TRAFFIC SIGN DETECTION AND CLASSIFICATION

Ryan Trumpore, Sagar Kukreja
Computer Engineering Department
Rochester Institute of Technology

Abstract

Various advancements have been made in the field of object recognition to detect and recognize the objects on the fly. The latest advancement in object recognition includes machine learning and deep learning techniques that works on multi-layered convolutional neural network to detect and recognize objects.

1. INTRODUCTION

The work presented in this project deals with the object recognition and classification. It deals with traffic sign detection and classification. The critical job in self-driving cars is to recognize and detect the traffic signs and take particular action based on that. In this project, we have worked on detection and classification of traffic signs using two different classifiers, namely Support Vector Machines(SVM) and a pre-trained Convolutional Neural Network (CNN) i.e. AlexNet and fine-tuned it to meet our requirements.

2. METHODS

Two different classifiers and feature extraction methods have been used in this project. The first one being HOG and SVM. We have used Histogram Of Oriented Gradients (HOG) for feature extraction. The technique counts occurrences of gradient orientation in localized portions of an image. We then feed the SVM with the HOG feature vector of all training set images and then tests the efficiency for other images. For extracting HOG features we used *extractHOGfeatures* function of the MatLab and for SVM we used the *fitcecoc* function from the Statistics and Machine Learning Toolbox™ to create a multiclass classifier using binary SVMs.

The other classifier used was pre-trained CNN, AlexNet. We used transfer-learning for this method. Fine-tuning a network with transfer learning is much faster and easier than training from scratch. We can quickly transfer learning to a new task using a smaller number of training images. The CNN has 25 layers in total. For fine-tuning, we transfer the layers to the new task by replacing the last three layers with a fully connected layer, a softmax layer, and a classification output layer. Then, specified the options of the new fully connected layer according to the new data. We then set the fully connected layer to be of the same size as the number of classes in the new data. To speed up training, we increased 'WeightLearnRateFactor' and 'BiasLearnRateFactor' values in the fully connected layer. We then classified the test images using *Classify* function of MatLab Neural Network toolbox.

3. EXPERIMENTAL SETUP

In order to properly measure the difference in detection rates, a sample set of images was collected from the BelgiumTSC[3] Test and Training Dataset. This database contains a 61 classes of traffic signs and staggering 40,000 images, including training images as well as test images of all sorts of signs.

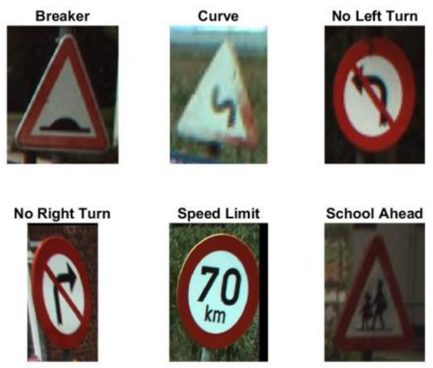


Figure 1: Sample Images

Before feature extraction for SVM and AlexNet, some preprocessing was done on images so as to fit the model. For SVM, all the images were resized to one size [180*193] selected randomly. Also, the images were converted to gray scale and binarized to compute HOG features. For HOG features, cell size of [4*4] was taken. A HOG feature visualization of a training image is as shown in figure-2.

For AlexNet all the images were resized to [227* 227] as the AlexNet is originally trained on images of this size. Each layer of a CNN produces a response, or activation, to an input image. However, there are only a few layers within a CNN that are suitable for image feature extraction. The layers at the beginning of the network capture basic image features, such as edges and blobs. To see this, visualize the network filter weights from the first convolutional layer. This can help build up an intuition as to why the features extracted from CNNs work so well for image recognition tasks. The sample feature image for AlexNet is shown in figure-3.

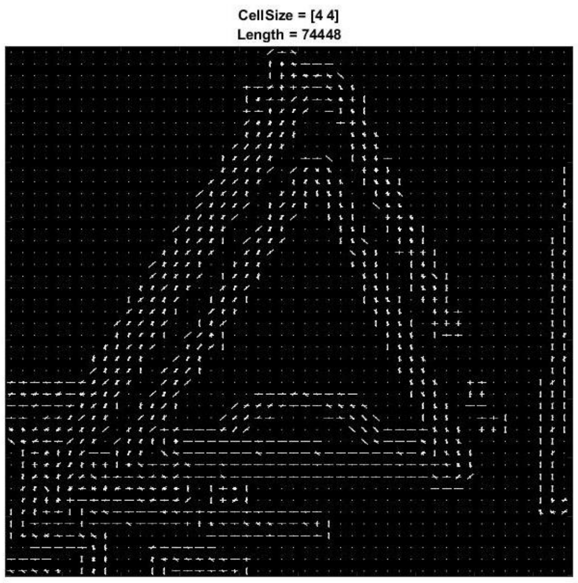


Figure 2: HOG Features for Speed Breaker class image

With both environments, we had the ability to easily utilize multiple training labels. As a result, 6 classes were chosen: Breaker, Curve, No Left, No Right, Speed Limits, and School Ahead. These signs featured two distinct shapes, circles and triangles, with differing shapes inside these. These samples, seen in Figure-1, provides two sets of images in regards to shape and patterns.

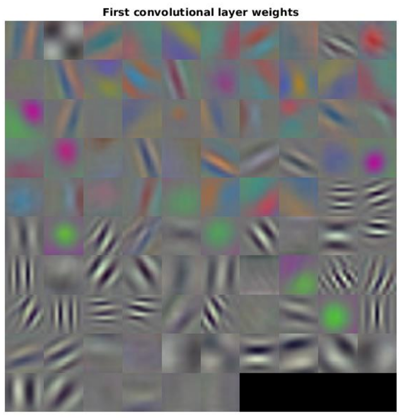
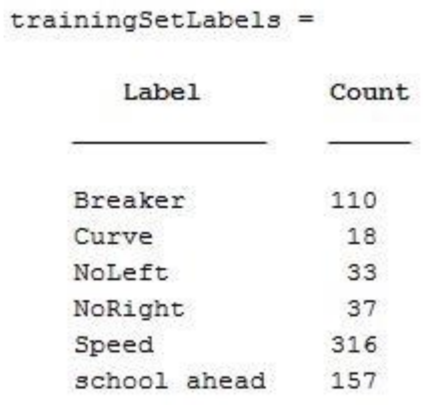


Figure-3 Alex-Net Convolution Layer Responses

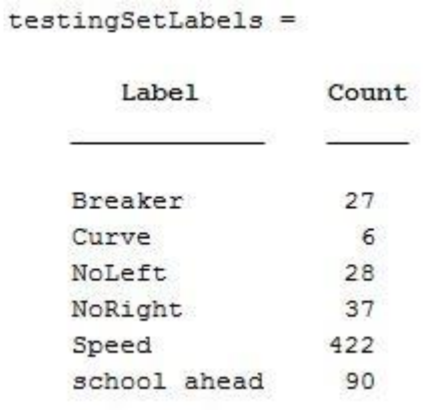


```
trainingSetLabels =

      Label          Count
    ____________     _____

    Breaker          110
    Curve             18
    NoLeft            33
    NoRight           37
    Speed            316
    school ahead     157
```

Figure 4: Training Label Count



```
testingSetLabels =

      Label          Count
    ____________     _____

    Breaker           27
    Curve              6
    NoLeft            28
    NoRight           37
    Speed            422
    school ahead      90
```

Figure 5: Testing Label Count

Distribution of test and training images among different classes is shown in Figure-4 and 5 above.

Finally, a confusion matrix is generated for both the classifiers consisting of predicted labels and test labels.

4. RESULTS

4.1. HOG-SVM

The Table-1 shows the confusion matrix. The columns of the matrix represent the predicted labels, while the rows represent the known labels. For this test set, curve sign is often misclassified as speed sign, most likely due to their similar orientation in HOG features. Training with a more representative data, which contain thousands of images for every class, is likely to produce a better classifier compared with the one created using this data set.

Traffic Signs	Breaker	Curve	NoLeft	NoRight	SchoolAhead	Speed
Breaker	27	0	0	0	0	0
Curve	1	0	0	0	0	5
NoLeft	0	0	22	1	5	0
NoRight	0	0	0	33	4	0
SchoolAhead	0	0	0	0	421	1
Speed	2	0	0	0	0	88

Table 1: HOG-SVM Confusion Matrix

Table:2 Shows the confusion matrix in percentage form.

Traffic Signs	Breaker	Curve	NoLeft	NoRight	SchoolAhead	Speed
Breaker	1	0	0	0	0	0
Curve	0.166667	0	0	0	0	0.833333
NoLeft	0	0	0.785714	0.035714286	0.178571429	0
NoRight	0	0	0	0.891891892	0.108108108	0
SchoolAhead	0	0	0	0	0.997630332	0.00237
Speed	0.022222	0	0	0	0	0.977778

Table 2: HOG-SVM Confusion Matrix (Percentage)

4.2. AlexNet

Table:3 Shows the confusion matrix in percentage form for AlexNet transfer learning

Traffic Signs	Breaker	Curve	NoLeft	NoRight	SchoolAhead	Speed
Breaker	1	0	0	0	0	0
Curve	0	0.166667	0	0	0	0.833333
NoLeft	0.035714	0	0.964286	0	0	0
NoRight	0	0	0.621622	0	0.378378378	0
SchoolAhead	0	0	0.004739	0	0.988151659	0.007109
Speed	0.011111	0	0	0	0	0.988889

Table 3: Alex Net Confusion Matrix (Percentage)

5. DISCUSSION

With the HOG-SVM system fully running, the system showed a surprisingly high rate of properly classifying the signs encountered. The only main issue was with the "curve" sign, as it almost always identified itself as a speed sign. However, aside from this nearly every sign had a 80% or greater success rate. With more training images, these signs would be identified with an even greater success rate. Curve, the sign with the worst success rate, only had 18 training images. No Left and No Right, with 78% and 89% rates, also only had 33 and 37 success rates respectively. However, the remaining images had over 100 training images, and all had over 97% success rates.

In comparison, the AlexNet code had similar failures when dealing with Curve signs, but also had a complete failure for No Right signs. All of the signs were classified as No Left or School Ahead signs. This may be due to the similar nature for the No Left signs, but the reasoning behind the matching for School Ahead is not known for sure. One possibility is that the lack of training for both would leave the No Right signs to not have enough identifying features. If this were the only reasoning then No Left signs would also match similarly with other signs. This is not the current case, as No Left signs match with itself in 96% of all cases.

Aside from the No Left and Curve issue, AlexNet and HOG-SVM both performed surprisingly well. With training data of over 100 images, both systems were able to match successfully with at least 97% accuracy. When dealing with less data, both systems still matched fairly well when possible. AlexNet had an overall success rate of approximately 68.5%, hindered by its issue for No Right signs, while HOG-SVM's success rate was around 77.5%. However, with proper training from all signs, this number would undoubtedly go even higher.

6.REFERENCES

[1] *Image Category Classification Using Deep Learning*. N.p., n.d. Web. 16 May 2017.

[2] J *Digit Classification Using HOG Features - MATLAB & Simulink Example*. N.p., n.d. Web. 16 May 2017.

[3]Blegium Traffic Sign Dataset, http://btsd.ethz.ch/shareddata/

7. APPENDIX

7.1. HOG-SVM

```
close all;
clear all;

myTrainingFolder = 'D:\RIT MS\computer vision\traffic sign
project\BelgiumTSC_Train\stop\stopSignImages';
testingFolder = 'D:\RIT MS\computer vision\traffic sign
project\BelgiumTSC_Train\stop\stopSignTest';


trainingSet =
imageDatastore(myTrainingFolder,'IncludeSubfolders', true,
'LabelSource', 'foldernames');
testingSet =
imageDatastore(testingFolder,'IncludeSubfolders', true,
'LabelSource', 'foldernames');

%display random training image with hog feature
imageSize = [180 193];
im = imread(trainingSet.Files{8});
figure;
imshow(im);
title('Breaker');
im = imresize(im,imageSize);
im = imbinarize(rgb2gray(im));

[feature, visualization] =
extractHOGFeatures(im,'CellSize',[4,4]);
figure;
plot(visualization);
title({'CellSize = [4 4]'; ['Length = '
num2str(length(feature))]});

cellSize = [4,4];
hogFeatureSize = length(feature);


trainingSetLabels = countEachLabel(trainingSet)
numImagesTraining = numel(trainingSet.Files)
numImagesTesting = numel(testingSet.Files)
testingSetLabels = countEachLabel(testingSet)


trainingFeatures = zeros(numImagesTraining,
hogFeatureSize);
testFeatures = zeros(numImagesTesting, hogFeatureSize);
for j = 1 : numImagesTraining
   img = imread(trainingSet.Files{j});
   img = imresize(img,imageSize);
   img = rgb2gray(img);
   img = imbinarize(img);
   trainingFeatures(j,:) =
extractHOGFeatures(img,'CellSize',cellSize);
end
trainingLabels = trainingSet.Labels;
classifier = fitcecoc(trainingFeatures,trainingLabels)

for l = 1 : numImagesTesting
   img = imread(testingSet.Files{l});
   img = imresize(img,imageSize);
   img = rgb2gray(img);
   img = imbinarize(img);
   testFeatures(l,:) =
extractHOGFeatures(img,'CellSize',cellSize);
end
testLabels = testingSet.Labels;
predictedLabels = predict(classifier, testFeatures);
[confMat,order] = confusionmat(testLabels,
predictedLabels);

%to convert in percentage form
confMatPercent =
bsxfun(@rdivide,confMat,sum(confMat,2));
```

7.2. AlexNet

```
myTrainingFolder =
'C:\Users\sk3126\Downloads\BelgiumTSC_Train\BelgiumTS
C_Train\Training';
testingFolder =
'C:\Users\sk3126\Downloads\Testing\Testing';


trainingSet =
imageDatastore(myTrainingFolder,'IncludeSubfolders',
true,  'LabelSource', 'foldernames');
testingSet =
imageDatastore(testingFolder,'IncludeSubfolders',
true,  'LabelSource', 'foldernames');
net = alexnet;
layersTransfer = net.Layers(1:end-3);
numClasses = numel(categories(trainingSet.Labels));
layers = [...
  layersTransfer
  fullyConnectedLayer(numClasses,'WeightLearnRateFactor'
,20,'BiasLearnRateFactor',20)
  softmaxLayer
  classificationLayer];

sz = net.Layers(1).InputSize
numImagesTraining = numel(trainingSet.Files)
numImagesTesting = numel(testingSet.Files)


for j = 1 : numImagesTraining
  img = imread(trainingSet.Files{j});
  img = imresize(img, [227 227]);
  imwrite(img,trainingSet.Files{j});
end

for j = 1 : numImagesTesting
  img = imread(testingSet.Files{j});
  img = imresize(img, [227 227]);
  imwrite(img,testingSet.Files{j});
end

options = trainingOptions('sgdm',...
  'MiniBatchSize',5,...
  'MaxEpochs',10,...
  'InitialLearnRate',0.0001);
testLabels = testingSet.Labels;
netTransfer = trainNetwork(trainingSet,layers,options);
predictedLabels = classify(netTransfer,testingSet);
[confMat,order] = confusionmat(testLabels,
predictedLabels);
confMatPercent =
bsxfun(@rdivide,confMat,sum(confMat,2))
```