Adaptive clustering in latent space for anomaly detection in videos using synthetic data

Seby Jacob, Martin D. Levine
Center for Intelligent Machines
McGill University
Montreal, Canada
{sejacob,levine}@cim.mcgill.ca

***Abstract*—We introduce a new framework for generating an arbitrarily large synthetic dataset for anomaly detection in videos, enabling the training of complex deep-learning models for the task. We also present a novel method for anomaly detection in videos using adaptive clustering in the latent encoding space of a convolutional autoencoder. Our algorithm produces results competitive with the state of the art using actual surveillance videos in benchmark datasets. Compared to other solutions, our algorithm has a very high true-positive to false-positive rate at lower thresholds. Furthermore, we experimentally analyze the relationship between dataset size and model accuracy to reveal that the classification error asymptotically tends towards zero with infinite training data.**

***Keywords*-Anomaly; Surveillance; Deep-Learning; Autoencoder;**

I. Introduction

The drop in prices for security cameras and hard-disk drives, coupled with increasing security concerns has fueled a boom in the security and surveillance industry. Surveillance cameras are widely used as a security measure to safeguard life and property. However, most of these cameras are not subject to active live monitoring. Even if they were, active live monitoring is a huge waste of human effort as noteworthy events seldom happen. Usually, the recorded data are used only as hindsight after a crime or an unfortunate event has occurred. Consequently, there is a need for reliable automatic video surveillance algorithms that can then forward only the events of interest to a human arbiter.

Events in a surveillance video can be categorized into:

- Normal events
  These are events characteristic of the scene that are unworthy of intervention or any action from humans. Consequently, these are the most frequent events, or the only type of events that occur in the scene.
- Abnormal or anomalous events
  These events are uncharacteristic of the scene. They are unusual, irregular or suspicious activities that are outstanding and worthy of attention. These events do not occur frequently (or they never occur), and are hence referred to as anomalous events.

Thus, an anomalous event can be implicitly defined as an event with an extremely low probability of occurrence, when compared to events previously observed in the scene. Detection of an anomalous event includes two goals, namely identifying an event as anomalous and localizing the manifestation of the identified event within the monitored space.

The key contributions of this work are:

1) Creation of an artificial variable length dataset with ground truth annotations for the training of deep-learning models for anomaly detection in videos.
2) A novel method for anomaly detection in videos using adaptive clustering in the latent space.

We demonstrate our method to be competitive compared to state of the art methods for anomaly detection in videos. The importance of having large datasets for enabling the use of deep-learning models for the task is also discussed. Section II provides a brief overview on recent related works. The datasets are presented in Section III and the proposed method is detailed in Section IV. Section V specifies the experiments conducted and details the results while Section VI provides a conclusion.

II. Related Work

A. Deep-learning for anomaly detection

Many works have recently employed deep-learning for anomaly detection. Sabokrou *et al.* [1] and Xu *et al.* [2] learn representative features for normal events using autoencoders [3]. These features are used to learn a one-class classifier to detect anomalies as outliers from the normal class. Extending their work further, in [4], Sabokrou *et al.* use a pre-trained Convolutional Neural Network (CNN) to generate feature maps of the surveillance frames that are then used to train a sparse autoencoder. The sparse autoencoder works in cascade with a Gaussian classifier to determine the anomaly score of each image patch in a test frame.

Fan *et al.* [5] train two Convolutional Variational Autoencoders (CVAE) for anomaly detection in videos. The first CVAE is trained on the raw image frames whereas the second is trained on the dynamic optical flow images [6]. A separate Gaussian Mixture Model (GMM) is fitted in the latent space encodings from each autoencoder to model the normal behaviour of the scene. The membership scores from both GMMs are then combined to determine whether a spatio-temporal test volume is an anomaly. In [7], a 3-D Convolutional Autoencoder (CAE) is trained on frame sequences. The reconstruction error of each test sequence is then evaluated to assess a regularity score.

B. *Synthetic datasets for deep networks*

The recent success of deep-learning algorithms can be partly attributed to the availability of very large labeled datasets. These make it possible to train complex deep neural networks with reduced risk of overfitting. Recent results ([8], [9]) show that the reliability of current models are not only limited by the algorithms themselves, but also by the quality and size of supervised data available. Hence, for very challenging tasks, progress towards collecting large high-quality supervised datasets is inevitable. However, collecting and labeling a large visual dataset could prove to be difficult. In this paper, we show that synthetic data is an excellent alternative.

Gaidon *et al.* [10] provides an interesting overview on the effectiveness of synthetic visual data. In [11], Müller *et al.* uses a simulator built on the Unreal Engine to create realistic virtual worlds. They provide a customizable simulation environment and validate it on two applications, UAV-based tracking of moving objects and autonomous driving. Jiang *et al.* [12] propose a complete learning-based pipeline to generate massive quantities of complex 3D synthetic indoor scenes. They validate the dataset based on the prediction of depth and surface normal, semantic segmentation and reconstruction. The authors of [13] adds synthetic fog in real-world images of clear-weather scenes to treat the problem of foggy scene understanding. Rogez and Schmid consider the problem of 3D pose estimation from a 2D image in [14]. The authors combine 3D motion capture data with real world images annotated with 2D poses to generate labeled data.

III. Datasets

A. *Artificial dataset*

A single surveillance camera inherently simultaneously records video frames of a scene containing two aspects, namely, the static and the dynamic components. The static component are the pixels in the scene that do not change while the dynamic component are the changes in pixel values in successive frames. The synthetic data we generate are founded on these facts.

The program initializes a background simulating the view of a scene overlooking a country road as shown in Fig. 1. This is the static background of the scene. For convenience, we make the assumption that the background data are solely static and do not contain any moving objects. Simultaneously, the foreground data are composed of randomly coloured shapes (circles and squares) moving along the two fixed lanes of the road in queues. The inter-shape distances in each queue is random. Frames are generated sequentially, moving the shapes one pixel at a time. This simulates a high sampling rate from the imaginary camera. The length of the training dataset can be determined by the user. This provides great flexibility and a source of a large amount of training data[1].

1) Training data: The training data consists of randomly colored squares and circles moving in two queues along the road. The squares are moving left to right in the top half of the image and the circles are moving right to left in the bottom half of the image. Some images from the training dataset can be observed in Fig. 2.

2) Test data: The test data mostly consists of randomly colored squares and circles moving along the road. The queues are interspersed with randomly colored triangles with a very low probability that can be set by the user. The presence of the triangles are the anomalies that are to be detected since the training data contains only squares and circles. The pixel level ground truth for the anomalies are generated as bitmaps. Some images from the test dataset are shown in Fig. 3. As seen, the pixels occupied by triangles in the frames are marked out as anomalies in the ground truth annotations. The task for any algorithm is to accurately detect all of the triangles from the test set as anomalies.

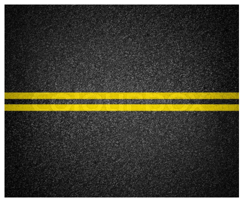
Figure 1: Background for the artificial dataset.

B. *Evaluation dataset*

We make use of the University of California San Diego (UCSD) pedestrian dataset [15] for evaluating our solution against the state of the art techniques in anomaly detection. The dataset contains videos overlooking walkways. The videos were captured from

[1]The code for generating the artificial dataset is available at: https://github.com/BitFloyd/SyntheticAnomalyDataset.

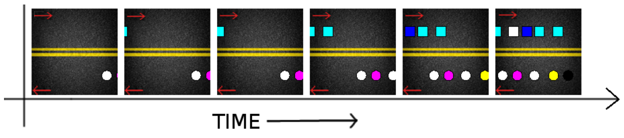

Figure 2: Examples of training set images ordered in a timeline. The red arrows in the figures are representative of the direction in which the shapes move on the road.

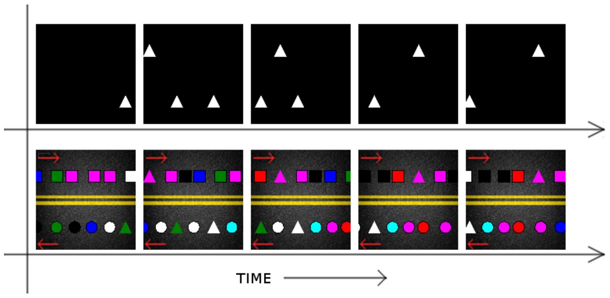

Figure 3: Images from the test set of the artificial dataset. Top row: Pixel level ground truth annotations for the corresponding test set images. Anomalous pixels are white in color. Bottom row: Examples of test set images ordered in a timeline.

stationary cameras at two locations. The normal events include pedestrians on walkways. Anomalies in the dataset are non-pedestrian entities and abnormal pedestrian patterns. Some examples of non-pedestrian entities are bikers, skaters, people on wheelchairs, small carts and people walking on the grass.

IV. Method

A. Data preprocessing

Raw video frames are high dimensional data vectors with a tremendous amount of spatial and temporal redundancy. In order to efficiently process these data, we need to reduce the redundancy and encode the video into a low dimensional space. Our goal is to detect anomalous events and localize them in the spatial extent of the frame. For our purpose, we can sub-sample the video both spatially and temporally. Such cubic video patches are called spatio-temporal cuboids. These spatio-temporal cuboids are assembled into a dataset for training our model.

We can assume that the anomalous events will be in the dynamic component of the scene. Hence, it is advantageous to make our model focus on the dynamics of the scene by doing a background subtraction in all of the video frames. The full data preparation algorithm is as follows:

1) Convert video file into sequential frames.
2) Evaluate background frame of video by averaging all of the frames.
3) Subtract the background frame from all the frames.
4) Normalize the collection of frames by re-scaling all pixels to lie in the range [0,1].
5) Densely sub-sample the frames into units spanning P frames each.
6) Spatially sub-sample each unit into spatio-temporal cuboids using a sampling window of dimensions: (S_1, S_2) and stride S_t.
7) Compile all the spatio-temporal cuboids into a dataset to be used for training.

P varies from dataset to dataset. Each unit must have length proportional to the average event duration in the dataset. For example, in a dataset with walking pedestrians as the key dynamic component, P must be large enough to contain one walking step of a person (on average).

B. The model

In our solution, the collected dataset is used to train a deep convolutional autoencoder.

1) Encoder: The encoder, given by $E_\gamma(X) = Y : X \in \mathcal{R}^N \rightarrow Y \in \mathcal{R}^n \mid N > n$, is a dimensionality reduction transformation of the input. The encoder receives a spatio-temporal cuboid of dimensions : (cuboid-height, cuboid-width, $P\times$channels) as input and transforms it into a vector of n features.

The encoder architecture for a spatio-temporal cuboid input of dimensions: (24,24,8$\times$3) is given in Table I. The encoder reduces the dimensionality of the spatio-temporal cuboid and encodes each observation into a 32 dimensional vector. G is a zero-centered Gaussian distribution that adds small perturbations to the input. This makes sure that the encoder cannot simply memorize the best transformation for each input. The model is forced to learn the structural and characteristic features of the input to perform the transformation [3]. LR is LeakyReLU activation [16].

2) Decoder: The decoder is a function of the form $D_\phi(Y) = X^{'} : Y \in \mathcal{R}^n \rightarrow X^{'} \in \mathcal{R}^N \mid N > n$. It essentially operates in reverse of the encoder. The decoder receives a vector of n features as input that is transformed back into a spatio-temporal cuboid of dimensions: (cuboid-height, cuboid-width, $P\times$channels) as output.

The decoder architecture corresponding to the encoder (Table I) is given in Table II.

3) The convolutional autoencoder: The autoencoder is a function of the form $A_{\gamma,\phi}(X) = X^{'} : X \in \mathcal{R}^N \rightarrow X^{'} \in \mathcal{R}^N$. The convolutional autoencoder is the encoder and decoder in series. The input and output of the autoencoder are spatio-temporal cuboids. The parameters of the autoencoder (γ and ϕ) are optimized to reconstruct the input at the output. By making the

Layer	Noise	Parameters	Activation
Input (24,24,24)			
2D Convolution	G(σ=0.05)	filters:64	LR (α=0.3)
Spatial dropout		rate:0.3	
Batch norm			
2D Average pooling		pool:(2,2)	
2D Convolution	G(σ=0.03)	filters:128	LR (α=0.3)
Spatial dropout		rate:0.3	
Batch norm			
2D Average pooling		pool:(2,2)	
2D Convolution	G(σ=0.02)	filters:256	LR (α=0.3)
Spatial dropout		rate:0.3	
Batch norm			
2D Average pooling		pool:(2,2)	
Fully Connected		units:32	LR (α=0.3)
Dropout		rate:0.3	
Batch norm			

Table I: Encoder architecture.

Layer	Parameters	Activation
Input (32)		
Fully Connected	neurons:4608	LR (α=0.3)
Reshape	shape:(3,3,512)	
2D Upsampling	upsample:(2,2)	
2D Convolution	filters:256	LR (α=0.3)
Batch norm		
2D Upsampling	upsample:(2,2)	
2D Convolution	filters:128	LR (α=0.3)
Batch norm		
2D Upsampling	upsample:(2,2)	
2D Convolution	filters:64	LR (α=0.3)
Batch norm		
2D Convolution	filters:32	LR (α=0.3)
2D Convolution	filters:24	sigmoid

Table II: Decoder architecture.

autoencoder reconstruct the input at the output as best as possible, it makes sure that the encodings are low dimensional representations of the input cuboids. The noise layers and dropout layers are present only in the encoder. This is to make sure that the decoder can always reliably reconstruct the encodings while training.

C. Training algorithm

The data used for training do not contain any anomalous events. Also, as stated earlier, they have a low probability of occurrence. If we can fit a probability distribution (G) to the encodings of the training set, from the encoding $E_\gamma(x')$ of each test set sample x', we may evaluate $P(E_\gamma(x') \in G)$. Since our model might be trained on complex scenes with a variety of behaviours, the encoding space (latent space) could be composed of multiple distributions. Hence our model must encode all samples in a space that is favourable for clustering and fitting multiple distributions.

In general, the K-Means algorithm [17] works very well when the data samples are spherically scattered (same variance in all dimensions) around the centroids in the multivariate feature space. These types of datasets are called K-Means friendly datasets. High dimensional multivariate data are generally not K-Means friendly. Since we are using a convolutional autoencoder as a nonlinear transformation to obtain encodings of spatio-temporal cuboids in a multivariate feature space, we need to make sure that this transformation produces an approximately K-Means friendly latent space. To ensure this, we perform the clustering in parallel to the training of the convolutional autoencoder, using the method suggested in [18]. During training, a K-Means algorithm is performed to initialize a set of K centroids = $\boldsymbol{M}$ on the encodings $\boldsymbol{E}_\gamma(X)$. These $\boldsymbol{M}$ centroids are then updated and used in the training objective function to guarantee that the multivariate feature space will favour clustering. To guarantee this, the convolutional autoencoder is to be trained with the following optimization objective in Eq:1.

$$\min_{\gamma,\phi,M} \sum_{i=1}^{N} \left(\ell\left(\boldsymbol{D}_\phi(\boldsymbol{E}_\gamma(\boldsymbol{x}_i)), \boldsymbol{x}_i\right) + \frac{\lambda}{2} \left\| \boldsymbol{E}(\boldsymbol{x}_i) - \boldsymbol{M}\boldsymbol{s}_i \right\|_2^2 \right) \tag{1}$$

$$\text{s.t.} \quad s_{j,i} \in \{0,1\}, \ \mathbf{1}^T s_i = 1 \ \ \forall i,j,$$

$$\boldsymbol{x} = \text{Spatio-temporal cuboid} \tag{2}$$

$$\ell(\cdot) : \mathbb{R}^M \to \mathbb{R} = \text{reconstruction loss} \tag{3}$$

$$\boldsymbol{D}_\phi(\cdot) = \text{Decoder} \tag{4}$$

$$\boldsymbol{E}_\gamma(\cdot) = \text{Encoder} \tag{5}$$

$$\lambda = \text{Clustering loss regularization} \tag{6}$$

$$\boldsymbol{M} = \text{Means} \tag{7}$$

$$\boldsymbol{s} = \text{mean membership} \tag{8}$$

$$\tag{9}$$

The reconstruction loss $\ell(\cdot) : \mathbb{R}^M \to \mathbb{R}$ used in this work is the structural dissimilarity loss as described in [19] and [20].

The constraints and the cost function of Eq:1 are both non-convex. A stochastic gradient descent algorithm using backpropagation cannot be used to jointly optimize $\gamma, \phi, \boldsymbol{M}$ and $\{\boldsymbol{s}\}$ since the mean memberships variable $\{\boldsymbol{s}\}$ is constrained to be discrete, making the joint-objective non-differentiable. We optimize Eq:1 exactly as in [18]. In summary, we alternate optimizing Eq:1 with respect to one of $\boldsymbol{M}$, $\{\boldsymbol{s}\}$ and γ, ϕ, while keeping the other two sets of variables fixed.

D. Feature space analysis

Once the training procedure is finished, we create a set of encodings of the training samples as given by the encoder. These features are $F = E_\gamma(X)$, where X is the set of spatio-temporal cuboids obtained from processing the training set.

Since there are no anomalies in the training videos, F is the collection of normal features. This set of features are now modelled as a multivariate distribution G. As we initialized K means for clustering the encodings in the feature space, we fit a multivariate Gaussian mixture distribution model G [21] on F, with K Gaussians using expectation maximization [22].

E. Evaluation and Anomaly Detection

To evaluate our model, an evaluation set V is first assembled by extracting spatio-temporal cuboids from the test videos that contain the labelled anomalies. All the datasets considered to evaluate this work are ground-truth annotated. Hence, we can assemble a set of labels Y, which informs us of the anomalous nature of all members in V. The spatio-temporal cuboids are first transformed into the feature space $F_v = E_\gamma(V)$. The multivariate Gaussian mixture model G is now queried for each encoding $f_v \in F_v$ to determine which of the K Gaussians' f_v belongs to. Let g be this Gaussian. The mean and co-variance of the distribution g , namely, μ_g and Σ_g are used to evaluate the Mahalanobis distance [23] of f_v from μ_g using:

$$l = \sqrt{(f_v - \mu_g)^T \Sigma_g^{-1} (f_v - \mu_g)} \tag{10}$$

The Mahalanobis distance, is a measure that evaluates the number of standard deviations from f_v to μ_g. Hence, we can use this distance as a measure of the anomaly score *for each spatio-temporal cuboid*. In other words, the greater the Mahalanobis distance, the higher the probability that the spatio-temporal cuboid is an anomaly.

Once the anomaly scores from all samples in V are collected, we can decide a threshold (T) to be applied on the anomaly score to classify each spatio-temporal cuboid as an anomaly using the labels Y. A good example of a threshold is one that maximizes the F1-score on the task of classifying V. Each incoming test sample (x') can then be classified to be normal or abnormal as follows:

1) $$f_x = E_\gamma(x') \tag{11}$$
2) $$g = \arg\min_i \sqrt{(f_x - \mu_i)^T (f_x - \mu_i)}, i \in K \tag{12}$$
3) $$l = \sqrt{(f_v - \mu_g)^T \Sigma_g^{-1} (f_v - \mu_g)} \tag{13}$$
4) $$x' = \begin{cases} anomaly & l \geq T \\ normal & l < T \end{cases} \tag{14}$$

V. Experiments and Results

A. Evaluation scheme

The classification metrics including accuracy, F1-score, Area Under the ROC Curve (AUC), etc. were calculated at the spatio-temporal cuboid level. The algorithm is determined to have made a correct prediction if there are pixels representing an abnormal event in a cuboid that it actually evaluates to be anomalous. In general, other methods use a frame-level evaluation in addition to the spatio-temporal level method in this paper. Since we focus on localizing the anomaly in both the time and spatial domains, we deem the frame-level measure to be less pertinent and have not included it in this paper.

B. Artificial dataset

The artificial dataset (described in Section III-A) was developed to make arbitrarily large ground-truth annotated data available for anomaly detection from videos. In our experiments, we used 60,000 frames for training and 6,500 frames for evaluation. There is a total of 100 anomalous events (triangles) recorded in all of the test frames. The best results obtained on the evaluation set, along with the hyperparameters for training the model on the artificial dataset is represented in Table III. The ROC curve for the evaluation set is shown in Fig. 4.

Since we have a system from which we can generate variable size datasets, we can examine the relationship between the classification results and the dataset size (Fig. 5). We can observe that the classification error ($1 - AUC$) asymptotically tends to zero with infinite training data. This relationship clearly reinforces the importance of having very large datasets for training deep-learning models for this task. For a task as challenging as anomaly detection in videos, we need to have a targeted effort to collect high quality datasets with ground truth annotations as an investment towards more reliable models.

Data preprocessing parameters	**Value**
P	8
(S_1, S_2)	(32,32)
S_t	7
Hyperparameter	**Value**
n (Number of features in the encoding space)	32
K (Number of Gaussians, cluster centroids)	20
R (Number of training epochs)	25
T (Distance threshold)	89.06
λ	0.001
Accuracy	**98.4%**
F1-Score	**83.21**

Table III: Hyperparameters for training the model on the artificial dataset and classification metrics.

C. UCSD dataset

The UCSD dataset is divided into the UCSDPed1 and UCSDPed2 datasets based on the two separate camera locations. The UCSD datasets were pre-processed using

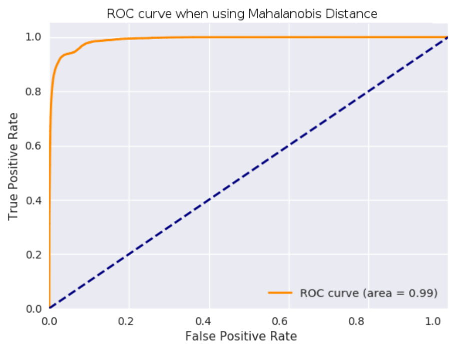

Figure 4: ROC curve for the artificial dataset. Dotted blue line represents a random classifier.

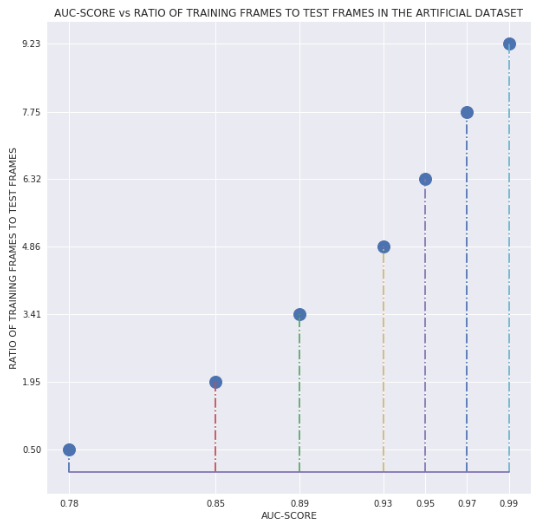

Figure 5: The AUC scores of the classifier for varying training-set sizes of the artificial dataset.

the parameters in Table IV. The best results obtained on the UCSDPed1 dataset and the hyperparameters for training the model are represented in Table V. Similarly, the best results obtained on the UCSDPed2 dataset and the hyperparameters for training the model are in Table VI.

D. Comparison with the state of the art

In this section, we compare our algorithm with some top candidates for the state of the art in the UCSD datasets.

In Fig. 6, ROC curves of our algorithm are plotted along with the ROC curves from the SF (Social Force) model [24], the MDT (Mixture of Dynamic Textures) model [25], SR (Sparse Reconstruction) [26], fast detection at 150 fps model [27], SHD (Statistical Hypothesis Detector) [28], AMDN (Appearance and motion deep-net) [2] and GM (Gaussian Mixture) variational autoencoder [5]. These results are from the UCSDPed1 dataset.

Parameter	Value
P	8
(S_1, S_2)	(48,48)
S_t	12

Table IV: Parameters for pre-processing the artificial dataset.

Hyperparameter	Value
n (Number of features in the encoding space)	32
K (Number of Gaussians, cluster centroids)	20
R (Number of epochs)	110
T (Distance threshold)	24.47
λ	0.01
Accuracy	**85.42%**
F1-Score	**37.18**

Table V: Hyperparameters for training the model on the UCSDPed1 dataset and classification metrics.

Hyperparameter	Value
n (Number of features in the encoding space)	32
K (Number of Gaussians, cluster centroids)	20
R (Number of epochs)	75
T (Distance threshold)	22.94
λ	0.01
Accuracy	**88.03%**
F1-Score	**44.3**

Table VI: Hyperparameters for training the model on the UCSDPed2 dataset and classification metrics.

Similarly, in Fig. 7, ROC curves from our algorithm can be observed along with the ROC curves from the SF model [24], MDT model [25], DA (Deep Anomaly) [4] and GM variational autoencoder [5]. These results are from the UCSDPed2 dataset.

It may be observed from Figs. 6 and 7, that our model (in orange) has a very high true positive to false positive ratio at lower thresholds compared to all other models. Hence, even though there are some misclassifications, none of the anomalies are missed at these thresholds (a desirable property). From the comparative ROC curves, and with the assistance of Table VII, it may be concluded that the results from our model can be considered competitive with the other methods. A few examples of classification using our model on the datasets can be seen in Fig. 8.

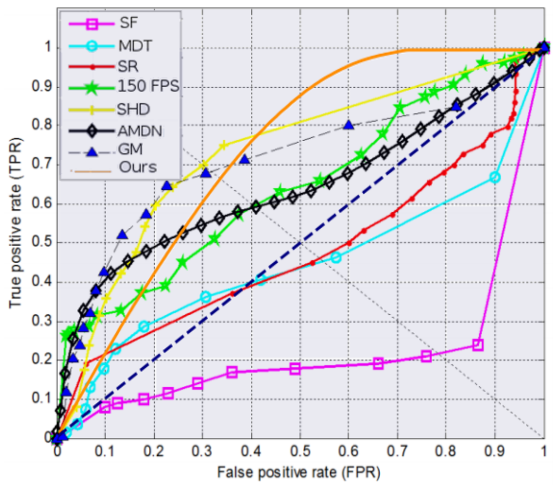

Figure 6: Comparison of the ROC curves of our algorithm vs SF [24], MDT [25], SR [26], Detection at 150 FPS [27], SHD [28] , AMDN [2]. and GM [5]. Results from UCSDPed1 dataset.

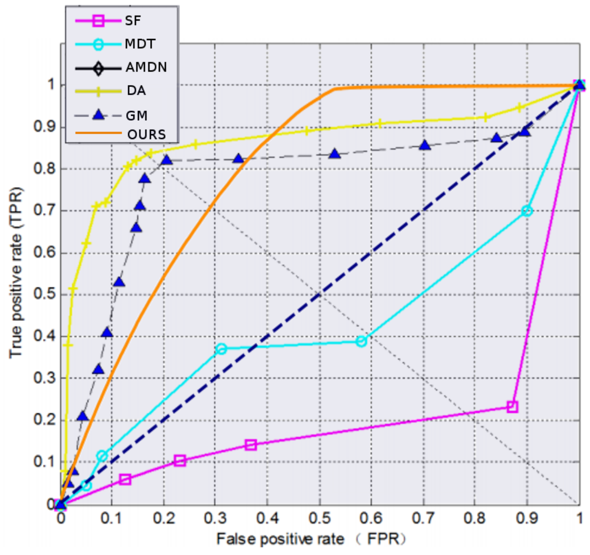

Figure 7: Comparison of the ROC curves of our algorithm vs SF [24], MDT [25], DA [4], GM [5]. Results from the UCSDPed2 dataset.

Dataset	Best AUC	AUC (ours)
UCSDPed1	0.731 (SHD)	**0.74**
UCSDPed2	0.782 (GM)	**0.79**

Table VII: Comparison of our algorithm vs the best AUC (Area under ROC curve) in literature.

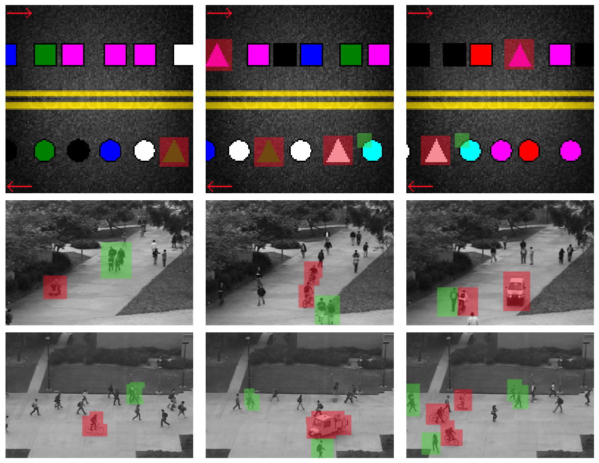

Figure 8: Illustration of spatio-temporal cuboid classification. Red hue is for correctly classified spatio-temporal cuboids, green hue is for misclassified spatio-temporal cuboids. Top row: Artificial Dataset, Middle row: UCSDPed1 dataset, Bottom row: UCSDPed2 dataset

VI. Conclusion

We introduce a novel method for anomaly detection in videos using adaptive clustering in the latent encoding space of a convolutional autoencoder. The results from our algorithm are competitive when compared to other state of the art methods for this task. Our algorithm has a very high true positive to false positive ratio at lower thresholds unlike other methods. We do not miss a single anomalous event at these thresholds, albeit risking a very low false positive rate.

As reflected in the experimental results, it is key to have large ground truth annotated datasets for training deep models for this task. We have additionally introduced and made available, a framework to generate an arbitrarily large synthetic video dataset for anomaly detection with ground truth annotations.

As an aside, this research was originally undertaken to determine a fool-proof thresholding methodology for practical anomaly detection problems. Instead, we have concluded (see Fig. 5) that the error (misclassification) reduces asymptotically to zero with infinite training data. Therefore, the rule is to choose the error to be as small as can be tolerated.

References

[1] M. Sabokrou, M. Fathy, M. Hoseini, and R. Klette, "Real-time anomaly detection and localization in crowded scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 56–62.

[2] D. Xu, E. Ricci, Y. Yan, J. Song, and N. Sebe, "Learning deep representations of appearance and motion for anomalous event detection," *arXiv preprint arXiv:1510.01553*, 2015.

[3] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[4] M. Sabokrou, M. Fayyaz, M. Fathy, Z. Moayed, and R. Klette, "Deep-anomaly: Fully convolutional neural network for fast anomaly detection in crowded scenes," *Computer Vision and Image Understanding*, 2018.

[5] Y. Fan, G. Wen, D. Li, S. Qiu, and M. D. Levine, "Video anomaly detection and localization via gaussian mixture fully convolutional variational autoencoder," *arXiv preprint arXiv:1805.11223*, 2018.

[6] J. Wang, A. Cherian, and F. Porikli, "Ordered pooling of optical flow sequences for action recognition," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 168–176.

[7] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, "Learning temporal regularity in video sequences," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 733–742.

[8] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *Computer Vision (ICCV), 2017 IEEE International Conference on*. IEEE, 2017, pp. 843–852.

[9] J. Hestness, S. Narang, N. Ardalani, G. Diamos, H. Jun, H. Kianinejad, M. Patwary, M. Ali, Y. Yang, and Y. Zhou, "Deep learning scaling is predictable, empirically," *arXiv preprint arXiv:1712.00409*, 2017.

[10] A. Gaidon, A. Lopez, and F. Perronnin, "The reasonable effectiveness of synthetic visual data," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 899–901, 2018.

[11] M. Müller, V. Casser, J. Lahoud, N. Smith, and B. Ghanem, "Sim4cv: A photo-realistic simulator for computer vision applications," *International Journal of Computer Vision*, pp. 1–18, 2018.

[12] C. Jiang, S. Qi, Y. Zhu, S. Huang, J. Lin, L.-F. Yu, D. Terzopoulos, and S.-C. Zhu, "Configurable 3d scene synthesis and 2d image rendering with per-pixel ground truth using stochastic grammars," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 920–941, 2018.

[13] C. Sakaridis, D. Dai, and L. Van Gool, "Semantic foggy scene understanding with synthetic data," *International Journal of Computer Vision*, pp. 1–20, 2018.

[14] G. Rogez and C. Schmid, "Image-based synthesis for deep 3d human pose estimation," *International Journal of Computer Vision*, pp. 1–16, 2018.

[15] A. Chan and N. Vasconcelos, "Ucsd pedestrian database," 2008.

[16] A. L. Maas, A. Y. Hannun, and A. Y. Ng, "Rectifier nonlinearities improve neural network acoustic models," in *Proc. icml*, vol. 30, no. 1, 2013, p. 3.

[17] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[18] B. Yang, X. Fu, N. D. Sidiropoulos, and M. Hong, "Towards k-means-friendly spaces: Simultaneous deep learning and clustering," *arXiv preprint arXiv:1610.04794*, 2016.

[19] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE signal processing magazine*, vol. 26, no. 1, pp. 98–117, 2009.

[20] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE transactions on image processing*, vol. 13, no. 4, pp. 600–612, 2004.

[21] D. Reynolds, "Gaussian mixture models," *Encyclopedia of biometrics*, pp. 827–832, 2015.

[22] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal processing magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[23] P. C. Mahalanobis, "On the generalized distance in statistics." National Institute of Science of India, 1936.

[24] R. Mehran, A. Oyama, and M. Shah, "Abnormal crowd behavior detection using social force model," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 935–942.

[25] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 1975–1981.

[26] Y. Cong, J. Yuan, and J. Liu, "Sparse reconstruction cost for abnormal event detection," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3449–3456.

[27] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2720–2727.

[28] Y. Yuan, Y. Feng, and X. Lu, "Statistical hypothesis detector for abnormal event detection in crowded scenes," *IEEE Transactions on Cybernetics*, vol. 47, no. 11, pp. 3597–3608, Nov 2017.