AAD: Adaptive Anomaly Detection through traffic surveillance videos

Mohammad Farhadi Bajestani
mfarhadi@asu.edu

Seyed Soroush Heidari Rahmat Abadi
Soroush.Heidari@asu.edu

Seyed Mostafa Derakhshandeh Fard
smd.fard@asu.edu

Roozbeh Khodadadeh
rkhodada@asu.edu

Abstract

Anomaly detection through video analysis is of great importance to detect any anomalous vehicle/human behavior at a traffic intersection. While most existing works use neural networks and conventional machine learning methods based on provided dataset, we will use object recognition (Faster R-CNN) to identify objects labels and their corresponding location in the video scene as the first step to implement anomaly detection. Then, the optical flow will be utilized to identify adaptive traffic flows in each region of the frame. Basically, we propose an alternative method for unusual activity detection using an adaptive anomaly detection framework. Compared to the baseline method described in the reference paper, our method is more efficient and yields the comparable accuracy.

1 Introduction

Anomaly detection is usually defined as the set of techniques and methods in machine learning and data mining that can detect an unusual or non-conforming pattern in the data. Anomaly detection is of special importance in image recognition because of the ubiquity of surveillance systems and the emerging of self-driving cars which heavily rely upon anomaly detection for correct operation. Obviously, the objective is to reduce the role of humans in detection of anomalies to a minimum and at the same time have the capability to detect anomalies no matter how rare they are.

One of the challenges in defining anomalies in the context of video surveillance is that in a non-stationary process, the concept of normality itself changes. Entry of a new object into the view is an anomalous event in the first few moments. After the first few moments, it becomes a normal event from the system's viewpoint. To address this problem, We will present an adaptive method to deal with the non-stationary nature of the problem.

Another challenge is that anomalous events are rare. We will also propose a method to deal with the sparsity of the data.

In short, our method is a combination of supervised and unsupervised learning, is based on analyzing a spatial signal, uses blocks to eliminate noise (Which would otherwise happen if we use only pixels), is capable of lossless representation the video sequence, and uses variance as the criteria to measure deviation from normality.

1.1 Anomaly Detection

Anomaly detection is a broad topic with many applications in various fields. Here, we confine our discussion of anomaly detection to its application in video surveillance.

The definition of anomaly is associated with our understanding of the normal state. For example, players running in a football field is considered normal activity, while people running in a parade is an instance of unusual activity [1]. One immediate consequence of this notion is that it is very easy for human observers to identify anomalies because they usually have a very good understanding of the context of the video. We need to come up with a way to define an anomaly for computers. For our purposes we define anomalies as temporal or spatial outliers which means "an action done in an unusual time or in unusual location"[1].

With this view of anomaly, it seems that clustering is one natural choice for detecting anomalies. We can cluster features together. Anomaly is detected if there are objects that are not contained in any of our "normal" clusters. This is the basis of the technique used in the paper written by Li et al[2].

To successfully detect anomalies, any proposed method needs to distinguish between empty and crowded scenes and between local and global anomalies. Behavior classification might also be needed to enable the algorithm to differentiate the normal from anomalous state.

1.2 Object Recognition

Object recognition is defined as the technology for finding and identifying objects in an image or video sequence[3]. Another way of thinking about neural networks is that they are function approximation machines[4]. Going back to the discussion on kernel tricks, neural networks differ from traditional models in that they *learn* the kernel transform Φ compared to using a generic or a manually engineered kernel function. Image recognition mostly employs Convolutional Neural Networks (CNN). These networks use the convolution operation in place of matrix multiplication in at least one of their layers. To recognize objects, we first need to detect images. The first stage is usually an edge detection layer. Edge detection can be done by subtracting pixel values from neighboring pixels. The result is then passed through a pooling layer. Pooling layer replaces a pixel's value (or a block) with the summary statistics of its neighbors. The used statistics can be maximum, average or any other valid summary statistic function. Convolution in general is an expensive operation and running it on whole images can be very computationally expensive.

The state of the art in the image recognition field is to use a subset of neural networks called Regions with Convolutional Neural Networks (R-CMN). Following the previous paragraph, the next step in R-CNN is image segmentation which divides image into separate regions that may be of interest. The algorithm will also give bounding boxes around each region.

R-CNN is still slow. Researchers have developed Fast-RCNN and Faster-RCNN which perform considerably faster compared to RCNN. We use Faster-RCNN in our project to recognize and detect images. Despite its limits, faster R-CNN adds a significant improvement to the efficiency of the our general algorithm.

2 Related Works

Our reference paper[2], uses a motion influence map(MIM) based on optical flow. Optical flow is the technique to recover three dimensional motion from a sequence of time-ordered two-dimensional images[5]. Optical flow allows the estimation of projected two dimensional image motion as either instantaneous image velocities or discrete image displacements[5]. It is a convenient motion representation method.

The reference paper then builds the MIM. MIM is based on the idea that a moving object can influence nearby blocks in two ways: 1)motion direction, and 2)motion speed. We can calculate all such influences and create a map in which each block represents the quantized motion orientation of the block.

Our contribution to the reference paper is twofold. First, we use Fast-RNN image recognition to detect and categorize objects that will later help us to detect the type of anomaly. Second, instead of a motion influence map and clustering, we use a Gaussian distribution function that represents the level of anomaly in each block. We use the output of our faster-RCNN object recognition algorithm to assign object probabilities to various regions in the image.

3 Adaptive Anomaly Detection (AAD)

This section first introduces how to detect and localize anomalies via motion characteristics estimation over consecutive frames. After that, the object recognition algorithm applied on original data and the distribution map construction will be elaborated. Finally, the proposed adaptive anomaly detection approach will be explained in details. Figure 1 shows the complete Anomaly Detection algorithm.

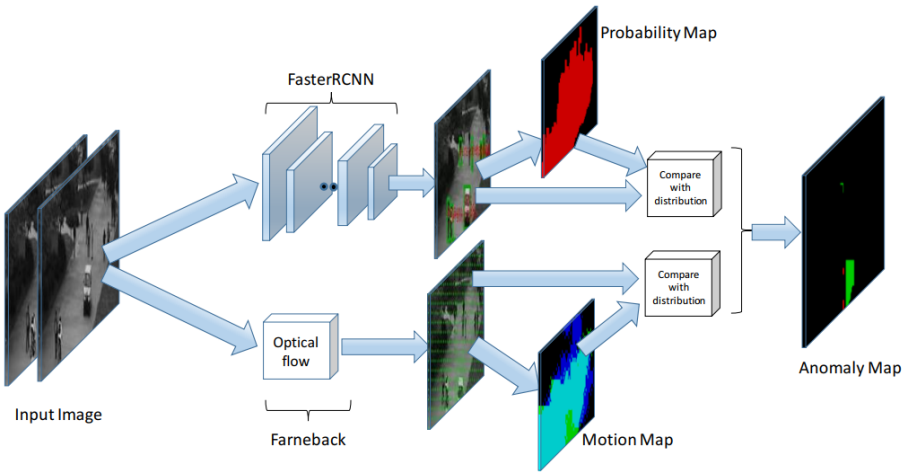

Figure 1: Complete Adaptive Anomaly Detection Flow

3.1 Optical Flow

In this paper, we observe different motion patterns at the block-level to spot unusual activities within a crowded scene. The motion patterns is estimated by optical flow using Farneback method. Optical flow is the pattern of apparent motion caused by brightness changes. Given two subsequent frames, the optical flow method tries to calculate how quickly the pixel is moving between the two frames which are taken at times $t + \Delta t$ and what is the movement's direction. One of main limitations of optical flow is that it fails to extract motion characteristics from static or very slow moving objects. Also, based on our primary results, pixels' displacements between consequent frames wasn't significant. Therefore, we picks every other frames to compare with and generate the optical flow's output. Initial output is a 3D matrix that has displacement information of each pixel across the image in both (x, y) directions, that basically points to where that pixel can be found in another frame.

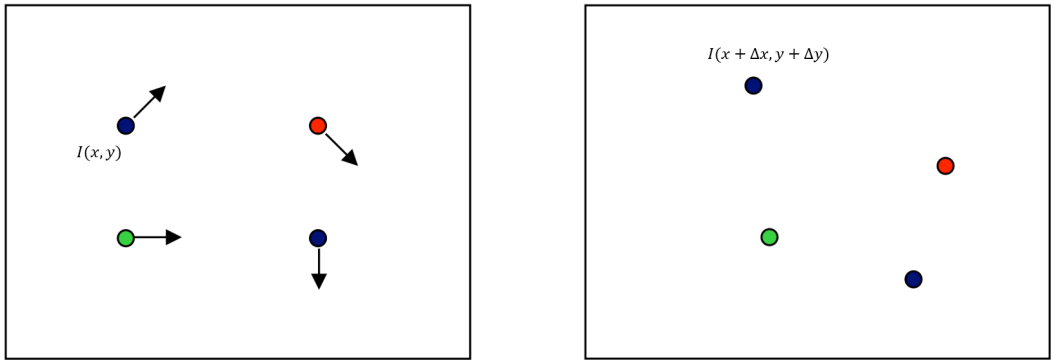

Figure 2: Pixel movement across the frame after Δt

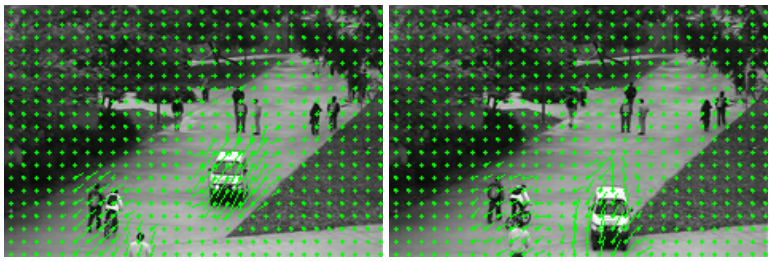

Figure 3: Optical Flow Outputs.

Given two frames at times $t-2$ and t, a voxel at location (x, y) with intensity $I(x, y)$ will have moved by $\Delta x, \Delta y$ between the two image frames with Δt time difference. The Brightness Constancy equation is as follows:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{1}$$

Then it can written as:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{2}$$

From these equations it follows that:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t}\frac{\Delta t}{\Delta t} = 0 \tag{3}$$

which can be transformed to:

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \tag{4}$$

Thus, the Optical Flow equation is stated as:

$$I_x V_x + I_y V_y = -I_t \tag{5}$$

The equation 5 cannot be solved with two unknown variables I_x and I_y. Therefore, several optical flow methods introduce additional conditions to calculate the actual flow and one of them is Gunner Farneback's algorithm which is explained in [6]. We have used OpenCV library that is equipped with Gunner Farneback's algorithm.
After estimating the pixel-wise dense optical flow, we process the data through two pooling stages in order to remove the noise and reduce the processing time. First, we apply $2 * 2$ average pooling with two major objective which are noise and dimensionality reduction that consequently results in improved speed and performance. Then, we kept decreasing the dimension by set the maximum change to be the the block representative via a $2 * 2$ max pooling stage. Figure 3 shows that Optical flow clearly is able to detect the car as an anomaly due to its speed difference with pedestrians.

3.2 Object Recognition

This object recognition is our complementary tool to expose and localize the anomalies within a packed and crowded scene. We have utilized Faster R-CNN that is an object recognition algorithm to find unusual objects in our surveillance videos. Faster R-CNN is equipped with Region Proposal Network (RPN) in order to rapidly and efficiently process the whole frame and determine the region of interest that needs further processing. The input image goes through a network of convolution layers and transforms to a convolutional feature map. Next, The RPN uses the computed a feature map to detect a predefined number of regions (bounding boxes), that may contain objects.

The bounding boxes is used as the input to the Region of Interest (ROI) pooling layer, which performs a pooling operation on a part of the input map that corresponds to region proposals in the original image. Finally, the R-CNN module either classifies the content of the bounding boxes or it adjust the bounding boxes coordinates to better fit the content and consequently can be classified precisely.

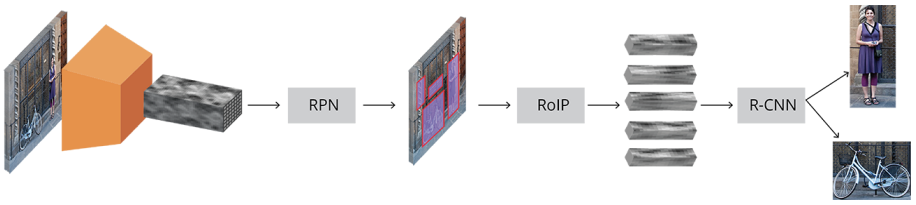

Figure 4: Faster R-CNN Architecture.

3.3 Distribution Map

In the proposed Distribution Map, we introduce a 5-tuple that includes pixel-wise mean, max, min, variance, and number of frames of video sequences. We build two distribution maps based on both optical flow and object recognition outputs.Then, the distribution map is used to expose pixel's values that shows drastic changes $(\mu \pm k\sigma)$ compared to velocity distribution or is classified as an unusual object in comparison with object's history in that area of the frame. Also, We update the computed temporal 5-tuple of video sequences continuously after each change in pixel's value. In order to calculate the temporal mean and variance over streaming video frames, each time that the new data is loaded and differs from previous values, the new 5-tuple is computed. The μ is updated based on incoming data as follows:

$$\mu_N = \frac{\mu_{N-1}(N-1) + x_N}{N} \tag{6}$$

and then if we detect an anomaly in the video sequences, we reduce by a half the number of frames, so the weight of the incoming data will be larger. That means, the algorithm will adapt itself and stop detecting false positive anomaly.Therefore the updated mean after anomaly detection is calculated in this way:

$$\mu_N = \frac{\mu_{N-1}\frac{(N-1)}{2} + x_N}{\frac{N-1}{2} + 1} \tag{7}$$

Next we calculated the σ^2 as follows.

$$\sigma_N^2 = \frac{(N-2)\sigma_{N-1}^2 + (x_N - \bar{x}_N)(x_N - \bar{x}_{N-1})}{N-1} \tag{8}$$

The Equation 8 can be derived from Welford's method as described below. Welfords method is a usable single-pass method to calculate the variance. It is quite surprising and computationally efficient that how simple the sums of squared differences for N and $N-1$ samples results in the new variance[7].

$$(N-1)\sigma_N^2 (N-2)\sigma_{N-1}^2 \tag{9}$$

$$= \sum_{i=1}^{N}(x_i - \bar{x}_N)^2 - \sum_{i=1}^{N-1}(x_i - \bar{x}_{N-1})^2 \tag{10}$$

$$= (x_N - \bar{x}_N)^2 + \sum_{i=1}^{N-1}\left((x_i - \bar{x}_N)^2 - (x_i - \bar{x}_{N-1})^2\right) \tag{11}$$

$$= (x_N - \bar{x}_N)^2 + \sum_{i=1}^{N-1}(x_i - \bar{x}_N + x_i - \bar{x}_{N-1})(\bar{x}_{N-1}\bar{x}_N) \tag{12}$$

$$= (x_N - \bar{x}_N)^2 + (\bar{x}_N - x_N)(\bar{x}_{N-1} - \bar{x}_N) \tag{13}$$

$$= (x_N - \bar{x}_N)(x_N - \bar{x}_N - \bar{x}_{N-1} + \bar{x}_N) \tag{14}$$

$$= (x_N - \bar{x}_N)(x_N - \bar{x}_{N-1}) \tag{15}$$

$$\tag{16}$$

4 Experimental Results and Analysis

4.1 Dataset Description

To evaluate the performance of the proposed method, we put to the test on several dataset, i.e., the PASCAL VOC, UMN, and USDC, which contain global and local anomalies.

A. *PASCAL Visual Object Classes (VOC)*[1]

The VOC challenge is a benchmark in visual object category recognition and detection, providing the vision and machine learning communities with a standard dataset of images and annotation, and standard evaluation procedures. Organized annually from 2005 to 2012, the challenge and its associated dataset has become accepted as the benchmark for object detection. In VOC 2012, the train/validation data (1.9 GB) has 11,530 images containing 27,450 ROI annotated objects and 6,929 segmentations. The goal of PASCAL VOC is to recognize objects from a number of visual object classes in realistic scenes (i.e. not pre-segmented objects). It is fundamentally a supervised learning problem in that a training set of labeled images is provided. It contains twenty object classes as shown in figure 5. For each of the twenty classes, VOC predict the bounding boxes of each object of that class in a test image, with associated real-valued confidence. We use PASCAL VOC to train faster RCNN and detect objects, in our work. [8]

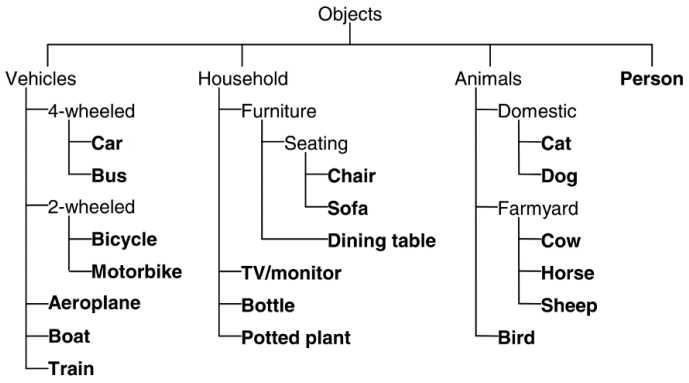

Figure 5: PASCAL Visual Object Classes

B. *UMN Dataset*[2]

We validated the effectiveness of the proposed method on UMN dataset. It consists of 11 video clips of crowded escape scenarios from three different indoor and outdoor scenes. It includes 7740 frames in total (1450, 4415 and 2145 for scenes 13, respectively), where the frame size is 320 240. The normal events are pedestrians walking randomly on the square or in the mall, and the abnormal events are human spread running at the same time.

C. *UCSD Dataset*[3]

There are two sets of video clips in the UCSD dataset: Ped1 and Ped2. Figure (6a6b) shows sample frames Ped1 and Ped2 dataset. The dataset provides both frame- and pixel-level ground truths, which localize the regions where the unusual activities occur. Ped1 consists of 34 training clips and 36 test clips, where the frame size is 238158. Ped2

[1] Available at http://host.robots.ox.ac.uk/pascal/VOC/

[2] Available at http://mha.cs.umn.edu/Movies/Crowd-Activity-All.avi

[3] Available at http://www.svcl.ucsd.edu/projects/anomaly

(a) (b)

Figure 6: (a) UCSD Ped1 and (b) UCSD Ped1 sample frames

consists of 16 training clips and 12 test clips, where the frame size is 360 240. The training clips include only normal activities of people walking along a pathway. For the test clips, unusual activities such as bicycling, skating, and driving a cart are taking place.

4.2 Framework

This project consists of two main feature extraction which needs different framework to implement. object recognition part has been implemented using PyTorch [9] and optical flow has been implemented using OpenCV[10].

PyTorch is an open source library in python for running machine learning algorithm [9]. This library has been implemented based on Torch. It is primarily developed by Facebook's artificial-intelligence research group. This library provide two high-level features, Tensor computation with GPU acceleration and deep neural networks.

OpenCV (Open Source Computer Vision Library) is an open source library to run machine learning and computer vision algorithm. This library has interface for C++, java, Python and MATLAB. This library best option to run image processing algorithm such as optical flow which has been used in our project.

All modules of this project has been implemented in python.An Ubuntu machine with two Xeon CPUs, 64 GB RAM and one NVIDIA TITAN-X PASCAL has been used in this project.

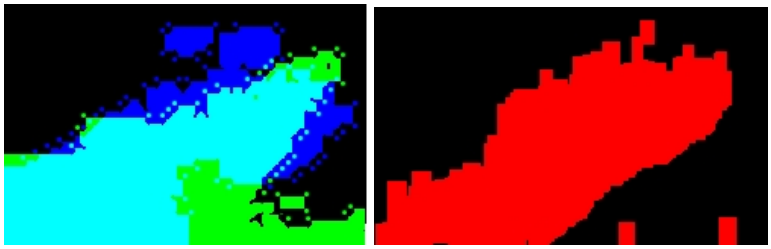

Figure 7: Motion Distribution Map and Object probability Map (Just Person probability) in UCSD dataset

4.3 Distribution Map

In figure 7, we show the motion distribution of UCSD in left figure. In this figure we show the mean of dx and dy which calculated frame by frame, dx represented by green color and dy represented by blue. In positions which one of the mean is significantly bigger than other one, we have dark blue or green. In middle of frame which we have motion to all direction we are more close to bright blue. black area represent area which we didn't observe any movement.

Next figure in 7 shows the probability map for one specific class. In this figure we select person as target class and show the existence probability of this class in each pixel with read color in UCSD dataset. It is clear that in region we have higher chance of observing human in that region we have very high probability to observe human. Dark region shows places which probability of observing human is close to zero. In real case, we have a tensor with size of image which each pixel has 22 variable that 21 of these keep number of observation for specific object and last total number of observed objects. In figure 7 we just shows the result for human class.

Combination of these create our distribution for UCSD dataset. In our we create same distribution for UMN dataset.

4.4 Parameter Settings

Object Recognition module: For object recognition module we are using FasterRCNN [11]. We are using ResNet 101 as feature extractor for FasterRCNN which has been pre-trained over ImageNet. The initial learning rate has been set to 0.001, for learning decay we gamma learning decay with 0.1 decay ratio. The model has been trained over Pascal Dataset. The object selection threshold has been set to 0.8, this means objects detected with probability lower than 0.8 will be removed from our consideration map. The model has been trained in 20 epochs with 10000 iteration. model with lowest error rate over test has been selected for our adaptive anomaly detection system.

Optical flow module: For this module we are using Gunnar Farneback's algorithm in OpenCV. To get better result we compare two last frame with each other. In optical flow function we set number of pyramids to 3, averaging windows to 15, number of iteration to 5 and standard deviation of the Gaussian to 1.2.

Hyper parameters: we are creating a window over distribution and if our incoming data is inside we would classify it as normal. Otherwise it would be treated as an anomaly. The window size is equal to $k\sigma$ centered at μ. The generated figure from both datasets stem from the fact that changing the window size k results in significant True Positive and False Positive variation in final results. In this work, we modified our model's k values with a range from 1 to 6.

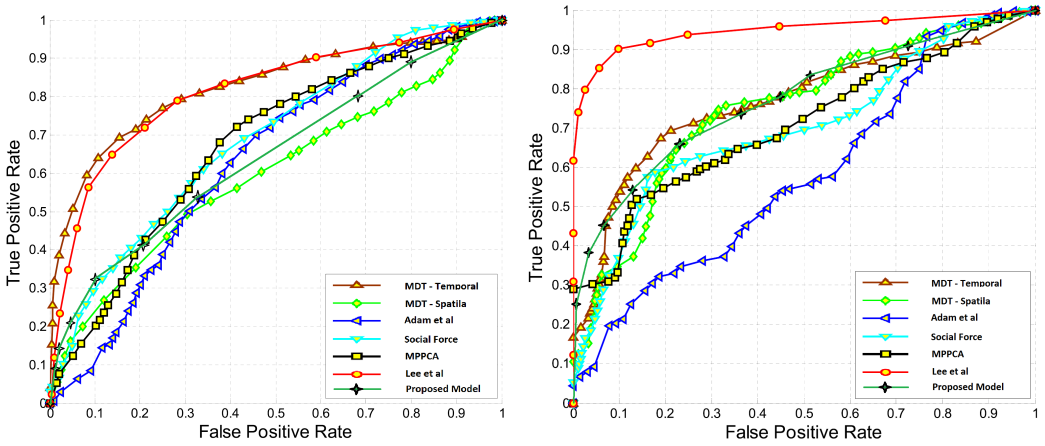

Figure 8: (a) UCSD Dataset - Ped1 (b) UCSD Dataset - Ped1

4.5 Performance Evaluation

Similar to the approach taken in the reference paper[2], We also use Receiver Operating Characteristic (ROC) curves to evaluate performance. ROC curves show the trade-off between hit rates and false alarms[12]. The idea is to map classification rates to a 2 by 2 matrix (for a binary classification algorithm) called *confusion matrix*. This matrix shows the number of true as well as false positives and negatives. An ROC graph takes this matrix and gives a two dimensional box plot in which the horizontal axis is the false positive rate and the vertical axis is the true positive rate. In this graph

the point (0,1) corresponds to perfect classification (no false positives) and point (1,1) corresponds to arbitrary assignment of class (a lot of false positives). We plotted our model performance based on different window size k. Note that the objective is getting large true positive values and small false positive values, that means we are interested in getting results close to upper left corner of the plot. Figure (8a,8b) shows that if we set the $k = 1$, we will end up with both false positive and true positive equal to 1. Also, if we set the value to the other extreme which is $k = 6$ because the model does not detect any object as anomaly, both values of false positive and true positive tends to zero. Also, the performance difference between two dataset can caused by dissimilar point of view and variation in object's velocity (both speed and its direction). Keep in mind that the plotted performance is obtained without considering object recognition module's output. Thus, if we add the object recognition module to the test model, we experience slightly better performance (True Positive $= 0.56$, Flase Positive $= 0.36$ for Ped1). The reason behind poor performance of the object recognition module is that the Faster R-CNN is not trained over this dataset.

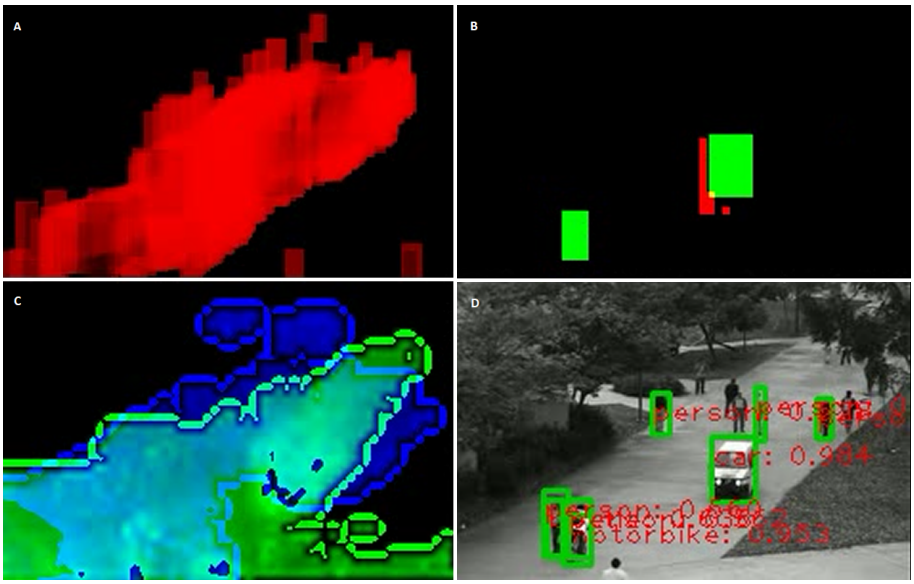

Figure 9: (a) Human Probability (b) Anomaly Map (c) Motion Distribution Map (d) Input Frame + Object Recognition Output

Also, Figure 9b shows that the presence of the car in the street is correctly detected based on optical flow distribution map, also the dark shadow over the red-colored region in fig. 9a support the same result. Although, we didn't meet the state of art accuracy, we achieved decent results compared to previous works. More importantly, we gained a extremely faster (2) running time due to simpler model complexity

5 Conclusion & Future Works

Unlike previous methods described in the literature, we proposed a simpler *adaptive* model that results in getting faster and reliable results. Also, it can deal with non-stationary nature of anomaly detection problem and adapt itself after observing an anomaly. Moreover, the added object recognition module (Faster R-CNN) improves the true positive with the tradeoff of having trivial false positive. The proposed method has a constraint when there is a strong noise in the input frames as the motion distribution map is built based on the motion velocity of the moving objects.
Many different training, tests, and experiments have been left for the future due to lack of time. Future works concern training Faster R-CNN over related anomaly detection datasets. Also, validating the proposed AAD model over longer video sequences.

References

[1] O. P. Popoola and in K. Wang. "video-based abnormal human behavior recognitiona review,". *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):865–878, November 2012.

[2] Dong-Gyu Lee, Heung-Il Suk, Sung-Kee Park, and Seong-Whan Lee. Motion influence map for unusual human activity detection and localization in crowded scenes. *IEEE Transactions on Circuits and Systems for Video Technology*, 25(10):1612–1623, oct 2015.

[3] Wikipedia contributors. Outline of object recognition — Wikipedia, the free encyclopedia, 2018. [Online; accessed 27-April-2018].

[4] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[5] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[6] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In Josef Bigun and Tomas Gustavsson, editors, *Image Analysis*, pages 363–370, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[7] Robert F. Ling. Comparison of several algorithms for computing sample means and variances. *Journal of the American Statistical Association*, 69(348):859–866, 1974.

[8] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.

[9] "facebook brings gpu-powered machine learning to python. `https://www.infoworld.com/article/3159120/artificial-intelligence/facebook-brings-gpu-powered-machine-learning-to-python.html`. Accessed: 2018-04-26.

[10] "opencv-about page. `https://opencv.org/about.html`. Accessed: 2018-04-26.

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Curran Associates, Inc., 2015.

[12] Tom Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861 – 874, 2006. ROC Analysis in Pattern Recognition.