Anomaly Detection System for Video Data Using Machine Learning

Tadashi Ogino

***Abstract*—We are developing an anomaly detection system for video data that uses machine learning. The proposed system has two subsystems: feature extraction and anomaly detection. We developed two feature extraction systems. One uses traditional manual steps and the other uses machine learning, i.e., a neural network. For the anomaly detection system, we employ machine learning technology that we have developed for a cyber-attack detection system. Results confirm that both prototypes can detect anomalous events in experimental video data.**

***Index Terms*—Anomaly detection, machine learning, video, Jubatus.**

I. Introduction

There are many surveillance cameras installed in many locations such as stores, banks, roads, and airports. Most video data from such cameras are only recorded for emergencies and are watched only after such an event occurs. In some cases, security guards watch the videos, but this is not effective from cost and performance perspectives. We are developing a video system that can automatically detect irregular events from video images in real time. In addition, we have been developing a cyber-attack detection system that uses machine learning technology [1]. We have applied this technology in our cyber-attack detection system to video images.

Our system is composed of two subsystems: a feature extraction subsystem and an anomaly detection subsystem. We developed two extraction subsystems. One prototype uses a traditional approach. We developed this system to be suitable for the specialized application and manually tuned the system. The other prototype uses machine learning technology, i.e., a neural network. The system parameters are automatically tuned. For the anomaly detection subsystem, we developed the system using Jubatus and used the local outlier factor (LOF) algorithm. This approach also uses machine learning technology.

II. Previous Work

We have been developing a cyber-attack detection system that uses machine learning technology [1]. An overview of this system is shown in Fig. 1. The system collects system logs, i.e., traffic and operation logs. All collected logs are statistically analyzed and learned as normal data. In this period, we estimated that the system is not under attack. The system can detect new attacks after the learning period.

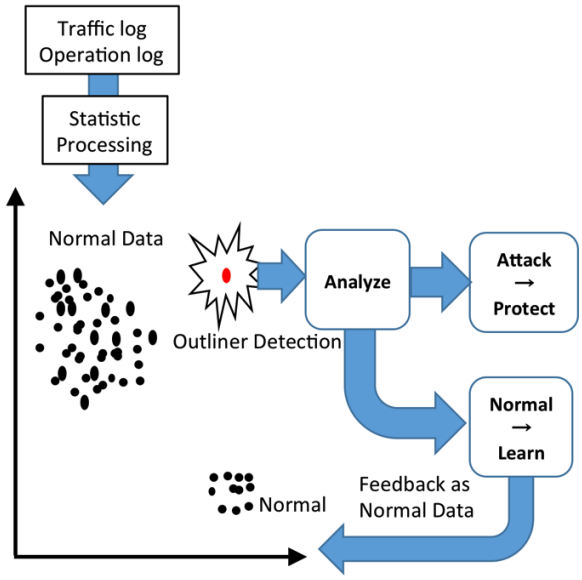

Fig. 1. Overview of system processing flow of cyber attack detection system.

In the cyber-attack detection system, the traffic or operation log data are well formatted, and it is easy to extract feature vectors using existing tools (e.g., tcpdump). However, this is not the case with video data. We require a special tool that is suitable for this purpose. Here, we describe two prototype feature extraction tools.

III. System Overview

Our anomaly detection system for video data is illustrated in Fig. 2.

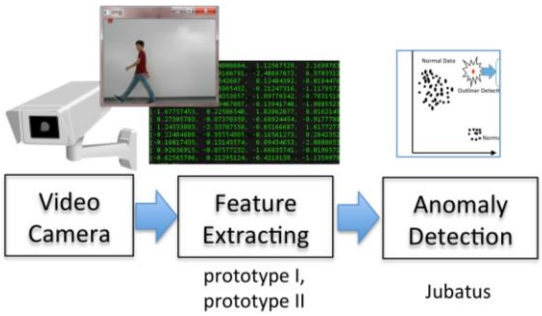

Fig. 2. Anomaly detection system for video.

Our system comprises two subsystems. The video image is sent to the first subsystem, i.e., feature extraction. We developed two feature extraction subsystems. One prototype is a hand-made system. All the inner steps are manually designed. The other prototype is a feature extraction system that uses machine learning technology. All features are determined in the machine training phase. Both cases are

Manuscript received July 29, 2015; revised December 17, 2015.

Tadashi Ogino is with School of Information Science, Meisei University, Hino, Japan (e-mail: tadashi.ogino@nifty.com).

doi: 10.18178/ijke.2016.2.2.062

detailed in the following sections.

After some features are extracted from the video, the anomaly detection subsystem determines if the features are normal or abnormal. This part runs on Jubatus, a machine learning framework. This part will be explained later.

IV. System Details

A. Feature Extracting Subsystem

We developed two feature extraction subsystems. One is a hand-made feature extraction subsystem. The other is an automatic feature extraction subsystem that uses machine learning technology.

1) Hand-made feature extraction: Prototype I

First, we manually developed a feature extraction system [2] using a traditional method that combines several image analyzing steps. The prototype is written in C++ using the opencv2 library.

This prototype has the following steps.

a) Background removal

The background is removed from each video frame in order to detect moving objects. In our prototype, the background is recorded separately (Fig. 3).

Fig. 3. Background removal.

Fig. 4. Key point detection.

b) Feature point extraction and matching

First, key points are found in the remaining moving objects (Fig. 4). Then, similar key points from two consecutive frames are matched. These two steps are executed using Oriented FAST and Rotated BRIEF [3].

From the matched similar points ($x0$, $y0$) and ($x1$, $y1$), we create a flow vector ($x0$, $y0$, $vx = x1 - x0$, $vy = y1 - y0$). This flow vector shows the speed of each key point (Fig. 5).

c) Labeling

At the same time as step b), we set the label for each object.

d) Create one flow vector for one label

For every flow vector with the same label, we calculate the average speed (vx_{ave}, vy_{ave}) and the center of the object (x_{center}, y_{center}). A 4D vector (vx_{ave}, vy_{ave}, x_{center}, y_{center}) is generated to represent every vector in a single object (Fig. 6).

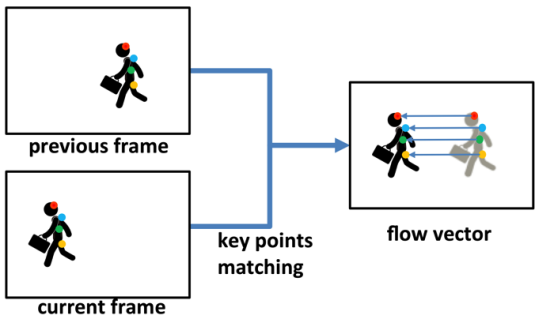

Fig. 5. Key point matching.

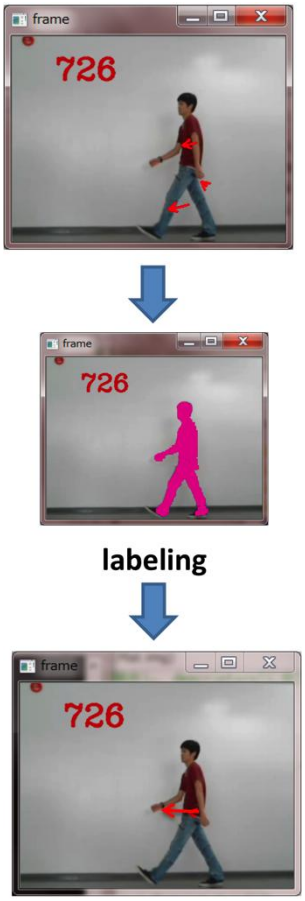

Fig. 6. Labeling and one flow vector.

e) Regularization

Each flow vector (vx_{ave}, vy_{ave}, x_{center}, y_{center}) is normalized such that each value is between 0 and 1.

At this point, each frame has several representative vectors for each object in the frame. We use these vectors as feature vectors for the frame and send them to the next step (i.e., the

anomaly detection subsystem).

2) Automatic feature extracting: Prototype II

We use neural networks to extract features in the second feature extraction prototype.

Neural networks have been extensively studied [4]. Recent prominent research in the area of speech recognition and image recognition [5], [6] has received significant attention from researchers and companies. Here, we provide a brief overview of neural networks [7].

A simple neural net is shown in Fig. 7. There is a single neuron and some input x. The output is as follows:

$$h_{w,b}(x) = f(W^T x) = f(\sum_{i=1}^{3} W_i x_i + b). \quad (1)$$

Here, $f(x)$ is an activation function. This is usually a nonlinear function, which can be expressed as follows:

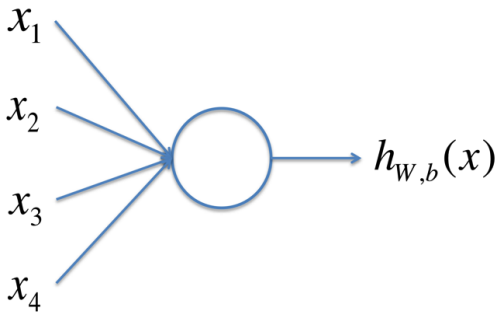

Fig. 7. Simplest neural net.

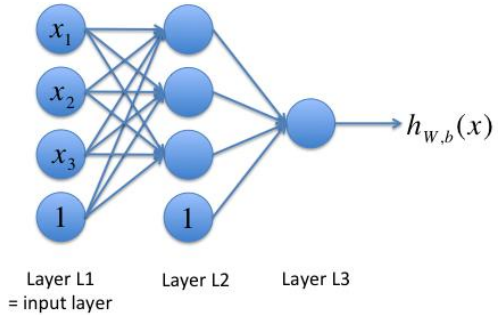

Fig. 8. Neural net.

$$f(z) = \frac{1}{1+\exp(-z)}. \quad (2)$$

A neural network is composed of many neurons (Fig. 8). The network structure is designed according to the target problem. Many architectures have been proposed by researchers.

After the network structure is determined, a good parameter set, i.e., (W, b) should be found to best describe the target problem. For a dataset (x_i, y_i), x_i is input and y_i is output. Then, the error J of this neural network can be calculated as follows:

$$J(W, b; x, y) = \frac{1}{2}\left\|h_{W,b}(x) - y\right\|^2. \quad (3)$$

Typically, J is referred to as a *cost* function. *A* back propagation algorithm is used to find a good set of (W, b) to minimize the *cost* function. However, in our case, we only have input dataset (x_i) and cannot use back propagation. Therefore, we use the auto encoder technique to train the network.

In the auto encoder, the neural network is composed of two layers: encoder and decoder. The input and output data are the same. Training is performed as in a normal neural network, such as with back propagation. After training, the encoder layer is used to represent the input data. Typically, the auto encoder is used to pre-train multilayer neural networks to find a better starting point to fine tune a huge neural network (Fig. 9).

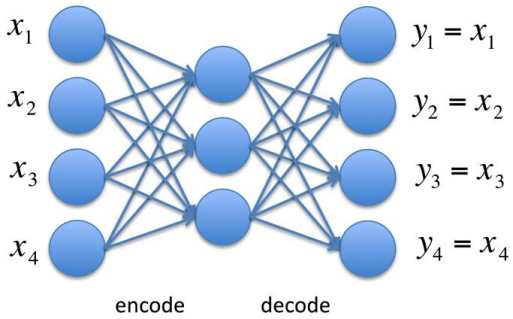

Fig. 9. Auto encoder.

As this is unsupervised learning, we use a denoising auto encoder to train the system. The network configuration is shown in Fig. 10.

Our prototype is written in Chainer [8]. In the prototype, the input image is converted to 8-bit grayscale and resized to 64 × 64 pixels. The neural network has one layer that consists of 64 × 64 nodes, 32 × 32 outputs, and is fully connected. We use the sigmoid function for the activation function.

B. Anomaly Detection Subsystem

After feature vectors are sent to the anomaly detection subsystem, all vectors are analyzed using Jubatus [9].

1) Jubatus

The amount of video data is increasing dramatically. To analyze such data in real time, it would be beneficial to apply "big data" analysis technology.

A well-known big data analysis tools is Hadoop. Hadoop is an open source software tool. It is used in many fields such as recommendation, web search, and text mining. Hadoop is typically used in batch processing. However, it is somewhat difficult to use Hadoop for real-time analysis.

We use Jubatus as our real-time analysis platform. Jubatus is a distributed machine learning platform developed by NTT and PFI. Jubatus was developed for real-time deep analysis in a distributed environment.

Jubatus includes many major machine learning algorithms. We use the LOF algorithm for anomaly detection.

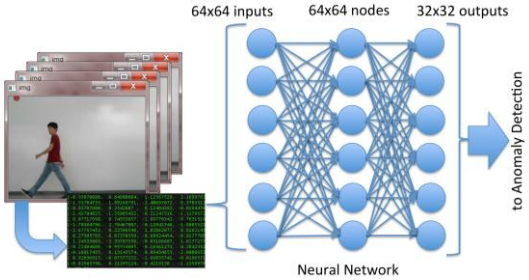

Fig. 10. Feature extracting subsystem.

2) LOF

There are several definitions of outliers. Hawkins

described outliers as "an observation that derives so much from other observations as to arouse suspicion that it was generated by a different mechanism" [10].

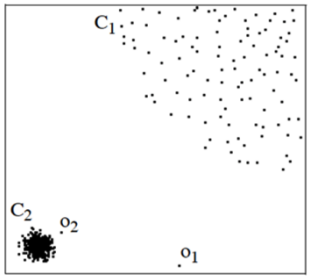

Fig. 11. 2-d dataset.

In most cases, this definition is sufficient to detect outliers. However, as seen in Fig. 11 [11], although o_1 can be detected as an outliner, o_2 cannot be detected as outlier. The LOF algorithm is designed to find such cases. The details of the LOF algorithm are explained in the literature [11].

3) Anomaly detection

In this subsystem, we must first train the Jubatus server with normal data (training phase). Then, we can determine if the data are normal or an anomaly (Fig. 12). With the LOF algorithm, normal data return an LOF value that is close to 1. Abnormal data return an LOF value that is much greater than 1.

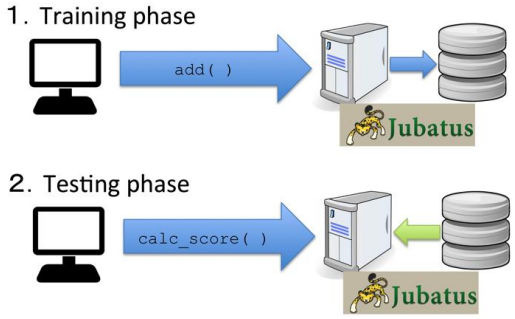

Fig. 12. Anomaly detection using Jubatus.

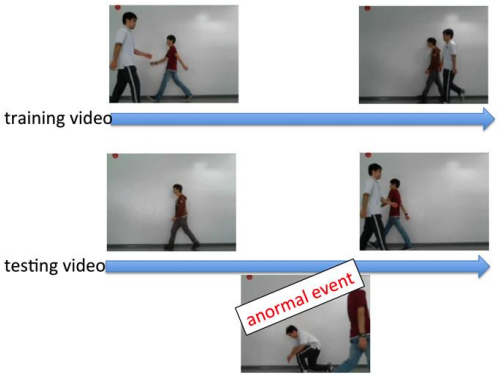

Fig. 13. Experimental video data.

V. Experiment

We evaluated our system using a simple video. The training (normal) video is a scene in which several humans are walking. The test (abnormal) video is a similar video; however, it contains a scene in which one person is falling down (Fig. 13). Both videos are one minute long.

The LOF scores obtained in the experiment are shown in Figs. 14 and 15.

In both prototypes, the falling scene (approximately frame 600) was detected with a high LOF value, i.e., abnormal.

In the hand made prototype I, frames with no objects had no flow vectors. Hence, there was no LOF score.

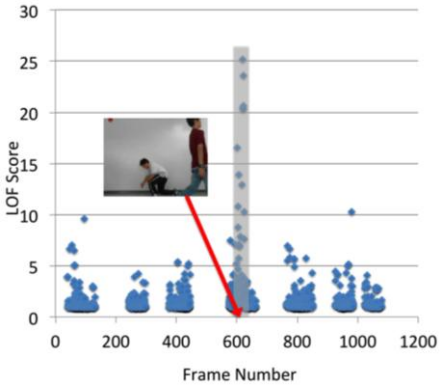

Fig. 14. Result of proto type I (hand-made feature extractor).

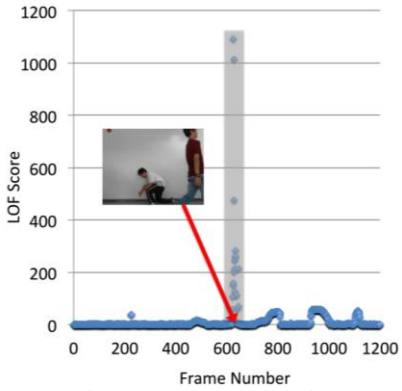

Fig. 15. Result of proto type II (automatic feature extractor).

Although both prototypes could detect anomalous events, we believe that prototype II is suitable for a wider application because it tunes the feature extraction process using neural network technology. Neural network technology is advancing rapidly, and such technology can be applied to our anomaly detection system.

VI. Future Work

The following issues will be the focus of future work.

A. Evaluation Method

In this study, we evaluated whether our prototypes could detect irregular events using the result graphs (Fig. 14 and Fig. 15). In the next step, we require a numerical evaluation index, such as error or detection rates. To make the evaluation more feasible, we also required standard video data.

B. Time Series Information

We used only frame data in the second prototype. As our target is video data, we must include time series information for feature vectors to detect irregular objects, e.g., objects that are moving too quickly, too slowly, or in irregular directions.

C. Extended Evaluation

In this study, our evaluation was performed using one minute of video data. Although we believe our approach can be applied to a consecutive real-time video sequence, some implementation problems may exist. Hence, we must develop an actual real-time system to confirm our idea.

VII. Conclusion

We presented two prototype anomaly detection systems for video data using machine learning. Both prototypes could detect anomalous events in the experimental video data. In future, we plan to apply more recent neural network technology such as deep learning to our system.

References

[1] T. Ogino, "Evaluation of machine learning method for intrusion detection system on Jubatus," *International Journal of Machine Learning and Computing*, vol. 5, no. 2, pp. 137-141, April 2015.

[2] R. Kuroki, K. Omine, and T. Ogino, "Realtime anomaly detection using machine learning," in *Proc. 2015 77th National Convention of IPSJ*, 2015.

[3] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. IEEE International Conference on Computer Vision 2011*, pp. 2564-2571, Nov. 2011.

[4] W. S. McCulloch and W. H. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, no. 4, pp. 115-133, 1943.

[5] D. Yu and F. Seide, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. 2011 INTERSPEEC*, 2011, pp. 437-440.

[6] J. Deng, A. Berg, S. Satheesh, H. Su, A. Khosla, and F.-F. Li, "Large scale visual recognition challenge 2012," *ILSVRC 2012 Workshop*, 2012.

[7] UFLDF Tutorial. [Online]. Available: http://deeplearning.stanford.edu/wiki/index.php/UFLDL_Tutorial

[8] Chainer. [Online]. Available: http://chainer.org/

[9] Jubatus. [Online]. Available: http://jubat.us/

[10] D. M. Hawkins, *Identification of Outliers*, Netherlands: Springer, 1980.

[11] M. M. Breunig, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," in *Proc.2000 ACM SIGMOD International Conference on Management of Data*, June 2000, pp. 93-104.

Tadashi Ogino received his BS, MS, and PhD. in electrical engineering from the University of Tokyo in 1983, 1985, and 1988, respectively. His research interests include distributed systems, cloud computing, M2M systems, and big data analysis.

He joined the Mitsubishi Electric Corp. in 1988 and developed mid-range business servers. After working at the Okinawa National College of Technology, he joined Meisei University as a professor. Dr. Ogino is a member of ACM, IEEE, IPSJ, and IEICE.