huang et al. [2017]
- Deeply-supervised nets, Lee et al. [2015]

This discussion should outline what problems the papers address, are these related to the problem you analyze in Task 1, what/which methods they adopt to solve it, brief details of the method and the final conclusion from the papers. Summary of each paper should be limited to 200-300 words.

(20 Marks)

6 Task 3: Solution and Experiments

Time budget[1]*: This section should take about 30% of the time you have allocated for your coursework.*

Solution Overview. The recommended solutions are in the four papers listed in section 5. Pick and discuss any one solution in detail in this section. Discuss theoretically and intuitively why you chose this solution and how it addresses/improves the training performance of the VGG_38 model.

Experiments. The recommended solutions are described in Ioffe and Szegedy [2015], He et al. [2016]. You can also find alternative ones in Huang et al. [2017], Lee et al. [2015] which can be more challenging to implement. We have written the MLP Pytorch framework in a way that allows you to implement said solutions with minimal effort. Each solution will require at approximately 8-10 lines of code.

Important Components to Note

1. The classes `ConvolutionalProcessingBlock` and `ConvolutionalDimensionalityReductionBlock` located within `pytorch_mlp_framework/model_architectures.py`. The former class implements a

basic cascade of 2 convolutional layers, each followed by a Leaky ReLU activation function, while the latter implements a basic cascade of 2 convolutional layers, with an average pooling layer in the middle, which effectively halves the height and width dimensions of the tensor volume passing through it. The former is used as the basic building block of the model (repeated `num_blocks_per_stage times`), while the latter is used only as a dimensionality reduction layer, usually once after each *network stage*. A network stage is considered a cascade of convolutional layers, followed by a dimensionality reduction function such as average pooling.

2. Lines 43-50 in `pytorch_mlp_framework/train_evaluate_image_classification_system.py` showcase how one can create an if-else structure that can read arguments from the command line in order to choose which processing and dim_reduction blocks will be used.
3. Lines 17-27 in the same file showcase how some simple data-augmentation strategies can be applied to the system (useful in improving the generalization of the model.)

Implementing a proposed solution

1. Copy the classes `ConvolutionalProcessingBlock` and `ConvolutionalDimensionalityReductionBlock` located within `pytorch_mlp_framework/model_architectures.py`, and provide a name referring to your proposed solution. Then change the two blocks to implement your proposed solution.
2. To create a new block ensure that you rewrite the `build_module()` and `forward()` methods.
3. Add a new clause in the if-else structure in lines 43-50 of `pytorch_mlp_framework/train_evaluate_image_classification_system.py` to add a choice for a configuration using your new blocks.
4. Write small unit-tests for your blocks to ensure that the layers can fprop some data without throwing errors related to Pytorch.
5. Once tested, you can use the argument parser to easily run experiments with your new modules. Hint: Have a look at the `block_type` argument under `pytorch_mlp_framework/arg_extractor.py`, to get an idea on how to easily pass module names as an argument.

Note: Some potential solutions might not require modification of the above, but those are considered more advanced. If you choose that path, you will also bear the responsibility of figuring out the design of your code. That being said, most of known methods can be implemented using the above 3 steps.

To break this down:

1. Select and implement one method to address the problem in Task 1.
2. Design and run experiments to investigate the performance of the method. Make sure to run your experiments on multiple seeds.
3. Build a final model using the knowledge you have gained and evaluate its test performance.
4. Quantitatively and intuitively analyze how this solution addressed the problem.
5. Write about what you did and why in your report.

Important thing to note about experiments:

1. The purpose of these experiments is NOT to report results with the best combination of hyper-parameters but finding the ones so that your solution addresses the problem discussed in Task 1 effectively. You will have to do a valid hyper-parameter search over learning rates, batch size, weight decay coefficients, etc. such that your model achieves a stable training and testing accuracy with the proposed solution. Make sure you choose the values wisely and not exhaust your google cloud credits.

2. Designing experiments so that you are in a position to draw conclusions from your experiments is more important than doing as many experiments as possible.
3. When reporting the results of experiments, make sure that the comparison/contrast you are exploring is clear in the way you present the results.

When writing the report on this task you should include the following:

- Outline and explain the research questions you are investigating. Provide citations if appropriate.
- Explain the methodology used – in this case the approach/method picked to solve the problem. Provide citations if appropriate.
- Describe carefully the experiments carried out, providing enough information to make them reproducible. Present your results clearly and concisely. Graphs and tables should be constructed to make clear the contrasts and comparisons you are interested in based on the research questions. For instance, some interesting evaluation criteria that can be used to compare different strategies are classification accuracy, learning speed of your network and gradients.
- Discuss your results, with reference to the research questions, and if appropriate with reference to the literature. What conclusions can you draw from your experiments and are they consistent with the literature?

Using Google Compute Engine

1. You will receive an email containing the URL you will need to access in order to request a Google Cloud Platform coupon, and information about how to do this.
2. In the `coursework_2` branch of the GitHub, `notes/google_cloud_setup.md` gives the instructions you should follow to set up a Google Compute Engine instance to carry out this coursework.
3. The PyTorch experimental framework that is used for this coursework is described in `notes/pytorch-experiment-framework.md` and in `notebooks/Coursework_2_Pytorch_experiment_framework.ipynb`

(35 Marks)

7 Report

Time budget[1]*: This section should take about 30% of the time you have allocated for your coursework.*

Your coursework will be primarily assessed based on your submitted report.

The report template is divided into sections which corresponds to each task in the coursework specs. In addition to the mark distribution for tasks, remaining sections of the report will be graded as follows:
Abstract and Introduction: *10 Marks*
Discussion and Conclusion: *15 Marks*

The directory `coursework_2/report` contains a template for your report (`mlp-cw2-template.tex`); the generated pdf file (`mlp-cw2-template.pdf`) is also provided, and you should read this file carefully as it contains some useful information about the required structure and content. The template is written in LaTeX, and we strongly recommend that you write your own report using LaTeX, using the supplied document style `mlp2020` (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. The coursework 1 spec outlines how to create a pdf file from a 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 6 pages long**, not counting the references. We will not read or assess any parts of the report beyond this limit.

As discussed in the coursework 1 spec, all figures should ideally be included in your report file as vector graphics files, rather than raster files as this will make sure all detail in the plot is visible.

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.

8 Mechanics

Marks: This assignment will be assessed out of 100 marks and forms 40% 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 after the deadline. (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 submission system on Learn 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.

Extension requests: 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 as these are handled by the ITO; 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.

8.1 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 constitute 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.

8.2 Submission

Your coursework submission should be done online on the Learn course webpage.

Your submission should include one zip file `sxxxxxxx.zip` that should contain a folder named sxxxxxxx. This folder should have -

- your completed report as a PDF file renamed as `sxxxxxxx_report.pdf`, using the provided template
- your local version of the Pytorch experiment framework (`mlp/pytorch_experiment_scripts`), including any changes you've made to existing files and any newly created files.
- a copy of your Pytorch experiment directories (`mlp/results`), including **only** the `.csv` files for your training, validation and test statistics. Please do not include model weights.

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

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

Your folder directory structure should look like this -

```
sxxxxxxx
├── sxxxxxxx_report.pdf
└── mlp
    ├── pytorch_experiment_scripts/
    └── results/
```

You can use this command on Linux machines to zip all the files together -

```
zip sxxxxxxx.zip sxxxxxxx
```

Replace `sxxxxxxx` with your student id.

Once you have successfully created the .zip file, you need to login to your Learn `Machine Learning Practical (2020-2021)[YR]` webpage and submit the file.

- Migrate to the section **Coursework** on the left column on the course page.
- Click on Coursework 2.
- A page will appear where you will need to browse and upload your .zip file that you created previously in `Attach Files` and then click **Submit**.

You can amend an existing submission by attaching a different .zip file using the Attach Files option and then Submit again.

Note that we will only mark the last uploaded coursework in case you amend your files. Thus it is your responsibility to make sure that correct files are uploaded.

9 Marking Guidelines

This document (Section 3 in particular) and the template report (`mlp-cw2-template.pdf`) provide a description of what you are expected to do in this assignment, and how the report should be written and structured.

Assignments will be marked using the scale defined by the **University Common Marking Scheme**:

Numeric mark	Equivalent letter grade	Approximate meaning
< 40	F	fail
40-49	D	poor
50-59	C	acceptable
60-69	B	good
70-79	A3	very good/distinction
80-100	A1, A2	excellent/outstanding/high distinction

Please note the University specifications for marks above 50:

A1 90-100 Often faultless. The work is well beyond what is expected for the level of study.

A2 80-89 A truly professional piece of scholarship, often with an absence of errors.
As 'A3' but shows (depending upon the item of assessment): significant personal insight / creativity / originality and / or extra depth and academic maturity in the elements of assessment.

A3 70-79
Knowledge: Comprehensive range of up-to-date material handled in a professional way.
Understanding/handling of key concepts: Shows a command of the subject and current theory.
Focus on the subject: Clear and analytical; fully explores the subject.
Critical analysis and discussion: Shows evidence of serious thought in critically evaluating and integrating the evidenced and ideas. Deals confidently with the complexities and subtleties of the arguments. Shows elements of personal insight / creativity / originality.
Structure: Clear and coherent showing logical, ordered thought.
Presentation: Clear and professional with few, relatively minor flaws. Accurate referencing. Figures and tables well constructed and accurate. Good standard of spelling and grammar.

B 60-69
Knowledge: Very good range of up-to-date material, perhaps with some gaps, handled in a competent way.
Understanding and handling of key concepts: Shows a firm grasp of the subject and current theory but there may be gaps.
Focus on the subject: Clear focus on the subject with no or only trivial deviation.
Critical analysis and discussion: Shows initiative, the ability to think clearly, critically evaluate ideas, to bring different ideas together, and to draw sound conclusions.
Structure: Clear and coherent showing logical, ordered thought. Additionally for code: re-usability may be somewhat limited. No unused variables or dead code.
Presentation: Clear and well presented with few, relatively minor flaws. For writing: Accurate referencing; using the correct referencing system. Figures and tables well-constructed and accurate. Good standard of spelling and grammar. Alternatively for code: well-documented, readable code.

C 50-59
Knowledge: Sound but limited. Inaccuracies, if any, are minor.
Understanding and handling of key concepts: Understands the subject but does not have a firm grasp and depth of understanding of all the key concepts.
Focus on the subject: Addresses the subject with relatively little irrelevant material.
Critical analysis and discussion: Limited critical analysis and evaluation of sources of evidence.
Structure: Reasonably clear and coherent, generally structuring ideas and information or code in a logical way. Additionally for code: Few or no unused variables or dead code.

Presentation: Generally well presented but there may be some flaws, for example in figures, tables, referencing technique and standard of English. Alternatively for code: generally well-documented, readable code, but with some weaknesses.

References

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

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.

Chen-Yu Lee, Saining Xie, Patrick Gallagher, Zhengyou Zhang, and Zhuowen Tu. Deeply-supervised nets. In *Artificial intelligence and statistics*, pages 562–570, 2015.