4-2017

Online growing neural gas for anomaly detection in changing surveillance scenes

Qianru SUN
Singapore Management University, qianrusun@smu.edu.sg

Hong LIU

Tatsuya HARADA

Pattern Recognition 64 (2017) 187–201

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Online growing neural gas for anomaly detection in changing surveillance scenes☆

Qianru Sun[a,b], Hong Liu[b,*], Tatsuya Harada[c]

[a] Max Planck Institute for Informatics, Saarbruecken 66123, Germany
[b] Shenzhen Graduate School, Peking University, Beijing 100871 China
[c] School of Information Science and Technology, University of Tokyo, Tokyo 113-8685, Japan

ARTICLE INFO

Keywords:
Anomaly detection
Video surveillance
Unsupervised learning

ABSTRACT

Anomaly detection is still a challenging task for video surveillance due to complex environments and unpredictable human behaviors. Most existing approaches train offline detectors using manually labeled data and predefined parameters, and are hard to model changing scenes. This paper introduces a neural network based model called online Growing Neural Gas (online GNG) to perform an unsupervised learning. Unlike a parameter-fixed GNG, our model updates learning parameters continuously, for which we propose several online neighbor-related strategies. Specific operations, namely neuron insertion, deletion, learning rate adaptation and stopping criteria selection, get upgraded to online modes. In the anomaly detection stage, the behavior patterns far away from our model are labeled as anomalous, for which *far away* is measured by a time-varying threshold. Experiments are implemented on three surveillance datasets, namely UMN, UCSD Ped1/Ped2 and Avenue dataset. All datasets have changing scenes due to mutable crowd density and behavior types. Anomaly detection results show that our model can adapt to the current scene rapidly and reduce false alarms while still detecting most anomalies. Quantitative comparisons with 12 recent approaches further confirm our superiority.

1. Introduction

Recent years have witnessed the development of video surveillance in public sites such as subways and airports. It is desirable to figure out the *rare* or *distinctive* behaviors which are probably anomalous. However, surveillance cameras capture a huge number of videos, dealing with which human operators have to burden a large amount of labor. With the development of artificial intelligence, this problem can be resolved using automated anomaly detectors [1–3]. Extensive works including supervised models [5–10] and unsupervised models [15,16,18,22,24] have been proposed in recent years [4].

In this paper, we mainly focus on the unsupervised approach without using manually labeled data. The aim is to detect human anomalous events in changing surveillance scenes which contain mutable crowd density and behavior types. To model the human behaviors in changing scenes, Feng et al. [24] proposed an online Self-Organizing Map (online SOM) model and obtained satisfying results for detecting local anomalies in outdoor crowded scenes. Compared with original SOM [26], online SOM model is more efficient for its ability of adjusting learning rates and neighborhood sizes according to changing scenes. However, as shown on the left of Fig. 1, SOM has the inherent limitation that training samples have to be fixed into a pre-structured lattice. This lattice structure make some null neurons (red dots) far away from realistic samples (small black points) and results in a bad representation of the data space. For example, when an anomalous sample comes, it might match well with one of the null neurons, thereby causing a missing detection. Besides, the number of neurons in SOM has to be set in prior and keep unchanged. However, it is very difficult to decide an optimal number in advance, especially for modeling changing scenes.

To avoid these problems of SOM, this paper utilizes a more flexible neural network called Growing Neural Gas (GNG) [32]. GNG learns data topology utilizing competitive Hebbian learning [34], and it has been successfully applied to tasks including gesture recognition [35], trajectory modeling [58,59], supervised stream data classification [57] and so on. Different from SOM, GNG organizes the topology of neurons based on realistic sample locations, as shown on the right of Fig. 1. Current inserted neuron (the blue dot) is determined by the regional density of new samples. In another respect, GNG neurons are not learned by the absolute distance in a fixed lattice but by the relative

☆ This manuscript is submitted on October 20, 2015, and revised on April 23, 2016.

* Corresponding author.
E-mail addresses: qianrusun@pku.edu.cn, qsun@mpi-inf.mpg.de (Q. Sun), hongliu@pku.edu.cn, liuh@szpku.edu.cn (H. Liu), harada@mi.t.u-tokyo.ac.jp (T. Harada).

http://dx.doi.org/10.1016/j.patcog.2016.09.016
Received 20 October 2015; Received in revised form 18 September 2016; Accepted 19 September 2016
Available online 10 October 2016
0031-3203/

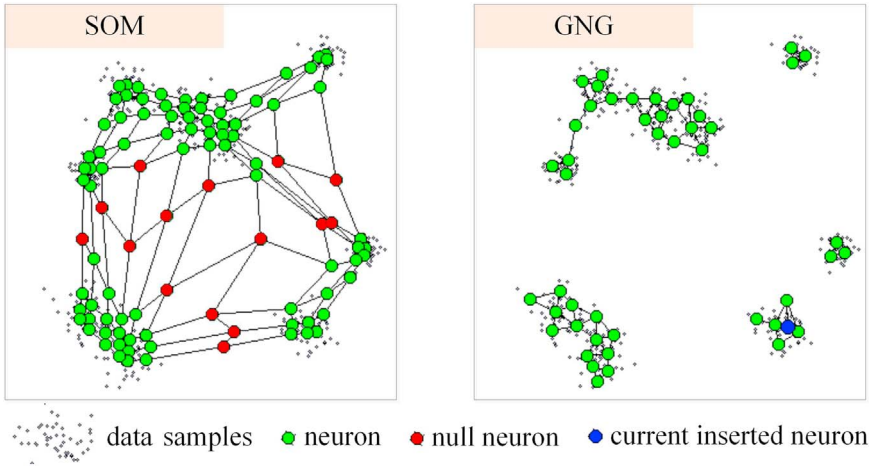


Fig. 1. 2D visualizations of SOM and GNG networks. Both of them are generated to cover the same group of data samples. Colored dots indicates the network neurons and black lines between pairwise neurons are edges which indicate the neighborhood information. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

distances between samples, making the network more flexible to adjust the number of neurons according to observations. However, in our case of modeling changing scenes, GNG has its problem that it can not change dominant learning parameters, such as learning rates, stopping criteria and so on, which constrains its efficiency. Although Waniek et al. [59] used GNG for real-time anomaly detection, they still used fixed learning parameters. Beyer et al. [57] introduced a mechanism to control the network size of GNG adaptively, but they conducted a supervised classification relied on labeled textual data. In our case with changing scenes and unlabeled data, it is desirable that the method can adjust dominant learning parameters systematically and optimize the model based on the realistic data [36,57].

An original GNG network is expanded by *inserting* neurons, adjusted by *learning* neuron weights, refined by *deleting* neurons, and continuously converges to the data space before *stopping* criterion is satisfied. In this paper, our contributions consist of proposing an online version of GNG (online GNG) to operate above steps with fully self-adaptive parameters, and solving the anomaly detection problem in changing scenes with time-varying anomaly thresholds.

For online GNG, we attempt to answer four specific questions: (1) when and how to insert a new neuron, (2) how to adjust learning rates to learn input samples efficiently, (3) how to define and delete useless neurons, (4) when to stop the learning. Accordingly, we propose a series of online neighbor-related strategies that utilize the relationships between input samples and network neurons. In other words, when each sample finds its nearest neuron, called the winner, the relationships among the sample, the winner, and winner's neighbors will be used for parameter self-adaptation.

In the online testing stage, it is assumed that an anomalous pattern is *significantly different* from any learned neuron, for which *significantly different* is measured by the Gaussian distribution of distances between all samples and their winners [24]. To reduce false alarms in noisy environments, this paper defines that an anomalous event contains *enough* anomalous patterns, for which *enough* is defined by the Gaussian distribution of anomalous patterns. Therefore, input samples are used, at some point, not only to train the model but also to update anomaly thresholds. This mechanism is very helpful for reducing false alarms and missing detections caused by model aging and threshold aging, making the online model converge to the changing data space according to new observations.

1.1. Related works

Although this paper mainly focuses on the unsupervised anomaly detection approach, many supervised approaches are very instructive and meaningful in this research field.

Some supervised approaches [5–7] rely on labeled data and well-defined rules to train the models of normal events. In the detection stage, they classify those falling outside of these models as anomalies, for which *outside* is usually defined by learning a threshold. Jodoin et al. [5] estimated object paths and recorded the maximum amount of activities occurring during the normal-only training. Any event which caused the activity amount to exceed the threshold would be detected as an anomaly. A similar method by Zweng and Kampel [6] drew a hit map to record the pixel location of human activity through normal data training. They used the hit map to generate an unexpected location map where the activity did not occur. An anomaly alarm would be triggered when enough activity intensity appeared in any unexpected region. Dong et al. [7] combined pointwise motion images with AdaBoost algorithm. In their work, a normal score range involving motion speed, orientation, duration, and shape, was firstly learned from the normal data, then any outlier of this range would be detected as an anomaly.

A more complex idea than learning a threshold is to train a multidimensional model of normal events [8–10]. Benezeth et al. [8] created a co-occurrence matrix indicating when and where pixels had active motion labels. They learned a normal co-occurrence matrix by a Markov Random Field (MRF). Some other related works studied the relationships between observed objects [9,10]. Loy et al. [9] first decomposed a complex behavior pattern according to its spatial–temporal visual contexts. Then, a normal behavior was formulated with a cascade of Dynamic Bayesian Networks. Yao et al. [10] observed the objects' spatio-temporal relationships concerning their sources, sinks, and intermediate tracks. If the probability of a hidden track were lower than the learned threshold, the track would be recognized as an anomaly.

These approaches have the benefit of not requiring any training data for anomalous events or behaviors, which are often rare and difficult to collect. However, they tend to produce a lot of false alarms, since any pattern not sufficiently represented due to the limited training data will be regarded as anomalous. To specify behavior classes, some approaches perform manual labeling for training data. Anomaly is identified if an observed pattern goes against all labeled classes [11–13].

Yin and Meng [11] proposed Self-Adaptive Hidden Markov Models to identify motions such as walking, sitting as normality, and more complex ones such as crouching, falling as anomalies. Chen et al. [12] utilized Hidden Markov Models to learn normal behaviors of swimming such as breaststroke, backstroke, and anomalies such as grasping ropes and struggling in the pool. Lao et al. [13] proposed a similar

approach to detect robbery behaviors. In their work, only postures such as pointing, squatting and lying down were recorded in the training stage. Anomalous behaviors were defined using some specified rules on the temporal relationship of these postures.

These models with supervised labeling have the drawback that only properly defined events can be recognized reliably. Moreover, it is clear that manual labeling needs too many offline operations and restricts the application in changing scenes where behavior density and types are mutable. Therefore, Xiang and Gong [14] proposed that supervised approaches should be substituted by unsupervised ones which do not require labeled data.

For unsupervised learning, anomaly definition is usually based on following assumptions: (1) anomalous events infrequently occur in comparison to normal events; (2) anomalous events have significantly different characteristics from normal ones. Concerning specific algorithms, recent unsupervised approaches are based on either sparse coding or pattern clustering. Both sides have achieved the state-of-the-art performance on many public datasets [47,45,16].

In sparse coding approaches, reconstruction costs are first pre-optimized to find the best combination of normal events, then testing samples causing higher costs are considered anomalous [15,16,18,22]. The problem lies in that sparse coding usually uses a very large dictionary, therefore, the optimization step, i.e., searching suitable basis vectors, is very time-consuming. In [16], Lu et al. proposed a method to accelerate the search speed, but they still needed to evaluate least square errors. Another example is that Zhu et al. [18] computed the sparse coding costs using Earth Mover's Distance (EMD) with a high computational complexity of $O(n^3 \log(x))$. Different from sparse coding approaches, clustering approaches detect the anomaly when the distance between input sample and its nearest cluster center exceeds a threshold and do not need take extra time to optimize basis vectors [20,21,23,24].

In clustering algorithms, Artificial Neural Network (ANN) has shown excellent performance with its ability on self-learning [25]. Among related algorithms for analyzing human behaviors, SOM has been successfully used in online systems, e.g., [27,24,28]. SOM performs competitive neuron learning [26], which is suitable for anomaly detection. The key competitive mechanism is that major events can be learned very well due to their strength in numbers while rare events are usually in conflict with the majority. To promote SOM for handling changing scenes, Feng et al. [24] proposed an online model of SOM, which adjusts the learning rate and neighborhood size during training the network. They tried to encode the primary behavior space into an 8×8 lattice of neurons after thousands of training epochs. In experiments, their model performed successfully on detecting local anomalies in crowded outdoor scenes.

As we mentioned, SOM has pre-structured lattice and its network size can not be changed. Comparatively speaking, GNG can quickly adjust the network topology and size according to input data. To make GNG more efficient for modeling changing scenes, we propose an online GNG model in the training stage and use a double Gaussian window method to identify anomalous events automatically in the testing stage. Quantitative results on three widely used anomaly datasets, namely UMN dataset [47], UCSD Ped dataset [45] and Avenue dataset [16], validate that our approach outperforms most of the state-of-the-art unsupervised works for detecting both global and local anomalies.

The rest of this paper is organized as follows. Section 2 briefly introduces original GNG and discusses self-adaptive proposals for online GNG. In Section 3, three neighbor-related strategies and a dynamic stopping criterion are proposed to update parameters for online GNG systemically. Local pattern descriptor and anomaly judge rule are supplemented in Section 4. Section 5 demonstrates the experiments on three recent datasets of human behaviors in public surveillance scenes. Conclusions are given in Section 6.

2. Original GNG and self-adaptive proposals for online GNG

In this section, we firstly introduce the algorithm of original GNG. Then, we discuss how to extend it to an online self-adaptive model.

2.1. Algorithm of original GNG

Growing Neural Gas (GNG) is an extension of neural-gas [31]. It aims to find the optimal representation of feature vectors. Its name comes from the behavior of feature vectors during the adaptation process that distributes them like gas in space. This paper relies on the original version of GNG introduced by Fritzke [32]. GNG network is composed by:

- A set of neurons denoted as $\mathcal{A}$ (the colorful dots shown on the right of Fig. 1). Each neuron $i \in \mathcal{A}$ has a weight vector $\mathbf{w}_i \in \mathcal{R}^D$. Each $\mathbf{w}_i$ has the same dimension D with the feature vector of input pattern (the small black points shown on the right of Fig. 1).
- A set of edges between pairs of neurons are denoted as C (the black lines between colorful dots shown on the right of Fig. 1). Edges are not weighted, and their sole purpose is to define the topological structure of neurons. An edge aging scheme is used to periodically remove edges that are invalid due to the motion of neurons, and then remove isolated neurons without any emanating edge.

Table 1 shows some dominant notations to be used.

1. Start the GNG network with two neurons a and b which have random weight vectors $\mathbf{w}_a$ and $\mathbf{w}_b$ selected from feature space $\mathcal{R}^D$.
2. Input a new pattern with feature vector $\mathbf{x}$, $\mathbf{x} \in \mathcal{R}^D$.
3. Find the nearest neuron s_1 (the winner) and the second nearest neuron s_2 (the second winner) by

$$s_1 = \arg\min_{s \in \mathcal{A}} \| \mathbf{x} - \mathbf{w}_s \|_2 \tag{1}$$

$$s_2 = \arg\min_{s \in \{\mathcal{A} \setminus s_1\}} \| \mathbf{x} - \mathbf{w}_s \|_2 \tag{2}$$

where Euclidean distance is used as the metric.

4. Increase the age of all edges emanating from s_1.
5. Add the distance between $\mathbf{x}$ and s_1 to error variable e_{s_1}:

$$e_{s_1} := e_{s_1} + \| \mathbf{w}_{s_1} - \mathbf{x} \|_2 \tag{3}$$

6. Move s_1 and its direct neighbors n ($n \in \mathcal{N}_{s_1}$) towards $\mathbf{x}$ by multiplying fixed learning rates ϵ_{s_1} and ϵ_n:

$$\mathbf{w}_{s_1} := \mathbf{w}_{s_1} + \epsilon_{s_1} \cdot \| \mathbf{x} - \mathbf{w}_{s_1} \|_2 \tag{4}$$

$$\mathbf{w}_n := \mathbf{w}_n + \epsilon_n \cdot \| \mathbf{x} - \mathbf{w}_i \|_2, \quad \forall\, n \in \mathcal{N}_{s_1} \tag{5}$$

Note that e_{s_1}, $\mathbf{w}_{s_1}$, $\mathbf{w}_n$ are time-varying variables, and this paper uses "$:=$" to indicate their updates following Fritzke's work [33].

Table 1
Algorithm notations.

$\mathbf{x}$	The feature vector of an input behavior pattern, and $\mathbf{x} \in \mathcal{R}^D$. For concision, this paper uses $\mathbf{x}$ to represent the pattern
$N_{\mathcal{A}}$	The number of neurons in $\mathcal{A}$
λ	The number of input patterns in each training epoch
$\mathbf{w}_i$	The weight vector of neuron i. It has the same dimension with input feature $\mathbf{x}$, i.e., $\mathbf{w}_i \in \mathcal{R}^D$
e_i	The error variable of neuron i. It updates when i becomes the winner
$\mathcal{N}_i$	The set of direct topological neighbors of neuron i
$age(i, j)$	The age of the edge that connects neuron i and neuron j
ϵ_{s_1}	The learning rate of winner s_1
ϵ_n	The learning rate of winner s_1's direct neighbor n, where $n \in \mathcal{N}_{s_1}$
α	The error variable adjustment factor used in neuron insertion
β	The global factor for adjusting all error variables used when a training epoch ends (i.e., every λ inputs)

7. If s_1 and s_2 are connected by an edge, set the age of this edge to zero. If this edge does not exist, create it.
8. Remove edges with an age $age(i, j)$ that exceeds age_{max}. If it results in some neurons without any emanating edge, remove them.
9. If the number of input patterns so far is an integral multiple of λ (i.e., each training epoch contains λ patterns, and here λ is a fixed parameter), then insert a new neuron (e.g., the blue dot on the right of Fig. 1) to network as follows:
   (a) Select neuron q with the maximum error e_q.
   (b) Insert a neuron r halfway between q and its neighbor f which has the largest error e_f. Set $\mathbf{w}_r = (\mathbf{w}_q + \mathbf{w}_f)/2$.
   (c) Insert edges connecting r with q and f, and remove the original edge between q and f.
   (d) Decrease e_q and e_f by multiplying them by a fixed parameter α. Initialize the error of r with e_q.
10. Decrease all error variables by multiplying by a fixed parameter β which is usually very close to 1.
11. If the stopping criterion (e.g., the maximum number of iteration) is not yet fulfilled, go to Step 2. Else stop the learning, and return $\mathcal{A}$ and C.

Dominant neuron operations are as follows. Learning towards a sample occurs in Step 3-6 with fixed learning rates (*Learning*). In Step 7, the edge between the winner and the second winner of sample creates an induced Delaunay triangulation [34]. Step 8 eliminates the edge between neurons that no longer comprise this triangulation because their ending neurons have moved. This results in some isolated neurons to be deleted (*Deleting*). In Step 9, error variables determine where it is necessary to insert a neuron (*Inserting*). Finally, the learning is terminated (*Stopping*) when the stopping criterion is fulfilled.

2.2. Self-adaptive proposals for online GNG

Inserting: Original GNG inserts neuron only between existing neurons, assuming that all input patterns obey the same distribution. However, most crowds act non-cross behaviors with multiple modalities [49], for which internal network expansion is not efficient. It is necessary to insert outer neurons as in [36]. Moreover, online GNG preserves the inner insertion of original GNG, and the associated parameters, namely λ, α and β, should be carefully selected since they may influence model efficiency.

Learning: In original GNG, the winner and its neighbors learn input patterns at fixed learning rate. Article [50] indicates that appropriate learning rate is crucial for constructing fast and robust self-organizing networks, in particular, for handling time-varying data. This paper hence studies a self-adaptive scheme for learning rates, making the model converge to the data space more efficiently.

Deleting: During the learning of original GNG, if the number of inputs so far is an integer multiple of λ, edges with the age value that exceeds age_{max} are deleted and then neurons with no topological neighbor are removed. This paper tackles changing surveillance scenes which contain a mutable quantity of behavior patterns, therefore, the network can hardly assign age_{max} with no basis. In other words, the deletion rule is hard to set. To solve this problem, this paper proposes to delete neurons with relatively low "density" which is a novel variable to combine *the frequency of the neuron being a winner* with *the distances between the neuron and its neighbors*.

Stopping: Original GNG can use different stopping criteria such as the maximal number of neurons and the minimal error between two training epochs. However, these fixed criteria have not been evaluated. It may be hard to use them for measuring the convergence of changing surveillance data. This paper, therefore, proposes a dynamic stopping criterion to measure whether the current network learns the data space well enough.

3. Online GNG

Above self-adaptive proposals compose the core idea of online GNG. This section first introduces three neighbor-related strategies to realize the proposals of inserting, learning and deleting, respectively. Related parameters get adjusted in a time-dependent manner to ensure the network best modeling the current data space. Then, a variation of Cluster Validity Index (CVI) [56] called Silhouette Index [55] is used to measure the network performance and to set a dynamic stopping criterion, i.e., for the last proposal.

3.1. Neuron insertion

Unlike original GNG, online GNG expands the network by additionally inserting outer neurons. First, the neighbor-related distance threshold will be defined for each neuron following Shen's work [36]. Then, distance thresholds will determine whether an input pattern is distinctive enough to be an outer neuron or not.

When a pattern with its feature vector $\mathbf{x}$ comes in, online GNG finds the winner s_1 and the second winner s_2, following Step 3 of original GNG. Then, it judges whether this pattern is distinctive enough from s_1 or s_2 by comparing with the distance threshold T computed as follows:

If s_i, where $i = 1, 2$, has directly connected neighbors, then the distance threshold T_{s_i} is calculated using the maximum distance between s_i and its neighbors:

$$T_{s_i} = \max_{n \in \mathcal{N}_{s_i}} \| \mathbf{w}_{s_i} - \mathbf{w}_n \|_2 \tag{6}$$

where $\mathbf{w}_{s_i}$ is the weight of s_i, and $\mathcal{N}_{s_i}$ is the neighbor set. If neuron s_i has no neighbor, the distance threshold T_{s_i} is defined as the minimum distance between s_i and other neurons:

$$T_{s_i} = \min_{n \in \mathcal{A} \setminus \{s_i\}} \| \mathbf{w}_{s_i} - \mathbf{w}_n \|_2 \tag{7}$$

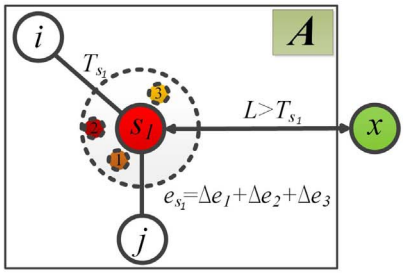
During learning, online GNG updates a neuron's distance threshold when it becomes the winner or second winner. Assuming that there are $N_{\mathcal{A}}$ neurons in the network, the costs of traversal comparisons are no more than $4 \times N_{\mathcal{A}}$.

The outer neuron insertion rule is as follows. If the distance between a new pattern and its winner s_i, where $i = 1, 2$, is greater than threshold T_{s_i}, this pattern will be inserted as an outer neuron x to represent the first neuron of a new class. Its distance threshold T_x will be initialized as $+\infty$. In particular, Fig. 2(a) shows an example in which the distance between the input $\mathbf{x}$ and its winner s_1 is greater than T_{s_1}. Online GNG inserts an outer neuron x whose initial neuron weight is $\mathbf{x}$, as illustrated in Fig. 2(b). Otherwise, if the distance between $\mathbf{x}$ and s_i is shorter than T_{s_i}, $\mathbf{x}$ will be learned by s_1 and s_1's neighbors i and j, as shown in Fig. 2(c).

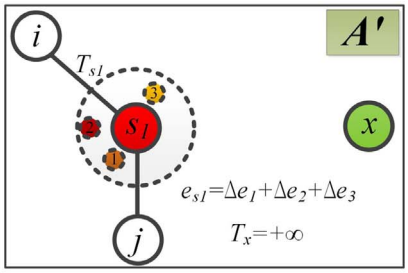
Besides inserting outer neurons, our online GNG also preserves inner insertion as in the original GNG, i.e., a new neuron is added every λ inputs and its directly connected neighbors decrease their error variables with a fixed parameter α, as given in Section 2.1 Step 9. To model changing scenes, however, online GNG uses unfixed λ and α. More specifically, λ is set following the speed of pattern input, e.g. the number of patterns observed every second by detectors like spatial–temporal interest points [41] and cuboids [42]. This scheme is simple, but it can combine scene changes directly with the model update. For parameter α, since it is tough to justify the theoretical value [37,38], this paper learns its candidate by performance validation on many samples. More details are given in algorithm in Section 3.5 and experiment in Section 5.1.

3.2. Learning rate adaptation

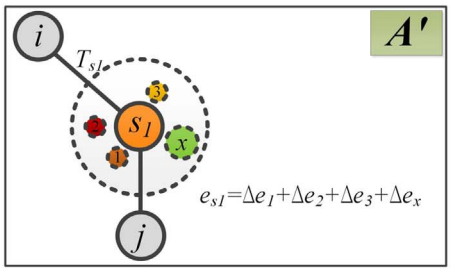
To learn the input pattern, the winner and its neighbors will change their weights by multiplying rates ϵ_{s_1} and ϵ_n. Previous works such as [24,36] indicate that using time-dependent learning rate is helpful for



(a) Distance L between $\mathbf{x}$ and s_1 is greater than the distance threshold T_{s_1}.



(b) Online GNG inserts $\mathbf{x}$ as an outer neuron.



(c) Original GNG changes the weights of s_1 and s_1's neighbors to learn $\mathbf{x}$.

Fig. 2. The difference between online GNG and original GNG for learning a distinct pattern. If the new pattern $\mathbf{x}$ is far away enough from its winner neuron s_1, online GNG will create an outer neuron based on $\mathbf{x}$ in (b), while original GNG just adjusts current neurons in (c). In these figures, A' is the next state of A. L indicates the distance from a new pattern $\mathbf{x}$ to its winner s_1. e_{s_1} is the error variable of s_1. Neurons i, j are direct connected neighbors of s_1. Patterns inside the dashed circles are previous data clustered to s_1.

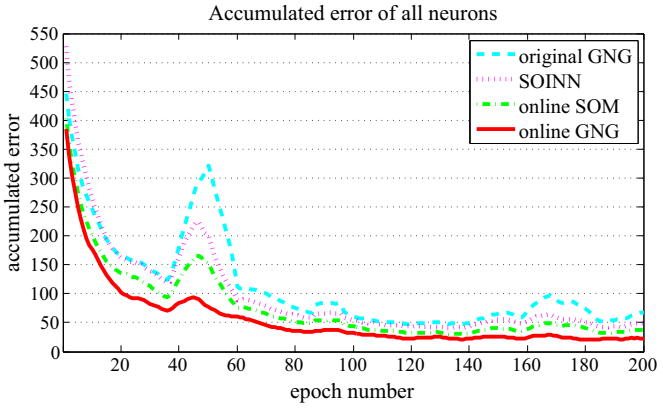


Fig. 3. Different accumulated errors (accumulated in each epoch) using different learning rates. Each epoch has 100 input samples extracted from the scene 1 of UMN dataset [47]. Initial accumulated errors are different since the first two GNG neurons are random. Note that all results are averaged over 10 runs.

modeling changing data. In the study of an improved GNG called Self-Organizing Incremental Neural Network (SOINN) [36], Shen and Osamu adopted a scheme similar to k-means to adjust learning rates as:

$$\begin{cases} \epsilon_{s1}(t) = \dfrac{1}{M_{s_1}(t)} & (8) \\ \epsilon_n(t) = \dfrac{1}{100M_n(t)}, \quad \forall\, n \in \mathcal{N}_{s_1} & (9) \end{cases}$$

where $M_{s_1}(t)$ represents the number of input patterns for which neuron s_1 has been a winner until time t. One time unit corresponds to each discrete insertion of input. This scheme slows down the learning process when s_1 becomes the winner for more and more patterns. It is hence useful to stabilize the weights of neurons which have gotten enough training. While this scheme focuses on learning frequencies, it does not decrease the error variables among irregular patterns which usually have different distances to their winners. In contrast, Feng et al. [24] adjusted the learning rate for online SOM using a feedback mechanism. Their approach took the effect of error variable into consideration and demonstrated that when an input is far away from its winner and causes a big error, it is necessary to increase learning rate to adapt to it.

Inspired by these ideas, we adjust learning rates considering both the frequency of neuron being a winner and the distance between the input and the winner. They are formulated as follows:

$$\begin{cases} \widetilde{\epsilon_{s_1}(t)} = \epsilon_{s_1}(t)\cdot e^{\frac{\|\mathbf{x}-\mathbf{w}_{s_1}\|_2}{T_{s_1}}} & (10) \\ \widetilde{\epsilon_n(t)} = \epsilon_n(t)\cdot e^{\frac{\|\mathbf{x}-\mathbf{w}_n\|_2}{T_n}}, \quad \forall\, n \in \mathcal{N}_{s_1} & (11) \end{cases}$$

where T_i has been defined in Eq. (6).

Eqs. (10) and (11) enable the network to adjust its neurons at appropriate step sizes. For example, if the distance between the input and its winner is large, the winner should adjust its weight fast. Meanwhile, if the winner is mature by winning enough number of patterns, this acceleration can be decreased.

For comparison, we record accumulated errors from different settings of learning rates for original GNG ($\epsilon_{s_1} = 0.1$, $\epsilon_n = 0.01$), SOINN (Eqs. (8) and (9)), online SOM (Eq. (7) in [24]), and online GNG (Eqs. (10) and (11)) respectively. Fig. 3 shows that online GNG gets the lowest accumulated errors in each training epoch. When samples at 40–60 epochs cause an interference, online GNG learns best and generates the smallest peak.

3.3. Neuron deletion

Neuron deletion is essential for learning human behaviors in changing scenes, e.g., crowd density gradually decreases from day to night. Besides, it also helps to reduce false negatives, i.e., missing detections. A prior assumption is that neurons caused by noises or anomaly events have significantly lower probability density than normal neurons. Shen et al. [36] identified low-density neurons by

defining a new density. During learning period, the frequency of the neuron being a winner is recorded and used as its density. If a neuron has fewer than two neighbors and has a lower density than the fixed threshold, it will be deleted.

Although *the frequency of being a winner* defining the neuron density seems reasonable, it has a problem that usually there are a significant amount of neurons in the high-density region, and the average chance for each neuron being a winner may not be higher than a neuron in the low-density region. Moreover, the fixed threshold is not suitable to segment the low-density region especially when the data space keeps changing. This paper proposes an improved strategy to achieve more reasonability. A *score* is defined to replace the *frequency*. It combines *the frequency of the neuron being a winner* with *the relationship between the neuron and its neighbors*.

Firstly, when neuron i is the current winner, the average distance $\overline{dist_i}$ between neuron i and its neighbors in set $\mathcal{N}_i$ can be calculated as:

$$\overline{dist_i} = \frac{\sum_{j\in\mathcal{N}_i} \| \mathbf{w}_i - \mathbf{w}_j \|_2}{|\mathcal{N}_i|}, \tag{12}$$

and the score of neuron i is calculated as:

$$score_i = e^{-\overline{dist_i}} \tag{13}$$

Eqs. (12) and (13) indicate that if neuron i is far from its neighbors, i.e., the region around i is sparse, i will get a low score for representing the characteristic of this low-density region. Otherwise, neuron i will get a higher score.

After K epochs of learning, the accumulated score as_i of neuron i is defined as:

$$as_i = \sum_{k=1}^{K}\sum_{j=1}^{\lambda} score_i(k, j), \tag{14}$$

where λ is the number of input patterns in each epoch.

The mean accumulated score in each epoch is as follows:

$$\overline{as_i} = \frac{1}{K}\sum_{k=1}^{K}\sum_{j=1}^{\lambda} score_i(k, j), \tag{15}$$

where $score_i$ is in the interval $(0, 1)$. Upon each input, only the score of winner i is updated, but the scores of other neurons j ($j \neq i$) are set to 0. In other words, the accumulated score as_i will be changed, but other as_j ($j \neq i$) remain unchanged. Finally, the network deletes the neuron j with $\overline{as_j} < \overline{as_{thr}}$, where $\overline{as_{thr}}$ is a self-adaptive threshold in our approach. An example of using a simple threshold $\overline{as_{thr}} = \sum_{j=1}^{N_\mathcal{A}} \frac{\overline{as_j}}{N_\mathcal{A}}$ to control neuron deletion is given in Fig. 4. It shows that our deletion strategy reserves high-score neurons. In Section 3.5, Step 9(f), $\overline{as_{thr}}$ will update online based on the stopping criterion in Section 3.4.

In summary, both original GNG and SOINN use fixed conditions for neuron deletion, e.g., original GNG uses a fixed age_{max}, and SOINN additionally deletes neurons with less than two neighbors. In contrast, our strategy depends on the frequency of the neuron being a winner as well as *the relationship between the neuron and its neighborhood* which are more objective and reasonable. More importantly, the only parameter $\overline{as_{thr}}$ is a self-adaptive parameter.

3.4. Stopping criterion

Online GNG updates network size continuously. To avoid overflowing, it is better to set a threshold on how many neurons are enough for representing current data space. Moreover, the threshold should be adjusted as time goes by for handling changing scenes. This paper presents a novel proposal of dynamic stopping criterion based on Cluster Validity Index (CVI) [56], which is usually used to measure the partition performance of clustering algorithm. This paper uses a variation of CVI called Silhouette Index (*Sil*) [55] for its outstanding performance in estimating sparse multidimensional clustering shown in [56].

Silhouette Index in conjunction with the network learning is formulated as follows:

$$Sil(t) = \frac{1}{N_\mathcal{A}(t)}\sum_{i\in\mathcal{A}}\sum_{\mathbf{x}\in\mathcal{W}_i(t)} \frac{\| d_1(\mathbf{x}, i; t) - d_2(\mathbf{x}, i; t)\|_2}{\max\{d_1(\mathbf{x}, i; t), d_2(\mathbf{x}, i; t)\}} \tag{16}$$

where $N_\mathcal{A}(t)$ is the number of neurons in $\mathcal{A}$ until time t, and $\mathcal{W}_i(t)$ represents the set of input patterns for which neuron i has been a winner. Meanwhile,

$$d_1(\mathbf{x}, i; t) = \frac{1}{|\mathcal{W}_i(t)|}\sum_{\mathbf{y}\in\mathcal{W}_i(t)} \| \mathbf{x} - \mathbf{y} \|_2 \tag{17}$$

indicates the *cohesion* measured by the distance between all the patterns assigned to neuron i, and

$$d_2(\mathbf{x}, i; t) = \min_{k\in\mathcal{A}\setminus\{i\}}\left\{\frac{1}{|\mathcal{W}_k(t)|}\sum_{\mathbf{z}\in\mathcal{W}_k(t)} \| \mathbf{x} - \mathbf{z} \|_2\right\} \tag{18}$$

indicates that the *separation* among neurons is measured by the nearest neighbor distance.

$Sil(t)$ measures the effectiveness of online GNG modeling on input data at time t. Based on this index, we propose a dynamic stopping criterion to decide when to stop changing the network size (but not stop the training). First, the CVI error $E_{sil}(t)$ is computed by:

$$E_{sil}(t) = Sil(t) - Sil(t-1) \tag{19}$$

When $E_{sil}(t)$ is continuously smaller than a tiny residual denoted as E_{thr}, i.e., $E_{sil}(t) \to 0$, it means input samples are well represented by the current number of neurons. Then, the threshold $\overline{as_{thr}}$ mentioned in Section 3.3 can be used to ensure that the number of deleted neurons equals to the number of inserted neurons. Details are given in Section 3.5, Step 9.

As illustrated in Fig. 5, online GNG network reaches a stable size after 200 training epochs. Note that our "stopping" is a dynamic replacement of neurons. Insertion and deletion synchronously regulate network states with time goes by.

3.5. Algorithm of online GNG in changing scenes

This section gives the complete algorithm of online GNG based on proposed self-adaptive strategies. It takes the scene-changing videos into account.

Table 2 shows some dominant notations to be used.

1. Initialize neuron set $\mathcal{A}$ to include a, b with $\mathbf{w}_a$, $\mathbf{w}_b \in \mathcal{R}^n$ randomly chosen from input pattern set $\mathcal{X}^{(1)}$, where superscript indicates the first video frame. Initialize connection set C, $C \subset \mathcal{A}\times\mathcal{A}$, to be empty. Set n_{new} and λ to be 0.
2. Extract a new feature vector $\mathbf{x}$ from frame k, i.e., $\mathbf{x} \in \mathcal{X}^{(k)}$. Input $\mathbf{x}$, and $\lambda := \lambda + 1$.
3. For $\mathbf{x}$, search for the winner s_1 and the second winner s_2 by Eqs. (1) and (2).
4. If the distance between $\mathbf{x}$ and s_1 (or s_2) is greater than the distance threshold T_{s_1} (or T_{s_2}) computed by Eq. (6) or (7), then $\mathbf{x}$ generates an outer neuron, i.e., if
$$\{\| \mathbf{x} - \mathbf{w}_{s_1} \|_2 > T_{s_1}\} \vee \{\| \mathbf{x} - \mathbf{w}_{s_2} \|_2 > T_{s_2}\} \tag{20}$$
then insert a new neuron x with $\mathcal{A} := \mathcal{A}\cup\{x\}$, $\mathbf{w}_x = \mathbf{x}$, $n_{new} := n_{new} + 1$, $T_x = \| \mathbf{x} - \mathbf{w}_{s_1} \|_2$ and go to Step 2 to process the next input. Else, go to the next step.
5. If a connection between s_1 and s_2 does not exist, create it and add it to C as $C := C \cup \{(s_1, s_2)\}$.
6. Add the distance between $\mathbf{x}$ and s_1 to error variable e_{s_1} by Eq. (3).
7. Update the mean accumulated score $\overline{as_{s_1}}$ of s_1 by Eqs. (12)–(15).
8. Adjust the weight vector of winner s_1 and its direct topological neighbor n ($n \in \mathcal{N}_{s_1}$) by multiplying $\widetilde{\epsilon_{s_1}(t)}$ and $\widetilde{\epsilon_n(t)}$ (Eqs. (10) and

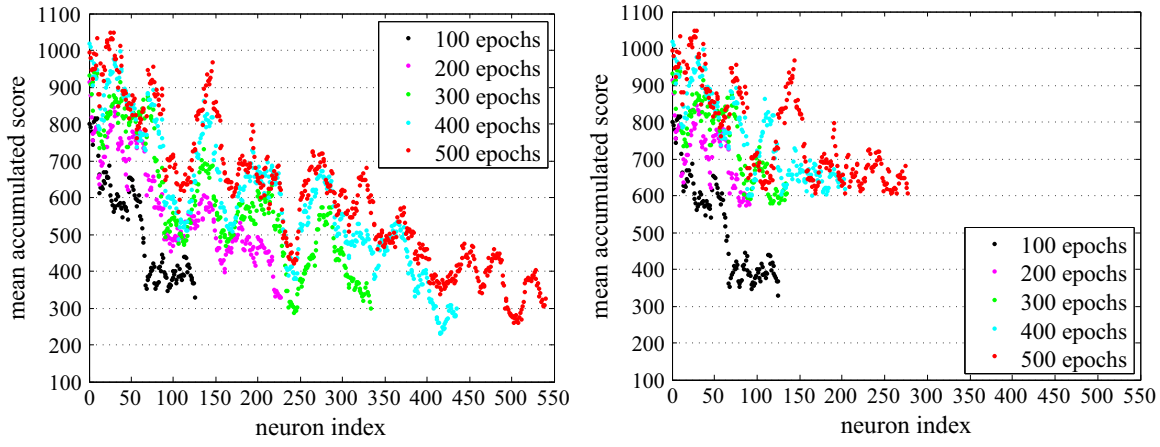


Fig. 4. Neuron deletion with a threshold $\overline{as_{thr}} = \sum_{j=1}^{N_{\mathcal{A}}} \frac{\overline{as_j}}{N_{\mathcal{A}}}$. Important notes are as follows. (1) The "neuron index" on x-axis implies that neurons are generated one by one, e.g., the 450th–540th neurons in (a) are generated after 400 training epochs. (2) The x-axis value of each curve's endpoint indicates the current quantity of neurons in the network after hundreds of training epochs. For example, as shown in (b), about 100 neurons are preserved after 200 training epochs. (3) "neuron index=100 and mean accumulated score=560 on the red curve in (a)" indicates when the network is trained for 500 epochs, the mean accumulated score of the 100th neuron is 560. (4) Each epoch contains 100 patterns extracted from UCSD Ped1 dataset [45]. All presented results are averaged over 10 runs. (5) It is shown in (b) that our neuron deletion strategy only preserves high-score neurons, e.g., only 300 neurons are preserved after 500 epochs.

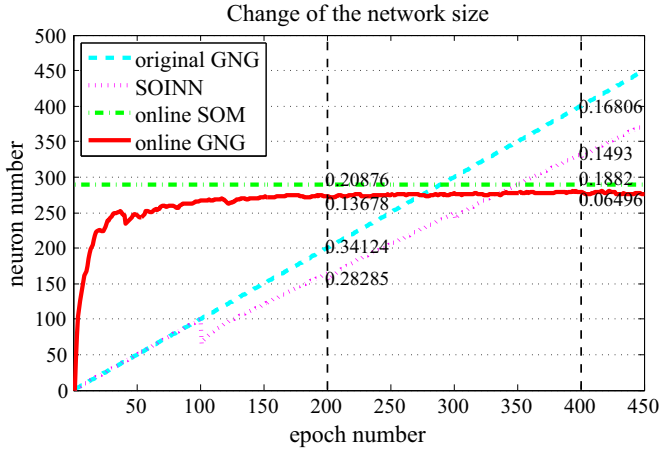


Fig. 5. The change of the network size is recorded for each model. Training samples are from UCSD Ped1 dataset [45]. For comparison, online SOM is set to a 17×17 lattice of neurons, close to the gradually stabilizing size of online GNG. Note that all results are the average of 10 runs, and average accumulated errors are marked at 200, 400 epochs.

Table 2
Algorithm notations.

$\overline{as_i}$	The local mean accumulated score of neuron i. It is updated when i is the current winner
T_i	The distance threshold of neuron i. If the distance between an input pattern and neuron i is larger than T_i, the input pattern generates a new neuron
n_{new}	The number of newly added neurons in each training epoch
$\widetilde{\epsilon_{s_1}(t)}$	The learning rate of winner neuron s_1 at time point t
$\widetilde{\epsilon_n(t)}$	The learning rate of winner neuron s_1's direct neighbor neuron n ($n \in \mathcal{N}_{s_1}$) at time point t
α	The error variable adjustment factor used in neuron insertion
γ	The mean accumulated score adjustment factor used in neuron insertion

(11)) by the distance between $\mathbf{x}$ and s_1 as follows:

$$\mathbf{w}_{s_1} := \mathbf{w}_{s_1} + \widetilde{\epsilon_{s_1}(t)} \cdot (\mathbf{x} - \mathbf{w}_{s_1}) \tag{21}$$

$$\mathbf{w}_n := \mathbf{w}_n + \widetilde{\epsilon_n(t)} \cdot (\mathbf{x} - \mathbf{w}_n), \quad \forall\, n \in \mathcal{N}_{s_1} \tag{22}$$

9. If the patterns on current frame k have been completely observed, i.e., λ increases to $|\mathcal{X}^{(k)}|$, insert an inner neuron and delete low-density neurons as follows:
   (a) Find the neuron q with the maximum error e_q.
   (b) Among the neighbors in set $\mathcal{N}_q$, find the neuron f with the maximum error variable e_f.
   (c) Add neuron r to neuron set $\mathcal{A} := \mathcal{A} \cup \{r\}$, $n_{new} := n_{new} + 1$, and tween-interpolate its weight vector as: $\mathbf{w}_r = (\mathbf{w}_q + \mathbf{w}_f)/2$.
   (d) Insert edges connecting new neuron r with q and f, and remove the original edge between q and f: $C := C \cup \{(q, r), (r, f)\}$, $C := C \setminus \{(q, f)\}$. Initialize e_r and $\overline{as_r}$ respectively with e_q and $\overline{as_q}$.
   (e) Decrease the error variable of q and f by fraction α: $e_q := \alpha e_q$, $e_f := \alpha e_f$. Meanwhile, decrease the mean accumulated score of q and f as: $\overline{as_q} := \gamma \overline{as_q}$, $\overline{as_f} := \gamma \overline{as_f}$.
   (f) *The first situation* with $\overline{as_{thr}}$ being null: Compute $E_{sil}(t)$ by Eqs. (16)–(19). If $E_{sil}(t) \geq E_{thr}$, then rank the mean accumulated scores of all neurons in a decreasing order, and delete the bottom neuron i and its edges. That is $\mathcal{A} := \mathcal{A} \setminus \{i\}$, and $C := C \setminus \{(i, *)\}$, where $(i, *)$ indicates the edges connected to i. Else if $E_{sil}(t) < E_{thr}$, rank the mean accumulated scores of all neurons in a decreasing order, and delete the bottom n_{new} neurons and their edges. Update $\overline{as_{thr}}$ with the minimum mean accumulated score in current network, i.e., $\overline{as_{thr}} = \min_{i \in \mathcal{A}} \{\overline{as_i}\}$.
   The second situation with $\overline{as_{thr}}$ assigned: Compute $E_{sil}(t)$ by Eqs. (16)–(19). If $E_{sil}(t) \geq E_{thr}$, compare the mean accumulated score of all neurons with $\overline{as_{thr}}$. Deletes all neurons j when $\overline{as_j} < \overline{as_{thr}}$, and remove their edges. Else if $E_{sil}(t) < E_{thr}$, do the same operations of *The first situation*.
   (g) Set both n_{new} and λ to 0.
10. Go to Step 2 to continue the training. When it comes to the testing stage, return current $\mathcal{A}$ and C to give an anomaly judge on input patterns and take in normal patterns for training.

Note that online GNG works in an online training-testing mode, and there is no real stop in Step 10. The "stop" here indicates the network size stops expanding, meanwhile, the network keeps on training.

4. Supplements to anomaly detection

In this section, we introduce the pattern descriptors used in surveillance scenes. Then, we define the anomaly judge rule in quantity

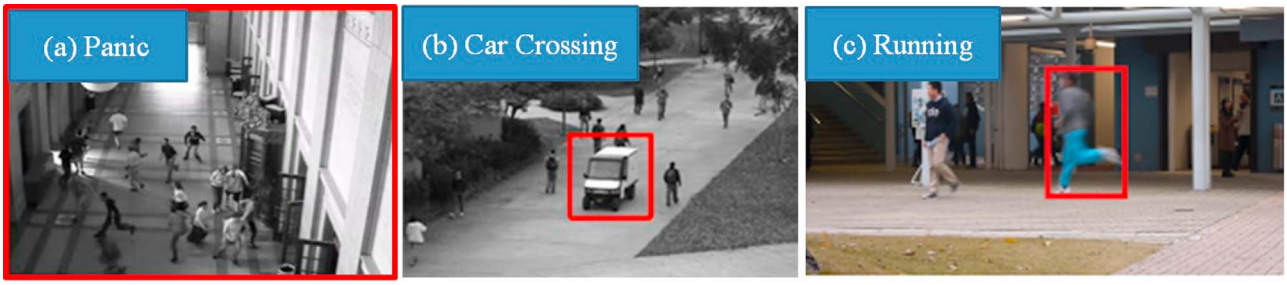


Fig. 6. GAE refers to the cases such as people running away in panic [47] in (a). LAE refers to the distinctive events such as a car crossing sidewalk [45] in (b) and a man running fast across subway entrances. Red grids mark the anomalous events.

according to general assumptions that anomaly means *rare* or *distinctive* events in contrast with normal ones.

4.1. Pattern description

Abnormal events can be divided into two categories [15]: global abnormal event (GAE), where the whole scene is unnatural like group panic; local abnormal event (LAE), where the local behavior is different from its neighbor regions. Both are illustrated in Fig. 6.

A reliable descriptor of behavior pattern is critical for distinguishing different behavior classes. Many image descriptors are available, as shown in the survey article [4]. This paper focuses on public surveillance scenes, where there are crowds of people, and it is very hard to segment individuals [46]. In our previous works [27,29,30], we utilized local motion descriptors [45,40,42] for action recognition. Moreover, such descriptors have been successfully tested in similar cases of anomaly detection [24,46]. Although our approach in this paper can be applied regardless of the elaborate choice of pattern descriptor, we select different local motion descriptors for different surveillance scenes to benefit from the accuracy they bring. Other descriptor analysis for anomaly detection can be found in a newly published article [62]. Sparse interest points (STIP) [40] are utilized for describing GAE scenes such as the crowd panic in Fig. 6(a). In contrast, dense cuboids [42] tend to describe more details than STIP. It is hence chosen for detecting the LAE in the scenes like Fig. 6(b) and (c).

4.2. Anomaly judge rule

Anomaly detection has the assumption that online GNG summarizes the high-density region of behavior space into a limited number of neurons, and it is hard for an anomalous pattern to be close to any one of these neurons [24]. Based on this assumption, we then define the anomaly judge rule by using Gaussian smoothing window as follows.

The distance between the input sample and its winner is assumed to be conforming to the Gaussian distribution whose mean $\mu_{feature}$ and variance $\sigma_{feature}$ are easily computed. To decide whether a new pattern is anomalous, we first find its winner and record the distance between them. If this distance is larger than the *overflowing threshold*:

$$th_{overflowing} = \mu_{feature} + 2\sqrt{\sigma_{feature}} \tag{23}$$

then this pattern will be labeled as an anomalous pattern. When detecting GAE, very few anomalous patterns are insufficient to make the conclusion, especially in noisy scenes. Therefore, we confirm the abnormity after observing *enough* anomalous patterns, i.e., if the number of anomalous patterns goes over *anomaly threshold*:

$$th_{anomaly} = \mu_{overflowing} + 2\sqrt{\sigma_{overflowing}} \tag{24}$$

then the GAE is confirmed. In this definition, $\mu_{overflowing}$ and $\sigma_{overflowing}$ respectively represent the mean and the standard deviation of anomalous pattern quantities. This threshold involves the conservative estimate of noises, and thus can resist the false alarms caused by sparse noises.

For LAE detection, the input frame is split into grids, and dense cuboid features are extracted in each grid. Then, above GAE detection method is applied to a grid to decide whether it is anomalous or not. Frame 1 and frame 2 in Fig. 7(b) contain some abnormal patterns which go over $th_{overflowing}$ but cannot cause anomalous events. Specifically, small-region noises on flickering light (frame 1), walking pedestrians and shaking vegetation (frame 2) are filtered out for not having enough anomalous patterns. Fig. 7(b) marks the abnormal events detected by $th_{anomaly}$ in red grids.[1]

4.3. Relationship with online SOM

Both online SOM [24] and online GNG are based on the competitive neural networks. Online SOM organizes a certain number of neurons in a lattice, and its *online* indicates that the neighborhood size and learning rate continue updating. It ignores the discussion of using different network sizes and topology. In contrast, GNG network has a more flexible topological structure, and the network size can be easily changed online. Moreover, online GNG achieves the update of all dominant learning parameters using neighbor-related strategies with clear statistical meanings. It exploits the advantages of both online SOM and original GNG, and can achieve better performance theoretically.

5. Experiments and discussions

Our anomaly detection approach has two conceptual stages: the training stage of online GNG and the testing stage using anomaly judge rule. Experiments are implemented in 3 public datasets filmed in changing surveillance scenes. UMN dataset [47] is used to detect GAE, while UCSD Ped1 dataset [45] and Avenue dataset [16] are used for LAE detection. The frame-level measurement is used to evaluate GAE detection while both frame-level and pixel-level measurements are used for the evaluation of LAE detection. Both measurements are defined in [45] that if more than 40% of ground truth anomalous pixels are detected, the corresponding frame or grid is considered as a correct detection.

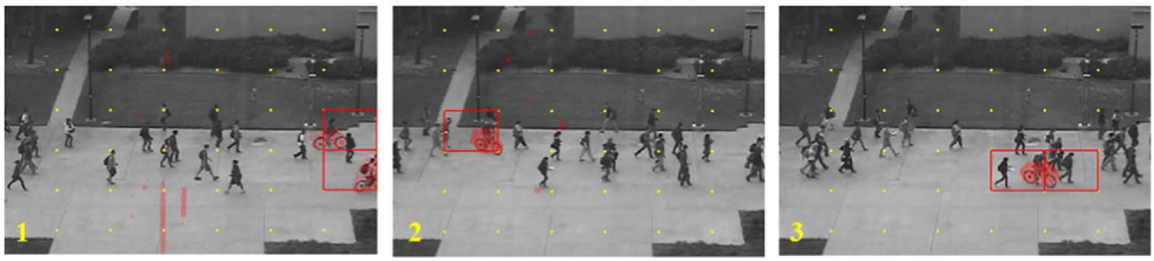
We demonstrate quantitative comparisons with 12 recent approaches [44,24,45,48,15,16,61,60,18,51,53,52], which are tested on the datasets in common. We show our approach outperforms theses approaches in terms of Receiver Operating Characteristic (ROC) [45], Area Under ROC Curve (AUC) [45], number of correct detections [16], and number of false alarms [16].

5.1. Parameter preferences

In GNG related works [24,36], researchers attempt to adjust

[1] For the statistic of anomalous pixels, only red pixels inside red boxes are counted. In following video snapshots, we omit red pixels for clearness, and only use red grids to indicate the locations of anomalous events.

(a) Red pixels detected by $th_{overflowing}$ are anomalous patterns.



(b) Red grids detected by $th_{anomaly}$ are confirmed as anomalous events.

Fig. 7. UCSD Ped2 anomaly samples detected by using $th_{overflowing}$, then using $th_{anomaly}$. Red marked pixels in (a) and (b) indicate the overflowing features (anomalous patterns). In (b), LAEs in red grids are confirmed by $th_{anomaly}$, and red pixels outside the grids are background noises. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

dominant parameters like learning rates ϵ_{s1}, ϵ_n, and keep some parameters like α, β unchanged. In online GNG, neuron insertion and deletion involve following undetermined parameters: λ, α, γ and E_{thr}. These parameters may affect our model efficiency. Hence, we propose specific schemes for parameter preferences.

At first, λ represents the number of input samples in each epoch and controls the neuron insertion period of original GNG. However, the insertion of online GNG dominantly relies on outer neuron selection, making the periodical influence brought by λ extremely weak. On the other hand, λ controls the neuron deletion frequency of online GNG. Although our neighbor-related strategy removes neurons based on network topology so that candidates in low-density regions tend to be eliminated, λ might make a difference before generating a mature network topology. This paper uses the pattern number on each video frame to define λ, as shown in Section 3.5, Steps 1, 2, 9, and 11. It means that network trimming is following the pace of observation, e.g., fewer patterns motivate a higher frequency of trimming, yielding fewer neurons to represent fewer patterns. The average pattern number of consecutive 5 frames is used for each frame to prevent the frame drop caused by the failure of feature detectors.

For fractions α and γ, we refer to the conclusions of corresponding β and γ in Shen's works [36,39]. They analyzed that Voronoi regions method could be used to determine that $\alpha = 2/3$ and $\gamma = 3/4$ under a supposition that signals are uniformly distributed over Voronoi regions. However, this supposition is untenable for some tasks, and resulting parameters may not be optimal. Fortunately, they drew the conclusion that the model is not very sensitive to the choice of these two parameters through extensive experimental validation on both stationary and non-stationary environments. This paper extends Shen's experiments, and tests $(\alpha, \gamma) = \{(2/3, 3/4), (2/3, 1/2), (1/2, 1/2), (1/4, 3/4)\}$ by performance validation on behavior data. We select the optimal pair that brings the minimum accumulated error after the first thousand of training epochs. Finally, $\alpha = \gamma = 1/2$ are selected for UMN and UCSD Ped1/Ped2 datasets, and $\alpha = 2/3$, $\gamma = 1/2$ for Avenue dataset.

Finally, we have an arbitrary parameter E_{thr}, which is a tiny residual and works in the Step 9(f) of online GNG algorithm. We set its value to be 0.0001 for all datasets.

For original GNG, there is no parameter self-adaptive scheme. Its constant parameters in Section 2.1 are set as follows: $\epsilon_{s1} = 0.1$, $\epsilon_n = 0.01$, $age_{max} = 200$, λ=100, $\alpha = 1/2$, and β=0.99.

5.2. Tests on UMN dataset

UMN dataset contains 2, 6, 3 panic events respectively in three indoor and outdoor scenes. Each event starts with some typical behaviors such as walking and hand waving, then ends with a short emergent panic. Normal/anomalous events and corresponding local features are shown in Fig. 8. On this dataset, online GNG training begins just as the feature vectors are input, and so does the testing. In the beginning, neurons are learning by first coming normal features. When a panic pattern comes, it will be figured out immediately since it is far from existing neurons. If a number of such patterns are observed, an anomaly is detected. In the meantime, online GNG keeps on training, in which neuron insertion and deletion cooperate to avoid anomaly patterns invading the model.

Statistics and final detection results of 3 scenes are shown in Fig. 9. It can be seen that *overflowing threshold* (blue line) and *anomaly threshold* (red line) fluctuate. For example, in Fig. 9(a), *overflowing threshold* and *anomaly threshold* are first learned with the features on frames 1–450, then they suddenly go up when the anomalous patterns on frames 450–600 come. After this short disturbance, normal patterns on frames 600–1300 are input, making these thresholds decline to their original levels gradually. Therefore, input data affect the anomaly threshold directly and automatically.

As shown in Fig. 9, *anomaly value* indicates the anomaly state of each video frame. Colors except blue denote the occurrence of GAE. The closer to red, the severer GAE is. The ground truth by manual labeling and the result of original GNG are given for comparison. It is clear that online GNG makes less false alarms than original GNG. Especially, online GNG avoids many false alarms in the beginning, because online GNG can insert outer neurons and quickly expand the network when different classes are observed for the first time. However, missing detections occasionally occur, for example after the 1400*th* frame in scene 1. This is because people gradually escape out of view in the video, leaving behind too few patterns being observed to

Fig. 8. Three crowd scenes on UMN dataset. Yellow circles represent the local features extracted by Laptev's detector [41]. Note that the anomaly marks in the second row are from raw videos. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

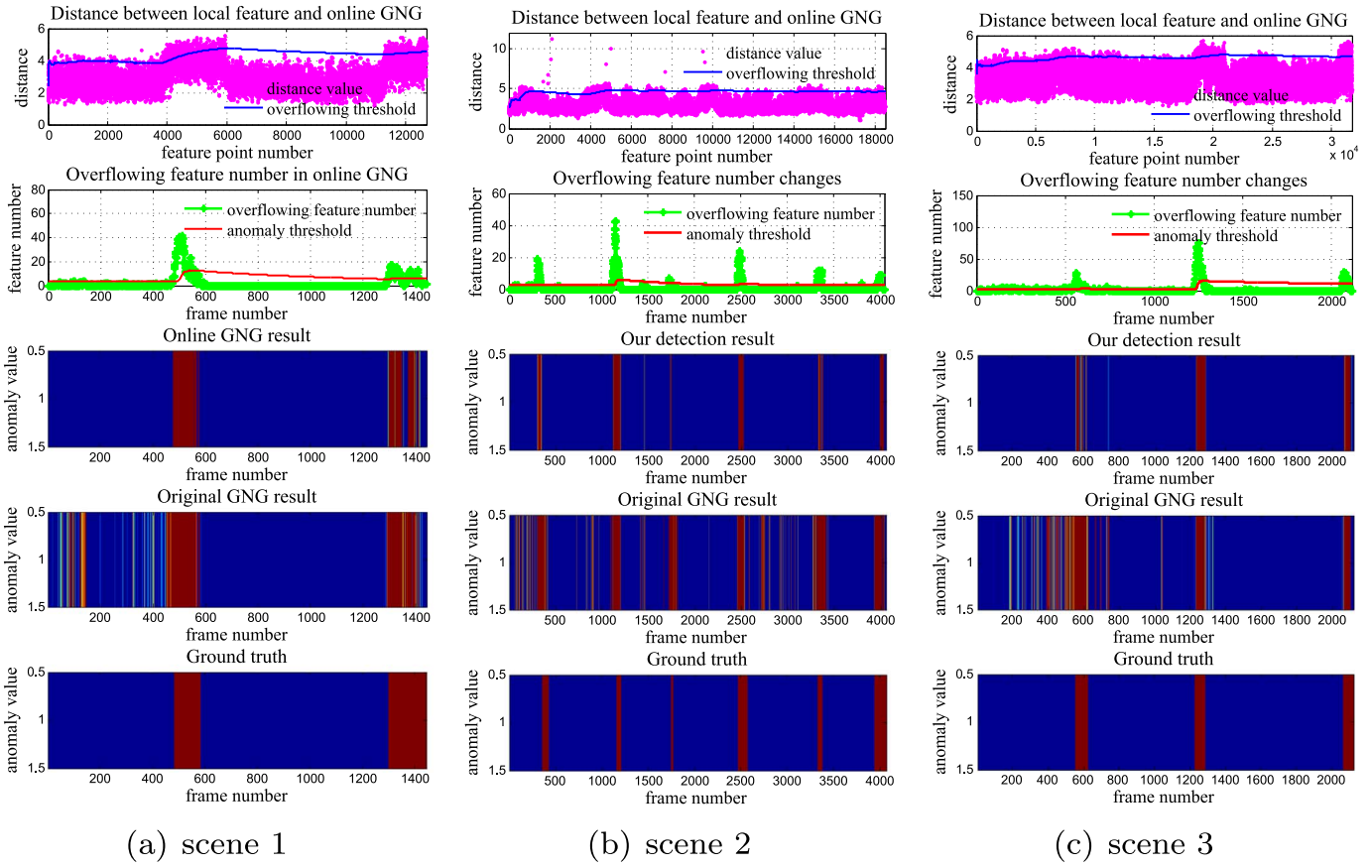


Fig. 9. Our anomaly detection results compared with original GNG in UMN scenes 1–3.

Table 3
Comparing AUCs with the state-of-the-art approaches in UMN dataset. The simple acronyms in brackets are used for other experiments.

Approach	Area under the curve
Chaotic invariants 2010 [48]	0.994
Non-negative sparse coding 2014 [18]	0.997
Social force (SF 2009) [44]	0.949
Hierarchical MDTs with CRF (MDT 2013) [45]	0.995
Sparse reconstruction cost (SRC 2013) [15]	0.996
Unsupervised kernel learning (CCUKL 2014) [51]	0.980
Histogram with SVDD model (Hist+SVDD 2016) [53]	0.9833
Online Structural Analysis (OADC-SA 2015) [52]	0.9967
Online GNG scene 1	0.9980
Online GNG scene 2	0.9928
Online GNG scene 3	0.9987
Online GNG average	0.9965

exceed *anomaly threshold*.

In this test, original GNG is limited to have no more than 300 neurons, and in fact, it always reaches this maximum number. In contrast, online GNG learns a necessary number of neurons with slight fluctuation around the average quantities 89, 93, and 112 in 3 scenes, respectively. It uses fewer neurons but obtains better performance. Moreover, the average distances between all observed patterns and winners are 7.6367 in online GNG and 21.8853 in original GNG, indicating that online GNG converges to the behavior space better.

For quantitative comparison, we present the results of Area Under Curve (AUC) in Table 3. AUC results of [44,48,45,15,18,51,53,52] are also given. The AUCs of our approach vary from 0.9928 to 0.9987 with the average 0.9965 which is lower but comparable to the highest record 0.997 in [18]. Zhu et al. [18] computed sparse coding costs using Earth Mover's Distance (EMD) [19] with a complexity of $O(n^3 \log(x))$. Yuan et al. [52] proposed a robust multi-object tracker, and they also used

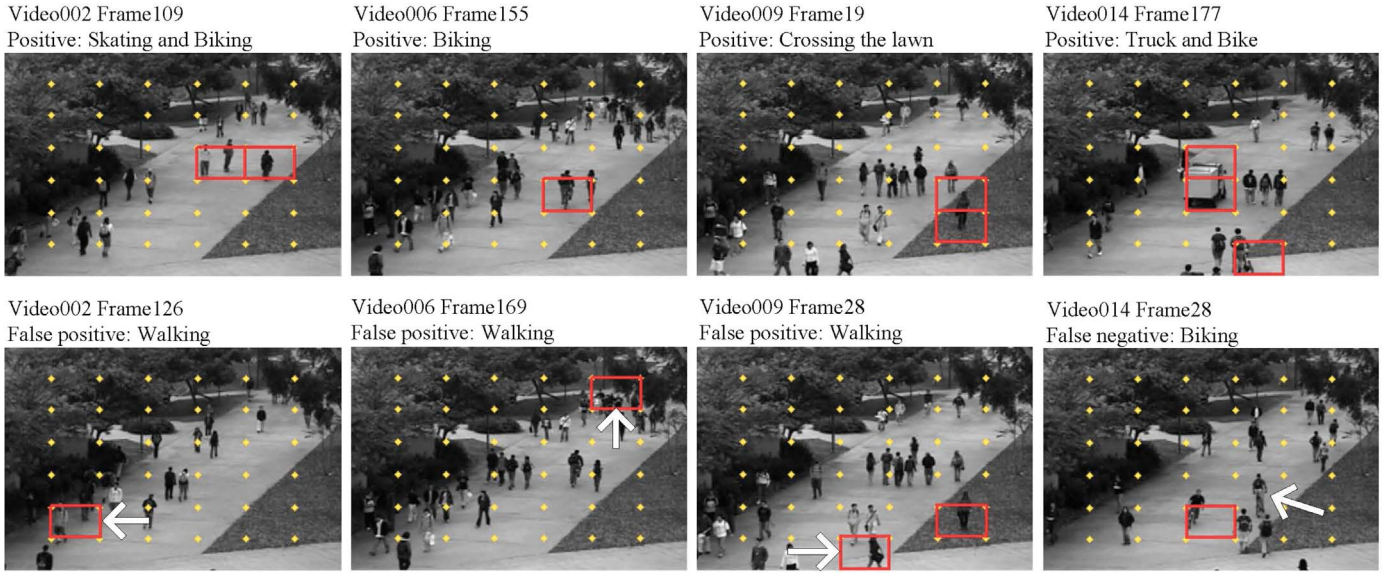


Fig. 10. Example results on UCSD Ped1. Although there are some false alarms and missing detections (pointed out by white arrows), most anomalous events such as biking, skating, car crossing and their co-occurrences are well detected. Note that red pixels are omitted for clearness. More results can be found in our Supplementary demo.

EMD measurement for anomaly detection. Though both of them achieved slightly better results than ours, we use a simple Euclidean distance which yields a much faster computational speed.

5.3. Tests on UCSD Ped1/Ped2 dataset

UCSD Ped1/Ped2 datasets have been well tested in recent works [15,16,44,60,61,45,51–53]. Ped1 scene contains 70 crowd clips with 34 clips of normal-only behaviors for training and other 36 clips for testing. Videos are acquired with a stationary camera. Ped2 scene is with pedestrian movement parallel to the camera plane, containing 16 training video samples and 12 testing video samples. In both scenes, pedestrian density changes from sparse to very dense. Anomaly indicates the local occurrence of the non-pedestrian entity in walkways such as truck crossing, people skating and biking. On this dataset, model training starts using normal clips, and it goes on by using the normal events observed in the testing stage.

Unlike UMN dataset, Ped1/Ped2 scenes both contain the local anomaly events (LAE). Hence, each frame is divided into 7×7 grids following [15]. Dense cuboid features [42] are detected on each grid. If a grid contains a number of anomalous features over *anomaly threshold*, it is labeled as LAE.

Example snapshots of successful/failed detections in Ped1 scene are shown in Fig. 10. Our approach detects most of the distinctive LAEs such as skating, riding a bike, crossing the grassland, and car crossing. Wrong detections and undetected events are marked with white arrows. They are possibly due to body occlusions and shadows, which might be alleviated by using sophisticated crowd features, e.g., the gait feature in conjunction with the temporal consistency of local appearance [54]. Our online model does not use this feature for its high computational complexity.

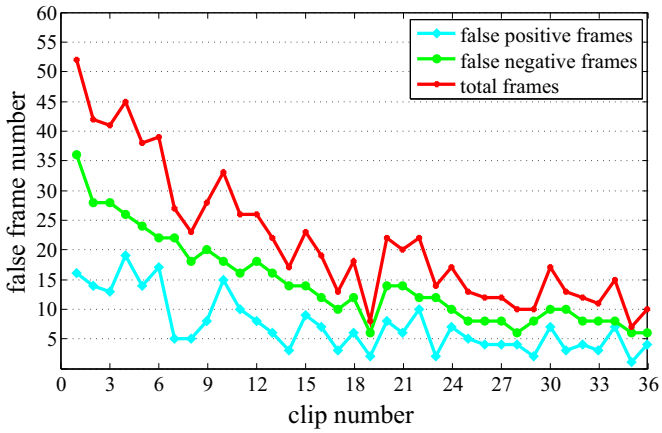


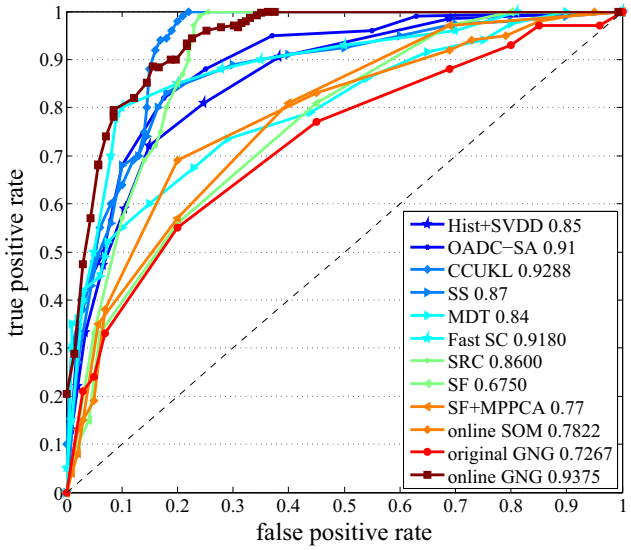
Fig. 11. The number of false frames drops off during the training-testing procedure with more and more video clips input.

In Fig. 11, we count the number of failure frames in 36 testing clips. It is worth noting that cyan and green curves show many fluctuations due to the different complexities of different clips. The stable tendency is that failure frames in later stages become much less than those in the beginning, which can be figured out from *total frames* dropping off from 55 (in clip-1) to 10 (in clip-36). The reason is that our online model does not stop learning even in testing stages, i.e., initially trained using 34 normal clips, then continues this training during the testing of 36 clips. Our model converges toward the behavior space better and better as long as there is new pattern observation.

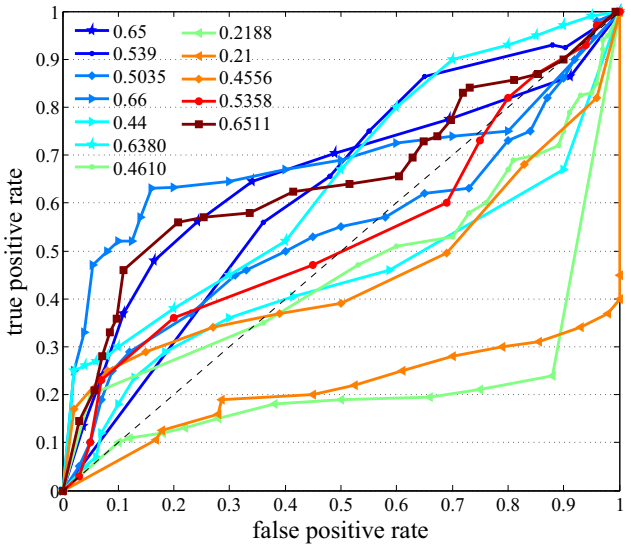
Following [16,45,51,53,52], we plot the Receiver Operating Characteristic (ROC) curves using frame-level and pixel-level measurements, as given in Fig. 12. Involved approaches are with following abbreviations: Hist+SVDD 2016 [53], OADC-SA 2015 [52], CCUKL 2014 [51], SS 2013 [61], MDT 2013 [45], Fast SC 2013 [16], SRC 2013 [15], SF 2009 [44], SF+MPPCA 2009 [60], online SOM 2010 [24], original GNG, and our online GNG.[2]

In Fig. 12, it is clear that our online GNG demonstrates the best performance in both scenes. The main reason could be that most other approaches use limited training samples for modeling while our online model keeps on training to improve itself also in the testing stage. Our self-adaptation strategies take full advantage of current input data to update dominant learning parameters. Therefore, they help to reduce false alarms caused by model aging. Meanwhile, the strategy of neuron deletion significantly reduces false negatives resulting from anomalous neurons. Particularly, Fig. 12(a) and (c) show that our approach achieves higher AUCs over the state-of-the-art Unsupervised Kernel Learning with Clustering Constraint model (CCUKL) [51] and Semiparametric Scan (SS) model [61] by frame-level measurement.

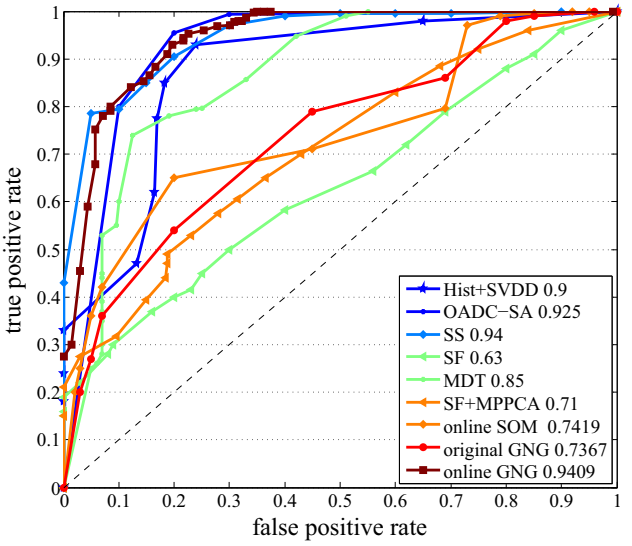
[2] We do not provide the comparison with a newly published article [62], since we cannot get the detailed results from its experiment section. Its ROC curves in Fig. 17 show that its results on UCSD Ped1 and Ped2 datasets are not very competitive to Fast SC 2013 [16] and MDT 2013 [45], respectively. These comparisons can be used as reference.



(a) Frame-level ROC curve for Ped1 scene



(b) Pixel-level ROC curve for Ped1 scene



(c) Frame-level ROC curve for Ped2 scene

Approach \ AUC	(a)	(b)	(c)
Hist+SVDD 2016	0.85	0.65	0.9
OADC-SA 2015	0.91	0.539	0.925
CCUKL 2014	0.9288	0.5035	-
SS 2013	0.87	**0.66**	0.94
MDT 2013	0.84	0.44	0.85
Fast SC 2013	0.9180	0.6380	-
SRC 2013	0.8600	0.4610	-
SF 2009	0.6750	0.2188	-
SF+MPPCA 2009	0.77	0.21	0.71
online SOM 2010	0.7822	0.4556	0.7419
original GNG	0.7267	0.5358	0.7367
online GNG (ours)	**0.9375**	0.6511	**0.9409**

(d) AUC statistics from (a), (b) and (c)

Fig. 12. The detection and localization results in UCSD Ped1 and Ped2 scenes. Involved approaches are with following abbreviations: Hist+SVDD 2016 [53], OADC-SA 2015 [52], CCUKL 2014 [51], SS 2013 [61], MDT 2013 [45], Fast SC 2013 [16], SRC 2013 [15], SF 2009 [44], SF+MPPCA 2009 [60], online SOM 2010 [24], original GNG, and our online GNG. Every approach has its AUC values marked in legends. The legend of (b) is the same to (a), and approach names are omitted for figure concision.

Fig. 12(b) shows that our approach has very comparable performance to the best result of SS model [61] in detecting anomaly locations by pixel-level measurement which signifies the precision for anomaly positioning. Intuitive comparisons are given in Fig. 12(d).

The disadvantage of our approach could be the lower computing speed because online parameter adjustment consumes computing resources. For example on UCSD Ped1 (360×240 resolution videos), the speed of our approach (0.073 second/frame) is about 10 times slower than that of Fast SC (0.00697 s/frame) [16]. However, this speed is fast enough for processing videos at about 15 fps, offering an efficiency guarantee for real environment applications.

5.4. Tests on avenue dataset

Avenue dataset [17] is the newest anomaly detection dataset proposed by Lu et al. [16]. There are 16 training videos and 21 testing videos. The average video length is about 2 minutes. In total, 38 discontinuous anomalies such as running, throwing objects, and loitering are observed. Following the settings in [16], we segment the image plane into 12×16 grids, then detect LAEs. On this dataset, model training starts using training videos, and it keeps on when detecting anomalies in testing videos.

Examples of anomaly detection are presented in Fig. 13. In the first row, a man is throwing papers and another man is running across the view at a high speed, which are distinctive events in front of a subway entrance. In the second row, the left snapshot shows an anomaly when the man picks up white papers from the ground. Meanwhile, an occasional false alarm is caused by a pedestrian nearby subway exit. Another false alarm occurs in the snapshot on the right side, due to the complex texture on a girl's handbag.

To analyze the contribution of our self-adaptive proposals, namely outer neuron insertion, neuron deletion and learning rate adaptation (l.r. adaptation), we test each of them in conjunction with other components from original GNG. Final detection results are given in

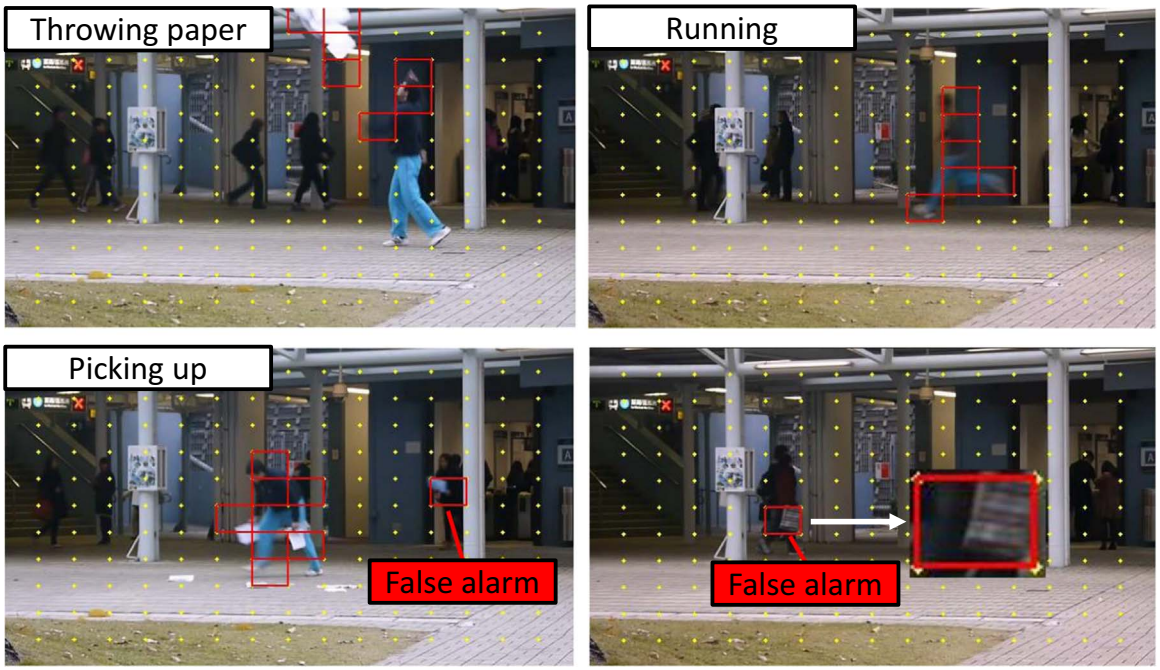


Fig. 13. Positive and negative detection results on Avenue dataset. Complete videos can be found in our Supplementary demo.

Table 4
Detection result comparisons on Avenue dataset.

	Run	Loiter	Throw	Dance	False alarm
Ground Truth	14	5	17	2	N/A
Fast SC [16]	14	4	15	2	10
Original GNG (λ=100)	12	4	14	2	8
With insertion	10	2	13	2	4
With insertion (λ=100)	10	2	13	2	4
With deletion	13	4	13	2	17
With deletion (λ=100)	13	4	14	2	19
With l.r. adaptation	14	4	16	2	8
Online GNG	14	4	16	2	6
Online GNG(λ=100)	14	4	16	2	6

Table 4. As introduced in Section 5.1, λ has possible impacts on neuron insertion and deletion. Table 4 shows that λ affects the final results only for neuron deletion. This effect vanishes when online GNG components work together, and corresponding results are given in the last two rows.

From the global perspective, it is evident that using deletion strategy increases false alarm but helps little to improve correct detection. The reason is when there is no outer neuron insertion, over deletion makes the model too compact to cover the data space. Furthermore, it is easy to understand why the opposite false alarm rate occurs when using neuron insertion only. Concerning learning rate adaptation, it improves the performance of original GNG in both decreasing false alarm and increasing detection rates because reasonable self-adaptation towards input data always tends to improve the model. Finally, there are some false alarms when using online GNG due to some salient events like someone passing by with a colorful handbag. In summary, online GNG achieves a satisfactory performance with only 2 missing detections and 6 false alarms.

6. Conclusions

This paper proposes an unsupervised model called online GNG to learn surveillance scenes with changing crowd density and behavior types. The Innovation lies in that dominant learning operations all employ online adaptive parameters. Moreover, the model itself can estimate the learning efficiency, and adjust the network size to fit changing data space automatically. In experiments, online GNG effectively reduces the false alarms and missing detections caused by model aging and threshold aging which frequently happen in unstable environments. Quantitative comparisons with 12 recent related works further confirm the superiority of our approach.

Our current neighbor-related strategies treat all neurons equally, which possibly results in some common parameters amplified by the sparsest or densest subregion of GNG network. The next step for optimization is to assign a weight to each neuron, and then update this weight according to proper density measurement. This scheme might improve the learning efficiency, thereby yielding a better representation for changing data space.

Acknowledgment

This work is supported by National Natural Science Foundation of China (NSFC, nos. 61673030, 61340046, 60875050, 60675025), National High Technology Research and Development Program of China (863 Program, no. 2006AA04Z247), Scientific Research Project of Guangdong Province (No. 2015B010919004), National high level talent special support program.

The authors wish to express special thank to Dr. Xiaofei Li (INRIA) and Andrew Shin (The University of Tokyo) for valuable discussions and suggestions, and to Dr. Shanshan Zhang, Wenbin Li, Siyu Tang, Yongqin Xian (all from Max-Planck Institute for Informatics) and Liqian Ma (from Peking University) for their help of proofreading.

Appendix A. Supplementary material

Supplementary data associated with this article can be found in the online version at http://dx.doi.org/10.1016/j.patcog.2016.09.016.

References

[1] K.C. Scott-Brown, P.D.J. Cronin, Detect the unexpected: a science for surveillance, Int. J. Police Strateg. Manag. 31 (3) (2008) 395–414.
[2] M. Valera, S.A. Velastin. A review of the state-of-the-art in distributed surveillance systems, in: Intelligent Distributed Video Surveillance Systems, IEE Professional Applications of Computing Series, vol. 5, 2006, pp. 1–30.
[3] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, S. Yan, Crowded scene analysis: a survey,

IEEE Trans. Circuits Syst. Video Technol. 25 (3) (2015) 367–386.
[4] A.A. Sodemann, M.P. Ross, B.J. Borghetti, A Review of Anomaly Detection in Automated Surveillance, IEEE Trans. Syst. Man Cybern. Part C: Appl. Rev. 42 (6) (2012) 1257–1272.
[5] P. Jodoin, J. Konrad, V. Saligrama, Modeling background activity for behavior subtraction, in: Second ACM/IEEE International Conference on Distributed Smart Cameras, ICDSC 2008, 2008, pp. 1–10.
[6] A. Zweng, M. Kampel, Unexpected human behavior recognition in image sequences using multiple features, in: The International Conference on Pattern Recognition, 2010, pp. 368–371.
[7] Q. Dong, Y. Wu, Z. Hu, Pointwise motion image (PMI): a novel motion representation and its applications to abnormality detection and behavior recognition, IEEE Trans. Circuits Syst. Video Technol. 19 (3) (2009) 407–416.
[8] Y. Benezeth, P.M. Jodoin, V. Saligrama, Abnormality detection using low-level co-occurring events, Pattern Recognit. Lett. 32 (3) (2011) 423–431.
[9] C.C. Loy, T. Xiang, S. Gong, Detecting and discriminating behavioural anomalies, Pattern Recognit. 44 (1) (2011) 117–132.
[10] B. Yao, L. Wang, and S. Zhu, Learning a scene contextual model for tracking and abnormality detection, in: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 2008, pp. 1–8.
[11] J. Yin, Y. Meng, Abnormal Behavior Recognition using Self-adaptive Hidden Markov Models, Image Analysis and Recognition, Springer, Berlin, Heidelberg, 2009, pp. 337–346.
[12] H.L. Chen, M.J. Tsai, C.C. Chan, A Hidden Markov Model-based approach for recognizing swimmers behaviors in swimming pool, in: IEEE International Conference on Machine Learning and Cybernetics, vol. 5, 2010, pp. 2459–2465.
[13] W. Lao, J. Han, P.H.N. de With, Automatic video-based human motion analyzer for consumer surveillance system, IEEE Trans. Consum. Electron. 55 (2) (2009) 591–598.
[14] T. Xiang, S. Gong, Video behavior profiling for anomaly detection, IEEE Trans. Pattern Anal. Mach. Intell. 30 (5) (2008) 893–908.
[15] Y. Cong, J. Yuan, J. Liu, Abnormal event detection in crowded scenes using sparse representation, Pattern Recognit. 46 (7) (2013) 1851–1864.
[16] C. Lu, J. Shi, J. Jia, Abnormal event detection at 150 fps in matlab, IEEE International Conference on Computer Vision, 2013, pp. 2720-2727.
[17] C. Lu, J. Shi, J. Jia, ⟨http://www.cse.cuhk.edu.hk/leojia/projects/detectabnormal/⟩.
[18] X. Zhu, J. Liu, J. Wang, C. Li, H. Lu, Sparse representation for robust abnormality detection in crowded scenes, Pattern Recognit. 47 (5) (2014) 1791–1799.
[19] E. Levina, P. Bickel, The earth mover's distance is the mallows distance: Some insights from statistics, in: IEEE International Conference on Computer Vision, 2001, pp. 251–256.
[20] P.L.M. Bouttefroy, A. Bouzerdoum, S.L. Phung, A. Beghdadi, Local estimation of displacement density for abnormal behavior detection, in: IEEE Workshop on Machine Learning for Signal Processing, 2008, pp. 386–391.
[21] I. Tziakos, A. Cavallaro, L. Xu, Local abnormality detection in video using subspace learning, in: IEEE International Conference on Advanced Video and Signal Based Surveillance, 2010, pp. 519–525.
[22] B. Zhao, L. Fei-Fei, E.P. Xing, Online detection of unusual events in videos via dynamic sparse coding, in: IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 3313–3320.
[23] G. Zhou, Y. Wu, Anomalous event detection based on Self-Organizing Map for supermarket monitoring, in: IEEE International Conference on Information Engineering and Computer Science, 2009, pp. 1–4.
[24] J. Feng, C. Zhang, P. Hao, Online learning with self-organizing maps for anomaly detection in crowd scenes, in: The International Conference on Pattern Recognition, 2010, pp. 3599–3602.
[25] S. Han, S. Cho, Evolutionary neural networks for anomaly detection based on the behavior of a program, IEEE Trans. Syst. Man Cybern. Part B: Cybern. 36 (3) (2006) 559–570.
[26] T. Kohonen, Self-Organizing Maps 30, Springer, Berlin Heidelberg, 2001.
[27] Q. Sun, H. Liu, Inferring ongoing human activities based on recurrent self-organizing map trajectory, in: The 24th British Machine Vision Conference, 2013, pp. 11.1–11.10.
[28] K. Haussermann, O. Zweigle, P. Levi, A novel framework for anomaly detection of robot behaviors, J. Intell. Robot. Syst. 77 (February (2)) (2015) 361–375.
[29] Q. Sun, H. Liu, Learning spatio-temporal co-occurrence correlograms for efficient human action classification, in: IEEE International Conference on Image Processing, 2013, pp. 3220–3224.
[30] Q. Sun, H. Liu, L. Ma, T. Zhang, A novel hierarchical bag-of-words model for compact action representation, Neurocomputing 174 (January (Part B)) (2016) 722–732.
[31] T. Martinetz, K. Schluten, A "neural-gas" network learns topologies, in: International Conference on Artificial Neural Networks, 1991, pp. 397–402.
[32] B. Fritzke, A Growing Neural Gas Network Learns Topologies, Advances in Neural Information Processing System 7, MIT Press, Cambridge, MA, 1995.
[33] B. Fritzke, A self-organizing network that can follow non-stationary distributions, in: International Conference on Artificial Neural Networks, 1997, pp. 613–618.
[34] T. Martinetz, Competitive Hebbian learning rule forms perfectly topology preserving maps, in: The International Conference on Artificial Neural Networks, Springer, London, 1993, pp. 427–434.
[35] J. Garcia-Rodriguez, J.M. Garcia-Chamizo, Surveillance and human-computer interaction applications of self-growing models, Appl. Soft Comput. 11 (7) (2011) 4413–4431.
[36] F. Shen, H. Osamu, An incremental network for on-line unsupervised classification and topology learning, Neural Netw. 19 (1) (2006) 90–106.
[37] B. Fritzke, Fast learning with incremental RBF networks, Neural Process. Lett. 1 (No.1) (1994) 2–5.
[38] J. Holmstrom, Growing neural gas—experiments with GNG, GNG with utility and supervised GNG (Master's thesis), Uppsala University, Department of Information Technology Computer Systems, 2002.
[39] F. Shen, O. Tomotaka, H. Osamu, An enhanced self-organizing incremental neural network for online unsupervised learning, Neural Netw. 22 (8) (2007) 893–903.
[40] I. Laptev, M. Marszalek, C. Schmid, Learning realistic human actions from movies, in: IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[41] I. Laptev, ⟨http://www.di.ens.fr/laptev/actions/⟩.
[42] P. Dollár, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, in: Joint IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005, pp. 65–72.
[44] R. Mehran, A. Oyama, and M. Shah, Abnormal crowd behavior detection using social force model, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 935–942.
[45] W. Li, V. Mahadevan, N. Vasconcelos, Anomaly detection and localization in crowded scenes, IEEE Trans. Pattern Anal. Mach. Intell. 36 (1) (2013) 18–32.
[46] L. Kratz, K. Nishino, Anomaly detection in extremely crowded scenes using spatio-temporal motion pattern models, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 1446-1453.
[47] Unusual Crowd Activity Dataset of University of Minnesota, ⟨http://mha.cs.umn.edu/Movies/Crowd-Activity-All.avi⟩.
[48] S. Wu, B. Moore, M. Shah, Chaotic invariants of Lagrangian particle trajectories for anomaly detection in crowded scenes, in: IEEE Conference on Computer Vision and Pattern Recognition, 2010, pp. 2054–2060.
[49] Y. Yang, I. Saleemi, M. Shah, Discovering motion primitives for unsupervised grouping and one-shot learning of human actions, gestures, and expressions, IEEE Trans. Pattern Anal. Mach. Intell. 35 (7) (2013) 1635–1648.
[50] H. Han, X. Wu, J. Qiao, Nonlinear systems modeling based on self-organizing fuzzy-neural-network with adaptive computation algorithm, IEEE Trans. Cybern. 44 (4) (2014) 554–564.
[51] W. Ren, G. Li, B. Sun, K. Huang, Unsupervised kernel learning for abnormal events detection, Vis. Comput. 31 (3) (2014) 245–255.
[52] Y. Yuan, J. Fang, Q. Wang, Online anomaly detection in crowd scenes via structure analysis, IEEE Trans. Cybernet. 45 (3) (2015) 562–575.
[53] Y. Zhang, H. Lu, L. Zhang, X. Ruan, Combining motion and appearance cues for anomaly detection, Pattern Recognit. 51 (2016) 443–452.
[54] D. Sugimura, K. M. Kitani, T. Okabe, Y. Sato, A. Sugimoto, Using individuality to track individuals: clustering individual trajectories in crowds using local appearance and frequency trait, in: IEEE International Conference on Computer Vision, 2009, pp. 1467–1474.
[55] P. Rousseeuw, Silhouettes: a graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.
[56] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, An extensive comparative study of cluster validity indices, Pattern Recognit. 46 (1) (2013) 243–256.
[57] O. Beyer and P. Cimiano, DYNG: dynamic online growing neural gas for stream data classification, in: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, 2013, pp. 24–26.
[58] J. Acevedo-Rodríguez, S. Maldonado-Bascón, R.López-Sastre, P. Gil-Jiménez, A. Fernández-Caballero, Clustering of trajectories in video surveillance using growing neural gas, in: The 4th International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC 2011), Lecture Notes in Computer Science, vol. 6686, 2011, pp. 461–470.
[59] N. Waniek, S. Bremer, J. Conradt, Real-time anomaly detection with a growing neural gas, in: The 24th International Conference on Artificial Neural Networks (ICANN 2014), Lecture Notes in Computer Science, vol. 8681, 2014, pp. 97–104.
[60] J. Kim, K. Grauman, Observe locally, infer globally: a space–time mrf for detecting abnormal activities with incremental updates, in: IEEE Conference on Computer Vision and Pattern Recognition, 2009, pp. 2921–2928.
[61] Y. Hu, Y. Zhang, L.S. Davis, Unsupervised abnormal crowd activity detection using semiparametric scan statistic, in: IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2013, pp. 767–774.
[62] A.C.B. Abdallah, M. Gouiffès, L. Lacassagne, A modular system for global and local abnormal event detection and categorization in videos, Mach. Vis. Appl. 27.4 (2016), pp.463-481

Qianru Sun received the Ph.D. degree from Peking University, in January 2016. She is currently a post-doctor in the Department of Computer Vision and Multimodal Computing, Max-Planck-Institute for Informatics, Germany. Her research interests include human action recognition, anomaly detection, and social information analysis.

Hong Liu received the Ph.D. degree in Mechanical Electronics and Automation in 1996, and serves as a Full Professor in the School of EE & CS, Peking University (PKU), China. Prof. Liu has been selected as Chinese Innovation Leading Talent supported by "National High-level Talents Special Support Plan" since 2013. He is also the Director of Open Lab on Human–Robot Interaction, PKU, his research interests include computer vision, pattern recognition and robot motion planning.

Tatsuya Harada received the M.S. degree in Mechanical Engineering from the University of Tokyo, Japan, in 1998, and the Ph.D. degree in Mechanical Engineering from the University of Tokyo, Japan, in 2001. Now he is a professor in the Department of Mechano-Informatics, Graduate School of Information Science and Technology, the University of Tokyo. He is the director of Machine Intelligence Lab. His research interests include real-world intelligent systems, object and scene recognition, and data-mining methods in time-series data.