CS771A: Project Report

CLASSIFICATION OF OBJECTS FROM THE VIDEO STREAM

Saurav Shekhar
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
sshekh@iitk.ac.in

Ekansh Gupta
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
egupta@iitk.ac.in

Palak Agarwal
Department of Computer Science & Engineering
Indian Institute of Technology, Kanpur
palakag@iitk.ac.in

Abhay Kumar
Department of Electrical Engineering
Indian Institute of Technology, Kanpur
abhayk@iitk.ac.in

Abstract

In this project, we explored various methods of detecting and classifying objects in a video stream into three classes- Pedestrian, Two-Wheeler and Four-Wheeler. For each frame, We extract region proposals using selective search or background subtraction, get features using a variety of techniques and classify them using various standard classifiers available.

1 Introduction

Classification of objects in a video frame is a well-known problem. Much work has been already done in this area and people have done this using various techniques of machine learning like image processing, Convolutional Neural networks, Object tracking etc.. We attempted to use some of these techniques here to get a analysis of using different methods for classification and feature extraction.

2 Preprocessing

We extracted the bounding boxes from labelled data for each frame and each video and sorted them according to their corresponding labels. The text files containing all the object details (`enum.py` was used to sort columns in order of frame numbers for easier pre-processing) were used for this. In total we get around 120,000 labelled images using this. Corresponding program is `fr.py`. Accuracy for feature extraction + classification is reported on this dataset. We also created the `None` class using background subtraction. Code for this is in `None features/none_extraction.py`.

2.1 Background Extraction

Background Subtraction methods are widely used for detecting and tracking moving objects in videos. It is useful in many applications such as traffic monitoring, video surveillances. Background subtraction is highly problematic when the camera is also moving. Basic Background Subtraction (BBS) algorithm computes the absolute difference between the current frame and a static background frame and compares each pixel to a threshold. Background is modeled by a mixture of Gaussians (MOG). Backgrounds having fast variations are not easily modeled with just a few Gaussians accurately, and it may fail to provide sensitive detection. However, Background subtraction

gives reasonable accuracy in the slowly varying background. We have used background subtraction for the following purposes:

- Object Detection : Pixels associated with the same object should have the same label; one can accomplish this by performing a connected component analysis. All the connected components are computed and they are considered as active regions if their area exceeds a given threshold. This step is usually performed after a morphological filtering to eliminate isolated pixels and small regions. Detected objects will be used as testing images for the trained classifiers.
- Defining 'None' Class: After foreground extraction, different random patches from the background images are collected and are assigned the 'None' label. These random patches will be used to extract features for the 'None' class. Basically, 'None' class is neither 'Pedestrian' nor 'Two-Wheeler', nor 'Four-Wheeler'. This has been implemented in `none_extraction.py`

2.2 Selective Search

Selective Search [UvdSGS13] generates all possible object locations in a given image. It is a data-driven approach which combines the strength of segmentation and exhaustive search. It exploits the structure of the image to generate object locations using the bottom-up segmentation and different sampling techniques. Different grouping criteria and complementary colour spaces are exploited to capture all scale and diversified light- ing conditions. Selective search uses hierarchical grouping algorithm to for image segmentation. Because the process of grouping itself is hierarchical, it captures locations at all scales by continuing the grouping process until the whole image becomes a single region.

We used selective search for

- Extracting candidate proposals from given frame of a given video. Program `sel_search.py` saves each proposal separately in a given folder
- In their landmark RCNN [GDDM14] paper, Girshick et al used selective search proposals used to find overlaps (IoU) with training boxes and labelled the poposals if IoU was $>$ 0.5. Else the proposal was used as a None label. We have used this with HoG features to generate features from each frame. Since slective search gives around 500 region proposals for each image, we use around 1k frames for generating data. Code is in `selHog.py`.

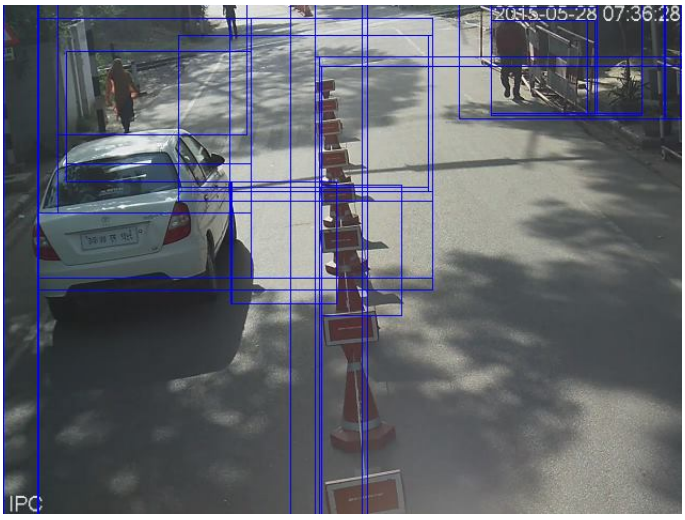

Figure 1: Figure illustrating the possible object locations given as output of selective search

3 Feature extraction

3.1 SIFT

Scale-invariant feature transform (or SIFT) is an algorithm in computer vision to detect and describe local features in images. We used SIFT to extract distinguishable and characteristic features (also known as keypoints) for images. `SIFT/sift.py` was used to calculate SIFT features for 60,000 of the 120,000 images (due to memory constraints). It resulted in the generation of roughly 8,000,000 keypoint descriptors.

A keypoint descriptor is a 128 dimensional vector that denotes the state of a specific area of an image. Different images have different number of keypoint descriptors. For training a classifier, we however need input vectors of a fixed dimension. Hence visualizing all the 8,000,000 keypoints as points on a 128 dimensional plane and then using a Bag-of-Words approach results in the creation of histograms for each image.

Due to memory constraints, 1,000,000 keypoints uniformly sampled from the 8,000,000 descriptors were used to cluster them into 1000 groups using k-means clustering. The result is a "code-book" that can subsequently assign a keypoint descriptor to any of these 1000 groups, essentially producing a 1000 dimentional vector for each image. The code file `SIFT/cd.py` computes the codebook and the file `SIFT/hist.py` computes the respective histograms. After these steps we get a 1000 dimensional vector for all 120,000 images that can be used for classification.

3.2 HOG

The histogram of oriented gradients (HOG) is a feature descriptor used for the purpose of object detection. The technique counts occurrences of gradient orientation in localized portions of an image.

This was another way of extracting features for images. HoG has the added advantage that it does result in vectors of a fixed size if all the images are of a fixed size and the HoG parameters are kept the same. For this reason, all images were scaled to 100×100 pixels and a cell size of 50×50 was used to compute 250 orientations, resulting in a 1000 dimensional vector for each of the 120,000 images. `HOG/hog.py` extracts the HoG features of the images.

3.3 CNN

We can extract features from images using convolutional neural networks. Training a CNN from scratch requires a lot of computational resources. Therefore, we used a model pre trained on the ILSVRC 2012 dataset and used caffe [JSD+14] for extracting features. `CNN` folder contains all the required code. File list for training is in `file_list.txt`. See `http://caffe.berkeleyvision.org/gathered/examples/feature_extraction.html` for explanation on feature extraction. We subtract the mean image of ILSVRC dataset from each image to get better features. `extractor.py` converts output data (in levelDB format) into numpy arrays and `train_cnn.py` runs classifiers and calculates accuracy. We use fc7 features, which are the higest level of features in the network with a batch size of 50.

3.4 RBM

Restricted Boltzmann Machine have been used in our project as it has been proved to be a good feature extraction method for text and image classification[GHW06]. It is another neural network technique which is basically a 2 layer network as shown in Figure 2.

We used BernoulliRBM method for dimensionality reduction on our data. We later applied classifiers on this reduced data to get the results. Before applying the classifier, all the images were resized to a fixed size of 78 x 128. The classifier can be varied on the learning rate, number of components and number of iterations.

We got the following results when we used python's sklearn function. The code is in the file `bernoulli_rbm.py`.

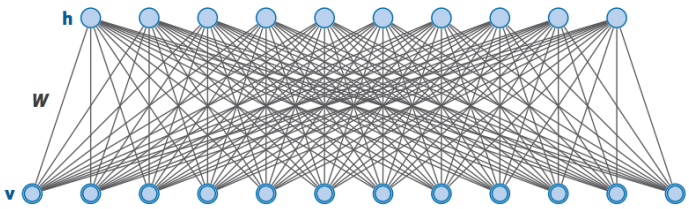

Figure 1

Restricted Boltzmann machine. The top layer represents a vector of "hidden" stochastic binary variables **h,** and the bottom layer represents a vector of "visible" stochastic binary variables **v.**

Figure 2: Figure taken from [Sal15]

BernoulliRBM.
Classifier- LinearSVC
Accuracy- 47%

3.5 Autoencoder

In simple cases, Auto-encoder is a 3-layer network where the hidden layer has lesser neurons than the input and the output layer. The input and the output layer have the same number of neurons. Auto-encoders are another neural network method used for extracting the features. It is another one of the non-linear dimensionality reduction methods which uses neural nets.

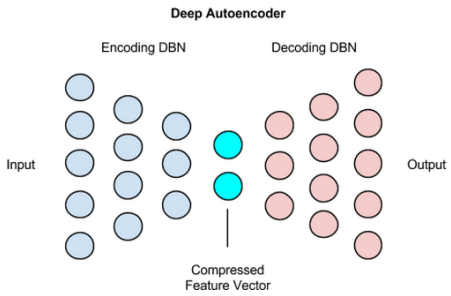

Figure 3: Figure taken from [dl4]

The code used by us has been taken from [den], which implements Denoising Auto encoders, which is a stochastic version of the auto encoder. For our data, we had 2 hidden layers of 1000 neurons each and the algorithm gave an accuracy of 47 %.

4 Results

Here is the accuracy results for different classifiers and feature extraction techniques over the dataset created in 2.

4.1 SVM

Differnt Features	Accuracy
HOG	0.78
SIFT	0.77
Selective Search + HOG	0.22
RBM	0.47
CNN	0.92

Table 1: SVM accuracy

4.2 Decision Trees

Differnt Features	Accuracy
HOG	0.72
SIFT	0.72
Selective Search + HOG	0.67

Table 2: Decision tree accuracy

4.3 Random Forests

Differnt Features	Accuracy
HOG	0.83
SIFT	0.82
Selective Search + HOG	0.78
CNN	0.90

Table 3: Random forest accuracy

4.4 Background subtraction

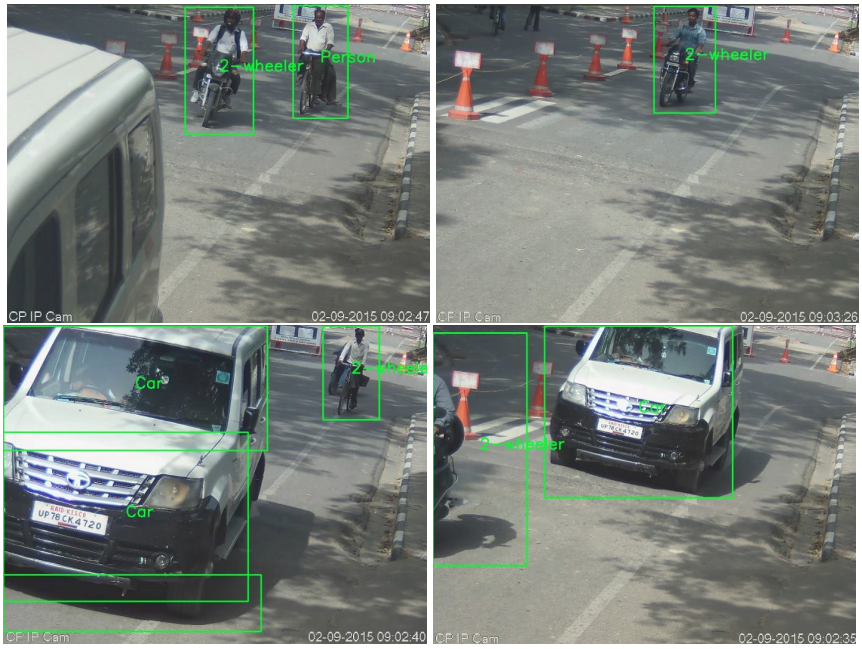

Figure 4: Classified detected objects

5 Future scope

For images, RCNN [GDDM14] solves the problem completely from object detection to classification. Developments over it [Gir15] and [RHGS15] have tried to pipeline the process and increase its speed. We tried running RCNN and its variants on the images but were unsuccessful. We can try comparing out current methods to RCNN in future.

Also, currently detection and classification happens for each frame independently. We can use continuous frame data to implement tracking of objects in the video.

6 Acknowledgement

We thank Prof Harish Karnick for giving us the opportunity to work on this project and for providing guidance whenever required. We also thank Mr Nagendra Yadav and Saurabh for providing help with the GPU setup.

References

[den] Denoising autoencoders.

[dl4] Deep autoencoders.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[GHW06] Peter V Gehler, Alex D Holub, and Max Welling. The rate adapting poisson model for information retrieval and object recognition. In *Proceedings of the 23rd international conference on Machine learning*, pages 337–344. ACM, 2006.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[RHGS15] 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.

[Sal15] Ruslan Salakhutdinov. Learning deep generative models. In *The Annual Review of Statistics and Its Application*, pages 361–387. 2015.

[UvdSGS13] Jasper RR Uijlings, Koen EA van de Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.