Classification of objects from Video Data (Group 30)

Sheallika Singh	Vibhuti Mahajan	Aahitagni Mukherjee	M Arvind
12665	12792	12001	12385

1 Motivation

Video surveillance has been employed for a long time to monitor security sensitive areas such as banks, department stores, highways, crowded public places and borders. The increase in the number of cameras in ordinary surveillance systems has overloaded the human operators with high volumes of data and made it infeasible to ensure human monitoring of sensitive areas for long times. In order to improve the response time to forensic events, assisting the human operators with identification of important events in video by the use of smart video surveillance systems has become a critical requirement. "Smart" video surveillance systems requires fast, reliable and robust algorithms for object detection and classification.

2 Aim

Using the IIT Kanpur Surveillance Video Dataset, we aim to build different classification models that detect objects of interest and correctly classifies between a person, a car, a cycle, and a motorcycle.

3 Data Preprocessing

The VATIC data in video format was split in frames. Since consecutive frames depicted almost the same objects, frames were picked at an interval of 15 for further processing. Using the labelled data for training set or object detection methods for testing set the images of the objects were cropped. These cropped images were then resized (64×128) and also gray scaled before features were extracted from these images.

4 Object Detection

For detecting vehicles and pedestrians from individual frames, 2 techniques were used:

- background-foreground separation
- region based CNN.

On an average the RCNN method performed better than background-foreground separation method and was used for further processing.

4.1 Background-Foreground Separation

The built-in cv2.createBackgroundSubtractorMOG2() function of OpenCV with default options was applied on the frames for separating the objects from the background.

It uses a method to model each background pixel by a mixture of K Gaussian distributions (K = 3 to 5). The weights of the mixture represent the time proportions that those colours stay in the scene. The probable background colours are the ones which stay longer and more static. This model is the same as has been described in the paper "An improved adaptive background mixture model for real-time tracking with shadow detection" by P. KadewTraKuPong and R. Bowden [2].

4.2 Region-based Convolutional Neural Networks

The system used is the same as in the paper "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" by Shaoqing Ren et al.(NIPS 2015) [4].

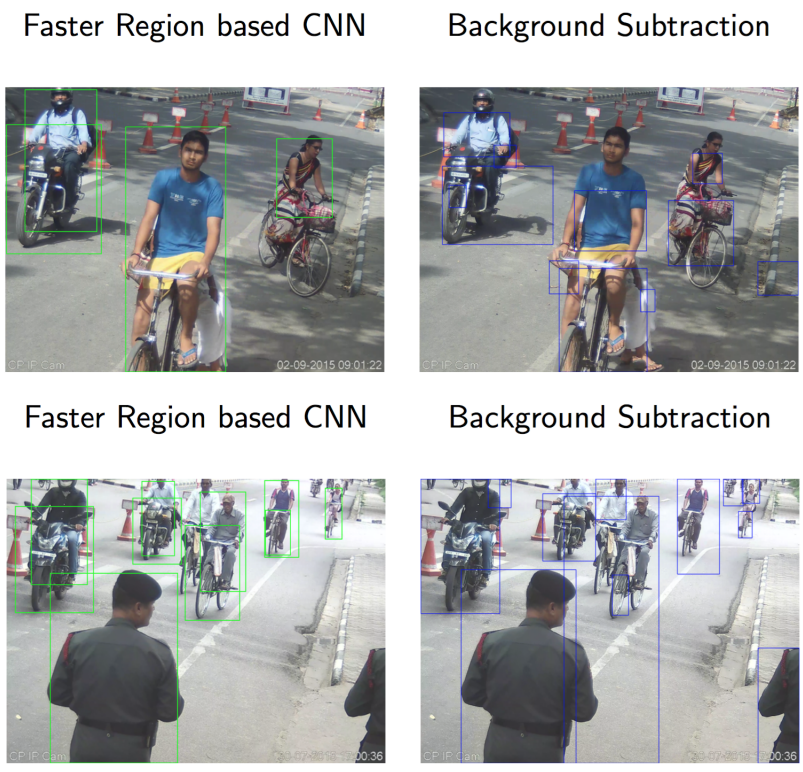
The system consists of three modules. The first generates category-independent region proposals. These proposals define the set of candidate detections available to the detector. The second module is a large convolutional neural network that extracts a fixed-length feature vector from each region. The third module is a set of class-specific linear SVMs.

Selective Search algorithm is used to generate category-independent region proposals. A 4096-dimensional feature vector is extracted from each region proposal using the Caffe implementation(`https://github.com/rbgirshick/py-faster-rcnn`) of the CNN. Features are computed by forward propagating a mean-subtracted 227×227 RGB image through five convolutional layers and two fully connected layers.

In order to compute features for a region proposal, the image data in that region is converted into a form that is compatible with the CNN (its architecture requires inputs of a fixed 227 227 pixel size). All pixels are warped in a tight bounding box around it to the required size. Prior to warping, the tight bounding box is dilated so that at the warped size there are exactly p pixels of warped image context around the original box ($p = 16$).

At test time, selective search is run on the test image to extract around 2000 region proposals (selective searchs fast mode). Each proposal is warped and forward propagated through the CNN in order to read off features from the desired layer. Then, for each class, each extracted feature vector is scored using the SVM trained for that class. Given all scored regions in an image, a greedy non-maximum suppression (for each class independently) is applied, that rejects a region if it has an intersection-over-union (IoU) overlap with a higher scoring selected region larger than a learned threshold.

4.3 Comparison



A visual comparison between the 2 methods for 2 different frames has been shown above. It is noticeable that the RCNN differentiates overlapping objects more clearly than Background-Foreground separation. Some objects moving together were detected in one box.

5 Feature Extraction

After the objects are detected the images of objects are cropped and features are extracted from the cropped images. 3 types of features were extracted and used separately from the detected objects, for the task of classification. These are

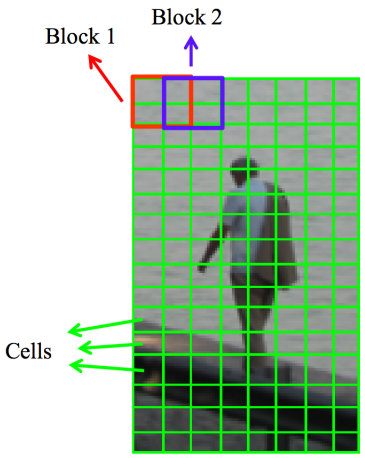
- Histogram of Oriented gradients(HOG) [1]
- Dense Scale Invariant Feature Transform(D-SIFT)
- BVLC-Alexnet Convolutional Neural Network

5.1 HOG feature extraction

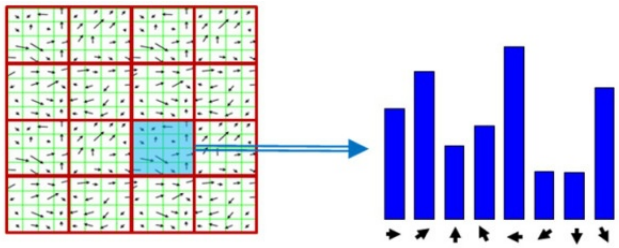
The histogram of oriented gradients (HOG) [1] is a feature descriptor that uses occurrences of gradient orientation in localized portions of an image.The steps are described as follows.

First, in the image, the centred horizontal and vertical gradients are computed with no smoothing. The gradient orientations and magnitudes are calculated separately. In color image, for each pixel, the color channel with the highest gradient magnitude is taken. The filter masks for centred gradient are $(-1, 0, 1)$ and $(-1, 0, 1)^T$.

Then the image is divided into overlapping blocks and gradient orientations are quantized. For example, for a 64×128 image, it it divided into 16×16 blocks of 50% overlap, which gives $7 \times 15 = 105$ blocks in total. Each block consists of 2x2 cells with size 8x8.



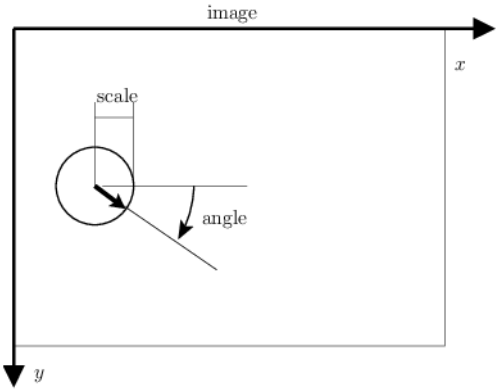
The gradient orientations are then quantized into 9 bins. The gradient magnitude is taken as the vote, which is also weighted with Gaussian to downweight the pixels near the edges of the block. The votes are interpolated bi-linearly between neighboring bin centers.



Finally, the histograms are concatenated to get a 1D feature vector of dimension $105 \times 4 \times 9 = 3780$

5.2 DSIFT feature

A SIFT feature is a selected image region (also called keypoint) with an associated descriptor. A SIFT keypoint is a circular image region with an orientation. It is described by a geometric frame of four parameters: the keypoint center coordinates x and y, its scale (the radius of the region), and its orientation (an angle expressed in radians).



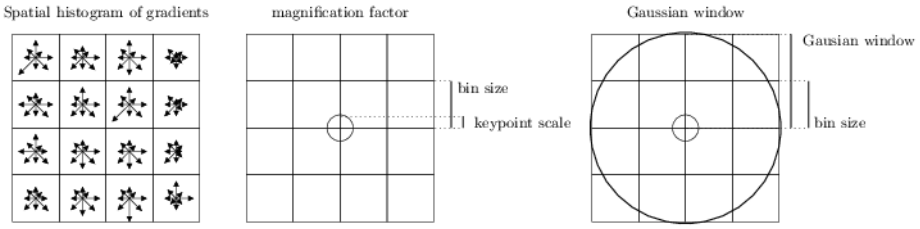
SIFT keypoints are circular image regions with an orientation.

Searching keypoints at multiple scales is obtained by constructing a so-called Gaussian scale space. The scale space is just a collection of images obtained by progressively smoothing the input image, which is analogous to gradually reducing the image resolution.

Keypoints are further refined by eliminating those that are likely to be unstable, either because they are selected nearby an image edge, rather than an image blob, or are found on image structures with low contrast.

A SIFT descriptor is a 3-D spatial histogram of the image gradients in characterizing the appearance of a keypoint. The gradient at each pixel is regarded as a sample of a three-dimensional elementary feature vector, formed by the pixel location and the gradient orientation.

Samples are weighed by the gradient norm and accumulated in a 3-D histogram h, which (up to normalization and clamping) forms the SIFT descriptor of the region. An additional Gaussian weighting function is applied to give less importance to gradients farther away from the keypoint center. Orientations are quantized into eight bins and the spatial coordinates into four each, as follows:



The SIFT descriptor is a spatial histogram of the image gradient.

D SIFT[1] module implements a fast algorithm for the calculation of a large number of SIFT descriptors of densely sampled features of the same scale and orientation. Based on SIFT algorithm, dense SIFT makes some new assumptions:

- The location of each keypoint is not from the gradient feature of the pixel, but from a predesigned location
- The scale of each keypoint is all the same which is also predesigned
- The orientation of each keypoint is always zero.

With this assumptions, DSIFT can acquire more feature in less time than SIFT does.

[1]D SIFT features were extracted using the modified implementation of `http://www.vlfeat.org/overview/dsift.html` for python language

5.3 CNN feature extraction

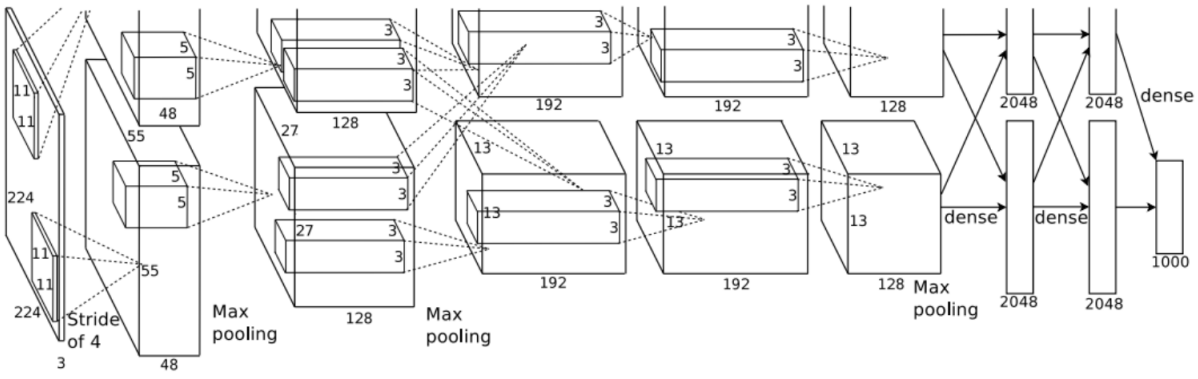
Features were extracted from the penultimate layer of the pre-trained Caffe model of a convolutional neural netwok.BVLC AlexNet [3] is trained on LSVRC-2010 ImageNet training set consisting of 1.2 million images.

The net contains eight layers with weights; the first five are convolutional and the remaining three are fullyconnected. The output of the last fully-connected layer is fed to a 1000-way softmax which produces a distribution over the 1000 class labels.

This network maximizes the multinomial logistic regression objective, which is equivalent to maximizing the average across training cases of the log-probability of the correct label under the prediction distribution.

The kernels of the second, fourth, and fifth convolutional layers are connected only to those kernel maps in the previous layer which reside on the same GPU. The kernels of the third convolutional layer are connected to all kernel maps in the second layer. The neurons in the fullyconnected layers are connected to all neurons in the previous layer. Response-normalization layers follow the first and second convolutional layers. Max-pooling layers follow both response-normalization layers as well as the fifth convolutional layer. The ReLU non-linearity is applied to the output of every convolutional and fully-connected layer.

The first convolutional layer filters the $224 \times 224 \times 3$ input image with 96 kernels of size $11 \times 11 \times 3$ with a stride of 4 pixels. The second convolutional layer takes as input the (response-normalized and pooled) output of the first convolutional layer and filters it with 256 kernels of size $5 \times 5 \times 48$. The third, fourth, and fifth convolutional layers are connected to one another without any intervening pooling or normalization layers. The third convolutional layer has 384 kernels of size $3 \times 3 \times 256$ connected to the (normalized, pooled) outputs of the second convolutional layer. The fourth convolutional layer has 384 kernels of size $3 \times 3 \times 192$, and the fifth convolutional layer has 256 kernels of size $3 \times 3 \times 192$. The fully-connected layers have 4096 neurons each. The architecture is depicted below.



6 Classification

For classifying the detected objects in 4 classes, different classifiers were used:

- Logistic Regression
- Support Vector Machine
- Random Forest
- Adaboost (with base classifiers are decision stumps)

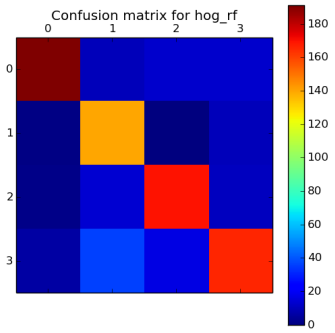
6.1 Classification with HOG features

The confusion matrix corresponding to each model fitted using HOG features is depicted below. The y-axis is for the actual class ans x-axis for predicted class.

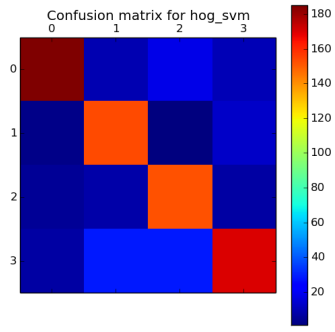
- class 0 is for bicycle
- class 1 is for car
- class 2 is for Motorbike
- class 3 is for person
- Rickshaw was not included as one of the classes and for training model because of the very less number of images extracted from video.

From the resulting confusion matrix it can be seen that all the models except the adaboost classifier. We also observed that wherever the object is being classified incorrectly it is the car being predicted as motorbike.

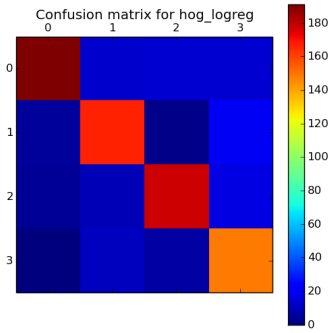
RandomForest



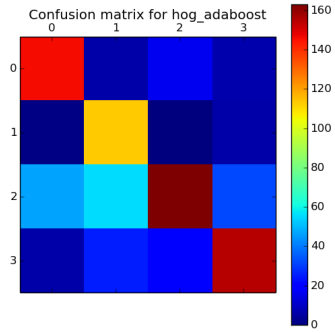
SVM



Logistic Reg



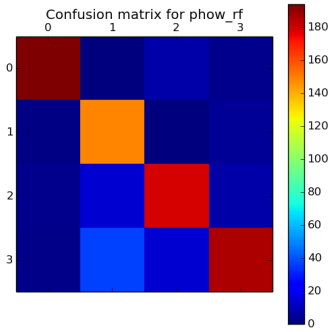
AdaBoost



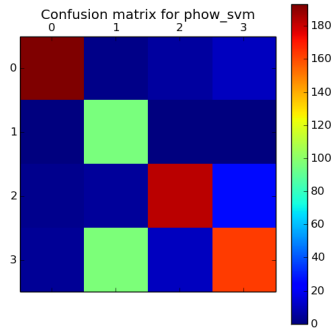
6.2 Classification with DSIFT features

Classes are the same as defined for the HOG. Similar results were observed for the models trained on the D-SIFT features. Linear SVM and Adaboost classifier were not that accurate in classifying car correctly.

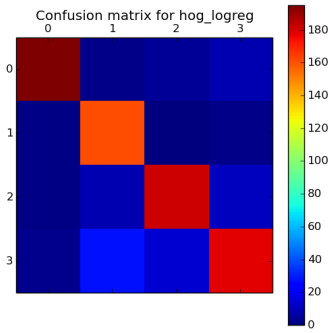
RandomForest



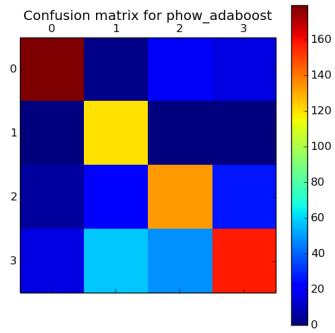
SVM



Logistic Reg



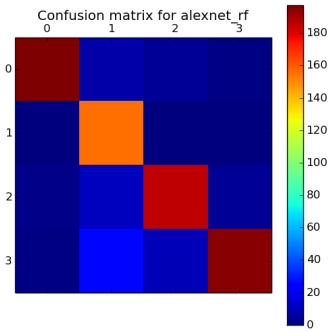
AdaBoost



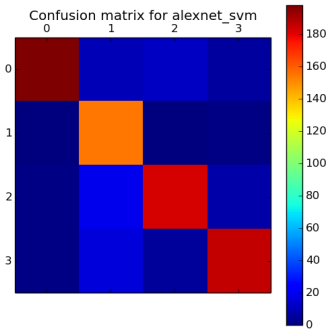
6.3 Classification with AlexNet penultimate layer features

We received the best results for AlexNet features among all other feature extraction techniques that we have used. The only backdraw for using AlexNet was that it required GPU due to large memory space requirements.

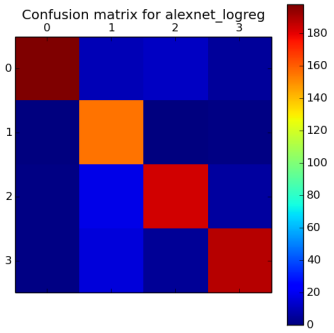
RandomForest



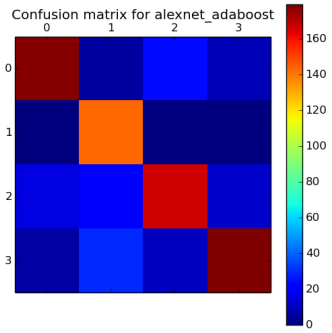
SVM



Logistic Reg



AdaBoost



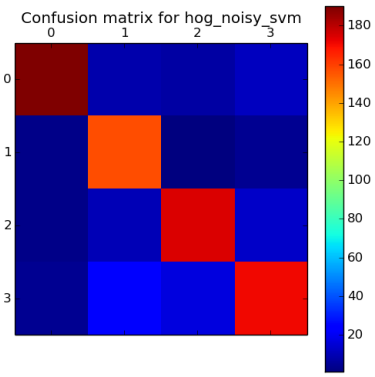
6.4 Accuracy Table

Model	HOG	PHOW	Alexnet
Logistic Regression	85.62	89.50	90.75
SVM	82.62	79.37	90.25
Kernel SVM	86.63	91.25	90.13
Random Forest	83.37	88.5	91.75
Adaboost	72.25	73.75	83.25

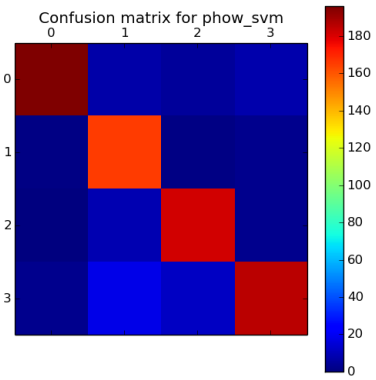
6.5 Kernel SVM

Due to the high rate of misclassification for linear SVM for D-SIFT features, we tried to incorporate non-linearity in the model by taking a gaussian kernel. There were significant improvements in the classification accuracy as depicted in the tables below.

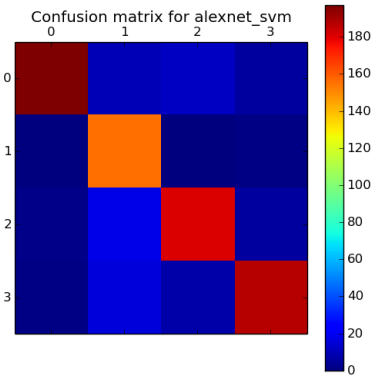
HOG



D-SIFT



AlexNet



6.6 Adding noise to the data

We also tried adding noise to the training data to make models more robust, but adding noise doesn't improve the accuracy. So we did not proceed further with it after 2 models.

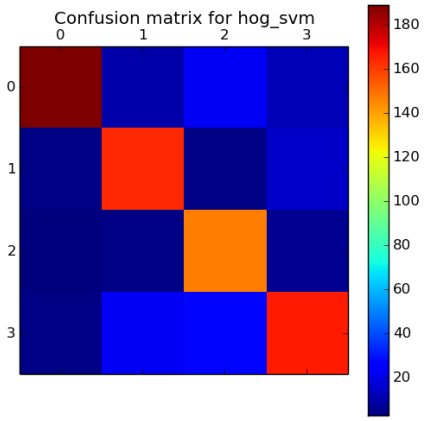


Figure 1: SVM trained on noisy data

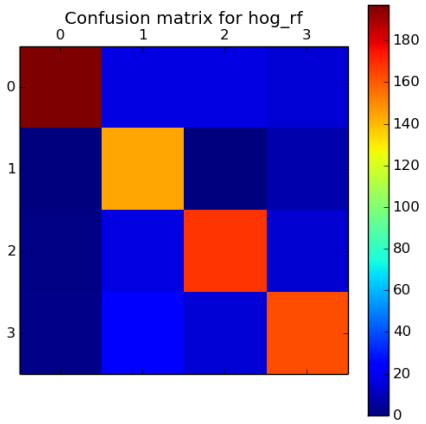


Figure 2: Random Forest trained on noisy data

References

[1] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

[2] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[4] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.