MLP Coursework 2: Exploring Convolutional Networks

s1546138

Abstract

The abstract should be a few sentences (100–200 words) long, providing a concise summary of the contents of your report including the key research question(s) addressed, the methods explored, the data used, and the findings of the experiments.

1. Introduction

This document provides a template for the MLP coursework 2 report. This template structures the report into sections, which you are recommended to use, but can change if you wish. If you want to use subsections within a section that is fine, but please do not use any deeper structuring. In this template the text in each section will include an outline of what you should include in each section, along with some practical LaTeX examples (for example figures, tables, algorithms). Your document should be no longer than **six pages**, with an additional page (or more!) allowed for references.

The introduction should place your work in context, giving the overall motivation for the work, and clearly outlining the objectives of the work and the research questions you have explored – in this case the implementation of convolutional networks and the exploration of how context is handled in convolutional networks. Most of the report will be to do with part 2.

This section should also include a concise description of the Balanced EMNIST task and data – be precise: for example state the size of the training, validation, and test sets.

2. Implementing convolutional networks

2D convolutional layers work by applying a set of learned filters over the input, creating a feature map which is the layer output. The number of filters in a layer affects the depth dimension of the feature map (number of features per area of input). Each filter is applied over the input in a sliding manner. Conv layers are better suited for image inputs versus fully connected layers because they take into account spatial locality of the inputs, and have much fewer parameters due to shared weights (we're only learning the filter parameters) and local connectivity which means faster training and better generalization.

We implement the forward pass as a matrix multiplication. We map the input into columns, where each column is a flattened receptive field. Note that the size of the mapped input matrix will be larger than the input in almost all

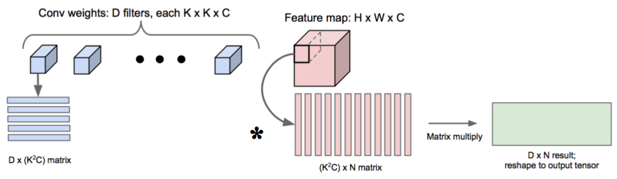

Figure 1. Convolution as matrix multiplication

cases, as the receptive fields overlap. This is a tradeoff with higher memory use for faster compute. We map the filters into rows, where each row is contains the flattened filter weights. The resulting matrix multiplication is the equivalent of many dot products between each receptive field and each conv filter, however being much faster than performing the dot products individually. The result can then be reshaped to match the layer output shape. Similarly, the backward pass is another convolution operation. Using the tests provided, it takes $964\mu s \pm 29.9\mu s$ to run the test fprop/bprop.

The purpose of max pooling layers is to reduce the dimensionality of the input, downsampling and taking only the maximum value in a window (of some size). This reduces the number of parameters in later layers significantly. The naive implementation of maxpooling layer is exactly the same as the im2col step in conv layer, except replacing the matrix product with the max of each receptive field. We remember the argmax for each receptive field so we can propagate the gradient through only the argmax in the backward pass. Note that we're using im2col here purely so that we can reuse code from the previous segment, there is not much of a performance increase over a for loop since we don't take advantage of fast matrix multiplications here.

There is one possible optimization, which is possible when none of the receptive fields overlap in the pooling layer. In this case, we can simply reshape the input by flattening the values in each receptive field (there is no need to create a new matrix since no value can be in 2 different receptive fields). As a benchmark, we run a maxpool of size 2 and stride 2 over the test inputs in `max_pooling_correct.npz` 10000 times using the `timeit` module. The naive im2col implementation took 1.615s, while the optimized implementation took only 0.186s, an 8x performance increase.

Concisely explain the idea of convolutional layers and pooling layers, and explain how you did the implementation. There is no need to include chunks of code. You may also discuss the pros and cons of different approaches to implementing convolutional layers, in terms of computational efficiency (running time, storage demands), and any analysis of your own implementation.

3. Context in convolutional networks

This section should introduce the main part of the report which is to do with exploring different ways of modelling context in convolutional networks. This section should present, in your own words, the different approaches you have adopted, explaining how they work and what the key differences between them are. You may reference the literature where appropriate. You should also outline the key research questions that you have addressed in this work.

If you present algorithms, you can use the `algorithm` and `algorithmic` environments to format pseudocode (for instance, Algorithm 1). These require the corresponding style files, `algorithm.sty` and `algorithmic.sty` which are supplied with this package.

Algorithm 1 Bubble Sort

```
Input: data $x_i$, size $m$
repeat
  Initialize $noChange = true$.
  for $i = 1$ to $m - 1$ do
    if $x_i > x_{i+1}$ then
      Swap $x_i$ and $x_{i+1}$
      $noChange = false$
    end if
  end for
until $noChange$ is $true$
```

4. Experiments

This section should cover the experiments carried out. For each experiment, make clear why it was carried out, what you were trying to discover. Describe carefully how you carried out the experiments, mentioning and justifying any hyperparameter settings. As always, your aim is to give enough information so that someone else (e.g. another MLP group) could reproduce the experiment precisely. Note that it is interesting to consider both accuracy / generalisation and runtime / memory requirements.

Present the experimental results clearly and concisely. Usually a result is in comparison or contrast to a result from another approach please make sure that these comparisons/contrasts are clearly presented. You can facilitate comparisons either using graphs with multiple curves or (if appropriate, e.g. for accuracies) a results table. You need to avoid having too many figures, poorly labelled graphs, and graphs which should be comparable but which use different axis scales. A good presentation will enable the reader to compare trends in the same graph – each graph should summarise the results relating to a particular research (sub)question.

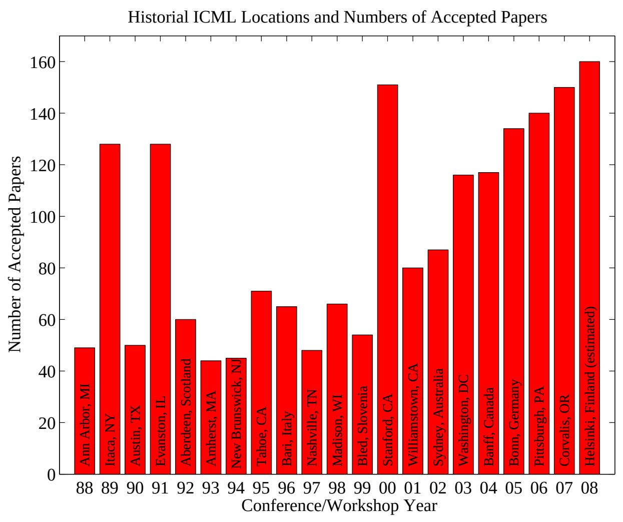

Figure 2. Historical locations and number of accepted papers for International Machine Learning Conferences (ICML 1993 – ICML 2008) and International Workshops on Machine Learning (ML 1988 – ML 1992). At the time this figure was produced, the number of accepted papers for ICML 2008 was unknown and instead estimated.

There is no need to include code or specific details about the compute environment.

As before, your experimental sections should include graphs (for instance, figure 2) and/or tables (for instance, table 1)[1], using the `figure` and `table` environments, in which you use `\includegraphics` to include an image (pdf, png, or jpg formats). Please export graphs as vector graphics rather than raster files as this will make sure all detail in the plot is visible. Matplotlib supports saving high quality figures in a wide range of common image formats using the `savefig` function. **You should use `savefig` rather than copying the screen-resolution raster images outputted in the notebook.** An example of using `savefig` to save a figure as a PDF file (which can be included as graphics in a LaTeXdocument is given in the coursework document.

If you need a figure or table to stretch across two columns use the `figure*` or `table*` environment instead of the `figure` or `table` environment. Use the `subfigure` environment if you want to include multiple graphics in a single figure.

5. Discussion

Your discussion should interpret the results, both in terms of summarising the outcomes of a particular experiment, and attempting to relate to the research question(s) which motivated the experiments . A good report would have some analysis, resulting in an understanding of why particular results are observed, perhaps with reference to the literature.

[1]These examples were taken from the ICML template paper.

Data set	Naive	Flexible	Better?
Breast	95.9± 0.2	96.7± 0.2	$\surd$
Cleveland	83.3± 0.6	80.0± 0.6	$\times$
Glass2	61.9± 1.4	83.8± 0.7	$\surd$
Credit	74.8± 0.5	78.3± 0.6	
Horse	73.3± 0.9	69.7± 1.0	$\times$
Meta	67.1± 0.6	76.5± 0.5	$\surd$
Pima	75.1± 0.6	73.9± 0.5	
Vehicle	44.9± 0.6	61.5± 0.4	$\surd$

Table 1. Classification accuracies for naive Bayes and flexible Bayes on various data sets.

Use bibtex to organise your references – in this case the references are in the file `example-refs.bib`. Here is a an example reference (**?**).

A good report will relate the results to published work which can help to give a better understanding of your work – related approaches, other work on the same data, ideas for future work.

6. Conclusions

The conclusions section should concisely summarise what you have learned from the experiments you carried out, and relate the findings of your work to the research questions you posed at the start. It is good if the conclusion from one experiment influenced what you did in later experiments – your aim is to learn from your experiments.

A good conclusions section would also include a further work discussion, building on work done so far, and referencing the literature where appropriate.