Comput Syst Sci & Eng (2018) 2: 105–113

International Journal of
Computer Systems
Science & Engineering

Automated and precise event detection method for big data in biomedical imaging with support vector machine

Lufeng Yuan*, Erlin Yao, and Guangming Tan

Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China {yaoerlin,tgm}@ncic.ac.cn

This paper proposes a machine learning based method which can detect certain events automatically and precisely in biomedical imaging. We detect one important and not well-defined event, which is called flash, in fluorescence images of Escherichia coli. Given a time series of images, first we propose a scheme to transform the event detection on region of interest (ROI) in images to a classification problem. Then with supervised human labeling data, we develop a feature selection technique to utilize support vector machine (SVM) to solve this classification problem. To reduce the time in training SVM model, a parallel version of SVM training is implemented. On ten stacks of fluorescence images labeled by experts, each of which owns one hundred 512 ·512 images with in total 4906 ROIs and 72056 labeled events, event detection with proposed method takes 19 seconds, while human labeling roughly costs 60 hours. With human labeling as the standard, the accuracy of our method achieves an F-value of about 0.81. This method is much faster than human detection and expects to be more precise with bigger data. It also can be expanded to a series of event detection with similar properties and improve efficiency of detection greatly

Keywords: Biomedical imaging; event detection; machine learning; support vector machine; big data

1. INTRODUCTION

As fluo-labelled technology and optical image technology develop and fluorescence detection is highly sensitive, fluorescence imaging is used for cell identification and sorting in flow cytometry, and to reveal the localization, movement of intracellular substances [1]. With fluorescence technique, large fluorescence images are generated, collected and analyzed in many fields such as molecular biology, neurology and bioinformatics. Although human labeling is the most precise in pattern recognition of the data in biomedical imaging, it is very time-consuming [2] and inefficient when the data is big [3]. For example, demarcation of appropriate landmarks for temporal lobe structures in linear measures of images can take up to 3 hours [4]. Even some optimal technologies can be used to reduce time, automation is still a problem [5]. Automated and efficient method for data analysis is needed.

The conventional rule and threshold based methods are intuitive and simple, however they are not precise enough with big data. Rule and threshold based methods are like expert systems, which work well with small data, but will lose its efficiency with big data. The reasons lie in two points: first, the number of rules cannot be big enough to reflect the intrinsic characteristics of big data; second, the thresholds used in each rule are not accurate for big data. Machine learning based method may act as a possible solution. One advantage of machine learning based method is that the whole process is automated, because algorithms are trained by datasets and carry out without human intervention. The other advantage is that the method will be more precise with more data. There have been a lot of successful examples in biological processing using machine learning, especially Support Vector Machine (SVM) [6]. In March of 2009, a glucose-

*Corresponding Author. E-mail: yuanlufeng@ncic.ac.cn

binding site classifier algorithm based on SVM was developed by Nassif et al. This algorithm can classify glucose binding sets and non-glucose binding sites in the result of 89.66% sensitivity and 93.33% specificity [7]. SVM was also used to detect protein-protein interaction (PPI) and got an accuracy of 87.98% [8]. Muda et al presented a two-layer SVM classifiers to detect remote protein homology and fold. Their model significantly improved the performance for three different structural classification of proteins (SCOP) datasets that raised 4.19%, 4.57% and 4.03% respectively compared to other best methods in 2011 [9]. Ciresan et al used a special type of Convolutional Neural Network (CNN) as a pixel classifier to segment neuronal membranes in the ISBI 2012 EM Segmentation Challenge. Their network won the championship and furthermore their method even performed better than human observer in pixel error [10].

Even machine learning algorithms have made great progress, there are two key challenges applying machine learning. First, the choice of features for algorithms is very difficult. In most cases, features only are considered by experts just like Nassif et al. Second, machine learning algorithms often cause huge computational cost. Quoc et al trained a 9-layered locally connected sparse autoencoder on a cluster with 1,000 machines (16,000 cores) for three days [11]. These challenges limit the wide use of machine learning.

For the detection of one special and important event in biomedical imaging, this paper proposes a machine learning based method which is both automated and precise with big data. Given time series of biomedical images, first we propose a scheme to transform the event detection on region of interest (ROI) in images to a classification problem. Then, with supervised human labeling data we develop a feature selection technique to utilize SVM to solve this classification problem. To reduce the time in training SVM model on big data, a parallel version of SVM training is implemented.

2. BACKGROUND AND OVERALL SCHEME

In this section, we will give a brief introduction to flash event that we detect in our work, and describe overall scheme of our method.

2.1 Importance of Flash Event

According to the finding of Wang et alâĽ™s study [12], individual mitochondria generate quantal bursts of superoxide in the matrix, called superoxide flashes when mitochondria lie in resting conditions. In main opinion, superoxide flashes play a part in a functional linking between transient mitochondrial permeability transition pore (mPTP) opening and electron transport chain (ETC) dependent reactive oxygen species (ROS) generation. Superoxide flashes compose elementary and indispensable incidents of quantal ROS generation and supply a crucial source of superoxide production. Relying on different situation, ROS can perform either positive effects or negative effects. There are many evidences demonstrating that normal and balanced levels of ROS are essential in regulating diverse cellular processes including gene translation [13], Ca2+ spark generation [14] and ion channel/transporter operation [15]. However, ROS will exert negative influences if the balance of ROS is broken. When cell generates too many ROS, ROS can cause apoptotic and necrotic damage or death of cell and tissue [16] and pathogenesis of a series of clinically distinguishing diseases and disharmony including diabetes, atherosclerosis, and neurodegeneration [17, 18]. If ROS damage is accumulated and then presents lasting and systemic, cell senescence and aging would be inevitable [19]. Considering above evidences, superoxide flash is key to research ROS and understand elementary intracellular and intercellular signaling processes. The detection of superoxide flash is typical event detection from fluorescence images. Fig. 1A describes a procedure of superoxide flash [12] in a single-mitochondrion of a rat cardiac myocyte which lasts about 20 seconds and intensity of ROI becomes from low to high and go back low finally.

In our work, we focus on another special kind of superoxide flashes happened in Escherichia coli. The original dataset we analyze is a stack consisted of one hundred fluorescence images of Escherichia coli whose size is 512 pixels with 512 pixels. The duration of stack is 100 seconds. Every stack records tens of to hundreds of Escherichia coli and each Escherichia coli perhaps occurs one flash, several flashes or none in the stack. Because of large number of Escherichia coli and randomness of flashes, a professional person needs about ten hours to detect all of flashes in one stack uninterruptedly. The main difficulty of flashes detection is that flash is still an unknown phenomenon and researchers can't build valid model. In these situations, automated detection for unknown flash events is necessary and significant. The original stack of Escherichia coli is shown in Fig. 1B.

2.2 The Scheme of Overall Solution

We register stack first because objects in original stack often drift and shake as cell movement and camera shake. Then we regard each Escherichia coli as a ROI and segment it from image. Only Escherichia coli are injected into fluorochrome and background is black enough, so we can segment cells easily by fluorescence intensity. After cell segmentation, number of each cell's pixels and the intensity of each pixel can be counted. We use cell's average fluorescence intensity at a certain second, which is computed by division between sum of fluorescence intensity of pixels and the number of pixels, as a representative of the whole cell at this second. This representative is based on observation that all parts of Escherichia coli have the same fluorescence intensity trend when flash happens. By this transformation, original 3D image becomes a 2D image called trace, which vertical coordinate is average fluorescence intensity and horizontal coordinate is time. A stack of one hundred images becomes hundreds of traces that one trace, corresponding to one Escherichia coli, means change of a cell in 100 seconds by above processing. In observation and experiments, there is a conclusion that flash always happens in the peak of trace but peak may not be flash.

We can find out all of peaks in a trace and every peak must be a flash or not. That means the problem of event detection becomes classification problem. We use three methods to solve this problem. The first method is human labeling. By human

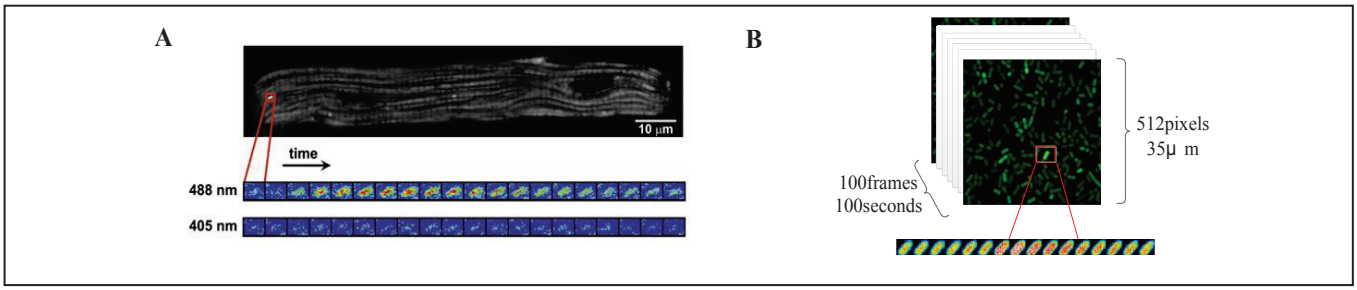

Figure 1 A: superoxide flash image in a single-mitochondrion of a rat cardiac myocyte. B: the original stack of superoxide flashes in Escherichia coli. (only first and last slice shown).

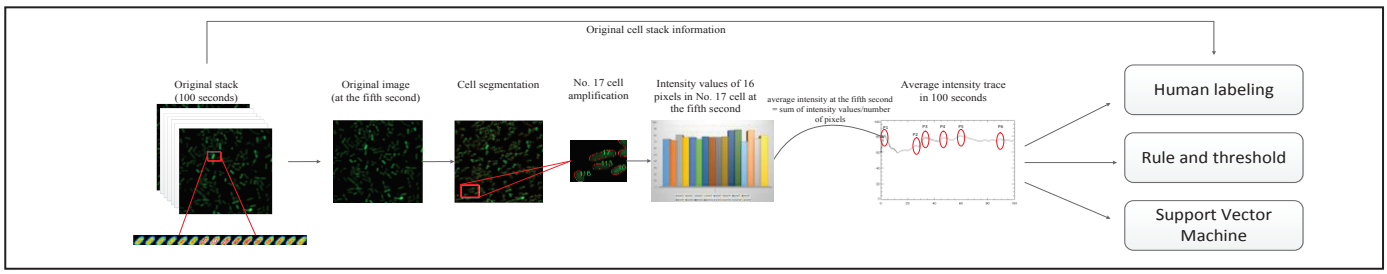

Figure 2 Scheme of overall solution.

labeling, all the data sets are labeled by experts, information of all the traces is collected and standard sets for another two methods are generated. The second method is rule and threshold method. We propose several rules to define a typical flash, build an abstract model to describe flashes, compute the probability of a peak being a flash. A threshold of probability is decided to whether a peak is a flash or not. The last method is machine learning based method using Support Vector Machine. SVM model is used as a classifier. Model is trained by a part of labeled data sets. Then this model predicts the other data sets to find flashes. The whole scheme of overall solution is demonstrated in Fig. 2.

2.3 Rule and Threshold based Method

Rule and threshold based method is intuitive. If flash can be defined by some rules, we can build models to compute probability of a peak. The peak whose intensity exceeds threshold is a flash, problem can be solved [20]. We analyze 11383 labeled flashes and sort out some flashes as typical flashes. A typical flash is presented in Fig. 3A. By analyzing typical flashes and expert experience, three rules are abstracted.

Rule1: A flash should have an obvious intensity change.

Rule 2: A flash should last for enough time.

Rule 3: A flash should be symmetrical to some extent as the center on top.

For each rule, we apply a factor to indicate it. Every factor is listed as follows.

$$r_1 = 1 - a^{-\frac{\text{peak}}{\text{height}}} \tag{1}$$

$$r_2 = 1 - a^{-\frac{\text{duration}}{\text{width}}} \tag{2}$$

$$r_3 = 1 - a^{-\frac{\text{delta}}{\text{delta}\,c}} \tag{3}$$

$$R = r_1 \times r_2 \times r_3 \tag{4}$$

Subject to

$$a > 1 \tag{5}$$

$$\text{delta} = \text{peak}/\text{duration} \tag{6}$$

$$\text{delta}_c = \text{height}/\text{width} \tag{7}$$

Peak means the maximum value of peak, duration means time from start to end of peak, height and width are statistical peak and duration of known labeled flash, is constant greater than 1. Three factors are all monotonic increasing. The maximum is 1 for each factor if peak, duration or delta is infinity. That means it would be a different peak with ordinary peaks and we confirm this peak is a flash with probability 1. Final probability which is product of three factors means only if peak, duration and delta are all large enough, probability can be large enough and this peak would be a flash probably.

This simple model is built on the base of typical flashes and can be a good abstract for typical flashes. However, real flashes have many complex forms. In Fig. 3, there are three flashes different from typical flash. Therefore, we develop a flash detection method based on SVM, which only uses a small number of labeled data sets.

3. MACHINE LEARNING BASED METHOD

3.1 Overview

Support Vector Machine (SVM), as a typical and efficient supervised machine learning algorithm [21], has been hugely successful in practice. In a recent experimental study [22], the SVM with Gaussian kernel [23] implemented using LIBSVM [24] [25] has the second best classification accuracy on the whole UCI data

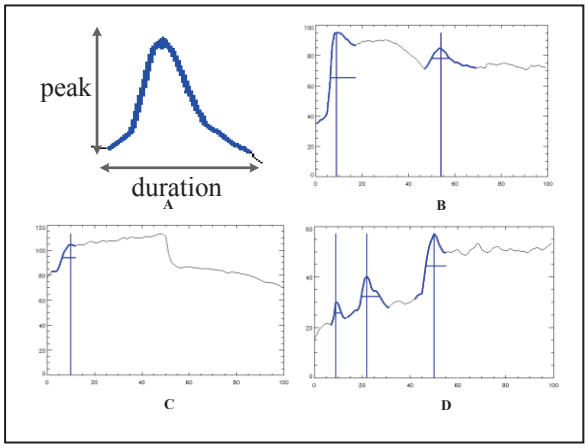

Figure 3 Fig. 3. A: a typical flash. B-D: Three atypical flashes. B: intensity of trace is increasing rapidly. C: intensity of trace declines obviously. D: three flashes' average intensities change continuously and flashes are asymmetric.

base among the existing 179 classifiers. Even original SVM is a two-class classification model with good generalization performance [26], SVM model can also be used for one-class analysis [27], then detects novelty further [28].

SVM need enough features to identify flashes. If there aren't enough exact features to present flashes clearly, the performance will be bad. But extracting exact features is very difficult if nature of flashes is unknown. Besides feature choice, the second difficulty is data skew. Probability of flash occurrence is different obviously in different species. If probability is low, most peaks aren't flashes. This phenomenon is called data skew. Accuracy, which is defined as the ratio of correct predicting samples to all of samples, is not appropriate for evaluating dataset with data skew. In our experiments, the distribution of flashes and non-flashes is not even in peaks, with a proportion about 1:33 within all the data. That means if an algorithm classifies all of samples as non-flashes, its accuracy reaches 97%. Task of detection can be seen as retrieving all flashes, so F value in information retrieval can be used to evaluate method. The F value is defined as [29]:

$$F = 2 \times \text{ precision } \times \text{recall}(\text{precision} + \text{recall}) \quad (8)$$

Where

$$\text{precision} = (\text{no. returned flashes})(\text{no. returned peaks}) \quad (9)$$

$$\text{recall} = (\text{no. returned flashes})(\text{no. all the flashes}) \quad (10)$$

We use both accuracy and F value as measurement and show F value is more reasonable in the flash detection.

For solving feature choice and data skew, we present an automated detection and analysis method to take full advantage of all the data sets. This method is composed of seven modules as shown in Fig. 4.

The first two modules, input module and preprocessing module, are shared by human labeling, rule and threshold method and machine learning based method. Stacks of images are read and prepared for next modules in input module. Behind input module, it is pre-processing module which includes image registration aligning the same cells in different images, cell segmentation labeling every cell and intensity average computing average of cell's fluorescence intensity and transforming cell's intensity change to a trace. After it, a stack of images becomes many traces. In skew elimination module, we introduce accuracy and F value to avoid data skew. Features of a peak are selected in feature selection module. After feature selection module, some significant features are presented and model can be trained in model train module. With respect to the high computational requirement of SVM, We implement model train module in parallel with message passing interface (MPI) to reduce train time. Trained model can predict another trace, detect and analyze flashes in prediction module. The results generated in model prediction module are concluded for meaningful information to researchers in result collection module.

3.2 Feature Selection

Feature selection has large impact on classification accuracy. If all characteristics have been recognized fully, we can select exact features to build model. Unfortunately, characteristics of flashes are not well understood. So we face two problems. One problem is what kinds of features should be selected for flashes. The other problem is how many features are enough to detect flashes and classify peaks [30].

For the first problem, we decide to abstract features from peaks themselves. We choose three intuitionistic features: (i) amplitude which is increasing ratio of intensity of one peak relative to the intensity of its endpoint, (ii) width which is time interval between left and right endpoints and (iii) ratio of amplitude and width which reflects the shape of one peak. These features are also corresponding to the rules in rule and threshold method. We speculate three-feature-set is the minimal set to diagnose flashes but it perhaps isnâŁ™t enough for some persistent disturbance and fluorochrome's change. As a contrast to three-feature-set, we abstract a new feature set consisting of nine features by expertise experience, which are left amplitude, right amplitude, left width, right width, left slope, right slope, the average intensity of trace, distance to the last peak, distance to the next peak. For verifying performance of features, we test a small data set consisting of ten stacks with 5831 peaks and 66 flashes. In ten data sets, one set is left as test set, training set number is 3, 6, 9 respectively. Models are trained using both three-feature-set and nine-feature-set. We modify libsvm to implement our algorithm, classifier is C-SVC using c meaning cost and g meaning gamma as both parameters of classifier [24]. The ranges of parameters are set as: $2^{-15} \leq g \leq 2^3$, $2^{-5} \leq c \leq 2^{15}$, all pairs of c and g are tested and results of best F value are listed. The experiment results are summarized in Table 1.

From these results, we can find accuracies and F values of two sets only have slight difference, and accuracy of three-feature-set even has better performance. The fact inspires us to focus on the second problem: how many features is enough. In our method, we use data over-fitting to verify completeness of features. Data over-fitting happens when model is trained excessively in the training process. Because kernel function and soft margin can compute and classify samples in a high-dimensional space in SVM, data over-fitting means that all samples can be separated linearly in this space when selected features of data are adequate. For verifying this idea, we perform the second experiment to over-fit data using three-feature-set and nine-feature-set. This experiment is carried in two datasets. One is above dataset, the

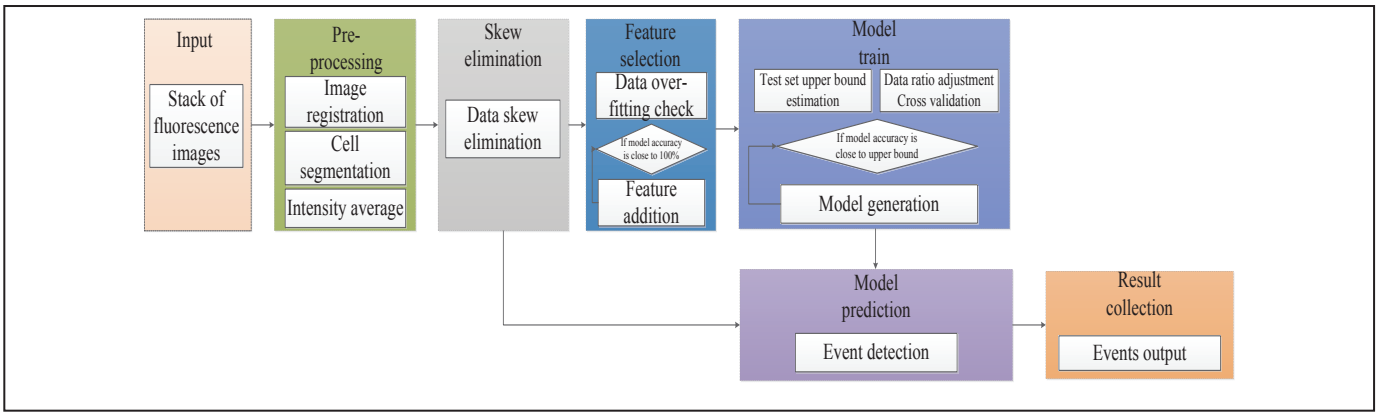

Figure 4 Overview of machine learning based method (see text).

other is a median dataset which has 30 stacks with 97991 peaks and 2095 flashes. Results are demonstrated in Table 2.

When three features are used on small dataset, the best F obtained using C-SVC with data over-fitting is 0.8852 while on median dataset, F descents to 0.7583. This indicates three features are not enough to model problem. Then nine features are used. Not only for small set, but also for median set, F approaches 1 in SVM. At the same time, generated support vectors (SVs) are a smaller part of flashes and peaks than that of three-feature-set. This indicates result that all of samples are almost separated is gained by computing features instead of recording all the samples. Nine features are adequate to model the classification problem.

3.3 Parallel Training of SVM Model

When training set is huge, training SVM model will need a lot of time. Furthermore time will increase rapidly using cross validation. In our method, we use cross validation to find best parameters for training [31]. Because there are hundreds of thousands of samples, computation time will be long if fold of cross validation is large. When fold of cross validation is 9, one month more is needed to get the best parameters. For increasing efficiency of method, we exploit several methods to speed up training model.

In the process of training SVM, most of time is dominated by kernel evaluations, which cost more than 70% training time. Because kernel matrix is a symmetric matrix computed by pairs of samples in training dataset, it is decided when training dataset is certain. Since kernel matrix is shared in cross validation, one strategy is to pre-compute kernel matrix and store it in main memory [33]. By this way, kernel evaluation is just read from kernel matrix so that computation time is reduced when kernel matrix is used many times. But this way has two drawbacks in practice. One drawback is that size of kernel matrix will increase quadratic with size of training dataset increases. If size of dataset is N, size of kernel matrix is $(N * N)/2$. Because our single node of clusters has 66GB memory, that means size of training dataset would not exceed 134000. However, our whole dataset owns 385996 samples whose kernel matrix will be about 555GB and this kernel matrix can't be stored in memory. The other drawback is that perhaps only a part of kernel matrix is used in cross validation. There exist some samples which are impossible to be support vectors, however their kernel evaluations are also prepared. This will waste lots of time. That is why we only get 30% performance improvement when we train a dataset owning 100000 samples.

Considering drawbacks of kernel matrix precomputation, we focus on optimize and speed up single training for reducing computation time. We exploit parallelism based on MPI. Kernel algorithm of libsvm [32] is Sequential Minimal Optimization (SMO) [34] as below.

SMO Algorithm

1. Initialize $\alpha_r = 0, G_r = -1, r = 1, \ldots, l$
2. Select violating pair $\{i, j\}$. If $\{i, j\}$ is a violating pair, goto step 3. Otherwise, stop.

   2.1. $i \in \arg\max_t \{-y_t G_t | t \in I_{\text{up}}(\alpha)\}$,

   $$I_{\text{up}}(\alpha) = \{t | \alpha_t < C,\ y_t = 1 \text{ or } \alpha_t > 0,\ y_t = -1\}$$

   2.2. $j \in \arg\min_t \left\{ -\frac{b_{it}^2}{\bar{a}_{it}} | t \in I_{\text{low}}(\alpha), -y_t G_t < y_i G_i \right\}$

   $$I_{\text{low}}(\alpha) = \{t | \alpha_t < C,\ y_t = -1 \text{ or } \alpha_t > 0,\ y_t = 1\}$$

3. Update $\alpha_i, \alpha_j, G_r, r = 1, \ldots, l$

   3.1. Count $\bar{a}_{ij}, b_{ij}$, old $\alpha_i = \alpha_i$, old $\alpha_j = \alpha_j$,

   $$\text{old } G_r = G_r,\ r = 1, \ldots, l$$

   3.2. $\text{sum} = y_i \text{old}\alpha_i + y_j \text{old}\alpha_j$

   $$\text{tmp} = \text{old}\alpha_i + y_i \frac{b_{ij}}{\bar{a}_{ij}} \text{ and } 0 \le \text{tmp} \le C$$

   3.3. $\alpha_j = y_j \text{sum} - y_i y_j \text{tmp}$ and $0 \le \alpha_j \le C$

   $$\alpha_i = y_i \text{sum} - y_i y_j \alpha_j$$
   $$G_r = \text{old } G_r + Q_{ir}(\alpha_i - \text{old } \alpha_i) + Q_{jr}(\alpha_j - \text{old}\alpha_j)$$

Go back to step 2.

Table 1 Results of three-feature-set and nine-feature-set when training set number is 3, 6, 9.

Training set number	Feature number	c	G	accuracy	F
3	3	32	8	93.0029%	0.6269
6	3	8	8	92.8571%	0.6142
9	3	8	2	92.6628%	0.5995
3	9	128	8	91.5938%	0.616
6	9	8192	8	92.1283%	0.641
9	9	8192	8	92.2741%	0.6411

Table 2 The feature selection results using data over-fitting.

Peak	Flash	Feature number	c	g	Accuracy	F	P	R	SV(flash)	SV(peak)
5831	66	3	8192	8	99.76%	0.8852	0.9643	0.8182	35	36
5831	66	9	32	2	100%	1	1	1	51	83
97991	2095	3	32768	8	99.09%	0.7583	0.862	0.6769	985	995
97991	2095	9	32768	8	99.98%	0.995	0.9976	0.9937	805	988

In this algorithm, $\bar{a}_{ij}$, b_{ij} are defined as in (11, 12). Q is kernel matrix and Q_{ij} is kernel evaluation between sample i and sample j. τ is a small positive number.

$$\bar{a}_{tk} = \begin{cases} a_{tk} = Q_{tt} + Q_{kk} - 2y_t y_k Q_{tk}, \ if, a_{tk} > 0 \\ \tau, \ \text{otherwise} \end{cases} \tag{11}$$

$$b_{tk} = -y_t G_t + y_k G_k \tag{12}$$

The first idea to reduce computation time of SMO is to improve efficiency of computing kernel matrix because it is bottleneck. In SMO, kernel matrix often accesses by row and the i-th row of kernel matrix means kernel evaluations between sample i and every sample of the whole dataset. So we use single instruction multiple data (SIMD) mode to evaluate the row of kernel matrix [35]. First, entire training dataset is equally partitioned into small subsets according to the number of processors used. Secondly, sample i is broadcasted to every processor and each of subsets is also distributed to corresponding processor. Finally, every processor computes a part of the i-th row of kernel matrix based on sample i and distributed training subset. The whole i-th row of kernel matrix consists of each part of kernel evaluations on each processor. Because evaluations on each processor are carried out at the same time, lots of computation time will be saved. If there are p processors used and Ts denotes total time of kernel evaluations in sequential SMO, the amount of kernel evalu-ations time in SIMD mode could be almost reduced to Ts/p.

Besides evaluating kernel values, selecting violating pair $\{i, j\}$ can also be performed in parallel mode. In sequential SMO algorithm, violating pair $\{i, j\}$ is selected based on objective function (13) (14) and objective functions are computed on the entire training dataset.

$$\text{obj}_i = \max_t(-y_t G_t), \ t \in I_{\text{up}}(\alpha) \tag{13}$$

$$\text{obj}_j = \min_t\left(-\frac{b_{it}^2}{\bar{a}_{it}}\right), t \in I_{\text{low}}(\alpha), -y_t G_t < y_i G_i \tag{14}$$

In parallel mode, each processor could compute a local optimal obji and objj using its assigned subset and find a corresponding local pair $\{i, j\}$. Then the local optimal obji and objj of each processor will be compared. Global obji is the maximum value of local obji and global objj is the minimum value of local objj. In MPI library, compare of local values can be implemented conveniently using the function MPI_Allreduce. When the global obji and global objj are determined, corresponding global violating pair $\{i, j\}$ is also selected, then broadcasted to each processor.

After selecting of violating pair $\{i, j\}$, the computation load is concentrated on update α_i, α_j and G_r in step 3 of sequential SMO. In this step, α_i and α_j are associated with violating pair $\{i, j\}$ and these values can be evaluated directly by using any processor. Array G consists of the gradient value of every sample and it can be updated by parallel mode, too. Because each processor owns a distributed training subset, it can keep a corresponding G subset. When new violating pair $\{i, j\}$ is sent to each processor, it can update local G subset by using new violating pair and local subset. By means of GATHER operation in MPI library, the entire G array is composed of local G subset on each processor easily.

The left parts of algorithm only take little time and can be handled by one processor. Communication among processors is required and costs some time. There is three global communications which are selection of global i and j and combination of gradient array G. The communicated data is small and communication can overlap computation by designing parallel algorithm carefully and using asynchronous communication, so communication time can be a little proportion of the whole training time. A brief summary of parallel algorithm is listed as follows. Corresponding procedure of parallel algorithm is shown in Fig. 5.

Parallel SMO Algorithm

1. Initialize $\alpha_q^k = 0$, $G_q^k = -1$, $(q = 1, \ldots, l_k,\ k = 1, \ldots, p)$. p is the number of processor and l_k is the size of distributed training subset on processor k.
2. Select violating pair $\{i, j\}$. If $\{i, j\}$ is a violating pair, goto step 3. Otherwise, stop.
   - 2.1. Calculate local obj_1^k, obj_j^k and corresponding local i, j on each processor.
   - 2.2. Reduce global obj_i, obj_j and determine global violating pair $\{i, j\}$ broadcast $\{i, j\}$ to every processor.
3. Update $\alpha_i, \alpha_j, G_r, r = 1, \ldots, l$
   - 3.1. update α_i, α_j on a certain processor according to $\{i, j\}$.
   - 3.2. broadcast α_i, α_j to every processor.
   - 3.3. update local G_q^k $q = 1, \ldots, l_k$, $k = 1, \ldots, p$ of every processsor and combine global G for every processor Go back to step 2.

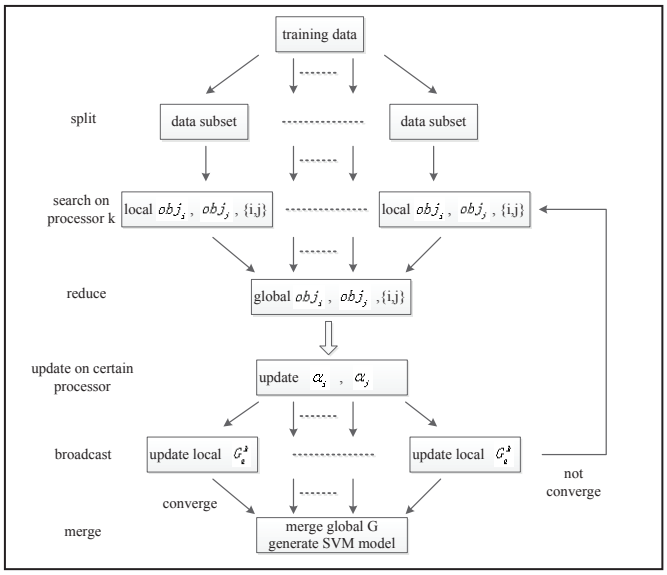

Figure 5 Procedure of parallel SMO algorithm.

4. EXPERIMENT RESULTS

4.1 Experimental datasets

The dataset in experiments includes 100 stacks of fluorescence images, each of which consists of 100 fluorescence images in 100 seconds. This dataset has been labeled manually and we can know details of flash. These stacks have 29048 traces of 29048 cells, 385996 peaks and 11383 labeled flashes.

4.2 Performance of Parallel SVM

We test parallel performance of our method using all the stacks. In this experiment, we test different values of parameter c because training time will be more with larger c. Parallel algorithm is analyzed from training time, communicating ratio time and workload balance. Table 3 depicts training time in sequential SVM and different number of processors. Training time will be reduced as processors increase in both value of c. Maximal speedup can reach 3.35 and 1.62 respectively when c are 1 and 32768. Communicating cost will increase with increasing of processors. Communicating ratio in all training time can reach 23.39% and 39.4% respectively according to 1 and 32768 when we use eight processors in parallel algorithm in Fig 6. Our algorithm implements excellent workload balance. We record workload execution time and compute corresponding ratio of standard deviation and mean value for each processor. When c are 1 and 32768 and eight processors are used, corresponding ratio are 5.77% and 5.07%. That means our parallel algorithm makes full use of each processor and implements excellent workload balance.

Table 3 Results of MPI speedup. Unit is second.

c	sequential SVM	Parallel SVM			
		1P	2P	4P	8P
1	396.59	590.1	305.73	168.37	118.25
32768	27498.99	53246.1	47322.6	35182.08	16995.01

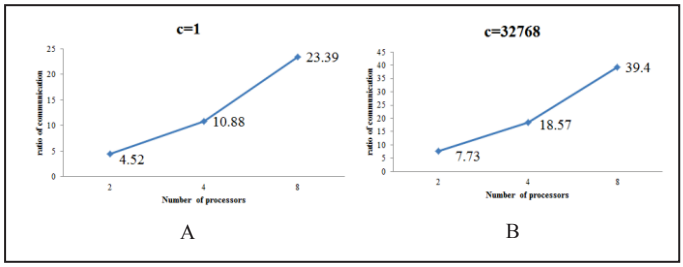

Figure 6 Ratios of communication in different c. (A) c=1. (B) c=32768.

4.3 Prediction Precision of rule and threshold method and SVM

We do experiments to compare rule and threshold method and SVM method. In our experiments, we carry parameter tuning methods to make the results reasonable and correct [36]. The whole data are split into training and test sets resulting in 7 cases, with proportion between size of training and test sets is respectively: 9:1, 4:1, 2:1, 1:1, 1:2, 1:4 and 1:9. Results are given in Table 4 and 5. In rule and threshold method, height and width, which are corresponding average of labeled flashes in training set, will change as training set changes. We suppose a peak, whose peak and duration are the same with height and width, is a flash with at least 90% probability. Then we can get an approximate equation (15) and α is set as 25 by this equation.

$$(1 - a^{-1})^3 \approx 0.9 \tag{15}$$

Then we change threshold from 0.1 to 0.95 with stride 0.05 and compute the best results in different threshold, training set and test set. Table 4 gives the best F value of rule and according threshold. We can see F value is stable between 0.7 and 0.727. Stability means typical flashes have a large proportion in flashes. Even weakly, F value still declines when training set is decreasing and test set is increasing. This phenomenon can be explained. When test set is increasing, there are more complex flashes. However original model has limited ability to measure flashes and perhaps have bad result to larger set. Because typical flashes are in majority, F value just declines a little.

Table 5 gives the best F obtained using optimizations that traversing all c and g in cross validation of SVM method. Ranges of parameters are respectively as: $2^{-15} \leq g \leq 2^3$, $2^{-5} \leq c \leq 2^{15}$. Fold v of cross validation is $2 \leq v \leq 9$. First, F values of SVM method are all better than that of rule and threshold method in each ratio except 1:9. Because performance of SVM depends on training set, we think training set is too small and trained model is weak so that SVM gets lower performance in the ratio of 1:9. The other conclusion is F value increases more quickly in SVM when training set is increasing. Performance of rule and threshold method is determined if model is confirmed. But SVM method can build stronger model to perform better as training sets is more. Two methods have different increase when ratio changes uniformly. F value is just 0.817 even in SVM method. Is this the best result of SVM? To answer this question, both training set and test set are set as all of 100 stacks and we test whether SVM method can detect all flashes. The result is listed in Table 6. F value can reach 0.938, which means SVM method can separate the most of flashes from peaks. So we infer that our datasets are not enough and a better model would be trained if there are more datasets.

Finally, we analyze flashes detected by SVM, find some faint and unconspicuous flashes missing are also detected by model. One example is demonstrated in Figure 7. This proves that SVM method is effective and can provide helpful supplement for manual work.

Table 4 The best F obtained using rule and threshold method.

Train:Test	threshold	Accuracy	F
9:1	0.9	99.25%	0.727
4:1	0.85	98.85%	0.715
2:1	0.85	98.69%	0.713
1:1	0.85	98.06%	0.701
1:2	0.9	98.81%	0.712
1:4	0.9	98.75%	0.701
1:9	0.9	98.7%	0.7

Table 5 The best F obtained using SVM method.

Train:Test	v	c	g	Accuracy	F
9:1	2	8	8	99.45%	0.817
4:1	2	512	2	99.26%	0.804
2:1	2	512	2	99.13%	0.804
1:1	2	128	8	98.72%	0.764
1:2	2	32	8	98.86%	0.76
1:4	2	8	8	98.82%	0.759
1:9	2	32	8	98.08%	0.662

Table 6 The best F obtained using SVM method.

Train:Test	v	c	g	Accuracy	F
9:1	2	8	8	99.45%	0.817
4:1	2	512	2	99.26%	0.804
2:1	2	512	2	99.13%	0.804
1:1	2	128	8	98.72%	0.764
1:2	2	32	8	98.86%	0.76
1:4	2	8	8	98.82%	0.759
1:9	2	32	8	98.08%	0.662

Table 7 Separation of flashes from peaks in 100 stacks.

Peak	Flash	Feature number	c	g	Accuracy
385996	11383	9	32768	8	99.63%
F	P	R	SV(flash)	SV(peak)	
0.9383	0.9471	0.9296	3282	4820	

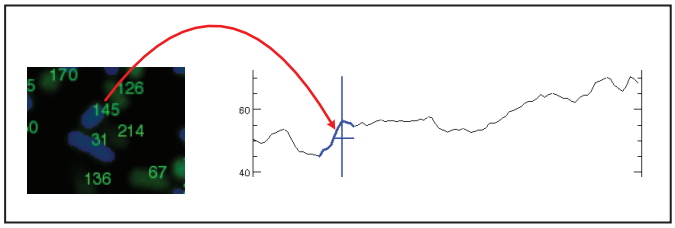

Figure 7 A flash missed by experts and detected by model. No 145 cell occurred a faint and nonstandard flash. This flash is missed by experts but detected by our model.

5. DISCUSSION AND FUTURE WORKS

In this paper, we use SVM to implement our method. However, there are many other outstanding models and algorithms which can be used to solve the classification problems. Perhaps certain special model exists which can be suited of classifying flash events. We use SVM due to the following reasons. First, considering that flash has not been analyzed by any machine learning method before, we decide to apply mature technology to analyze flash and hope to get some reasonable results at the beginning. Secondly, SVM owns outstanding classification performance and performs steadily especially for multiple-dimensional data. Furthermore, SVM has high computing efficiency to reduce a lot of running time. Meanwhile, it is easy to be parallelized and can save time further. Thirdly, SVM owns firm basis in mathematics. Its result and computing procedure are useful for us to build a more practical biological model of flash.

In our paper, the method based on SVM to analyze flash by standard data is successful. But that does not mean the whole work finishes. We plan to research and analyze flashes further from three aspects in the future.

1. Our work investigates flashes of Escherichia coli mainly. However, flash is a kind of life phenomenon exists in different species and tissues extensively. So far we have also tried to detect flash in neuron by our method. Some experimental results have also shown our method makes good

effect to reduce the time of detecting flash and increase the accuracy of analyze flash. In fact, flashes in different species and tissues are very different and lack of corresponding standard data. So we will also consider to adopt unsupervised learning such as clustering to analyze flashes of neuron. We hope to find some similarities in different flashes of different species and tissues.

2. We try to describe flash events quantitatively and build an accurate mathematical model of flash according to the model trained by SVM and standard flash data.

3. Extracting features of flash mainly depends on expert experience now. We want to utilize some algorithms to optimize feature extraction for better results. At present, we are trying to extract more accurate features by training deep learning network to get better classification results.

6. CONCLUSION

In this paper, we implemented a systematic and general method based on SVM to detect and analyze flash that is an important biological event in Escherichia coli. This simple and efficient method can take full advantage of all labeled data, receive more satisfactory prediction than rule and threshold method and find some flashes omitted by experts. Sufficient results indicate that performance of our method will be better if we have more labeled data. The most important contribution of our method is that this method not only can be used in flash, it but also can be used in other fluorescence events.

ACKNOWLEDGMENT

We would like to express our gratitude to all revieweråŁ™s constructive comments for helping us polishing this paper. This work is supported by The National Key Research and Development Program of China (2016YFB0201305), National Science and Technology Major Project(2013ZX0102-8001-001-001) and National Natural Science Foundation of China, under grant no. (91430218, 31327901, 61472395, 61272134, 61432018).

REFERENCES

1. J. R. Lakowicz, âŁ˜âŁ˜Principles of fluorescence spectroscopy.âŁ™âŁ™ Kluwer Academic/Plenum, NewYork,1999.
2. L. A. Dade, F. Q. Gao, N. Kovacevic, P. Roy, C. O. Rockel, C. M. Toole, N.J. Lobaugh, A. Feinstein, B. Levine, S.E. Black, “Semiautomatic brain region extraction: a method of parcellating brain regions from structural magnetic resonance images.” Neuroimage 2004, 22, pp. 1492-1502.
3. W. C. Tiffany, T. Shinichiro, H. Kie, E. P. Christina, L. J. Peggy, M. L. Kusano, C. B. Caldwell, J. Ramirez, S. Black, N. P. Verhoeff. “Comparison of manual and semi-automated delineation of regions of interest for radioligand PET imaging analysis.” BMC Nuclear Medicine 2007, 7:2
4. J. Crane, “Right medial temporal-lobe contribution to object-location memory.” Doctoral Thesis. McGill University, Montreal, 1999.
5. H. Alishavandi, G.H. Gouraki, H. Parvin, “An enhanced dynamic detection of possible invariants based on best permutation of test cases”. Comput. Syst. Sci. Eng. 31(1) (2016)
6. B. Naul, “A review of support vector machines in computational.” June 6, 2009
7. H. Nassif, H. Al-Ali, S. Khuri, K. Walid, “Prediction of protein-glucose binding sites using support vector machines.” Proteins: Structure, Function, and Bioin-formatics, 2009.
8. H.W. Liu. “Protein-Protein interaction detection by SVM from sequence.” The Third International Symposium on Optimization and Systems Biology (OSBâŁ™09) Zhangjiajie, China, September 20-22, 2009. pp. 198-206.
9. H. M. Muda, P. Saad, R. M. Othman. “Remote protein homology detection and fold recognition using two-layer support vector machine classifiers.” Computers in Biology and Medicine. Volume 41, Issue 8, August 2011, pp 687-699.
10. D. Ciresan, A. Giusti, L. M. Gambardella, J. Schmidhuber. “Deep neural networks segment neuronal membranes in electron microscopy images.” Advances in Neural Information Processing Systems 25. 2012
11. V. L. Quoc, A. R. Marc, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, Y. Ng Andrew. “Building high-level features using large scale unsupervised learning.” Proceedings of the 29th International Conference on Machine Learning (ICML), Edinburgh, Scotland, UK, 2012.
12. W. Wang, H. Q. Fang, L. Groom, A. Cheng, W. R. Zhang, J. Liu, X. H. Wang, K. T. Li, P. D. Han, M. Zheng, J. H. Yin, W. D. Wang, M. Mattson, J. P. Y. Kao, E. G. Lakatta, S. S. Sheu, K. F. Ouyang, J. Chen, R. T. Dirksen, H. P. Cheng. “Superoxide flashes in single mitochondria.” Cell, 134(2),pp. 279-290, 2008.
13. W. Droege. “Free radicals in the physiological control of cell function.” Physiol. Rev. 82, pp. 47-95. 2002.
14. E. V. Isaeva, V. M. Shkryl, N. Shirokova, “Mitochondrial redox state and Ca2+ sparks in permeabilized mammalian skeletal muscle.” J. Physiol. 565, pp. 855-872. 2005.
15. A. V. Zima, L. A. Blatter, “Redox regulation of cardiac calcium channels and transporters.” Cardiovasc. Res. 71, pp. 310-321. 2006.
16. P. S. Brookes, Y. Yoon, J. L. Robotham, M. W. Anders, S. S. Sheu, “Calcium, ATP, and ROS: a mitochondrial love-hate triangle.” Am. J. Physiol. Cell Physiol. 287, pp. 817-833. 2004.
17. N. S. Dhalla, R.M. Temsah, T. Netticadan, (2000). “Role of oxidative stress in cardiovascular diseases.” J. Hypertens. 18, 655âŁ“673.
18. J. K. Andersen, (2004). “Oxidative stress in neurodegeneration: cause or consequence?” Nat. Med. 10 (Suppl), S18-S25.
19. K. B. Beckman, B. N. Ames, (1998). “The free radical theory of aging matures.” Physiol. Rev. 78, 547âŁ“581.
20. D. S. Gregory, E. K. Joel, D. S. Michael, W. Jonathan Lederer , H. P. Cheng. “A Simple Numerical Model of Calcium Spark Formation and Detection in Cardiac Myocytes.” Biophysical Journal Volume 75 July 1998. pp. 15âŁ“32.
21. C. Cortes, V. N. Vapnik. “Support-vector networks.” Machine Learning, 20 (3): 273, 1995.
22. M. F. Delgado, E. Cernadas, S. Barro, D. Amorim. “Do we need hundreds of classifiers to solve real world classification problems?.” Journal of Machine Learning Research, 15(Oct), pp. 3133â˜3181, 2014.
23. B. E. Boser, I. M. Guyon, V. N. Vapnik. “A training algorithm for optimal margin classifiers.” In Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory, pp. 144âŁ“152, Pittsburgh, PA, July 1992. ACM Special Interest

Group on Artificial Intelligence.
24. C. C. Chang, C. J. Lin. “LIBSVM: a library for support vector machines.” ACM Transactions on Intelligent Systems and Technology, 2:27:1-27:27, 2011.
25. J. C. Platt. “Fast training of support vector machines using sequential minimal optimization.” Advances in kernel methods, MIT Press, Cambridge, MA, Pages 185âŁ“208, 1999.
26. V. N. Vapnik. “The nature of statistical learning theory.” Springer-Verlag, New York, NY, 1995.
27. L. M. Manevitz, M. Yousef. “One-class SVMs for document classification.” The Journal of Machine Learning Research. Volume 2, 1, March,2002. pp. 139-154.
28. Schölkopf, Bernhard, “Support Vector Method for novelty detection.” NIPS. Vol. 12. 1999.
29. M. D. Christopher, R. Prabhakar, S. Hinrich. “Introduction to information retrieval.” Cambridge University Press, 2008.
30. M. Khonji, A. Jones, Y. Iraqi, “An Empirical Evaluation for Feature Selection Methods in Phishing Email Classification”. Comput. Syst. Sci. Eng. 28(1) (2013).
31. T. Liu, J. Cui, B. Wang, G. Chen, “Parallelization of normal mode-WKBZ computation for ocean acoustic field”. Comput. Syst. Sci. Eng. 27(5) (2012).
32. C. C. Chang, C. J. Lin. LIBSVM 3.20, released on November 15, 2014. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm.
33. A. Athanasopoulos, A. Dimou, V. Mezaris, I. Kompatsiaris. “GPU acceleration for support vector machines”. Proc. 12th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2011), Delft, The Netherlands, April 2011.
34. R.E. Fan, P.H. Chen, C.J. Lin. “Working set selection using second order information for training Support Vector Machines”. Journal of Machine Learning Research 6 (2005) 1889âŁ“1918.
35. B. Chen, C.Y. Duan, X.E. Gao, “A frequent pattern parallel mining algorithm based on distributed sliding window”. Comput. Syst. Sci. Eng. 31(2) (2016).
36. R.D. Al-Dabbagh, S. Mekhilef, M.S Baba, “Parameters’ fine tuning of differential evolution algorithm”. Comput. Syst. Sci. Eng. 30(2) (2015).