Deep-Cascade: Cascading 3D Deep Neural Networks for Fast Anomaly Detection and Localization in Crowded Scenes

Mohammad Sabokrou, Mohsen Fayyaz, Mahmood Fathy, and Reinhard Klette

***Abstract*— This paper proposes a fast and reliable method for anomaly detection and localization in video data showing crowded scenes. Time-efficient anomaly localization is an ongoing challenge and subject of this paper. We propose a cubic-patch-based method, characterised by a cascade of classifiers, which makes use of an advanced feature-learning approach. Our cascade of classifiers has two main stages. First, a light but deep 3D auto-encoder is used for early identification of "many" normal cubic patches. This deep network operates on small cubic patches as being the first stage, before carefully resizing the remaining candidates of interest, and evaluating those at the second stage using a more complex and deeper 3D convolutional neural network (CNN). We divide the deep auto-encoder and the CNN into multiple sub-stages, which operate as cascaded classifiers. Shallow layers of the cascaded deep networks (designed as Gaussian classifiers, acting as weak single-class classifiers) detect "simple" normal patches, such as background patches and more complex normal patches, are detected at deeper layers. It is shown that the proposed novel technique (a cascade of two cascaded classifiers) performs comparable to current top-performing detection and localization methods on standard benchmarks, but outperforms those in general with respect to required computation time.**

***Index Terms*— Anomaly detection, deep neural network, video analysis, pedestrian scenes.**

I. Introduction

Anomaly detection and localization is a challenging problem in intelligent video surveillance. There are various definitions for an "anomaly" in a video context. In general, an *anomaly* is opposed to *normal events*; it refers to an event which occurs rarely. Consequently, we are looking for basically unknown events for which modeling is very hard, or even impractical. Usually, state-of-the-art approaches learn one or several reference models for normal events based on training data. In a testing phase, patches or regions of video data are checked in relation to those reference models. If rejected, a patch or region is considered to indicate an anomaly. There is a diversity of normal events for crowded scenes. Representing different normal events by just one set of features leads to inaccurate results and high computational costs. The descriptional complexity of a reference model is defined by the "hardest" normal event.

Additionally, modeling a complex event by high-dimensional features requires many training samples for achieving a good performance. This is a time-consuming and cumbersome process. Proposing a time-efficient solution appears to be difficult. For being accurate and fast, selecting discriminative features is crucial for the performance of the system. Emphasis has been, for some time, on low-level feature-based techniques such as *histograms of gradients* (HoG) or *histograms of optical flow* (HoF). Yang *et al.* [1] argue that hand-crafted features cannot represent video events very well. They suggest unsupervised feature learning. Recently, deep learning provides state-of-the-art results for various tasks in computer vision such as for image classification [2], object detection [3], or activity recognition [4]. *Deep neural networks* (DNNs), such as *convolutional neural networks* (CNNs), are powerful tools, but they are very slow [3], [5], especially when used as a patch-based classifier. Due to the computational costs of DNNs, real-time scanning of patches in video frames is simply impractical.

Solving an anomaly problem with one of the mentioned methods is likely to be inaccurate and time consuming for the following reasons: (1) The lack of anomaly training samples, (2) the representation all normal events by one feature set, (3) the processing of all regions following a unique model, without considering region complexity, (4) the use of hand-crafted features which insufficiently discriminate patches, and which are even time-consuming. We propose an approach to overcome these four difficulties, and show that this leads to both accurate and fast anomaly detection.

Our approach is defined by a *competitive cascade of DNNs*. This cascade classifier combines two stages, a very small stack of auto-encoders, and a CNN. Following the general paradigm of feature detection [6], first we detect interest or key points with low time complexity (our Stage 1), then we detect anomalies by densely extracting and modeling discriminative patches at interest points. Processing only patches at interest points, instead of all possible patches, is a general key for

Manuscript received July 20, 2016; revised January 5, 2017 and February 5, 2017; accepted February 5, 2017. Date of publication February 16, 2017; date of current version March 9, 2017. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Vladimir Stankovic.

M. Sabokrou and M. Fayyaz are with the Department of ICT, Malek-Ashtar University of Technology, Tehran 1774-15875, Iran (e-mail: sabokro@gmail.com).

M. Fathy is with the Iran University of Science and Technology, Tehran 1684-613114, Iran (e-mail: mahfathy@iust.ac.ir).

R. Klette is with the School of Engineering, Computer & Mathematical Sciences, Auckland University of Technology, Auckland 1142, New Zealand (e-mail: rklette@aut.ac.nz).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2017.2670780

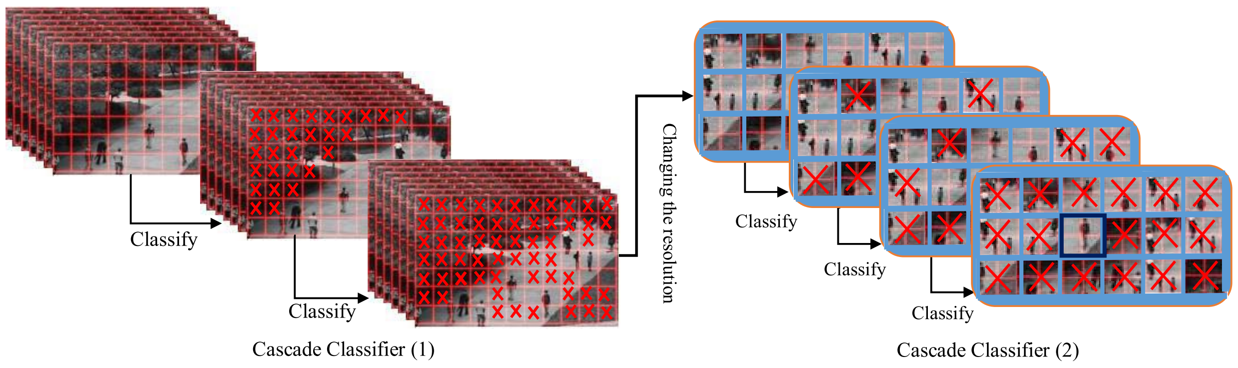

Fig. 1. Our cascade for anomaly detection. *Top:*Processing of small patches for early rejection of normal patches. *Bottom:*Resized remaining patches and further rejection of normal patches

achieving time efficiency. Every deep network represents the input data hierarchically (layer by layer). Thus, we model the normal patches using different representations, which are generated by equivalent intermediate layers. Based on these models, the normal patches are identified in intermediate layers of the deep network, quickly. Similar to a cascade classifier, we expect that the detection of simple normal patches is done in the shallow layers, and that of more complex normal patches in deeper layers.

We call such a network a *cascaded deep network* (CDN). As all layers of such a CDN are not included in detecting any of the patches, it works faster than a conventional DNN. Altogether, we propose a cascade classifier with two stages. These two stages are both equipped with a CDN. Consequently, we speak about *cascading of CDNs*.

Figure 1 shows an example of anomaly detection and localization. At first, a small deep neural network represents the small patches of a frame, and a weak classifier rejects many normal patches in two steps. Remaining patches are then resized into larger patches where the centroid (the *point of interest*) of a patch remains at the same relative location; these larger patches are processed in a deep network and by a weak classifier in four steps; here we reject many more normal patches. The final anomaly detection is done based on features in deeper layers of the deep network using a Gaussian classifier. The shown $\times$ mark labels rejected patches in each step.

We are not the first who consider the use of a DNN in a cascade. Our work is similar to methods reported in [7]–[9] for pedestrian detection, and in [10] for face detection. However, to the best of our knowledge, we are the first in cascading a DNN with embedding a classifier at middle layers, aiming at identifying objects in different layers of the network based on their complexity. Our main contributions are as follows:

- We introduce a DNN-based cascade classifier for fast anomaly detection by hierarchically modeling normal patches using deep features and Gauss distributions.
- Our accuracy is comparable to top-performing methods, but we achieve 15 fps on a mediocre CPU, and 130 fps on a mediocre GPU.
- We present a paradigm for extracting interest-points in a video defined in the spatio-temporal domain.

Our work benefits from considering cascade classifiers, feature detectors, and ideas of deep learning. The rest of this paper is organized as follows. Section II reviews related work. Section III introduces the proposed approach: first the overall framework, then the cascaded deep network structure, time complexities of cascaded CNN vs. CNN, and the anomaly classification scheme, one after the other. Section IV presents experimental results, comparisons, and an analysis. Section V concludes.

II. Related Work

Anomalous event detection and localization is an important and interesting, but also cumbersome task in computer vision. Researchers contribute to solutions for more than 10 years. A wide variety of methods has been introduced for anomaly detection over those years. Initially, methods studied the trajectories of objects in video data [11]–[19]. An object was labeled as being an anomaly if it does not follow the learned normal trajectories. These methods cannot handle the occlusion problem, and they are also computationally very expensive for crowded scenes.

For handling high scene complexity and occlusion, some methods were introduced using low-level features such as HoG or HoF. These methods attempt to learn shapes and spatio-temporal relations using low-level feature distributions.

[20] introduces a joint detector of temporal and spatial anomalies; the authors use a *mixture of dynamic textures* (MDT) for modeling the normal crowd behavior. The authors consider the outliers with respect to the model as being anomalous events. Authors use discriminant saliency for distinguishing spatial anomalies from normal events. An extended version of this work has been provided in [31].

In [23], unusual events are detected by multiple monitors which use local and low level features. [24] proposes a probabilistic method to handle local spatio-temporal anomalies. The authors use spatio-temporal features and a K-nearest neighbor method to detect anomalies among the video regions.

Behavior is modeled in [25] by using pixel-motion properties. For normal events, a spatio-temporal co-occurrence matrix is trained. Video data are represented by using this matrix and a Markov random field (MRF). This representation is then used to detect anomalies.

Local optical flow patterns are used in [26] represented by a mixture of probabilistic PCA (MPPCA) models. Normal patterns are then defined by using this representation and an MRF.

[27] uses extracted spatio-temporal gradients to train a Gaussian model. Then, for detecting anomalies, the authors use a hidden Markov model (HMM). [28] defines abnormal patterns of the video in respect to features: rarity, unexpectedness, and irrelevance. Based on these features, they model the video with an MRF for detecting the anomalies. A method based on social force (SF) is proposed in [29]. This method models motion which plays a key role in anomaly detection.

[30] proposes a hierarchical Bayesian model which combines low-level visual features, simple "atomic" activities, and multi-agent interactions. The proposed model includes improved latent Dirichlet allocation (LDA) and a hierarchical Dirichlet process (HDP) by using unsupervised modeling of interactions. [32] introduces an informative *structural context descriptor* (SCD) to represent a crowd; the spatio-temporal SCD variation of a crowd is analyzed for localizing anomaly regions. [33] proposes a method based on spatio-temporal directed energy filtering to model behaviors. Authors use a histogram comparison method to detect anomalies.

[34] uses a reconstruction method for anomaly detection; the authors label a new observation as being an anomaly if it cannot be reconstructed using previous observations. A similar framework is proposed in [35] based on learning a dictionary representing normal events.

[41] proposes a context-aware anomaly detection algorithm for which the authors represent video data by using motions and video context. In [36], a descriptor is proposed for modeling both motion and appearance, called *motion context*. Anomaly detection is considered as being a matching problem. In other words, if a test patch does not match the training normal patches, it is considered as being an abnormal patch.

Currently, sparse representations of events [35], [37] in video data achieve a good performance. [37] proposes a high-speed method for anomaly detection based on sparse representation. This method has succeeded with high detection rates at a speed of 140- 150 frames per second.

[35] takes advantage of a learned over-complete normal basis set from training data; then the authors introduce a cost for the sparse reconstruction of a test patch for detecting abnormal patches. A scene-parsing approach is presented in [38] where all the object hypotheses of the foreground in a frame are explained by normal training. Those hypotheses, which cannot be explained by normal training, are then considered to be abnormal.

[40] introduces a method for learning the events in video data by constructing a hierarchical code-book for dominant events in the video. [50] uses sparse semi-nonnegative matrix factorization (SSMF) for learning local patterns of pixels. Then, a probability model using those local patterns of pixels is learned for considering both the spatial and temporal context. This method is totally unsupervised, and anomalies are detected by learned models.

Work reported in [42], [43] models normal events based on sets of representative features which are learned on auto-encoders [45]; these authors use a one-class classifier for detecting anomaly as being an outlier compared to the target (i.e. normal) class. In [51] two novel cubic-patch-based anomaly detector are proposed. The detectors use an auto-encoder reconstruction error and sparse representation of the input video for detecting anomalies.

Performances for anomaly *detection* of methods reported in [29], [31], [35]–[37], [39], [40] are already reasonable but still not yet with respect to accurate anomaly *localization*. These methods are typically also not capable to run in real-time on real-world anomaly problems.

III. Proposed System

We present and use a specific design of DNNs for a hierarchical representation of normal patches via discriminative features. The proposed method is a cascade classifier which is built on two DNNs. The intermediate layers of the DNNs are equipped with a one-class Gaussian classifier. They are considered as a cascade classifier sub-stage. The two DNNs define two *main stages*, and the classifiers are the *sub-stages*.

A. Overall Framework

Figure 2 shows a sketch of our anomaly detector.As mentioned before, motion of objects play a substantial role in detecting anomalous events. Therefore, processing should be done with respect to irregular motion or speed of an object (or of a patch). Consequently, we consider all patches to be cubic: for evaluating a frame I_t, we divide it into windows of size $M \times N$, and a cubic patch is then defined by K windows of size $M \times N$ at the same location in frames I_{t-K+1} to I_t. These cubic patches provide useful information for both motion and shape, as well as local signal distribution in a patch. Stage 1 detects quickly most of the normal patches; the remaining (i.e. “challenging”) patches are sent to Stage 2. For checking a frame in a video at Stage 1, we partition it into cubic patches of size $10 \times 10 \times 5$ (i. e. without any overlap).

For example, a 250×250 frame converts into 625 cubic patches, and a short video of 200 frames into $625 \cdot 200 = 125,000$ patches. As the number of anomaly patches is much smaller than of normal patches (similar to [10] for face detection), it is feasible that we leave 30% of patches as being potentially an anomaly patch. These remaining challenging patches are considered at the next stage. We mark the centroid of a remaining small patch as being an interest point. We extract nine large patches of size $40 \times 40 \times 20$ around an interest point and pass those to Stage 2, for more accurate checking. For obtaining those nine large cubic patches, we resize a small patch into a larger one, and generate eight more with a step distance of 5 (away from the interest point) into eight different directions. See Fig. 2.

The use of small patches for detecting anomalies leads to a good performance with respect to true positives, but small patches also generate many false positives [42], [53]. Thus, it is reasonable that we reject at first many of the normal patches at Stage 1 quickly, but, to prevent a high false anomaly detection rate, we process the remaining patches again more carefully. This design considers both accuracy and

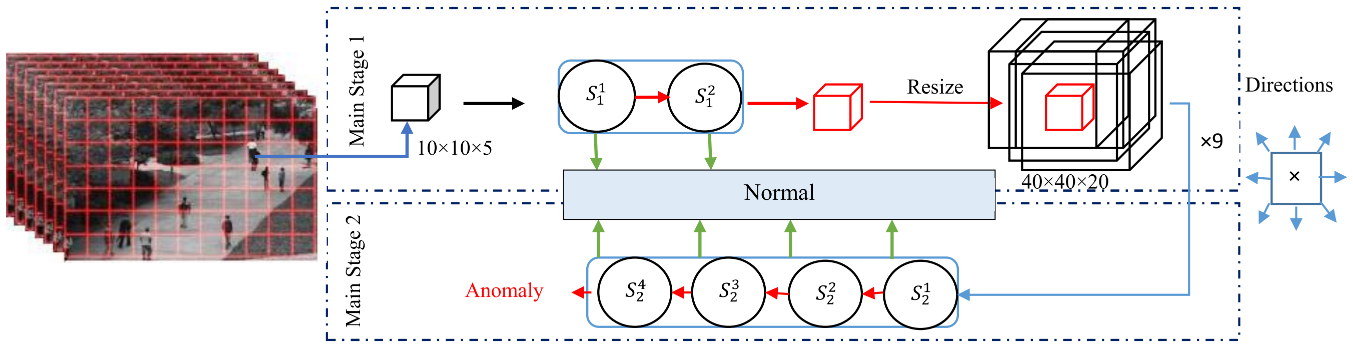

Fig. 2. Work-flow of the proposed system.*Top:* Stage 1 has a cascaded stack-auto-encoder with two (S_1^1 and S_1^2)layers as sub-stages.*Bottom:* Stage 2 has a cascaded CNN with four sub-stages (S_2^1, S_2^2, S_2^3 and S_2^4).

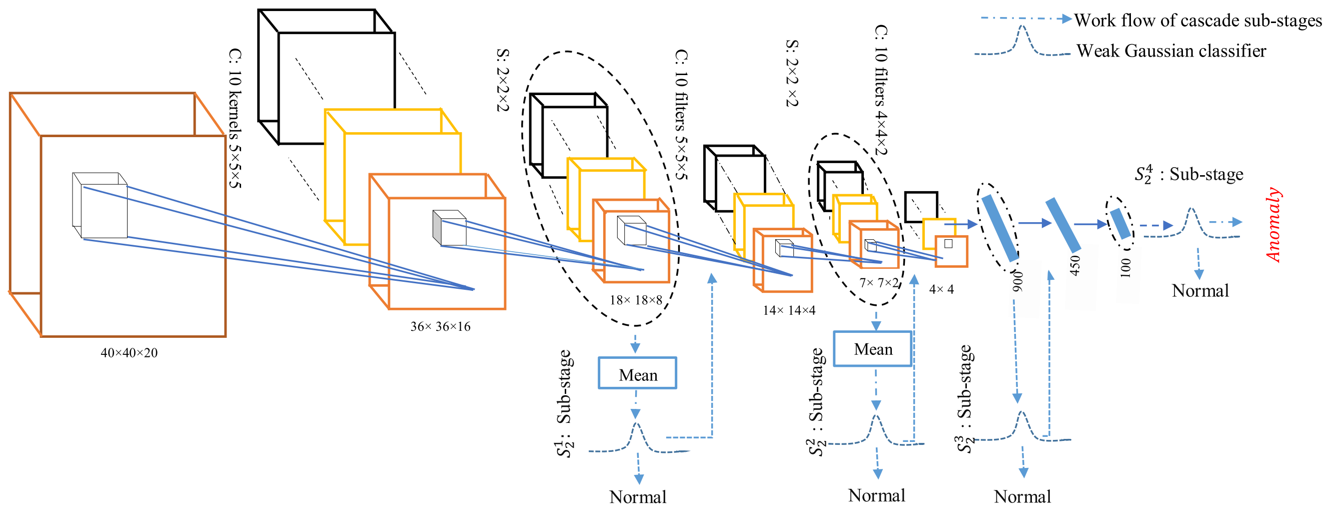

Fig. 3. This sketch of our CNN architecture illustrates subsequent processing of one cubic kernel in the first convolution layer; C and S stands for “convolution” and “sub-sampling” (non-padded convolution is used). The mean operation is defined by channel-wise averaging.

speed. Both stages contain a DNN for representing the video. Sabokrou *et al.* [42] and Xu *et al.* [46] show that feature learning (by an auto-encoder) performs as well as state-of-the-art methods, even more time-efficient. Deeper features are more discriminative. He and Sun [47] confirm that network depth is of high priority for improving accuracy, also in cases where width or kernel size are reduced to compensate increases in computation time.

Thus, instead of a shallow auto-encoder as used in [42] with high-dimensional features, we use a small but deep stack auto-encoder at Stage 1 and a CNN which has more types and numbers of layers at Stage 2. By analyzing the performance of features that are generated in layers of different depths we were able to confirm that shallow features can work well, but those generated at larger depth lead to a better performance. Thus, for solving the anomaly problem, the shallow features define time-efficient cues for a fast detection of simple normal patches, such as in the background.

B. Cascaded Deep Network Structure

In a real-world anomaly problem, there are events of varying levels of complexity. Simple events might be defined by background appearance, and more complex ones by dynamic anomalies. Based on our observations (as outlined in the previous section), we design a cascaded classifier where the intermediate layers of a CNN (or of a stacked-auto encoder) take the roles of sub-stages of a cascade classifier. This means that we detect simple normal patches in early layers with low computational costs, and more difficult patches at the end of the CNN, causing higher computational costs. For speeding up the classification of a patch, we do not use all of the generated maps for every feature in intermediate layers, just the average of all maps is used as an approximation of all maps to represent an input patch.

For example, see the first sub-stage of the cascaded CNN in Fig. 3. There we have $18 \times 18 \times 8$ features in 10 different maps (i.e. each patch is represented by a feature block of size $18\times18\times8\times10$). This means that these 10 maps are considered for classification, consequently every patch is represented by $18 \times 18 \times 8$ features. In other words, we exploited an average pooling operation over the feature vector of each patch to pool the average of all the features of each patch (not to be confused with a spatial pooling operation). As an expected side effect of this, we may decrease the power of the used features. But consider that we exploit this classifier just as a weak one, similar to [42].

1) Cascaded Stacked Auto-Encoder: A stacked auto-encoder is a popular method for unsupervised feature learning.

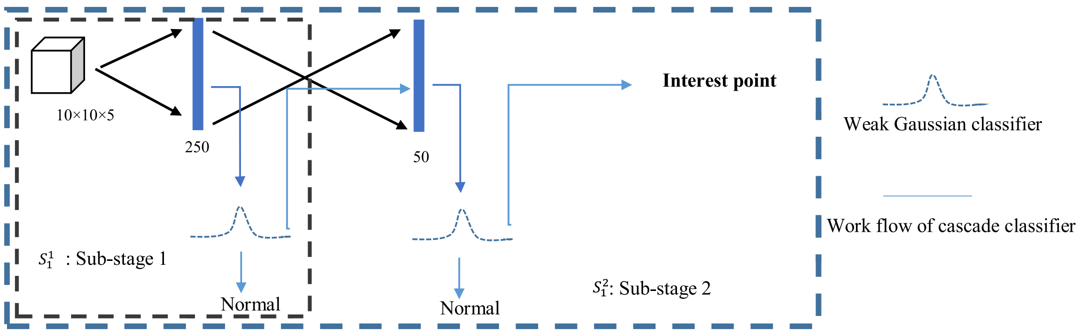

Fig. 4. A small stacked auto-encoder with two layers which work as Stage 1

In this paper, we use a stacked auto-encoder with two layers. It works as a fully connected neural network. Training a deep auto-encoder using large patches is not time-efficient [1]. It may even be too slow in the testing phase. Consequently, only small patches of size $10 \times 10 \times 5$ are considered (see Fig. 4), and the number of hidden layers is selected to be just 2. We embed a Gaussian classifier after each layer in the same way as we already did for the cascaded CNN. They are the same as the classifiers of the cascaded CNN; their training process is independent of the cascaded stacked auto-encoder training.

2) Cascaded Convolution Neural Network: Figure 3 shows the structure of a cascaded CNN as used in this paper. We consider Gaussian classifiers after some intermediate layers at different depths, as shown in Fig. 3, indicated by sub-stages $S_1, \cdots, S_4$. The CNN is considered to be a cascade which consists of the set of sub-stages $S_1, \cdots, S_4$. Let A be an anomaly patch, and A_i be the event that sub-stage S_i classifies A as being an anomaly event. Let D and F be the detection rate and the false positives of the cascade; they are computed as follows [49]:

$$D = \Pr[A_4, \cdots, A_1|A] = \prod_{i=1}^{4} d_i \tag{1}$$

$$F = \Pr[A_4, \cdots, A_1|\bar{A}] = \prod_{i=1}^{4} f_i \tag{2}$$

where, for stage i, $d_i = \Pr[A_i|A_{i-1}, \cdots, A_i, A]$, and $f_i = \Pr[A_i|A_{i-1}, \cdots, A_i, \bar{A}]$ are the detection rate and the false positive rate. We create four weak Gaussian classifiers using the features which were learned in four different depths. The learning process for these classifiers is independent of the cascaded network learning process. We discuss about the training in Section III-D.

According to Eqs. (1) and (2), achieving a low number of false positives is possible by training these weak classifiers. We adjust the classifiers also such that they perform with a very high detection rate. Because our patches are cubic ($40 \times 40 \times 20$), at first 10 cubic kernels of size $5 \times 5 \times 5$ are convolved with a cubic input patch at step size 1 in x, y, and time direction. Next, a $2 \times 2 \times 2$ sub-sampling kernel is applied. Features generated by this layer are used for the S_2^1 sub-stage.

The two convolution and sub-sampling layers are repeated, with $5 \times 5 \times 5$ and $2 \times 2 \times 2$ kernels as 3^{rd} and 4^{th} layers of the cascaded CNN; the output of the 4^{th} layer is considered for sub-stage S_2^2. Then we convolve 10 kernels of size $4 \times 4 \times 2$ as 5^{th} layer, and a fully connected layer of size 900 as the 6^{th} for consideration in S_2^3. Finally, the CNN uses two fully connected layers of sizes 450 and 100, to be considered for S_2^4. We discuss a complexity reduction of the cascaded CNN in Section III-C.

3) Deep Structure Layer-Wise Training: In real-world anomaly problems, training data is typically not labeled, or there are only normal samples. Consequently, training a deep network for this task as an end-to-end network is impossible.

To benefit from the hierarchical feature extraction, a CNN is trained layer-wise, and independent of classifiers between its layers, similar to the work by Ranzato *et al.* in [48].

Both kernels of convolutional layer and weights of fully connected layers are trained by a sparse auto-encoder (SAE). To train s kernels, all training videos are converted to a set of patches with the same size of kernels. In regarding to these training patches, an SAE with s neurons in its hidden layer is trained.

Suppose that we have m normal patches with (w, h, t) dimensions while $\mathbf{x_i} \in \mathbb{R}^D$ is the raw data for learning kernels where $D = w \times h \times t$. The auto-encoder attempts to minimize Equ. (3) by reconstructing the raw data:

$$L = \frac{1}{m}\sum_{i=1}^{m} \|\mathbf{x_i} - W_2\delta(W_1 x_i + b_1) + b_2\|^2 + \lambda \sum_{i=1}^{D}\sum_{j=1}^{s}(W_2^{ji})^2 + \beta \sum_{j=1}^{s} KL(\rho|\rho'_j) \tag{3}$$

where s is the number of nodes in the hidden layer of auto-encoder, $\mathbf{W}_1 \in \mathbb{R}^{s \cdot D}$ and $\mathbf{W}_2 \in \mathbb{R}^{D \cdot s}$ are the weight matrices, which map the input layer nodes to hidden layer nodes and hidden layer nodes to the output layer nodes, respectively. W_2^{ji} is the weight between the j^{th} hidden layer node and the i^{th} output layer node, and δ is a sigmoid function. Furthermore, b_1 and b_2 are the bias of the output layer and the hidden layer, respectively. KL is the Kullback-Leibler divergence [44], and $KL(\rho|\rho'_j)$ is a regularization function; it is set in order to enforce the activation of the hidden layer to be sparse.

KL is based on the similarities between a Bernoulli distribution with ρ as its parameter and the active node distribution. The parameter β is the weight of the penalty term in the sparse

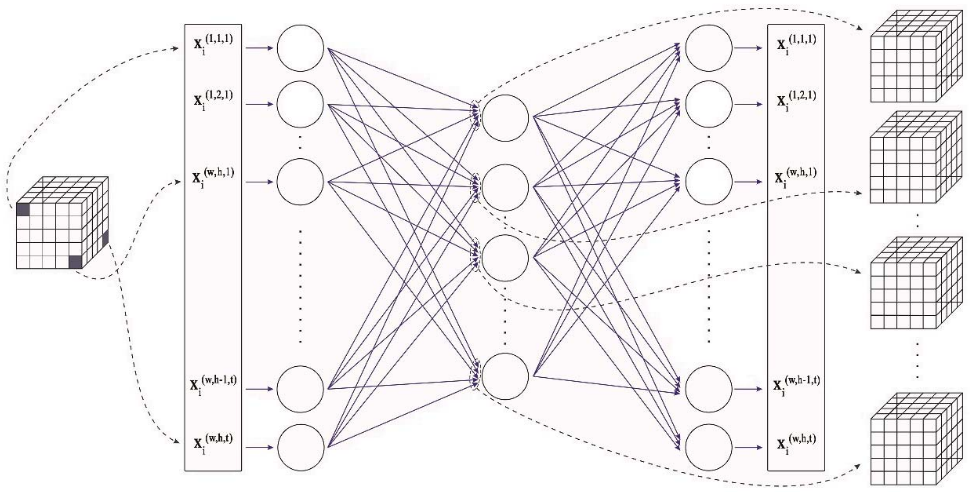

Fig. 5. Illustration for training the kernels of a convolutional layer of a CNN using an SAE. This SAE is trained using all training patches. The $(W_1^{1:D,i})^{i\in[1..s]}$ are considered as being the learned kernels which are shown in the last column in this figure.

auto-encoder objective. We can efficiently optimize the above objective with respect to $\mathbf{W}_1$ via the stochastic gradient descent approach. Moreover, we consider the $W_1^{1:D,i}$ as weights of i^{th} kernel of respective CNN layer, in which i refers to the i^{th} hidden layer node of SAE. Using the above mentioned method, we trained all kernels of all our cascaded CNN layers. As our CNN is a hierarchy of convolutional layers, the input data of each layer is the represented output data of its previous layer. However, the first layer input data is a raw video patch.

In the first layer, we have a 3D receptive field of size $5\times5\times5$ over the input patches extracted from the video frames. For representing these patches, we apply 10 cubic kernels. For training these kernels, we employed an SAE of input size equal to that of the CNN receptive field (i.e. $5\times5\times5$), one hidden layer of size equal to the kernels quantity (i.e. 10), and as output size the same as for the input.

The SAE is trained on the input video patches. When the training is done, we use the encoder part of the SAE as the kernels of the CNN, i.e. the used kernels are $(W_1^{1:D=5\times5\times5,i})^{i=1:10}$; see Fig. 5.

We apply these kernels on the video patches and the 3D max pooling operation is performed on $2\times2\times2$ regions afterwards. At top of the first sub-stage, we have a feature map of size $18\times18\times8\times10$. Then we have the second convolutional layer which is fed by output feature map of sub-stage 1.

The second sub-stage's convolution and pooling have the same specifications as the first sub-stage. For training the convolution kernels, the extracted patches are fed over the output feature map of previous sub-stage and fed them to an SAE with specifications for training the kernels. At top of the second sub-stage a feature map of size $7\times7\times2\times100$ is extracted.

In the third sub-stage we apply a convolution with 10 kernels of size $4\times4\times2$ which are learned like at previous sub-stages, at the top of third sub-stage, we have a $4\times4\times1\times1000$ feature map. Same as previous, we apply an SAE with input size of the feature map and hidden layer size of 900. Subsequently, we apply another SAE with input size of 900 and hidden size of 450.

Eventually, the same technique is applied on 450 feature vectors, and a feature vector of size 100 is extracted at the top of the whole CNN architecture for the fourth sub-stage. In this hierarchy of features, by moving from shallower to deeper levels, more complex features are extracted.

C. Time Complexities of Cascaded CNN vs. CNN

Following [47], assuming a CNN with d convolutional layers, the CNN time complexity for one sample can be described by

$$C_{sample} = \mathcal{O}\left(\sum_{l=1}^{d} n_{l-1}\cdot s_l^2\cdot n_l\cdot m_l^2\right) \tag{4}$$

where n_{l-1} and n_l are the number of input maps and number of kernels in the l^{th} convolutional layer, respectively, and s_l and m_l are the spatial size of the kernel and the spatial size of the output feature map, respectively. As a result, the time complexity of a CNN for N samples can be computed due to Eq. (4) as follows:

$$C_{CNN} = N\times C_{sample} \tag{5}$$

The 1st to the 4th sub-stage work similar to four CNNs of different depths. Let K_i be the time complexity of the ith sub-stage (or CNN), where $K_i > K_{i-1}$ for $i>1$.

In the i^{th} sub-stage (except the 4th which is adjusted for the final detection result), we aim at detecting $(1-\alpha)\times100\%$ of patches as normal patches, and the remaining pass to the next layers (stages). Thus, the S_i stage, $i>0$, just exploits $\alpha^{i-1}\times$ 100% of all patches, instead of $N = 100\%$. Consequently, the time complexity of a CNN is equal to $N\times\sum_{i=1}^{4}K_i$, and the time complexity of a cascaded CNN equals

$$C_{casCNN} = N\times\sum_{i=1}^{4}\alpha^{i-1}K_i \tag{6}$$

Thus, the cascaded CNN speed-up is equal to the ratio of C_{CNN} to C_{casCNN}. The time for transferring data between layers is not considered in these estimates; this takes typically about 5-10% of the total computation time. By using $\alpha = 0.2$, our method is about 2.6 times faster than the CNN.

D. Anomaly Classifiers

Let S_i^j be Sub-stage j of Stage i; S_i^j generates features denoted by P_i^j. For every sub-stage S_i^j, a classifier C_i^j is created (see Section III-B). As an example, all available S_1^1 training patches are represented by P_1^1 feature vectors. A Gaussian model f_1^1 is fitted to them and a C_1^1 classifier is created based on the distance of the samples with f_1^1. In a similar way, the equivalent classifier of each sub-stage is trained and adjusted, but this is different to $C_1^{\{1,2\}}$ which are trained by $10 \times 10 \times 5$ patches, and $C_2^{\{1,2,3,4\}}$ which are trained by $40 \times 40 \times 20$ patches.

In the testing component, we check the Mahalanobis distance between test patch and Gaussian models. If the distance is greater than a threshold τ_i^j, the patch is considered as being an anomaly. We consider $[(C_1^{\{1,2\}}), (C_2^{\{1,2,3,4\}})]$ as being the sub-stages of the cascade classifier. Thus, the training samples of each C_i^j are the challenging patches which are labeled as anomaly using the C_i^{j-1} classifier.

The following equation summarizes anomaly detection:

$$C_i^j(P_i^j) = \begin{cases} Anomaly & \text{if } D(P_i^j, f_i^j) > \tau_i^j \\ Normal & else \end{cases} \tag{7}$$

where $D(.)$ is the Mahalanobis distance between the Gaussian model and a testing patch. If $C_1^{\{1,2\}}$ labels a patch as being an anomaly, then the nine large extracted patches around it are checked with $C_2^{\{1,2,3,4\}}$; if these four classifiers detect a patch as being an anomaly, then it is considered to be an anomaly as the final output of the system.

For constructing the C_i^j weak Gaussian classifier, all training data (i.e. challenging patches in previous stages) in S_i^j are represented by $P_i^j \in \mathbb{R}^{1\times k}$ feature vectors. The mean $\mu \in \mathbb{R}^{1\times k}$ and the variance $\Sigma \in \mathbb{R}^{1\times k}$ of all features of the training samples (i.e. normal patches) are calculated as the parameter of a Gaussian distribution (the normal reference model), which is shown in the following equation:

$$f_i^j(\mathbf{P_i^j}) = \frac{1}{\sqrt{(2\pi)^k|\mathbf{\Sigma}|}} \exp\left[-\frac{1}{2}(\mathbf{P_i^j}-\boldsymbol{\mu})^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{P}_i^j-\boldsymbol{\mu})\right] \tag{8}$$

In the test component, the Mahalanobis distance of feature vector P_i^j of a test sample from the Gaussian distribution f_i^j is calculated by $(P_i^j-\mu)^T\Sigma^{-1}(P_i^j-\mu)$. If this distance exceeds the τ_i^j threshold then we conclude that it is an anomaly because it did not follow from the Gaussian models. The τ_i^j value is selected based on training samples. As mentioned above, we have trained cascaded-CNN and auto-encoders independent of classifiers. So, for training the classifiers we embed them inside the trained networks' layers which they used from the layers of network as feature extractor.

We calculate the Mahalanobis distance of all training samples from the reference model (which is constructed based on training samples). Then, the τ_i^j value is selected to divide the Mahalanobis distance range of surviving training samples of S_i^{j-1} into [0, α_i^j] and (α_i^j,1] parts, where $0 < \alpha < 1$. For S_1^1, the input patches from S_1^0 are patches which are chosen in video frames for the first stage. For S_2^1, the input patches from S_2^0 are the ones which are chosen around the interest points found at the first stage. Up to S_2^4, we choose the value of τ to be the same. However, at the last sub-stage S_2^4, we choose τ_2^4 to be equal to the mean of the top γ fraction of the Mahalanobis distance of represented training and surviving patches; this is to ensure that we only pass anomaly patches at the final sub-stage.

IV. Experimental Results and Comparisons

We empirically demonstrate that our approach can be used in surveillance systems. For this paper, our algorithms have been implemented in Matlab and run on a PC with 2.4 GHz Intel(R) Core(TM) i5 CPU, 8G RAM. The SAE and CNN are implemented on GPU. We used an NVIDIA GT 620 GPU which is a "low-end" GPU for our tests.

Training of both, stack auto-encoder and kernels of CNN, are done with the same training parameter specifications. The sparsity parameter ρ, which defines the desired average activation of hidden neurons, is adjusted to be equal to 0.01. The values of weight λ and sparsity penalty β are 0.0001 and 3, respectively. The learning rate of auto-encoders equals 0.0001.

In our experiments, the τ_1^1 and τ_1^2 thresholds are, respectively, equal to 0.5 and 0.6. With respect to the fact that anomaly patches are very rare (much less than 0.2 of all the patches), for $\tau_2^{1,2,3}$ we allow the classifiers to detect 0.2 of all patches ($\alpha = 0.2$) at each stage as anomaly; it is obvious that all classifiers have thus a high detection rate. For the final classifier we choose τ_2^4 to be equal to the mean of the top 0.1 fraction of Mahalanobis distances at the final stage ($\gamma = 0.1$). We selected these thresholds based on our training samples, and this worked efficiently on the UMN and USCD datasets.

A. Datasets

We compare the performance of our algorithm with that of state-of-the-art methods on UCSD and UMN benchmarks [21], [22].

The **UCSD dataset** is recorded with a static camera at 10 fps. It has two subsets, called Ped1 and Ped2. They show two different outdoor scenes. Dominant mobile objects in the scenes are walking pedestrians. The crowd density varies from low to high. An anomaly is defined by seeing a car, a skateboarder, a wheelchair, or a bicycle moving within the normal pattern of pedestrians. All training frames in Ped1 or Ped2 are normal, i.e. they contain pedestrians only. We evaluate our algorithm on both Ped1 and Ped2.

Ped1: There are 34 normal video samples for training, and 36 abnormal video sequences for testing. There is no frame-level ground truth available for all the video sequences in Ped1, only for some of them.

Each sequence is about about 200 frames long. Frames have a resolution of 158×238. The total number of abnormal frames and of normal frames is $\approx 3,400$ and $\approx 5,000$, respectively.

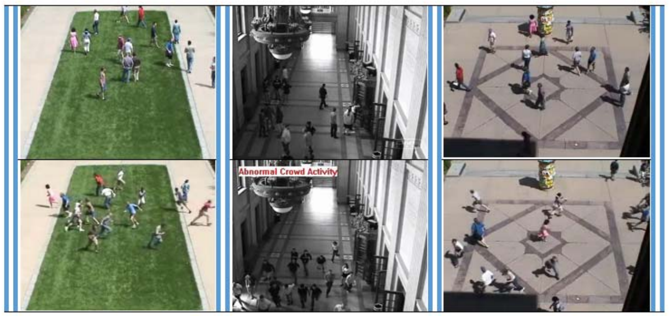

Fig. 6. Examples of normal and abnormal crowd activities in scenes of the UMN dataset.*Top*: Normal. *Bottom*: Abnormal.

Ped2: This set contains 12 and 16 video sequences for testing and training, respectively, at resolution 320×240. To evaluate localization of moving objects, ground truth is available for all test frames. The total number of abnormal and normal frames is $2,384$ and $2,566$, respectively.

UMN Dataset: The UMN dataset is recorded for three different scenarios. In each scenario, a group of people walks normal in an area, but suddenly all people run away (escape); the escape is considered to be the anomaly. Figure 6 shows examples of normal and abnormal frames of this dataset.

B. Evaluation Methodology

We compare our results with state-of-the-art methods. The *receiving operating characteristic equal error rate* (ROC-EER) is the accuracy at the ROC operating point where false positive and false negative rates are equal. The ROC curve and EER have been used in [20], [42] for result evaluation.

We use three measures: at frame level, at pixel level, and at dual pixel level. The measures are as follows:

1) Frame-Level: A frame is considered to be an anomaly if an anomaly is detected for at least one pixel of the frame.

2) Pixel-Level: If 40% (or more) of those pixels are detected as anomaly which are annotated as anomaly in the available ground truth, then the frame is considered to be an anomaly.

3) Dual Pixel Level (DPL): In this measure, a frame is considered as being an anomaly if (1) it satisfies the anomaly condition at pixel level, and (2) at least β percent (for example, 10%) of pixels detected as anomaly are covered by the anomaly ground truth. If, in addition to the anomaly region, irrelevant regions are also considered as being an anomaly, then this measure does not identify the frame as being positive (i.e. an anomaly). This measure is more accurate than the pixel-level measure for evaluating the localization performance [42].

C. Parameter Analysis for Proposed Method

In this sub-section, we analyze the important aspects of the proposed structure for anomaly detection. The impact of the α parameter on run-time and accuracy of the method is explained. Also, we analyze the number of surviving patches in sub-stages of the network. Finally, the performance of each convolutional layer of the CNN is calculated, compared, and analyzed.

TABLE I
SURVIVING SAMPLES VS. CASCADE STAGES

Sub-stages	S_1^1	S_1^2	S_2^1	S_2^2	S_2^3
Survived Patches	233.72	68.16	124.58	23.65	5.23

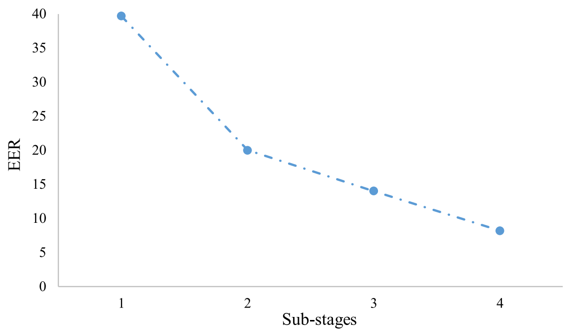

Fig. 7. Performance vs. cascade stages.

All of the experiments in this sub-section are done on UCSD Ped2. Furthermore, the reported results for analyzing the surviving patches are calculated by averaging the results on all of the test frames.

1) Surviving Samples vs. Cascade Stages: The idea of the proposed architecture is rejecting the normal patches based on their complexity as soon as possible, so in each sub-stage only a subset of all input patches is survived. The curve of number of surviving patches v. s. cascade stages is shown in Table. I.

As it is shown in this table, each sub-stage detects most of patches and just a few percent of them remains and reaches to the next sub-stage. The centroids of survived patches in (S_1^2) are considered as interest points of input video frames. Then, as mentioned before, 9 new large patches are extracted around each interest points. These new patches are fed to (S_2^1). Consequently, the amount of the survived patches at this sub-stage has been increased.

2) Performance vs. Cascade Sub-Stages: To show the performance of each cascade sub-stage of the CNN, the proposed method has been analyzed in regard to different numbers of sub-stages. First, we consider only the (S_2^1) sub-stage as an anomaly detector, afterward we have added (S_2^2), (S_2^3) and (S_2^4) as the cascade sub-stages. The performance of all cascade classifiers with different sub-stages is shown in Fig. 7. As shown in the figure, the EER drops as the number of cascades increases due to high detection rates of the classifiers.

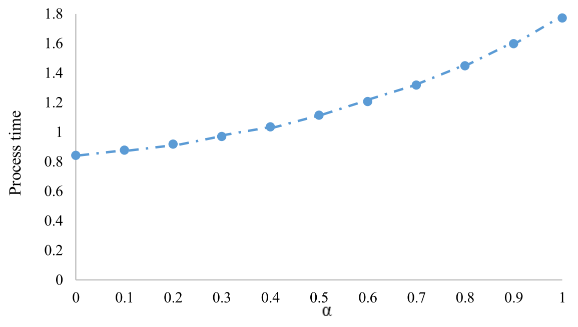

Fig. 8. Run-time vs. α.

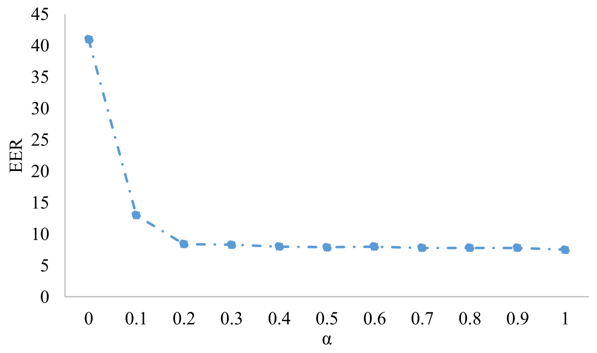

Fig. 9. Performance vs. α.

3) Run-Time vs. α: In each sub-stage, $(1-\alpha)$ of patches have been labeled as normal and remaining patches have been sent to the next sub-stages. Consequently, by selecting a high value for α, the most of patches has been labeled, and a few of them has been passed on to deeper sub-stages. Figure 8 shows the effect of α parameters on the processing time of a video frame. Selecting a high value of α results in reduction of the run-time of the system for processing a frame, however, selecting it larger than a specific threshold will cause that no surviving patch is passed anymore to the next sub-stages.

Moreover, different values of α mean the following: 1) $\alpha = 0$ means that all input patches have been processed just by sub-stage 1; it is obvious that this is very fast, but as patches do not reach deeper sub-stages, the performance drops. 2) $\alpha = 1$ means that all patches are passed on to the final sub-stage and processed at all sub-stages; thus the time complexity is high in this case. All in all, parameter α must be set in a way which adjusts a trade off between run-time and performance of the proposed system.

4) Performance vs. α: Optimum value of α is dependent on the depth of the used CNN. Deeper CNN must be adjusted with higher α to have survived patches in end sub-stage of cascade classifier. In a specific structure such as one is used in this paper, this parameters is selected experimentally. Although, selecting α very low leads to the speedup of the cascade CNN, as the class of all test data is predicted based on shallow layer (not enough discriminative features), doing so leads to the reduction in performance of system. Increasing this parameter from 0 to a specific threshold leads to improving the performance, but from this threshold to 1, the improving is not significant. This is due to the fact that, with large α the most of the processes, process in latest sub-stage, where the patches are classified based an more discriminate patches rather previous sub-stage. Figure 9 shows the EER of the our cascaded CNN in respect to different value of α.

D. Qualitative and Quantitative Results

As a qualitative example, Fig. 10 shows the output of our proposed system on frames from Ped2 UCSD and UMN. First, using a very small deep-network with low complexity, many of the $10 \times 10 \times 5$ normal patches (extracted with step width 10) are quickly detected, and just the challenging (interesting) patches are considered in the next steps. The remaining patches are indicated by green squares in the 2^{nd} column. For more accuracy, for each interest point (patch) we extract nine $40 \times 40 \times 20$ patches for the next anomaly detection step. Examples of those large patches are shown in the 3^{rd} column. Finally the output of Stage 2 is shown in the 4^{th} column. This figure illustrates that our algorithm works as a smart method which just focuses on "suspects" instead of on all patches.

Results for three frames of the UCSD Ped1 dataset are shown in Fig. 11. Detected abnormal events are shown by green borders. Figure 11 illustrates how the proposed method is able to detect and localize anomaly. For simplicity, for the output of our system on those UCSD data, if two patches are detected as anomaly having more than 50% overlap, only one of both is shown.

1) Equal Error Rate Comparison: Table II shows our results in comparison with those achieved by others on Ped1. Our method outperforms at frame level most of the state-of-the-art methods, and is back by 0.1% and ahead by 0.9% to the best previous methods of Yuan *et al.* [32] and Xiao *et al.* [50], respectively.

The pixel level EER of the presented method is also better than for other methods, and close to the value for [50]. For verifying our method on Ped2 at frame level, a ROC comparison with existing methods is provided in Fig. 12 (left). It shows that not only our method is comparable to other methods, even it is superior to them.

More quantitative results for frame-level and on Ped2 are shown in the 2^{nd} column of Table III. This confirms that our method has a good performance compared to that of others. Methods which are reported by Xiao *et al.* [50] and iHOT [54] have close result to ours.

Figure 12 (right) is related to pixel-level ROC on Ped2. Our method is again superior compared to other methods. In Table III (3^{rd} column), we compare the pixel-level EER of our approach to that of other approaches. Our method's EER is 19% where the best result is 17% reported by Xiao *et al.* [50]. Thus, our method is 2% worse in this case. Results show (ROC and EER) that our algorithm (except for [50]) outperforms all the other methods with respect to the pixel-level measure.

2) Dual Pixel Level Comparison: We also use the DPL measure, as defined above, for analyzing the accuracy of anomaly localization. Our algorithm has a good performance, even better than other state-of-the-art methods, at pixel level with $\beta = 5\%$.

This DPL measure is defined in [42], and other state-of-the-art methods did not report the localization performance on

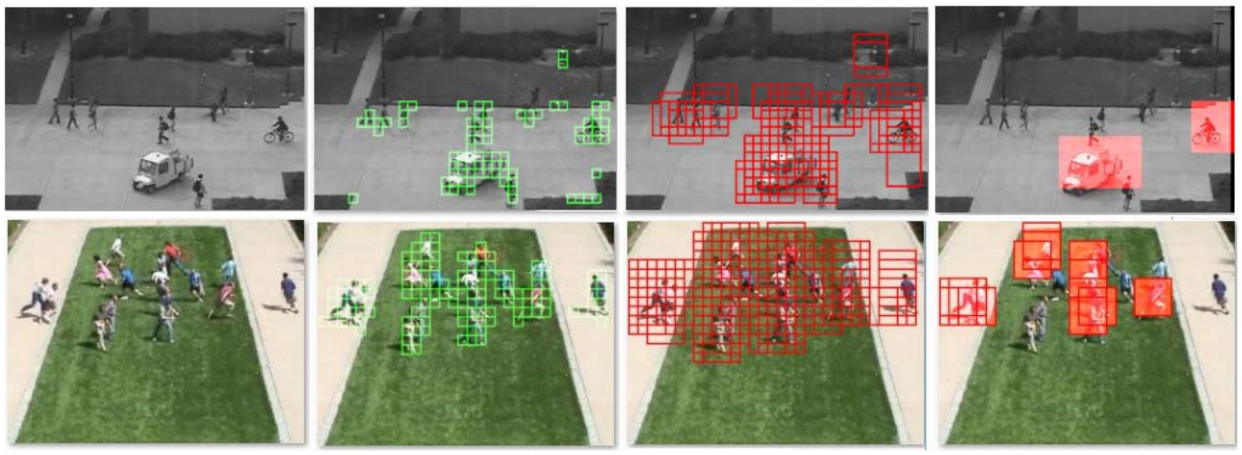

Fig. 10. An example of anomaly detection using our method on Ped2 UCSD and UMN.*Left to right:* (1) Original frame. (2) Points of interest as detected by a light auto-encoder at Stage 1. (3) Large patches which are extracted from interest-points. For simplifying the figure, instead of showing all 9 patches for each interest point, we show just one of them.(4) Output of the system

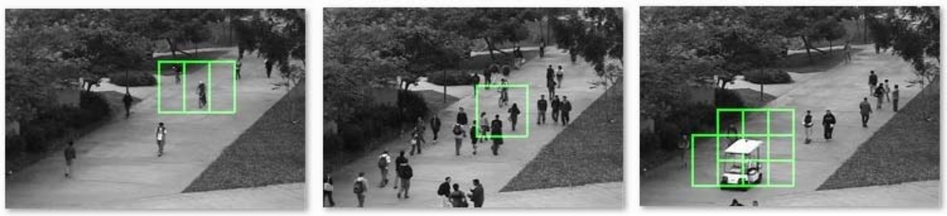

Fig. 11. Examples of anomaly detection on UCSD Ped1.

TABLE II
EER FOR FRAME AND PIXEL LEVEL COMPARISONS ON PED1

Method	Frame-Level	Pixel-Level	Method	Frame-Level	Pixel-Level
IBC [34]	14%	26%	Adam et al. [23]	38%	76%
MPCCA [26]	40%	82%	Zaharescu et al. [33]	29%	41%
MDT [20]	25%	58%	Reddy et al. [52]	22.5%	32%
SRC [36]	19%	54%	Bertini et al. [53]	31%	70%
Dan Xu [43]	22%	—	Saligrama et al. [24]	16%	—
Li et al [31]	17.8%	25.5%	OADC-S [32]	9%	26%
iHOT et al. [54]	19.37%	—	Tan Xiao et al. [50]	10%	16%
Ours	**9.1%**	**15.8%**			

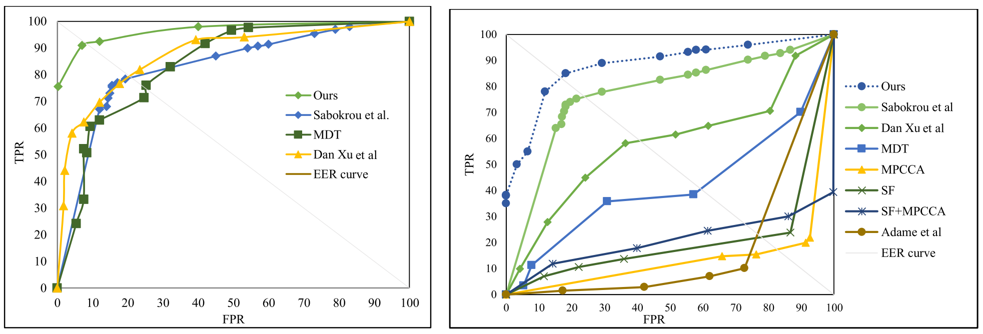

Fig. 12. Comparative ROC curves.*Left*: Frame-level evaluation.*Right*: Pixel- level evaluation.

Ped2 (for $\beta = 5\%$) with respect to this measure. Consequently we just compare our method with [42]. The dual-pixel EER (for $\beta = 5\%$) of our method on Ped1 and Ped2 is 24.5% and 23.8%, respectively. We are better by 3.7% on Ped2 than [42].

Limitations of the UMN dataset are as follows: It contains only three anomaly scenes, it shows abrupt temporal-spatial changes between normal and abnormal frames, and it does not come with pixel-level ground truth.

TABLE III
EER FOR FRAME AND PIXEL LEVEL COMPARISONS ON PED2

Method	Frame-level	Pixel-level	Method	Frame-level	Pixel-level
IBC [34]	13%	26%	Adam et al. [23]	42%	76%
SF [29]	42%	80%	MPCCA [26]	30%	71%
MPCCA+SF [20]	36%	72%	Zaharescu et al. [33]	17%	30%
MDT [20]	24%	54%	Reddy et al. [52]	20%	—
Bertini et al. [53]	30%	—	Saligrama et al. [24]	18%	—
Dan Xu [43]	20%	42%	Li et al [31]	18.5%	29.9%
Tan Xiao et al. [50]	10%	**17%**	Sabokrou [42]	19%	24%
iHOT et al. [54]	8.59%	—	Ours	**8.2%**	19%

TABLE IV
ANOMALY DETECTION PERFORMANCE IN EER AND AUC

Method	Chaotic invariants[12]	SF[29]	Sparse[35]	Saligrama et al[24]	Li et al [31]	Ours
EER	5.3	12.6	2.8	3.4	3.7	2.5
AUC	99.4	94.9	99.6	99.5	99.5	99.6

TABLE V
RUN-TIME COMPARISON (THE TIME IS IN seconds)

Method	IBC	MDT	Roshtkhari et al.	Li et al.	Xiao et al.	SRC	OADC-S	Ours
Ped1	55	17	0.16	0.65	0.22	3.8	0.6	**0.06**
Ped2	68	23	0.18	0.80	0.29	–	0.6	**0.08**

In conclusion, we evaluate our method with EER and AUC at frame-level. Table IV shows EER and AUC results. Previous methods performed perfectly on this dataset, and the AUC of our method is comparable with the previous best result. The EER of our approach is even better (by 0.3%) than the one of the best previous method.

3) Run-Time Analysis: We compare the average time to process one frame in Table V. It confirms that our method has a performance like state-of-the-art methods, and our method even works faster. Our method detects the anomaly at high speed on a CPU which is rather fast.

V. CONCLUSIONS

Deep-cascade, the method which is introduced in this paper, is a way for using advanced features which can be learned by a deep neural network in a cascade structure. We proposed a new anomaly detection method which is efficient in both run-time and accuracy. Two deep networks cooperate together for identifying the challenging patches and for detecting anomalies. The identification of challenging patches is done using a small deep network which works on small patches. Nine large patches in the neighborhood of every detected challenging patch are extracted and passed on to a deep network which comes with more accuracy and complexity. These procedures lead to a speed-up. We detect normal patches based on their complexity and the discriminative power of increasing depth of the networks, with intermediate layers acting as weak Gaussian classifier. Altogether, our approach is a cascade classifier which uses advanced features.

REFERENCES

[1] Y. Yang, G. Shu, and M. Shah, “Semi-supervised learning of feature hierarchies for object detection in a video,” in *Proc. CVPR*, Jun. 2013, pp. 1650–1657.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “ImageNet classification with deep convolutional neural networks,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in *Proc. CVPR*, 2014, pp. 580–587.

[4] K. Simonyan and A. Zisserman, “Two-stream convolutional networks for action recognition in videos,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 568–576.

[5] A. Giusti, D. C. Ciresan, J. Masci, L. M. Gambardella, and J. Schmidhuber, “Fast image scanning with deep max-pooling convolutional neural networks,” in *Proc. ICIP*, Sep. 2013, pp. 4034–4038.

[6] R. Klette, *Concise Computer Vision*. London, U.K.: Springer, 2014.

[7] P. Luo, Y. Tian, X. Wang, and X. Tang, “Switchable deep network for pedestrian detection,” in *Proc. CVPR*, Jun. 2014, pp. 899–906.

[8] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson, “Real-time pedestrian detection with deep network cascades,” in *Proc. BMVC*, 2015, pp. 1–12.

[9] W. Ouyang and X. Wang, “Joint deep learning for pedestrian detection,” in *Proc. ICCV*, Dec. 2013, pp. 2056–2063.

[10] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, “A convolutional neural network cascade for face detection,” in *Proc. CVPR*, Jun. 2015, pp. 5325–5334.

[11] F. Jiang, J. Yuan, S. A. Tsaftaris, and A. K. Katsaggelos, “Anomalous video event detection using spatiotemporal context,” *Comput. Vis. Image Understand.*, vol. 115, no. 3, pp. 323–333, 2011.

[12] S. Wu, B. E. Moore, and M. Shah, “Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes,” in *Proc. CVPR*, Jun. 2010, pp. 2054–2060.

[13] C. Piciarelli, C. Micheloni, and G. L. Foresti, “Trajectory-based anomalous event detection,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1544–1554, Nov. 2008.

[14] C. Piciarelli and G. L. Foresti, “On-line trajectory clustering for anomalous events detection,” *Pattern Recognit. Lett.*, vol. 27, no. 15, pp. 1835–1842, Nov. 2006.

[15] F. Tung, J. S. Zelek, and D. A. Clausi, “Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance,” *Image Vis. Comput.*, vol. 29, no. 4, pp. 230–240, Mar. 2011.

[16] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank, “A system for learning statistical motion patterns,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 9, pp. 1450–1464, Sep. 2006.

[17] B. T. Morris and M. M. Trivedi, “Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2287–2301, Nov. 2011.
[18] S. Calderara, U. Heinemann, A. Prati, R. Cucchiara, and N. Tishby, “Detecting anomalies in people’s trajectories using spectral graph analysis,” *Comput. Vis. Image Understand.*, vol. 115, no. 8, pp. 1099–1111, Aug. 2011.
[19] P. Antonakaki, D. Kosmopoulos, and S. J. Perantonis, “Detecting abnormal human behaviour using multiple cameras,” *Signal Process.*, vol. 89, no. 9, pp. 1723–1738, Sep. 2009.
[20] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, “Anomaly detection in crowded scenes,” in *Proc. CVPR*, 2010, pp. 1975–1981.
[21] *UCSD Dataset*, access on May 1, 2016. [Online]. Available: http://www.svcl.ucsd.edu/projects/anomaly/dataset.html
[22] *UMN Dataset*, access on May 1, 2016. [Online]. Available: http://mha.cs.umn.edu/Movies/Crowd-Activity-All.avi
[23] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, “Robust real-time unusual event detection using multiple fixed-location monitors,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 555–560, Mar. 2008.
[24] V. Saligrama and Z. Chen, “Video anomaly detection based on local statistical aggregates,” in *Proc. CVPR*, Jun. 2012, pp. 2112–2119.
[25] Y. Benezeth, P.-M. Jodoin, V. Saligrama, and C. Rosenberger, “Abnormal events detection based on spatio-temporal co-occurences,” in *Proc. CVPR*, Jun. 2009, pp. 1446–1453.
[26] J. Kim and K. Grauman, “Observe locally, infer globally: A space-time MRF for detecting abnormal activities with incremental updates,” in *Proc. CVPR*, Jun. 2009, pp. 2921–2928.
[27] L. Kratz and K. Nishino, “Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models,” in *Proc. CVPR*, Jun. 2009, pp. 1446–1453.
[28] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, “Semi-supervised adapted HMMs for unusual event detection,” in *Proc. CVPR*, Jun. 2005, pp. 611–618.
[29] R. Mehran, A. Oyama, and M. Shah, “Abnormal crowd behavior detection using social force model,” in *Proc. CVPR*, Jun. 2009, pp. 935–942.
[30] X. Wang, X. Ma, and E. Grimson, “Unsupervised activity perception by hierarchical Bayesian models,” in *Proc. CVPR*, Jun. 2007, pp. 1–8.
[31] W. Li, V. Mahadevan, and N. Vasconcelos, “Anomaly detection and localization in crowded scenes,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 1, pp. 18–32, Jan. 2014.
[32] Y. Yuan, J. Fang, and Q. Wang, “Online anomaly detection in crowd scenes via structure analysis,” *IEEE Trans. Cybern.*, vol. 45, no. 3, pp. 548–561, Mar. 2015.
[33] A. Zaharescu and R. Wildes, “Anomalous behaviour detection using spatiotemporal oriented energies, subset inclusion histogram comparison and event-driven processing,” in *Proc. ECCV*, 2010, pp. 563–576.
[34] O. Boiman and M. Irani, “Detecting irregularities in images and in video,” *Int. J. Comput. Vis.*, vol. 74, pp. 17–31, Aug. 2007.
[35] Y. Cong, J. Yuan, and J. Liu, “Sparse reconstruction cost for abnormal event detection,” in *Proc. CVPR*, Jun. 2011, pp. 3449–3456.
[36] Y. Cong, J. Yuan, and Y. Tang, “Video anomaly search in crowded scenes via spatio-temporal motion context,” *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 10, pp. 1590–1599, Oct. 2013.
[37] C. Lu, J. Shi, and J. Jia, “Abnormal event detection at 150 FPS in MATLAB,” in *Proc. ICCV*, Dec. 2013, pp. 2720–2727.
[38] B. Antic and B. Ommer, “Video parsing for abnormality detection,” in *Proc. ICCV*, Nov. 2011, pp. 2415–2422.
[39] M. J. Roshtkhari and M. D. Levine, “An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions,” *Comput. Vis. Image Understand.*, vol. 117, no. 10, pp. 1436–1452, Oct. 2013.
[40] M. J. Roshtkhari and M. D. Levine, “Online dominant and anomalous behavior detection in videos,” in *Proc. CVPR*, Jun. 2013, pp. 2611–2618.
[41] Y. Zhu, N. M. Nayak, and A. K. Roy-Chowdhury, “Context-aware modeling and recognition of activities in video,” in *Proc. CVPR*, 2013, pp. 2491–2498.
[42] M. Sabokrou, M. Fathy, M. Hoseini, and R. Klette, “Real-time anomaly detection and localization in crowded scenes,” in *Proc. CVPR Workshops*, Jun. 2015, pp. 56–62.
[43] D. Xu, R. Song, X. Wu, N. Li, W. Feng, and H. Qian, “Video anomaly detection based on a hierarchical activity discovery within spatio-temporal contexts,” *Neurocomputing*, vol. 143, pp. 144–152, Nov. 2014.
[44] S. Kullback and R. A. Leibler, “On information and sufficiency,” *Ann. Math. Statist.*, vol. 22, no. 1, pp. 79–86, 1951.
[45] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proc. Int. ACM Conf. Mach. Learn.*, 2008, pp. 1096–1103.
[46] D. Xu, E. Ricci, Y. Yan, J. Song, N. Sebe, and F. B. Kessler, “Learning deep representations of appearance and motion for anomalous event detection,” in *Proc. BMVC*, 2015, pp. 8.1–8.12.
[47] K. He and J. Sun, “Convolutional neural networks at constrained time cost,” in *Proc. CVPR*, Jun. 2015, pp. 5353–5360.
[48] M. Ranzato, F. J. Huang, Y.-L. Boureau, and Y. LeCun, “Unsupervised learning of invariant feature hierarchies with applications to object recognition,” in *Proc. CVPR*, Jun. 2007, pp. 1–8.
[49] J. Wu, S. C. Brubaker, M. D. Mullin, and J. M. Rehg, “Fast asymmetric learning for cascade face detection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 3, pp. 369–382, Mar. 2008.
[50] T. Xiao, C. Zhang, and H. Zha, “Learning to detect anomalies in surveillance video,” *IEEE Signal Process. Lett.*, vol. 22, no. 9, pp. 1477–1481, Sep. 2015.
[51] M. Sabokrou, M. Fathy, and M. Hoseini, “Video anomaly detection and localization based on the sparsity and reconstruction error of auto-encoder,” *IET Electron. Lett.*, vol. 52, no. 13, pp. 1122–1124, 2016.
[52] V. Reddy, C. Sanderson, and B. C. Lovell, “Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture,” in *Proc. CVPR Workshops*, Jun. 2011, pp. 55–61.
[53] M. Bertini, A. Del Bimbo, and L. Seidenari, “Multi-scale and real-time non-parametric approach for anomaly detection and localization,” *Comput. Vis. Image Understand.*, vol. 116, no. 3, pp. 320–329, Mar. 2012.
[54] H. Mousavi, M. Nabi, H. K. Galoogahi, A. Perina, and V. Murino, “Abnormality detection with improved histogram of oriented tracklets,” in *Proc. ICIAP*, 2015, pp. 722–732.

Mohammad Sabokrou received the M.S. and Ph.D. degrees in computer engineering from the Malek-Ashtar University of Technology, Tehran, Iran, in 2013 and 2016, respectively. He is currently a Researcher HPC Lab., in Iran University of Science and Technology, under the supervision of Dr. M. Fathy. His research interests include machine learning (deep learning and outlier detection) and computer vision (crowded scene analysis and activity recognition).

Mohsen Fayyaz received the M.S. degree in artificial intelligence from the Malek-Ashtar University of Technology, Tehran, Iran, in 2016. He is currently a Researcher HPC Lab., in Iran University of Science and Technology, under the supervision of Dr. M. Fathy. His research interests include machine learning (deep learning and multimodal learning) and computer vision (crowded scene analysis and activity recognition).

Mahmood Fathy received the Ph.D. degree in image processing and processor design from the Institute of Science and Technology, University of Manchester, Manchester, U.K., in 1991. He is currently a Professor with the Department of Computer Engineering, Iran University of Science and Technology, Tehran, Iran, where he has been the Head of the Security Information Technology Group, since 2009. He has co-authored five books, over 70 scientific papers for different journals, and 300 conference papers. His research interests include computer networks, network on chip, computer architecture, distributed image and video processing, and intelligent transportation systems.

Reinhard Klette is a fellow of the Royal Society of New Zealand and also a Professor with the Auckland University of Technology, Auckland, New Zealand.

He has co-authored over 400 publications in peer-reviewed journals or conferences, and books on computer vision, object tracking, computer vision for autonomous vehicles, image processing, geometric algorithms, and panoramic imaging. He has authored the book *Concise Computer Vision* (London: Springer, 2014). He was the founding Editor-in-Chief of the *Journal of Control Engineering and Technology* from 2011 to 2013 and an Associate Editor of the IEEE TRANSACTIONS PAMI from 2001 to 2008. He presented over 20 keynotes at international conferences. He is currently on the editorial boards of the *International Journal of Computer Vision* and the *International Journal of Fuzzy Logic and Intelligent Systems*.