Car Image and Traffic Sign Classification Using Neural Network and SVM

Iowa State University, Department of Computer Science
Han-Shu Chang, He Jin, Xuan Lu

Abstract

With the advent of driverless car, it is critical for driverless vehicle to acquire the as much as possible information from the its environment. In our team project, we focus on the recognition the car brand and transportation sign from vehicle and traffic sign images respectively. We apply CNN and SVM models from what we learned in class, and explore beyond that to SIFT (Scale Invariant Feature Transform) for image feature selection and detection. In our results, with the combination of SIFT and SVM, the accuracy to identifying three classes of car can achieve to 70%. And for traffic sign recognition, the accuracy of identifying 62 classes get to up to 82% with CNN model.

Keywords: CNN, SVM, SIFT, Car Images, Traffic Signs

1. Introduction

Nowadays, driverless car is becoming popular. Some big companies such as Uber, Google, and Baidu already work on this area. There are several aspects of functions such as object detection, distance measurement, traffic sign recognition to be implemented to make driverless car come true. Even though the companies like Uber already put such driverless technology into use, the technology is still treated as driver assistance system. Countless papers intend to improve performance of driverless car especially for the car recognition like[1], has reported that they lifted the 2D car object to 3D with the emphasis on local feature appearance and location. Moreover, they showed that 3D representation outperforms 2D images for fine-grained image categorization. With the motivation of this paper, we want to apply some classifiers we learned like CNN, SVM and extended beyond what we learned to combine with SIFT feature extraction algorithm to help us implement car types classification and traffic sign recognition. For the outline of this report, in section 2, we will explain how we build our model and especially how to connect SIFT to SVM. In section 3, we have test set to measure our model's performance. Last we will give a conclusion in the end.

2. Design & Implementation

Preprocess DataSet

1. We collect our training and testing dataset from AI group in Stanford with 16185 images of 196 classes, and dataset as follows: [2] 2. We resize all of the training picture as the size of 128 * 256 pixels, and cut the images according to the bounding box coordinate of car location. 3. In order to simplify the training process, we manually select three representative classes of images including car, SUV and truck. And around 200 pictures are in each class, and there are number of 621 images in total. Figure 1 shows the sampling images from the three classes.

Figure 1. Three Car type representatives

CNN

Convolutional neural network is good at image recognition due to the more efficient feature detection and less parameters. Max polling will report maximum outputs within the rectangular neighborhood, which help for feature detection in images. We applied the CNN for both vehicle and traffic sign image recognition. The architecture of CNN shown as follows:

1. Convolutional Layer #1: 32 3x3 filters, ReLU activation function, stride = 1
2. Pooling Layer #1: Performs max pooling with a 2x2 filter, stride = 2
3. Convolutional Layer #2: 64 5x5 filters, ReLU activation function
4. Pooling Layer #2: Performs max pooling with a 2x2 filter, stride = 2
5. Dense Layer #1: 256 neurons
6. Dense Layer #2 (Logits Layer): 3 neurons, one for each cars target class.

Here is the selected list of parameters we applied:

Number of Convolutional Layer	2
Number of Filters	64
Kernel Size	3*3
Pooling Size	2*2
Activation function	ReLU
Error function	Cross-entropy
Learning rate	0.01
Number of Epochs	15
Batch Size	32

Table 1. CNN parameters

SVM

Given the labeled training data, SVM outputs the optimal hyperplane which categorized the new examples. We use image pixels as the inputs for SVM. To be more specific, we converted the 2D images to 1D array for the input of SVM. And we used the linear model with SVM.

SVM + SIFT

As previous results by using SVM are not good, we want to add some feature detectors method to help our model learn better. For example, corner detectors like Harris etc., they are rotation-invariant, which means, even if the image is rotated, we can find the same corners. But it does not suit for car classification in which some cars are bigger in image, and some are not. So how could we find the same specific point in different images that has different view of cars.

Here we try to use SIFT. Full name is a Scale Invariant Feature Transform, which is an algorithm to help us detect and describe local features in images before we apply SVM. There are mainly four steps involved in SIFT algorithm [3]: scale-space extrema detection, keypoint localization, orientation assignment, and keypoint Descriptor. These four steps already implemented in openCV. Although it’s patented by UBC, there are still a package online free to download to invoke SIFT on study use.

Steps to connect SVM and SIFT like below [4]:
1.Extract SIFT from training set of images
2.Compute K-Means over the entire set of SIFTs
3.Assigning each SIFT of image to one of K clusters
4.Take descriptors(K-dimensional) as train data to SVM to learn Classifier

In Figure 2, it explains in details how to connect SIFT to SVM. As previously mentioned, there are four steps in SIFT. Keypoint is a kind of point which has specific distinction with other points. That’s why we get interested in. And SIFT will find

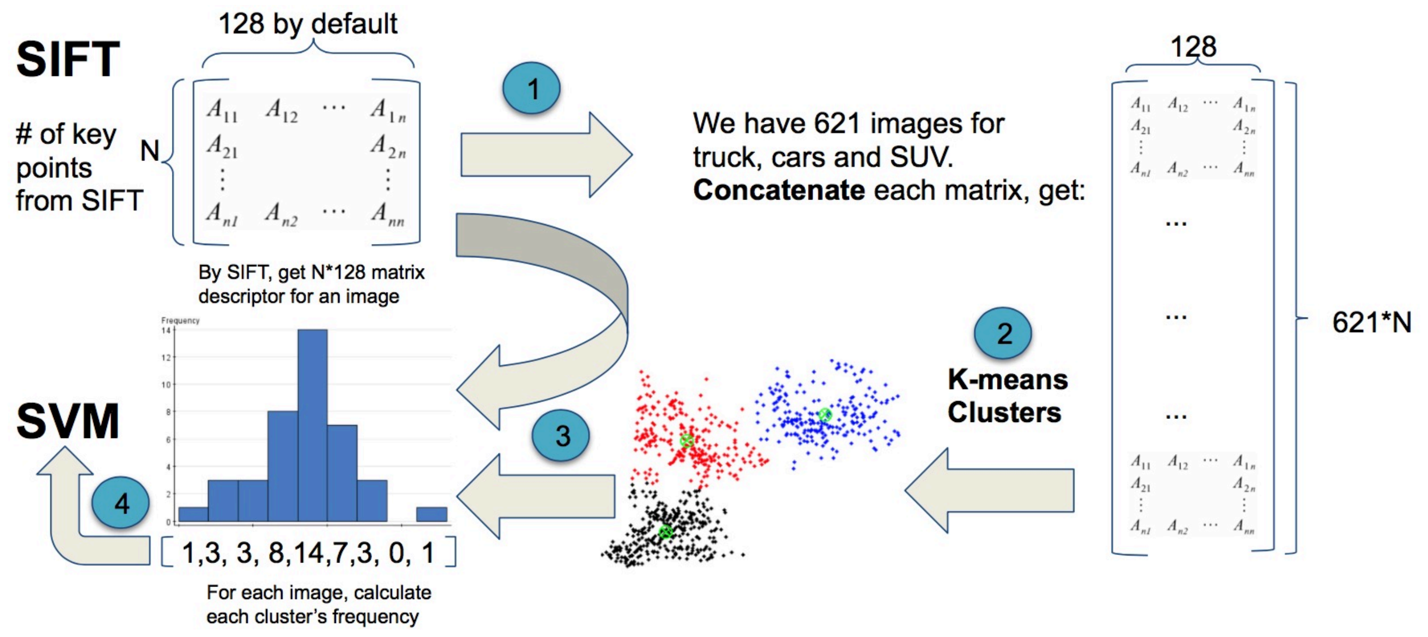

SIFT
128 by default
# of key points from SIFT
N
By SIFT, get N*128 matrix descriptor for an image
1
We have 621 images for truck, cars and SUV. Concatenate each matrix, get:
128
621*N
2
K-means Clusters
3
SVM
4
Frequency
1,3, 3, 8,14,7,3, 0, 1
For each image, calculate each cluster's frequency

Figure 2. SIFT connected to SVM

these potential keypoint on an image. In last step, keypoint descriptor is created. A 16x16 neighbourhood around the keypoint is taken. It is divided into 16 sub-blocks of 4x4 size. For each sub-block, 8 bin orientation histogram is created. So a total of 128 bin values are available. It is represented as a vector to form keypoint descriptor.

In Figure 2, the first matrix is in the form of N*128, where N is the number of the keypoint extracted from an image. Next, based on all the keypoints from training image set, we want to do the clustering. As previously mentioned, 128 used to describe a keypoint and we have 621 images for training truck, cars, and SUV. Thus, to get all keypoint, in first step, we concatenate each matrix descriptor for an image, and get a matrix in size of (621*N) * 128. In second step, we use K-Means clustering to separate those distinctive keypoint and put similar keypoint together as a cluster. Each cluster will be represented by the mean value calculated from the keypoint in that cluster. In this step, we group similar keypoint and take them as a feature and there are lots of features. We can set K value in K-Means clustering to decide how many features we want to use to distinguish three types of cars. Usually, K is set to 80-120, and performance may look good. In third step, as we've already got features from second step, for each image, we re-compute which feature or cluster each keypoint in an image belongs to. Then we get a histogram similar with the one in Figure 2. A histogram is represented by the vector in which index 0 of vector represents for first cluster or feature, index 1 represent second cluster, etc. Now, we already get all inputs, 621* each image corresponding histogram represented by a vector, that SVM needs to do classification. In last step, just put these training data with each image's label together into SVM and do training.

3. Evaluation and Discussion

Car Images

For the car image dataset, we tried to apply CNN and SVM models. In the following, our evaluation separates CNN into 3 phases experiments where each classifier has different number labels to classify. And SVM separate into 2 phases, where the first being a regular SVM, and the second is the SIFT+SVM.

·CNN

We first trim the existing 196 classes into 49 classes. We apply different combinations of CNN models. We got around 2% in average and 5% is the best accuracy we can get via adjusting parameters.

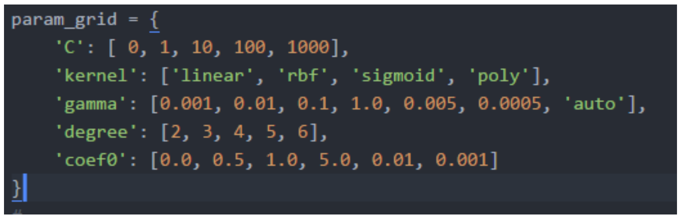
We then go ahead trim the 49 classes into 5 classes. We chose 5 classes that look different to each other on the appearance, hoping to have higher accuracy outcome. However, with 5 classes, we still only get around 20% accuracy.

From the above 2 experiments, we observed that there are only about 40 images for each class. Therefore, from 49 classes, we manually classify them into 3 classes: SUV, Truck, and regular car. Now with each class containing around 200 images, we applied the CNN. However, we still got around 20-30% accuracy.

From above 3 phases experiments, we conclude that the nature of this dataset is hard to classify just by a simple model. We have experiment different parameters such as the structures of the CNN, the number of nodes, different optimizers, different according parameters. Unfortunately, the outcome does not look optimistic.

·SVM

We applied SVM model to the 3-classes dataset. The result of SVM performs more stable on the result, which means the outcome accuracy does not fluctuate. Using 'linear' kernel function, the SVM model gave us 33% accuracy, which is more stable and higher than what we got from CNN. But 33% means it still does not perform better than a random guess. Additionally, We tried the auto-adjusting technique using grid search to find the best parameters. Figure 3 shows the parameters we look into form the best model. The best model we found for SVM has the accuracy of 52%.



```
param_grid = {
    'C': [ 0, 1, 10, 100, 1000],
    'kernel': ['linear', 'rbf', 'sigmoid', 'poly'],
    'gamma': [0.001, 0.01, 0.1, 1.0, 0.005, 0.0005, 'auto'],
    'degree': [2, 3, 4, 5, 6],
    'coef0': [0.0, 0.5, 1.0, 5.0, 0.01, 0.001]
}
```

Figure 3. Parameters for grid search

technique for finding best parameters

·SVM+SIFT

We apply the feature detector method, SIFT, to grab features and form new input data. Then, we feed it to SVM model. A boost on the accuracy present. We got 70% of accuracy on the car data finally.

In conclusion to these whole experiments and dataset, we have two main observations:
1. The nature of this car dataset cannot be classify straight forward, so some feature detection methods or preprocessing are needed. In fact, the original paper[1] that research on classifying dataset used some 3D analysis and many other techniques to preprocess data and achieve highest 94% accuracy.
2.The number of images in one label should be at least sufficient for the model to learn its features.

Traffic Sign

For the traffic sign dataset [5], we tried to apply CNN and SVM models. In the following paragraphs, we show the evaluations on traffic sign dataset with CNN model and SVM model.

·CNN

With the CNN model we trained, the best model we got had the accuracy 82%. The structure of CNN model can be found in section 2. Figure 4 shows the 10 random images of traffic sign with its actual label. Figure 5 shows the labels predicted by this CNN model, where the green-colored text means classified correctly, the red-colored text means classified incorrectly.

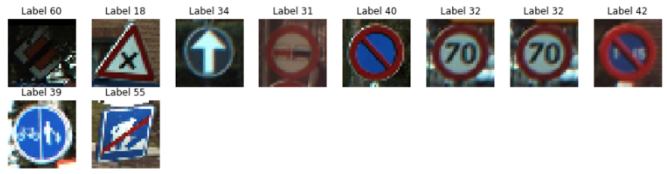


Figure 4. Demonstration of 10 random chosen traffic sign images

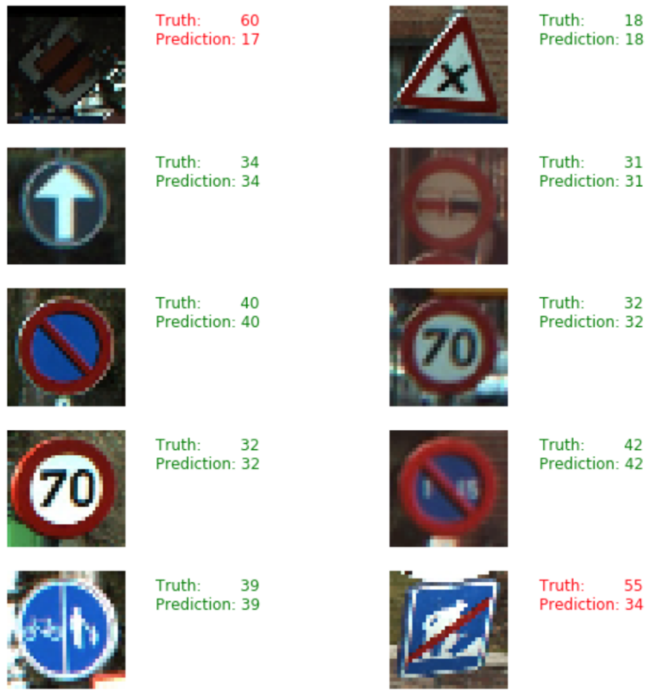


Figure 5. The predictions that CNN model made

·SVM
For the SVM model, we used ‘linear’ kernel function and it was able to achieve a even more higher accuracy, 88.5%.

The conclusion to both car dataset and traffic sign dataset are as follow:
1.Compared to car image data, traffic signs with 2D graphics give a better result. The features of traffic signs are easier to catch than of cars image.
2.The amount of images in one class affect result.
3.SVM works a little better than CNN in the dataset.

4. Conclusion

We apply CNN on our car classification dataset, because CNN usually work better on images classification. But in our case, we tried to adjust lots of parameters including number of layers, number of units in each layer, types of activation function, kernel size, pooling size, learning rate and etc. written in Table 1. But all attempts does not work. It almost fail on car classification job. The reasons that cause this issue may be due to background affects training process, this model needs more training data due to complicated images, and image dataset for a type of car has different angles, in which our model cannot relate them together.

From results, SVM actually work better than CNN, but it’s far from good. Thus, we add SIFT feature extraction to SVM in order to make it better. In fact, SVM indeed does better with SIFT than itself. It gets nearly 70% acc on classifying truck, SUV and normal cars. But both of method performance does not meet our expectation and market’s requirement. Thus, there must be other classifiers or combine some together to let it work.

Consider that traffic sign recognition is still one of hot areas in driverless car and we already built two models for car types classification. Through adjusting some parameters, we apply our models on such area and found that our models work much better on simple image dataset like traffic sign.

5. References

[1] Jonathan Krause, Michael Stark, Jia Deng, Li Fei-Fei, 3D Object Representations for Fine-Grained Categorization, *4th IEEE Workshop on 3D Representation and Recognition, at ICCV 2013* (3dRR-13). Sydney, Australia. Dec. 8, 2013.

[2] Stanford Car Images Dataset:
https://ai.stanford.edu/~jkrause/cars/car_dataset.htm

[3] OpenCV, Introduction to SIFT
https://docs.opencv.org/3.1.0/da/df5/tutorial_py_sift_intro.html

[4] Image classification using SIFT features and SVM,
https://www.researchgate.net/post/Image_classification_using_SIFT_features_and_SVM2

[5] Belgian Traffic Sign Image Dataset:
http://btsd.ethz.ch/shareddata/