ISSN:2348-2079

Volume-6 Issue-2

International Journal of Intellectual Advancements and Research in Engineering Computations

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

[1]Y.Dhushara, [2]P.Kaviranjani, [3]S.S.Kovarrthina, [4]V.Gokul brindha

[1]sridhusha143@gmail.com,[2]kaviranjani.p@gmail.com,[3]kovarrthina@gmail.com,[4]gokulabrindha.v@nandhaengg.org

CSE, NANDHA ENGINEERING COLLEGE, ERODE, INDIA

ABSTARCT

The aim of the project is detecting abnormalities in the crowd behavior.In terms of crowd behavior, while people walks in slow place, if they suddenly started to run then the alarm rings.The propose system first detect the optical flow between frames by using two frame polynomial expansion co-efficients.The magnitude of the difference of two consecutive frames will give us the activity map. The activity map shows how much motion is estimated between consecutive frames. The second by using the differences between activity maps, we obtain Temporal Occupancy Variation (TOV) level and Entropy level.

I.INTRODUCTION

ANOMALY detection and localization is a challenging Aproblem 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.

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 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 _ 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], [8], [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.

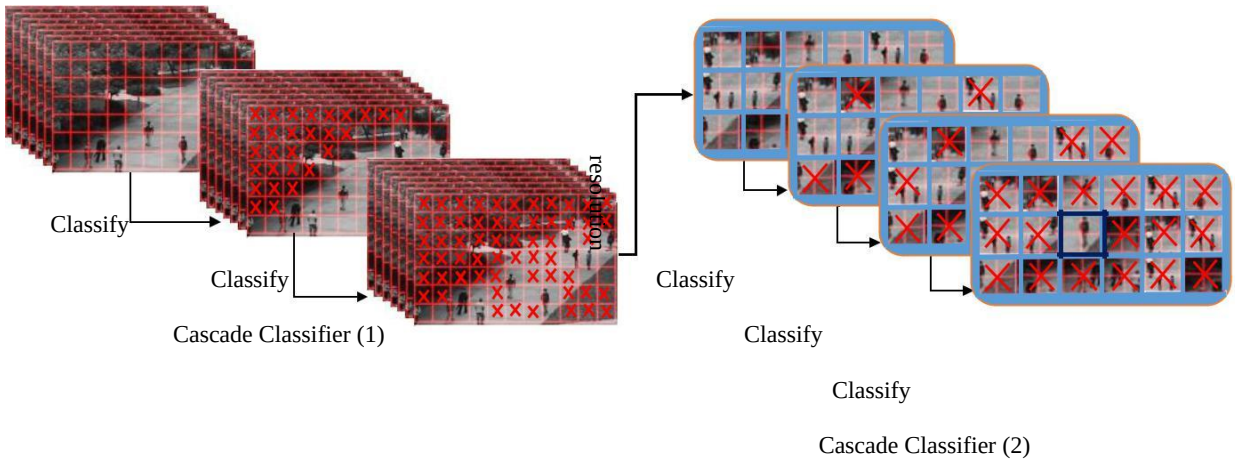

Fig. 1. Our cascade for anomaly detection. Top: Processing of small patches for early rejection of normal patches. Bottom: Resized remaining patches and further rejection of normal patches

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], [12], [13], [14],[15], [16], [17], [18], [19]. An object was labeled as being ananomaly if it does not follow the learned normal trajectories.These methods cannot handle the occlusion problem, and theyare 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.

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 dis-tinguishing 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 proba-bilistic 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, unexpect-edness, 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.

[28]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 con-text. 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 tar-get (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], [36], [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 N, and a cubic patch is then defined by K windows of size M 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 10 5 (i. e. without any overlap).

For example, a 250x250 frame converts into 625 cubic patches, and a short video of 200 frames into 625 200 = 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 40 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 speed. Both stages contain a DNN for representing the video. Sabokrou et al. and Xu et al. [46], [42] show that feature learning (by an auto-encoder) performs as well as state-of-the-art methods, even more time-efficient.

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 _ 18 _ 8 features in 10 different maps (i.e. each patch is represented by a feature block of size 18_18_8_10). This means that these 10 maps are considered for classification, consequently every patch is represented by 18_18_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 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].

Cascaded stacked auto-encoder: A stacked auto-encoder is a popular method for unsupervised feature learning. In this paper, we use a stacked auto-encoder with two layers. It works as a fully connected neural network. Training a deep autoencoder 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_10_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.

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 S1; _ _ _ ; S4.The CNN is considered to be a cascade which consists of the set of sub-stages S1; _ _ _ ; S4. 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

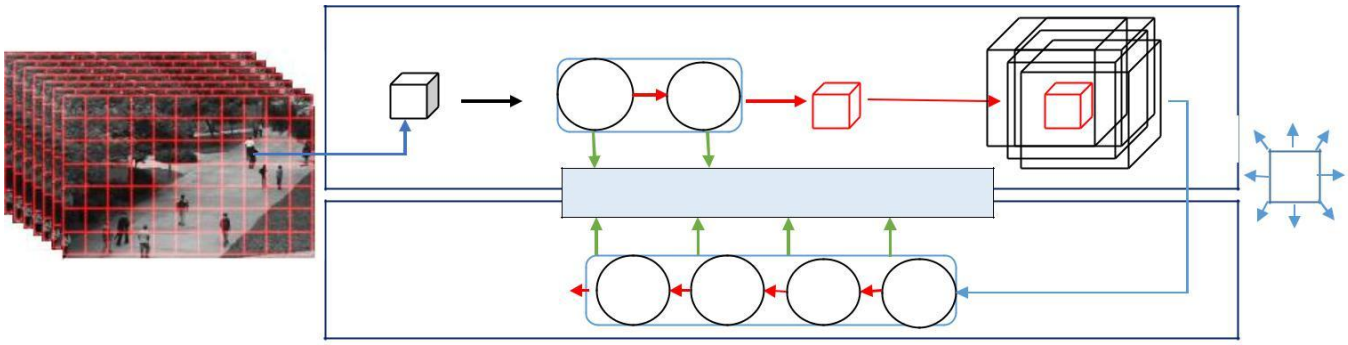

Fig. 2. Work-flow of the proposed system. Top: Stage 1 has a cascaded stack-auto-encoder with two (S_1^1 and S_1^2) layers as sub-stages. Bottom: Stage 2 has a cascaded CNN with four sub-stages (S_2^1, S_2^2, S_2^3 and S_2^4).

KL(j ^{0}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 distri-bution with as its parameter and the active node distribution. The parameter is the weight of the penalty term in the sparse auto-encoder objective. We can efficiently optimize the above objective with respect to 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 5 5 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 5 5), 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

$(W1^{1:D=5\ 5\ 5;i})$i=1:10; see Fig. 5.

We apply these kernels on the video patches and the 3D max pooling operation is performed on 2 2 2 regions afterwards. At top of the first sub-stage, we have a feature map of size 18 18 8 10. Then we have the second convolutional layer which is fed by output .Thesecond 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 7 2 100 is extracted.

In the third sub-stage we apply a convolution with 10 kernels of size 4 4 2 which are learned like at previous sub-stages, at the top of third sub-stage, we have a 4 4 1 1000 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

In the i^{th} sub-stage (except the 4^{th} which is adjusted for the final detection result), we aim at detecting (1) 100% of patches as normal patches, and the remaining pass to the next layers (stages). Thus, the Si stage, i > 0, just exploits _i□1 _ 100% of all patches, instead of N = 100%. Consequently, the time complexity of a CNN is equal to N _
P4 i=1 Ki, and the time complexity of a cascaded CNN equals
CcasCNN = N _ X4 i=1 _i□1Ki (6)

Thus, the cascaded CNN speed-up is equal to the ratio of CCNN to CcasCNN. 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 _ = 0:2, our method is about 2.6 times faster than the CNN.

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.

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 3; 400 and 5; 000, respectively.

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.

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:

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

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.

Dual pixel level (DPL): In this measure, a frame is consid-ered as being an anomaly if (1) it satisfies the anomaly condi-tion at pixel level, and (2) at least 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].

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)

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 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.

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.

TABLE I

SURVIVING SAMPLES VS. CASCADE STAGES

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

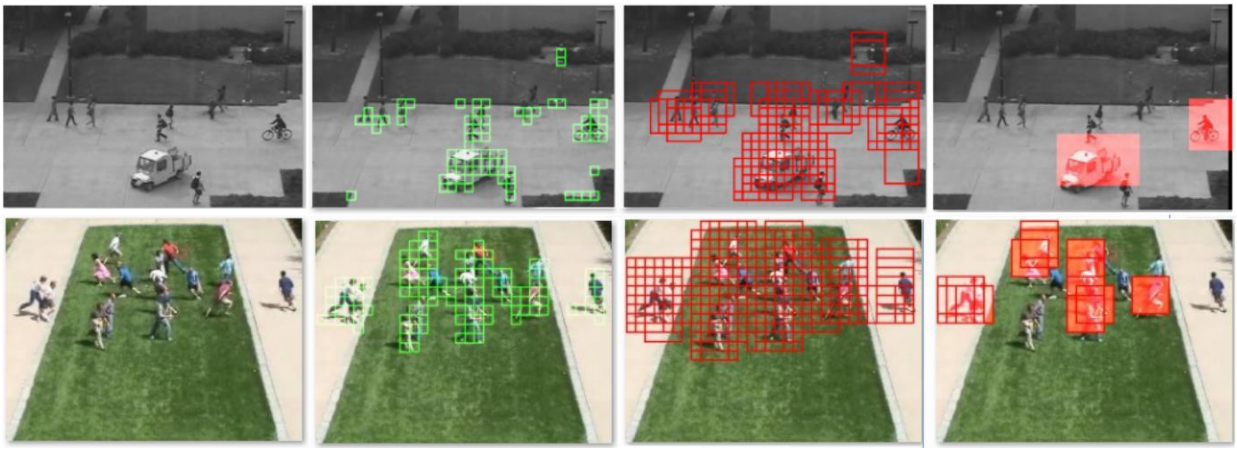

Fig. 10. An example of anomaly detection using our method on Ped2 UCSD and UMN. Left to right: (1) Original frame. (2) Points of interest as detected by a light auto-encoder at Stage 1. (3) Large patches which are extracted from interest-points. For simplifying the figure, instead of showing all 9 patches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

Table III (3rd 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 Tan 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 = 5%.

This DPL measure is defined in [42], and other state-of-the-art methods did not report the localization performance on Ped2 (for = 5%) with respect to this measure. Consequently we just compare our method with [42]. The dual-pixel EER (for = 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].

Module 1: Frame Conversion

- First create a Video Reader object in order to read the video.
- Converted that frame into gray image and applied a Gaussian filter to smooth.

Module 2:optical flow

- The propose Two-Frame Motion Estimation algorithm Based on Polynomial Expansion.
- In the first step, it uses polynomial expansion transform to approximate each neighborhood of both frames by quadratic polynomials.
- After a series of refinements, he estimates displacement fields form polynomial expansion coefficients.
- The resulting flow object has two properties: angle and magnitude.
- The important property for us is the magnitude property since we care how much the optical flow has changed between frames in order to detect the alarm situation.
- The more magnitude is the more possibility of detecting alarm situation.
- Less magnitude means actually the crowd is walking in slow pace. After getting the magnitude part of the flow object, threshold the magnitude with >0.3 in order to get rid of noise.

Module 3: Temporal Occupancy Variation

- Create a 3D matrix whose z-axis has length of 30 and represents each frame's magnitude image. use the matrix that has 30 frame in its z-axis because it is said that the dataset is captured by cameras that are able to take 30 frames per second.
- This means that hold 1 second of the video sequence in our 3D matrix.
- TOV frame only shows the different pixels between the consecutive activity maps which means we can now 'see the motion'.
- The entropy can also help us to estimate how big our motion in our difference frame.

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 anoma-lies. 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 the 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 CVPR, pages 1650–1657, 2013.

[2] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In Advances Neural Informa-tion Processing Systems, pages 1097–1105, 2012.

[3] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierar-chies for accurate object detection and semantic segmentation, In CVPR, pages 580–587, 2014.

[4] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In Advances Neural Information Processing Systems, pages 68–576, 2014.

[5] A. Giusti, D.C. Ciresan, J. Masci, L.M. Gambardella, and J. Schmidhu-ber. Fast image scanning with deep max-pooling convolutional neural networks. In ICIP, pages 4034–4038, 2013.

[6] R. Klette. Concise Computer Vision. Springer, London, 2014.

[7] P. Luo, Y. Tian, X. Wang, and X. Tang. Switchable deep network for pedestrian detection. In CVPR, pages 899–906, 2014.

[8] A. Angelova, A. Krizhevsky, V. Vanhoucke, A. Ogale, and D. Ferguson. Real-time pedestrian detection with deep network cascades. In BMVC, 2015.

[9] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In ICCV, pages 2056–2063, 2013.

[10] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua. A convolutional neural network cascade for face detection. In CVPR, pages 5325–5334, 2015.

[10] F. Jianga, J. Yuan, S. A. Tsaftarisa, and A. K. Katsaggelosa. Anomalous video event detection using spatiotemporal context. Computer Vision Image Understanding, 115(3):323–333, 2011.

[11] S. Wu, B. Moore, and M. Shah. Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes. In CVPR, 2010.

[12] C. Piciarelli, C. Micheloni, and G.L. Foresti. Trajectory-based anoma-lous event detection. IEEE Trans. Circuits Systems Video Technology, 18(11):1544–1554, 2008.

[13] C. Piciarelli and G.L. Foresti. On-line trajectory clustering for anoma-lous events detection. Pattern Recognition Letters, 27(15):1835–1842, 2006.

[14] F. Tung, J.S. Zelek, and D.A. Clausi. Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance. Image and Vision Computing, 29(4):230–140, 2011.

[15] W. Hu, X. Xiao, Z. Fu, D. Xie, T. Tan, and S. Maybank. A system for learning statistical motion patterns. IEEE Trans. Pattern Analysis Machine Intelligence, 28(9):1450–1464, 2006.

[16] B.T. Morris and M.M. Trivedi. Trajectory learning for activity under-standing: Unsupervised, multilevel, and long-term adaptive approach. IEEE Trans. Pattern Analysis Machine Intelligence, 33(11):2287–2301, 2011.

[17] S. Calderara, U. Heinemann, A. Prati, R. Cucchiara, and N. Tishby. De-tecting anomalies in people's trajectories using spectral graph analysis. Computer Vision Image Understanding, 115(8):1099–1111, 2011.

[18] P. Antonakaki, D. Kosmopoulos, and S.J. Perantonis. Detecting ab-normal human behaviour using multiple cameras. Signal Processing, 89(9):1723–1738, 2009.

[19] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos. Anomaly detection in crowded scenes. In CVPR, pages 1975–1981, 2010.

[20] www.svcl.ucsd.edu/projects/anomaly/dataset.html

[21] mha.cs.umn.edu/Movies/Crowd-Activity-All.avi

[22] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz. Robust real-time unusual event detection using multiple fixed location monitors. IEEE Trans. Pattern Analysis Machine Intelligence, 30(3):555–560, 2008.