

Summer 2009 MAP Projects

Jerod Weinman

Abstract

This document provides a background on summer MAP projects for Grinnell students and the high expectations I have for summer students.

Contents

1	Introduction	1
2	GPU Learning	2
2.1	Background	2
2.2	Project Work	3
3	Visual Categories	3
3.1	Background	3
3.2	Project Work	4
4	Gamma Ray Astronomy	4
4.1	Background	5
4.2	Project Work	5
5	Approximate Schedule	5
6	Expectations	6
6.1	Spring	6
6.2	Summer	6
6.3	Fall and Beyond	7

1 Introduction

The general focus of my research is in probabilistic machine learning for computer vision. Because reconstructing a 3-D image from a 2-D projection is a difficult inference problem, some computational machinery is necessary. Furthermore, understanding and extracting meaning from images is a problem that has been solved by humans, but remains elusive for machines. Since it is nearly impossible to specify and hand-code models for these tasks, machines must be endowed with some amount of learning capabilities.

There are three particular projects I am hoping to work on in summer 2009

- Parallel processing with GPUs for learning character recognition
- Using category relations to improve visual category recognition
- Using machine learning to improve gamma ray astronomy

Slides from the presentation given at the departmental Thursday Extras Series can be found here:

- [Summer 2009 Research \[pdf/slides\]](#)

2 GPU Learning

The context of this project is character recognition. While this is one of the oldest problems in pattern recognition, the most general form of the problem—automated recognition that matches human performance in any situation—is still very far from solved. My primary aim is to advance information availability with automated character recognition. The primary focus of this project will be to improve performance by adding training data through massive parallelization.

2.1 Background

The application for the project, already underway, is an aid to the blind. By recognizing text in arbitrary images, we hope to improve the ability of the visually impaired to navigate. In order to correctly recognize characters, computers must, in essence, learn to read. That means they are shown many examples of characters in many fonts. Given a model with many free parameters, the machine then learns how to set these parameters such that it is likely to recognize the examples it has seen, and others like it.

It is known in the speech recognition community that many millions of training examples are required for a machine perform satisfactorily. This realization, and a corresponding data-gathering effort, has made speech recognition more successful when deployed in real-world applications, such as automated telephone operators.

Commercial optical character recognition (OCR) software for document text recognition has also seen modest success. However, it depends on many assumptions that are not warranted in other text recognition settings. One example mentioned above is scene text recognition (STR), the recognition of text anywhere in the environment, such as on store fronts, traffic signs, movie marquees, or parade banners. While superficially similar to OCR, STR is significantly more challenging because there are innumerable fonts to consider, uncontrolled viewing conditions, and minimal language context. Difficult viewing angles, shadows, occlusions, unique fonts, and lack of language context are all problems that make the typical STR problem significantly more difficult than a straightforward OCR application of document recognition.

Our current models perform at the state-of-the art on modestly constrained STR problems—95% accuracy for normal resolution and 85% accuracy at low resolution. But there is still clear room for improvement—a seven year old child makes fewer recognition errors. One important development necessary to increase performance to human level will be to drastically increase the amount of training data available to the system. However, training time is already on the order of one to two weeks. With the current methodology, any increase in training set size by multiple orders of magnitude will make timely error analysis impossible.

For a high-level view of what I have been doing in this area, please see the following:

- [Recognition Overview](#)
- [Publication Poster](#)
- [Presentation Slides](#)

For a more detailed perspective, you might look at the following papers. In particular, the introduction, experiments, and conclusions are likely to be interesting and easier to follow than the “model” sections, which you certainly can try reading:

- [Scene Text Recognition using Similarity and a Lexicon with Sparse Belief Propagation](#). In *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, in press.
- [A Discriminative Semi-Markov Model for Robust Scene Text Recognition](#). In *Proc. International Conference on Pattern Recognition (ICPR)*, Dec. 2008.

For the truly curious (or deranged), a plethora of details on the current state of the project are in my [Ph.D. thesis](#)

2.2 Project Work

The accuracy of our current text recognition model on *training* data is very high (near 100%). Therefore, we believe the capacity of our model to recognize complex characters in many fonts is great. However, one of the primary bottlenecks of improving the system is the amount of training data we are able to use. I want to improve system performance and move toward more challenging problems by radically parallelizing the training process with graphics processing units (GPU). GPUs feature many processing units on a single board and is a hardware platform perfectly suited to performing the same instructions on different data—i.e., the same learning algorithm on different examples. Since we can generate a virtually unlimited amount of training data artificially, we should be able to see an immediate gain in performance.

There is a programming language (called CUDA) for GPUs allowing developers to more easily port their existing algorithms into a parallelized version. Even though CUDA exists, there will still be open questions about the optimal approach for moving the massive amounts of data between the computer’s main memory and the more limited GPU memory. Variations to the training algorithm are possible, thus approximations that may increase throughput should be investigated.

Once the parallel training is achieved, it will then be necessary to investigate what types of variations in the training data provide the greatest improvement on an evaluation benchmark. These might include number and variety of fonts, scale/size variations, noise, contrast, rotations, and the magnitude or degree of all of these.

In addition (or as an alternative), we may also investigate the type of representation used by the model. Currently, a “flat” representation is used. Here, a character is only decomposed into the orientations of its edges, and these are recognized as a whole. As an alternative, the character may be broken up into representative parts (T-junctions, corners, straight lines, round bowls), and these parts can be detected and recognition based on the parts. This is the approach used for visual categories and it has not yet been shown that a flat model (of the sort currently in use) with sufficient training data cannot outperform or rival a parts-based hierarchical model.

I expect applicants to have a strong understanding of programming in C and systems architecture, since system throughput and bandwidth will be important considerations for maximizing training efficiency. Students with strong backgrounds in linear algebra and probability/statistics will also be at an advantage. In addition, students on this project should expect to become very familiar with the NVIDIA CUDA programming language as well as MATLAB, being prepared to handle the challenges of communicating between the two. Some design work on how best to create a data pipeline from disk to GPU will be needed, and we may decide that it is optimal to forego MATLAB altogether.

It is my desire to release any software produced in the most general form possible so that our efforts may be used to reproduce experiments or apply the technology to other problems. Therefore, a code repository (SVN) will be used to manage code, and high standards of code quality and documentation are expected.

3 Visual Categories

A closely related project is generalizing to visual categories. In this problem, many computational models have been proposed, but few take advantage of the relationships between categories. We will investigate whether performance improvements can be realized by learning not only to distinguish between cars, motor-bikes, cheetahs, and zebras, for instance, but by learning relations between their hierarchical categories, such as “wheeled transport” and “quadrupeds.”

3.1 Background

Researchers have proposed a standard training and test set for evaluating visual category recognition models. While humans are thought to recognize 30,000 object categories, this data set incorporates just 256. One reason for the limitation is the overhead involved in annotating data. In order for a computer to learn adequately general models and representations, there must be a vast amount of training data or a great deal of prior knowledge engineered into the system. Unfortunately, both “solutions” are inadequate and very hard to come by. Since training data for each category is limited, generalizations are challenging, and models tend

to either overfit the limited training data, or make weak, uncertain predictions. State of the art performance is just 35% accuracy.

You can browse the data set here: [Caltech256](#). This is a press-release discussing a model used for this problem:

- [Computer model mimics neural processes in object recognition](#)

For those who want details, a recent journal article (complete with a very informative Figure 1), is here:

- [Robust object recognition with cortex-like mechanisms](#)

This project will build-upon the work described at a high-level here:

- [Joint Detection & Recognition](#)

For a more detailed perspective, you might look at the following paper. In particular, the introduction, experiments, and conclusions are likely to be interesting and easier to follow than the “model” sections, which you certainly can try reading:

- [Joint Feature Selection for Object Detection and Recognition](#). Technical Report UM-CS-2006-054, University of Massachusetts Amherst, Oct. 2006.

3.2 Project Work

I have already performed a preliminary study of the joint learning of multiple categories. However, the images used were artificially generated and formed only a few “super categories.” To demonstrate the method as a viable strategy for more complex recognition problems, we will plan to expand the experiments to the 256 category benchmark, using a subset of these categories that form at least a dozen super-categories (e.g. wheeled transport, quadruped, etc.).

While models and software exist for such category recognition, it is not clear they are optimized for the problem at hand. Some experimentation will need to be done to be sure that the “features” used to represent the images are in fact appropriate for the task. Once this is done, the experiments will test whether awareness of super categories leads to better generalization and/or stronger predictions.

This process can be computationally time-consuming, but there is also room for parallelization. If adequate progress is made on establishing the preliminaries above, we will investigate tools for parallelizing the learning process in this model as well. In this case, the MATLAB® Distributed Computation Toolbox would be used (as opposed to CUDA for the GPU), since the development environment for the project is MATLAB®.

Students should have sufficient maturity in the discipline to teach themselves a new programming language (MATLAB®). Students with strong backgrounds in linear algebra and probability/statistics will also be at an advantage. Students who have taken Neuroscience 250, Psychology 260/360, or Philosophy 256/257 may also be at an advantage.

It is my desire to release any software produced in the most general form possible so that our efforts may be used to reproduce experiments or apply the technology to other problems. Therefore, a code repository (SVN) will be used to manage code, and high standards of code quality and documentation are expected.

4 Gamma Ray Astronomy

This project is being conducted in conjunction with Professor Charles Duke, from the Grinnell College Department of Physics. He is a member of the VERITAS gamma-ray astronomy research collaboration centered at the Smithsonian Observatory in Tucson.

4.1 Background

Astronomers have a multitelescope configuration that allows for detection of upper-atmosphere gamma-ray showers. The three main problems are:

- separate gamma-ray showers from proton-induced showers
- determine the origin of the gamma ray
- determine the energy of the gamms rays producing these showers

.All of this must be done with the four images generated by the telescope (or in this case, simulations of such images).

More information on the VERITAS project can be found here:

- [VERITAS \(Very Energetic Radiation Imaging Telescope Array System\)](#)

4.2 Project Work

Image processing will be used to separate proton-produced atmospheric showers from gamma-ray produced atmospheric showers. Current techniques are straightforward and make decisions based on simple image shape. Machine learning could be used to improve the results for automatically distinguishing gamm-ray from proton-induced showers. A large library of Machine Learning algorithms is available to the research community in an Open Source Java package called WEKA. Identifying useful features of the images and testing the performance of certain algorithms could be a key component of this research. This will require gaining some understanding of image processing and features, as well as becoming versed in several machine learning algorithms (or at least how to use them in WEKA).

Another possibility is to develop new methods for reconstructing the gamma-ray source location in the sky and the on the ground. This work would make extensive use of vector algebra and coordinate transformations using C++ language and the ROOT framework, an astronomy code library.

Depending on the results, both of these projects could result in a journal publication, with work continuing beyond the summer.

5 Approximate Schedule

This schedule largely follows that officially approved by the division. However, since other (off-campus) options have different schedules, I will need to know if you are considering other opportunities, what the schedule is, and whether you are likely to choose an off-campus opportunity if accepted. The ten weeks of summer research are currently under discussion.

Friday, 20 February 2009: Application forms due. You must submit the division-wide form in hardcopy to the division office and your responses to my questions to me.

Friday, 6 March 2009: Initial selections announced (provided that the college has approved funding).

Friday, 13 March 2009: Preliminary acceptances/rejections due.

Week of 7 April 2009: First meeting.

Unspecified other dates: Additional meetings.

Monday, 18 May 2009: Commencement.

Monday, 25 May 2009: Summer research begins.

Friday, 31 July 2009: Summer research concludes.

6 Expectations

I have very high expectations of my summer research students. Among other things, I expect my students to begin their summer research during spring semester and continue their summer research into fall semester (and sometimes beyond). By applying for summer research you are agreeing to meet these expectations if I take you on as a research student. You are unlikely to receive explicit credit or compensation for work in the spring and fall.

I also expect my students to be self-reliant. While I do my best to be around, I expect you to be able to do many things on your own or with a small group.

6.1 Spring

Topic Preparation You are expected to begin your background research during the spring. In particular, you must identify at least four papers on related projects. You are also encouraged to use the web to aid your search. Some useful resources are:

- [Google Scholar](#)
- [CiteSeer^x](#)

Some of the top related conferences to find this work are

CVPR	Computer Vision and Pattern Recognition
ICCV	International Conference on Computer Vision
ECCV	European Conference on Computer Vision
ICDAR	International Conference on Document Analysis and Recognition
ICPR	International Conference on Pattern Recognition
ICIP	International Conference on Image Processing
ICASSP	International Conference on Acoustics, Speech, and Signal Processing

and some related journals include

PAMI	IEEE Transactions on Pattern Analysis and Machine Intelligence
IJCV	International Journal on Computer Vision
TIP	IEEE Transactions on Image Processing

though there are of course many, many others. Once you have identified potentially useful resources, if you cannot find an author preprint online (they nearly always are), consult with the librarians about obtaining a copy of an article or conference paper.

Skill Preparation If your project will require a programming language (e.g., MATLAB), data language (e.g., CUDA), or library (e.g., WEKA) that you do not yet know, you are expected to begin studying that language. You need not master the language or libraries, but should develop some basic familiarity.

6.2 Summer

During the summer, you are expected to work full-time on the project (40-50 hours per week for ten weeks). This work will include regularly scheduled group meetings.

Topic Preparation For the first week of summer research, you will continue your preparation from the Spring, developing a survey of the state of the art in whatever project you've decided to undertake. You should prepare a short survey paper. This will serve as an introduction/literature review for a later paper. On the first day of the second week, you will give a public presentation of your work.

Core Research and Development For the next eight weeks of the summer, you will work on your project, using what you've learned during preparation for guidance. Some of this time may be spent developing skills. We will have a full-group meeting several times a week. Each group will present at least once per week at that meeting.

Writing For the last week of the summer (and, preferably, as you do your work), you will work on a five-to-ten page paper describing your work and placing it in the context of related work. Your paper should meet the highest standards of writing at Grinnell. If you work as part of a small group, the group need only prepare one paper. In most cases, you will be required to submit a version of this paper to a conference or journal. (I will provide significant assistance in developing the submitted version, in which case I will probably ask to be listed as a co-author.)

6.3 Fall and Beyond

Poster Presentation You will create a poster describing your work and present it at the Grinnell Science Poster Seminar (typically during parents' weekend).

Internal Public Presentation You will give a twenty-five or fifty minute presentation on your work as part of the Computer Science Thursday Extras series.

External Conference Presentation If your work is submitted and accepted to a conference, and there is funding available for you to attend the conference, you are expected to attend and present your work.

External Pew Presentation You must submit your work to the Pew Midstates Science and Mathematics Consortium Fall Symposium on Undergraduate Research in the Physical and Mathematical Sciences. You must attend the symposium (including non-CS talks and present your work (in poster or talk form) if your work is accepted. You must give at least one practice talk before going to the conference.

Acknowledgement

With thanks to Professor Sam Rebelsky for many elements of Section 6 and Professor Charles Duke for information in Section 4.