Kadie Clancy
Pedestrian Detection

Compressed HOG features for Image Classification

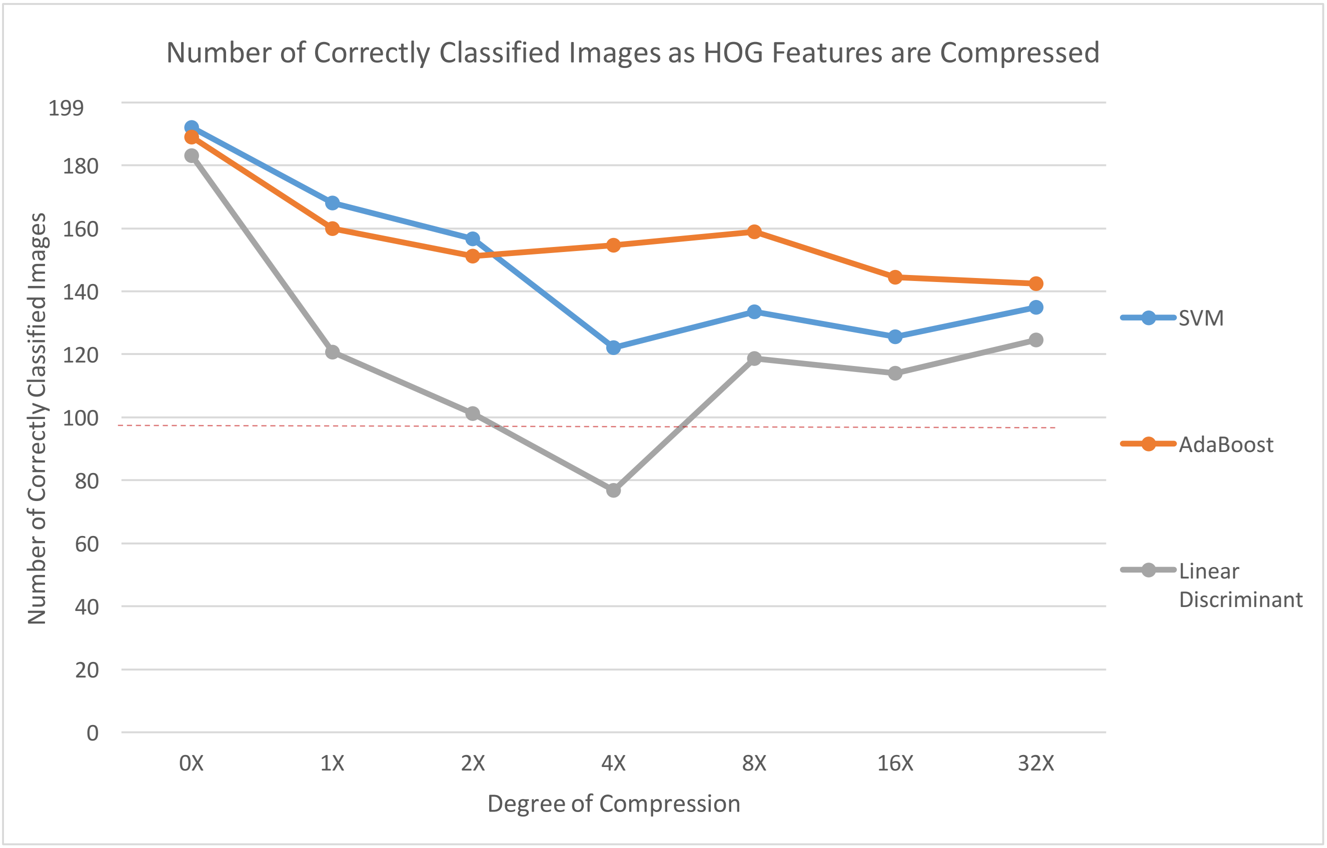
I created an image classifier using features from MatLab's Computer Vision System Toolbox and Statistics and Machine Learning Toolbox that can classify an image as either pedestrian or non-pedestrian. I created the initial system using SVM, AdaBoost (with a Linear Discriminant weak learner), and Linear Discriminant Analysis as classifiers to compare performance trained on the image's HOG feature vectors. Since the HOG feature vectors used to classify images can be significantly long, it proves efficient to compress these vectors for practical applications such as computational speed or hardware purposes. I explored whether or not boosting algorithms can restore the performance of the image classifier after the HOG feature vectors have been compressed.

Initially, I created a benchmark on which to compare performance of the compressed feature vectors. I created code that reads in sets of testing and training images. These images feature either pedestrian or non-pedestrian objects from the INRIA Person database, the MIT Pedestrian Database, and the CalTech Image Collection. Non-pedestrian images included bikes, cars, and motorcycles from different angles. Since the purpose of this code was to classify the entire image into a single category, I cropped the images so the main focus of each was either the pedestrian or non-pedestrian object. After the images are read in, each one is resized and preprocessed to remove noise artifacts. The testing set contained 199 images, and the training set contained 451 images.

Histogram of Oriented Gradient features are a well-established method for pedestrian detection and prove simple enough to implement in MatLab for the scope of this project. Also the relative speed of extracting HOG features, as compared to neural networks, is ideal due to the fact that the classifiers needed to be run multiple times to compare average performance (for reasons that will be addressed later). To extract HOG features, each image in the testing and training set is divided into 8 by 8 pixel “cells.” For each cell, a one dimensional histogram of gradient directions is computed over the pixels in each cell. These histograms are contrast normalized to account for variance due to illumination or shadowing over several cells called “blocks.” These histogram values are then converted into a vector for each testing or training image. HOG feature vectors are extracted, stored in a matrix, and assigned either a pedestrian or non-pedestrian label separately for the testing and training sets.

This matrix of HOG feature vectors for training images is used to train the AdaBoost, SVM, and Linear Discriminant classifiers in parallel. These classifiers then separately and independently predict labels for the matrix of HOG features for the testing set. I employed these classifiers in my system for particular reasons. Linear SVM is the standard classifier implemented in combination with HOG features due to simplicity and speed of the algorithm. AdaBoost is a popular boosting algorithm and algorithms in this family improve in accuracy the closer to random guessing their weak learner performs. I predict that the more and more compressed the feature vector becomes, the poorer the performance of the weak learner will become and, therefore, the better the boosting algorithm will perform. I employed Linear Discriminant as a classifier as it is the weak learner employed in AdaBoost and I predict that as the accuracy of the Linear Discriminant Analysis classifier degrades, the accuracy of AdaBoost will improve. Linear Discriminant Analysis (LDA) is a predictive algorithm that models based on t look for linear combinations of variables which best explain the data. LDA explicitly attempts to model the difference between the two classes of data, in the case of this pedestrian classifier. Before any compression is implemented, the SVM model has an original accuracy of 96.48%, the AdaBoost model has an accuracy of 94.97%, and the Linear Discriminant model has an accuracy of 91.96%.

Next, I implemented Bernoulli projection to compress the HOG feature vectors to see how the performance degrades. Bernoulli projection compresses the original matrix in the following manner. A random n by m matrix is generated with entries having an equal chance of being a 1 or -1 with n being the number of HOG feature vectors in the original set and m being the desired size of compression. The original matrix is then multiplied by the random matrix to compress to the desired vector length. Because of the nature of the algorithms stated above, I originally predicted that the AdaBoost classifier would outperform the other two classifiers. The HOG features of both the testing and training set are compressed before they are used by the classifiers. Since Bernoulli projection will produce a different compressed vector each time, the classifiers will be trained on different features. This means that the the accuracy will vary with each run. For this reason, at each compression stage I used the average number of correct predictions across ten runs for comparison. The original HOG feature vectors have a length of 1,008. I compressed the vector 1X (to the original length of 1,008), 2X (length of 504), 4X (length of 252), 8X (length of 126), 16X (length of 63), and 32X (length of 31).



The graph above shows how the number of correctly classified images degrades from the original accuracy as the HOG features become more and more compressed. AdaBoost performed better than both SVM and Linear Discriminant for 4X compression and all subsequent compression stages. The AdaBoost algorithm works best when its weak learner performs close to random guessing. AdaBoost's accuracy can be attributed to the fact the Linear Discriminant performed poorly on the compressed vectors. At 32X compression, AdaBoost has a 76.6% accuracy on average. While AdaBoost did not restore the original uncompressed classification accuracy, it seems promising that the accuracy can be further increased.

Kadie Clancy
Pedestrian Detection

In the future, I hope to extend this project to use Machine Learning algorithms to create an object detection system. Specifically, these “objects” will be pedestrians, which have proved challenging to identify in the field of Computer Vision due to the high variance of poses a human can take [3].I have created code that uses a sliding window technique to run the SVM classifier employed in this Image Classification system over and image segmented into windows of varying size and shape. The system is able to take an image of a scene as input and output that image with visual boundaries marking each pedestrian detected in that image. The current implementation has issues with overlapping windows of “detected humans” and the question of ideal window sizes.