Vehicle Classification And License Plate Recognition

Group 26

Amlan Kar Nishant Rai Sandipan Mandal Sourav Anand

Department of Computer Science and engineering

Abstract

The project aims at detecting and classifying relevant objects in a video stream (Surveillance Video) in real time. In case of four wheelers, detection and recognition of license plate is also desired. We discuss the important intermediate tasks required for the same. We then move on to discussing existing work in the area along with popular methods [6], [11] used for performing the involved steps. We propose multiple methods and show the related results. We also study the dataset used [13] and discuss our findings for the same. Visualizations and the inferences drawn are also discussed. We then move to object classification and study the performance of various classifiers. We study the effect of performing data multiplication on the quality of our model. Finally, we discuss the steps involved in License Plate Recognition [1] and the methods used for the same.

CS771A: Project Report

Under the guidance of Dr. Harish Karnick
Indian Institute of Technology, Kanpur

Contents

1 Introduction

1.1 Problem Statement

The project was aimed at detecting and classifying relevant objects in a video stream (coming directly from fixed security gate cameras). We also tried to detect and recognize the characters on the license plate in case of 4-wheelers.

1.2 Steps Involved

The steps involved can be roughly summarized as follows. The work flow is in the same order as shown below,

1. Background Detection
2. Background Subtraction
3. Identifying Objects
4. Tracking Objects
5. Object Classification
6. Number Plate Localization
7. OCR Text Detection

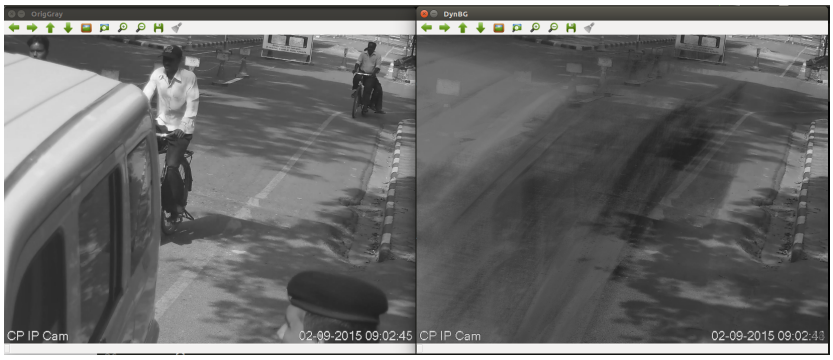
2 Details of the steps Involved

We cover the details of the steps described above.

2.1 Background Detection

- Static Background
  A static background was generated using the video by getting an average of all the frames of the video.

- **Dynamic Background:** A dynamic background was generated which was updated as the video proceeded. The background was initially set to the static background as computed above and for every new frame of the video, the back ground was updated by keeping 99% of the previous dynamic background and adding 1% of the current frame.

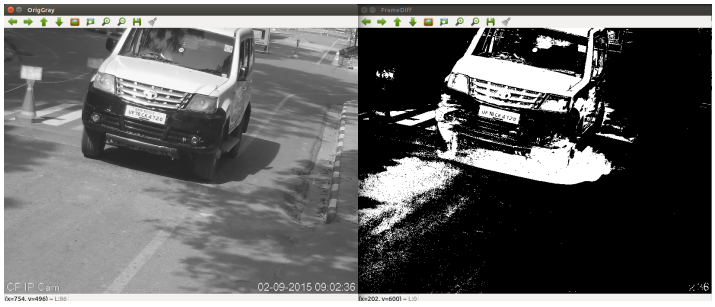


The above figure shows the video frame in left side and the background computed at that frame. It can be seen that the background is updated with respect to the car as compared to the static background (above).

2.2 Background Subtraction

- **Static Background:** Subtracting the frames from the static background computed at the start.

- **Dynamic Background:** Subtracting the frames from the background which is dynamically updated with the video.



- **Gaussian Mixture-based Background/Foreground Segmentation**: We'll be referring to this as MOG. As we can see, the results given by MOG are extremely unsatisfying as the shadow and object proposals are being mixed up.

We decide not to go further with MOG due to its poor performance. The main culprit behind it could be the noise in the data and also the lack of parameter tuning in the algorithm.

2.3 Object Detection

Object 'detection' is one of the most important parts in the complete setup explained earlier. Mainly because all the consequent stages depend upon it.

We discuss some of the popular object detection methods below,

- **Interest Point based Detection**; Find interesting points in images/objects express their respective localities. A few popular examples: SIFT features [9], Harris features .
- **Background Subtraction**; Object detection can be achieved by building a representation of the scene called the background model and then finding deviations from the model for each incoming frame. We use this background model to extract live areas from the video and do further operations on them.
- **Segmentation** - The aim of image segmentation algorithms is to partition the image into perceptually similar regions. There has been work showing the closeness to multi object tracking (Region-based Segmentation and Object Detection) [3]

2.3.1 Using Image Processing Operations

We construct a background model and extract objects using the constructed model by considering deviations from it. Objects are detected using morphological operations.
The main reason for choosing to construct a background model is the fact that the security camera is still and thus a reasonable (accurate and nearly constant) background can be easily computed.
We use morphological operations because of the fast computation time involved and robustness and effectiveness of the proposed method (i.e. Invariance to Camera angle and Illumination)

2.3.2 Using Machine Learning Techniques

We use the LPO (Learning to Propose Objects) [6] code (CVPR '15) by Phillipe Kranhebuhl from UC Berkeley's BVLC group to act as the object proposer in our videos.

- An image on average took 25 seconds for processing which was too high for our requirements.
- The results were very impressive
- The method uses an ensemble of jointly trained binary classification models (Global and Local CRFs, global and local in terms of connectivity) that are locally trained at locations decided using combinatorial optimization techniques.

2.3.3 Our approach

As discussed earlier, the aim during in this stage is identifying blobs and proposing a bounding contour for it. We extract the dominant blobs using the following steps:

- Background Subtraction (Discussed earlier)
- Thresholding and Binarizing the image
- Gaussian Blur (To smooth image and reduce the noise)
- Closing and Dilation operations (Converting objects to blobs)
- Finding contours (Of computed blobs)
- Eliminating Noisy Contours (Remove Noisy Blobs)

All the above operations can be performed in real time (Around 33 fps).

2.3.4 Intermediate Results

- Thresholding and Binarizing the Image. Left Image shows the final result (i.e. Detected Contour).

Figure 1: Result after binarizing

- Blurring the image.

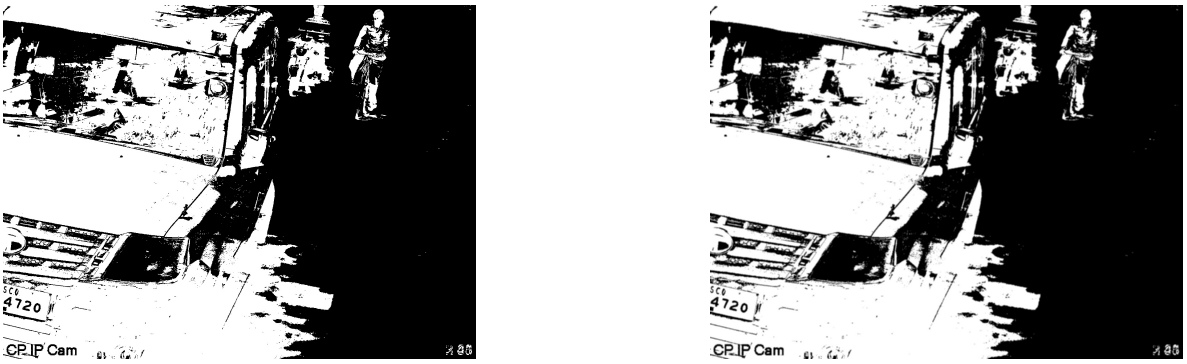


Figure 2: Blurring

- Closing and opening the image.

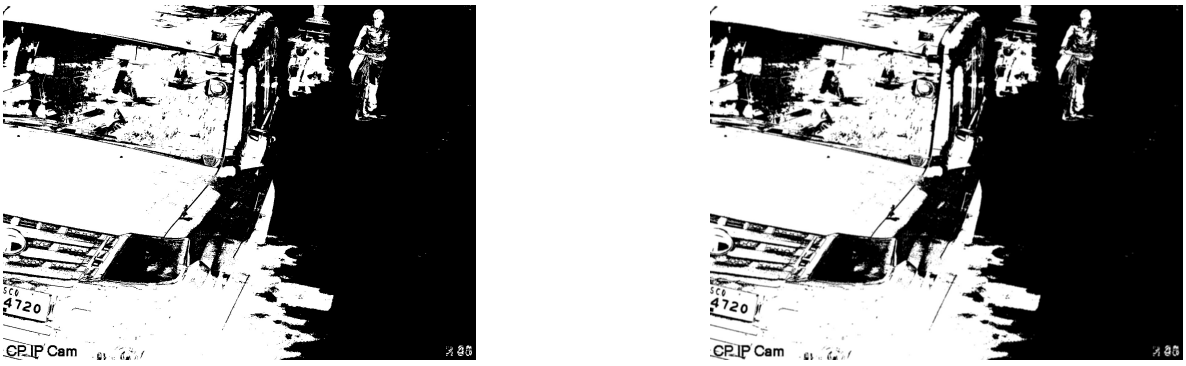


Figure 3: Closing and Opening

- End Result.

Figure 4: Results (Boxes indicate Objects)

2.4 Object Tracking

The previous sub-section covered the Object Detection Stage. Note that we've not used the sequential information of the input i.e. it is a continuous video stream. We name this sub-section Object 'Tracking'. The reason it is different from the previous section is because we ignored the inter frame relationships in Object Detection, while we're using it to infer how the object moves.

We tried the following two methods for the same:-

- Contour similarities (Spatial Closeness) Mean-Shift
- CamShift based tracking (Very poor results, tuning required)
- SIFT Based feature Matching [9] (Point matching variants tried: Least Squared Distance (Multiple), Hungarian based matching (Single))

Other possible models:-

- Region Proposal Networks [11]
- Region Based Segmentation and Detection [3]

(Not used due to data constraints, Pre trained Models available but defeat the purpose of the project)

2.4.1 Contour Similarity Based Tracking

Extremely simple idea: Compute similarities between the contours formed in consecutive frames. The closest contours represent the same objects. In case of a completely new blob formed, we declare it as a new object (And start tracking it).

Related Issues:

- Formed object can also be a group of objects.
- High Computation Time. Complexity involved is O(n**2), where n is the number of points in the contour.

Solutions Proposed:

- Compute momentum (speed) of each box. Assumption involved is that the objects move with constant velocity (Which is pretty reasonable). Requires additional heuristics.
- Represent (Approximate) contours as rectangles (Bounding rectangles). O(1) computation.

2.4.2 SIFT Feature Based Tracking

Observe that the objects in consecutive frames are extremely similar. Thus, matching the images on the basis of SIFT vectors (feature points) gives 'extremely' good results (Almost an exact match).

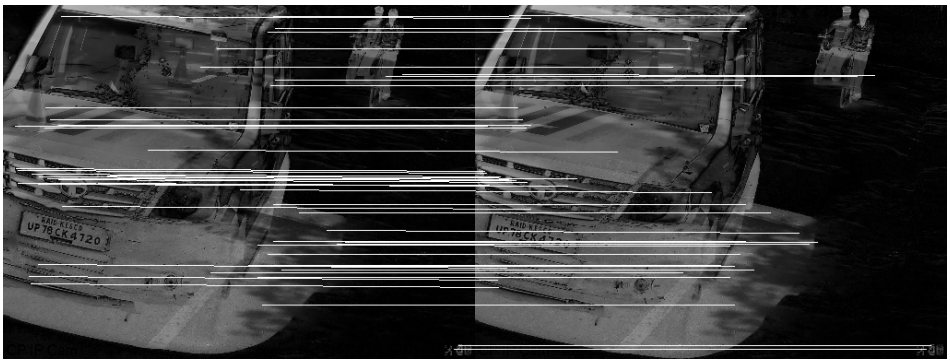
Related Issues:

- Computation time increases (Computing SIFT vectors takes around 0.09 seconds), The matching part takes around 0.4 seconds (Which is the main bottleneck).
- In case of multiple objects, the interest point matching algorithm sometimes matches different objects (Can be rectified by changing the distance parameter)

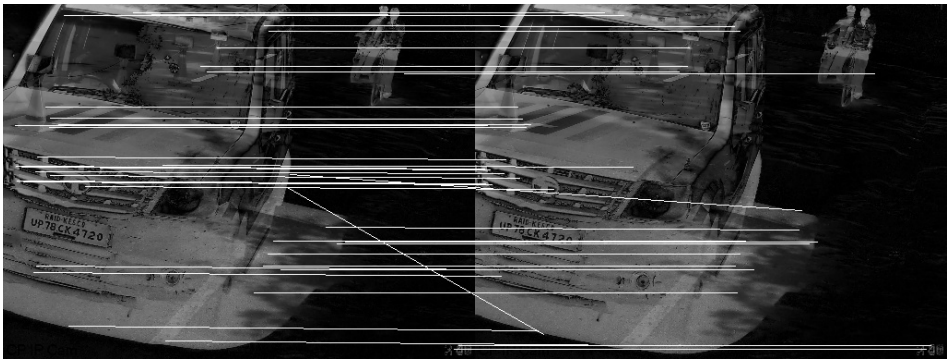
Possible Solutions:

- Consider only the **prominent** SIFT vectors. This can reduce the time to allow processing around **10fps**.
- The incorrect matches can be removed after **ignoring** the absurd matches (Based on the **expected translation**) (Explained Later)

Some Results: Notice that the matching lines are parallel (Can be used to remove incorrect matches) (Showing an almost exact match and translation of the object)



Notice that the correct matching lines are parallel. This can be used to remove the incorrect matches. Since, the slope of the incorrect match is very different from the correct matches. This is due to the fact that there is only linear (No rotational) translation of the object in small intervals.



3 Dataset

The data used is collected via crowd-sourcing, the details are discussed in this section.

3.1 Dataset Description

Data collected by crowd sourcing (Using the architecture made by UC Irvine (VATIC)) [13] Consists of 7 classes: Car, Person, Motorcycle, Bicycle, Rickshaw, Auto-rickshaw, Number-Plate

Dataset Issues:

- Large number of Outliers (Incorrectly marked objects)
- Extremely similar Images (Discussed later)
- Poorly labeled objects (Occlusion, excessive translation, Overlap with others)
- Low number of images, Low diversity

We discuss a few results in the later sections. The train and test set used are described in the next slide. We also discuss a few points about the dataset.

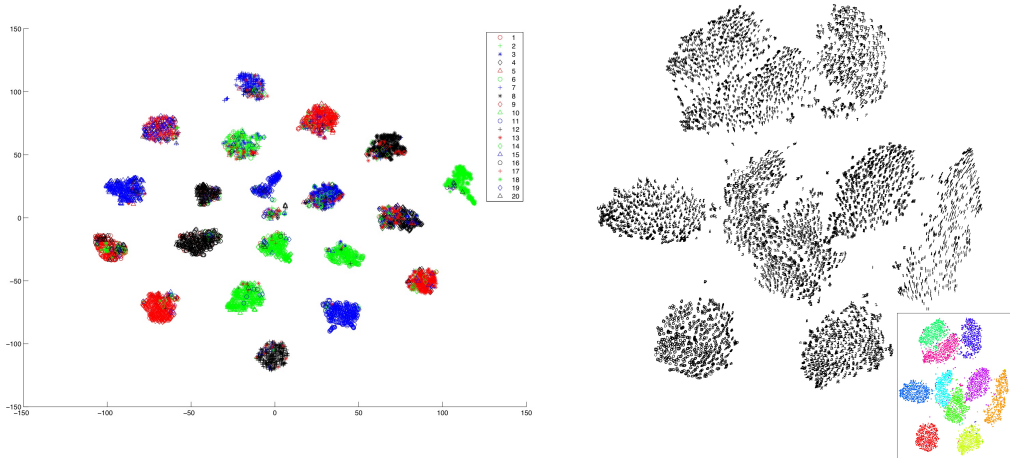
3.2 Data Pre-processing

Our Training set consists of around 80% of the total data. Our Testing set contains the rest of the data (Ensuring that we choose completely different objects from the training set).

Failing to ensure the above mentioned condition led to absurdly high accuracies (Around 96-98%). This is due to the earlier mentioned reason of extremely similar images, which makes the dataset equivalent to a small set multiplied by a constant.
We (try to) minimize incorrectly labeled data using a few heuristics during image extraction. We also consider only those images of an object which are sufficiently different.

In order to increase the data set size, we can perform the following,

- Flipping the image (Mirroring it)
- Taking a slightly reduced part of the image (Reference)
- Varying the intensities (Through Gamma correction or other methods)



(a) 20-NewsGroup Dataset (b) MNIST Dataset

Figure 5: Visualization Using tSNE (examples)

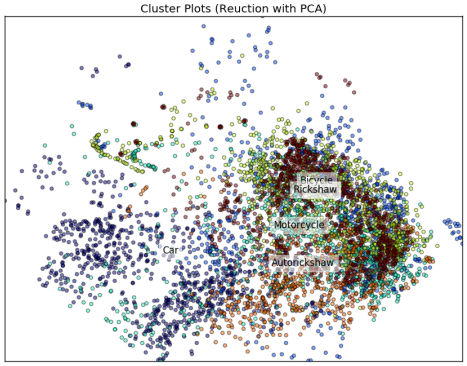
Our motivation for taking a reduced part of the image comes from the fact that a model should be able to decide on an object based on a few distinctive parts of the object.
We use the above mentioned methods to obtain a larger dataset (Around 16 times larger) and train a few models upon it. We observe an improvement in the results (Around 3-4%) which is discussed later and further supports the earlier claim.

3.3 Understanding the Dataset

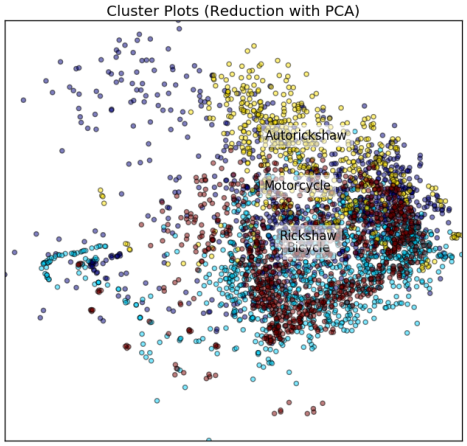
Before discussing the approaches used for classification. We discuss a few features of the dataset in the following slides.
This is an extremely important part which is generally ignored. This section focuses on a few features which motivates further discussions (Or proposals).
We plot the distribution of the data points in 2 (and 3) dimensions using dimensionality reduction algorithms namely PCA and tSNE [12] (Which has been shown to perform extremely well for visualizing high dimensional data).
Assuming the classes to be easily separable, we should expect the cluster centers to be visibly different. (Examples shown later)

Visualizing our Dataset:

- The given image is reduced using PCA (Used due to speed for a primary inference)
  We can see that the only reasonably separated cluster is Cars. Note that 'Rickshaw' and 'Bicycle' are extremely similar, which should be expected.
  This motivates us to consider a two class problem i.e. 'Car' and 'Not Car'. Then use another classifier to make further predictions.



- The given plot consists of classes after removing 'Car' and 'Person'.
  We can see that again 'Rickshaw' and 'Bicycle' are extremely similar. Which also motivates to merge the two classes together into one.
  We can also see that 'Auto-Rickshaw' and 'Rickshaw + Bicycle' are pretty different (Far apart), thus can be discriminated nicely.



- Two Class Plots (Rickshaw, Auto-Rickshaw) (Top) Two Class Plots (Car, Not Car) (Bottom)
- tSNE Plots (All classes except Person) (Left)

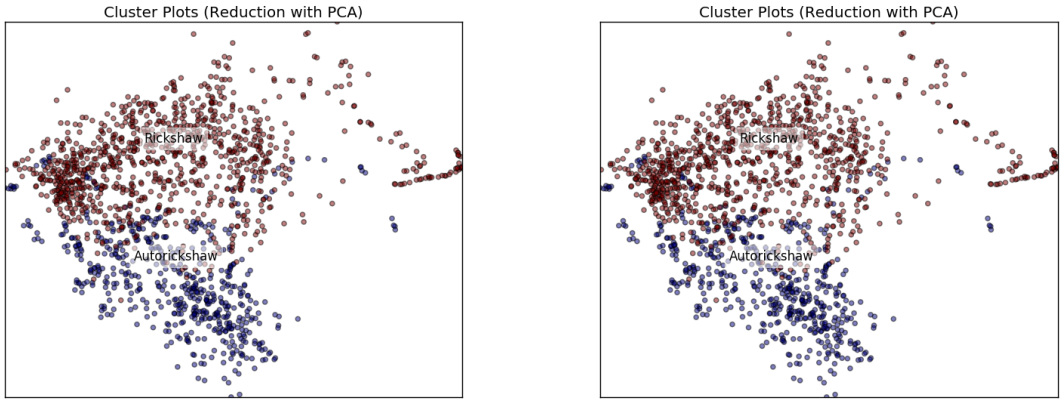
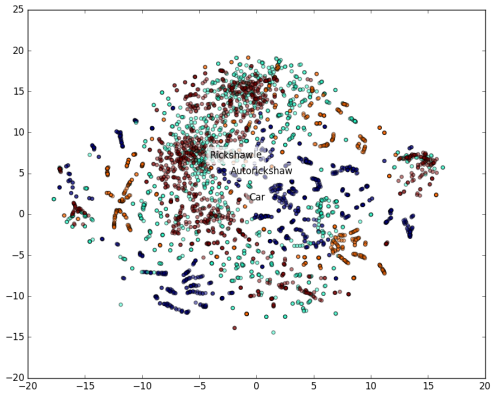


Figure 6: Reduction Using PCA



4 Object Classification

We tried various stuffs, to arrive at the the classifier and feature to use.

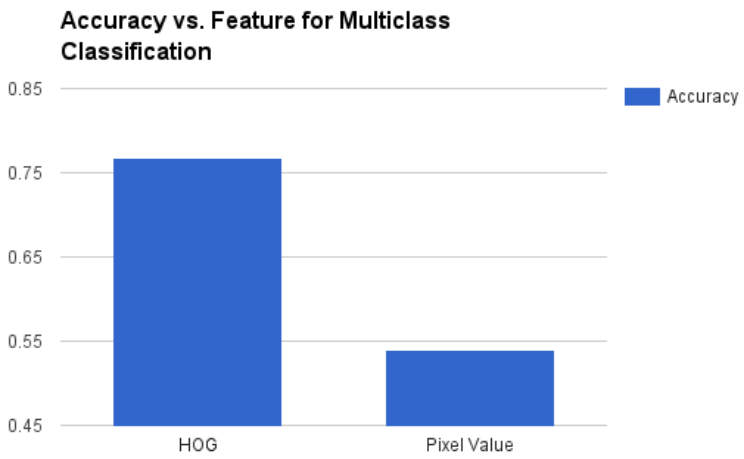
4.1 Haar Cascade

The very first classifier we tried using was HAAR cascade classifier for vehicle classification. We trained it on an external dataset. We used the following dataset from stanford [7].
The results were very poor. So later we tried using the same classifier with data extracted from the video. But we could get only limited number of training samples for HAAR cascade. Hence the classifier could not be properly trained. So again the results were not satisfactory. This made us move to other common classifiers like Linear SVMs, SVMs (With other kernels), Random Forest and Adaboost.

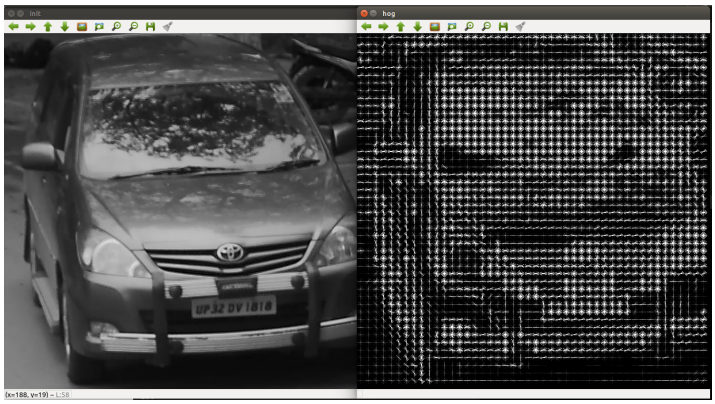
4.2 Analysis of features

We experimented with two types of features. We used the following features:-

- HOG
- Pixel Value
- PCA on Pixels



As expected, HoG features outperformed raw pixel features by a large margin.

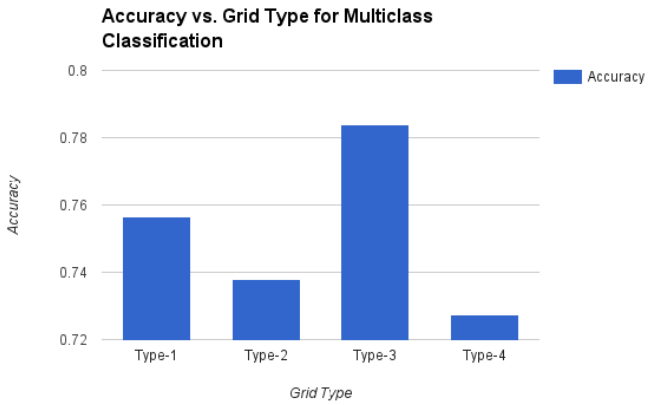


4.3 Analysis of various parameters of SVC

We tried Parameter Tuning using a simple Grid Search. WeMhave used the following grids for SVM,

- 'C': [0.1, 1, 10, 100, 1000], 'kernel': ['rbf'] (Type-1)
- 'C': [0.1, 1, 10, 100, 1000], 'kernel': ['linear'] (Type-2)
- 'C': [10,20,30,40,50,60,70,80,90,100], 'kernel': ['rbf'] (Type-3)
- 'C': [10,20,30,40,50,60,70,80,90,100], 'kernel': ['linear'] (Type-4)

Our observations are summarized in the following plot.

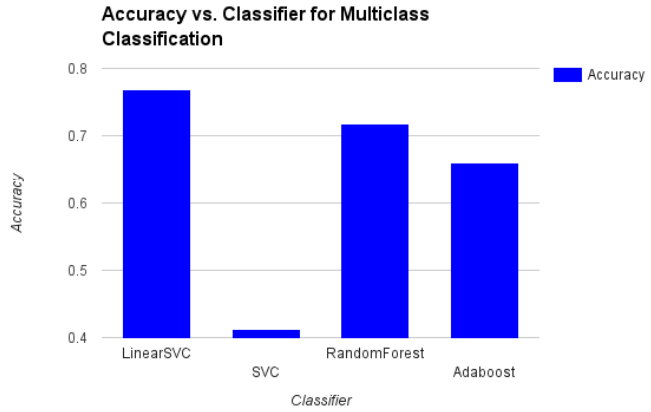


4.4 Analysis of various classifiers over HOG features

For Multiclass Classification we have tried the following classifiers:

- Linear SVM
- SVM (With RBF Kernel)
- Random Forest
- Adaboost

The Performance of various classifiers is shown in the following plot.

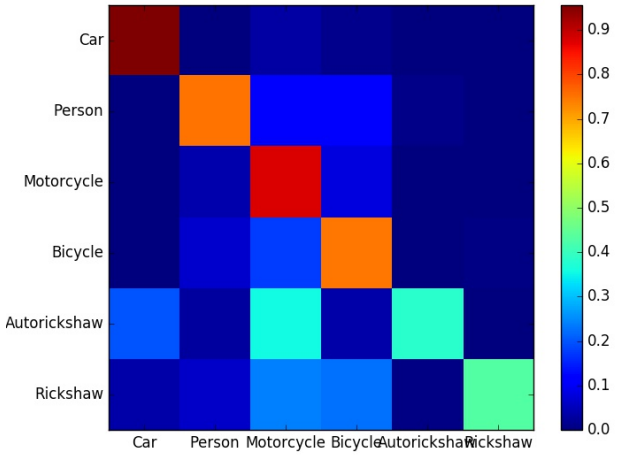


4.5 Performance of Linear SVM

Linear SVM gave the best results with hinge loss and HoG features for vehicle classification. The Prediction statistics is summarized below:-

Class	Precision	Recall	F1-score	Support
Car	0.8444	0.9559	0.8967	227
Person	0.6268	0.7507	0.6832	349
Motorcycle	0.7823	0.8784	0.8276	1522
Bicycle	0.8058	0.7492	0.7765	1224
Autorickshaw	0.8615	0.3836	0.5308	146
Rickshaw	0.9074	0.4317	0.5851	227

The following figure shows the confusion matrix.



4.6 Convolutional Neural Network

We trained two ConvNet architectures on the Augmented Dataset (11000 training samples, 2000 test samples)

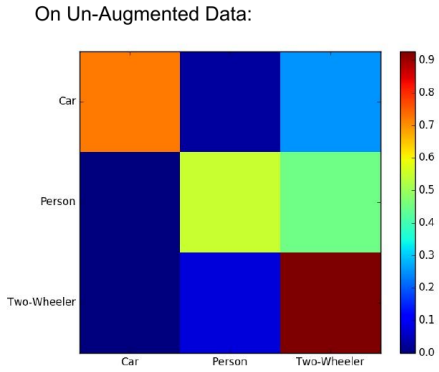
- Architecture 1:
  Image[100x100] → Conv1 [8 filters, 3x3] → ReLU → MaxPool(2x2) → Flatten → Dense[64] (tanh activation) → Softmax[6]
- Architecture 2:
  Image[100x100] → Conv1 [8 filters, 3x3] → ReLU → MaxPool(2x2) → Conv2 [8 filters. 3x3] → ReLU → MaxPool(2x2) → Flatten → Dense[64] (tanh activation) → Softmax[6]

Both the architectures were trained with aggressive Dropout rates to prevent Over fitting

4.7 ConvNet Performance

In this section, we provide results showing the effect of data augmentation (Discussed earlier)

Effect of Data Augmentation
The results before data augmentation is,

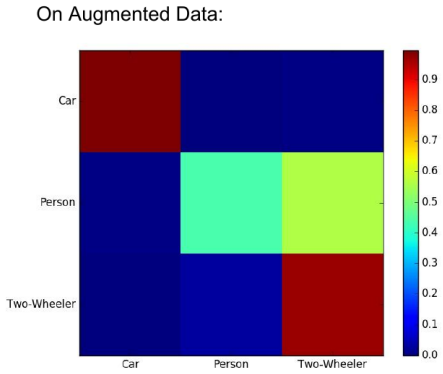


	precision	recall	f1-score	support
Car	1.0000	0.7216	0.8383	176
Person	0.4579	0.5484	0.4991	248
Two-Wheeler	0.9270	0.9270	0.9270	2136
avg / total	0.8865	0.8762	**0.8794**	2560

Both tested on same Test Set!

Figure 7: Results before Data Augmentation

The results after data augmentation is,

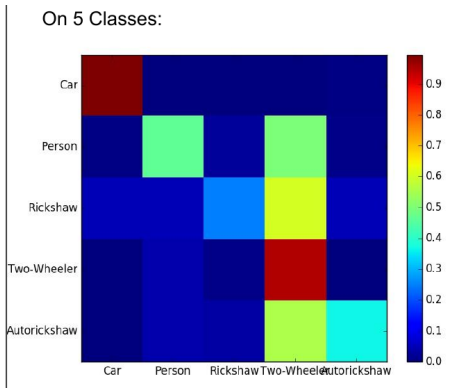


	precision	recall	f1-score	support
Car	0.9831	0.9943	0.9887	176
Person	0.6243	0.4355	0.5131	248
Two-Wheeler	0.9366	0.9686	0.9524	2136
avg / total	0.9096	0.9187	**0.9123**	2560

Figure 8: Results before Data Augmentation

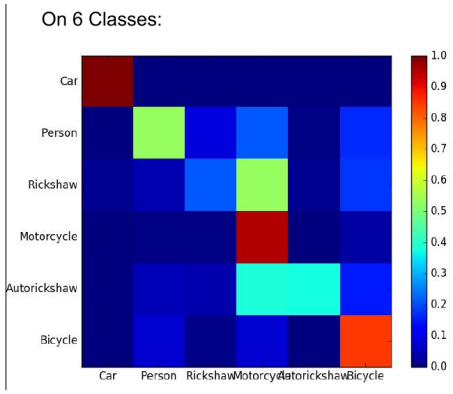
As we can see that the augmentation actually helps improving the model. This can be due to the fact that our model learns a more part wise representation of the dataset through the augmented dataset.

The performance of the convNet is summarized as follows,



	precision	recall	f1-score	support
Car	0.9358	0.9943	0.9642	176
Person	0.5425	0.4637	0.5000	248
Rickshaw	0.5974	0.2500	0.3525	184
Two- Wheeler	0.8704	0.9494	0.9082	2136
Autorickshaw	0.7586	0.3667	0.4944	120
avg / total	0.8238	0.8408	0.8232	2864

Figure 9: Results for 5 Classes



	precision	recall	f1-score	support
Car	0.9724	1.0000	0.9860	176
Person	0.6055	0.5323	0.5665	248
Rickshaw	0.4875	0.2120	0.2955	184
Motorcycle	0.8102	0.9552	0.8767	1184
Autorickshaw	0.9184	0.3750	0.5325	120
Bicycle	0.8606	0.8498	0.8552	952
avg / total	0.8030	0.8142	0.7977	2864

Figure 10: Results for 6 Classes

5 License Plate Detection

After the object classification stage, we come to License plate Recognition part. The first stage in this involves localizing or computing regions where the license plate may lie.
The final part is recognizing the text on the license plate. This involves, segmenting the characters and then using OCR to recognize the text on the license plate. We also require lots of data (text information) to train such an OCR model, thus we have used pre-trained models on text from US License plates. Thus, even if the regions proposed are reasonable, the OCR part is not able to recognize the characters.
There are many prevalent methods for localising license plates. We use image processing operations (Mainly gradients and morphological operations) to narrow down the regions.

5.1 License Plate Localization

As used in the object detection stage. We use a background model to remove the irrelevant information. The first stage involves computing (horizontal) gradients. The motivation being that the license plate region will contain a lot of gradients (Due to the text present in it).
We also experimented with vertical and total (Horizontal and vertical) gradients. The improvements were not significant.
This is followed by morphological operations as discussed earlier (In object detection). The following slide contains the intermediate results.

5.2 License Plate Recognition

We used the pretrained OpenALPR model (on US License Plates) for the task and tried out their detection pipeline on our proposed number plate regions.

License Plate Recognition pipeline (From OpenALPR):

- **Char Analysis -** Find connected character-sized blobs in binarized image
- **Deskew -** Affine transformation to change perspective to straight-on view
- **Character Segmentation -** Individual character segmentation and cleaning
- **OCR -** Analyzing each character given by the Character Segmenter and predicting a character
- **Post Processing -** Creating a list of plate possibilities based on OCR confidences

The results of License Plate detection and recognition are shown below,

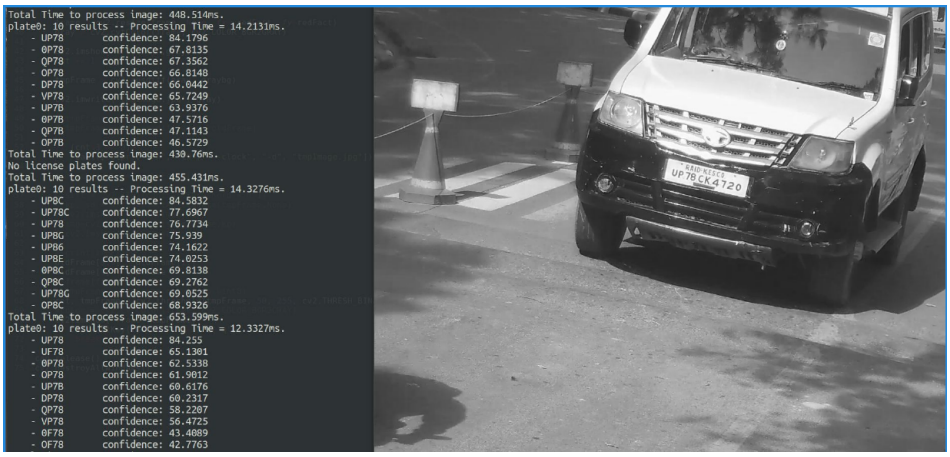


Figure 11: License Plate Recognition

6 Possible Improvements

- Get more relevant data for digit-wise OCR
- Use a Convolutional Neural Network for the OCR task
- Use Semantic Image Segmentation (eg. DeepLab Segmentation Engine). Use Semantic Segmentation to get object proposals
- The tracking can be done using SIFT correspondence, Mean Shift, learning a small CRNN (Convolutional Recurrent Neural Network (Shi et al., CoRR 2015)) to get corner change values given an images and their corresponding corner positions
- Using faster-RCNN (Ross Girschick et al.) (Training requires a huge amount of data)
- Using a Scalable Vocabulary Tree of SIFT Features for fast and efficient class detection (Nister et al.).

7 Implementation Details

- Languages Used: Python
- Libraries Used:
  - Scikit-Learn
  - Tesseract
  - OpenALPR
  - Keras
  - Theano
- Pre-Trained Models Tested:
  - OpenALPR
  - LPO
  - OverFeat

References

[1] Openalpr: License plate recognition. `https://github.com/openalpr/openalpr`.

[2] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998.

[3] Stephen Gould, Tianshi Gao, and Daphne Koller. Region-based segmentation and object detection. In *Advances in neural information processing systems*, pages 655–663, 2009.

[4] Deng-Yuan Huang, Chao-Ho Chen, Wu-Chih Hu, Shu-Chung Yi, Yu-Feng Lin, et al. Feature-based vehicle flow analysis and measurement for a real-time traffic surveillance system. *Journal of Information Hiding and Multimedia Signal Processing*, 3(3):279–294, 2012.

[5] Marcin Iwanowski. Automatic car number plate detection using morphological image processing. *Przeglad Elektrotechniczny*, 81(3):58–61, 2005.

[6] Philipp Krahenbuhl and Vladlen Koltun. Learning to propose objects. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1574–1582. IEEE, 2015.

[7] Jonathan Krause, Jia Deng, Michael Stark, and Li Fei-Fei. Collecting a large-scale dataset of fine-grained cars. 2013.

[8] Mahmood Ashoori Lalimi and Sedigheh Ghofrani. An efficient method for vehicle license plate detection in complex scenes. *Circuits and Systems*, 2(04):320, 2011.

[9] Pauline C Ng and Steven Henikoff. Sift: Predicting amino acid changes that affect protein function. *Nucleic acids research*, 31(13):3812–3814, 2003.

[10] Choudhury A Rahman, Wael Badawy, and Ahmad Radmanesh. A real time vehicle license plate recognition system. In *null*, page 163. IEEE, 2003.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[12] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

[13] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013.

[14] Ronghui Zhang, Pingshu Ge, Xi Zhou, Tonghai Jiang, and Rongben Wang. An method for vehicle-flow detection and tracking in real-time based on gaussian mixture distribution. *Advances in Mechanical Engineering*, 5:861321, 2013.