Key Clips and Key Frames Extraction of Videos Based on Deep Learning

Key Clips and Key Frames Extraction of Videos Based on Deep Learning

Junyu Chen[1], Ganlan Peng[2], Yuanfang Peng[3], Mu Fang[1], Zhibin Chen[1], Jianqing Li[2] and Liang Lan[4*]

[1] Sichuan Communication Scientific Research Planning and Design Co., Ltd, Chengdu, China
[2] School of Electronic Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China
[3] Southwest Petroleum University, Chengdu, China
[4] China Telecom Sichuan Branch, Chengdu, China
Email: 18908002518@189.cn; working_gl@163.com; 3614834@qq.com; 531778771@qq.com;18980090869@189.cn; lijq@uestc.edu.cn; *lanliang@189.cn

Abstract. The surveillance camera network covering the city, while protecting the safety of people's lives and property, generate a large amount of surveillance video data every day, but few videos contain useful information. Surveillance cameras in sparsely crowded areas may capture most of the video in the background. A large number of surveillance videos bring greater storage pressure, and also increase the difficulty of the staff's work. In this paper, we use the auto-encoder network to extract features of video frames. By comparing the feature differences between video frames, we can automatically select key video clips and key frame images that contain useful information to slim down the surveillance video. Through experiments on actual cameras, we found that this method can achieve the purpose of reducing the pressure of storage and traversal.
Keywords. Auto-encoder; key frames; deep learning; image restoration.

1. Introduction

In recent years, the increasingly popular video surveillance network has covered most public places in cities. Because of the popular video surveillance network, people's lives and property safety have been more fully guaranteed. But countless video data is generated every day. A large number of surveillance videos have brought huge storage pressure. At the same time, when staff traverse a large amount of surveillance video content, a lot of manpower will be wasted and the efficiency will be low.

In the massive surveillance video, not all scenes contain valid information at all times. In fact, areas with low personnel mobility, such as server rooms, corridors in buildings with small traffic, etc., may remain unchanged for most of the time under monitoring. These videos with no change time do not contain useful information, but take up a lot of storage space and consume a lot of manpower when traversing. If we can automatically remove these useless video clips and only keep the potentially useful video clips and the key frame images in the clips, the pressure of storage and traversal can be greatly reduced.

In this work, we use a deep learning method to extract video frame features using an auto-encoder network. By calculating the difference between the features, we can automatically remove the useless parts of the video, and only keep the video clips and key frames that contain useful information. In this

way, we can not only greatly reduce the storage pressure of surveillance video, but also quickly know the events that occur during this time period by key frames.

2. Proposed Approach

There are many ways to extract key video clips and key frames from a video. In the field of deep learning, we can use the method of object detection to detect the targets in the video, save the video segment when the target appears, or extract the video feature through the auto-encoders, and retain the video segment when the feature changes significantly. In the actual application scenario considered, when an event occurs, there may not necessarily be a target in the frames, but it may also be other emergencies such as facility damage, environmental abnormality, etc. Therefore, this paper uses the auto-encoders to extract the features, and retains the video key segments and key frame images by comparing the changes of the features. In this chapter, we will introduce the methods that will be used in this article.

2.1. Loss Function for the Network

To train an auto-encoder network, we need a loss function to measure the results. In order to make the features extracted by the auto-encoder network contain all the features of the input image as much as possible, we hope that the output of the auto-encoder can be as similar as possible to the original input image. In other words, we hope that the features extracted from the auto-encoder can be restored to the original image. In evaluating the similarity of image restoration, commonly used loss functions include L2 loss function, SSIM loss function, MS-SSIM loss function, and loss function combined with MS-SSIM and L1.

L2 loss function $loss_{L2}$ is a common loss function used to compare difference between two images, and is often used to measure the error of image restoration [1-3], but its performance is poorly correlated with the perception of human observers [4]. The sensitivity of the human visual system to noise depends more on local luminance, contrast, and structure [5].

The SSIM loss function $loss_{ssim}$ is derived from the comparison standard SSIM[5] of image similarity, and is often used for image compression [6], image reconstruction [7], image denoising and super-resolution [8] tasks. The SSIM method compares the similarity of two images from three aspects, namely the luminance, contrast and structure of the image.

The MS-SSIM standard [9] is a multi-scale version of the SSIM standard. MS-SSIM loss function $loss_{ms-ssim}$ has a good performance in the restoration ability of the edge and local area of the image.

Although the loss functions of MS-SSIM and SSIM restore clearer images, the restoration of colors is relatively poor. So we can combine MS-SSIM loss function with the L1 loss function as $loss_{Mix}$ to get more accurate color reproduction [10].

2.2. Structure of the Network

After selecting the loss function of the auto-encoder network, we need to confirm the basic structure of the auto-encoder network. In order to ensure the speed of extracting video features, we hope to reduce the amount of network calculation and the size of features as much as possible. Therefore, we compared the effects of the network in terms of the number of channels, input size, and feature size of the network. First, we compare the effect of reducing the number of channels on the network. Finally, we fixed the input size and reduced the feature size to compare the effects of different feature sizes on the network effect.

2.3. Training Method for the Network

After confirming the basic structure of the self-encoding network, we compared the training methods of the network in order to achieve better results on the current network structure. First, we compared the effects of training for each camera individually and training a generalized network directly on a large data set. Then, we confirmed whether the network should be pre-trained with a large data set.

3. Experiments

In order to determine the loss function, structure and training method, we have verified the influence of different conditions on the effect of the network through a series of experiments. The following content is a brief introduction of our experimental data set and the display of experimental results.

3.1. Data Sets

Our data sets are divided into two groups. One is the Caltech 256 data set with richer image categories, and the other is the images captured by the single actual cameras.

3.1.1. Caltech 256. Caltech 256 is a data set collected and compiled by the California Institute of Technology, contains 256 classes of images and a total of about 30,000 images. We use this data set with rich image types to train the generalized auto-encoder networks.

3.1.2. Camera Data. We use images taken by three different surveillance cameras as three data sets, called Cam A, Cam B, and Cam C. The three data sets contain approximately 3800, 9700 and 7800 images respectively. These data sets are used to train the network separately for each camera.

3.2. The Impact of Different Loss Functions on the Network

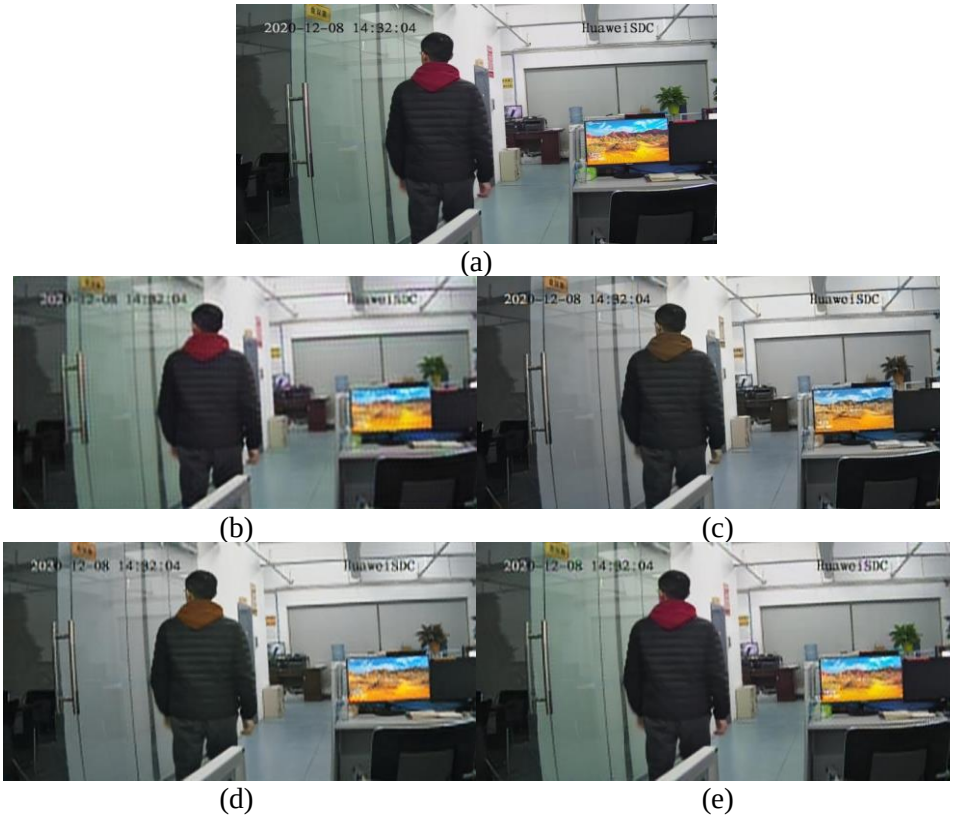
We use a simple convolutional auto-encoder network to test the effects of the four loss functions, and call this network AE_test-01. The network uses a full convolution connection, and each layer of convolution uses a batch normalization (BN) layer. The auto-encoder network is divided into an encoding layer (encoder) and a decoding layer (decoder). The encoder structure of the network is shown in table 1:

Table 1. Network AE_test-01 structure.

Network name	AE_test-01
Image size(H*W)	192*384
Convolutional layer 1	3*3 conv-8, 192*384
Convolutional layer 2	4*4 conv-16, 96*192
Convolutional layer 3	4*4 conv-32, 48*96
Convolutional layer 4	4*4 conv-64, 24*48
Convolutional layer 5	3*3 conv-8, 24*48

We use the Caltech 256 data set and four loss functions $loss_{L2}$, $loss_{ssim}$, $loss_{ms-ssim}$, and $loss_{Mix}$ to train the auto-encoder network AE_test-01. After the training is completed, save the model with the lowest loss value, and compare the image restoration effects of the four loss functions. Figure 1a is the original input image of the network. Figures 1b and 1c are the result of $loss_{L2}$ and the result of $loss_{ssim}$, respectively. Figures 1d and 1e are the result of $loss_{ms-ssim}$ and the result of $loss_{Mix}$, respectively.

From the restoration effect in figure 1, it can be seen that the restored image of $loss_{L2}$ is the blurriest, but the color is basically not distorted. The restored image of $loss_{ssim}$ is the sharpest, but the color is seriously distorted, and the color patches in some positions disappear directly. The color distortion of the restored image of $loss_{ms-ssim}$ is slightly better than that of $loss_{ssim}$, but it is more blurred. The color distortion of $loss_{Mix}$ is very small, and the sharpness is higher than that of $loss_{ms-ssim}$. From the comparison of several results, the comprehensive effect of $loss_{Mix}$ is the best, so we all use $loss_{Mix}$ as the loss function in the training of the following auto-encoder networks.



(a) (b) (c) (d) (e)

Figure 1. Comparison of restoration effects of different loss functions.

3.3. Confirmation of Network Structure

According to the needs of the project and application scenarios, the network needs to ensure the restoration effect as much as possible while compressing the resource occupation and calculation amount. And in order to reduce the resource consumption for calculating feature differences, the feature size extracted from the encoder should be as small as possible. This section confirms the network structure from three aspects: the number of channels, input size, and feature size.

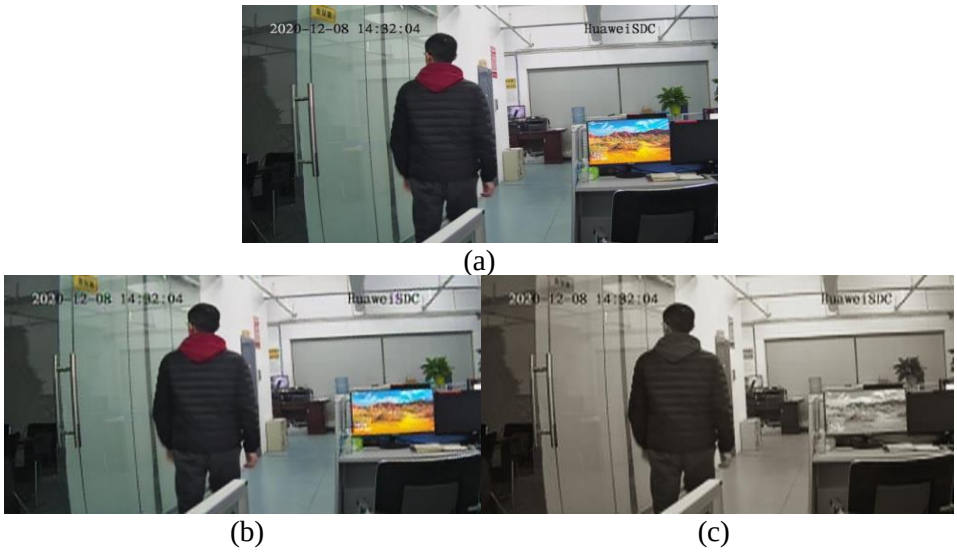
3.3.1. Number of Channels in the Network. We construct two simple convolutional self-encoding networks, called AE_test-01 and AE_test-02, which differ only in the number of channels. Then we use the Caltech 256 data set to train two networks separately and compare their restoration effects on the test set. The encoder structure of the two networks is shown in table 2:

Table 2. Comparison of the structure of the two networks.

Network name	AE_test-01	AE_test-02
Image size(H*W)	192*384	
Convolutional layer 1	3*3 conv-8, 192*384	3*3 conv-8, 192*384
Convolutional layer 2	4*4 conv-16, 96*192	4*4 conv-8, 96*192
Convolutional layer 3	4*4 conv-32, 48*96	4*4 conv-8, 48*96
Convolutional layer 4	4*4 conv-64, 24*48	4*4 conv-8, 24*48
Convolutional layer 5	3*3 conv-8, 24*48	

As shown in table 2, the network AE_test-01 has a larger number of channels for feature extraction, and the network AE_test-01 has a smaller number of channels. The actual test results of the two

networks after training are shown in figure 2. Figure 2a is the original input image of the network. Figures 2b and 2c are the result of AE_test-01 and the result of AE_test-02, respectively:



(a)

(b) (c)

Figure 2. Comparison of network restoration effects of different channel numbers.

As can be seen from figure 2, after reducing the number of network channels, the restored image color completely becomes black and white, and the reduction of channels affects the extraction of color's features. This shows that the method of reducing the amount of network calculation and resource occupation by reducing the number of channels is not feasible.

3.3.2. The Input Size of the Network. The output of the encoder is the feature extracted by the auto-encoder. We use four convolutional auto-encoder networks with different input sizes, and the features they extract are all the same size. Their loss values and encoder structure is shown in table 3:

Table 3. Comparison of network effects of different input sizes.

Network name	P1_1	P1_2	P1_3	P1_4
Image size (H*W)	384*786	192*384	96*192	48* 96
Convolutional layer 1	4*4 conv-32, 192*384	3*3 conv-32, 192*384	3*3 conv-32, 96*192	3*3 conv-32, 48*96
Convolutional layer 2	4*4 conv-64, 96*192	4*4 conv-64, 96*192	3*3 conv-64, 96*192	3*3 conv-32, 48*96
Convolutional layer 3	4*4 conv-128, 48*96	4*4 conv-128, 48*96	4*4 conv-128, 48*96	3*3 conv-128, 48*96
Loss value	0.0024	0.0024	0.0015	0.0030

It can be seen from table 3 that when the size of the input image is 96*192, the loss value is the lowest. The reason for this phenomenon may be that, for the same feature size, the input image of a larger size loses relatively more information in the process of extracting features, which leads to a poor restoration effect. The smaller size of the input image may extract redundant information in the process of extracting features, which also leads to the deterioration of the restoration effect. Therefore, from the experimental results, the size of 96*192 is a more appropriate input size. This paper will use this size as the input size of the subsequent network.

3.3.3. Feature Size Extracted by the Network. We add several additional convolutional layers after the third layer of the network P1_3 in the above, so that the auto-encoder has smaller features, and compare the effects of auto-encoders with different feature sizes. Training on the data set captured by the actual camera, the effect comparison is shown in table 4:

Table 4. Comparison of network effects of different feature sizes.

Network name	P2_1	P2_2	P2_3
Convolutional layer 1-3	Same as P1_3		
Convolutional layer 4	4*4 conv-64, 24*48	4*4 conv-64, 24*48	4*4 conv-64, 24*48
Convolutional layer 5	4*4 conv-8, 12*24	4*4 conv-32, 12*24	4*4 conv-32, 12*24
Convolutional layer 6	N/A	4*4 conv-8, 6*12	4*4 conv-32, 6*12
Convolutional layer 7	N/A	N/A	4*4 conv-8, 3*6
Loss value	0.0088	0.0136	0.0145
Validation loss value	0.0106	0.0338	0.0372

It can be seen that when the feature size is 12*24, the loss between the training set and the validation set is relatively close. When the feature size is smaller, the over-fitting situation becomes more obvious. Therefore, we believe that for the actual camera shooting scene, the feature size of 12*24 is more appropriate.

3.4. Confirmation of the Training Method of the Network

According to the basic network structure determined in the above, our auto-encoder network has 10 layers, of which the encoder and decoder have 5 layers each. The input size of the network is 96*192, and the number of channels is 3; the feature size is 12*24, and the channels of the feature is 8. We call the final structure of the network AE_final, and its structure diagram is shown in figure 3. AE_final will be the structure of the network used in the experiment later.

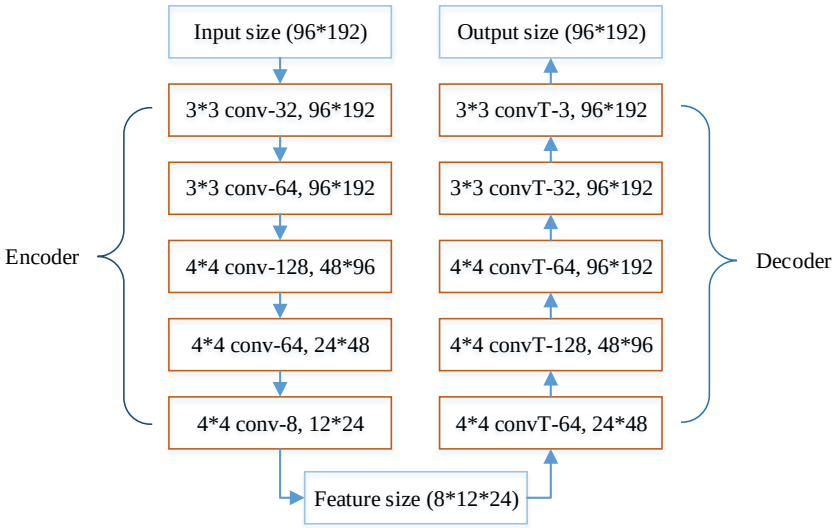


Figure 3. The structure of AE_final.

3.4.1. Training Each Camera Individually. From experience, the effect of training a network for a single camera scene is better than training a generalized network. In order to verify this conclusion, we use the Caltech 256 data set, which contains more image types, to train a generalized network, and use

the validation sets of Cam A, Cam B, and Cam C to calculate the validation set loss. Then we use the network weights trained on the Caltech 256 data set as the pre-training weights, and continue to train separately in Cam A, Cam B, and Cam C. Finally, we compare the effects of the network trained on Caltech 256 with the network trained on Cam A, Cam B, and Cam C. The effect is judged by their loss value on the validation set of Cam A, Cam B and Cam C. The comparison results are shown in table 5:

Table 5. Comparison of whether to train each camera individually.

Training set	Caltech 256			Cam A	Cam B	Cam C
Validation set	Cam A	Cam B	Cam C	Cam A	Cam B	Cam C
Validation loss value	0.0103	0.0122	0.0151	0.0016	0.0027	0.0071

It can be seen from table 5 that the loss value of the validation set that is trained separately for each camera is significantly lower. This shows that training the camera alone can achieve better results. Therefore, in practical applications, it is a better choice to train the data of each surveillance camera separately.

3.4.2. Using Pre-trained Weights. In the above, we used the weights trained on the Caltech 256 data set as the pre-training weights when training each camera individually. In order to confirm whether pre-training is necessary, in this section, we compare the effect of pre-training with Caltech 256 data set and training only on Cam A, Cam B, and Cam C. The effect comparison is shown in table 6:

Table 6. Comparison of whether to use pre-trained weights.

Data set	Cam A		Cam B		Cam C	
pre-training or not	No	Yes	No	Yes	No	Yes
Validation loss value	0.0023	0.0016	0.0028	0.0027	0.0078	0.0071

It can be seen from table 6 that the network using pre-training weights can usually obtain a lower loss value when training the camera images. taking the training on Cam C as an example, figure 4 is a comparison of the effect of using pre-training weights and not using pre-training weights. It can be seen that the initial effect and convergence speed of the network using pre-training weights are faster than the network without pre-training weights. In practical applications, this means that a trained network can be obtained faster. In practical applications, this means that we can get a trained network faster by using a network with pre-trained weights. Therefore, it is feasible to initialize the network with pre-training weights.

3.5. Display of Key Frame Extraction

After confirming the network structure and training method, we trained and obtained the required auto-encoder network model. Using the video taken by the camera of Cam A to test the effect of the model. We input video frames into the network in chronological order, and calculate the value of $loss_{Mix}$ between the current frame and the latest key frame. If the value of $loss_{Mix}$ is greater than the threshold (set to 0.03 in the experiment), we save the current image as the new key frame image. Some of the key frames saved are shown in figure 4.

In an event, when the first key frame is generated, we start to save the key video clip. When no new key frames are generated for a period of time, we conclude that the event is over and end the saving of the key video clips. In this way, when no event occurs in the surveillance camera screen, the captured video will not be saved. This can reduce the storage space occupied by a large number of useless video clips, and can also help people quickly browse the captured video content.

Figure 4. Display of some saved key frames.

4. Conclusion

In this paper, we have completed the work of extracting key video clips and key frames from massive videos. In order to allow the network to have a lower amount of calculation without significant interference, we experimented to determine the number of channels, input size, and feature size of the network. After determining the network structure, we found the most suitable training method through experiments. Finally, we got the required network and completed the goal of extracting key video clips and key frames from the complete surveillance video.

Acknowledgments

This work was supported by Sichuan Science and Technology Program (Grand No. 2020YFG0388).

References

[1] Burger H C, Schuler C J and Harmeling S 2012 Image denoising: Can plain neural networks compete with BM3D? *IEEE Conference on Computer Vision and Pattern Recognition* (Providence) pp 2392-99.
[2] Dong C, Loy C C, He K and Tang X 2014 Learning a deep convolutional network for image super-resolution *European Conference on Computer Vision* (Zurich) pp 184-99.
[3] Wang Y 2014 A multilayer neural network for image demosaicking *IEEE International Conference on Image Processing* (Paris) pp 1852-56.
[4] Zhang L, Zhang L, Mou X and Zhang D 2012 A comprehensive evaluation of full reference image quality assessment algorithms *IEEE International Conference on Image Processing* (Lake Buena Vista) pp 1477-80.
[5] Wang Z, Bovik A C, Sheikh H R and Simoncelli E P 2004 Image quality assessment: From error visibility to structural similarity *IEEE Transactions on Image Processing* vol 13 pp 600-12.
[6] Wang Z and Bovik A C 2009 Mean squared error: Love it or leave it? A new look at signal fidelity measures *IEEE Signal Processing Magazine* **26** 98-117.
[7] Brunet D, Vrscay E R and Wang Z 2010 Structural similarity-based approximation of signals and images using orthogonal bases *Proc. Int. Conf. Image Anal. Recognit*. (Povoa de Varzim) pp 11-22.
[8] Rehman A, Rostami M, Wang Z, Brunet D and Vrscay E R 2012 SSIM-inspired image restoration using sparse representation *EURASIP J. Adv. Signal Process.* **16** (2012).
[9] Wang Z, Simoncelli E P and Bovik A C 2003 Multi-scale structural similarity for image quality assessment *Asilomar Conference on Signals (Systems & Computers)* (Pacific Grove) vol 2 pp 1398-402.
[10] Zhao H, Gallo O, Frosio I and Kautz J 2017 Loss functions for image restoration with neural networks *IEEE Transactions on Computational Imaging* **3** 47-57.