Integrating CNN with an Interactive Machine Learning System
Google Summer of Code - 2015

Nelson Isao Nauata Junior

March 22, 2015

Contact Information:

36, Castle Frank Road, Toronto
Province of Ontario
March 16th, 2015
Tel: +1 647 867 6260

LinkedIn: http://goo.gl/SMHga4
Github: github.com/ennauata
Email: ennauata@gmail.com
Skype: nelson.nauata

1 Benefits to Community and Goals

This project aims to improve the performance of the current Convolutional Neural Network being used in the prototype, by basically using three approaches: testing different possible features other than the current ones, implementing a new model to include the unlabeled data, fine tuning and optimising the new model:

Testing different features: As we can see from the current CNN model, the inputs being used consist only of 32 by 32 images of cells. In this project, we are proposing to extract different features to feed the CNN and compare the results to analyse the gains in performance. One possibility could be the inclusion of multi resolution images to learn features from multi scaled images, this method could be achieved by creating a Laplacian Pyramid for each image to train the model. More details can be found at: [1].

Another possible feature that could be extracted consists of the colours from the images by including the three channels to feed the CNN. The inclusion of color could be a little tricky, since this can vary for reasons that are not

biologically meaningful as stated by the mentor Lee Cooper, however, we could include to see how the CNN would perform.

New model: The current CNN model are being trained using only labeled data, this project propose the implementation of a different model to include unlabeled data in order to increase the performance of the classifier. The new model proposed is the Convolutional Deep Belief Networks, which represents an improvement of the regular DBN, since it can be scaled to full-sized images, due to the inclusion of the convolutions, taking into account the spatial positioning of the pixels, more details can be found at: [2].

The inclusion of the unsupervised learning in the pre-training phase has been showing to be advantageous by guiding the training of the parameters towards better local minima and serving as a regulariser to the parameters of the CNN. More details about this can be found at: [3].

Fine tuning and optimization: In this part of the project, we would be fine-tuning the CDBN with a regulariser called Dropout. This regulariser has been shown to give big improvements by randomly dropping out some hidden units with under a certain probability, avoiding potential adaptations on the training set and preventing overfitting. More details about this can be found at: [4]. Note that, this regulariser can be applied to unsupervised pre trained models, however as for the other models there would be some modification from the regular back propagation algorithm, as we can see from [5]. In addition, in this part of the project, we propose to perform a sequence of tests to optimize the hyper parameters and show the results of the project. After performing all the tests and selecting a promising model, we propose the implementation of some tweaks to speed up the training of the CDBN.

To turn the the model scalable we propose to implement the model on-the-fly by distributing jobs to extract features and feed a buffer and jobs to retrieve features from the buffer and train the model at the same time, since we could face memory issues due to the number of cells that each big image can have, hence storing all the images would be unviable. This method would increase the training speed as well, since we would be avoiding loading the same big image using OpenSlide to the memory many times, which can be very time consuming. To implement this mechanism of managing the cores of the GPUs and to implement the model, we would make use of the Caffe Framework, which allows us to distribute jobs among the GPUs and provide some implementations of Machine Learning in C++ or using the python wrapper.

2 Deliverables

$0 \sim 1$ week - Set up the programming environment.
$2 \sim 3$ weeks - Implement the extraction of the new features and report the results for comparison.
$3 \sim 4$ weeks - Implement the new model using Caffe and test in relative small

portions of data.
$1 \sim 2$ weeks - Work on the fine-tuning of the new model, test different regularisers and report results for comparison.
$1 \sim 2$ weeks - Implement the distribution of jobs, inclusion of buffers and integrate bigger portions of unlabeled data.
$0 \sim 1$ weeks - Report comparison and final results.

3 Related Work

I had the opportunity of being part of the development of the Deep Learning Application (`http://deeplearning.cs.toronto.edu`). This project is related to the prototype mentioned in the FAQ page (`cooperlab.net/GSOC_deep.html`), where a huge dataset of labeled images are used to train a Convolutional Neural Network to predict new images uploaded in the web interface or taken in the mobile apps. In this project, however, instead of the Theano, the Toronto group used the CUDAMat package to implement the Convolutional Network.
Another related project that I have being part is the application of Convolutional Neural Networks in videos for classifying activities. This project is in the state of feature extraction and modelling of the CNN.

4 Biographical Information

I am an undergraduate student at State University of Campinas finishing my undergrad at University of Toronto in Computer Science. I have been specialising myself in Machine Learning, taking specifics courses (e.g. Deep Neural Nets, NLP and AI) and working on undergraduate research projects in the Toronto Machine Learning Group (http://deeplearning.cs.toronto.edu/people).

4.1 Relevant Courses taken at University of Toronto:

CSC411H1 - Machine Learning
CSC494H1 - Project in CSC - Machine Learning for Video Analysis. Supervisor: R. Salakhutdinov
CSC320H1 - Intro Visual Compting
CSC321H1 - Neural Networks
CSC384H1 - Intro Artif Intelligence
CSC401H1 - Natural Lang Compt
CSC418H1 - Computer Graphics

4.2 Awards:

1st place in SportsHack - Hackaton placed in Toronto with a prize valued at $10,5k.
`http://www.sportshackweekend.org/`

4.3 Relevant Skills:

C++, C, Python, Java, Web development, Android and iOS, Matlab, OpenCV, R, Git, Unix, MySQL, DB2, Machine Learning APIs, Haskell, Racket, Prolog.

4.4 Projects:

Deep Learning Android
https://play.google.com/store/apps/details?id=utoronto.deeplearning&hl=en

Deep Learning iOS
https://itunes.apple.com/us/app/deep-learning/id909131914?mt=8

Roote
http://rooteapp.com/

Moses for Android and iOS

4.5 Internships:

My first internship was in IBM Research, where I worked for 1 year with back-end development, databases and research projects, which provided me my first paper publication in a conference.
I have worked in a summer internship for University of Toronto in the Machine Learning lab. I had the opportunity to work on the development of Android and iOS applications and Machine Learning algorithms, such as convolutional neural networks. Later on, I was supervised by R. Salakhutdinov in a project course for Computer Science. Since then, I have been collaborating to the lab and participating on meetings.

4.6 References:

[1] http://www.cs.toronto.edu/~tang/papers/mr_dbn.pdf
[2] http://ai.stanford.edu/~ang/papers/icml09-ConvolutionalDeepBeliefNetworks.pdf
[3] http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.308.3887&rep=rep1&type=pdf
[4] http://arxiv.org/pdf/1207.0580.pdf
[5] http://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf