Final Report for CS39440 Major Project

Author: Theodoros Nikopoulos-Exintaris (thn2@aber.ac.uk)
Supervisor: Dr./Prof. Chuan Lu (cul@aber.ac.uk)

4th May 2016
Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a BSc degree in
Computer Science (GG47)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for Unacceptable Academic Practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the regulations on Unacceptable Academic Practice from the University's Academic Quality and Records Office (AQRO) and the relevant sections of the current Student Handbook of the Department of Computer Science.
- In submitting this work I understand and agree to abide by the University's regulations governing these issues.

Name .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Name .....

Date .....

Acknowledgements

I am grateful to...

I’d like to thank...

Abstract

Include an abstract for your project. This should be no more than 300 words.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Introduction

This project is an investigation into training Artificial Neural Networks for the purpose of image classification and labelling. We will explore alternative approaches in computer vision to solve this problem arriving to the what is considered the current state of the art in machine learning comparing it to a number of different alternatives both contemporary but also historical.

1.1.1 Motivation

Perhaps the first question we might wish to ask ourselves is why we wish to embark on this project in the first place. Computer vision is field of computer science in essence involving extracting information and building models of the real world, this might involve sensors, cameras or perhaps even audio.

There are numerous applications and types of computer vision. The problem we are looking in particular is identifying images and labelling them extracting metadata from the image itself. Naturally this has many applications. Self-driving cars, search engines, and any problem where we need a computer to be able to identify images.

Our interest is particularly related to image acquisition and labelling. Suppose a user has a large set of images and you are looking for a particular one. They know what it is but not its title. Or perhaps its title does not hold semantic meaning to them, but wish to search the semantics of the image itself.

1.1.2 Machine Learning and Pattern Recognition

To solve this problem computer scientists set out to devise a set of 'rules' and algorithms that were good at solving these problems. Eventually it became apparent that no single algorithm could solve the problem for all sets of cases as the 'rules' change between cases. The rules that define a cat are different than what defines a dog and so on so one would have to write rules for each edgecase. Where you needed to solve this kind of problem one would analyse the data they expect to receive and think of features they are interested in extracting. This approach is still used for some CV problems.

A solution that emerged for this was machine learning (ML), that is sets of algorithms that could be used to derive the rules based on data. Most machine learning algorithms work by identifying patterns in the data either via statistical analysis as in Bayesian Learning or other means. Suppose you have a set of data describing cats and dogs. Cats are small, but there is both small and large types of dogs, most cats have pointed ears while some dogs have floppy ones. A machine learning algorithm would process a set of examples of cats and dogs then derive patterns that can be used to classify them.

1.2 Background

1.2.1 SVMs and other Machine Learning Alternatives

1.2.2 Artificial Neural Networks

Artificial Neural Networks(ANN) are a type of machine learning inspired by the way some biological organisms process stimulus and learn. In essence an ANN is a very naive emulation of a 'meat computer' with the biological processes being replaced with mathematical approximations of the actual process of biological neurons.

Being that these are only approximations, they have several components which help them function and emulate select processes of real neurons.

1.2.2.1 Advantages and Disadvantages of ANNs

ANNs have many advantages and disadvantages to conventional ML methods some of the most important ones include:

- They do not require a feature extraction step
- Can easily be applied in many different problems

It is possible to apply ANNs in to solve many problems by just feeding in appropriate training data and get reasonably accurate results in some cases, matching the state of the art for that problem. As for disadvantages:

- They are difficult to train both in terms fo requirements and speed.
- It is comparatively difficult to gain insight on why a neural network has learned something.

Recent advances in ANNs have greatly improved on both of these areas. GPU compute in particular has made training quite large neural networks practical on consumer computers. There is also very active research at the moment in improving insight and visualising neural network decisions. We will talk in more detail about all of this later in our report.

1.2.2.2 Datasets

With most methodologies having data to test against is important. With ML, data is what drives the whole process so to produce any sort of solution to this problem one needs to decide on the

dataset to use. For classification problems a dataset might be a set of inputs and outputs, separated in a set to train on and a set to test performance against afterwards.

For our network we are looking at datasets with actual images with associated labels. There is several of that type of image dataset, we will discuss them in more detail in a later section of this report.

1.3 Analysis

To complete this project we must solve a number of problems and answer a number of questions. As part of our analysis we identified some of them while several others would emerge as we embarked on our project.

1.3.1 Types of ML and ANN architectures used in the past

To start we should look at what models have been successful in the past for this kind of problem. After all this area has several decades of research behind it which we would not be able to replicate from scratch.

Looking at older results also enables us to get a baseline for what our model might be capable of doing and provides goals to hit or exceed. For this report we will be looking at SVMs, Multi Layer Perceptrons (MLP) and Convolutional Neural Networks (CNN)

1.3.2 Architectures of CNN

We need to decide on the architecture of our neural network. For CNN choices appear to be relatively simple with most networks opting for very similar architectures.

1.3.3 Variables to test and optimise

We will need to investigate different ways we can vary our model and attributes that we can change in order to limit the scope of the research area. This process involves, reading documentation for our solutions as well as research into the technology we are using to determine the highest value optimisations to make.

1.3.4 Are CNNs ready for market?

One other interesting question to answer as a conclusion is whether CNNs are ready to be deployed in commercial products and services. Is it possible to produce something robust enough for such uses?

1.4 Research Method

Chapter 2

Experiment Methods

In this chapter we will explain the basic

This section should discuss the overall hypothesis being tested and justify the approach selected in the context of the research area. Describe the experiment design that has been selected and how measurements and comparisons of results are to be made.

You should concentrate on the more important aspects of the method. Present an overview before going into detail. As well as describing the methods adopted, discuss other approaches that were considered. You might also discuss areas that you had to revise after some investigation.

You should also identify any support tools that you used. You should discuss your choice of implementation tools or simulation tools. For any code that you have written, you can talk about languages and related tools. For any simulation and analysis tools, identify the tools and how they are used on the project.

If your project includes some engineering (hardware, software, firmware, or a mixture) to support the experiments, include details in your report about your design and implementation. You should discuss with your supervisor whether it is better to include a different top-level section to describe any engineering work.

Chapter 3

Experiment Implementation

To implement our experiments we need to make a series of decision.

3.1 Choice of Platform

To start we need to decide how we are going to implement the neural networks. This involved several steps, of course one would be tempted to write their own implementation for an ANN, however that might be a major project on its own so a better solution is required.

We would like to minimise implementation effort in parts that do not improve the final project as much as possible so a library of some sort is necessary.

3.1.1 Criteria of a Framework

For our framework we wanted something that is both performant but also easy to use. We also want support for the latest ANN features while it needs to work on our workstation.

3.1.2 Performance

One of the greatest developments of the past 8 years in ANNs and machine learning in general is the advent and wide availability of parallel computing. Specifically GPU compute solutions like nVidia's CUDA technology have been invaluable in making Deep Neural Network research viable offering orders of magnitude better training times. Our frameword must run on a GPU.

3.1.3 Ease of use

We would like a library that makes testing an idea quickly, two modern candidates emerged at that point Google's new Tensor Flow software was one of them and was our favourite for the early parts of the project. Keras was another. Both use Python to define models and a backend implementation to do processing

3.1.4 Features

We were not looking for specific features in this part of our project, what we were looking for however was a framework that is used in current research and is geared towards experimental CNNs both Keras and Tensor Flow support similar advanced CNN features which makes them fairly equivalent as a choice. They were both alpha software when the project started but so are most libraries in the field as CNNs are fairly new. We initially settled on Tensor Flow.

3.1.5 Platform Support

Most of the frameworks for this task are designed to work on Linux primarily. Tensor Flow works on Linux and OSX. This was a problem when we were running Windows on our main machine so the initial solution was to run Tensor Flow on a Virtual Machine with Linux Mint. This initially appeared to work well but we quickly discovered issues with this approach. To get good training performance CUDA is required. However CUDA requires a direct connection to the graphics adapter.

A solution we considered was to use Intel's VT-d and pass the PCI-E device of the GPU directly to the VM using the integrated Intel GPU in Windows instead allowing us to use it under a VM but this proved to be too complicated to set up and any other solution was equally complex. We had to abandon Tensor Flow. Keras, supported Windows as well so we ended up switching our efforts to that. This was relatively early on in the project's lifecycle so we could afford to make the change.

3.2 The Stack

In our architecture we ended up having several layers of software contributing to our final software stack.

3.2.1 GPU Drivers and CUDA

Because of the task of training and running through a neural network is an inherently parallel task we use Nvidia's CUDA technology in order to run computations on the GPU which produces at least an order of magnitude in performance gains over a modern traditional CPU architecture.

Nvidia's CUDA API has emerged as the dominant solution for scientific and industrial compute applications after competing open APIs like OpenCL failed to gain any traction. If you want good ANN performance CUDA is necessary.

3.2.2 CuDNN

CuDNN is an nVidia library written in C++ which provides performance optimisations for several ANN computations to run on CUDA enabled GPUs. This library isn't necessary, but it provided us with a significant performance boost so ended up being invaluable in our experiments.

3.2.3 Backend Theano

To execute computations we are using a library called Theano. Keras interacts with Theano and does weight computations on either the CPU or GPU giving a layer of compatibility and portability to our software as well as provides performance which is essential for training larger models. It uses CuDNN if it's installed to further increase performance on some nVidia GPUs.

3.2.4 Keras

Keras is a framework for creating and testing ANNs. This is the level our code directly interacts with. It provides us with implementations of ANN layers we use to build our models as well as data loaders analytics tools and more.

It can use either Theano or Tensor Flow as a backend to run its computations on the GPU or CPU. Theano then compiles the model in C++ which the CUDA SDK is compatible with.

3.2.5 Our Application

Inside our application we define models and hyper parameters which we are using to test. Our application trains a model and stops when a local minimum is identified for a set span of epochs and then saves the best model ie the local minimum. The model is then evaluated and a classification report is output.

There is methods also for loading models and to use a loaded model to classify an image input. These interfaces can be used to allow an external application to use our system.

3.3 The Components of our Application

Our code is organised in two major structures:

3.3.1 Utilities

This file contains implementations of some image processing algorithms as well as other utility functionality our program needs these are 'features' we use which aren't part of the experiment logic but enable certain experiments to be conducted.

3.3.1.1 Colour Conversion

A useful utility is the ability to convert training example between RGB colour and grayscale. We achieve this by implementing a number of conversion methods to try and extract the luminance of an RGB image. These methods range from averaging the intensity of the RGB channels at their simplest and least effective to weighted averages using the NTSC/W3C and Rec.702 colour weights for each RGB channel.

3.3.1.2 Scaling and Cropping

For this we use a library called OpenCV. Because our neural networks expect inputs to have certain dimensions we must first ensure that the pictures match. Aspect ration must first be matched by cropping trying to get the most prominent object in the cropped frame, then we scale to the desired size using OpenCV's scaling routines to match our ANN inputs.

3.3.2 Main Program

Initial prototypes were created using inspiration from code from an introductory paper written by Keiron O'Shea. [2]

Chapter 4

Results and Conclusions

This section should discuss issues you encountered as you tried to implement your experiments. What were the results of running the experiments? What conclusions can you draw from these results?

During the work, you might have found that elements of your experiments were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings?

It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant?

If you had multiple experiments to run, it may be sensible to discuss each experiment in separate sections.

Chapter 5

Critical Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

Appendices

Appendix A

Third-Party Code and Libraries

For this dissertation I have used the following libraries as stated in the design documentation:

1.1 Keras

Neural network library allowing quick model prototyping. [1]

Appendix B

Ethics Submission

2.1 Submission Number 4111

AU Status
Undergraduate or PG Taught

Your aber.ac.uk email address
thn2@aber.ac.uk

Full Name
Theodoros Nikopoulos Exintaris

Please enter the name of the person responsible for reviewing your assessment.
Rayer Zwiggelaar

Please enter the aber.ac.uk email address of the person responsible for reviewing your application
rrz@aber.ac.uk

Supervisor or Institute Director of Research Department
cs

Module code (Only enter if you have been asked to do so)
CS39440

Proposed Study Title
MMP: Deep Learning for Deep Neural Networks

Proposed Start Date
20-01-2016

Proposed Completion Date
04-05-2016

Are you conducting a quantitative or qualitative research project?
Mixed Methods

Does your research require external ethical approval under the Health Research Authority?
No

Does your research involve animals?
No

Are you completing this form for your own research?
Yes

Does your research involve human participants?
No

Institute
IMPACS

Please provide a brief summary of your project (150 word max)
Investigation of Deep Neural networks for computer vision (classification) problems.

Where appropriate, do you have consent for the publication, reproduction or use of any unpublished material?
Yes

Will appropriate measures be put in place for the secure and confidential storage of data?
Yes

Does the research pose more than minimal and predictable risk to the researcher?
No

Will you be travelling, as a foreign national, in to any areas that the UK Foreign and Commonwealth Office advise against travel to?
No

Please include any further relevant information for this section here:

If you are to be working alone with vulnerable people or children, you may need a DBS (CRB) check. Tick to confirm that you will ensure you comply with this requirement should you identify that you require one.
Yes

Declaration: Please tick to confirm that you have completed this form to the best of your knowledge and that you will inform your department should the proposal significantly change.
Yes

Please include any further relevant information for this section here:

Appendix C

Code Examples

3.1 Colour Conversion

This code enables conversion between RGB channel intensities into greyscale luminance using different conversion methods based on their respective colourspace recommendations in a variety of modes.

```
import numpy as np
from math import sqrt

def convert_set_to_greyscale(cifar_set, method=0, gamma=1.0):
        converted_set = np.empty((cifar_set.shape[0], 1, cifar_set.shape[2]
        for image_index, image in enumerate(cifar_set):
                for row_index, row in enumerate(image[0]):
                        for pixel_index, pixel in enumerate(row):
                        grey = 0.0
                        if method == 0:  # Rec.709 luminance
                                grey = (0.2126 * cifar_set[image_index, 0,
                                (0.7152 * cifar_set[image_index, 1, row_inde
                                (0.0722 * cifar_set[image_index, 2, row_inde
                        elif method == 1:  # NTSC/W3C luminance
                                grey = (0.299 * cifar_set[image_index, 0, ro
                                (0.587 * cifar_set[image_index, 1, row_index
                                (0.114 * cifar_set[image_index, 2, row_index
                        elif method == 2:
                                grey = sqrt(((0.299 * cifar_set[image_index
                                ((0.587 * cifar_set[image_index, 1, row_inde
                                ((0.114 * cifar_set[image_index, 2, row_inde
                        elif method == 3:
                                grey = sqrt(((0.2126 * cifar_set[image_index
                                ((0.7152 * cifar_set[image_index, 1, row_ind
                                ((0.0722 * cifar_set[image_index, 2, row_ind
                        elif method == 4:  # Simple mean of RGB
                                grey = ((cifar_set[image_index, 0, row_index
```

```
                        (cifar_set[image_index, 1, row_index, pixel
                        (cifar_set[image_index, 2, row_index, pixel
                else:
                        print 'Error: This is not a valid conversion
                return cifar_set.astype('float32') / 255

        converted_set[image_index, 0, row_index, pixel_index] = np.
print 'Converted ', len(converted_set), ' images to greyscale.'
return converted_set
```

Annotated Bibliography

[1] F. Chollet, “Keras,” https://github.com/fchollet/keras, 2015.

[2] R. N. Keiron O’Shea, “An introduction to convolutional neural networks,” *arXivaij*, vol. 1511.08458, 2015.

Helpful primer into convolutional neural networks

[3] Various, “Cifar,” http://www.cs.toronto.edu/~kriz/cifar.html, Feb. 2016, accessed February 2016.

Site for the CIFAR-10 and CIFAR-100 datasets.

[4] ——, “Mnist,” http://yann.lecun.com/exdb/mnist/, Feb. 2016, accessed February 2016.

Site for the MNIST dataset.

[5] ——, “Tensor flow,” https://www.tensorflow.org/, Feb. 2016, accessed February 2016.

Site for Google’s Tensor Flow machine learning framework