Optimizing Neural Networks using Deformable Part Models

Robert Griswold and Ryu Muthui
CSS 487: Computer Vision
December 7th, 2016

1. Introduction

This document provides the overview for our computer vision final project. The following sections outline the goals, accomplishments, results, lessons learned, potential future work, and works cited for the project.

2. Project Goals

The original inspiration for the project was a curiosity of how objects could be identified within video feeds and image sets with high accuracy and speed. For example, a real-world application for this could be a video feed that monitors cars on a stretch of road that can identify each make and model of vehicle that drives along it. Shape recognition can currently identify a basic class of object in a video frame in real-time, but a neural network would be unable to accurately and timely achieve the same result.

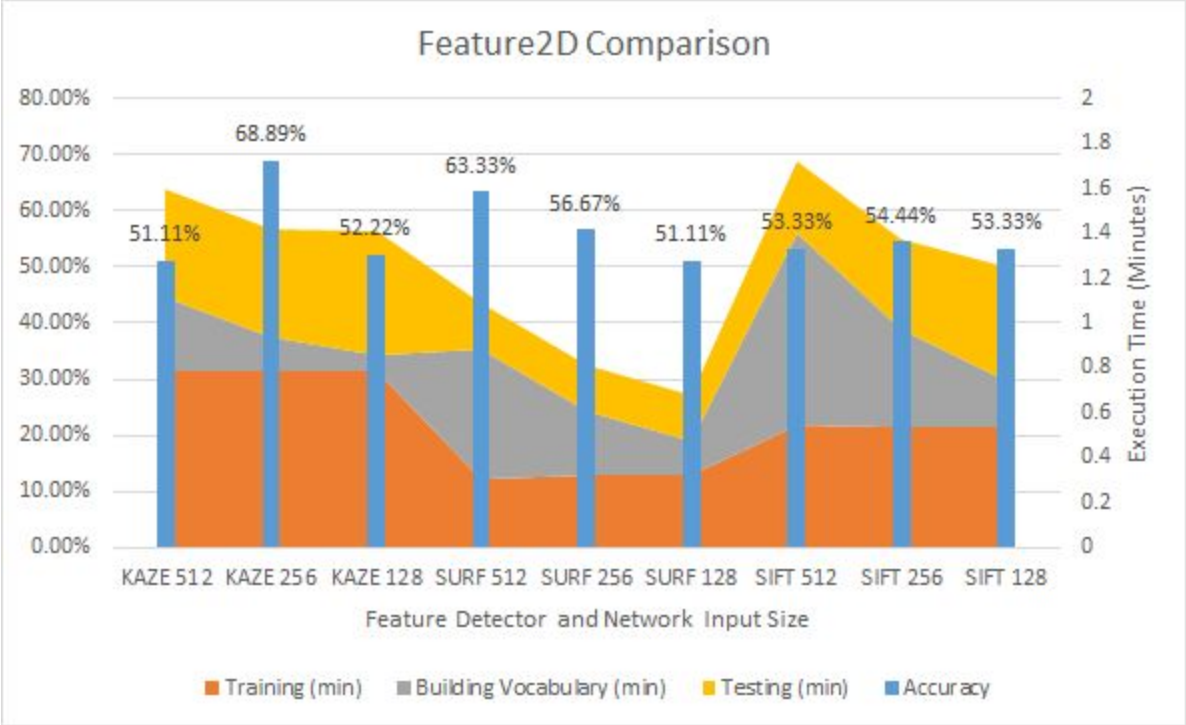
To achieve our goal, we first focused on researching ways to remove confounding information in a search image for a neural network. Our next focus was an optimization of the neural network so that it could be run in real-time. Ultimately, our goal evolved into making a neural network robust enough to be able to be used in real-time computer vision applications.

3. Accomplishments

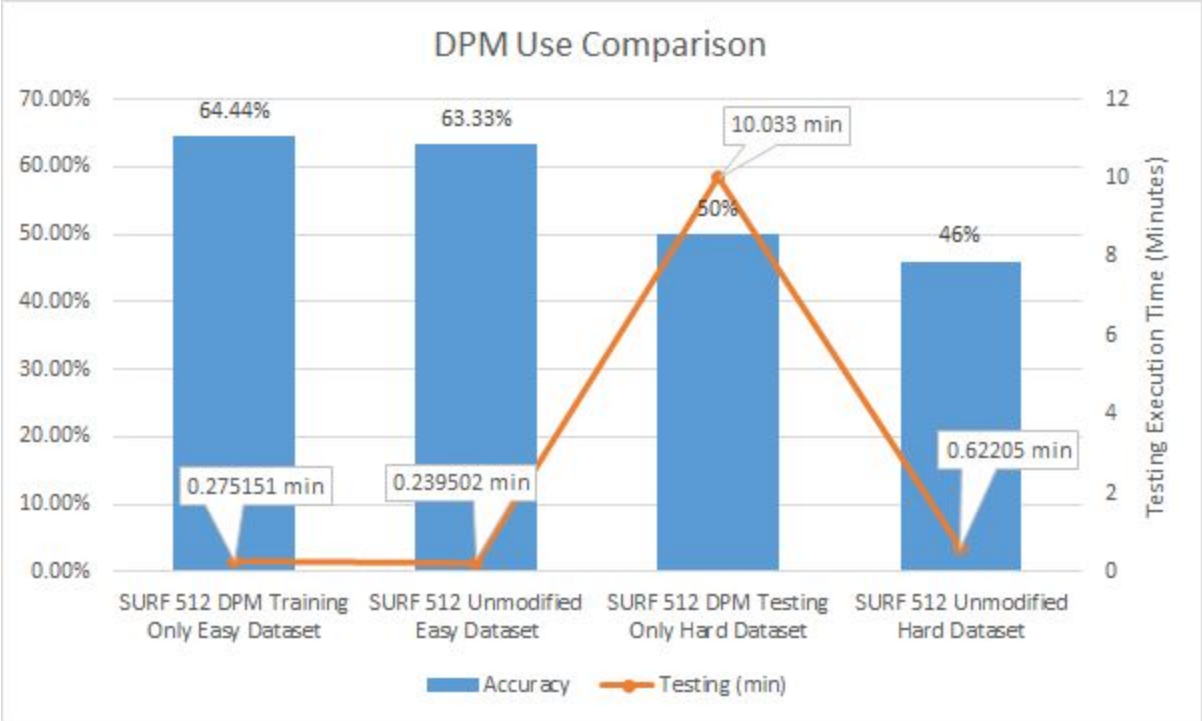
Starting from an example OpenCV application written by Abner Matheus, we improved upon the neural network application by implementing the following:

- We rebuilt the OpenCV library from source enabling the ability to switch from debug mode to release mode as well as inclusion of the OpenMP API for shared-memory parallel programming. Being able to switch from debug mode to release mode significantly reduced the execution time. On our small image set, the difference shifted from hours to minutes.
- We implemented the ability to load the serialized neural network from file allowing us to create a large neural network and iterate changes to it without having to rebuild the neural network data each time.
- We resolved the example code's inability to change feature detectors supporting a simple configuration setting to switch between SURF, SIFT, and KAZE algorithms.
- We altered the example code's confusion matrix function to support a n x n matrix so that we could train and test a neural network with more than two classes.
- We implemented a way to evaluate the neural network by converting the raw image output of the neural network into a user-readable, color coded, confidence level.
- We implemented functionality to use a Deformable Part Model for use in both training and testing with a provided pre-trained cascade xml model.

4. Results



The Feature2D graph compares feature detectors and neural network sizes in respect to execution time and accuracy.



The DPM use comparison graph compares the accuracy and testing time for the SURF feature detector using DPM for testing and DPM for training on two different datasets.

5. Lessons Learned

We expected that DPM optimization would improve the speed of testing the image sets. However, the DPM algorithm appears to be unoptimized and or the current DPM models are poor.

Additional challenges that we faced during the project include:

- Lack of data specific to our needs largely shaped the direction of the project.
- Our neural network optimization relied primarily on existing DPM data that our code could not produce.
- Much of the code relied on various libraries that all required unique ways to compile, install, and utilize.
- Finding example code that is compatible with the latest version of OpenCV required investigation into what changed between versions so that the example code could be updated and utilized for our purposes.

6. Future Work

Implement more finite detection of features, for example in car object detection, it can identify the make, model, year, and color and not just that it is a ‘car’.

7. References

OpenCV repository with example code for dpm use:
https://github.com/opencv/opencv_contrib/tree/master/modules/dpm

OpenCV repository, data files for haar cascades xml:
https://github.com/opencv/opencv/tree/master/data/haarcascades

Git repository by Andrews Sobral, vehicle detection using haar cascades car xml::
https://github.com/andrewssobral/vehicle_detection_haarcascades

OpenCV repository, data files used in our code for dpm:
https://github.com/opencv/opencv_extra/tree/master/testdata/cv/dpm/VOC2007_Cascade

OpenCV documentation on how to use cascades classifier, was useful for video set up:
http://docs.opencv.org/3.1.0/db/d28/tutorial_cascade_classifier.html

Blog by Abner Matheus, our project based on improving this neural network:

http://picoledelimao.github.io/blog/2016/01/31/is-it-a-cat-or-dog-a-neural-network-application-in-opencv/

Test data and image from the following databases:
Parking-Lot dataset that focus on moderate and heavily occlusions on cars in the parking lot scenario provided by B. Li, T.F. Wu and S.C. Zhu
http://www.stat.ucla.edu/~boli/projects/context_occlusion/context_occlusion.html
http://host.robots.ox.ac.uk/pascal/VOC/

GRAZ University of Technology Database by Prof. Axel Pinz (for Bikes, cars, people)
http://www.emt.tugraz.at/~pinz/data/GRAZ_02/