Implementing Machine Learning Algorithms on GPUs for Real-Time Traffic Sign Classification

Dashiell Bodington, Eric Greenstein, and Matthew Hu
Department of Electrical Engineering, Stanford University
{dashb, ecgreens, matthu}@stanford.edu

Abstract

This paper investigates traffic sign classification, which is an important problem to solve for autonomous driving. Linear discriminant analysis and convolutional neural networks achieved an accuracy of 98.25% and 98.75% respectively when classifying eight different types of traffic signs. The CNN was implemented on a GPU for real-time traffic sign classification: testing time for the CNN on a GPU was 4 ms/image, which was 7.5x as fast as running LDA on a CPU and 60.2x as fast as running CNN on a CPU. Additionally, different types of classification errors and the effects of adding a new sign to the dataset were explored.

1 Introduction

Traffic sign recognition (TSR) is a computer vision problem that has received significant attention due to its relevance for autonomous driving, advanced driver assistance systems, and mobile mapping. The problem is split into two components: detection and classification. Road signs are designed to be easily identified by human drivers and differ based on their shape, color, icons, and text. However, traffic sign classification is made difficult for computers by varying illumination and weather conditions, occlusions, and subsets of signs that are similar to each other.

Many researchers have focused on improving the accuracy of sign detection and recognition, but less work has focused on improving processing time through different hardware implementations. This paper will demonstrate how machine learning algorithms in multi-core GPU architecture can reduce computing time compared to CPU methods to enable real-time traffic sign classification at typical video frame rates of 30 frames per second.

In this paper the impact of computer architecture (CPU vs. GPU), algorithm, and training set size on the accuracy and speed of traffic sign classification will be explored. First, we will discuss the methods we used, namely the dataset, features, and algorithms, as well as give a brief background of the present research. Results and discussion will follow the methods section. Finally, a conclusion and vision for future work will be presented.

2 Methods

In literature, many different features and machine learning algorithms are proposed for traffic sign classification. Some commonly used features include histograms of oriented gradients (HOG), Haar-like features, and color histograms [1, 2]. Linear discriminant analysis (LDA), support vector machines (SVM), neural networks, subspace analysis, ensemble classifiers, slow feature analysis, kd-trees, and random forests have been investigated for traffic sign classification [3]. This paper will examine two classification algorithms- LDA and convolutional neural networks (CNNs). LDA was chosen as a benchmark algorithm, as it is fairly accurate and computationally inexpensive. CNNs give some of the highest accuracies reported for TSR, but they are computationally intensive. A few groups have run CNNs on GPUs for image classification with improved speeds [4, 5, 6].

2.1 Dataset, Preprocessing, Hardware

The LISA (Laboratory for Intelligent & Safe Automobiles) Traffic Sign Dataset is a set of annotated images

Figure 1: Image from the LISA dataset (left) and the eight signs classified (right). The signs are: added lane, stop ahead, speed limit 25, speed limit 35, stop, pedestrian crossing, merge, and keep right.

and videos containing traffic signs [7]. It is comprised of over 6,000 frames that contain over 7,000 signs of 47 different types. Sign dimensions vary from 6x6 to 167x168, and are recorded from different perspectives in grayscale and color. The wide variation in this dataset makes it ideal for realistic applications, but for the purposes of our classification, images were each cropped to a square area including only the sign of interest. All images were converted to grayscale and scaled to 32x32 pixels.

We focused on classifying the eight most common traffic signs in the dataset: pedestrian crossing, stop, signal ahead, added lane, keep right, merge, speed limit 25, and speed limit 35 signs (see Figure 1). We trained our algorithms on 200 images of each sign and then tested them on an additional 50 images, resulting in a training set size of 1600 images and a test set size of 400 images. Images that were taken in the same time sequence were grouped and then the groups were randomly divided into the training and test sets, thus preserving independence between the sets.

An Intel i7-3770K processor and Nvidia GeForce GTX 780 GPU were used to run all of the algorithms.

2.2 Histograms of Oriented Gradients

Histograms of oriented gradients have been shown to be a good feature for classifying traffic signs, and are relatively fast to compute. The HOG descriptor utilizes the image's edge orientations and distribution of intensity gradients in order to describe each image. The HOG features were calculated using the 32x32 pixel raw images, with the cell size, block size, and block overlap tailored for performance. These HOG features, computed using MATLAB, served as the inputs into the LDA classifier.

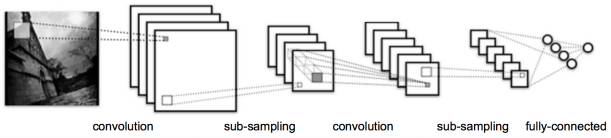

Figure 2: Structure of the CNN implemented [8]

2.3 Linear Discriminant Analysis

LDA is a common classification technique. It models the class conditional probabilities using multivariate normal distributions and assumes the classes have a common covariance matrix. Ultimately, LDA training solves for the directions that will represent the axis that maximize the separation between multiple classes. While simple, LDA often gives very good results in traffic sign classification, with accuracies of approximately 95% [3]. In this report, MATLAB was used to run the LDA on a CPU.

2.4 Convolutional Neural Networks

CNNs have achieved several state-of-the-art performances in traffic sign classification. In the second stage of the German Traffic Sign Recognition Benchmark (GTSRB) held at IJCNN 2011, two groups using CNNs obtained classification accuracies upwards of 98%, which is comparable to human accuracy [3]. CNNs are composed of three different types of layers: convolutional, subsampling, and fully-connected layers. Figure 2 shows the structure of the CNN used in this paper. Raw grayscale images, scaled to 32x32 pixels, were input into the CNN. Iterative training is done by feeding images forward through the network, and optimizing weights in the various layers through stochastic gradient descent.

In this report, CNNs were implemented using Caffe, which is a publicly available deep learning framework created by Yangquing Jia at UC Berkeley [9]. Caffe supports several tools for accelerating CNN calculations, such as Nvidia's CUDA parallel computing platform; it can be further accelerated with Nvidia's cuDNN, a GPU-accelerated library developed specifically for neural networks and deep learning. Additionally, it is easy to switch between CPU and GPU calculations within Caffe, which suits the purpose of this paper.

3 Results and Discussion

Table 1 gives a summary of the speed and accuracy results for the LDA and CNN on CPU and GPU.

Model	Accuracy (%)	Total Training Time (s)	Testing Time (ms/image)
LDA	98.25	13	30
CPU CNN	98.75	783	42
GPU CNN	98.75	51	4

Table 1: Comparing the speed and accuracy of LDA and CNN on CPU and GPU

Both the LDA and CNN are highly accurate classifiers, achieving accuracies of 98.25% and 98.75% respectively. The LDA results are superior to the 92%-95% accuracies reported in literature, and the CNN has similar accuracy to what is reported in literature [3]. This could be due to differences in the training and test sets.

In terms of training time, running LDA was 3.9x as fast as running the GPU CNN, and 60.2x as fast as running the CPU CNN. For testing time, the GPU CNN was the fastest, running 7.5x as fast as LDA and 10.5x as fast as the GPU CNN. The 4 ms testing time using the GPU CNN is fast enough for real-time classification. However, running a detection algorithm before the recognition algorithm, which is needed to solve the entire problem of traffic sign recognition, would take significant time. The overall speed of the GPU CNN is due to the parallel processing capabilities of the GPU being well suited for the task of training and testing neural networks.

To identify the types of mistakes the algorithms made, confusion matrices were analyzed. Since the optimized algorithms made few errors, the confusion matrices for the LDA and CNN shown in Figure 3 are from less optimized versions that show interesting examples of errors. In the confusion matrix, the columns represent the predictions made by the model while the rows represent the true classification of the image.

For the LDA, there are two groups of signs that are commonly misclassified. There is a cluster of errors in the top left corner of the plot, which consists of the added lane, pedestrian crossing, signal ahead, and merge signs. These four signs all have the same diamond shape. The second cluster of errors lies on the bottom right corner of the confusion matrix. The LDA algorithm had difficulty classifying the two speed limit signs, which are extremely similar in both their shape and inner symbol.

From these clusters of errors, we can see that sign shape is a fairly important aspect for LDA classifiers. The LDA algorithm separated the rectangular signs from the diamond signs accurately. There are also few errors associated with the stop sign, which has

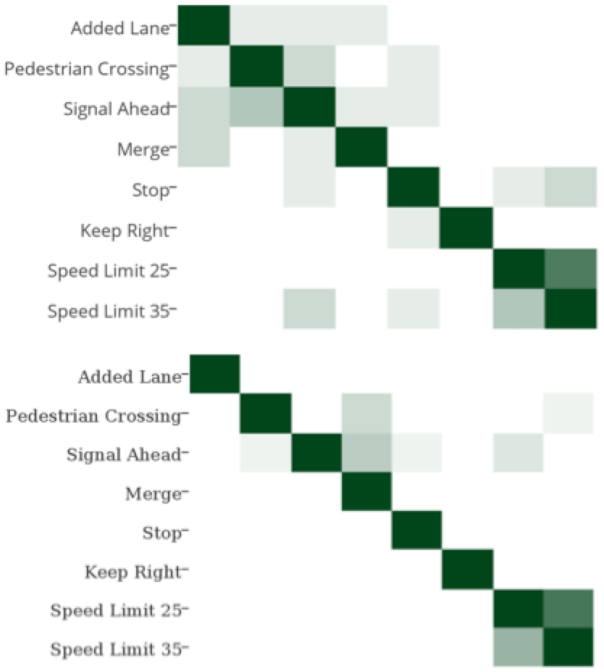

Figure 3: Confusion Matrices for LDA (top) and CNN (bottom)

a unique shape. The difficulty of separating signs with similar shape is expected given the fact that we used HOG as a feature. HOG feature are descriptors that give a distribution of edge directions, so it makes sense that HOG features contain information about sign shape but possibly miss other types of information that would help differentiate between signs of the same shape.

The confusion matrix for the CNN shows some of the same misclassification tendencies as the LDA model. Again, the most common errors are with the two types of speed limit signs and the diamond-shaped signs. Overall, misclassifying speed limit signs accounts for 60% of the misclassification error in the LDA and CNN algorithms. The majority of the remaining error comes from the misclassification of diamond-shaped signs.

Figure 4 shows a few examples of signs that are difficult to classify. The leftmost image is only 18x18 pixels and demonstrates the limited information sometimes available to the models. The LDA model classifies this as a speed limit sign, but the CNN model, based on the same training data, is able to correctly classify it as a stop sign. The center and rightmost images show how poor lighting and obstructions can make classification extremely difficult. Similarly, in a blurred image (many of which are

Figure 4: Example of misclassified signs due to small size (left), poor lighting (center), and occlusions (right)

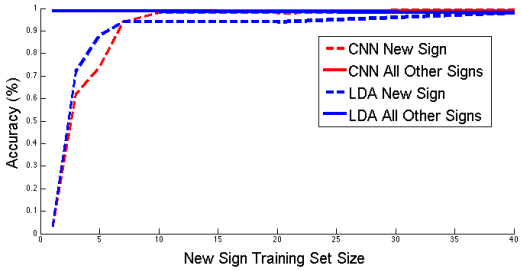

Figure 5: Accuracy vs. New Sign Training Set Size

contained in our dataset), it is easy to imagine how a merge and added lane sign or a pedestrian crossing sign might be confused given their shapes and similar content.

Some of these misclassification errors can be solved using the sequence of images that would be obtained while driving. In a typical driving scenario, a sign will start small in the field of view of the recording device and may not be classified accurately, but as the image grows closer, the amount of information available increases and the classification accuracy should be higher. Our analysis confirms the expectation that larger size images are better classified. The median side dimension of our test set of sign images is 34 pixels, but for images that are misclassified, the median image side dimension is 26 pixels. This amounts to 41% less information considering the total number of pixels per image. Additionally, lighting, obstructions, and blur may be present in some images of a specific sign but not others.

While the LDA and CNN sometimes misclassified the same image, there are examples for both cases where the LDA or CNN outperform one another. This can be due to the properties of the algorithm or because of the optimality of the training set. There is a very strong dependence of the models' accuracy on the similarity between the training and testing datasets, and the size and diversity of the training dataset.

Next, we explored how adding a new sign to the dataset affected classification accuracy. We added images of the turn right sign to our test dataset and measured the accuracy of our new classifier relative to the size of the training set of the new sign. The goal of this test was to determine how well our algorithms would perform with a small training set and determine a rough threshold for the number of images required to adequately retrain our classifier. Additionally, we were interested in how a new sign affected the classification of the other signs.

For this experiment, we created a training set ranging from 1 to 40 images and a test set of 50 images of the new sign. We added the new sign datasets onto the original training and test sets and then reran the two algorithms. Figure 5 plots the accuracy vs. new sign training set size for both the LDA and CNN.

From figure 5, we can see that both the LDA and CNN learn to classify the new sign fairly quickly. Within 10 images, both algorithms were able to correctly classify over 90% of the new test images. By the time we reached 25 training examples, we were able to consistently achieve over 95% accuracy for both algorithms on the new sign. This shows that adding a new sign to an existing classifier does not take a significant number of training examples to achieve high accuracy.

We also discovered that adding a new sign did not significantly affect the accuracy of the classifiers on the original testing set. The CNN received zero additional errors and the LDA classifier received less than 1% additional error on the original test set images when adding images of a new sign to the training and test sets. These values remained fairly constant as we continued to increase the size of the new sign training set.

In terms of application, this result is useful. A fully trained classifier can learn a new sign and begin achieving high accuracy on it fairly quickly. Additionally, we can be confident that adding the new sign will minimally affect the accuracy of the rest of the signs. These results would be applicable if a new traffic sign is introduced to a city, for example. It is also impressive considering that the right hand turn sign added is another diamond-shaped sign, which caused numerous errors in the LDA and CNN algorithms previously.

4 Conclusion

Overall, this project provided interesting insight into the problem of traffic sign recognition, and our classification algorithms performed well. Though the classifiers were applied to a smaller dataset with

fewer signs, both LDA and CNN classifiers performed nearly as well as other researchers' results, having accuracies of 98.25% and 98.75% respectively.

Not surprisingly, the CPU implementation of CNN classification was the slowest, and CNN training was the most time consuming process of either method. Running the GPU implementation of our CNN, however, was significantly faster than either CPU methods. This demonstrates the usefulness of large parallel computing for classification and reveals the CPU/GPU speedup we can expect, even on less powerful devices. The GPU CNN testing time of 4ms/image is fast enough to be applied to a real-time video feed.

Though each classification method makes some errors, these errors follow predictable trends and are highly dependent on the training of the algorithms. It is most common for the classifiers to misclassify within common sign shapes, such as different speed limit signs and different diamond shaped signs. This is expected, especially given small pixel dimensions, blurry images, difficult lighting conditions, and occlusions.

When the training data is representative of the testing conditions however, both models can achieve accurate classification on a small number of samples. Adding the turn right sign to the established 8-sign classifier showed that even with a small training set for the new sign, the classifiers were able to accurately classify it. Furthermore, the classifier did not make many additional errors on the original sign types.

With all the challenges of dealing with a small dataset and small, blurry, and occluded images, the accuracy and speed achieved by both algorithms is impressive. These classification results offer insight into successful traffic sign recognition, and are a significant step towards complete recognition when applied in conjunction with detection methods.

5 Future Work

Without changing the models we use, performance could be improved by improving the robustness of our training dataset. This can be done by simply obtaining more images of the same types of signs, or by manipulating the current dataset. Other research groups have shown improved performance by including translated, scaled, or rotated variations of images in their training dataset.

Higher accuracy may also be possible with further optimization of the networks. Adding layers to the CNN of different types and connections, or adding features to the LDA model may increase accuracy.

Temporal and movement information can also be useful in tracking signs across the recorded field so that multiple samples of the same sign can be captured to decrease noise and uncertainty.

Detection, which is the isolation of a traffic sign in a scene, must be done to solve the full problem of traffic sign recognition, but it also complicates the process significantly. We attempted to implement this, but were unable to do this given time constraints. One common practice is to use regional analysis to estimate which portions of an image may contain a traffic sign, and then to classify these areas using a classifier.

Acknowledgement

We would like to thank Milad Mohammadi for his guidance on the project.

References

[1] Dalal, N., and Triggs, B. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[2] Viola, P., and Jones, M. "Robust real-time face detection." International journal of computer vision 57.2 (2004): 137-154.

[3] Stallkamp, J., et al. "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition." Neural networks 32 (2012): 323-332.

[4] Ciresan, D., et al. "A committee of neural networks for traffic sign classification." Neural Networks (IJCNN), The 2011 International Joint Conference on. IEEE, 2011.

[5] Mussi, L., Cagnoni, S., and Daolio, F. "GPU-based road sign detection using particle swarm optimization." Intelligent Systems Design and Applications, 2009. ISDA'09. Ninth International Conference on. IEEE, 2009.

[6] Oh, K., and Jung, K. "GPU implementation of neural networks." Pattern Recognition 37.6 (2004): 1311-1314.

[7] Møgelmose, A., et al. "Vision based Traffic Sign Detection and Analysis for Intelligent Driver Assistance Systems: Perspectives and Survey," IEEE Transactions on Intelligent Transportation Systems, 2012.

[8] Lisa Lab. "Convolutional Neural Networks." http://deeplearning.net/tutorial/lenet.html, 2014.

[9] Jia, Y. et al. "Caffe: Convolutional Architecture for Fast Feature Embedding." http://caffe.berkeleyvision.org/, 2013.