


Dynamic video anomaly detection and localization using sparse denoising autoencoders

Medhini G. Narasimhan¹ · Sowmya Kamath S.¹ 

Received: 31 October 2016 / Revised: 7 June 2017 / Accepted: 11 June 2017 /
Published online: 21 June 2017
© Springer Science+Business Media, LLC 2017

Abstract The emergence of novel techniques for automatic anomaly detection in surveillance videos has significantly reduced the burden of manual processing of large, continuous video streams. However, existing anomaly detection systems suffer from a high false-positive rate and also, are not real-time, which makes them practically redundant. Furthermore, their predefined feature selection techniques limit their application to specific cases. To overcome these shortcomings, a dynamic anomaly detection and localization system is proposed, which uses deep learning to automatically learn relevant features. In this technique, each video is represented as a group of cubic patches for identifying local and global anomalies. A unique sparse denoising autoencoder architecture is used, that significantly reduced the computation time and the number of false positives in frame-level anomaly detection by more than 2.5%. Experimental analysis on two benchmark data sets - UMN dataset and UCSD Pedestrian dataset, show that our algorithm outperforms the state-of-the-art models in terms of false positive rate, while also showing a significant reduction in computation time.

Keywords Anomaly detection · Deep learning · Sparse autoencoders · Crowded scene parsing · Real-time systems · Unsupervised learning

✉ Medhini G. Narasimhan
medhini95@gmail.com

Sowmya Kamath S.
sowmyakamath@nitk.edu.in

¹ Department of Information Technology, National Institute of Technology Karnataka, Surathkal, India

1 Introduction

Video analysis of crowded environments such as public parks, universities, stadiums and stations has garnered significant interest in pattern recognition and surveillance applications that aim to detect any anomalies. An anomaly or abnormality can be defined as any unusual event that is aberrant from those usual ones that frequently take place at a particular spot in a given scene. Manual scrutiny of surveillance videos to identify such abnormalities is a highly time consuming process. In situations where a robbery or trespassing incident is reported, often many days of recordings need to be checked. It is, therefore, highly possible that due to manual error such abnormalities might be missed, and so, automatic systems are particularly beneficial in such situations. Anomaly detection has its applications in medical imaging as well. Computer aided abnormality detection in MRI Images is useful in diagnosing malformations often missed by the human eye.

The description of unusual activities in complex scenes like pedestrian walkways and parks is an arduous task, as an anomalous event can be anything ranging from a dog to a vehicle. As such, it is practically impossible to develop an exhaustive list of all anomalies in any given situation. Yet, this problem is usually tackled by deploying high-dimensional feature descriptors to define all known irregularities. An event can be considered a normal activity in one context and suspicious in another. So when an event is not represented by sufficient samples in the training data set, the unusual behavior is labeled as a context-related anomaly. To develop an efficient model to be trained with these descriptors, an extensive amount of training data is needed. Also, the computational complexity of this is usually very large because of high dimensional feature sets required to represent large variety of anomalous situations.

Scene modeling and anomaly behaviors are challenging problems in computational approaches for anomaly detection. Occlusion is a problem when multiple objects are present in a crowded scene, which often makes object tracking difficult. Also, within the same scene, the behavior of individuals may vary greatly, due to which, distinguishing between anomalies and a normal event is a difficult task. To this end, training videos are used to learn a set of models which are used in the test phase. This method labels a test video as an anomaly if it does not match the learned model, and specific feature descriptors are used to build the reference models.

In this paper, an unsupervised learning model capable of detecting and localizing anomalies in videos is proposed. Our primary goal is to develop a robust anomaly detection system for crowded scene videos. Although the model was trained and tested on a pedestrian walkway, it can be easily extended to any other situation by training on the pertinent video data. Further, the proposed method also encompasses a novel technique of representing the videos using local and global descriptors that account for any spatial and temporal changes. These two views are integrated using Gaussian classifiers to identify the anomalous frame (detection) and the location of the anomaly within that frame (localization). Previous works relied on several low level features which not only added to the computational time and model complexity but also restricted the applicability of the model. To overcome this, we use a deep learning method known as sparse denoising autoencoders to automatically learn feature representations of a video. We introduce the concept of denoising as described by Vincent et al. [28] as a sparse autoencoder [21]. Our main contribution was incorporating a drop out layer, which was converted to a denoising autoencoder to curb model over fitting. KL Divergence regularization [8] helped further generalize the model. The problem of dying weights was countered by using Parametric ReLU [9] activation function. The proposed anomaly detection and localization framework was applied to popular data sets and

experimentally validated. Our method is capable of real-time anomaly detection in a video which is being currently played. The proposed approach was evaluated on two real world datasets, the UCSD Pedestrian dataset¹ and UMN dataset.² It was observed that it outperformed the state-of-the-art methods in terms of false positive rate by achieving an improvement of 2.5% over the previously reported best results. The main advantage of our approach is the significant reduction in execution time. Our algorithm takes 4.3 seconds to evaluate each frame whereas the existing best method takes 7.5 seconds per frame. This greatly reduces the time taken to detect anomalies in large videos with many frames.

The rest of the paper is organized as follows: Section 2 presents a discussion on existing methods for automatic anomaly detection in videos. Section 3 illustrates the proposed methodology and describes the working of each of the components. In Section 4, we describe the experiments performed for the validation of the proposed model and compare its performance with 7 other state-of-the-art methods. Section 5 summarizes the advantages of the proposed framework and explains directions of future work, followed by references.

2 Related work

Automatic anomaly detection techniques can be categorized as *trajectory learning methods*, *spatial-temporal gradients* and *feature learning models*. We discuss the state-of-the-art in each of these categories.

Trajectory learning methods In these methods, the pedestrians and other objects in motion are extracted using background subtraction and frame difference. Trajectory based methods involve learning the normal walking patterns of pedestrians from normal trajectories obtained using tracking algorithms. The trajectories from a test video are then compared with these established standard and any deviation is reported as anomalous. Li et al. [14] represented normal behavior trajectories as cubic B-Spline curves and a normal dictionary set is constructed. Dictionary sets are divided into Route sets and every test trajectory is compared with Route Set using sparse reconstruction analysis to determine the residual. If the residual is larger than a certain threshold, it is classified anomalous. Kong et al. [13] described a technique to identify the goal of individuals within a crowd based on the same tracking principle. The authors proposed another method [7] where any moving objects that are detected are initially classified as vehicles or pedestrians through a co-trained classifier. This method capitalizes on the multi-view information of objects and results in a framework that is able to learn motion patterns automatically respectively for vehicles and pedestrians, thus addressing the problem of diversity of objects in a scene.

The main drawback of trajectory based methods is the effect of occlusion. In crowded scenes, it is common for some objects to be hidden behind other objects partially or completely, causing partial or full occlusion. Since trajectory modeling first detects an object to determine its trajectory, the technique will fail if the detection of the object is unsuccessful due to occlusion [27]. Further, crowded scenes often have several trajectories which need to be tracked in a single scene. This poses a great challenge in trajectory based learning methods.

¹UCSD Pedestrian dataset <http://www.svcl.ucsd.edu/projects/anomaly/dataset.htm>

²UMN Detection of Unusual Crowd Activity dataset http://mha.cs.umn.edu/proj_events.shtml

Spatial-temporal gradients These techniques are suitable for crowded scenes where multiple objects are difficult to track. Adam et al. [1]’s method was based on multiple local monitors, where each local monitor detects unusual event by collecting low-level statistics and generates an alert. The final decision is taken by integrating all these alerts. In Jiang et al. [10]’s method, anomalies are detected in the spatio-temporal context considering sequential, point and co-occurrence anomalies. This is achieved by employing a hierarchical data mining approach wherein a frequency based analysis is performed at each level. Rules are established to define normalcy in a spatio-temporal context and their method detects hazardous events. However, synthesizing rules for all scenarios is a tedious task. In Zhang et al. [33], a Locality Sensitive Hashing function is defined which hashes anomalous activities into different feature buckets and thus helps filter out anomalies. Li et al. [15] proposed an unsupervised statistical learning framework to detect anomalies in crowded scenes by dividing videos into cubic spaces. Using sparse coding and clustering, their method can learn spatial and temporal gradients of global activities and local salient behavior patterns. With this, an activity pattern code book is constructed that extends their model to localize anomalies through multiple scale analysis. Their model outperforms five other well known methods but it cannot be updated to automatically work on any input video. For each new video, the dictionary learning process needs to be repeated which makes their technique scenario specific. A social force model was used for the first time by Mehran et al. [19] to detect and localize anomalous events. These represent interaction forces between moving particles which are placed in the form of a grid on top of the image. Force flow is estimated by mapping these interaction forces into the image frame. Normalcy in crowd behavior is modeled by randomly chosen spatio-temporal patches. Events are classified using a bag of words technique and localized using the forces of interaction defined earlier.

Feature learning models For solving the problem of occlusion, a third class of anomaly detection algorithms emerged. These methods involved slicing the video into frames or cuboids and extracting features such as texture and optical flow gradient from these segmented regions of the video. Reddy et al. [23] proposed a technique where foreground frames obtained from foreground segmentation are split into cells that are non-overlapping and features like texture, motion and size are extracted from these for cell-wise anomaly detection. However, the cell size used is determined heuristically and can cause large variations of accuracy across different scenes. Mahadevan et al. [18] and Li et al. [16] modeled normal crowd behavior by capturing both guise and motion by using a mixture of dynamic texture (MDT) models. Spatial discriminate saliency detector aided in detection of spatial anomalies whereas background behavior subtraction was used to detect temporal anomalies. The final detection result was obtained by combining the two results obtained from both temporal and spatial dimensions. Techniques for mining for patterns in temporal data have been studied in works such as Aljawarneh et al. [3] and Radhakrishna et al. [22]. Radhakrishna et al. [22] proposes a novel approach for estimating temporal association pattern prevalence values and a fuzzy similarity measure is proposed. Aljawarneh et al. [3]’s approach was able to find temporal patterns whose true prevalence values vary similar to a reference support time sequence satisfying subset constraints through estimating temporal pattern support bounds and using a novel fuzzy dissimilarity measure which holds monotonicity to find similarity between any two temporal patterns.

A Markov random field model based on space-time was employed by Kim and Grauman [12] to detect activities that are considered abnormal in video. A local region was represented as a node and spatio-temporal relations with other local regions was shown with

the help of links. Normal patterns at each node were learned by mixture of optical flow distributions and probabilistic PCA models. For any new optical flow patterns in the test videos, a degree of normality was computed using maximum a posteriori algorithm. Their model was also adaptive to new situations. Wu et al. [30] proposed a system for modeling crowd flow and anomaly detection by efficiently representing trajectories and introducing chaotic invariant features into the scenes. A probabilistic framework for detecting anomalous behaviors in crowded scenes by learning dominant behaviors online was proposed by Roshtkhari and Levine [24]. Crowd behavior was shown as a spatio-temporal arrangement of small cuboids of the video with anomalous instances being those which had less frequency of occurrence in terms of arrangement. Another application of anomaly detection was presented by Aljawarneh et al. [2], where an anomaly-based intrusion detection system based on a hybrid model was used to determine the intrusion scope threshold degree based on the network transaction data.

Cheng et al. [6] divide anomalies into two kinds: local and global. Global anomalies denote anomalies across frames whereas local anomalies fixate on the anomalous patch within a frame. Cheng et al. [6] use a hierarchical feature representation and Gaussian process regression technique is used to detect anomalies. A linear sparsity model for superior class separability and detection of anomalies involving multiple objects was proposed by Mo et al. [20]. Saligrama and Chen [26] focus on localizing spatio-temporal anomalies by using optimal decision rules. They use a local nearest neighbor search to define score functions of the empirical rules. Antić and Ommer [4] proposed an indirect method of anomaly detection by defining a set of hypotheses to describe normal behavior in the foreground by finding examples that support the hypotheses. They describe a probabilistic model which uses statistical inference to localizes abnormalities. In extension to the works related to local and global anomaly detection in videos, Sabokrou et al. [25] proposed a framework for defining spatio-temporal feature descriptors for a video. These descriptors are passed as inputs to two Gaussian classifiers which are then combined using a fusion technique. Appearance and Motion DeepNet Model introduced by Xu et al. [32] uses deep neural networks to automatically learn feature representations coupled with a deep fusion technique. Motion and appearance features are learned separately using stacked denoising autoencoders and then fused together in this work.

3 Methodology and framework

With reference to state-of-the-art methods discussed earlier, we identified several shortcomings, which we intend to address in the proposed methodology. Feature extraction is highly dependent on the context of the video. However, features extracted for some crowded scenes are not applicable everywhere and there is no significance associated with the features. For complex scenes, many features would have to be considered, which increases the dimension of the descriptor, thus resulting in added complexity and computational time. High false positive rate is also a common shortcoming in all these methods. To address these setbacks, we propose a model that efficiently overcomes the aforementioned drawbacks, which we discuss in detail in Section 3. Our significant contributions are manifested in the following aspects -

1. A sparse denoising autoencoder with Parametric ReLU as the activation function and KL Divergence as the regularization function is used to automatically learn features. Thus, our model is context independent and can adapt to being trained on any data.

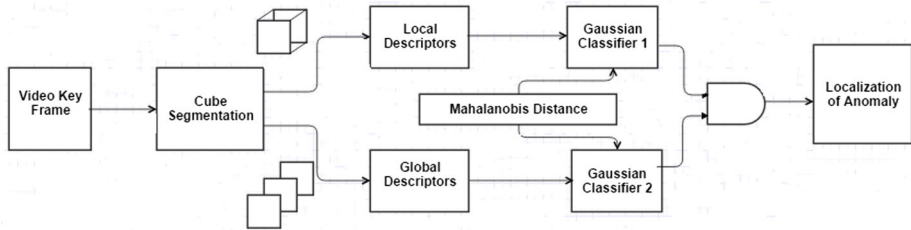


Fig. 1 Proposed methodology

2. Different feature descriptors are used for local and global anomalies. Local feature descriptors account for spatio-temporal changes and global feature descriptors account for changes across many frames.
3. Gaussian classifiers are utilized to classify the local and global descriptors separately and a fusion technique is used to combine the results of both.
4. Inclusion of drop-out layers in the autoencoder and KL-Regularization considerably diminishes over-fitting of the model.

Figure 1 describes the proposed model's architecture. The system works in real-time and reports anomalies in an incoming video. The methodology consists of five distinct phases, each of which is described in detail next.

1. Preprocessing the video to generate cubic patches.
2. Generation of local descriptors.
3. Generation of global descriptors using sparse denoising autoencoders.
4. Gaussian Classification of local and global features.
5. Aggregation of results from both the classifiers and anomaly localization.

3.1 Video preprocessing and cubic patch generation

Each video is represented by a set of key frames. For a given set of key frames we obtain non overlapping cubical patches as shown in Fig. 2. The main concept of anomaly detection is that the probability of occurrence of an anomalous patch is much less compared to a normal one. There will be one or more patches which are similar to the neighboring patches and

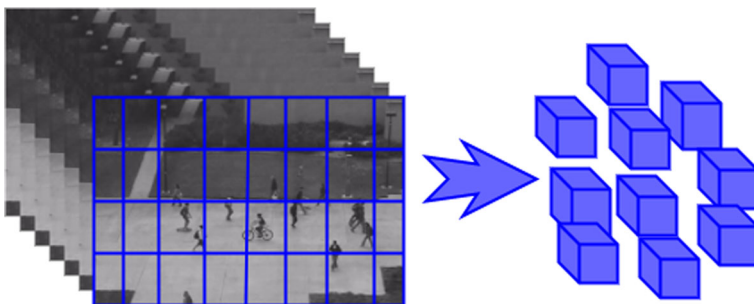


Fig. 2 Cubic patch generation

have a high probability of occurrence in the video. Thus, an anomalous patch would meet the following requirements:

1. The situation of a similarity between the patches that are normal and the patches which are adjacent, which is described by the spatial descriptor, follow a certain pattern which cannot be seen when it comes to the similarity of the anomaly patches and patches that are adjacent to them.
2. Temporal changes in normal patches follow a particular pattern. This is not seen in the temporal changes of abnormal patches.
3. Lower probability of occurrence of a patch that is anomalous than that of patches that are normal under the given circumstances.

The points 1 and 2 are local to a set of frames and correspond to spatio-temporal changes. They are represented by the local feature descriptors. The 3rd point is quite similar to the scene that is global in nature and is encoded by a set of global descriptors. Considering all the aforementioned dimensions at once can result in an inefficient and exceedingly complex framework. Thus, we construct the two independently. To arrive at a decisive conclusion, the results from the models are fitted to Gaussian distributions and a boundary that is used to make a decision is calculated for every model. A patch is classified as an anomaly depending on the results obtained from both classifiers, as shown in Fig. 3.

3.2 Local descriptor generation

For a given cubic patch, the local descriptors are those which describe its spatio-temporal relations with its neighboring patches. Algorithm 1 illustrates the process of local descriptor generation. The local descriptor is composed of two parts, which are generated as per the defined processes.

- *Spatial Descriptors*: Similarity of the chosen patch with 8 surrounding patches is considered, in addition to the five frames which constitute the patch. Frame 1 is compared with Frame 2 and 2 with 3 and so on, after which a total of 12 spatial descriptors are obtained.
- *Temporal Descriptors*: The patches that are obtained before and after the patch under consideration are considered for determining any anomaly quickly, such that it is identified even before proceeding to the next patches in the video. This process produces 14 local descriptors for each patch.

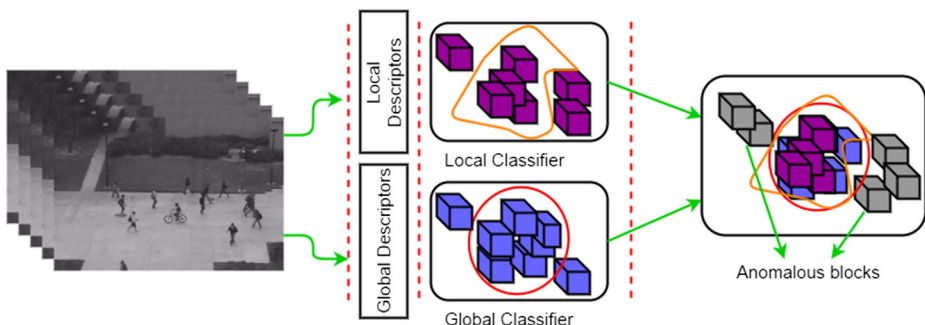


Fig. 3 Extracting video information

Algorithm 1 Local descriptor generation process

```

for Each frame in video do
  LocalDescriptor = [ ];
  Divide frame into 10 x 10 patches;
  for Each patch in frame do
    Compute SSIM value with surrounding 8 patches;
    LocalDescriptor.append(SSIM values);
  end
  for Each patch in frame do
    Compute SSIM value with the next 5 frames for the same patch;
    LocalDescriptor.append(SSIM values);
  end
end

```

The similarity between the patch and its neighbors is computed by using a image quality assessment tool, SSIM [29]. SSIM³ is a good measuring tool when it comes to similarity measurement between images in terms of the image structure. The SSIM index is a metric that predicts the measurement of the quality of the image based on a primary reference image which is distortion-free or uncompressed. SSIM, which is a perception based model, differs from other techniques namely the peak signal-to-noise ratio (PSNR) and the mean squared error (MSE) by not estimating absolute errors. It is perceptive due to the consideration of image degradation as a structural information change, and it also takes care of perceptual phenomena, such as contrast masking terms and luminescence masking. This structural information is based on the fact of spatially close pixels having strong inter-dependencies which contain information about the structure of the objects in the visual scene. Hence, SSIM fits perfectly into our application wherein we need structural information to distinguish between anomalous and normal frames.

Next, the SSIM index is applied on many image windows. The measure between two windows and common size $N \times N$ is given by:

$$SSIM(a, b) = \frac{(2\mu_a\mu_b + d_1)(2\sigma_{ab} + d_2)}{(\mu_a^2 + \mu_b^2 + d_1)(\sigma_a^2 + \sigma_b^2 + d_2)} \quad (1)$$

where, $[\mu_a]$ is average of a , $[\mu_b]$ is average of b , $[\sigma_a^2]$ is variance of a , $[\sigma_b^2]$ is variance of b , $[\sigma_{ab}]$ is covariance of a and b , $[d_1, d_2]$ are variables for weak denominator division stability and $j_1 = 0.01$, $j_2 = 0.03$ by default.

Figure 4 shows local feature assessment through the spatio-temporal neighboring which is the combination of the values, [d0 - d9;D0 - Dt1]. SSIM tool is used to compare two frames. To extend this to patches we compute the sum of the SSIM index for the corresponding frames in two patches. This is described in Fig. 5. Thus, the similarity between any two patches is the sum of the similarities between the corresponding frames.

3.3 Global descriptor generation

Global descriptors are features set used to represent the video holistically. Using HOG, HOF and SIFT features isn't suitable for anomaly detection as they are not discriminating enough.

³Structural Similarity Index Measurement. Online: <http://live.ece.utexas.edu/research/quality/SSIM/>

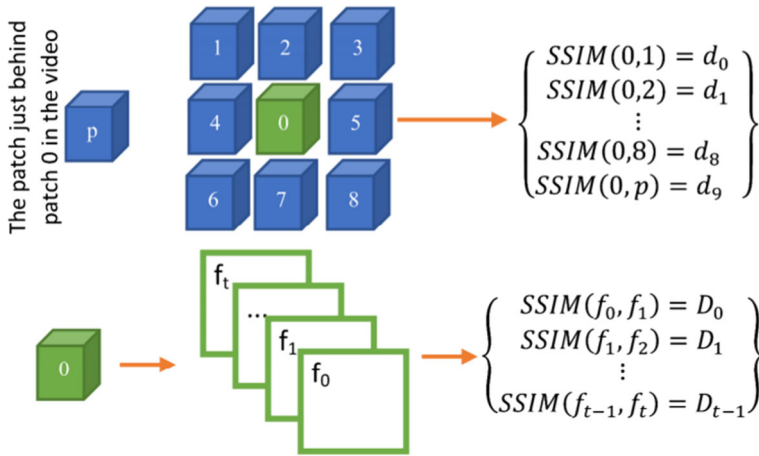


Fig. 4 Illustration of local descriptor generation [25]

Including several features can make the problem exceedingly complex and difficult to learn. To overcome this, we use autoencoders for unsupervised feature learning. Autoencoder neural networks [21] are unsupervised learning algorithms that apply gradient descent. Figure 6 depicts the architecture of a sparse auto encoder neural network.

In the proposed system, the autoencoder is used as a feature selection technique. Given an unlabeled set of training examples $\{x(1), x(2), x(3), \dots\}$ where $x^i \in R^n$, an autoencoder neural network will set the target values to be equal to the value of the inputs: $x^i = y^i$. The autoencoder learns a function $hW, b(z) \approx z$. By equating output to input, it learns an approximation to the identity function. In spite of the identity function being a trivial function, interesting underlying structure about the data can be found by incorporating network constraints which limit the number of hidden layer nodes and modifying average activation at each of these nodes.

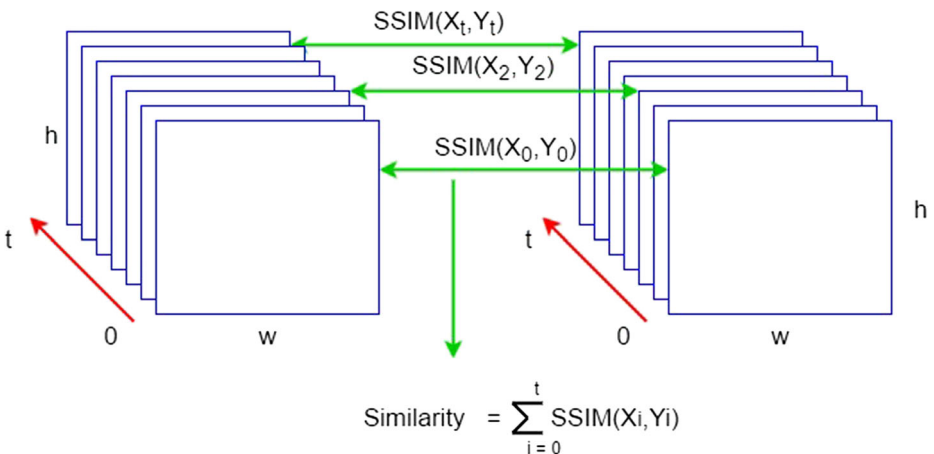
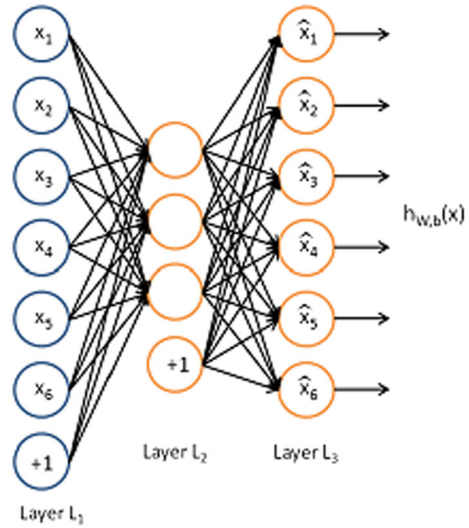


Fig. 5 Cubic patch comparison with SSIM tool

Fig. 6 Sparse autoencoder



Let the inputs x be the intensity values of pixels from a 10×10 image (100 pixels). So, $n = 100$ and there exist $s_2 = 50$ hidden units in L_2 . The hidden layer nodes are fewer than those in the input layer hence forced to learn a ‘compressed’ input representation, which means, trying and reconstructing the 100-pixel input from 50 hidden neurons. If a random input is provided, i.e., each x_i , independent of the other features is from a two-dimensional Gaussian function, then, the task of compression is difficult. However, if some of the input features are correlated and there is a structure in the input, the algorithm will be able to find some correlations. Just like in Principal Component Analysis, an autoencoder tries to learn data that is of a lower dimension.

The autoencoder, with the help of gradient descent, can learn sparse features. Suppose there exist n normal patches having dimensions (l, b, h) create a structure of $x_i \in R^D$, $D = l \times b \times h$. An autoencoder does the task of minimization of the defined objective in (2) by re-reconstructing the raw original data:

$$L = \frac{1}{m} \sum_{i=1}^m ||x_i - W_2 \delta (W_1 x_i + b_1) + b_2||^2 + \sum_{i=1}^{l.b.h} \sum_{j=1}^8 (W_{ji}^2) + \beta \sum_{j=1}^8 KL(\rho || \rho') \quad (2)$$

where,

- s - number of hidden layer nodes in autoencoder.
- $W_1 \in R^{s \times D}$ - weight matrix mapping the input to hidden layer nodes.
- $W_2 \in R^{D \times s}$ - weight matrix which maps nodes of hidden layer to nodes of output layer
- W^{ji} - weight between the i^{th} layer node and the j^{th} hidden layer node
- δ - Parametric ReLU activation function defined in (3).

$$\delta(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \quad (3)$$

Here α is a small constant, say 0.01.

- b_2 and b_1 - hidden layer and output layer bias.
- KL - Kullback-Leibler measures divergence between two Bernoulli random variables with mean ρ and mean ρ_j respectively. $KL(\rho \rho' j)$ - regularization function and it

enforces sparse hidden layer activation. KL is based on similarity between active node distribution and a Bernoulli distribution with ρ as parameter. It is a standard function for measuring the variability of two distributions.

- Parameter β is the penalty term’s weight used in the sparse autoencoder cost function.

In the above example, we consider the count of hidden units s_2 being less than the count of input units. However, interesting structure can be uncovered when the hidden unit number is more than input pixel numbers by imposing other network constraints such as sparsity constraint. The autoencoder will then find fascinating structure in the given data, despite the presence of a large number of hidden units. A neuron is considered to be ‘inactive’ if the output value obtained is approximately αx as a result of the Parametric ReLU function and ‘active’ if the output value is approximately x . Parametric ReLU avoids the problem of dying weights which is encountered while using sigmoid or ReLU activation functions. Most of the neurons must be inactive for a sparse autoencoder. This way it is easier to achieve the desired goal of reconstructing the output nodes from the input nodes.

Parameter (a_j^2) is the hidden unit j ’s activation in the autoencoder. But, this notation doesn’t clearly explain which input x led to activation. Therefore, we denote $(a_j^2)(x)$ as the hidden unit activation when a specific input x is given to the network. Equation (4) is the average activation of unit j over the training set.

$$\rho = \frac{1}{m} \sum_{i=1}^m [(a_j^2 x^i)] \tag{4}$$

The sparse autoencoder tries to enforce the constraint shown in (5), where, ρ is a sparsity parameter approximately zero (say $\rho = 0.05$). Each hidden neuron should have an average activation j approximately 0.05. The hidden unit’s activation must be near 0 in order to satisfy this constraint. In order to obtain this, the optimization objective is added to an extra penalizing term that $\hat{\rho}_j$ deviates significantly from ρ (given by (6)). In our work, the Kullback-Leibler divergence term is used as the penalty (7).

$$\hat{\rho}_j = \rho \tag{5}$$

$$\hat{\rho}_j = \sum_{j=1}^s \rho \log \frac{\rho}{\hat{\rho}_j} + (1 - \rho) \log \frac{1 - \rho}{1 - \hat{\rho}_j} \tag{6}$$

$$Penalty = \sum_{j=1}^s KL(\rho || \hat{\rho}_j) \tag{7}$$

In the proposed framework, the sparse autoencoder learns the optimal weights to represent the features of all the patches in the training video. Each input is a vector containing the intensities of all the pixels in the $10 \times 10 \times 5$ patch. As we read the frames in grey scale, the intensity is a single value. Therefore, each input node of the autoencoder is a vector containing 500 intensity values. The total number of input nodes is equal to the number of patches in the entire training dataset containing all the videos.

The count of the number of unites in the hidden layer indicate optimal feature count being used to represent a patch. When this exceeds the number of input nodes, we have a sparse autoencoder with sparsity greater than 1. The autoencoder tries to learn the right set of weights by making the input nodes equal to the output nodes. After each input node, the back propagation step of the neural network is performed to adjust the weights as per

the error at the output nodes. The training is continued until the threshold on the maximum number of iterations or the minimum cost (given by (8)) is attained.

$$J_{sparse}(W, b) = J(W, b) + \beta \sum_{j=1}^s KL(\rho || \hat{\rho}_j) \quad (8)$$

Algorithm 2 Global descriptor generation process

```

while iterations < maxIters do
  for Each frame in video do
    GlobalDescriptor = [ ];
    InputLayer = Intensity of each pixel in frame;
    HiddenLayer = Sigmoid(InputLayer * Weights1);
    OutputLayer = Sigmoid(HiddenLayer * Weights2);
    diff = OutputLayer - InputLayer;
    KLDivGrad = beta * (-(rho / rhoCap) + ((1 - rho) / (1 - rhoCap)));
    errOut = multiply(diff, multiply(outputLayer, 1 - outputLayer));
    errHid = multiply((transpose(W2), errOut) + transpose(matrix(KLDivGrad)),
    multiply(hiddenLayer, 1 - hiddenLayer));
    weight2Gradient = LearningRate*(errOut * HiddenNodes);
    Weight1Gradient = LearningRate*(errHid * InputNodes);
    Update weights of hidden and outer layers;
  end
end

```

3.4 Denoising sparse auto-encoder using dropout technique

A problem commonly encountered in any learning process is over fitting. The number of false-positives can be greatly reduced by incorporating techniques such as regularization and drop-out which counter overfitting. Denoising the auto encoder enhances the model to account for corruptness in input patterns and forces the hidden layer to learn more robust features. The network now learns an identity function of the corrupted version of the image. As defining normalcy under all circumstances given the limited training data is difficult, denoising the auto encoder proves to be an efficient solution. The image is distorted by setting a fixed percentage of the pixels to 0, i.e., dropping out some pixels. Figure 7 shows the architecture of a denoising auto encoder.

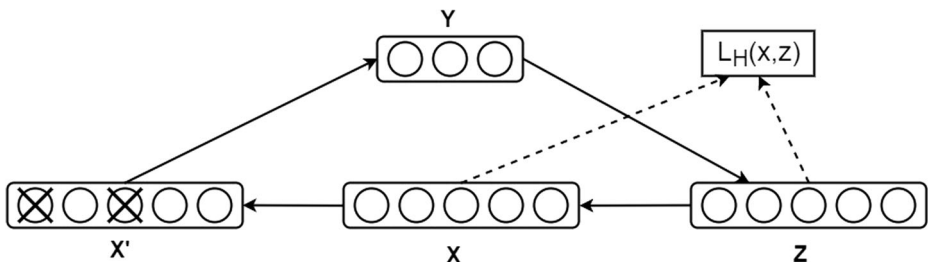


Fig. 7 Denoising autoencoder by dropping out pixels

3.5 Anomaly detection using gaussian classifiers

Gaussian classifiers are used to classify the local and global descriptors as anomalous or normal [17]. The local and global descriptors are constructed patch wise. Thus the job of the classifier is to distinguish the normal patches of the video from the anomalous patches. Gaussian Classifiers C_1 and C_2 use partially independent feature sets which are both global and local, to compute the Mahalanobis distance [11] which measures the distance of point X from a data distribution D , characterized by a mean and covariance matrix, and hence it is hypothesized as a Gaussian distribution that is multivariate and measures how far the point X is from the mean of the Gaussian distribution by taking into account all the dimensions of the dataset. If X is at the mean of D , then this distance is zero. This distance increases as X goes farther away from the mean and is computed by measuring the standard deviation along each principal component axis from X to the mean of D . Mahalanobis distance represents standard Euclidean distance in transformed space when each of these axes is re-scaled to have a variance of unity. It considers correlations of the dataset as it is scale-invariant and unitless [5]. The Mahalanobis distance for a data point is given by:

$$C_i(x) = \begin{cases} \text{Normal} & f(x) \leq \text{threshold} \\ \text{Anomaly} & \text{Otherwise} \end{cases} \tag{9}$$

where, Σ and μ are covariance and mean matrices respectively. For the classifier C_1 , we compute the Mahalanobis distance for each of the global descriptors - w_1x' . For C_2 , we give the local descriptors [d0...d9,D0..D4] as input to the distance function. If the distance exceeds a threshold value then, it specifies an abnormal patch (Fig. 8).

$$f(x) = (x - \mu)^T \Sigma^{-1} (x - \mu) \tag{10}$$

The threshold value greatly influences the performance of the system. Thus, choosing an optimal threshold value is essential to model an efficient classifier. This threshold value is got from the training patches. If both C_1 and C_2 classifiers classify a patch as anomalous,

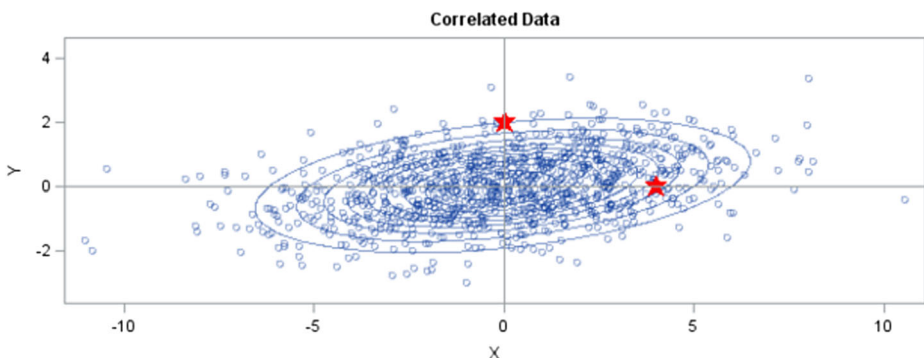


Fig. 8 Mahalanobis distance of two points for a distribution D

then it is an anomaly, but the patch is considered as normal if any one of them label it as normal. The following equation summarizes it mathematically.

$$F(x) = \begin{cases} \text{Anomaly} & \text{if } C_1 = C_2 = \text{Anomaly} \\ \text{Normal} & \text{Otherwise} \end{cases} \tag{11}$$

The above steps describe a model that helps identify anomalous video patches. The main challenge is to decide the patch size. Using large patches would result in low true positive rate and using small patches would result in high false positive rate. Furthermore, using large patches would increase the number of input dimensions to the autoencoder as the model has to be trained for a larger descriptors. Increasing the number of weights makes the program run slower and increases the complexity of the autoencoder, making it difficult to check for errors.

To overcome this problem, in the training phase we learn features from small patches of size $10 \times 10 \times 5$. During the testing phase, patches of size $40 \times 40 \times 5$ are considered. To bridge the dimension gap, the patches from the training phase are grouped into patches of size $40 \times 40 \times 5$ patches, without overlapping, and 16 feature vectors that have been extracted from the $16 - 10 \times 10 \times 5$ patches are combined using mean pooling technique, to obtain $40 \times 40 \times 5$ patch representation that can be verified by the learned classifier. Figure 9 shows this procedure.

4 Implementation specifics

4.1 Local feature extraction

During this process, 14 local features were extracted to form a local descriptor- 8 spatial and 6 temporal. Each frame was divided into patches of size 10×10 pixels giving 864 patches, thus 864 local descriptors (each descriptor containing 14 features) for each frame. This process took about 12–15 seconds to extract all the local descriptors from each frame. To do the same with one video which contains 120–150 key frames, the time taken was close to 45 minutes. For extracting local features from all the 16 training videos, the time taken was about 25–30 h.

4.2 Global feature extraction

The training videos were divided into $5 \times 10 \times 10$ cubic patches as learning from large patches is impractical. The intensity of these 500 pixels constituted a single input node. For all the 12 videos considered, we had 440640 input nodes. We assumed the hidden layer unit

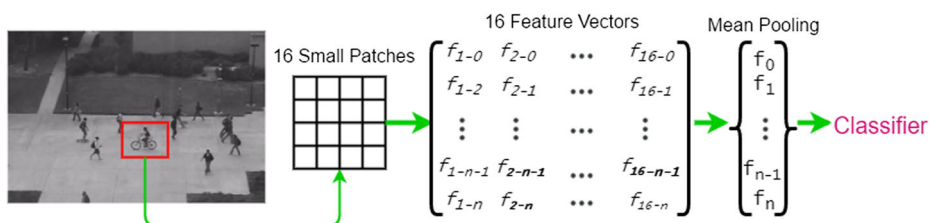


Fig. 9 Using feature learning for large patch anomaly detection

to consist of 100 nodes. This represents the number of sparse features we were trying to build. The sparse autoencoder details are as follows:

- *Input layer size:* 500
- *Hidden layer sizes:* We tested the autoencoder on the following sizes- 50, 100, 250, 400 and 600. Finally 100 was chosen as it gave the least cost.
- *Sparsity parameter of the hidden units (ρ):* The average activation of each node was chosen as 0.05 (standard value chosen for a sparse autoencoder).
- *Weight decay parameter (λ):* 0.0001
- *Sparsity penalty term (β):* 0.03, 0.05, 1, 1.5, 3, 6 of which 0.03 gave the least cost
- *Max iterations:* To prevent the problem of overfitting we reduced the number of iterations from 1000 to 400.
- *Learning Rate:* 0.5
- *Weights:* were chosen randomly using a Normal Distribution function. The weights connecting input to hidden nodes and hidden nodes to the output were both chosen using the same technique.
- *Bias:* Bias values at hidden nodes and output nodes are initialized to 0.
- *Error Threshold:* 0.05 was determined as the best threshold on performing several runs on the training data.

During preprocessing, each frame in the video was distorted by 25%. Pixels were chosen at random and their values were set to 0. The input nodes were then multiplied with the weights and the output at the hidden nodes was computed using the Parametric ReLU function. These outputs act as inputs for the second part of the forward pass connecting hidden layer to output layer. Error was calculated using the difference between the output and input layer, using which final set of optimal weights were obtained. These were then multiplied with the input nodes of all the videos. Since this is a computationally expensive process, we found the mean of 20 nodes at a time and compiled a new list of nodes. These input nodes were then multiplied with the weights connecting input layer to hidden layer. The 100×500 weight matrix multiplied with 500×22032 input nodes. This results in a global descriptor matrix of size 100×22032 . The Mahalanobis classifier was applied on this to compute the optimal threshold. The threshold was obtained as 192 on choosing the above parameters.

With the above experimental setup, the global feature extraction took about 30–40 seconds to extract all the local descriptors from each frame. To do the same with one video which contains 120–180 key frames, the time taken was close to 120 minutes. For the extraction of local features from all the 16 training videos, the time taken was about 30–32 h.

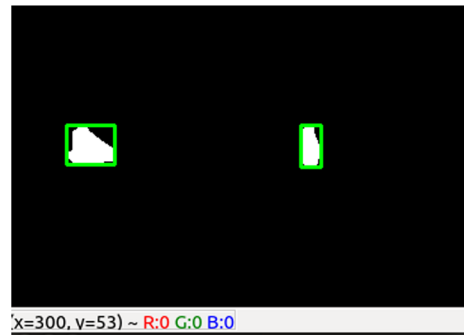
4.3 Anomaly classification

The Gaussian classifier was used separately for local and global features obtained during the process described earlier. For the Gaussian Local Classifier, the Mahalanobis distance of

Table 1 Parameters used in anomaly classification using Gaussian Classifier

Classifier	Feature set	Avg. dist	Max. dist	Threshold
Gaussian local	Local features	14.2324	78.33432	15
Gaussian global	Global features	45.4564	148.65481	45

Fig. 10 Ground Truth with bounding box



all videos in the training set, which contains all normal patches was calculated first. Mean of all these distances was used as threshold during classification. The values of the other parameters is indicated in Table 1. A low threshold was chosen to decrease the false negative rate, at the expense of false positive rate. This was motivated by the fact that we can afford to detect a normal event as anomaly, but we cannot afford to miss an anomalous event by reporting it as normal. For the Gaussian Global Classifier, the Mahalanobis distances for the training patches were computed (given in Table 1). The threshold value chosen was again motivated by the same reason as mentioned for the local classifier.

5 Experimental results

For local testing, local feature descriptors from the test videos are extracted in the same manner as done for the training set. Each descriptor corresponding to a spatio-temporal patch is classified using the Gaussian classifier. The results of each video, in the form of list of detected anomalous patches in each frame, are written to a separate file. To evaluate the results, we extracted the *ground truth* from the testing videos. The ground truth videos had white patches against black background. The position of the white patch represented the ground truth anomaly in the video. We found the contours along the anomaly, and developed a bounding box around the contour as shown in Fig. 10. The co-ordinates of this bounding box were written to a separate file, to be used while evaluating the results.

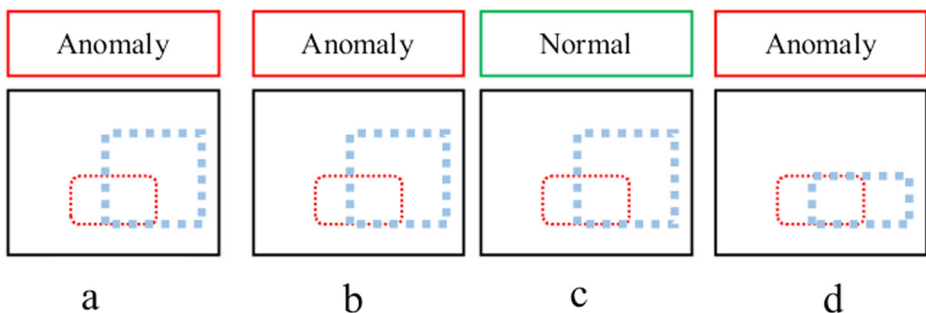


Fig. 11 Levels of evaluation - **a** Frame **b** Pixel **c** Dual pixel **d** Dual pixel



Fig. 12 Anomalous activity identified by Local classifier

Evaluation The results of the global classifier and the local classifier were evaluated separately. Finally, the results of both the classifiers were combined and evaluated. A logical AND was applied to the results of the two classifiers to combine their results, i.e., a patch is considered as an anomaly if it is detected as an anomaly by both the classifiers. The files containing the anomaly patch numbers detected from the local and global classifiers, along with those containing the ground truth values were also read. Three levels of evaluation were performed (illustrated in Fig. 11)

- *Frame Level:* Here, the entire frame was classified as anomalous if an anomaly was detected in at least one pixel in that frame.
- *Pixel Level:* The entire frame is marked anomalous if the detected anomaly is covering at least 40 percent of the pixels in ground truth anomaly.



Fig. 13 Anomalous activity identified by Global classifier



Fig. 14 Anomalous activity identified after combining Local and Global classifier results

- *Dual Pixel Level*: A frame is determined an anomaly if the anomaly condition at pixel level is satisfied and the ground truth pixels cover at least ‘B’, which in our case is 15% of the detected anomaly pixels.

Figures 12 and 13 show an example of the anomaly being detected using local and global classifiers respectively. The results from both the classifiers were combined using the dual pixel level technique and the result of an anomaly being detected is shown in Fig. 14.

For experimental evaluation, the UCSD Pedestrian Dataset and UMN Dataset were used. The UCSD dataset was created with the aid of an elevated camera in order to capture movement of pedestrians and contains 2 subsets. Only the UCSD Peds2 dataset containing 16 training video samples and 12 testing video samples of pedestrians walking in the view-plane of the camera was used to evaluate our algorithm. The UMN dataset contains videos shot in three different environments. Each video shows a group of people initially walking normally, who then run all of a sudden, which is the anomalous activity.

Table 2 Frame and pixel level comparison of EER

Algorithm	Frame-Level(%)	Pixel-Level (%)
Social Force [19]	42	79
MPCCA [12]	30	82
MPCCA+SF [18]	36	72
[1]	42	76
MDT [18]	25	55
[31]	20	42
[16]	18.5	29.9
[25]	19	24
[32]	17	–
Proposed method	16	20

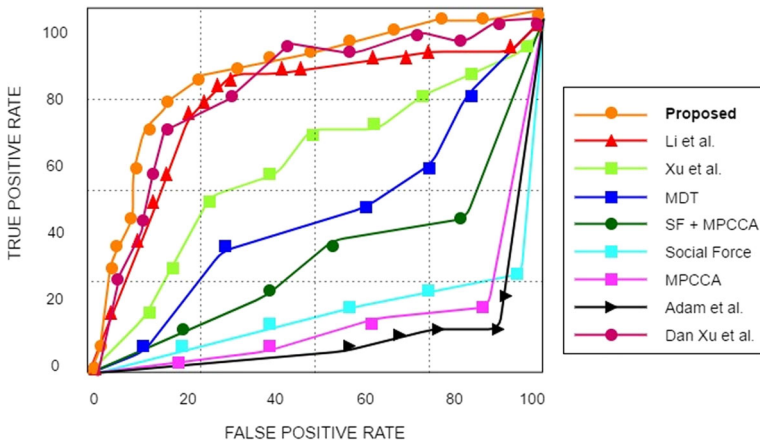


Fig. 15 ROC Curves for pixel level evaluation on UCSD Ped2 Dataset (Existing approaches vs. Proposed Method)

Testing For global testing, the test data is divided into patches of size $40 \times 40 \times 5$. Mean pooling is performed based on the weights obtained from the training samples of size $10 \times 10 \times 5$. The test input nodes are then multiplied with these weights to obtain the global descriptors. On this, the Mahalanobis distance metric is applied to determine if the patch is an anomaly or not. The mean and co-variance for Mahalanobis classifier are the same as the training set.

Table 2 shows the Equal Error Rate (EER) values obtained for different algorithms when applied to the UCSD Pedestrian 2 dataset. The lower the EER value the better. As is evident from the tabulated data, the pixel-level EER of the proposed algorithm is the lowest among the state-of-the-art algorithms considered for comparison and is at par with [32]’s method.

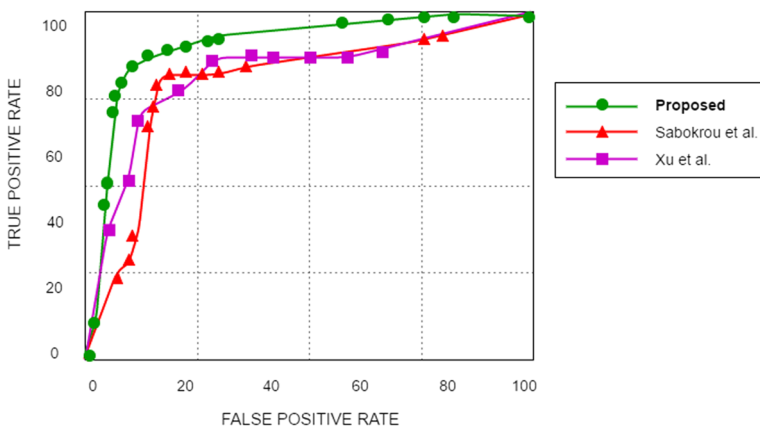


Fig. 16 ROC Curve for frame level evaluation on UCSD Ped2 Dataset (Existing approaches vs. Proposed Method)

Table 3 Comparative performance measured in AUC and EER

Algorithm	AUC	EER
Chaotic invariant [30]	99.4	5.3
Social force [19]	94.9	12.6
Sparse [7]	99.6	2.8
[26]	99.5	3.4
[16]	99.5	3.7
[25]	99.6	2.5
[32]	90.8	2.4
Proposed method	99.6	2.2

However, the frame-level EER outperforms all the existing algorithms. While our proposed method achieves a frame-level EER of 16%, [32]’s method attained the second best result of 17%. In summary, we found that our method was better by a margin of 1% than the previous best result. The dropout technique used in the sparse autoencoder helped reduce the rate of false positives (Figs. 15 and 16).

Figure 15 depicts the ROC curve for pixel level measure. The proposed method was compared against 7 other existing algorithms on the basis of pixel level anomaly detection performance. Again, it was found that the proposed method performed the best in this case too. The lowest frame level EER of 17 was obtained for proposed method. Figure 16 shows the ROC curve for frame level evaluation of our algorithm. The rate of false positives was also lower than any other method as can be interpreted from the curve.

The Area under ROC Curve (AUC) and EER values for the UMN Dataset are shown in Table 3. This dataset has no pixel-level ground truth and there are only three anomalous scenes. Moreover, the spatio-temporal changes between normal and abnormal frames are very high. Considering these limitations, we used the EER and AUC at frame-level to evaluate the proposed model. From the tabulated results, it is clear that our algorithm obtained the best EER value of 2.2% which is 0.2% better than the current best achieved by Xu et al. [32]. The AUC value (99.6) is at par with the previous best value obtained by Sabokrou et al. [25].

From Table 4, it can be observed that our method also achieved the least running time when compared to all current state-of-the-art methods. Xu et al. [32]’s algorithm is presently the fastest and takes 7.5 seconds per frame while testing on the UCSD Ped-2 dataset. Our

Table 4 Comparison of different methods in terms of computational time during test (seconds per frame)

Method	UCSD Ped 2
[32] without foreground detection	13.5
[32] with foreground detection	7.5
Proposed method	4.3

algorithm takes 4.3 seconds to test on the same dataset and is faster by a margin of 3.2 seconds, thus resulting in a $1.7\times$ speedup.

6 Conclusion and future work

In this paper, a dynamic anomaly detection system for crowded scene videos using sparse autoencoders was proposed. The incoming video stream was divided into voxels of size $10 \times 10 \times 5$ from which the local and global feature descriptors are captured. The local descriptors help learn the spatial and temporal relations that aid in localizing anomalies while the global descriptors constructed using sparse denoising autoencoders are used for interpreting the video as a whole. Gaussian classifiers were modeled using the descriptors and the minimum threshold to define abnormality was learned using the Mahalanobis distance metric. Further, a dual pixel measure was employed to evaluate anomalies detected using our method. Experimental validation clearly showed that the frame level anomaly detection rate achieved by the proposed method on the UCSD dataset was the best result achieved yet when compared to several existing methods. Similarly, for the UMN Dataset, an AUC value of 99.6 was obtained, which is at par with the current best. The time taken by our algorithm (4.3 seconds per frame) is the least compared to existing works. Also, an improved EER of 2.2 (0.2% improvement) was observed, which makes the proposed method the most efficient when compared to state-of-the-art methods. This implies that the proposed method can be applied on other crowded scene videos to detect anomalies in less time and with a high accuracy.

For further improving our approach, we plan to extend our framework and employ it over parallel environments such as SPARK, so that video inputs of larger sizes can be handled and the model can be trained faster. Likewise, using tools like Caffe and Tensor Flow which offer GPU support is another possibility for further optimization. With reference to the Gaussian Classifier, we intend to evaluate the performance of statistical distance measures in place of the Mahalanobis distance used currently.

References

1. Adam A, Rivlin E, Shimshoni I, Reinitz D (2008) Robust real-time unusual event detection using multiple fixed-location monitors. *IEEE Trans Pattern Anal Mach Intell* 30(3):555–560
2. Aljawarneh S, Aldwairi M, Yassein MB (2017) Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*. Elsevier
3. Aljawarneh SA, Vangipuram R, Puligadda VK, Vinjamuri J (2017) G-SPAMINE: An approach to discover temporal association patterns and trends in internet of things. *Future Generation Computer Systems*. Elsevier
4. Antić B, Ommer B (2011) Video parsing for abnormality detection. In: 2011 international conference on computer vision. IEEE, pp 2415–2422
5. Bertini M, Del Bimbo A, Seidenari L (2012) Multi-scale and real-time non-parametric approach for anomaly detection and localization. *Comput Vis Image Underst* 116(3):320–329
6. Cheng KW, Chen YT, Fang WH (2015) Video anomaly detection and localization using hierarchical feature representation and gaussian process regression. In: The IEEE conference on computer vision and pattern recognition (CVPR)
7. Cong Y, Yuan J, Liu J (2011) Sparse reconstruction cost for abnormal event detection. In: 2011 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 3449–3456

8. Goldberger J, Gordon S, Greenspan H (2003) An efficient image similarity measure based on approximations of kl-divergence between two gaussian mixtures. In: 2003. Proceedings. Ninth IEEE international conference on computer vision. IEEE, pp 487–493
9. He K, Zhang X, Ren S, Sun J (2015) Delving deep into rectifiers: surpassing human-level performance on imagenet classification. In: Proceedings of the IEEE international conference on computer vision, pp 1026–1034
10. Jiang F, Yuan J, Tsaftaris SA, Katsaggelos AK (2011) Anomalous video event detection using spatiotemporal context. *Comput Vis Image Underst* 115(3):323–333
11. Joseph E, Galeano P, Lillo RE (2013) The mahalanobis distance for functional data with applications to classification. arXiv:13044786
12. Kim J, Grauman K (2009) Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental upyears. In: IEEE conference on computer vision and pattern recognition, 2009. CVPR 2009. IEEE, pp 2921–2928
13. Kong D, Gray D, Tao H (2005) Counting pedestrians in crowds using viewpoint invariant training. In: BMVC, Citeseer
14. Li C, Han Z, Ye Q, Jiao J (2013) Visual abnormal behavior detection based on trajectory sparse reconstruction analysis. *Neurocomputing* 119:94–100. doi:10.1016/j.neucom.2012.03.040. <http://www.sciencedirect.com/science/article/pii/S0925231213000179>, intelligent Processing Techniques for Semantic-based Image and Video Retrieval
15. Li N, Wu X, Xu D, Guo H, Feng W (2015) Spatio-temporal context analysis within video volumes for anomalous-event detection and localization. *Neurocomputing* 155:309–319
16. Li W, Mahadevan V, Vasconcelos N (2014) Anomaly detection and localization in crowded scenes. *IEEE Trans Pattern Anal Mach Intell* 36(1):18–32
17. Lippmann R (1987) An introduction to computing with neural nets. *IEEE Assp magazine* 4(2):4–22
18. Mahadevan V, Li W, Bhalodia V, Vasconcelos N (2010) Anomaly detection in crowded scenes. In: CVPR, vol 249, p 250
19. Mehran R, Oyama A, Shah M (2009) Abnormal crowd behavior detection using social force model. In: IEEE conference on computer vision and pattern recognition, 2009. CVPR 2009. IEEE, pp 935–942
20. Mo X, Monga V, Bala R, Fan Z (2014) Adaptive sparse representations for video anomaly detection. *IEEE Trans Circuits Syst Video Technol* 24(4):631–645
21. Ng A (2011) Sparse autoencoder. CS294A lecture notes 72:1–19
22. Radhakrishna V, Aljawarneh SA, Kumar P, Janaki V (2017) A novel fuzzy similarity measure and prevalence estimation approach for similarity profiled temporal association pattern mining. *Future Generation Computer Systems*. Elsevier
23. Reddy V, Sanderson C, Lovell BC (2011) Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture. In: CVPR 2011 WORKSHOPS. IEEE, pp 55–61
24. Roshkhari MJ, Levine MD (2013) Online dominant and anomalous behavior detection in videos. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 2611–2618
25. Sabokrou M, Fathy M, Hoseini M, Klette R (2015) Real-time anomaly detection and localization in crowded scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition workshops, pp 56–62
26. Saligrama V, Chen Z (2012) Video anomaly detection based on local statistical aggregates. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 2112–2119
27. Stauffer C, Grimson WEL (2000) Learning patterns of activity using real-time tracking. *IEEE Trans Pattern Anal Mach Intell* 22(8):747–757
28. Vincent P, Larochelle H, Bengio Y, Manzagol PA (2008) Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th international conference on machine learning, ACM, New York, NY, USA, ICML '08, pp 1096–1103. doi:10.1145/1390156.1390294
29. Wang Z, Bovik A, Sheikh HR (2004) Image quality assessment from error measurement to structural similarity. *IEEE Trans Image Process* 13(4):600–612
30. Wu S, Moore BE, Shah M (2010) Chaotic invariants of lagrangian particle trajectories for anomaly detection in crowded scenes. *IEEE*
31. Xu D, Song R, Wu X, Li N, Feng W, Qian H (2014) Video anomaly detection based on a hierarchical activity discovery within spatio-temporal contexts. *Neurocomputing* 143:144–152
32. Xu D, Yan Y, Ricci E, Sebe N (2017) Detecting anomalous events in videos by learning deep representations of appearance and motion. *Comput Vis Image Underst* 156:117–127
33. Zhang Y, Lu H, Zhang L, Ruan X, Sakai S (2016) Video anomaly detection based on locality sensitive hashing filters. *Pattern Recogn* 59:302–311. doi:10.1016/j.patcog.2015.11.018



Medhini G. Narasimhan was born in India on February 14th, 1995. She is pursuing her B.Tech. in Information Technology at the National Institute of Technology Karnataka, India and will graduate in the year 2017. Her primary research interests are Deep Learning, Machine Learning, Computer Vision and Computational Biology. In the past, she has worked on designing automated systems for semantic content retrieval from videos, video anomaly detection and sentential description of videos. Her work on “Predicting Symptom Onset and Contagiousness of Respiratory Viral Infections”, was presented as a poster at the 12th Intelligent Systems for Computational Biology Conference and won the Best Poster Award.



Sowmya Kamath S. is an Assistant Professor in the Department of Information Technology at National Institute of Technology Karnataka, Surathkal, where she has been a faculty member since 2009. She earned her Ph.D. from NITK Surathkal and undertook her graduate & undergraduate studies at Manipal University, India. Her research interests lie in the areas of machine learning, natural language processing, big data analytics and semantics in information retrieval. She has actively collaborated with researchers in her areas of interest and is currently involved in several ongoing research projects with reputed universities like CMU-USA, INRIA-France and IIT Bombay.