Video Anomaly Detection With Sparse Coding Inspired Deep Neural Networks

Weixin Luo⋆, Wen Liu⋆, Dongze Lian, Jinhui Tang, Lixin Duan, Xi Peng, and Shenghua Gao

Abstract—This paper presents an anomaly detection method that is based on a sparse coding inspired Deep Neural Networks (DNN). Specifically, in light of the success of sparse coding based anomaly detection, we propose a Temporally-coherent Sparse Coding (TSC), where a temporally-coherent term is used to preserve the similarity between two similar frames. The optimization of sparse coefficients in TSC with the Sequential Iterative Soft-Thresholding Algorithm (SIATA) is equivalent to a special stacked Recurrent Neural Networks (sRNN) architecture. Further, to reduce the computational cost in alternatively updating the dictionary and sparse coefficients in TSC optimization and to alleviate hyperparameters selection in TSC, we stack one more layer on top of the TSC-inspired sRNN to reconstruct the inputs, and arrive at an sRNN-AE. We further improve sRNN-AE in the following aspects: i) rather than using a predefined similarity measurement between two frames, we propose to learn a data-dependent similarity measurement between neighboring frames in sRNN-AE to make it more suitable for anomaly detection; ii) to reduce computational costs in the inference stage, we reduce the depth of the sRNN in sRNN-AE and, consequently, our framework achieves real-time anomaly detection; iii) to improve computational efficiency, we conduct temporal pooling over the appearance features of several consecutive frames for summarizing information temporally, then we feed appearance features and temporally summarized features into a separate sRNN-AE for more robust anomaly detection. To facilitate anomaly detection evaluation, we also build a large-scale anomaly detection dataset which is even larger than the summation of all existing datasets for anomaly detection in terms of both the volume of data and the diversity of scenes. Extensive experiments on both a toy dataset under controlled settings and real datasets demonstrate that our method significantly outperforms existing methods, which validates the effectiveness of our sRNN-AE method for anomaly detection. Codes and data have been released at https://github.com/StevenLiuWen/sRNN_TSC_Anomaly_Detection.

Index Terms—Sparse Coding, Anomaly Detection, Stacked Recurrent Neural Networks.

1 Introduction

Anomaly detection is an important task in computer vision, and it has many potential applications in video surveillance, activity recognition and scene understanding, *etc*. However, anomaly detection is also an extremely challenging task.[1] Because of the unbounded and rare nature of anomalies, it is extremely expensive and sometimes infeasible to collect different types of abnormal events. For example, spontaneous car combustion is rare, and it is difficult to collect or simulate this kind of anomaly. Consequently, it seems infeasible to formulate anomaly detection with a binary classification framework because if some types of abnormal events are not included in the training set, the test phase may misclassify these kinds of anomalies. Further, considering the rare and unbounded nature of anomaly detection as well as to simplify the data collection procedure, only normal data is given in the training set in the common setup, with anomaly detection here aiming at discovering abnormal events in the test set.

To tackle anomaly detection when only normal data is given, an intuitive approach is to model the distribution of regular patterns, where data that does not agree with the distribution of regular patterns are classified as irregular. Recently, with the success of Convolutional Neural Networks, people leverage deep Convolutional Auto-Encoder [1] or Convolutional LSTM Auto-Encoder [2] to model the normal distribution in the training set, and irregular patterns will be distinguished by large reconstruction errors. These deep learning solutions are very efficient in the testing phase, but they rely on some delicately designed deep neural network architectures, and the principles for network design are still not well formulated. In addition, dictionary learning based approaches [3] [4], especially sparse coding based approaches, have been proposed and have shown their expertise in tackling such a task. Specifically, sparse coding based approaches encode regular patterns with a dictionary. Regular patterns can be linearly reconstructed by the entries in the dictionary with small reconstruction errors. In contrast, irregular patterns would lead to large reconstruction errors. However, the dictionary learning procedure during training is very time consuming for sparse coding based anomaly detection, and the optimization of sparse coefficients in the test phase is also very time-consuming, which restricts the deployment of these methods in real applications. Further, frame-wise sparse coding does not consider the coherence among neighboring frames for normal events.

Recently, Wisdom *et al.* [5] have shown that actually the optimization of sparse coding with Iterative Soft-Thresholding Algorithm (ISTA) is essentially a special type of deep neural network. Motivated by the success of sparse coding based anomaly

W. Luo and W. Liu are the co-first authors. S.Gao is the corresponding author.

- *W. Luo, W. Liu, D. Lian are with School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China, and Chinese Academy of Sciences, Shanghai Institute of Microsystem and Information Technology, Shanghai 200050, China, and University of Chinese Academy of Sciences, China.*
- *Jinhui Tang is with Nanjing University of Science and Engineering, Nanjing, China.*
- *Lixin Duan is with University of Electronic Science and Technology of China, Chengdu, China.*
- *Xi Peng is with Sichuan University, Chengdu, China.*
- *S. Gao is with School of Information Science and Technology, ShanghaiTech University, Shanghai 201210, China.*

1. This paper focuses on frame-level anomaly detection, and such frame-level prediction can meet the requirement of video surveillance.

detection and the interpretation of ISTA with deep learning, we propose a sparse coding inspired Deep Neural Networks (DNN) framework for anomaly detection. Specifically, we add a temporally-coherent term into the sparse coding objective that utilizes the similarity between neighboring frames to weight the distance between their corresponding sparse coefficients. Then we arrive at a Temporally-coherent Sparse Coding (TSC). The optimization of sparse coefficients in TSC with Sequence Iterative Soft-Thresholding Algorithm (SISTA) [5] in essence is a special type of stacked Recurrent Neural Networks (sRNN). To reduce the expensive computational costs in alternatively optimizing the dictionary and sparse coefficients, as well as to avoid the hyperparameters selection in TSC, we propose stacking one more layer on top of the TSC counterpart sRNN, arriving at an sRNN Auto-Encoder (sRNN-AE). With this sRNN-AE, the dictionary, the reconstruction coefficients and all of the hyperparameters can be automatically learned. In addition, the testing phase for each video frame is equivalent to a forward pass in this sRNN-AE. Further, to reduce computational costs in the test phase, we reduce the depth of the sRNN in sRNN-AE. Such a shallow architecture not only alleviates the gradient vanishing/exploding effect in the optimization of the DNN, thus improving the performance of sRNN-AE, but also improves efficiency in the test phase. Another advantage of sRNN-AE is that rather than using a predefined similarity measurement between neighboring frames, motivated by the kernel trick [6], we propose mapping the features of two frames to a new feature space with multi-layer perception and using the inner product of the new features as a similarity measurement. Experiments validate the effectiveness of such a similarity learning module in sRNN-AE.

Our sRNN-AE works as a classifier and its architecture is inspired by the ISTA based optimization of sparse coding. Besides a good classifier, a discriminative video representation is also desired for the good performance of anomaly detection. In real scenes, anomalies can be caused by unseen objects, namely appearance, unusual moving patterns, namely motion. Therefore, we propose extracting both appearance and motion features as the input of sRNN-AE. Inspired by the success of two-stream CNNs for video representation in activity recognition [7], in this paper, we propose learning a separate sRNN-AE with different features for anomaly detection. In two-stream CNNs, the stream corresponding to motion takes optical flow as input, where the calculation of optical flow is time-consuming. Considering that the difference among neighboring frames also characterizes the motion of objects [8], we propose conducting temporal pooling over appearance features of several consecutive frames for summarizing information temporally. Such a strategy would reduce the costs in calculating optical flow and motion features extraction with optical flow based CNN. Experiments validate the effectiveness and efficiency of this type of feature-aggregation based temporal representation.

It is desirable to learn an anomaly detection model which works well under multiple scenes. However, almost all existing datasets only contain videos captured by one camera with a fixed view, so these datasets lack scene diversity. Further, a large-scale dataset is in high demand for the evaluation of deep learning based anomaly detection approaches. In this paper, we build a new large-scale anomaly detection dataset. We set up multiple cameras with different view angles to capture real events in the teaching, research and living areas of our campus, and we name our new dataset the ShanghaiTech Campus anomaly detection dataset. To the best of our knowledge, our dataset is the largest one in terms of volume of frames, scene diversity, as well as viewing angles.

Contribution: We summarize our contributions of this work as follows: i) We design a sparse coding inspired sRNN-AE framework for anomaly detection, which alleviates the hyperparameters selection and dictionary training in TSC. Further, similarity can also be automatically learned in sRNN-AE; ii)we propose an appearance features based temporal characterization strategy. Then we propose learning a separate sRNN-AE for both spatial and temporal features for anomaly detection; iii) we collect a large-scale anomaly detection dataset, which greatly facilitates the evaluation of anomaly detection algorithms.

This paper is an extension of our previous work [9]. We extend the framework in the following aspects: i) motivated by the kernel trick, we introduce a similarity learning module in sRNN-AE, which demonstrates its effectiveness over a predefined similarity for anomaly detection; ii) we propose an appearance features based temporal characterization strategy, and propose learning a separate sRNN-AE using both spatial and temporal features for anomaly detection; iii) more details of our implementation are given, and more experiments are conducted for performance evaluation.

The rest of this paper is organized as follows: In Section 2, we introduce work related to anomaly detection. In Section 3, we first briefly revisit sparse coding based anomaly detection and introduce the TSC formulation. Based on the optimization of TSC, we arrive at an sRNN based framework for anomaly detection. In Section 4, extensive experiments under both controlled and uncontrolled settings are conducted to validate the effectiveness of our work. We also evaluate the different components of our sRNN-AE algorithm with an ablation study in this section. We conclude our work in Section 5.

2 Related work

Most existing work on anomaly detection can be categorized into two steps: i) Feature extraction; One can leverage hand-crafted or deep learning based features. ii) Normal distribution learning; In this phase, a distribution is learned over the normal data of the training set, so that abnormal data of the test set will have a large reconstruction error over this distribution.

Hand Craft Feature and Distribution Modeling. Early work utilizes low-level trajectory features to represent regular patterns [10]. However, these methods are not robust in complex or crowded scenes. In order to solve this problem, spatial-temporal features, such as histograms of oriented gradients (HOG) [11] and histograms of oriented flows (HOF) [12] have been widely leveraged. Based on these spatial-temporal features, Zhang *et al.*. [13] model the normal patterns with a Markov random field (MRF). Adam et al. [14] fit the regular histograms of optical flow in local regions with an exponential distribution. To represent local optical flow patterns, Kim and Grauman [15] utilize a mixture of probabilistic PCA model. Leyva *et al.* [16] propose an online framework by leveraging Gaussian Mixture Models, Markov Chains, and Bag-of-Words for anomaly detection.

Sparse Coding Based Anomaly Prediction. Dictionary learning based approaches are widely used in anomaly detection [4] [3] [17] [18]. A fundamental assumption of these methods is that any feature can be linearly represented as a linear combination of the bases of a dictionary that encodes regular patterns of the training set. [4] [3] [17] use the reconstruction error to determine whether a frame is abnormal or not. Ren *et al.* [18] point out that reconstruction errors, such as those in least squares, do not take a sparsity term into consideration, while in fact, they do help to improve anomaly detection accuracy. To avoid this, Ren *et al.* [18] propose two solutions, *i.e.* maximum coordinate (MC) and non-zero concentration (NC), to detect anomalies. However, sparse reconstruction based methods are usually time-consuming during the optimization of sparse coefficients. To solve this problem, Jia *et al.* [3] propose to discard the sparse constraint and learn multiple dictionaries to encode the patches at multiple scales, which inevitably leads to additional costs in the training phase.

Deep Learning Based Anomaly Detection. Deep learning approaches have demonstrated their success for image classification [19] [20], object detection [21] [22], as well as anomaly detection [1] [23]. In [1], Hasan *et al.*propose a 2D convolutional Auto-Encoder (Conv-AE) by stacking frames in channels to model regular frames. Such a 2D convolution, however, cannot characterize spatial and temporal information very well, as shown in activity recognition [24]. In light of the capabilities of convolutional neural networks (ConvNets) to represent spatial features and the strong capabilities of recurrent neural networks (RNN) and long short term memory (LSTM) to model temporal patterns, [25] [26] [2] make attempts to leverage a convolutional LSTM Auto-Encoder (ConvLSTM-AE) to characterize both appearance and motion information. Ryota *et al.* [27] combine both detection and the recounting of abnormal events. Sabokrou *et al.* [28] leverage a pre-trained Fully Convolutional Neural Networks (FCNs) to training an unsupervised FCN for anomaly detection. Sultani *et al.* [29] apply multiple instance learning (MIL) for anomaly detection, but in their setting both normal and abnormal videos are equally provided in the training set, and in many scenarios the acquisition of different types of abnormal data is very expensive and even infeasible.

Even though Auto-Encoder based methods have shown some good performance for anomaly detection, they may be prone to learn an identity mapping, and fail to detect the abnormal events. In order to prevent learning a trivial solution, a generative model can be utilized to model a normal distribution. In [30], Thomas *et al.*apply a GAN [31] model to detect anomalies in medical images. More specifically, they first train a generator from the latent space to the image space, by fooling a discriminator. Once the generator and discriminator are trained using the training set only with normal data, all of their parameters are fixed. Further, for a query sample, they leverage gradient descent to search for the optimal latent variable to reconstruct the sample. Normal samples would cause a low reconstruction residual error while abnormal ones would cause a higher value. In addition, pixel errors between the query and reconstruction indicate lesions.

Although RNNs or LSTMs are powerful and effective for processing sequential data, they are actually ”black boxes” whose internal structures are hard to interpret. Recently, Scott *et al.* [5] show that a special type of RNN actually enforces a sparse constraint on features. Inspired by the work of sparse coding based anomaly detection and interpretable RNNs, we propose a TSC and its sRNN-AE counterpart for anomaly detection.

Methods without a Training Phase. Except for those methods mentioned above, there are also others methods without a training phase for anomaly detection. In [32], Giomo *et al.*propose the direct estimation of the discriminability of frames with references to the context in the test video, without any training set. In addition to that, a similar setting is adopted in [33], which trains a binary classifier to distinguish between two consecutive video sequences while removing the most discriminant features at each step. The higher training accuracy rates of the intermediately obtained classifiers represent abnormal events.

3 Our Approach

In this section, we first revisit sparse coding based anomaly detection. To model the coherence between neighboring frames for normal events, Temporally-coherent Sparse Coding (TSC) is introduced, then we show that the optimization of TSC with the Sequential Iterative Soft-Thresholding Algorithm (SISTA) is equivalent to a special type of stacked Recurrent Neural Networks (sRNN). Further, to reduce the time cost in training and inference stage as well as alleviate the hyperparameter selection in TSC, we reduce the number of layers in sRNN and stack one more layer on top of sRNN to reconstruct the input, which arrives at an sRNN-AE. We also propose to learn similarity with a multi-layer perceptron within the sRNN-AE framework, which further improves anomaly detection accuracy. Finally, we will show how to combine spatial and temporal features for real time anomaly detection.

3.1 A Revisit of Sparse Coding Based Anomaly Detection

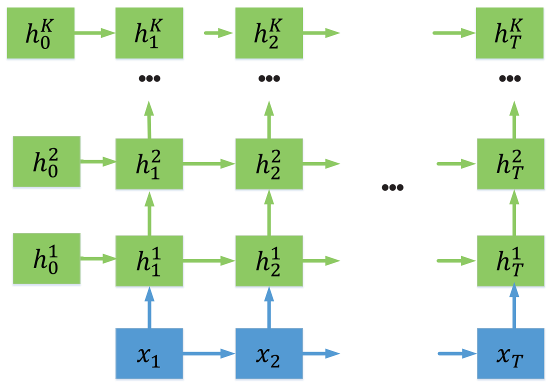
Sparse coding based anomaly detection aims to learn a dictionary to encode all normal events with small reconstruction errors [4] [3]. Mathematically, we denote a feature corresponding to a normal input as x_i, then it is desirable that x_i can be linearly reconstructed by a dictionary A with a small reconstruction error ϵ_i, *i.e.*, $x_i = A\alpha_i + \epsilon_i$. Under the assumption that $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I)$, and $\alpha_i \sim \text{Laplace}(0, 2\sigma^2/\lambda)$, we arrive at the following objective function:

$$\min_{A,\alpha_i} \frac{1}{2}\|x_i - A\alpha_i\|_2^2 + \lambda\|\alpha_i\|_1 \tag{1}$$

In this formulation, the first term corresponds to a reconstruction error, where it measures how well the feature can be reconstructed by the dictionary. The second term corresponds to a sparsity term while λ balances the sparsity and the reconstruction error. A larger λ corresponds to an even more sparse solution. To avoid trivial solutions to the problem, usually an L2 norm constraint is imposed on each column of A: $\|A(:,j)\| \leq 1$. By alternatively optimizing the dictionary and the sparse coefficients on the training set [4], a dictionary can be learned that encodes all normal patterns. In the test phase, when a feature comes in, we first compute its sparse coefficients based on the dictionary A. Then, based on its reconstruction error, we can classify whether it belongs to normal or abnormal events.

3.2 Temporally-coherent Sparse Coding (TSC) for Anomaly Detection

One advantage of sparse coding based anomaly detection is that it learns a dictionary to encode all normal events with small reconstruction errors, thus an abnormal event is associated with a large reconstruction error. It does not consider, however, the temporal coherence between neighboring frames within normal/abnormal events. Further, as shown in previous works [3] [34], with sparse coding, similar features may be encoded as dissimilar sparse codes, *i.e.*, locality information is lost. To preserve the similarity


(a) Vanilla stacked RNN

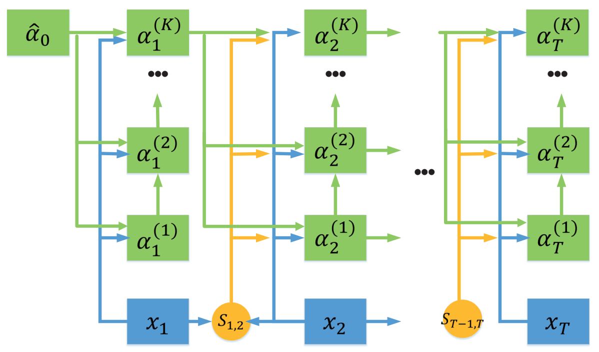

(b) Stacked RNN counterpart of TSC

Fig. 1. The blue boxes represent the input x_t. The green boxes represent hidden states h_t^k or coding vectors α_t^k. The orange circles are similarities between neighboring frames.

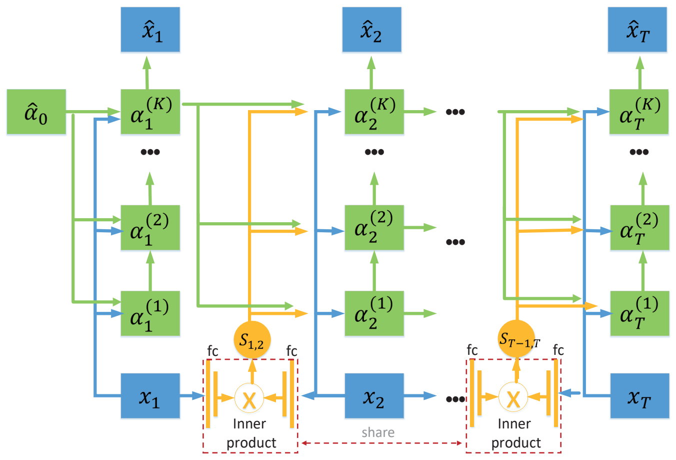


Fig. 2. sRNN-AE with trainable similarity measurement

between neighboring frames, and as motivated by the work of [4], we propose a Temporally-coherent Sparse Coding (TSC) model. Specifically, if two neighboring frames are similar, it is desirable that their sparse coefficients are similar as well. To achieve this goal, we use the similarity between neighboring frames to weight the distance between their sparse coefficients. We denote the similarity between the t-th frame and $(t-1)$-th frame as $S_{t-1,t}$, which can be either predefined or learned with a data-driven approach. Then we use $S_{t-1,t}$ to weight $\|\alpha_t - \alpha_{t-1}\|_2^2$ and substitutes the temporally coherent constraint into the sparse coding objective function, which gives the objective function of TSC:

$$\min_{A,\alpha_t} \sum_{t=1}^{T} \|x_t - A\alpha_t\|_2^2 + \lambda_1\|\alpha_t\|_1 + \lambda_2 S_{t,t-1}\|\alpha_t - \alpha_{t-1}\|_2^2 \quad \text{s.t.} \quad \|A(:,i)\| \le 1 \tag{2}$$

This objective 2 is not convex. Following the classical optimization strategy in sparse coding [35] [36], we can alternatively update A and α_t $(t = \{1,\dots,T\})$.

Optimization of A**.** When all α_t $(t = \{1,\dots,T\})$ are fixed, the objective function corresponding to A can be written as follows:

$$\min_{A} \sum_{t=1}^{T} \|x_t - A\alpha_t\|_2^2 \quad \text{s.t.} \ \|A(:,i)\| \le 1 \tag{3}$$

Then, we use a projected gradient descent algorithm to optimize A.

Optimization of α_t**.** When A is fixed, we arrive at the following objective function w.r.t. reconstruction coefficients of all features:

$$\min_{\alpha_t} \sum_{t=1}^{T} \|x_t - A\alpha_t\|_2^2 + \lambda_1\|\alpha_t\|_1 + \lambda_2 S_{t,t-1}\|\alpha_t - \alpha_{t-1}\|_2^2 \tag{4}$$

After that, we update α_t $(t = \{1,\dots,T\})$ with a Sequential Iterative Soft-Thresholding Algorithm(SISTA) [5] whose main steps are algorithm 1. In this algorithm, $\text{soft}_b(x) = \max(x-b, 0) = \text{ReLU}(x-b)$, K corresponds to the steps of the ISTA algorithm. γ is a hyperparameter.

Algorithm 1 Sequential iterative soft-thresholding algorithm.

```
Input: extracted feature $x_{1:T}$, hyper-parameter $\lambda_1, \lambda_2, \gamma$, initial
    $\hat{\alpha}_0$, the steps of ISTA $K$
1: for $t = 1$ to $T$ do
2:     $\hat{\alpha}_t^0 = \alpha_{t-1}$
3:     for $k = 1$ to $K$ do
4:         $z = [I - \frac{1}{\gamma}(A^T A + S_{t-1,t}\lambda_2 I)]\hat{\alpha}_t^{k-1} + \frac{1}{\gamma}A^T x_t$
5:         $\hat{\alpha}_t^{(k)} = \text{soft}_{\lambda_1/\gamma}(z + \frac{S_{t-1,t}\lambda_2}{\gamma}\alpha_{t-1})$
6:     end for
7:     $\alpha_t = \hat{\alpha}_t^K$
8: end for
9: return $\alpha_{1:T}$;
```

3.3 Interpreting TSC with a Stacked RNN (sRNN)

A traditional RNN is based on the assumption that $h_t = f(x_t, h_{t-1})$, which introduces a recurrent structure. Many previous work [37] [38] shows that by stacking multiple RNNs on top of each other, the performance of classification or regression can be further boosted. We denote x_t as an input at time t and denote h_t^k as an output of hidden nodes in the k-th layer at time t. σ_b is the nonlinear activation function parameterized by b. In this paper, we choose $\sigma_b(x) = \text{soft}_b(x)$. Mathematically, the stacked RNN (sRNN) can be written as follows:

$$h_t^{(k)} = \begin{cases} \sigma_b(W^{(1)} h_{t-1}^{(1)} + V x_t), & k = 1, \\ \sigma_b(W^{(k)} h_{t-1}^{(k)} + U^{(k)} h_t^{(k-1)}), & k > 1. \end{cases} \quad (5)$$

The first layer accepts the last moment output at the same layer h_{t-1}^1 and the current moment input x_t as its inputs. Similarly, the rest of the stacked layers accept the last moment output h_{t-1}^k at the same layer and the previous layer output h_t^{k-1} at the same moment as their inputs.

It should be noticed that the sRNN mapped from TSC is slightly different from that of the formulation in (5). For the stacked layers, slightly different from Equation (6), they also accept the current moment input x_t as its inputs.

$$h_t^{(k)} = \sigma_b(W^{(k)} h_{t-1}^K + U^{(k)} h_t^{(k-1)} + V x_t), k > 1. \quad (6)$$

By comparing the optimization procedure in Algorithm 1 with the stacked RNN, we can see that Equation (2) can be interpreted with an sRNN: The K steps in the Sequential Iterative Soft-Thresholding Algorithm correspond to the number of layers in the sRNN. Comparing the proposed sRNN to classical RNN [37], the difference between them is that x_t is fed into all sRNN layers in our sRNN, while the vanilla RNN only takes x_t as its input in the first layer. Further, $S_{t,t-1}$ takes x_t and x_{t-1} as inputs, which means that h_t^k also depends on the input of the last moment x_{t-1}. Additionally, $S_{t,t-1}$ is the input of each hidden state h_t^k. We illustrate the stacked RNN in our problem in Figure 1.

More specifically, the mapping from the variables in TSC to the variables in sRNN in Equation (5) is:

$$\begin{aligned} W^{(1)} &= I - \frac{\lambda_2}{\gamma} A^T A \quad (7) \\ W^{(k)} &= \frac{S_{t-1,t}\lambda_2}{\gamma} I, k > 1 \\ U^{(k)} &= I - \frac{1}{\gamma}(A^T A + S_{t-1,t}\lambda_2 I), k > 1 \\ V &= \frac{1}{\gamma} A^T \\ b &= \lambda_1 / \gamma \\ h_t^{(k)} &= \alpha_t^k \end{aligned}$$

To demonstrate the mapping, we copy line 4 and 5 in Algorithm 1 here and denote each replacement under each component in Equation (8).

$$z = \underbrace{[I - \frac{1}{\gamma}(A^T A + S_{t-1,t}\lambda_2 I)]}_{U^{(k)}} \underbrace{\hat{\alpha}_t^{k-1}}_{h_t^{(k-1)}} + \underbrace{\frac{1}{\gamma} A^T}_{V} x_t \quad (8)$$

$$\hat{\alpha}_t^{(k)} = \underbrace{\text{soft}}_{\sigma} \underbrace{\lambda_1/\gamma}_{b} (z + \underbrace{\frac{S_{t-1,t}\lambda_2}{\gamma}}_{W^{(k)}} \underbrace{\alpha_{t-1}}_{h_{t-1}^K})$$

3.4 sRNN Auto-Encoder

First, as shown in Figure 7, TSC is sensitive to the weight of the sparsity term and is a temporally-coherent term. In addition to that, different datasets prefer different parameters. Therefore, it is desirable to derive a data-dependent way to automatically learn these parameters. Second, the training of TSC is done by the alternative optimization of the dictionary and the sparse coefficients, which is also time-consuming, while it is observable that dictionary learning is equivalent to learning the sRNN. Thirdly, if the number of layers in sRNN (K) is very high, our network is identical with TSC, which guarantees that all α_t's are sparse. A very deep sRNN, however, is very time-consuming in the inference stage. To tackle these problems, we first reduce the number of layers in sRNN. Then, we propose the training of the sRNN with an Auto-Encoder (sRNN-AE), *i.e.*, we use the last layer output (h_t^K) of the sRNN to reconstruct the input x_t with the mapping function parameterized by Z, *i.e.*, $\hat{x}_t = Z h_t^K$. We denote the parameters in the sRNN as $\theta = \{A, \lambda_1, \lambda_2, Z, \alpha_0, \gamma\}$. Finally, we can simultaneously optimize all parameters including the dictionary, hyperparameters and reconstruction coefficients in the following way:

$$\min_\theta \sum_{t=1}^T \|x_t - Z h_t^K\|_F^2 + \beta \|\theta\|_F^2 \quad (9)$$

To solve Equation (9), we use a min-batch based Stochastic Gradient Descent (SGD) algorithm. Specifically, we use the RMSPROP [39] based SGD method, and set the weight for the weight decay term as $\beta = 0.005$. Further, a larger K will inevitably introduce a higher computational cost. Therefore, rather than using a very large K, we use a small one (K=3). As shown in the experiments section, such a shallow architecture achieves much better performance than all other existing methods. Our sRNN has two advantages: i) we can learn all of the parameters in the sRNN rather than choosing the hyperparameters in TSC; ii) the architecture of our sRNN is not deep. In the test phase, we can get $\alpha_t = h_t^K$ in one forward pass, which greatly accelerates anomaly detection.

3.5 Similarity Measurement

The similarity measurement is a key factor for the performance of TSC and sRNN-AE. One simple way to obtain it is to directly define the similarity between neighboring frames with some commonly used functions, such as the Gaussian function, which is defined as follows: [2]

$$S_{t-1,t} = \exp(-\frac{\|x_t - x_{t-1}\|_2^2}{\delta^2}) \quad (10)$$

It is desirable, however, to learn a data-driven similarity measurement for different data.

Inspired by the kernel trick of SVM [6], we can define the similarity as follows:

$$S_{t-1,t} = \kappa(x_t, x_{t-1}) = \phi(x_t)^T \phi(x_{t-1}) \quad (11)$$

Here $\kappa(\cdot,\cdot)$ is a kernel function, and $\phi(\cdot)$ is some mapping function, usually unknown. In this paper, we leverage a data-driven approach to learn the mapping function $\phi(\cdot)$ within sRNN-AE. Specifically, we leverage a multi-layer perceptron with the ReLU activation function as the $\phi(\cdot)$ function. We denote the

2. $\delta^2 = 100$ in our experiments. It is worth mentioning that since $S_{t-1,t}$ is multiplied by λ_2, thus we can set δ to any value and tune λ_2 accordingly.

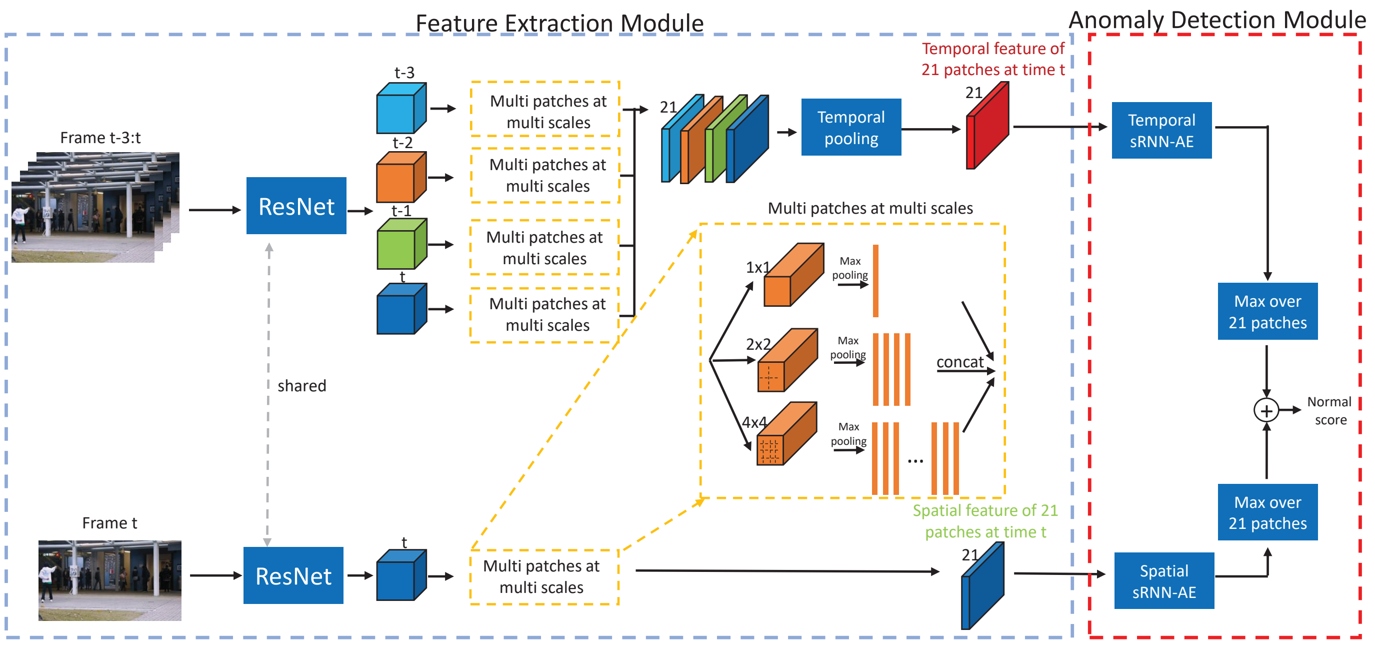


Fig. 3. The whole pipeline of our proposed anomaly detection. It consists of a feature extraction module and an anomaly detection module.

parameters of $\phi(\cdot)$ as w_ϕ and put it into the trainable parameter set of sRNN-AE. In this way, the mapping function $\phi(\cdot)$ can be automatically learned with other parameters in sRNN-AE in an end-to-end learning manner. We also normalize the output of the multi-layer perceptron with an l_2 normalization to make the length of the output be 1. In this way, the similarity $S_{t-1,t} \in [0, 1]$. In the following sections, without specification, the similarity in sRNN-AE is trained in this way, as shown in Figure 2.

3.6 The Combination of Spatial and Temporal Features for Anomaly Detection

Both TSc and sRNN-AE are actually used to model the distribution over normal patterns and discover abnormal ones. Hence, they function as classifiers. In addition, their performance is feature dependent. Since anomalies can be caused by unseen or unexpected objects or unusual motion patterns, it is desirable to combine both spatial and temporal features for anomaly detection. In action recognition, the prevalent method is to use two-stream CNNs [7] [40] [41] for explicitly characterizing appearance and motion information, respectively. These works have shown the effectiveness of the two-streams solution over 2D convolution by stacking frames in channels for feature extraction. Therefore, we also propose to extract the spatial and temporal features separately. For appearance features, we use a spatial ConvNet (ResNet) pretrained with the UCF101 dataset [42] to extract appearance features. For motion features, one way is to feed the optical flow features to a temporal ConvNet. However, the extraction of optical flow is time-consuming, which is not desirable. Following the work of [8], we propose to conduct pooling over the appearance features of several consecutive frames and use this as temporal features. But a bit different from [8] where max pooling, sum pooling, histogram of time series gradients pooling are all used. In our experiments, we find that max pooling already corresponds to good performance. Thus we only use max pooling. Further, we use multiple patches at multiple scales for appearance representation, and do the max pooling for features corresponding to patches at different scales. Thus, these pooled features over patches at different scales encode the spatial change of some objects (spatial features) over time, which also gathers information temporally. As shown in Table 7, our solution greatly accelerates anomaly detection, and also improves accuracy.

Sampling multiple patches at multiple scales has been shown to be a very effective way for improving anomaly detection [3]. We also use the same strategy on videos-based anomaly detection. Specifically, for both spatial and temporal features, we gradually partition the feature map over spatial dimensions into increasingly finer regions: 1×1, 2×2, and 4×4. We use max pooling over each sub-region. Thus the feature dimension of all sub-regions are the same. Rather than learning multiple dictionaries for features at different scales [3], which brings additional computational costs, features at all scales share the same dictionary in our method. For features at multiple scales, we only enforce a temporal coherent constraint for features at the same scale and spatial location.

After extracting spatial and temporal features, there are two possible ways to combine them. One way is to directly stack the spatial and temporal features at the same moment and feed them to one sRNN-AE, which is referred to as early fusion. Another way is to feed them to separate sRNN-AE algorithms and combine the outputs of each sRNN-AE for anomaly detection, which is referred to as late fusion. Previous work [43] has shown that late fusion achieves a better performance for video classification, and our experiments also demonstrate a similar phenomenon for anomaly detection, as shown in Table 9.

The whole pipeline of our proposed anomaly detection system is demonstrated in Figure 3. For an input video, we sample 4 continuous frames with an interval of 1. Then, a pretrained ResNet is used to extract spatial and temporal features. We adopt features extracted from different regions and conduct spatial pyramid pooling [44] over them, achieving 21 feature vectors for each frame. Further, we use temporal pooling over these 4 frames for temporal information summarization, where temporal pooling is an element-wise maximum operation. Further, spatial and temporal features go through two separate sRNN-AE algorithms to achieve 21 reconstruction errors for each modality. For each modality, we take the maximum reconstruction error of the 21 patches. Finally, two reconstruction errors are normalized to scores and linearly combined with a weight, as shown in Section 3.7.

3.7 Anomaly Detection on Testing Data

In the training phase, we can learn the dictionary A which encodes normal events well. In the test phase, we feed the feature of each patch corresponding to the t-th frame into our special sRNN. With one forward pass, we can get α_t. We denote the feature of the i-th patch in the t-th frame as $x_{t,i}$, where super-script s and t correspond to spatial and temporal, respectively, then we can calculate the reconstruction error corresponding to the i-th patch in the t-th frame as follows:

$$\begin{aligned} l^s(t,i) &= \|x^s_{t,i} - A^s\alpha^s_{t,i}\|_2^2 \\ l^t(t,i) &= \|x^t_{t,i} - A^t\alpha^t_{t,i}\|_2^2 \end{aligned} \tag{12}$$

As for sRNN-AE model, the reconstruction error can be measured as follows

$$\begin{aligned} l^s(t,i) &= \|x^s_{t,i} - \hat{x}^s_{t,i}\|_2^2 \\ l^t(t,i) &= \|x^t_{t,i} - \hat{x}^t_{t,i}\|_2^2 \end{aligned} \tag{13}$$

Next, we pick the maximum reconstruction error among all patches within this frame as the frame level reconstruction error, *i.e.*, $l(t) = \max_i l(t,i)$. The reason for using the maximum reconstruction error as a measurement of anomaly is that it is robust to the small changes in spatial and temporal directions. In other words, if the patch corresponding to the maximum reconstruction error is normal, then the frame should be a normal event. Otherwise, if the patch with the maximum reconstruction error is abnormal, then the frame is highly likely to be abnormal. Further, following the work of [1] [25], and after calculating all frame level reconstruction errors for a testing video, we normalize the errors to the range $[0,1]$ and calculate a regularity score for each frame based on the following equations:

$$\begin{aligned} s^s(t) &= 1 - \frac{l^s(t) - \min_{k:1\ldots T} l^s(k)}{\max_{k:1\ldots T} l^s(k) - \min_{k:1\ldots T} l^s(k)} \\ s^t(t) &= 1 - \frac{l^t(t) - \min_{k:1\ldots T} l^t(k)}{\max_{k:1\ldots T} l^t(k) - \min_{k:1\ldots T} l^t(k)} \end{aligned} \tag{14}$$

where T means the length of a video. A smaller $s(t)$ means that the t-th frame more likely corresponds to an abnormal event. Finally, we combine spatial and temporal scores with a weight β that refers to the final score of a frame t is $s(t) = s^a(t) + \beta s^m(t)$, where $\beta \in [0,1]$. This is because anomalies can usually be easily discovered by appearance changes. Further, it is hard to properly characterize motion features compared to appearance features. Thus spatial anomaly detection is more robust than temporal anomaly detection, as shown in Table 8.

4 Experiments

In Section 4.1, we first introduce measurements used in all experiments. We empirically evaluate our proposed method under a controlled setting on a synthesized dataset in Section 4.2. Then, we compare our methods with other state-of-the-art methods on real anomaly detection datasets as well as our new ShanghaiTech

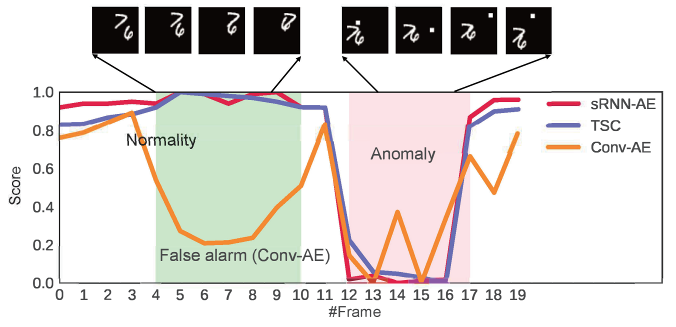


Fig. 4. A sample with an anomaly caused by appearance on the Moving-MNIST dataset.

TABLE 1
AUC on Moving-MNIST dataset.

	Conv-AE	TSC	sRNN-AE
Spatial Anomaly	~~74.30%~~	~~88.19%~~	~~**90.11%**~~
Temporal Anomaly	60.02%	65.47%	**68.51%**

anomaly dataset in Section 4.3. Different parameters in TSC and sRNN-AE are also empirically evaluated in Section 4.4. In addition, in order to describe the effectiveness of our proposed sRNN-AE, compared with other variant RNN-AE formulations, we conduct some experiments in Section 4.5. Two options for the similarity definition between neighboring codes will be discussed in Section 4.6. Different combinations of spatial and temporal streams will be discussed in Section 4.7. Finally, the running time will be reported as well in Section 4.8.

4.1 Experimental Setup

Measurements. We can predict whether an abnormal event occurs based on $s(t)$. One can set a threshold and if the score of a frame is smaller than the threshold, the frame can be categorized as an abnormal case. Obviously a higher threshold may cause a higher false negative ratio, while a lower one may lead to more false alarms. By changing the threshold gradually, we can arrive at an ROC curve. The Area Under the Curve (AUC) is a commonly used measurement for detecting irregularity [34]. In this paper, we use frame-level AUC to evaluate the performance of different methods.

Implementation Details. In our implementation, the learning rate for sRNN-AE is 0.00001. Many stacked RNNs including LSTMs illustrated in Equation (5) contain different trainable parameters such as W, U, V. However, our proposed method interpreting TSC with a stacked RNN finally result in only one trainable parameter A, which means all gradients will be accumulated into A. As shown in Fig. 1, if the number of blocks contributing to the calculation of the gradient of a trainable parameter in a vanilla stacked RNN is T, the number of blocks contributing to the calculation of the gradient of A in TSC counterpart sRNN is $T \times K \times 2$, where T is time steps, K is the number of stacked layers and 2 means that each cell accept the current moment input x_t as input. In our experiments, K can be larger than 10. Thus, we use a small learning rate for all ablation studies. The training sequence length is 10. The batch size in the training phase is 4. The dimension of the fully-connected layer in the trainable

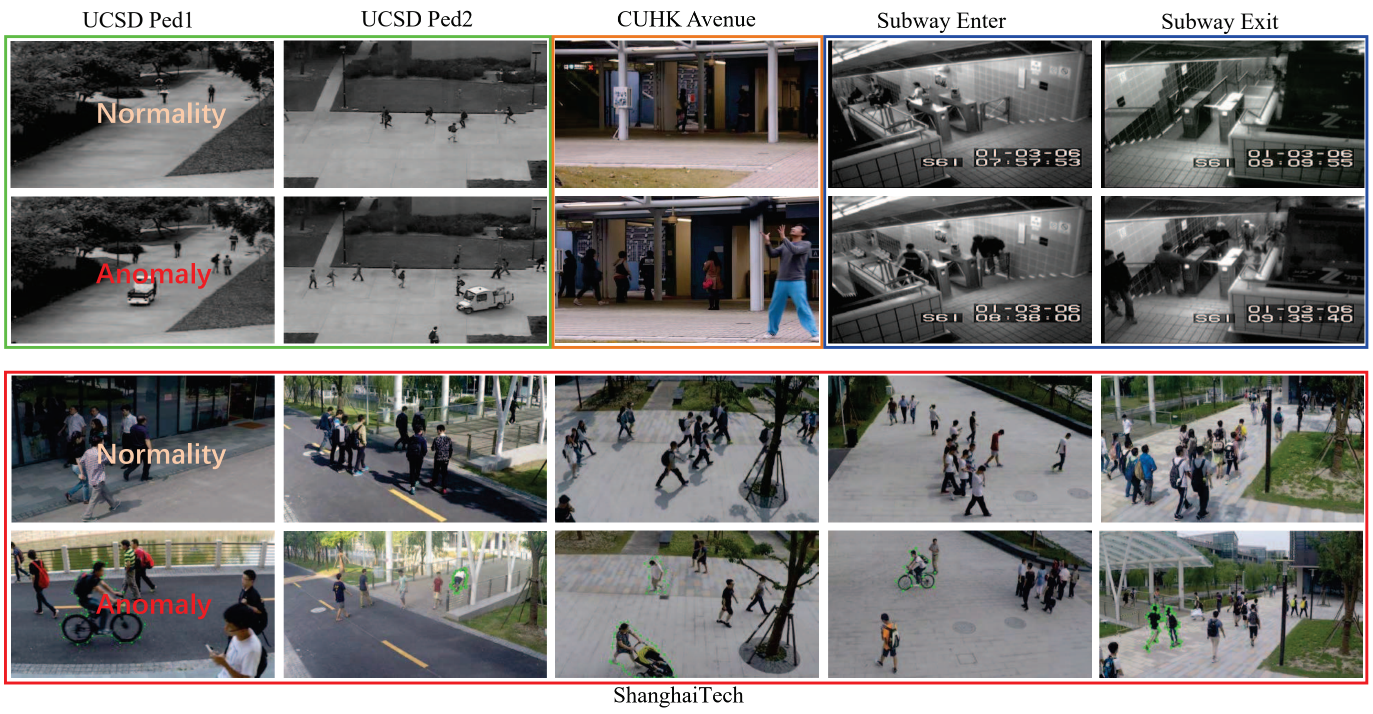


Fig. 5. Some samples from our new proposed dataset and other datasets. The first tow rows represent some samples from the UCSD Ped1, UCSD Ped2, CUHK Avenue and Subway Entrance and Subway Exit datasets, respectively. The last two rows represent normal and abnormal scenes from our proposed dataset (ShanghaiTech Campus).

TABLE 2
Comparision of our dataset with other released datasets.

Dataset	#Frames					#Abnormal Events	#Scenes
	Total	Training	Testing	Regularity	Irregularity		
Our Dataset	**317,398**	**274,515**	**42,883**	**300,308**	**17,090**	**130**	**13**
CUHK Avenue	30,652	15,328	15,324	26,832	3,820	47	1
UCSD Ped2	4,560	2,550	2,010	2,924	1,636	12	1
UCSD Ped1	14,000	6,800	7,200	9,995	4,005	40	1
Subway Entrance	136,524	20,000	116,524	134,124	2,400	66	1
Subway Exit	72,401	7,500	64,901	71,681	720	19	1
LV	309,940	127,500	182,440	240,951	68,989	34	30

similarity measurement is 512. We fix the number of iterations of sRNN-AE to 20,000 for all datasets. In the training phase, we leverage a ResNet pretrained on UCF101 for feature extraction [42]. Then we fix the pretrained ResNet in the feature extraction module and use a RMSPROP based SGD method to train the anomaly detection module for sRNN-AE. Specifically, for the spatial ResNet for appearance feature extraction, its architecture is the same with that in [40], then a pooling over the appearance features is conducted to summarizing information temporally. For the anomaly detection, we train the TSC with the (2) and normalize each column of A to be 1 to avoid the trivial solution in each iteration. We train the sRNN-AE with the Equation 9. It is worth noting that we optimize A, α, λ_1 and λ_2 in sRNN-AE rather than U, V and W because different from vanilla sRNN, in our sRNN-AE, U, V, and W depend on A, α, λ_1, and λ_2, as shown in Equation 7. In other words, the vanilla sRNN cannot characterize the dependencies between different layers. After training Spatial and Temporal sRNN-AEs, we add the normal scores of these two streams together. The weights corresponding to the spatial and temporal scores are fixed to be 1 and 0.5, respectively on all datasets. The whole pipeline is implemented with the Tensorflow framework [45].

4.2 Evaluate with A Synthesized Dataset

Anomaly in Appearance. To evaluate the performance of our method for the anomalies caused by a sudden change in appearance, we deploy experiments on a synthesized Moving-MNIST dataset. Specifically, we randomly choose two digits from the MNIST dataset, and put them in the center of a black image whose size is 225×225 pixels. Then in the next 19 frames, the digits randomly move horizontally or vertically. In this way, we can get a sequence with 20 frames. In our experiments, we synthesize 10,000 sequences for training data and train the network. For each testing sequence, 5 consecutive frames are randomly occluded by randomly inserting a 3×3 white box. We generate 3,000 sequences in total as test data. Then we use the intensity of the images as features and normalized them with an l_2 normalization.

Anomaly in Motion. We also evaluate the performance of our methods for the anomalies caused by a sudden change in motion. Specifically, we randomly choose two digits from the MNIST dataset, and put them in the center of a black image whose size is 225×225 pixels. In the first 10 frames, these two digits move together in a straight direction. After that, in each one of the next 10 frames, they move separately in two random directions. We can then also get a sequence with 20 frames. We generate the same amount as for anomalies in appearance.

The performance of the different methods are shown in Table 1. We can see that both TSC and sRNN-AE outperform Conv-AE when the anomalies are caused by either motion or appearance. Further, sRNN-AE outperforms TSC by around 2% and 3% for the anomalies caused by appearance and motion, respectively. We also show a sample with an anomaly caused by appearance in Figure 4.

4.3 Evaluation with Real Anomaly Datasets

We also evaluate our TSC and sRNN-AE with real anomaly detection datasets. It is desirable that the trained anomaly detection model can be directly applied in multiple scenes with multiple viewing angles. Most of existing datasets, however, only contain videos captured with one fixed angle camera, and they lack diversity of scenes and viewing angles. To increase scene diversity, we build a new anomaly detection dataset, which is named the ShanghaiTech Campus dataset. To the best of our knowledge, it is the biggest dataset for anomaly detection, which is even bigger than the sum of all existing datasets except for the LV in terms of the volume of data and the diversity of scenes. Further we introduce more anomalies caused by sudden motion in this dataset, such as chasing and brawling, which are not included in existing datasets. These characteristics make our dataset more suitable for real scenarios. We show some samples of our dataset in Figure 5 and list some statistics of different datasets in Table 2.

Specifically, we conduct experiments on our new proposed dataset as well as the two recently most used datasets, including ShanghaiTech Campus, CUHK Avenue [3], UCSD Ped2 [34], Subway [14] and LV [50].It is worth noting that for the UCSD pedestrian datasets, Ped1 is more frequently used for pixel-wise anomaly detection [23], while our work focuses on frame-level prediction, so we only conduct experiments on Ped2. For the Entrance dataset, ConvLSTM-AE [2] removes timestamps embedded in videos because the timestamps in videos usually leads to large reconstruction errors and hurts reconstruction based methods. But timestamps may appear at different places of videos, and it is trivial to remove it. Therefore, we rerun the ConvLSTM-AE without removing timestamps for fair comparison with our method. To better understand the differences between our dataset and existing anomaly detection datasets, we briefly summarize all anomaly detection datasets as follows:

- The CUHK Avenue [3] dataset contains 16 training videos and 21 testing videos with a total of 47 abnormal events, including throwing objects, loitering and running. The apparent size of people may change because of the camera position and angle.
- The UCSD Pedestrian 2 (Ped2) [34] dataset contains 16 training videos and 12 testing videos with 12 abnormal events. All of these abnormal cases are about vehicles such as bicycles and cars.
- The Subway [14] dataset is 2 hours long in total. There are two categories, *i.e.*Entrance and Exit. Unusual events contain walking in wrong directions and loitering. More importantly, this dataset was recorded in an indoor environment while the above ones were recorded in an outdoor environment.
- The LV dataset [50] is a challenging dataset, where all videos are collected online and abnormal events are realistic. Following the setting of [50], an abnormal frame is labelled as a true positive when at least 20% of abnormal regions of a frame is correctly detected, otherwise it is a false positive.
- Our ShanghaiTech Campus dataset has 13 scenes with complex lighting conditions and camera angles. It contains 130 abnormal events and over 270, 000 training frames. Moreover, the pixel level ground truth of abnormal events is also annotated in our dataset.

Baselines. Besides comparing our method with other state-of-the-art anomaly detection methods, including Conv-AE [1], Del *et al.* [32], Unmasking [33] and Hinami *et al.* [27], we further design another two baselines to evaluate how well does the proposed feature extraction module do directly on the anomaly detection module without the sRNN-AE counterpart. Specifically, on all datasets, we firstly extract appearance feature with dimensionality of 2048 for each frame, then give a testing frame, we calculate its similarity/distance to the training/normal frames for anomaly detection. Since there are too many training frames, and it is very time consuming to do the frame-wise comparison between each testing and training pair, and it is also very time consuming to do the sorting. To reduce the computational complexity, we propose two solutions:
i) Nearest Subspace: we use K-means to cluster training data into a dictionary A with size of 1000×2048, where 1000 is the dictionary size and 2048 is the dimensionality of feature. In the testing phase, we calculate the distance between testing frame and training with Nearest Subspace distance: $min_{\alpha} \|y - A\alpha\|_2$ for each testing frame feature y, where α is a coefficient of linear combination, and the optimal $\alpha^* = (A^T A)^{-1} A^T y$. After that, the reconstruction errors are normalized as normal scores.
ii) OC-SVM: we train a one-class SVM with all training data for anomaly detection.

We list the performance of different methods on these datasets in Table 3 and Table 4. It clearly shows that both our methods outperform all existing methods, including Conv-AE [1], Del *et al.* [32], Unmasking [33] and Hinami *et al.* [27], which are state-of-the-art methods for anomaly detection. Further, we can see that the extracted features are discriminative for anomaly detection, but both Nearest Subspace and one-class SVM based classifier is not as as good as our sRNN-AE and TSC on all datasets. Specifically, since our dataset contains multiple scenes which makes our dataset more realistic and challenging, the performance on our dataset is not as good as that on Avenue, Ped2, Entrance and Exit. Further, on all datasets, our sRNN-AE outperforms TSC, which validates the effectiveness of sRNN-AE. The reasons contributing to the improvement of sRNN-AE are two-fold: i) sRNN-AE can automatically learn the weights of the sparsity term and the temporally-coherent term. ii) the trainable similarity module leans a data dependent similarity, which is better than predefined similarities. The results in Table 4 also show that our

TABLE 3
AUC of different methods on the Avenue, Ped2, Entrance, Exit and our dataset (ShanghaiTech Campus).

	Avenue	Ped2	Entrance	Exit	Our dataset
MPPCA [34]	N/A	69.30%	N/A	N/A	N/A
MPPC+SFA [34]	N/A	61.30%	N/A	N/A	N/A
HOFME [46]	N/A	87.50%	N/A	N/A	N/A
Conv-AE [1]	74.50%	81.10%	91.00%	80.20%	60.85%
Del et al.. [32]	78.30%	N/A	69.10%	82.40%	N/A
ConvLSTM-AE [2]	77.00%	88.10%	84.30%	87.7%	55.00%
Unmasking [33]	80.60%	82.20%	71.30%	86.30%	N/A
Hinami *et al.* [27]	N/A	92.20%	N/A	N/A	N/A
Nearest Subspace	76.67%	83.87%	77.04%	83.34%	65.33%
OC-SVM	78.23%	78.58%	78.34%	81.56%	53.11%
TSC	80.56%	91.03%	84.24%	87.54%	67.94%
sRNN-AE	**83.48%**	**92.21%**	**85.38%**	**89.73%**	**69.63%**

TABLE 4
AUC on the LV dataset

Lu*et al.* [3]	Biswas *et al.* [47]	Reddy*et al.* [48]	Javan *et al.* [49]	Conv-AE [1]	ConvLSTM-AE [2]	TSC	SRNN-AE
11.20%	15.10%	32.50%	42.70%	33.64%	39.41%	55.34%	58.27%

sRNN-AE achieves AUC of 58.27% which is much better than the state-of-the-art one of 42.7%, which further demonstrates the effectiveness of our proposed method.

Finally, we show the change of the score ($s(t)$), the similarities between neighboring frames ($S_{t,t-1}$) and the distances between sparse codes of neighboring frames ($\|\alpha_t - \alpha_{t-1}\|$) for some normal and abnormal events on the Ped2 and ShanghaiTech datasets in Figure 6. We can see that some smooth similarities and distances can be found for the frames within normal or abnormal events, which agrees with the motivation of our TSC.

4.4 The Effect of Different Hyper-Parameters in TSC and sRNN-AE

In this subsection, we conduct some experiments on the effect of different hyperparameters in TSC and sRNN-AE. Since dictionary training and coefficient optimization is very time-consuming, experiments conducted in this subsection are only based on appearance features.

Weight of the Sparsity Term(λ_1) in TSC. λ_1 in Equation (2) controls the sparsity of α_t. As shown in Algorithm 1, α_t^k is optimized based on a soft-thresholding operator. The bigger λ_1 is, the more sparse α_t will be. We fix λ_2 and the dictionary size to 2.0 and 2048×2048, respectively, and change λ_1 to observe how this parameter affects the AUC on Ped2, Avenue and ShanghaiTech. As shown in Figure 7(a), a bigger λ_1 improves the AUC on Avenue but reduces the performance for the Ped2 and ShanghaiTech datasets.

Weight of the Temporally-coherent Term (λ_2) in TSC. λ_2 in Equation (2) controls the smoothness of the sparse codes between neighboring frames. Figure 7(b) demonstrates that different datasets may be affected differently by λ_2. For example, Ped2 and Avenue prefers a larger λ_2 but ShanghaiTech prefers a smaller λ_2.

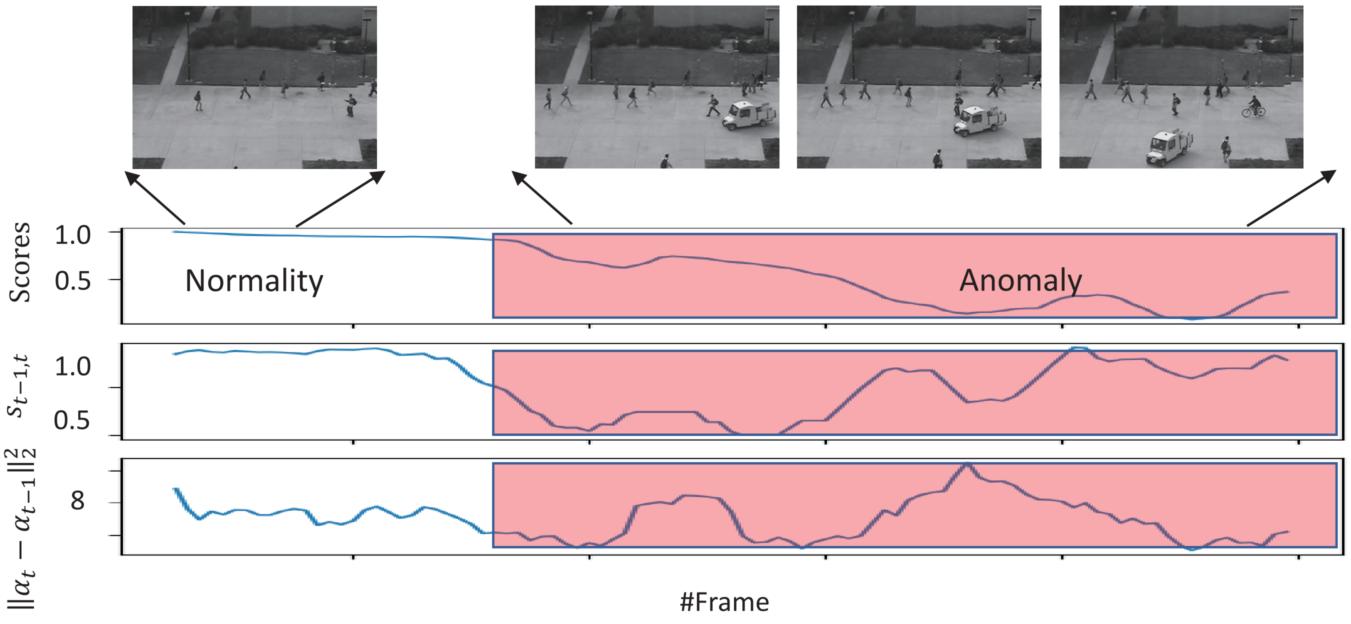
Dictionary Size. We show the change of TSC performance with respect to the change of dictionary size on the Avenue dataset in Figure 7(c). We can see that a larger dictionary does not always improve AUC and that the optimal dictionary size varies for different datasets. In addition, we report the performance of different dictionary sizes in sRNN-AE. We can see that sRNN-AE always outperforms TSC when dictionary size varies. For saving both training and testing time, we set dictionary size to 2048 for all datasets.

Number of Layers in sRNN-AE. The optimization of the SISTA algorithm requires a very large K to achieve a sparse solution with a small reconstruction error. Fewer iterative steps may harm the optimization of TSC. A larger K means a deeper sRNN, as the counterpart of TSC.

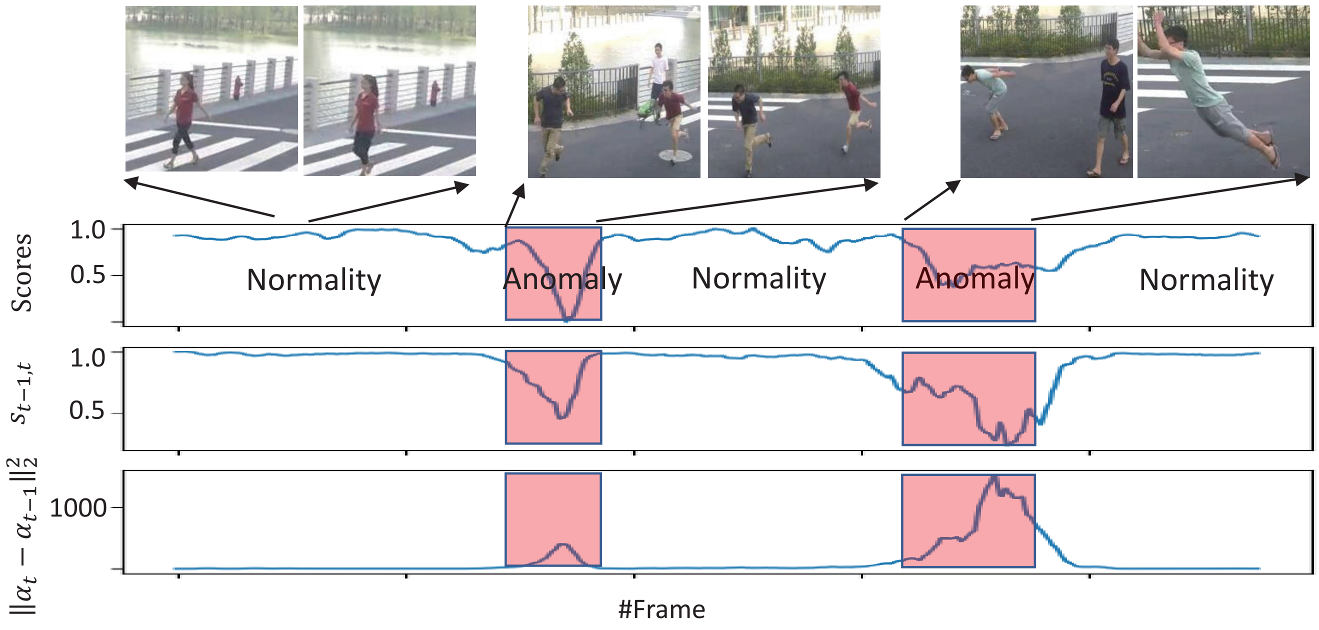
However, a very deep sRNN-AE may lead to gradient vanishing or explosion, which is harder to optimize. To validate how K affects the performance of our sRNN-AE, we set it to different values (1, 2, 3, 5, 10, 20, 30), respectively. The sparsity and AUC of sRNN-AE with different numbers of layers on Avenue are shown in Figure 8. The sparsity (percentage of zero entries) of 3-layers-based sRNN-AE and that of 30-layers-based sRNN-AE is 80.0% and 90.0%, respectively, while the AUC for 3-layers-based sRNN-AE and 30-layers-based sRNN-AE is 83.48% and 79.19%, respectively. This experiment shows that sparsity does not necessarily lead to better performance in sRNN-AE. In our experiments, we set $K = 3$ for all datasets. Such a shallow architecture also accelerates the inference of $\alpha_t(h_t)$ in the test phase. We also show the change of the objective with respect to iteration in Figure 9. We can see that our sRNN-AE converges at around 10,000 iterations.

4.5 Comparison between sRNN-AE and Other Types of RNN

Our sRNN-AE is a special type of Recurrent Neural Networks (RNN) based Auto-Encoder. To verify the effectiveness of such a sparse coding inspired sRNN, we also compare our sRNN-AE with the LSTM based Auto-Encoder, where the same features are used, and the ConvLSTM based Auto-Encoder which extracts features from raw pixels. The AUC of these methods on Avenue, Ped2 and ShanghaiTech is listed in Table 5. We can see that our sRNN-AE also outperforms these two baselines. In addition, our sRNN-AE can be well interpreted compared to other types of RNN based Auto-Encoders.



(a) Ped2



(b) ShanghaiTech

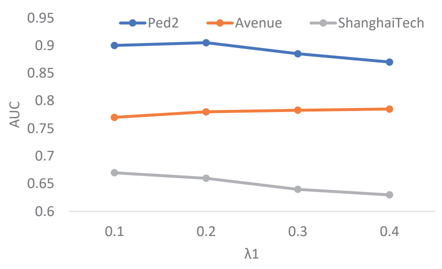
Fig. 6. Scores, similarities and distances between neighboring codes of two video samples on the Ped2 and ShanghaiTech. We can see that the similarities between neighboring frames can be kept for normal events. We highlight the abnormal events with red boxes. (Best viewed in color)

TABLE 5
AUC of different RNN variants on Avenue, Ped2 and ShanghaiTech datasets.

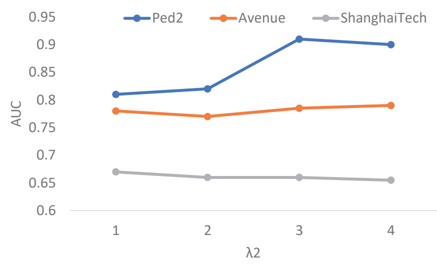
	Avenue	Ped2	ShanghaiTech
LSTM-AE	75.33%	83.62%	53.30%
ConvLSTM-AE	77.00%	88.10%	55.00%
sRNN-AE	**83.48%**	**92.21%**	**69.63%**

4.6 Similarity Measurement

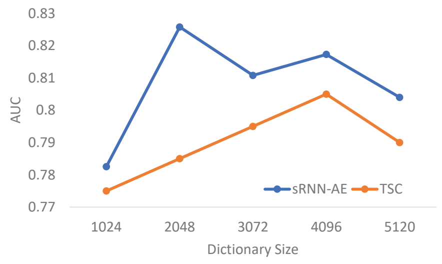
To verify the effectiveness of the similarity learning module in sRNN-AE, we also use a Gaussian kernel based similarity measurement and fix λ_2 in sRNN-AE and learn other parameters, including λ_1. The results of the Gaussian kernel based and trainable similarity measurement are shown in Table 6. We can see that the trainable similarity measurement achieves a better performance than the predefined similarity method, which verifies the importance of trainable data dependent similarity. We also show a pair of normal and abnormal images in Figure 10. We can



(a) AUC versus λ_1 (λ_2=2.0)



(b) AUC versus λ_2 (λ_1=0.2)



(c) AUC versus dictionary size (λ_1=0.2, λ_2=2.0)

Fig. 7. The change of AUC w.r.t. λ_1, λ_2 and dictionary size . (a) and (b) are conducted on Avenue and Ped2 datasets. (c) is conducted on Avenue. (Best viewed in color)

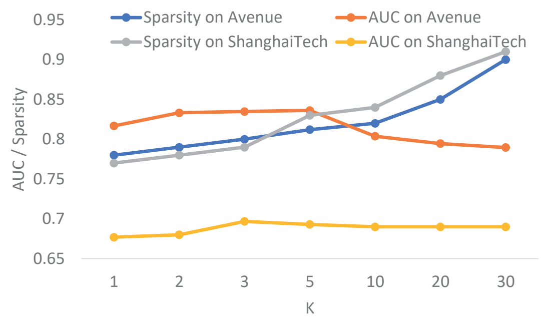


Fig. 8. AUC and sparsity of different numbers of layers in SRNN-AE on the Avenue and the ShanghaiTech dataset. (Best viewed in color)

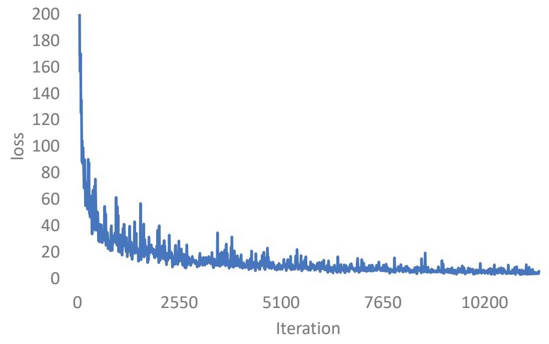


Fig. 9. The change of training loss with different number of iterations on Avenue dataset.

see that our trainable similarity better characterizes similarities for both normal and abnormal image pairs.

4.7 The Combination of Spatial and Temporal Features

Optical Flow Based Motion Features vs. Our Temporal Aggregated Features. In our implementation, we use a ResNet [51] for appearance feature extraction and use temporal pooling over the appearance features of 4 consecutive frames as temporal features. We also compare our temporal features with optical flow based ones, as done in two-streams CNNs for action recognition. For optical flow based motion features, we also use a ConvNet pretrained with the UCF101 dataset for motion feature extraction. The performance based on these two types of motion features is shown in Table 7. We can see that our temporal features is more effective than optical flow based ones. The possible reason for

TABLE 6
AUC of different similarity measurement on Avenue, Ped2 and ShanghaiTech datasets.

	Avenue	Ped2	ShanghaiTech
Gaussian kernel	81.71%	91.03%	68.00%
Trainable similarity	**82.58%**	**91.20%**	**69.63%**

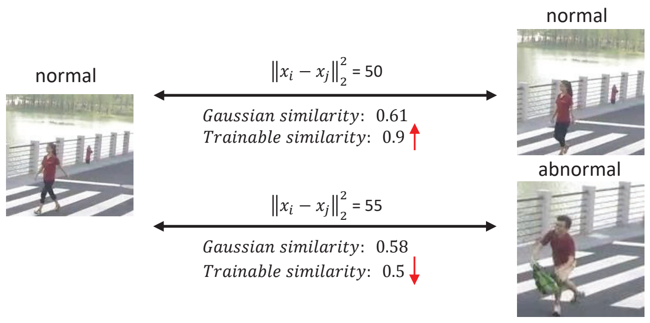

normal
normal
abnormal
$\|x_i - x_j\|_2^2$ = 50
Gaussian similarity: 0.61
Trainable similarity: 0.9
$\|x_i - x_j\|_2^2$ = 55
Gaussian similarity: 0.58
Trainable similarity: 0.5

Fig. 10. The learned similarity with two different strategies.

this is that some types of anomalies are caused by a very fast movement, while optical flow estimation for very fast movement is not easy and usually inaccurate. As a result, the performance of optical flow based motion estimation is reduced. Further, our temporal feature extraction strategy is much faster than optical flow based motion feature extraction, which guarantees real-time anomaly detection.

Early Fusion vs. Late Fusion We also list the performance of anomaly detection based early fusion and late fusion in Table 9. We can see that late fusion always outperforms early fusion, which agrees with the findings for action recognition [43]. There are two possible reasons for this. First, when the number of nodes are the same in hidden layers for both modalities, two separate sRNN-AE algorithms reduces the number of parameters by a half compared with early fusion. Thus, late fusion facilitates the training of a more robust sRNN-AE. Second, late fusion is more plausible for anomaly detection because humans infer anomalies either by appearance or motion, thus it may be more suitable to combine the spatial and temporal anomaly scores at the final stage during anomaly detection. On the one hand, for early fusion, the nodes in hidden layers may receive signals from both spatial and temporal directions. On the other hand, late fusion enforces that the hidden nodes only receive signals from one type of feature,

TABLE 7
AUC and average inference time of different temporal features on three datasets.

	Optical flow based only	Appearance aggregation based only	Appearance+Optical flow based	Appearance+Appearance aggregation based
Avenue	81.84%	82.42%	82.20%	**83.48%**
Ped2	86.43%	88.82%	88.84%	**92.21%**
ShanghaiTech	66.25%	68.16%	67.12%	**69.63%**
average inference time	5 FPS	10 FPS	3 FPS	**10 FPS**

TABLE 8
AUC of only spatial and only temporal features on three datasets.

	Spatial features	Temporal features
Avenue	**82.58%**	82.42%
Ped2	**91.20%**	88.82%
ShanghaiTech	**69.63%**	68.16%

TABLE 9
AUC of different feature fusions on three datasets.

	Early fusion	Late Fusion
Avenue	82.90%	**83.48%**
Ped2	89.80%	**92.21%**
ShanghaiTech	68.60%	**69.63%**

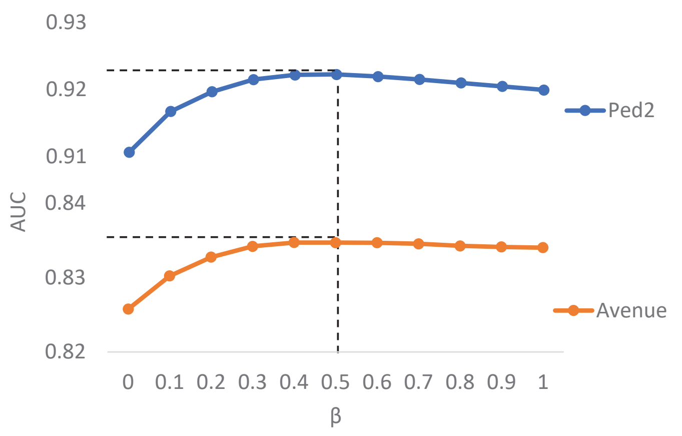


Fig. 11. The AUC with different β on Avenue and Ped2 dataset.

making judgments merely based on one type of feature.

The Combination of Spatial Normal Scores and Temporal Normal Scores. Appearance is a strong cue for anomaly detection (for example, if unexpected objects appear, then we have confidence to say such a phenomenon is abnormal.). In addition, CNN has shown its expertise for appearance feature extraction, thus many existing work only leverages appearance for anomaly detection [1] [2]. Temporal features representation is not as good as spatial features. Therefore, spatial normal scores are more reliable than those temporal normal scores, as shown in Table 8. Thus it is more reasonable to combine normal scores inferred by spatial features and temporal features with a weight. Here we show the change of the AUC with respect to β on Avenue and Ped2 in Figure 11 where we can see that $\beta = 0.5$ corresponds to a higher AUC. Thus we simply fix $\beta = 0.5$ on all datasets.

4.8 Running Time

We report training and inference time of Avenue in Table 10. It is obvious that sRNN-AE with 3 layers is much faster than TSC with 30 layers. This means our proposed sRNN-AE effective and efficient for anomaly detection. Further, if the time for feature extraction is included, our sRNN-AE can run at a speed of 10 FPS.

TABLE 10
Running time of Conv-AE with feature extraction, TSC and sRNN-AE without feature extraction on Avenue dataset.

	K	Training	Inference
Conv-AE	N/A	6 hours	30 FPS
TSC	30	30 hours	7 FPS
sRNN-AE	3	1.2 hours	152 FPS

5 Conclusion

In this paper, we propose a TSC framework for anomaly detection which preserves the similarities between frames within normal and abnormal events. Our TSC can be interpreted with a special sRNN. By optimizing all parameters in sRNN-AE simultaneously, we avoid nontrivial parameter selection and reduce the computational cost for inferring the reconstruction coefficients in the test phase. Further, we propose a multi-layer perceptron based similarity measurement in sRNN-AE which learns a data dependent similarity, which demonstrates better performance than predefined similarity measurement. In addition, we propose combining the spatial and temporal features in a late fusion manner which further improves performance. Considering the fact that most anomaly detection datasets only contain one scene with the same view angle, we build a new dataset which is the most challenging one in terms of data volume and scene diversity. Extensive experiments on both synthesized datasets and real datasets validate the effectiveness of sRNN-AE for anomaly detection.

Acknowledgment

This work was supported in part by the National Key Research and Development Program of China under Grant 2016YFB1001001 and NSFC (No. 61502304).

References

[1] M. Hasan, J. Choi, J. Neumann, A. K. Roy-Chowdhury, and L. S. Davis, “Learning temporal regularity in video sequences,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2016.
[2] W. Luo, W. Liu, and S. Gao, “Remembering history with convolutional lstm for anomaly detection,” in *Multimedia and Expo (ICME), 2017 IEEE International Conference on*. IEEE, 2017, pp. 439–444.

[3] C. Lu, J. Shi, and J. Jia, "Abnormal event detection at 150 fps in matlab," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 2720–2727.

[4] B. Zhao, F. Li, and E. P. Xing, "Online detection of unusual events in videos via dynamic sparse coding," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. IEEE, 2011, pp. 3313–3320.

[5] S. Wisdom, T. Powers, J. Pitton, and L. Atlas, "Interpretable recurrent neural networks using sequential sparse recovery," *NIPS 2016 Workshop on Interpretable Machine Learning in Complex Systems*, 2016.

[6] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[7] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in neural information processing systems*, 2014, pp. 568–576.

[8] R. Villegas, J. Yang, S. Hong, X. Lin, and H. Lee, "Decomposing motion and content for natural video sequence prediction," *arXiv preprint arXiv:1706.08033*, 2017.

[9] W. Luo, W. Liu, and S. Gao, "A revisit of sparse coding based anomaly detection in stacked rnn framework," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct 2017.

[10] F. Tung, J. S. Zelek, and D. A. Clausi, "Goal-based trajectory analysis for unusual behaviour detection in intelligent surveillance," *Image and Vision Computing*, vol. 29, no. 4, pp. 230–240, 2011.

[11] N. Navneet and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[12] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *European conference on computer vision*. Springer, 2006, pp. 428–441.

[13] D. Zhang, D. Gatica-Perez, S. Bengio, and I. McCowan, "Semi-supervised adapted hmms for unusual event detection," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 611–618.

[14] A. Adam, E. Rivlin, I. Shimshoni, and D. Reinitz, "Robust real-time unusual event detection using multiple fixed-location monitors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 30, no. 3, pp. 555–560, 2008.

[15] J. Kim and K. Grauman, "Observe locally, infer globally: a space-time mrf for detecting abnormal activities with incremental updates," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2921–2928.

[16] R. Leyva, V. Sanchez, and C.-T. Li, "Video anomaly detection with compact feature sets for online performance," *IEEE Transactions on Image Processing*, vol. 26, no. 7, pp. 3463–3478, 2017.

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

[18] H. Ren, H. Pan, S. I. Olsen, and T. B. Moeslund, "A comprehensive study of sparse codes on abnormality detection," *arXiv preprint arXiv:1603.04026*, 2016.

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[20] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[21] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1440–1448.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

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

[24] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE transactions on pattern analysis and machine intelligence*.

[25] Y. S. Chong and Y. H. Tay, "Abnormal event detection in videos using spatiotemporal autoencoder," in *International Symposium on Neural Networks*. Springer, 2017, pp. 189–196.

[26] J. R. Medel and A. Savakis, "Anomaly detection in video using predictive convolutional long short-term memory networks," *arXiv preprint arXiv:1612.00390*, 2016.

[27] R. Hinami, T. Mei, and S. Satoh, "Joint detection and recounting of abnormal events by learning deep generic knowledge," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct 2017.

[28] M. Sabokrou, M. Fayyaz, M. Fathy, and R. Klette, "Fully convolutional neural network for fast anomaly detection in crowded scenes. arxiv preprint," *arXiv preprint arXiv:1609.00866*, 2016.

[29] W. Sultani, C. Chen, and M. Shah, "Real-world anomaly detection in surveillance videos," *Center for Research in Computer Vision (CRCV), University of Central Florida (UCF)*, 2018.

[30] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *International Conference on Information Processing in Medical Imaging*. Springer, 2017, pp. 146–157.

[31] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[32] A. D. Giorno, J. A. Bagnell, and M. Hebert, "A discriminative framework for anomaly detection in large videos," in *European Conference on Computer Vision*. Springer, 2016, pp. 334–349.

[33] R. Tudor Ionescu, S. Smeureanu, B. Alexe, and M. Popescu, "Unmasking the abnormal events in video," in *Proceedings of the IEEE International Conference on Computer Vision*, Oct 2017.

[34] V. Mahadevan, W. Li, V. Bhalodia, and N. Vasconcelos, "Anomaly detection in crowded scenes," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, vol. 249, 2010, p. 250.

[35] M. Aharon, M. Elad, and A. Bruckstein, "rmk-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on signal processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[36] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," *Advances in neural information processing systems*, vol. 19, p. 801, 2007.

[37] A. Graves, A. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[38] J. Chung, C. Gülçehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks." in *International Conference on Machine Learning*, 2015, pp. 2067–2075.

[39] T. Tieleman and G. E. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," in *COURSERA: Neural Networks for Machine Learning*, 2012.

[40] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Convolutional two-stream network fusion for video action recognition," *arXiv preprint arXiv:1604.06573*, 2016.

[41] C. Feichtenhofer, A. Pinz, and R. P. Wildes, "Spatiotemporal multiplier networks for video action recognition," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017, pp. 7445–7454.

[42] K. Soomro, A. R. Zamir, and M. Shah, "Ucf101: A dataset of 101 human actions classes from videos in the wild," *arXiv preprint arXiv:1212.0402*, 2012.

[43] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[44] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *european conference on computer vision*. Springer, 2014, pp. 346–361.

[45] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[46] T. Wang and H. Snoussi, "Histograms of optical flow orientation for abnormal events detection," in *Performance Evaluation of Tracking and Surveillance (PETS), 2013 IEEE International Workshop on*. IEEE, 2013, pp. 45–52.

[47] S. Biswas and R. V. Babu, "Real time anomaly detection in h. 264 compressed videos," in *Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG), 2013 Fourth National Conference on*. IEEE, 2013, pp. 1–4.

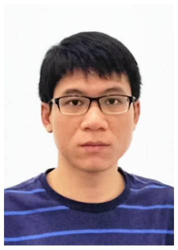
[48] V. Reddy, C. Sanderson, and B. C. Lovell, "Improved anomaly detection in crowded scenes via cell-based analysis of foreground speed, size and texture," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2011 IEEE Computer Society Conference on*. IEEE, 2011, pp. 55–61.

[49] M. J. Roshtkhari and M. D. Levine, "An on-line, real-time learning method for detecting anomalies in videos using spatio-temporal compositions," *Computer Vision and Image Understanding*, vol. 117, no. 10, pp. 1436–1452, 2013.

[50] R. Leyva, V. Sanchez, and C.-T. Li, "The lv dataset: A realistic surveillance video dataset for abnormal event detection," in *Biometrics and Forensics (IWBF), 2017 5th International Workshop on*. IEEE, 2017, pp. 1–6.

[51] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

Weixin Luo received the bachelor degree from Shenzhen University, Shenzhen, China. He is currently pursuing the Ph.D. degree at ShanghaiTech University.



Wen Liu received the bachelor degree from Northwestern Polytechnical University, Xian, China. He is currently pursuing the Ph.D degree at ShanghaiTech University.

Dongze Lian received the bachelor degree from Dalian University of Technology, Dalian, China. He is currently pursuing the Ph.D. degree at ShanghaiTech University.

Jinhui Tang (M'08SM'14) received the B.Eng. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2003 and 2008, respectively. He is currently a Professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. From 2008 to 2010, he was a Research Fellow with the School of Computing, National University of Singapore, Singapore. He has authored over 150 papers in top-tier journals and conferences. His current research interests include multimedia analysis and search, computer vision, and machine learning. Dr. Tang was a recipient of the best paper awards in ACM MM 2007, PCM 2011, and ICIMCS 2011, the Best Paper Runner-up in ACM MM 2015, and the best student paper awards in MMM 2016 and ICIMCS 2017. He has served as an Associate Editor for the IEEE TNNLS, IEEE TKDE, and IEEE TCSVT.

Lixin Duan Dr. Lixin Duan is a professor at University of Electronic Science and Technology of China. He received Ph.D. from the School of Computer Engineering at the Nanyang Technological University in 2012 and B.Eng. from the Department of Electronic Engineering and Information Science at the University of Science and Technology of China in 2008.

Xi Peng received the Ph.D. degree in Computer Science from the Sichuan University, China in 2013. He currently is a research professor with the College of Computer Science, Sichuan University, Chengdu, China. From 2014 to 2017, he was a research scientist at Institute for Infocomm, Research Agency for Science, Technology and Research (A*STAR) Singapore. His current research interests include machine intelligence and computer vision and has authored more than 30 articles in these areas.

Dr. Peng has served as a Associate Editor/Guest Editor for *IEEE Access*, *IEEE Trans. on Neural Network and Learning Systems*, and *Image and Vision Computing*, a Session Chair for AAAI'17, a Senior Program Committee Member for IJCAI'17; a Program Committee Member and reviewer for over 30 international conferences and international journals. He has organized a tutorial at ECCV'16 and a special session at VCIP'17.

Shenghua Gao is an assistant professor, PI in ShanghaiTech University, China. He received the B.E. degree from the University of Science and Technology of China in 2008 (outstanding graduates), and received the Ph.D. degree from the Nanyang Technological University in 2012. From Jun 2012 to Jul 2014, he worked as a postdoctoral fellow in Advanced Digital Sciences Center, Singapore. His research interests include computer vision and machine learning.