

Final Report

W15QKN-05-D-0011, Task Order 43

(September 15, 2009)

Submitted by S. Tewksbury

Contract Number: W15QKN-05-D-0011, Task Order 43

Contract Name: **Embedded Intelligent Sensor Network Systems**

Task 3

Information Assurance in Sensor Networks

Subtask 1

Investigation of acoustic scene analysis and multi-modal sensing

Prof. Hong Man and Yafeng Yin

Department of Electrical and Computer Engineering

Stevens Institute of Technology

Hoboken, NJ 07030

REPORT DOCUMENTATION PAGE		Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a current valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.		
1. REPORT DATE (DD-MM-YYYY) 09-15-2009	2. REPORT TYPE Final Report	3. DATES COVERED (From - To) 08-30-2008 - 09-30-2009
4. TITLE AND SUBTITLE <i>Information Assurance in Sensor Networks</i>		5a. CONTRACT NUMBER W15QKN-05-D-0011
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) <i>Prof. Haibo He. Hona Man and Yafeng Yin.</i>		5d. PROJECT NUMBER
		5e. TASK NUMBER 43
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) <i>Department of Electrical and Computer Engineering Stevens Institute of Technology Hoboken, NJ 07030</i>		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) DoD-ARDEC ACOE Building 407, Picatinny 07806		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION / AVAILABILITY STATEMENT Distribution Statement A Unlimited Distribution		
13. SUPPLEMENTARY NOTES GPDM, Particle Filter, Complex Environment, Multiple-Stage HOG Human Detector 3		
14. ABSTRACT Detection and tracking of a varying number of people is very essential in surveillance sensor systems. In the real applications, due to various human appearance and confounders, as well as various environmental conditions, multiple targets detection and tracking become even more challenging. During this year, our major contributions of multiple targets detection and tracking are as follows: Firstly, we extend the Particle Filter Gaussian Process Dynamical Model (PF-GPDM) to track multiple targets in complex visual environment. With the PF-GPDM, a high-dimensional training target trajectory data set of the observation space is projected to a low-dimensional latent space through Probabilistic Principal Component Analysis (PPCA), which will then be used to classify test object trajectories, predict the next motion state, and provide Gaussian Process dynamical samples for the particle filter. In addition, Histogram-Bhattacharyya and GMM Kullback-Leibler are employed respectively, and compared in the		

15. SUBJECT TERMS GPDM, Particle Filter, Complex Environment, Multiple-Stage HOG Human Detector					
16. SECURITY CLASSIFICATION OF: UNCLASSIFIED			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON Shafik Quoraishee
a. REPORT	b. ABSTRACT	c. THIS PAGE	SAR	93	19b. TELEPHONE NUMBER (include area code) 9737249462

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

ABSTRACT

Detection and tracking of a varying number of people is very essential in surveillance sensor systems. In the real applications, due to various human appearance and confounders, as well as various environmental conditions, multiple targets detection and tracking become even more challenging. During this year, our major contributions of multiple targets detection and tracking are as follows: Firstly, we extend the Particle Filter Gaussian Process Dynamical Model (PF-GPDM) to track multiple targets in complex visual environment. With the PF-GPDM, a high-dimensional training target trajectory data set of the observation space is projected to a low-dimensional latent space through Probabilistic Principal Component Analysis (PPCA), which will then be used to classify test object trajectories, predict the next motion state, and provide Gaussian Process dynamical samples for the particle filter. In addition, Histogram-Bhattacharyya and GMM Kullback-Leibler are employed respectively, and compared in the particle filter as complimentary features to coordinate data used in GPDM. Experimental tests are conducted on the PETS2007 benchmark data set. The test results demonstrate that the approach can track more than four targets with reasonable run-time overhead and performance. Secondly, we propose a new framework integrating a multiple-stage Histogram of Oriented Gradients (M-HOG) based human detector and the Particle Filter Gaussian Process Dynamical Model (PF-GPDM) for multiple targets detection and tracking. The multiple-stage HOG human detector takes advantage from both the HOG feature set and the human motion cues. The detector enables the framework detecting new targets entering the scene as well as providing potential hypotheses for the particle sampling in the PF-GPDM. After processing the detection results, the motion of each new target is calculated and projected to the low dimensional latent space of the GPDM to find the most similar trained motion trajectory. In addition, the particle propagation of existing targets integrates both the motion trajectory prediction in the latent space of GPDM and the hypotheses detected by the HOG human detector. Experimental tests are conducted on the IDIAP data set. The test results demonstrate that the proposed approach can robustly and efficiently detect and track a varying number of targets.

Keywords: GPDM, Particle Filter, Complex Environment, Multiple-Stage HOG Human Detector

1. INTRODUCTION AND RELATED WORK

1.1 Multiple Human Tracking in Complex Visual Environment

Multiple targets tracking in complex visual environment is a very important issue in the surveillance sensor systems, which have attracted considerable attention in recent years. In many real applications, such as surveillance systems in the airport, the camera monitoring area is always crowded with large amount of different people, who may be occluded with each other and have complex motion behaviors. Multiple targets tracking is challenging in such complex scenes. One general approach for multiple target tracking is particle filters, which is able to deal with in deterministic, non-linear, non-Gaussian motions and accessible for multi-modal sensor fusion. However, joint particle filters can normally track up to three or four identical targets due to the exponential complexity. In this work, we present a novel particle filter tracking framework integrated with Gaussian Process Dynamic Model [1] (GPDM) for multiple targets tracking in complex visual environment. Based on the assumption that most human have similar motion trajectory patterns in a particular environment, we apply GPDM to learn a low dimensional motion trajectory representation in the latent space by training the extracted high dimensional human motion data in the observation space. A particle filter tracking framework is then formulated during the tracking process, and the motion pattern of each target is projected to the latent space. By sampling around the latent space, the most similar motion trajectory will be determined and mapped back to the observation space. The back projected motion trajectory will be used for each particle to evaluate the appearance feature in the observation space. In addition, the Markov Dynamic in this latent space will increase the prediction accuracy for each particle.

In contrast to traditional methods of the multiple targets tracking, the major novelty of our PF-GPDM is integrating the particle filter framework with GPDM, which can provide prior trained motion trajectory for the each particle and makes the efficient multiple targets tracking in a complex scene possible. GPDM has the advantage of projecting high-dimensional observation motion data to low dimensional latent space as well as mapping the subspace data back to the observation space. This allows our framework to utilize back projected similar motion information to reduce the sampling ambiguity and improve the particle efficiency. Moreover, the Markov Dynamic in the latent space models the time-series motion smoothly. We extend our previous work [2,3] by performing targets tracking in complex visual environment with more crowded people and targets having different motion behaviors and occlusion. In addition, instead of manually initializing the motion trajectory, our framework can automatically determine the initial motion pattern by employing the normal particle filter tracking in the first five frames for the new targets.

1.2 Multiple Human Detection and Tracking Using Multiple-Stage HOG Detector and PF-GPDM

In order to automatically detect different targets entering or leaving the scene, we proposed a new framework which integrated HOG feature based human detection and the PF-GPDM for tracking multiple targets. By incorporating the HOG human detector with particle filter tracking, the framework takes the advantage both from detection and tracking, which greatly improves the performance of multiple targets tracking.

This framework extends our previous work³ based on the PF-GPDM, in which we used the pre-trained motion trajectories and sampling in the latent space to reduce the particle sampling complexity. This makes it possible for real time multiple targets tracking in the particle filter framework. However, each target's initial position needs to be marked manually before performing tracking.

Our approach to multiple human tracking is to make use of both detection and tracking cues. The main features of our approach are as follows: Firstly, we implement an efficient mechanism to compute the possible targets' positions in the motion frame. Then we combine human motion cues with the HOG feature set for human detection, which greatly improves the detection rate and reduced the computational cost for multiple targets detection in video sequences. Secondly, we integrate the HOG human detector with the PF-GPDM tracking framework. On one hand, the HOG human detector can automatically identify a new person entering the monitoring scene as well as providing potential hypothesis for particle tracking. On the other hand, particle filter tracking provides human motion information for existing targets, which can be used for localizing possible human position for the HOG human detector, thus greatly reduces the computational cost for the HOG human detection.

1.3 Related Work

Gaussian Process Latent Variable model [4] (GPLVM) developed by Neil Lawrence in 2006 provided probabilistic mapping from high-dimensional observation data to low-dimensional latent space, which represented the joint distribution of observation data. Compare to other dimensional algorithms, such as LLE [5], ISOMAP [6], GPLVM has the advantage to project the latent data back to observation space. Wang et al. incorporates Markov dynamics on latent variable state transitions lending Gaussian Process Latent Variable Model to handle time series data and robustly track human body motion and pose changes by classifying poses and motions. Leonid et al. [1] proposed a Gaussian process annealing-particle-filter-based method to perform 3D target tracking by exploring color histogram features [7], while he focused on pose reconstruction rather than human trajectory tracking. A real time body particle tracking framework was introduced by Hou [8] to capture human motion. However, he aimed to track complex motion of one target and used the motion data for the pose estimation.

Human detection has attracted much attention in recent years. Human detection can be treated as a classification problem. Two main steps for solving this problem are feature extraction and learning. However, as human body shapes are non-rigid, and may have various appearances due to different pose and clothing, human detection is still a challenging task in computer vision research. Viola et.al [9] applied the Haar-Like feature for pedestrian detection, in which both the appearance and motion information were used. Jones et al. [10] extended their work by using many more frames as input to the detector and achieved a more detailed analysis of motion. In [11] Dalal and Triggs proposed the histogram of oriented gradients feature for human detection in images. Their experiment results showed that the HOG feature outperform existing feature set in the human detection. Zhu et.al [12] showed that the combination of the cascade of rejecters approach and the HOG features led to a fast and accurate human detection system. They used AdaBoost for selecting the best blocks for human detection. They claimed that their system can nearly achieve real time performance.

Particle filter is a general approach used in multiple targets tracking. Khan et.al proposed a MCMC based particle filter for tracking multiple interacting ants [13], while ant have less shape change comparing to human. K. Smith also provided a particle framework for tracking varying number targets [14]. Multiple objects were formulated by a joint state-space model while efficient sampling is performed by deploying trans-dimensional MCMC on the subspace. It failed to track some targets due to the weakness of color models.

Our jointed detection and tracking framework was directly inspired by the Boosted particle which was introduced by K. Okuma [15]. To track multiple human objects simultaneously, K. Okuma integrated a Boosted classifier with particle framework together, in which Boosted classifier is used to detect different hockey

players and provide hypothesis for each particle. In general, the aforementioned particle frameworks all take the advantage of the relatively simple environment, such as ground, hockey play rink, or fixed background.

2. MULTIPLE HUMAN TRACKING IN COMPLEX VISUAL ENVIRONMENT

2.1 Particle Filter and GPDM

A particle filter is a Monte Carlo method for non-linear, non-Gaussian models, which approximates continuous probability density function by using large number of samples, i.e., discrete distribution approximation. Hence the accuracy of the approximation depends on high dimensional state space which causes exponential increase of the number of particles. Given the time complexity constraint, the reduction of particles, and hence the computation power is a potential solution. Once a GPDM is created, sampling from the dynamical field provides meaningful prediction on the future motion changes. In GPDM, an observation space vector represents a pose configuration and motion trajectory captured by a sequence of poses. The latent space defines the temporal dependence between poses by employing Gaussian Process integrated by Markov Chain on the latent variable transitions. Since motion prediction, the temporal dependence and sampling are performed on the latent space, potential computation benefits may be obtained.

2.2 Gaussian Process Dynamical Model Particle Filter

This research aims at developing a low complexity and highly efficient algorithm for tracking Multiple targets in the complex environment. Since the tracking environment is complex, the prior trajectory information can help to track each target efficiently and robustly. Future more, with the general framework of GPDM, it can be extended to estimate pose and motion changes as proposed by Wang et al. Hence, if a target is suspected of malicious behavior, the system can trade performance off time complexity.

The basic procedure of the proposed Particle Filter Gaussian Process is as follows.

1. Creating GPDM: GPDM is created on the basis of the trajectory training data sets, i.e., coordinate difference values, and the learning model parameters $\Gamma = \{Y^T, X^T, \bar{\alpha}, \bar{\beta}, W\}$, where Y^T is the training observation data set, X^T is the corresponding latent variable sets, $\bar{\alpha}$ and $\bar{\beta}$ are hyperparameters, and W is a scale parameter.
2. Initializing the model parameters and the particle filter: The latent variable set of the training data and parameters $\{X^T, \bar{\alpha}, \bar{\beta}\}$ are obtained by minimizing the negative log-posterior function $-\ln p(X^T, \bar{\alpha}, \bar{\beta}, W | Y^T)$ of the unknown parameters $\{X^T, \bar{\alpha}, \bar{\beta}, W\}$ with scaled conjugate gradient (SCG) on the training datasets. The prior probability is derived on the basis of the created model. In this step, target templates are obtained from the previous frames as reference images for similarity calculation in the later stage.
3. Projecting from the observation space to latent space: After initializing the targets' position, each target will be tracked by the regular particle filter in the first 5 frames. Then test observation motion pattern data is calculated and projected on the latent coordinate system by using probabilistic principal component analysis (PPCA). As a result, the dimensionality of the observed data is reduced.
4. Predicting and Sampling: Particles are generated by using GPDM in the latent space and the test data to infer the likely coordinate change value $(\Delta x_i, \Delta y_i)$.

5. Determining probabilistic mapping from latent space to observation space: The log posterior probability of the coordinate difference values of the test data is maximized to find the best mapping in the training data sets of the observation space. In addition, the most likely coordinate change value ($\Delta x_i, \Delta y_i$) is used for predicting the next motion.

6. Updating the weights: In the next frame, the similarity between the template's corresponding appearance model and the cropped region centered on the particle is calculated to determine the weights w_i , and the most likely location ($\hat{x}_{t+1}, \hat{y}_{t+1}$) of the corresponding target, as well as to decide whether re-sampling is necessary or not.

7. Repeat Step 3 - 6.

2.2.1 Observation Space

The targets of interest are detected and tracked for trajectory analysis. Instead of studying the coordinate values, the differences of the same target in two neighboring frames are calculated as the observed data. The location of the target can be obtained by adding the difference to the previous coordinate values. The 2D coordinate difference values of the head, centroid and feet form a 6 dimensional vector for each object, given by $Y_k = (\Delta(x_1), \Delta(y_1), \Delta(x_2), \Delta(y_2), \Delta(x_3), \Delta(y_3))$, where Y_k is the observation value of the k^{th} target, and $(x_k + \Delta(x_k), y_k + \Delta(y_k))$ is the coordinate value of the corresponding body part. With the 3 sets of coordinate values, the boundary, width and height of an object can be determined. If there are 5 targets, the observation data has 30 dimensions.

2.2.2 Establishing Trajectory Learning Model and Obtaining Appearance Templates

GPDM is deployed to learn the trajectories of moving objects. The probability density function of latent variable X and the observation variable Y are defined by the following equations,

$$P(X_k | \alpha) = \frac{p(x_k)}{\sqrt{(2\pi)^{(N-1)d} |K_x|^d}} \exp\left(-\frac{1}{2} \text{tr}(K_x X_{2:N}^T)\right) \quad (1)$$

where α is the hyperparameter of kernel, $p(x)$ can be assumed to have Gaussian prior, N is the length of latent vector, d is the dimension of latent space, and K_x is the kernel matrix.

$$P(Y_k | X_k) = \frac{|W|^N}{\sqrt{(2\pi)^{ND} |K_Y|^D}} \exp\left(-\frac{1}{2} \text{tr}(K_Y^{-1} Y W^2 Y^T)\right) \quad (2)$$

where k is the k^{th} target, K_Y is the kernel function, and W is the hyperparameter.

In our study, RBF kernel given by the following is employed for the GPDM model,

$$k_Y(x, x') = \exp\left(-\frac{\gamma}{2} \|x - x'\|^2\right) + \beta^{-1} \delta_{x, x'} \quad (3)$$

where x and x' are any latent variables in the latent space, γ controls the width of the kernel, β^{-1} is the variance of the noise.

Given a specific surveillance environment, certain patterns may be observed and worth exploring for future inferences. To initialize the latent coordinate, the d (dimensionality of the latent space) principal directions of the latent coordinates is determined by deploying probabilistic principal component analysis on the mean-subtracted training dataset Y^T , i.e. $Y^T - \text{mean}(Y^T)$. Given Y^T , the learning parameters are estimated by minimizing the negative-log-posterior using scaled conjugate gradient (SCG) [16]. SCG was proposed to optimize the multiple parameters of large training sets by deploying Levenberg-Marquardt approach to avoid line-search per learning iteration, which increases calculation complexity.

Besides position training datasets, the appearance database is created by obtaining the template images of human head, feet and torso from the initial frames.

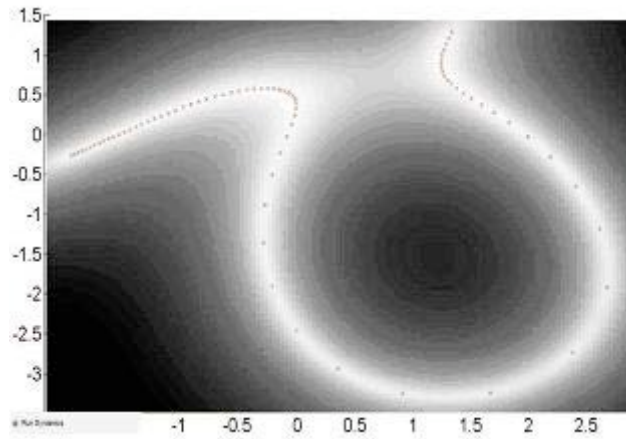


Figure 1 Latent space projections of a 2-target training vector sequence

2.2.3 Latent Space Projecting, Predicting and Particle Sampling

Since GPDM was constructed in the latent space, at the beginning of the test process, the target observation data of first five frames has to be projected to the same 2-dimensional latent space in order to be compared to the trained GPDM. This projection is achieved by using probabilistic principal component analysis (PPCA), same as the first stage in GPDM learning. The feature vector of each frame contains three pairs of coordinate change values for every target being tracked in that frame. For n targets, the feature vector will contain $3 \times n$ pairs of coordinate change values. The PPCA projection will reduce this $3 \times n \times 2$ dimensional feature vectors to a 1×2 latent space vector to be used in particle filtering. The purpose of projecting the test data from the observation space to the latent space is to initialize the testing data in the latent space and obtain a compact representation of the similar motion patterns in the training data set. With PPCA and trained GPDM, we can learn certain common motion patterns (e.g. velocities, directions etc.) from multiple training targets, and then use the learned latent space motion behavior to predict multiple targets' future trajectories using particle filter with much improved efficiency. This is based on the presumption that many human trajectories possess similar properties in common video surveillance applications. It should be noted that the number of targets being tracked does not need to be identical to that in the training data. This is possible because that PPCA aggregates (or projects) multiple training objects as well as test objects onto the same low dimensional space, and therefore the number of objects does not pose a constraint on the tracking process. If we can obtain the

templates and the corresponding initial coordinates of n objects at the beginning of the test phase, the proposed framework can track these n targets regardless the number of training targets.

Particles are generated on the basis of the Gaussian process dynamical model in the latent space, taking the motion model property and unpredictable motion into consideration. The next possible position is predicted by determining the most similar trajectory pattern in the training database and using the corresponding position change value plus noise. The number of particles is reduced from over one hundred to about twenty by deriving the posterior distribution over functions, instead of parameters, and taking advantage of the learned knowledge. The simulation indicates that the decreasing number of particles does not compromise the tracking results, even in temporary occlusion cases. An example of the learned GPDM space is shown in Figure 1. Each point on this 2D latent space is a projection of a feature vector representing two training targets, i.e., 6 pairs of coordinate change values. A total of 72 points in the figure correspond to feature vectors of these two targets over 73 image frames. The grayscale intensity represents the precision of mapping from the observation space to the latent space, and the lighter the pixel appears the higher the precision of mapping is.

2.2.4 Mapping from Latent Space to Observation Space

Thereafter, the latent variables are mapped in a probabilistic way to the location difference data in the observation space, defining the active region (i.e., distribution) of an observed target. However, the exact predicted coordinate values of the motion trajectory in the observation space need be calculated so that the importance weight for each particle in the observation space can be updated. Estimation maximization (EM) approach is employed to determine the most likely observation coordinates in the observation space after the distribution is derived. The non-decreasing log posterior probability of the test data is given by

$$\log(P(Y_k | X^T, \bar{\beta}, W)) = \log \left(\frac{|W|^N}{\sqrt{(2\pi)^{ND} |K_Y|^D}} \exp \left(-\frac{1}{2} \text{tr}(K_Y^{-1} Y W^2 Y^T) \right) \right) \quad (4)$$

where W is the hyperparameter, N is the number of Y sequences, D is the data dimension of Y , K_Y is a kernel matrix defined by a RBF kernel function given by the equation (3). The log posterior probability is maximized to search for the most probable correspondence on the training datasets. The corresponding trajectory pattern is then selected for predicting the following motion. The simulation results show that it returns better prediction results than averaging the previous motion values. In addition, various targets can share the same database to deal with different future situations.

2.2.5 Importance Weights Update

The weights of the particles are updated in terms of the likelihood estimation based on the appearance model. The importance weight equation is given by

$$P(\hat{Y}_t | Z_t, k_t) = \frac{P(Z_t | k_t, \hat{Y}_t) P(k_t, \hat{Y}_t)}{P(Z_t)} \quad (5)$$

$$\omega_t \propto P(Z_t | k_t, \hat{Y}_t) P(k_t, \hat{Y}_t) \quad (6)$$

where $\hat{Y}_t$ is the estimation data, Z_t is the observation data, k_t is the identity of the target, and w_t is the weight of a particle. In our study, the likelihood function $P(Z_t | k_t, \hat{Y}_t)$ is defined to be dependent on the similarity between the appearance model distribution of the template and that of the test object. Therefore, the choice of

appearance model is important for updating the weights of particles. Edge feature is not used in this study due to its ambiguity in term of foreground and background, as well as the computation efficiency consideration. Histogram-Bhattacharya, GMM-KL Appearance Model, and rotation invariant model were tested to determine the resulting performance and time complexity.

2.3 Histogram-Bhattacharyya and GMM-KL Appearance Model

Histogram-Bhattacharyya was used for its simplicity and efficiency. The RGB histogram of the template and the image region under consideration are obtained respectively. The likelihood $P(Z_t | k_t, \hat{Y}_t)$ is defined to be proportional to the similarity between the histogram of the template and the candidate, i.e. the region centered on the considered particle of the same size as the template. The above-mentioned similarity is measured by using Bhattacharyya distance, since it provides complex nonlinear correlations between distributions.

Gaussian Mixture Model-Kullback-Leibler (GMM-KL) approach is also employed to measure the similarity between the image and the test object' template. GMM is a semi-parametric multimodal density model consisting of a number of components to compactly represent pixels of image block in color space with illumination changes. Image can be represented as a set of homogeneous regions modeled by a mixture of Gaussian distributions in color feature space [17]. The Kullback-Leibler distance is a measure of the distance between two probability distributions given the metric of relative entropy [18]. Since the image approximated by Gaussian mixture model can be considered as independently identically distributed (iid) samples following Gaussian mixture distribution, comparison of the template image to that of the test image is formulated as measuring the distance between the two Gaussian mixture distributions.

2.4 Simulation Results and Discussion

The proposed PF-GPDM was implemented by using MATLAB running on a desktop of 2.33GHz Intel Core2 Duo PC with 2GB memory and tested on the PETS 2007 datasets [19] Neil Lawrence's Gaussian Process software provides the related GPDM functions for conducting simulations [20].

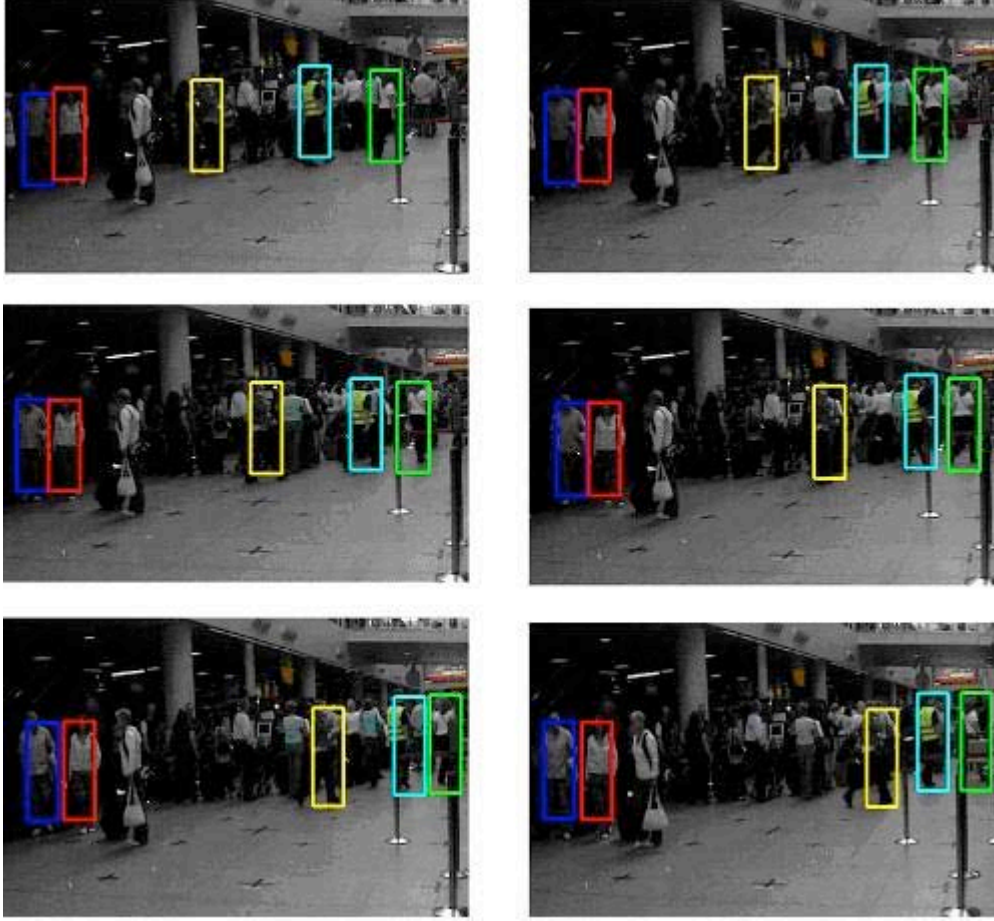


Figure 2 Sampling result of tracking 5 targets using Histogram-Bhattacharya approach

The experiments were designed to evaluate the performance of the proposed PF-GPDM method under complex environment, as well as on targets with occlusions. The performance measures include sample image frames labeled with tracking results, error rate, and runtime. Error rate is defined as the percentage of frames that contain one or more miss-tracked target. Table 1 summarizes the experimental results in terms of % error rate and runtime.

Frames	Targets	Appearance Model	Runtime	%Error Rate
70	5	Histogram GMM	204 sec	6.68%
70	5	Histogram	418 sec	2.84%
80	4		300 sec	3.14%

Table 1 Tracking %Errors on two types of sequences

The training dataset consists of four sequences from the PETS dataset with a total of 276 frames. One target in each sequence is identified and tracked to build up a latent space trajectory database. The selected PETS test dataset includes one sequence of seventy frames with five walking people with different motion behavior and another sequence of eighty frames with four walking people, which contain several temporary occlusion. The background of selected sequences are blurred and crowded with different people and different motion behaviors.

For the first sequence with five targets, we apply our PF-GPDM with two different appearance models, GMM-KL and Histogram-Bhattacharyya. The tracking results are shown in the Figure 2 and Figure 3. Our experiment indicates that the GMM-KL is more discriminative in terms of the blurred background and the targets, compared to the Histogram model. However, the Histogram-Bhattacharyya approach is more efficient in the computation time.

In the second sequence, several temporary occlusions occur among the four targets. The man who walked toward to the right overlapped three times with the women. Figure 4 shows that temporary occlusions in the test sequence are resolved successfully and each target is tracked correctly with different colored bounding boxes. The yellow bounding box in the last frame of Figure 4 indicates a missing target case of our framework. It should be noted that the number of particles is 20 for each target in our PF-GPDM, thus it's more efficient than conventional particle methods.



Figure 3 Sampling result of tracking 5 targets using GMM-KL approach



Figure 4 Sampling result of tracking 4 targets with occlusion

3. MULTIPLE HUMAN DETECTION AND TRACKING BY USING MULTIPLE-STAGE HOG DETECTOR AND PFGPDM

3.1 Multiple-Stage HOG Detector and PFGPDM

In this paper, we combine the Multiple-Stage HOG human detector and the PFGPDM together to improve the robustness for multiple targets tracking. Besides identifying the new targets entering the scene the Multiple-Stage Human detector can help the tracker to detect the overlapped targets under temporary occlusion. In addition, the learned motion information from the PFGPDM tracker can reduce the false alarm of the HOG human detector.

3.2 Appearance Model

Gaussian Mixture Model-Kullback-Leibler (GMM-KL) approach is employed to measure the similarity between the image and the test object' template. GMM is a semi-parametric multimodal density model consisting of a number of components to compactly represent pixels of image block in color space with illumination changes. Image can be represented as a set of homogeneous regions modeled by a mixture of Gaussian distributions in color feature space [17]. The Kullback-Leibler distance is a measure of the distance between two probability distributions given the metric of relative entropy [18]. Since the image approximated by Gaussian Mixture model can be considered as independently identically distributed (iid) samples following Gaussian Mixture distribution, comparison of the template image to that of the test image is formulated as the distance measure between the two Gaussian mixture distributions.

3.3 The Procedure of Proposed Framework

The basic procedure of the proposed HOG human detector and the Particle Filter Gaussian Process Dynamical Model is shown in the Figure 5.

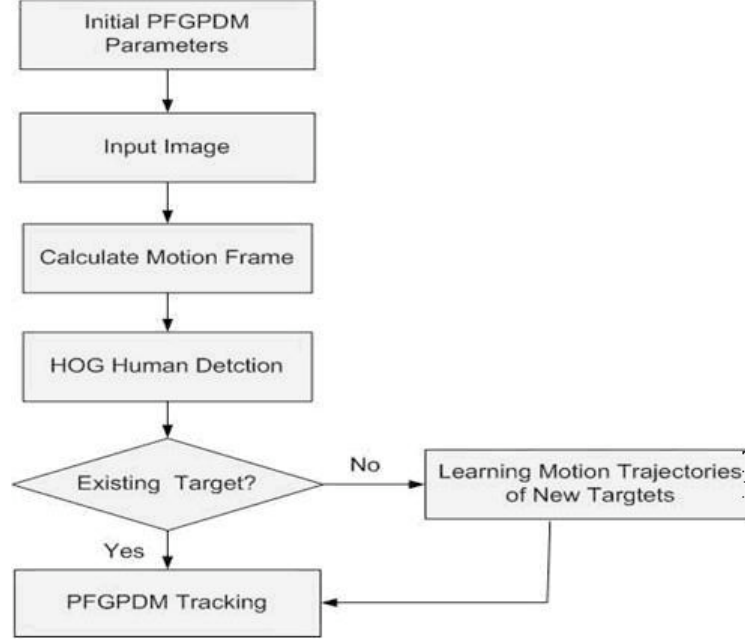


Figure 5 The procedure of detection and tracking framework

1. **Creating GPDM:** GPDM is created on the basis of the trajectory training data sets, i.e., coordinate difference values, and the learning model parameters $\Gamma = \{Y^T, X^T, \bar{\alpha}, \bar{\beta}, W\}$, where Y^T is the training observation data set, X^T is the corresponding latent variable sets, $\bar{\alpha}$ and $\bar{\beta}$ are hyperparameters, and W is a scale parameter.
2. **Initializing the model parameters and the particle filter:** The latent variable set of the training data and parameters $\{X^T, \bar{\alpha}, \bar{\beta}\}$ are obtained by minimizing the negative log-posterior function $-\ln p(X^T, \bar{\alpha}, \bar{\beta}, W | Y^T)$ of the unknown parameters $\{X^T, \bar{\alpha}, \bar{\beta}, W\}$ with scaled conjugate gradient (SCG) on the training datasets. The prior probability is derived on the basis of the created model. In this step, target templates are obtained from the previous frames as reference images for similarity calculation in the later stage.
3. **Multiple-Stage HOG human detection:** For each incoming frame, motion frame is computed first to get the possible targets' region in the current frame. The HOG human detector is applied to determine the human target in the scene. For every 5 frames, the whole region of frame is scanned by HOG human detector in case we miss some stationary targets.

For the new targets:

Extract the observation motion data: If a new target is detected, the target will be tracked by the regular particle filter in the first 5 consecutive frames. The target's position in the first five frames then is stored as the observation motion pattern data.

Projecting from the observation space to the latent space: The observation motion pattern data is calculated and projected on the latent coordinate system by using probabilistic principal component analysis (PPCA). As a result, the dimensionality of the observed data is reduced.

Predicting and Sampling: Particles are generated by using GPDM in the latent space and the test data to infer the likely coordinate change value ($\Delta x_i, \Delta y_i$).

Determining probabilistic mapping from latent space to observation space: The log posterior probability of the coordinate difference values of the test data is maximized to find the best mapping in the training data sets of the observation space.

In addition, the most likely coordinate change value ($\Delta x_i, \Delta y_i$) is used for predicting the next motion.

For the existed targets:

Compute the prediction by using sampled trajectory If the detected target has already tracked by the particle filter framework, then we will use it for updating the particle weights

4. Updating the weights: In the next frame, the similarity between the template's corresponding appearance model and the cropped region centered on the particle is calculated. To determine the weights w_i , and the most likely location $(\hat{x}_{t+1}, \hat{y}_{t+1})$ of the corresponding target, both the HOG human detection result and the particle prediction are considered. We used a weighted and linear combination of both from the similarity function and detection result. .

5. Repeat Step 3 - 5.

3.3.1 Multiple-Stage HOG Human Detection

In this step, the motion cues in the motion frame are calculated. Then the HOG human detector focuses on these possible regions. In case of missing new targets, the detector will scan the whole image at every five frames. The HOG human detection is very essential in our framework. After determining the location of a target, we first check whether it contains the region which includes targets in the previous frame. If not, it will be recognized as a new target entering the scene, otherwise we treat it as an existing target and use the tracking motion trajectory to refine the detection results.

3.3.2 Observation Space

The new targets of interest are detected and tracked for trajectory analysis. Instead of studying the coordinate values, the differences of the same target in two neighboring frames are calculated as the observed data. The location of the target can be obtained by adding the difference to the previous coordinate values. The 2D coordinate difference values of the head, centroid and feet form a 6 dimensional vector for each object, given by $Y_k = (\Delta(x_1), \Delta(y_1), \Delta(x_2), \Delta(y_2), \Delta(x_3), \Delta(y_3))$, where Y_k is the observation value of the k th target, and $(x_k + \Delta(x_k), y_k + \Delta(y_k))$ is the coordinate value of the corresponding body part. With the 3 sets of coordinate values, the boundary, width and height of an object can be determined. If there are 5 targets, the observation data has 30 dimensions.

3.3.3 PFGPDM Initialization and Projection

GPDM is deployed to learn the trajectories of moving objects. The probability density function of the latent variable X and the observation variable Y are defined by the equation (1).

In our study, RBF kernel given by the following is employed for the GPDM model, which is defined in the equation (2), where x and x' are any latent variables in the latent space, γ controls the width of the kernel, $\beta-1$ is the variance of the noise.

3.3.4 Importance Weights Update

The weights of the particles are updated in terms of the likelihood estimation based on the appearance model. The importance weight equation is given by

$$P(\hat{Y}_t | Z_t, k_t) = \alpha \frac{P(Z_t | k_t, \hat{Y}_t)P(k_t, \hat{Y}_t)}{P(Z_t)} + (1-\alpha)P_{HOG} \quad (7)$$

$$\omega_t \propto P(Z_t | k_t, \hat{Y}_t)P(k_t, \hat{Y}_t) \quad (8)$$

where $\hat{Y}_t$ is the estimation data, Z_t is the observation data, k_t is the identity of the target, and w_t is the weight of a particle, and P_{HOG} is the likelihood of the HOG detection for the existing object, which represents the confidence of human detection. The parameter α can be set dynamically depending on the tracking situations, including collisions or occlusions. When $\alpha = 1$, our framework regresses to the PF-GPDM. We can place more attention to the HOG human detection by increasing α . Such as in the occlusion situation, large value of α can find overlapped targets and improve the tracking performance.

In our study, the likelihood function $P(Z_t | k_t, \hat{Y}_t)$ is defined to be dependent on the similarity between the appearance model distribution of the template and that of the detected object as well as the posterior probability of the HOG detection. Therefore, the choice of appearance model is important for updating the weights of particles. In this paper, a GMM-KL Appearance Model is tested to determine the resulting performance and time complexity.

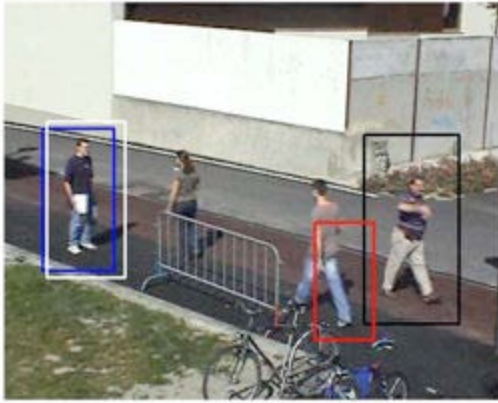
3.4 Experimental Results and Discussion

The proposed framework was implemented by using MATLAB running on a desktop of 2.33GHz Intel Core2 Duo PC with 2GB memory. The human motion trajectory of the PFGPDM is trained on the PETS [19] and Neil Lawrence's Gaussian process software provides related GPDM functions for conducting simulations [20].

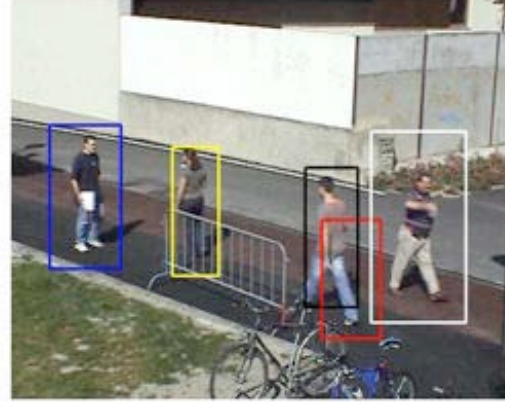
3.4.1 HOG Human Detection Results

To train the HOG human detector, we used the INRIA data set, which contains various human in different scenarios. By following the instruction in the [11], we get the trained human detector. However, when we directly apply the HOG detector on the selected sequences of IDIAP data set, the detection rate is relatively lower (about 76%). Figure 6(a) shows one of the false detection results, in which the second left target was not detected successfully. However, for each new target entering the scenario, the HOG human detector can identify it correctly. Figure 6(b) shows the detection results of Multiple-Stage Human detector. Although there still exists some overlapping detection windows of the same targets, all the pedestrians have been identified in

the whole image. Comparing to original HOG human detector, the detection rate increases to 93% in the IDIAP data set.



(a) HOG human detector



(b) Multiple-Stage HOG human detector

Figure 6 Detection results of four people

3.4.2 Jointed Detection and Tracking Results

The first sequence contains 132 frames. Figure 7 shows our framework detects and tracks two targets correctly. Although the camera is fixed, the background is changing as the car entering the scene. If we only use the motion information for detection, the human target will not be detected correctly. Frame 1 and Frame 67 shows that our system can detect each new human target correctly. The other frames show that our framework can track each target correctly with temporal occlusion.



Figure 7 Sampling results of tracking 2 targets (Frame 1, 40, 67, 95, 106, 132)

The second sequence contains 112 frames. Figure 8 shows that our framework detects and tracks two to four targets successfully with occlusion and various targets entering and leaving the scene. Although more people enter and leave in the sequence, the HOG human detector identifies each new target correctly. During the occlusion situation, the human detector also helps the tracker to identify the correct targets. Comparing with the PFGPDM tracking [3] in the second sequence, our current framework reduces the tracking error rate from 6.68% to 2.84%. In addition, the HOG human detector improves the robustness of PFGPDM tracking, which means our framework can reliably detect and track more frames than the PFGPDM.

4. CONCLUSION AND FUTURE WORK

4.1 Conclusion

During this year, we firstly extended the PFGPDM to track multiple targets in complex visual environment; the test results demonstrate that the approach can track more than four targets with reasonable run-time. After that, we introduced a framework which integrated the Multiple-Stage HOG human detector with the PFGPDM

tracker. The Multiple-Stage HOG human detector improves the detection rate when applying on the IDIAP data set, while the PFDPGM can provide additional motion information for human detection after initializing the target's position. Therefore, by combining the tracking and detection together, we can get a good performance for multiple targets tracking.

4.2 Future Work

In our current PFGPDM, GPDM is not updated online after being created; we just search the similar motion trajectory in the GPDM according the trained data. One possible future work is to adaptively add the new trajectory pattern of test data to the trained GPDM during the multiple targets tracking. Whenever accounted new motion trajectory of a target, it will add to our motion trajectory data set. This will enrich our motion trajectory pattern and improve the robustness of our framework.

In the future, we also want to add adaptive learning for the HOG human detection part, which will online update the human classifier and improve the detection rate online. In addition, we want to improve our tracking framework to achieve real time performance.

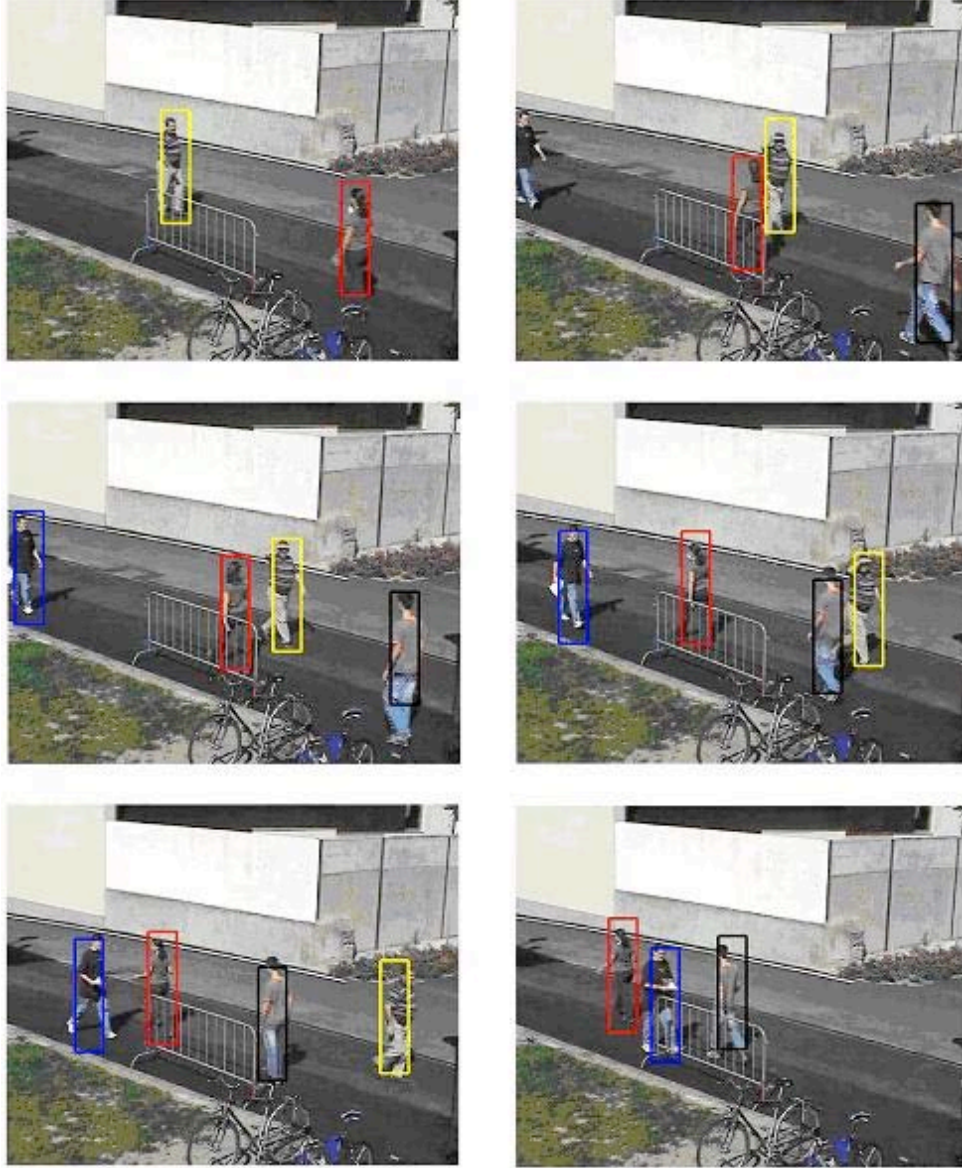


Figure 8 Sampling result of tracking 4 targets (frame 1,42,48,73,88,112)

REFERENCES

- [1] Wang, J., Fleet, D., and Hertzmann, A., “Gaussian process dynamical models,” in [Advances in Neural Information Processing Systems 18], Weiss, Y., Scholkopf, B., and Platt, J., eds., 1441–1448, MIT Press, Cambridge, MA (2006).
- [2] Wang, J., Man, H., and Yin, Y., “Multitarget tracking using gaussian process dynamical model particle filter,” in [ICIP08], 1580–1583 (2008).
- [3] Wang, J., Yin, Y., and Man, H., “Multiple human tracking using particle filter with gaussian process dynamical model,” JIVP 2008 (2008).
- [4] Lawrence, N., “Probabilistic non-linear principal component analysis with gaussian process latent variable models,” J. Mach. Learn. Res. 6, 1783–1816 (2005).
- [5] Roweis, S. and Saul, L., “Nonlinear dimensionality reduction by locally linear embedding,” Science 290, 2323–2326 (December 2000).

- [6] Tenenbaum, J. B., “Mapping a manifold of perceptual observations,” in [Advances in Neural Information Processing Systems] 10 , 682–688, MIT Press (1998).
- [7] Raskin, L., Rivlin, E., and Rudzsky, M., “Using gaussian process annealing particle filter for 3d human tracking,” EURASIP Journal on Advances in Signal Processing (2007).
- [8] Hou, S., Galata, A., Caillette, F., Thacker, N., and Bromiley, P., “Real-time body tracking using a gaussian process latent variable model,” 1–8 (2007).
- [9] Viola, P., Jones, M., and Snow, D., “Detecting pedestrians using patterns of motion and appearance,” IJCV 63, 153–161 (July 2005).
- [10] Jones, M. and Snow, D., “Pedestrian detection using boosted features over many frames,” in [ICPR08], 1–4 (2008).
- [11] Dalal, N. and Triggs, B., “Histograms of oriented gradients for human detection,” in [In CVPR], 886–893 (2005).
- [12] Zhu, Q., Yeh, M., Cheng, K., and Avidan, S., “Fast human detection using a cascade of histograms of oriented gradients,” in [CVPR06], II: 1491–1498 (2006).
- [13] Khan, Z., Balch, T., and Dellaert, F., “An mcmc-based particle filter for tracking multiple interacting targets,” in [Proc. ECCV, pages 279–290,2004.], (2004).
- [14] Smith, K., Gatica Perez, D., and Odobez, J., “Using particles to track varying numbers of interacting people,” in [CVPR '05], I: 962–969 (2005).
- [15] Okuma, K., Taleghani, A., de Freitas, N., Little, J., and Lowe, D., “A boosted particle filter: Multitarget detection and tracking,” in [Proc. ECCV, pages 28–39,2004.], (2004).
- [16] Riedmiller, M. and Braun, H., “Rprop- a fast adaptive learning algorithm,” in [Proc. of 7th Int’l Symp. on Computer and Information Sciences (ISCIS VII)], (1992).
- [17] Greenspan, H., Goldberger, J., and Ridel, L., “A continuous probabilistic framework for image matching,” Compute. Vis. Image Underst. 84(3), 384–406 (2001).
- [18] Kullback, S., [Learning Textures], Dover (1968).
- [19] Data, P. B., “Pets in conjunction with 11th IEEE international conference on computer vision.” <http://www.cvg.rdg.ac.uk/PETS2007/data.html>.
- [20] “Neil Lawrence Gaussian process software.” <http://www.cs.man.ac.uk/neill/software.htm>

FINAL PROJECT REPORT

Submitted to

U.S. Army
The Armament Research, Development and Engineering Center (ARDEC)
Picatinny FY08-FY09 Program

Task 3: Information assurance in sensor networks

Haibo He and Hong Man

Department of Electrical and Computer Engineering
Stevens Institute of Technology
Hoboken, NJ 07030

Graduate students:
Sheng Chen and Yuan Cao

Contents

Contents	2
1. Introduction.....	3
2. Approach taken	11
2.1 RAMOBoost – A Ranked minority over-sampling in Boosting approach (RAMOBoost) for learning from imbalanced data set	11
2.2 ADAIN – An adaptive incremental learning framework (ADAIN) for learning from the data flow.....	15
2.3 SERA – A Selectively Recursive Approach towards nonstationary imbalanced data classification	20
2.4 Network Intrusion Detection Based on KDE and SOM	25
2.4.1 Definition of Outliers	25
2.4.2 Kernel Density Estimation (KDE).....	26
2.4.3 Self-organizing Map (SOM)	28
2.4.4 Network Intrusion detection Algorithm.....	29
3. Results.....	31
3.1 RAMOBoost – A Ranked minority over-sampling in Boosting approach (RAMOBoost) for learning from imbalanced data set	31
3.2 ADAIN – An adaptive incremental learning framework (ADAIN) for learning from the data flow.....	41
3.3 SERA – A Selectively Recursive Approach towards nonstationary imbalanced data classification	44
3.4 Detection of Network Intrusions.....	50
3.4.1 System Model And Dataset	50
3.4.2 Simulation Results	51
4. FPGA-based Reconfigurable Platform for Video and Image Processing.....	53
4.1 Four-direction Edge Detection.....	55
4.2 Scaling Functionalities: Image Zoom-in and Zoom-out.....	57
4.3 System Implementation and Experimental results.....	58
4.3.1 System Development	58
4.3.2 Simulation and Experimental Results.....	62
5. Potential Applications	63
Reference	65

1. Introduction

During this project period, we have advanced the theoretical understanding of the fundamental challenges of the data mining in sensor networks, and developed numerous algorithms and models across different application domains. Specifically, our major efforts are focused on how to transform large volumes of raw data into information and knowledge representation, and how to adaptively learn from such data to support the decision-making processes within realistic environments, as well as the network security. The following specific problems have been investigated in this project: imbalanced learning problems, incremental learning problems, incremental learning for imbalanced data stream problems, network intrusion detection, and an FPGA-based, general-purpose, multi-task, and reconfigurable platform for video and image processing. In this section, we will provide a brief review of the state-of-the-art research related to these topics.

Learning from imbalanced data (imbalanced learning) has become a critical and significant research issue in many of today's data intensive applications, such as financial engineering, anomaly detection, biomedical data analysis, and many others [1]. The amount and complexity of raw data that is captured to monitor, analyze, and support decision-making processes continues to grow at an incredible rate. Consequently, this enriches the opportunities for computationally intelligent methods to play an essential role in applications involving large amounts of data. On the other hand, these opportunities also raise many new challenges for the research community in general. In regards to imbalanced learning, the importance and complexity of this problem is reflected in the recent installment of dedicated special issue symposiums and conference workshops, such as the Association for the Advancement of Artificial Intelligence workshop on Learning from Imbalanced Datasets (AAAI'00) [2], the International Conference of Machine Learning workshop on Learning from Imbalanced Datasets (ICML'03) [3], and the Association for Computing Machinery Special Interest Group on Knowledge Discovery and Data Mining explorations (ACM SIGKDD Explorations'04) [4].

Generally speaking, any dataset that exhibits an unequal distribution between its classes can be considered imbalanced. In real-world applications, datasets exhibiting severe imbalances are of great interest since they generally present significant difficulties for learning mechanisms. Typical imbalance ratios can range from \$1:100\$ in fraud detection problems [5] to \$1:100000\$ in high-energy physics event classification [6]. However imbalances of this form are just one aspect of the imbalanced learning problem. The imbalance learning problem generally materializes in two forms: *relative imbalances* and *absolute imbalances* [7]. *Absolute imbalances* arise in datasets where minority examples are definitively scarce and underrepresented, whereas *relative imbalances* are indicative of datasets in which minority examples are well represented but remain severely outnumbered by majority class examples. Some studies have shown that the degradation of classification performance attributed to imbalanced data is not necessarily the result of relative imbalances but rather by the lack of representative examples (absolute imbalances) [8] [9] [10] [11]. In particular, for a given dataset that contains several sub-concepts, the distribution of minority examples over the minority class concepts may yield clusters with insufficient representative examples to form a classification rule. This problem of concept data representation within a class is also known as the *within-class imbalance problem* [9]

[12] [13] and was verified to be more difficult to handle than datasets with only homogeneous concepts for each class [9] [11].

As follows, we formulate the four main categories of the solutions for imbalanced learning problems, and their most popular derived algorithms.

(1) Sampling methods:

Building on the foundations of the random (simple) over-sampling and under-sampling techniques, researchers have developed advanced sampling methods to address the shortcomings of these basic techniques, such as overfitting and information loss.

One of the more popular forms of advanced sampling, the synthetic sampling methods, creates synthetic instances to compensate for skewed distributions. For instance, the SMOTE algorithm [14] searches for the nearest neighbors of every minority instance and generates synthetic minority data by calculating linear interpolations between an original minority class instance and a randomly selected neighbor. Expanding on the SMOTE framework, the Borderline-SMOTE algorithm [15] locates those minority class examples that reside along the borders between majority and minority classes; this sample set is then used to generate synthetic instances similar to the interpolation methods in SMOTE. In [16] the DataBoost-IM method is proposed which over-samples both minority and majority class examples by synthetically generating instances based on “seed examples”, which are generally selected from difficult-to-learn examples. In another example, the JOUS-Boost approach [17], fuses the AdaBoost algorithm with over/under-sampling by introducing data perturbations (jittering) at every ensemble iteration in order to break the “ties” produced by duplicated samples created from simple over sampling.

Another form of advanced sampling deals with minimizing the overlap that can arise between classes due to noise or simple over-sampling. Most of these methods use a variant of Tomek links [18] to identify overlapping instances. A Tomek link is a pair of minimally distanced nearest neighbor examples of opposite classes. Specifically, if two instances form a Tomek link then they either both reside on the borderline of the two classes or one of them is attributed to noise; therefore, by removing Tomek links noise can be suppressed. Similar under-sampling techniques include the edited nearest neighbor (ENN) method, which removes examples that differ from two of their nearest neighbors. Additionally, algorithms such as SMOTE+Tomek and SMOTE+ENN [8] integrate these under-sampling techniques with SMOTE to address overlapping issues produced by SMOTE and improve classification performance.

In general, the above-mentioned sampling algorithms solely focus on relative imbalances; i.e. they do not explicitly confront within-class imbalances. However, as mentioned earlier, within-class imbalances have a greater effect on classification performance compared to relative imbalances. In an effort to explicitly address the within-class imbalance problem, the Cluster Based Over-sampling (CBO) algorithm was proposed in [10]. The CBO algorithm over-samples both minority and majority class examples by over-sampling (inflating) all majority class clusters other than the largest to be the same size as the largest, and inflating all minority class clusters to have an equal number of instances given by a proportionality factor. To achieve this, the training data of both the minority and majority classes must first be clustered (by the k -means algorithm,

for instance), and then simple oversampling is performed on a cluster-by-cluster basis. In another example of cluster-based sampling, [19] proposed the use of Support Cluster Machines to learn sub-concepts while bypassing any inherent within-class over-lapping by mapping the feature space to higher dimensions. Each of these sampling algorithms introduces different considerations for tackling the imbalanced learning problem.

(2) Cost-sensitive learning methods

Cost-sensitive learning methods typically employ the use of cost-matrices to estimate the costs of different classification errors. In particular, cost-sensitive learning methods facilitate imbalanced learning by assigning different weights to different instances according to their misclassification cost. These techniques have shown great success when applied to imbalanced learning problems. For example, in [20] an instance-weighting method was presented to induce cost-sensitive trees. This method generalizes the standard tree induction process by having only the initial instance weights determine the type of trees to be induced - minimum error trees or minimum high-cost error trees. In [21] [22], the Asymmetric AdaBoost method is proposed to handle the face detection problems for which the skewed class ratio can be quite high. The idea of Asymmetric AdaBoost is straightforward: the sampling distribution over the training data is modified at the beginning of each loop, i.e., the weights of positive examples could be increased. Then hopefully, the number of false negatives in the minimum error criteria could be minimized. The AdaCost proposed in [23] combines cost-sensitive learning with boosting. By referring to the cost-sensitive matrix, AdaCost assigns different cost values to misclassified minority and majority examples by the trained hypothesis at each iteration loop. In this way, the decision boundary will be forced to move toward the minority examples. Additional examples of cost-sensitive learning include the MetaCost method proposed in [24] that can make any arbitrary classifier cost-sensitive by wrapping a cost-minimizing procedure according to specific requirements, the cost-sensitive neural networks proposed in [25] that produce learning algorithms with powerful applicative abilities, and the various cost-sensitive techniques fused with support vector machines (SVMs) proposed in [26].

Cost-sensitive learning is a popular solution for imbalanced learning problems, and is at times the best alternative for particular domains. For example, in [26] a cost sensitive SVM was used to counter the skewed distributions inherent in face recognition applications. The discussions presented in this work highlight a critical shortcoming of sampling methods, namely the \emph{preservation of data orientation}. Due to the special orientation of facial features, random manipulation of data or random data generation cannot provide useful information for face recognition. As a result, data sampling techniques were not considered in that work - illustrating the need for a diverse selection of methods to handle imbalanced learning problems across different application domains.

(3) Kernel-based learning methods

Kernel-based methods have recently become very popular across various fields including imbalanced learning. In general, kernel-based methods facilitate learning by maximizing the separation margin between concepts in linearly separable feature spaces. More specifically, kernel-based methods use kernel-mapping functions to map low dimension feature spaces to higher dimension spaces where linear separation is achievable. For

instance, in [27] [28] the Kernel Based Alignment (KBA) algorithm was proposed in which the imbalanced information of the data set is used as information prior to adjusting the kernel matrix in order to facilitate SVM learning for improved prediction accuracy. Another example of kernel-based learning presents a kernel classifier construction algorithm using orthogonal forward selection (OFS) to optimize the generalization model for two-class imbalanced learning problems [29]. This is accomplished by using the regularized orthogonal weighted least squares (ROWLS) method and a model selection criterion of maximal leave-one-out area under curve (LOO-AUC) of the ROC graph.

(4) Active learning methods

Active learning methods were originally developed for learning from data sets with unlabeled instances. Recently, active learning methods have found increasingly used in imbalanced learning applications as well. For example, a SVM based active learning approach for imbalanced data sets was proposed in [30] [31]. This algorithm locates a “most informative” sample by evaluating a small, fixed number of randomly selected examples instead of the entire data set [31]. In [32], the stopping condition for active learning applications in word sense disambiguation (WSD) domains was investigated. To alleviate the complications introduced by within-class imbalances, this work proposed a bootstrap-based oversampling technique (BootOS) to improve active learning performance for imbalanced WSD applications.

Solutions that target both relative and absolute imbalances should logically be more adept to handling a wide spectrum of imbalanced learning problems. To this end, we propose RAMOBoost, a ranked minority over-sampling technique embedded with a boosting procedure to facilitate learning from imbalanced datasets. Based on an integration of over-sampling and ensemble learning technique, RAMOBoost systematically generates synthetic instances by considering the class ratios of surrounding nearest neighbors of each minority class example in the underlying training data distribution. Unlike many existing approaches that use uniform sampling distributions, RAMOBoost adaptively adjusts the sampling weights of minority class examples according to their data distributions. Moreover, by integrating the ensemble learning methodology, RAMOBoost adopts an iterative learning procedure which assesses the hypothesis developed at each boosting iteration to adaptively shift the decision boundary to focus more attention on those difficult-to-learn instances of the both majority and minority classes.

Incremental learning has also attracted growing attention from both academia and industry. Numerous new algorithms and architectures have been developed and successfully applied to different domains. For instance, an incremental linear discriminant analysis (ILDA) was proposed in [33] to handle the inverse of the within-class scatter matrix issue. Based on ILDA, a new algorithm named GSVD-ILDA, the generalized singular value decomposition LDA, was proposed and successfully applied to the face recognition problem. In [34] [35], incremental learning for autonomous navigation systems was presented. Various experiments with mobile robots and a vision-based autonomous land vehicle (ALV) in the indoor learning environment were used to demonstrate the effectiveness of such learning methods. A study of incremental learning for machine intelligence research [36] described various learning and memory architectures that achieve high-level intelligence. In [37], a system named SwiftFile was proposed to help different users organize their e-mail messages into folders, which can be dynamically

adjusted according to users' mailing habits. Some other works on incremental learning and its application include the incremental learning fuzzy neural (ILFN) network for fault detection and classification in a machinery condition or health monitoring environment [38], incremental learning for multi-sensor data fusion [39], incremental genetic learning for data classification [40], incremental semi-supervised learning [41], incremental learning for human-robot interaction [42], and others.

There is controversy existing in the community regarding the definition of incremental learning. For instance, in [43] [44], whether the previous data can be accessed by the current learning process in the scenario of incremental learning was debated. Besides, in [45], whether the incremental learning should be motivated to handle the unexpected emergent new class, i.e., concept shifting issue, was discussed. Recently, it was proposed in [46] that the incremental learning should be capable of learning the new information, and retaining the previously acquired knowledge, while without having access to the previously seen data. Along with this direction, the incremental learning framework discussed in this article mainly focus on two important questions: how to adaptively pass the previously learned knowledge to the current received data to benefit learning from the new raw data, and how to accumulate experience and knowledge over time to support future decision-making processes. We consider these two characteristics are the most critical aspects to understand the foundation of the adaptive incremental learning, therefore facilitating the development of principled methodologies across different domains to benefit the computational intelligence community towards the long-term goal of machine intelligence research [36].

Considering the following learning scenario: Let D_{j-1} represent the data chunk received between time t_{j-1} and t_j , and h_{j-1} be a hypothesis developed on D_{j-1} . The important question is how should the system adaptively learn information when a new chunk of data, D_j , is received? Conventionally, there are two categories of methods used to answer this question.

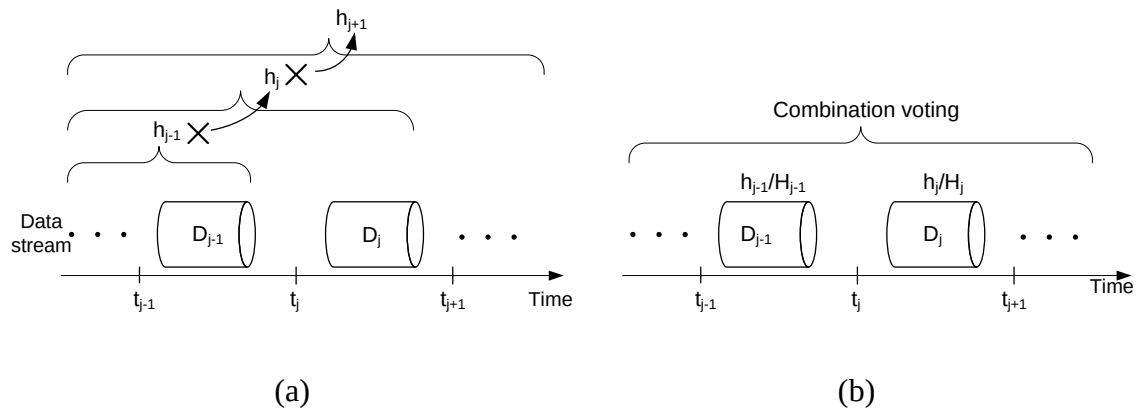


Fig. 1 Traditional approaches of learning from data flow

The first group of methods employs a simple data accumulation approach, as illustrated in Fig. 1(a). In these methods, one simply discards h_{j-1} (denoted by the cross sign) and develops a new hypothesis h_j based on all the available datasets accumulated so far $\{\dots, D_{j-1}, D_j\}$. This is a very intuitive approach. The major disadvantage of this approach is that it loses all previous knowledge, therefore suffering “catastrophic forgetting” [47]. In addition, the requirement for storage of all accumulated data sets may not be feasible in many real-world applications due to limited memory and computational resources.

The second approach employs ensemble learning methodology as illustrated in Fig. 1(b). Briefly speaking, whenever a new chunk of data is available, either a single new hypothesis, h_j , or a set of new hypotheses $H : h_j, j = 1, \dots, L$, are developed based on the new data. Finally, a voting mechanism can be used to combine all the decisions from different hypotheses to reach the final prediction. The major advantage of this approach is that we do not require storage or access to the previously observed data. Instead, the knowledge has been stored in a series of hypotheses developed along the learning life. For example, the *Learn⁺⁺* method is based on this idea and adopts an “ensemble-of-ensembles” learning paradigm [48].

Although the idea as illustrated in Fig. 1(b) has been successfully applied to many application domains, it also has its own limitations. As each chunk of the data flow is considered separately during the learning stage, there is no experience accumulation and knowledge transformation from old data to the new data. The knowledge learned in time period of $[t_{j-1}, t_j]$, i.e., the hypothesis h_{j-1} , cannot be used to benefit the learning process in $[t_j, t_{j+1}]$ though both hypothesis will participate the combination voting process. The only knowledge integration process is in the final voting stage. Therefore, an essential problem of incremental learning, that is to say, the accumulation of experience over time and its usage in facilitating future learning process, is poorly addressed. This work aims to address this issue.

Motivated by the successful application of IMORL for video and image data learning [49], we propose the ADAIN methodology to be a general incremental learning framework, which can be adapted and adjusted by different motivations and domain-knowledge. For instance, different base learners can be embedded into the ADAIN framework according to different application requirements, which provides the flexibility of ADAIN to be a general incremental learning framework across a wide range of domains. The design of the mapping function in ADAIN can also be accomplished through different means such as nonlinear mapping functions instead of the fixed Euclidean distance function as used in IMORL algorithm. Furthermore, in lieu of specifying on one sort of specific application, such as video data analysis in [49], in this project period, we generalize the proposed ADAIN framework to different application domains to demonstrate the effectiveness of this method.

Given the rapid development and successful application of imbalanced learning algorithms and incremental learning algorithms, the problem of how to incremental

learning from imbalanced data stream with concept drifts is shockingly ignored and attract relatively less attention from the community.

Generally speaking, concept drifts occur when the target concepts of the datasets change over time. In such scenarios, given the fact that the data chunks with different timestamp -- the time record by which the data chunk is coming -- render varying target concepts, one naive algorithm is to simply discard all previous training examples and build a learning model based on only the current data chunk. Such method ignores the fact that there always exist some previous data chunks whose target concepts however drift not so far away from the current one, and including the knowledge extracted from them into the learning process may potentially improve the learning performance on the current data chunk significantly. Therefore the fundamental problem in learning concept-drifting data streams is how to identify in a time manner those data are no longer consistent with the current concepts [50]. One way to handle such problem is the sliding-windows approach [51] [52], which maintains a window with either fixed or adaptively determined length to decide how many previous hypotheses should be retained so as to reinforce the prediction performance of the current instances. This approach somehow confronts the *stability-plasticity* dilemma [53], since it is quite difficult to strike a balance between maintaining relevant information and accommodating new knowledge. One compromise for the sliding window is that while all the learning models built previously are retained, their weights to the learning process of the current data chunk can be manipulated differentially. Then hopefully the utmost integrity of the target concept is maintained while the previous feasible information is incorporated as much as possible. Dynamic weights (DW) updating method [50] [54] [55] generally takes this way to handle the concept-drifting data streams, which can be viewed as a methodology of adaptively updating the weights of previous generated hypothesis towards learning the data chunk under consideration by evaluating each of them on the current data chunk under consideration. Despite its success in many literature reports, one critical flaw for DW is that if the learning rules concluded from the most data chunks by normal approaches cannot fully represent the target concept within it, e.g. learning from the imbalanced dataset, then the weighted combination of all built hypotheses cannot significantly improve the learning performance on the current data chunk under consideration.

On the imbalanced learning wise, the existing algorithms are almost all designed for the static imbalanced dataset. Given the intrinsic deficiency of the imbalanced dataset, these algorithms can only mitigate rather than overcome the impact of the with-class imbalance on the learning process. Then the question arises in the scenarios of stream data mining: how should one efficiently make use of the knowledge of previous data chunks to facilitate the learning from the current imbalanced data chunk?

To this end, we propose the SERA framework to address the nonstationary imbalanced stream data mining problem which can be explicitly formalized as learning from data chunks of imbalanced class ratio, which are becoming available in an incremental manner. SERA selectively absorbs the minority examples from the previous data chunks into the current data chunk to improve the learning performance on minority examples. We argue and empirically prove that the minority examples whose target concept deviates from the current target concept are still much better than the synthetic instances for the learning process. We also formulate biased bagging approach (BBagging) to boost the single

learner's performance on the imbalanced datasets, which motivates the learner to be more focused on the minority examples.

Another important topic we have investigated in this project is the network security issue. With the rapid development of network technology, information security has become a major concern for the cyber system research. For instance, for a business firm, sensitive and personally identifiable information in the network, such as financial transactions, employee records, and passwords, is potentially accessible to millions of Internet users, and becomes susceptible to security attacks, such as unauthorized disclosure, modification, misuse, destruction, and others. In the modern net-centric and tactical warfare networks, the situation is much more critical to provide real-time protection for the availability, confidentiality, and integrity of the networked information.

Commonly used security measures, such as cryptographic systems, anti-malware software, and firewalls can provide effective protection for the networked computers. They, however, have difficulties to monitor the network traffic where majority of attacks take place. In order to monitor the network traffic and capture the attacks, intrusion detection systems (IDSs) become indispensable components in any network security systems. Based on the source of the input data, an intrusion detection system can be classified as host-based IDS, such as Haystack [78] and MIDAS [79], multi host-based IDS, such as NIDES [80] and CSM [81], network-based IDS, such as NSM [82] and SNORT [83], and hybrid/hierarchical IDS, such as EMERALD [84]. A host-based IDS is installed on a host computer and monitor only the activity of that particular host; A multi host-based IDS involves a set of hierarchical host-based IDSs running on multiple hosts and coordinating to detect potential intrusions; A network-based IDS is installed on a host computer and monitors the network activities of a particular host or a network of hosts; A hybrid/hierarchical IDS monitors the host, as well as the network activities and have the advantages of the host-based IDSs and the network-based IDSs. Based on the approach used for intrusion detection, an IDS can be classified as misuse-based IDS and anomaly-based IDS. In the misuse-based detection, also known as signature-based detection, the IDS detects a specific attack that has already been documented. In other words, the IDS maintains a database of the network activity patterns of well-known intrusions. The IDS continuously compares the observed network activity pattern with those stored in the database. Once a match is found, the IDS reports an intrusion alert. Although this approach hardly misses any well-known or stored attacks, it often fails to capture the unknown or novel attacks. Due to the growing number of new attacks, the misuse-based IDS shows its own limitations, because it is very difficult to update the database of the attack patterns instantaneously and continuously. An anomaly-based IDS, on the other hand, detects the critical network activity parameters and defines profiles of normal genuine traffic. When an observed network activity is sufficiently derived from the normal state defined by the system, an alert report will be triggered. Therefore, how to detect the outliers/anomalies from a group of observations becomes a critical issue in the design of the anomaly-based IDS.

Generally, outlier detection methods can be categorized into five groups: statistical approaches [85], distance based approaches [86], profiling approaches [87], model-based approaches [88], and rule based approaches [89].

Finally we present a general-purpose, multi-task, and reconfigurable platform for video and image processing. With the increasing requirements of processing power in many of today's video and image processing applications, it is important to go beyond the software implementation to provide a real-time, low cost, high performance, and scalable hardware platform. In this paper, we propose a system by using the powerful parallel processing architecture in the Field Programmable Gate Array (FPGA) to achieve this objective. Based on the proposed system level architecture and design strategies, a prototype system is developed based on the Xilinx Virtex-II Pro XC2VP30 FPGA with the integration of embedded processor, memory control and interface technologies. Our system includes different functional modules, such as edge detection, zoom-in and zoom-out functions, which provides the flexibility of using this system as a general video processing platform according to different application requirements. The final system utilizes about 20% of logic resource, 50% of memory on chip, and has total power consumption around 203 mw.

2. Approach taken

In this part, we will give detailed mathematical foundations and algorithms of the proposed approaches for data mining. I will also present detailed analysis and discussions of the major characteristics of our method with comparison to the existing state-of-the-art research.

2.1 RAMOBoost – A Ranked minority over-sampling in Boosting approach (RAMOBoost) for learning from imbalanced data set

Motivated by SMOTE [14], SMOTEBoost [56], and ADAYN [57], we propose RAMOBoost, a ranked minority over-sampling technique embedded with a boosting procedure, to facilitate learning from imbalanced data sets. The objective of RAMOBoost is two-fold: to reduce the induction bias introduced from imbalanced data and to adaptively learning information from the data distribution. This is achieved in two respects: First, an adaptive weight adjustment procedure is embedded in RAMOBoost that shifts the decision boundary towards the difficult-to-learn examples from both the minority and majority classes. Second, a ranked sampling probability distribution is used to generate synthetic minority instances to balance the skewed distribution. The way our algorithm creating synthetic instances differs from SMOTE in that: in lieu of sampling minority examples indiscriminately and uniformly as in SMOTE, RAMOBoost evaluates the potential learning contribution of each minority example and determines their sampling weights accordingly.

The proposed RAMOBoost algorithm for imbalanced learning from binary classes is formulated as follows:

[Algorithm: ***RAMOBoost*** (N, T, k_1, k_2)]

Input:

-- Training data set with m class examples $\langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) \rangle$, where $\mathbf{x}_i$ ($i=1, \dots, m$) is an instance of the n dimensional feature space $\mathbf{X}$ and $y_i \in Y = \{major, minor\}$ is the class identity label associated with instance $\mathbf{x}_i$ for majority and minority class.

-- N : number of synthetic data to be generated at each iteration

-- T : number of iterations, namely number of the base classifiers

-- k_1 : number of nearest neighbors in adjusting the sampling probability of the minority examples

-- k_2 : number of nearest neighbors used to generate the synthetic data instances

Set $B = \{(i, y) : i \in \{1, \dots, m\}, y \neq y_i\}$

Initialize:

$D_i(i, y) \in \frac{1}{|B|}$ for $(i, y) \in B$ (for two class problems, $|B| = m$)

Do for $t = 1, 2, \dots, T$:

(1) Sampling the mislabeled training data with D_t , get back the sampling data set S_e and slice it into the majority data set e_1 and the minority data set e_2 , with the number of examples as m_{lt} and m_{st} , respectively.

(2) For each example $\mathbf{x}_i \in e_2$, find its k_1 nearest neighbors in the data set S_e according to the Euclidean distance in n dimensional space and calculate r_i defined as:

$$r_i = \frac{1}{1 + \exp(-\alpha \cdot \delta_i)}, \quad i = 1, 2, \dots, m_{st} \quad (1)$$

Where α is a coefficient and δ_i is the number of majority cases in k_1 examples.

(3) Normalize r_i according to:

$$\hat{r}_i = \frac{r_i}{\sum_{i=1}^{m_{st}} r_i} \quad (2)$$

Such that $\{\hat{r}_i\}$ is a distribution function: $\sum_{i=1}^{m_{st}} \hat{r}_i = 1$. Define $d_t = \{\hat{r}_i\}$.

(4) Sample e_2 with d_t , get back a sampling minority data set g_t , where there are m_{st} data inside.

(5) For each example $\mathbf{x}_i \in g_t$, find its k_2 nearest neighbors in e_2 according to the Euclidean distance in n dimensional space, and use linear interpolation to generate N synthetic data.

(6) Provide the base classifier with sampling data set S_e along with the N synthetic data.

(7) Get back a hypothesis $h_t : \mathbf{X} \times Y \rightarrow [0, 1]$.

(8) Calculate the pseudo-loss of h_t :

$$\varepsilon_t = \frac{1}{2} \sum_{(i,y) \in B} D_t(i,y) (1 - h_t(\mathbf{x}_i, y_i) + h_t(\mathbf{x}_i, y)) \quad (3)$$

(9) Set $\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$.

(10) Update D_t :

$$D_{t+1}(i, y) = \frac{D_t(i, y)}{Z_t} \beta_t^{(1 + h_t(\mathbf{x}_i, y_i) - h_t(\mathbf{x}_i, y))} \quad (4)$$

Where Z_t is a normalization constant.

End Loop

Output: The output hypothesis $h_{final}(\mathbf{x})$ is calculated as follows:

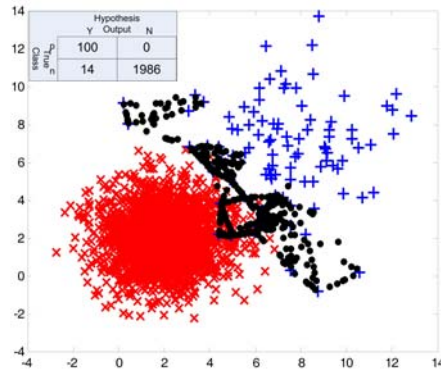
$$h_{final}(\mathbf{x}) = \arg \max_{y \in Y} \sum_{i=1}^T \left(\log \frac{1}{\beta_t} \right) h_t(\mathbf{x}, y) \quad (5)$$

According to this description, the RAMOBoost algorithm includes two mechanisms to facilitate learning from imbalanced data. The first consists of steps (2) to (5), where instances are adaptively generated according to their distributions. In this way, more synthetic instances are created for difficult-to-learn minority examples that are more likely to be misclassified compared to easy-to-learn minority examples. This is significantly different from the SMOTE algorithm where each minority example has equal weight and therefore the same number of synthetic instances are created for each minority example. The second mechanism: steps (6) to (10) use the pseudo-loss of the current hypothesis h_t to update the sampling distribution D_t , which is employed to sample the training data set in the next iteration as shown in step (1). Similar to the AdaBoost.M2 algorithm [58] [59], the pseudo-loss mechanism can adaptively shift the final hypothesis towards the decision boundary to facilitate the learning process.

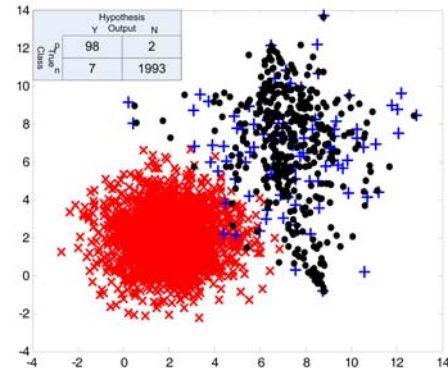
Similar to RAMOBoost, our previous work ADASYN [57] also aims to systematically generate synthetic minority instances according to the data distribution instead of using a uniform distribution. However, ADASYN does this in an aggressive manner: almost all of the generate synthetic minority instances are very close to the decision boundary. In contrast, RAMOBoost employs a parameter-specified logistic function to map the number of majority cases within the k nearest neighbors of a minority

examples under consideration to real number in the range $[0, 1]$ to determine the sampling probability of each minority example. In this way, RAMOBoost considers all minority examples for synthetic generation, albeit at varied level.

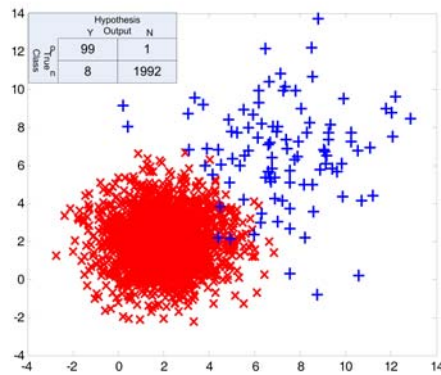
In order to compare and visualize the data generation mechanism of RAMOBoost with that of SMOTE and ADASYN, we provide a case study of a data set with 2000 majority examples and 100 minority examples, the result of which is shown in Fig. 2. Fig. 2(a) shows the original imbalanced data distribution, and Fig. 2(b), 2(c), and 2(d) show the post-SMOTE data distribution, the post-ADASYN data distribution, and the post-RAMOBoost data distribution, respectively. In all these figures, the x-mark, plus, and point shapes represent the original majority data, original minority data, and the generated synthetic data, respectively. In this case study, CART (Classification and Regression Tree) is used as classifier. The confusion matrix (in terms of instant counts) is used for performance assessment for different algorithms. Followed by the suggestions of [2] [14] [60], the minority class and the majority class are used as positive class and negative class, respectively. From Fig. 2, one can figure that the data generation process of RAMOBoost is more adaptive and systematic according to the data distribution. And RAMOBoost is credited accordingly with better performance than other algorithms for comparison (by confusion matrix).



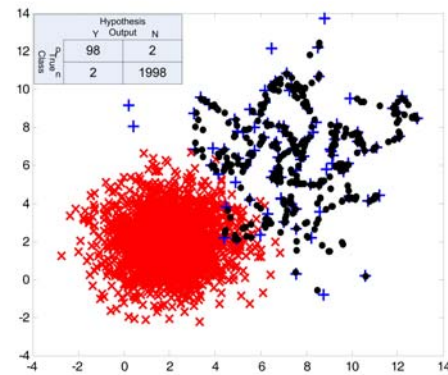
(a)



(b)



(c)



(d)

Fig. 2. Comparison of different synthetic data generation mechanisms. (a) The original imbalanced data distribution (2000 majority examples and 100 minority examples). (b) The data distribution after SMOTE method. (c) The data distribution after ADASYN method. (d) The data distribution after RAMOBoost method.

2.2 ADAIN – An adaptive incremental learning framework (ADAIN) for learning from the data flow

Motivated by the adaptive boosting principle and ensemble learning methodology [58] [59] we propose the ADAIN framework to enable knowledge accumulation and transformation to benefit learning from continuous data flow. Unlike traditional learning approaches, the objectives here are two-fold: 1) integrate previously learned knowledge with currently received data to improve learning from the new raw data, and 2) accumulate experience over time to support future decision-making processes.

Assume a learner is presented with a data flow over time. At time t , a new set of training data D_t is received. The previous knowledge in this case includes the hypothesis h_{t-1} , which was developed at time $t-1$ from the distribution function P_{t-1} applied to the data set D_{t-1} . Here the distribution function can be either a sampling probability function or weight distribution function for different instances in the data. Difficult examples that are hard to learn will carry higher weights compared to those examples that are easy to learn [58]. For the first chunk of the received data, if there is no *a priori* knowledge about the data distribution, the initial distribution function P_1 can be uniform because nothing has been learned yet. Otherwise, P_1 can be set according to any given prior knowledge. The proposed system level framework is illustrated in Fig. 3, followed by a detailed learning algorithm.

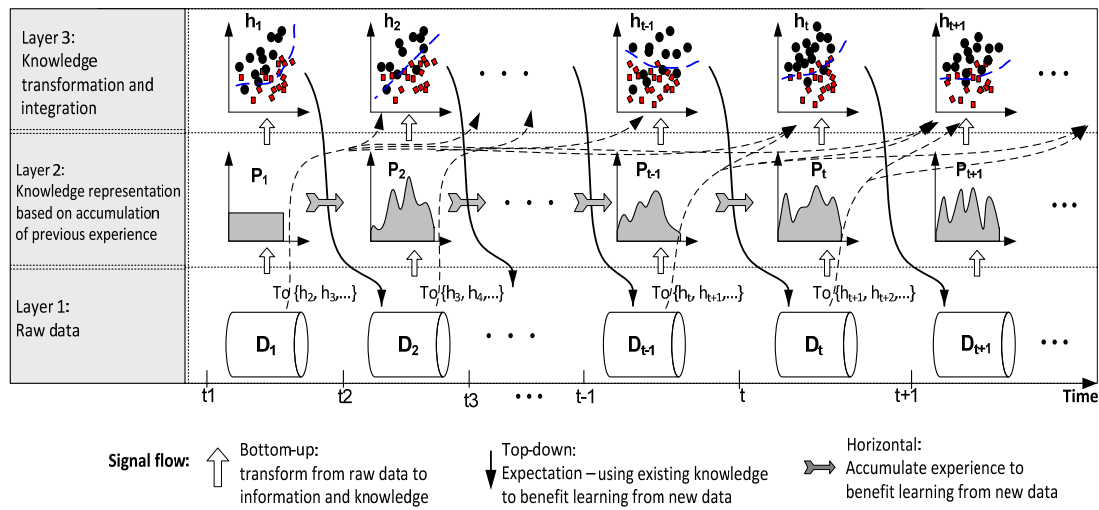


Fig. 3 Adaptive incremental learning for classification

[The ADAIN Framework]

Previous knowledge at time $(t-1)$:

-- Data set, D_{t-1} , with m instances: $\{\mathbf{x}_i, y_i\}$, $(i=1, \dots, m)$, where $\mathbf{x}_i$ is an instance in the n dimensional feature space $\mathbf{X}$ and $y_i \in Y = \{1, 2, \dots, c\}$ is the class identity label associated with $\mathbf{x}_i$.

-- Distribution function: P_{t-1} .

-- A hypothesis, h_{t-1} , developed by the data based on D_{t-1} with P_{t-1} .

Current input at time t :

-- A new data set, D_t , with m' instances, where m' may or may not be the same size as m , and can be represented as $\{\mathbf{x}_j, y_j\}$, $(j=1, \dots, m')$.

Learning procedure:

(1) Define a mapping function, φ , and estimate the initial distribution function $\hat{P}_t$ for D_t :

$$\hat{P}_t = \varphi(D_{t-1}, P_{t-1}, D_t) \quad (6)$$

(2) Apply hypothesis h_{t-1} to D_t , calculate the error of h_{t-1} .

$$\varepsilon_t = \sum_{j: h_{t-1}(\mathbf{x}_j) \neq y_j} \hat{P}_t(j) \quad (7)$$

(3) Refine the distribution function for D_t :

$$P_t = \frac{\hat{P}_t}{Z_t} \times \begin{cases} \varepsilon_t & \text{if } h_{t-1}(\mathbf{x}_j) = y_j \\ 1, & \text{otherwise} \end{cases} \quad (8)$$

where Z_t is a normalization constant so that P_t is a distribution

(4) Repeat the procedure when the next chunk of new data set D_{t+1} is received.

Output: The final hypothesis

$$h_{final}(\mathbf{x}) = \arg \max_{y \in Y} \sum_{T: h_T(\mathbf{x}) = y} \log \left(\frac{1}{\varepsilon_T} \right) \quad (9)$$

where T is the set of incrementally developed hypothesis in the learning life.

Fig. 3 visualizes the architecture of the learning process, which includes three layers and three signal flow directions for information exchange, experience accumulation, and knowledge integration. Layer 1 is the continuous stream raw data, Layer 2 is used to transform the raw data into knowledge representation based on the accumulation of previous experience (through a dynamic learning process to effectively adjust the distribution function from existing knowledge), and Layer 3 then develops multiple hypotheses by effective weight adjustments, which serves as the knowledge integration platform from multiple hypotheses.

The *bottom-up* flow transforms the original data D_t to information and knowledge representation: P_t and h_t . Based on P_t , a learning hypothesis h_t is developed, in which the decision boundary is automatically forced to be more focused on the difficult regions. After P_t and h_t have been obtained, the system uses its knowledge to facilitate learning from the next chunk of raw data, D_{t+1} . This is achieved by the top-down and horizontal signal flow, as illustrated in Fig. 4. The objective here is to inherit the adaptive boosting characteristic to improve incremental learning.

There are two mechanisms in ADAIN framework to facilitate the adaptive incremental learning capability. First, a mapping function φ (equation (6)) is used to find the data distribution relationship between D_t and D_{t-1} . The definition of φ can be user-specified in accordance with the requirements of specific applications. The objective of the φ function is to provide a quantitative representation of the relationship between different data distributions. Second, an initial estimation of P_t , denoted as $\hat{P}_t$, is made from knowledge contained in the hypothesis, h_{t-1} , applied to the new chunk of data D_t . The error measurement is calculated in equation (7), which represents the goodness-of-learning when the previous knowledge h_{t-1} is applied to the new data. This in turn is used to refine the distribution function in equation (8). In this way, misclassified instances (difficult examples) will receive higher weights, and the learning algorithm will adaptively push the decision boundary to focus on those hard-to-learn instances. Furthermore, since the hypothesis developed at the previous time step is used to evaluate its performance over the current data chunk, ADAIN implicitly takes into consideration all previous domain datasets for the current hypothesis, as illustrated by the dashed-arrow in Fig. 3.

We also want to point out that the proposed incremental learning framework is a general learning methodology. Therefore, different base learning algorithms, such as decision trees, neural networks, support vector machines, and others, can be embedded into this framework for incremental learning.

In the proposed learning framework, the mapping function φ (equation (6)) provides connections from past experience to the newly received data, and adapts such knowledge to future data chunks. Different design strategies of the φ function can be

used. For instance, Euclidean distance function can be employed as the mapping function to find the relationship between data D_{t-1} and D_t to estimate the initial distribution P_t as illustrated in Fig. 4 [49].

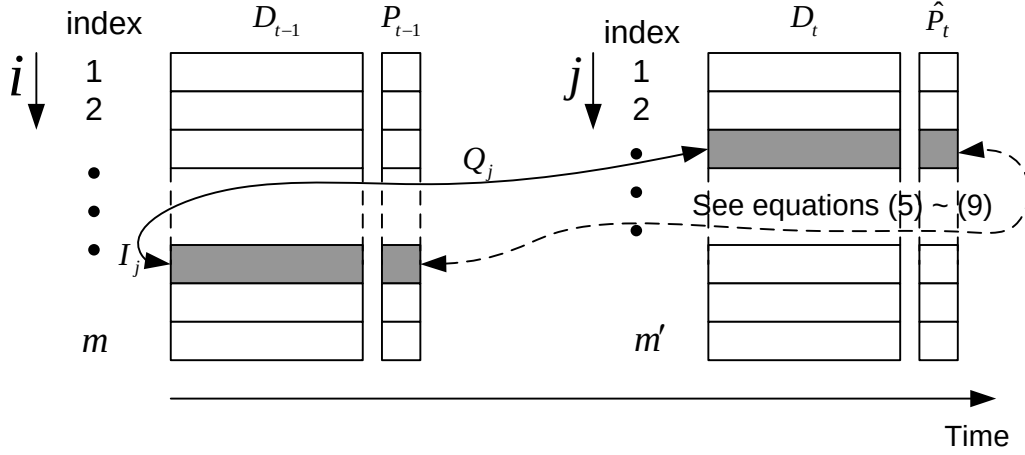


Fig. 4 Mapping function based on Euclidean distance

The fundamental mechanism is summarized as follows:

(1) a distance map ($\mathbf{DM}$) function between D_{t-1} and D_t is calculated:

$$\mathbf{DM}_{ji} = \sqrt{\sum_{k=1}^n (\mathbf{x}_{jk} - \mathbf{x}_{ik})^2}, \quad k=1, \dots, m \quad (10)$$

$$I_j = \arg \min_{i \in \{1, \dots, m\}} (\mathbf{DM}_{ji}) \quad (11)$$

$$Q_j = \min (\mathbf{DM}_{ji}) \quad (12)$$

Where $\mathbf{I} = [I_j] \in \{1, \dots, m\}$ is the index of the nearest neighbor in D_{t-1} for each data instance of the nearest neighbor in D_t , and $\mathbf{Q} = [Q_j] \in [0, \infty)$ is the corresponding distance value.

(2) After the distance Q_j ($j=1, \dots, m$) is determined, it is scaled according to:

$$\mathbf{Q}_s = \frac{1}{\exp(1 - \exp(-\mathbf{Q}))} \quad (13)$$

where $\mathbf{Q}_s \in \left(\frac{1}{e}, 1\right]$.

(3) With $\mathbf{Q}_s$, the initial estimation of the distribution function is updated:

$$\hat{P}_t = \frac{P_{t-1}(I) \times Q_s}{Z'_t} \quad (14)$$

where Z'_t is a normalization constant so that $\hat{P}_t$ is a distribution.

From equations (10) to (14), one can see that the key idea of using Euclidean distance mapping function is to provide a mechanism to pass previous knowledge to the new data analysis to facilitate incremental learning. When the boosting idea is applied to traditional static learning problem [58] [59], the weights can be updated iteratively based on the static training data in a sequential format. However, in the incremental learning scenarios, one can not directly obtain/update such weights when a new chunk of the data flow is received. Equations (10) to (14) provide such a connection (equivalent to the mapping function φ the pseudo code).

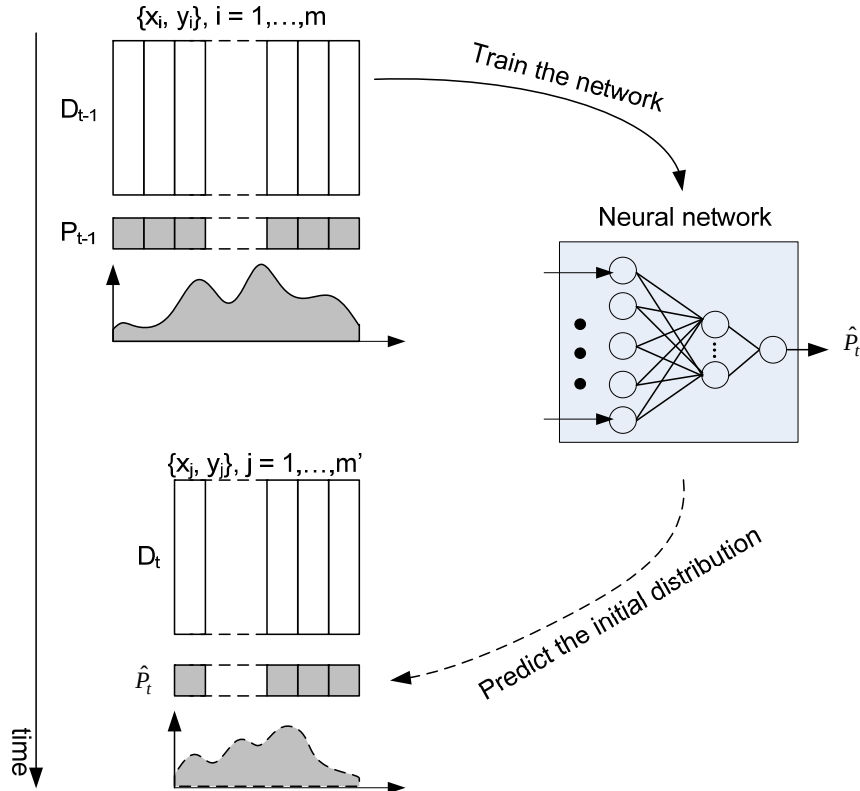


Fig. 5 Mapping function based on MLP

In this work, we propose that the *nonlinear regression models* can also be utilized as the mapping function of the proposed incremental learning framework. For instance, the multi-layer perceptron artificial neural network with backpropagation, which is abbreviated as “MLP” in the remaining parts of this article, can also be integrated into the

framework to implement the mapping function φ ; its idea is shown in Fig. 5. Based on the previous data information, D_{t-1} , and its associated distribution function, P_{t-1} , one can develop an MLP model to learn the relationships between the feature space and its corresponding numerical weight function, P_{t-1} . Then, when the new chunk of data, D_{t-1} , is received, one can use the trained MLP to obtain the initial estimation of the distribution function: $\hat{P}_t$ (equation (16)). Once the MLP output is predicted, one can normalize them to be a distribution function (summation equals to 1). We would also like to point out that technically speaking, other types of the regression models, such as SVMs and CART, can also be integrated into the proposed learning framework to accomplish the incremental learning capability, which provides the flexibility of using the ADAIN as a general incremental learning framework across a large variety of application domains.

2.3 SERA – A Selectively Recursive Approach towards nonstationary imbalanced data classification

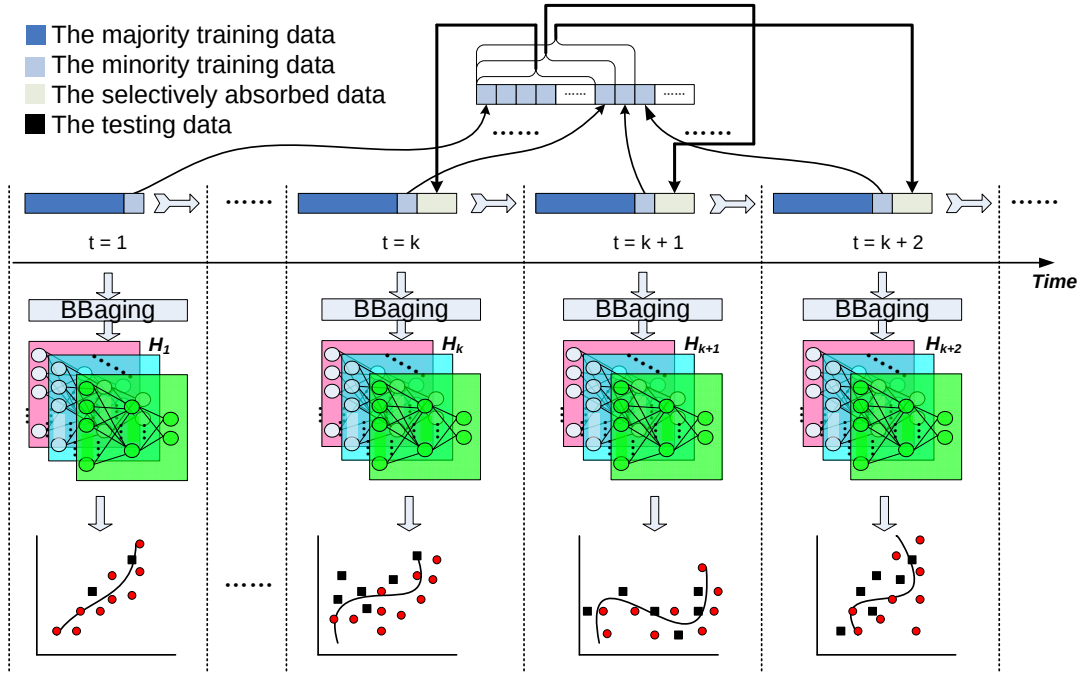


Fig. 6 The SERA framework

The SERA framework [115] is depicted in Fig. 6. And the pseudo-code of the proposed SERA algorithm for nonstationary imbalanced stream data mining is formulated as follows:

[The SERA Algorithm]

Input:

- The imbalanced ratio r specifying the proportions between the minority examples and majority examples in the training data chunk.
- Current training data chunk S_k with m training examples $\langle (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m) \rangle$, where k is the timestamp of the current data chunk, $\mathbf{x}_i$ is an instance of the n dimensional feature space $\mathbf{X}$ and $y_i \in Y = \{major, minor\}$ is the class identity label associated with the instance $\mathbf{x}_i$.
- Current testing data chunk T_k
- The data set C_{k-1} preserving all the minority examples $\{(\mathbf{x}'_i, y_i)\}$ within the training data chunk prior to the current timestamp k .
- The post-balance ratio f specifying the class ratio after balancing the current training data chunk by selectively incorporating the minority examples into C_{k-1} .

Algorithms:

- (1) Split S_k into P_k and N_k , where P_k denotes the minority example set and N_k denotes the majority example set.
- (2) If $f > (k-1) \times r$, then include all minority examples of C_{k-1} into the current training data chunk S_k for learning, i.e., $S'_k = \{S_k, C_{k-1}\}$.
- (3) Else
 - (3.1) Calculate the Mahalanobis distance d_i between P_k and each minority instance $\mathbf{x}'_i$ of C_{k-1} .
 - (3.2) Sort $\{d_i\}$ in ascending order, then pick out minority examples of C_{k-1} with respect to the first $(f-r) \times m$ terms in the sorted $\{d_i\}$ and associate them as the set M_k .
 - (3.3) Accommodate M_k into the current training data chunk S_k , i.e., $S'_k = \{S_k, M_k\}$.
- (4) Build the learning model based on S'_k , where ramifications exist as:
 - (4.1) Simply establish a single scoring hypothesis h_k based on S'_k .
 - (4.2) Otherwise: call *BBagging* function to establish the ensemble hypotheses $\{h_k^1, \dots, h_k^T\}$, where T specifies the number of iterations for BBagging.

Output:

-- If the single hypothesis establishment is adopted: Apply h_k on T_k , and get back the decision output Φ_k .

-- If the BBagging approach is adopted: Apply $\{h_k^1, \dots, h_k^T\}$ on T_k , and get back a decision output vector: $\{\Phi_k^1, \dots, \Phi_k^T\}$. Then:

$$\Phi_k = \frac{1}{T} \sum_{i=1}^T \Phi_k^i \quad (15)$$

[The BBagging Algorithm]

Input:

- The training example set $D = \{P, N\}$ with the size m , where P is the minority example set with the index set I_P in D , and N is the majority example set with the index set I_N in D .
- The number of iterations T .
- The cost factor k .

Initialize:

- Sampling probability mass function $\Psi(I_P) = k$ and $\Psi(I_N) = 1/k$

Algorithm:

Do for $t = 1, 2, \dots, T$

(1) Sampling D , and get back a sampled data set E_t .

(2) Provide the base classifier with E_t .

(3) Get back a hypothesis $h_t \rightarrow [0, 1]$

End Do

Output:

The Ensemble scoring hypotheses: $\{h_1, \dots, h_T\}$

Rather than adopting either of the sliding window or dynamic weights updating methods to handle imbalanced data streams with concept drifts, our proposed approach

consistently collects the minority examples $\{(\mathbf{x}'_i, y_i)\}$ from the training data chunks prior to the present timestamp t_k , and associates them together as C_{k-1} . Instead of feeding the entire C_{k-1} into the current training data chunk S_k to facilitate the learning process [61], our approach will apply the post-balance ratio f to proportionally accommodate C_{k-1} by somehow measuring the similarity between each of them and the current minority set P_k . The Mahalanobis distance [62] is employed for measuring such similarity.

Mahalanobis distance is part of the exponential term of the multi-dimensional Gaussian density function. It is considered generally more effective than Euclidean distance in determining the similarities among variables. Formally, Mahalanobis distance Ω from a set of n -variate instances with mean value $\boldsymbol{\mu} = [\mu_1, \dots, \mu_n]^T$ and covariance matrix Σ to an arbitrary instance $\mathbf{x} = [x_1, \dots, x_n]^T$ is defined as [62]:

$$\Omega = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T \Sigma (\mathbf{x} - \boldsymbol{\mu})} \quad (16)$$

Then those that are close to the set of present minority examples in Mahalanobis distance will be granted priority to be added into S_k . The balanced data set S'_k is thus obtained.

Towards learning from S'_k , we either directly build a hypothesis upon it or adopt BBagging to build an ensemble. Rather than maintaining a consistently uniform sampling function as done by Bagging [63], BBagging manually makes the sampling weights of minority examples greater than the majority examples by a proportional cost factor k . Apparently, BBagging will be reduced into the normal Bagging if the cost factor k is equal to 1.

SERA does not take either of the sliding windows or dynamic weights updating methods. The reason of this is based on the following two concerns:

Question 1:

Is the similarity of the target concept between the two data chunks solely or largely dependent on the difference of their timestamps?

Question 2:

Can the hypotheses plainly built upon the previous imbalanced data chunks significantly help the learning process of the data chunk under consideration?

In reference [64], the density function for the target concept of the KDD cup 1999 network intrusion dataset [65] in the timeframe of five weeks is plotted. The density curve keeps fluctuating all the time along the timeline and does not present any non-subtle patterns. Such observation on real-world dataset greatly undermines the foundation of sliding window approach: since the window length only concerns how far away the learning process for the current data should seek help from, it implicitly assumes that the

more adjacent in timestamp the model was learned, the more relevant it is to the current data chunk under consideration. Even if the gradual changes exist in the real-world applications, it cannot be possibly known beforehand the concept drifts of the data stream under consideration belong to what category. Furthermore, the data stream exhibiting gradual concept drifts at the beginning may dive into an unpredictably sudden change pattern in the future. By taking all these concerns into consideration, the DW method may not be a brilliant choice for dealing with nonstationary data streams.

The answer to the 2nd question is also largely negative. Since the minority examples are severely overwhelmed by the majority examples, the information regarding the minority examples from the previously built hypothesis on the data chunk under consideration is skinny, even if the overall accuracy is high. For instance, by predicting all the instances of a dataset with the minority class ratio being 0.01 to belong to the majority class, the classifier undoubtedly performs terribly on the minority instances even if its overall accuracy can reach to 0.99. To this end, including all previously built hypotheses into the current learning process can only make limited contribution to accurately predicate the minority instances.

There are studies showing that some classifiers can perform much better on post-balanced dataset than imbalanced dataset [11] [66]. We indeed take over this idea to handle the imbalanced learning problem by making the training data chunk more balanced. Although there are versatile definitions across various communities for over-sampling technique, we explicitly define it in the scenario of imbalanced learning as replicating the selected examples and adding them into the original dataset to augment its volume [67], which is widely applied in imbalanced learning research [14] [57]. The reason we opt out the over-sampling technique to balance the dataset is that: First, Solely depending on synthetic instances to balance the current imbalanced training data chunk tears apart the connection to all previous knowledge and thus results in the “catastrophic forgetting” [53]. Second, The over-sampling technique is somehow related to the discriminative algorithm which is more focused on learning the decision boundary, i.e., $p(y|x)$. The data distribution of the synthetic minority instances, i.e., $p(x|y)$, is therefore more likely to be severely deflected from the target concept of the original data chunk than the previous minority examples with not that severe drifting concepts.

Given that the accommodation of previous minority examples can help facilitate the learning on imbalanced dataset, is it better to include all previous minority examples into the current data chunk for learning, or we should constrict the scale of the inclusion at a certain level? As aforementioned, we believe only the minority examples with not that drifting-away target concepts are actually helpful for the learning process. And it is why we introduce the Mahalanobis distance to measure the severe degree of concept drifts for minority examples. Besides, since the minority examples becoming available previously and currently after all do not share the same distribution, it is better to limit the number of accommodated minority examples to keep them from non-trivially undermining the target concepts of the current data chunk.

The proposed BBagging is assumed to improve the prediction accuracy of the minority instances by introducing the cost factor k . Higher k brings more minority examples in the sampling dataset. However an excessively high cost factor may also

severely deteriorate the prediction performance on the majority instances. How to balance the performance on minority examples and the performance on the entire dataset is something one should carefully think about. One more comment to the learning model building as the 3rd step in the pseudo-code of SERA is that we do not claim the proposed BBagging is definitely better than a single classifier on an imbalanced data chunk.

2.4 Network Intrusion Detection Based on KDE and SOM

In this project, we have investigated of the use of the anomaly-based IDSs that are described mathematically by a group of random variables or a random vector $V = \{v_1, v_2, \dots, v_n\}$, with all components v_i 's scalar-valued random variables on the same probability space. We use kernel density estimation technique to estimate the probability density function (pdf) for the observed random variables without any underlying distributions specified *a priori*. Based on the preset confidence level, the two-side cutting limits are searched heuristically and iteratively. Due to the huge volume of data and the inherent characteristic of KDE method, it is very difficult to perform online intrusion detection and provide real-time protection because of the heavy computational load. In order to reduce the computational complexity of KDE, several methodologies have been proposed in literature. For instance, in [106], a reduced kernel density estimation algorithm that combines KDE with SOM is proposed and some theoretical results on binned kernel density estimation are also presented. In [107], self-organizing mixture networks are proposed for probability density estimation, and applications of this model over density profile estimation and pattern classification are presented to illustrate the effectiveness and efficiency of the proposed method. Fast Gauss transform technique is used in [108] to speed up the kernel density estimation and this algorithm is applied to vision tracking problems. We take advantage of the learning and clustering capabilities of SOM and employ SOM to preprocess the input data. The principle of learning in SOM is to self-organize the network of neurons to seek similar properties for certain input patterns. Therefore, SOM can form an approximation of the distribution of input space in a compact fashion. Using only the trained SOM neurons, instead of all the input vectors, as kernels to calculate pdf can significantly reduce the number of terms in a kernel density estimator, and thus greatly improve the efficiency and performance for the intrusion detection.

2.4.1 Definition of Outliers

Generally, an outlier is defined as an observation that lies outside the overall pattern of a distribution [90]. In a sense, the principles used to decide what will be considered abnormal are largely determined by the analysts or application settings.

In this project, we consider the univariate problems, and assume that there is only one cluster in the observed data. In other words, the outliers all lie in either the left or the right side of the cluster. We also assume an underlying distribution for the observed data. Then the outliers are defined as any sample X_{out} that lies in the outlier region $\Theta(\alpha, \beta)$ defined as

$$\Theta(\alpha, \beta) = \Theta_{upper}(\alpha) \cup \Theta_{lower}(\beta), \quad (17)$$

where:

$$\Theta_{upper}(\alpha) = \left\{ X : X \geq \lambda, P(X \geq \lambda) = \int_{\lambda}^{\infty} \hat{f}(x) dx = \alpha \right\},$$

$$\Theta_{lower}(\beta) = \left\{ X : X \leq \gamma, P(X \leq \gamma) = \int_{-\infty}^{\gamma} \hat{f}(x) dx = \beta \right\},$$

λ, γ are the upper and the lower cutting interval limits, respectively, which are determined by α, β that are the upper and the lower cutting probabilities dependent on specific problems or IDS systems. $\hat{f}$ is the estimate of the underlying pdf for the given observed data. We call $\Theta_{upper}(\alpha)$ the upper outlier region and $\Theta_{lower}(\beta)$ the lower outlier region. Fig. 7 demonstrates three cases of the outlier regions. In Fig. 7 (a) and (b), we only consider one-side tail outlier, i.e., $\beta = 0$ or $\alpha = 0$, whereas in Fig. 7 (c), we consider two-side tail outliers.

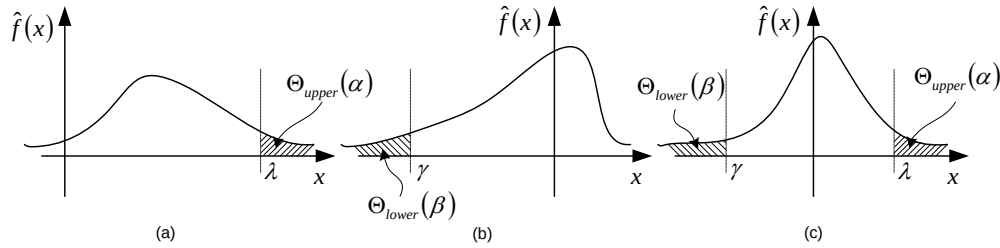


Fig. 7 Definition of outliers and outlier regions

2.4.2 Kernel Density Estimation (KDE)

Density estimation is defined as the construction of an estimate of the density function from the observed data. Various density estimation methods are summarized in [91]. Generally, there are two classes of density estimation methods, parametric and non-parametric. Parametric density estimation is conducted under the assumption that the data are drawn from a known parametric type of distribution, such as Gaussian distribution or uniform distribution, whereas nonparametric estimation has no such assumption. In this project, since the observed random variables of network activities are all with unknown pdfs, we adopt kernel density estimation method that is a non-parametric algorithm to estimate the underlying pdfs of the observed data.

Given a sample of N observations $X_1, X_2, \dots, X_N$, the kernel estimator can be obtained by

$$\hat{f}_h(x) = \frac{1}{N} \sum_{i=1}^N K_h(x - X_i) \quad (18)$$

where $K_h(\cdot) = \frac{1}{h}K(\cdot/h)$, $K(\cdot)$ is the kernel function. Usually, K is a symmetric probability density function that satisfies

$$\int_{-\infty}^{\infty} K(x) dx = 1 \quad (19)$$

There are various choices among kernels as shown in Table 1. In this project, we adopt the popular Gaussian kernel for analysis.

Table 1 Kernel functions

Kernel	$K(x)$
Uniform	$K(x) = \frac{1}{2}1_{(x \leq 1)}$
Triangle	$K(x) = (1 - x)1_{(x \leq 1)}$
Epanechnikov	$K(x) = \frac{3}{4}(1 - x^2)1_{(x \leq 1)}$
Quartic	$K(x) = \frac{15}{16}(1 - x^2)^2 1_{(x \leq 1)}$
Triweight	$K(x) = \frac{35}{32}(1 - x^2)^3 1_{(x \leq 1)}$
Gaussian	$K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$
Cosinus	$K(x) = \frac{\pi}{4} \cos\left(\frac{\pi}{2}x\right) 1_{(x \leq 1)}$

In the kernel density estimation, the bandwidth h is an important parameter. Too large h will lead to over-smoothing while too small h will result in an under-smoothed estimate. Therefore, h has to be chosen carefully. There are several ways to calculate h . These methods include the likelihood cross-validation method [92], the least-squares cross-validation method [93], the biased validation method [94], and the plug-in methods [95] [96]. Some empirical studies [97] show that most methods can work equally well. Interested readers can refer to [98] [99] for more detailed information. In this article, we calculate the window width as [100] [91]

$$h = 1.06\hat{\sigma}N^{-1/5} \quad (20)$$

where $\hat{\sigma}$ is the standard deviation of the sample X and N is the size of X . We note that the better estimation results may be obtained using a robust measure of spread or further improved by considering a skewness factor if the data show heavy skewness.

Interested reader may refer to [91] for further details.

2.4.3 Self-organizing Map (SOM)

Self-organizing map is a power learning model based on competitive learning and unsupervised learning [101] [102] [103]. The principle goal of SOM is to project the input vector with high dimension into low dimensional (normally less than three dimensions) discrete map in topologically ordered pattern. Therefore, SOM can be used for visualization, dimension reduction, vector quantization, and clustering.

Generally, SOM consists of a group of neurons that are organized as a low dimensional grid, usually a 2-D grid. Consider that all the input data are n -dimensional feature vectors, $X_i = [x_{i1}, \dots, x_{in}]^T \in \mathbb{R}^n$. Then, each neuron is associated with an n -dimensional feature vector or weight, $\omega_i = [\omega_{i1}, \dots, \omega_{in}]$. These weights associated to the neurons are adjusted according to the input patterns. In the training phase, three learning processes are involved, including competition, cooperation, and synaptic adaptation [104].

During the competition stage, at each training step t , an input vector $X(t)$ is randomly sampled from the input space. Euclidean distances between the input vector and each neuron are calculated, and the winning neuron is the neuron $n_{win}(t)$ with the smallest distance (maximum similarity) to the input vector

$$n_{win}(t) = \arg \min_i \|X(t) - \omega_i\|, \quad i = 1, 2, \dots, \ell \quad (21)$$

where ℓ is the total number of neurons in the SOM.

In the cooperation phase, a topological neighborhood around the winning neuron has to be determined. In order to guarantee the neurobiological correctness, the choice of the neighborhood should satisfy two conditions: (1) the topological neighborhood should be symmetric around the winning neuron that has the maximum value; (2) the rate of learning in the topological neighborhood decreases monotonically with increasing lateral distance between the synaptic neuron and the winning neuron. A common selection of the topological neighborhood is Gaussian function $h_{n_{win}, n_i}(t)$ defined as

$$h_{n_{win}, n_i}(t) = \exp\left(-\frac{d_{n_{win}, n_i}^2}{2\sigma^2(t)}\right), \quad i = 1, 2, \dots, \ell \quad (22)$$

where $d_{n_{win}, n_i}^2 = \|r_{n(X_t)} - r_i\|^2$ is the lateral distance between $n_{win}(t)$ and n_i , $i = 1, 2, \dots, \ell$, and $\sigma(t) = \sigma_0 \exp\left(-\frac{t}{\tau_1}\right)$, $t = 0, 1, 2, \dots$, is the effective width of the topological neighborhood.

Finally, in the synaptic adaptation stage, the weight of the winning neuron, as well as those of the excited neurons, is adjusted to the input pattern $X(t)$ based on the topological neighborhood function (22). The weight- updating rule of SOM can be written as

$$\omega_i(t+1) = \omega_i(t) + \eta(t)h_{n_{win},n_i}(t)(X(t) - \omega_i(t)) \quad (23)$$

where $\eta(t) = \eta_0 \exp\left(-\frac{t}{\tau_2}\right)$ is the monotonically decreasing learning rate.

2.4.4 Network Intrusion detection Algorithm

For our network intrusion detection method, we take the advantages of both SOM and KDE [116]. In this algorithm, we use KDE to estimate the underlying density function and search the cutting limits iteratively and heuristically. However, due to the huge amount of data collected from the network, it is very difficult to perform this task online because of the heavy computational load. Therefore, SOM is first employed to preprocess the input data and this operation can dramatically reduce the number of kernels that are used to calculate the pdfs while still being able to match its estimation accuracy. Specifically, since we obtained ℓ neurons using SOM trained from the input set, we can use these neurons as kernels to calculate the pdf instead of using all the input vectors as we did in equation (18), which will dramatically reduce the number of kernels used to calculate the pdf and reduce the computational load of KDE. We can rewrite equation (18) as

$$\hat{f}_h(x) = \sum_{i=1}^{\ell} \varepsilon_i K_h(x - n_i) \quad (24)$$

where $\varepsilon_i = \frac{N_i}{N}$, N_i is the number of input samples in the Voronoi region of the reference neuron n_i . Fig. 8 demonstrates an example of using the traditional KDE and the KDE based on SOM. Fig. 8 (a) shows the mechanism of the traditional KDE in which all data X_i , $i=1,2,\dots,18$, are directly used to estimate the pdf at X_e as $\hat{f}_h(X_e) = \frac{1}{18} \sum_{i=1}^{18} K_h(X_e - X_i)$, while Fig. 8 (b) shows the idea of the KDE based on SOM in which only the trained SOM neurons n_i , $i=1,2,3$ directly contribute to the calculation of the pdf at X_e as $\hat{f}_h(X_e) = \sum_{i=1}^3 \varepsilon_i \cdot K_h(X_e - n_i)$.

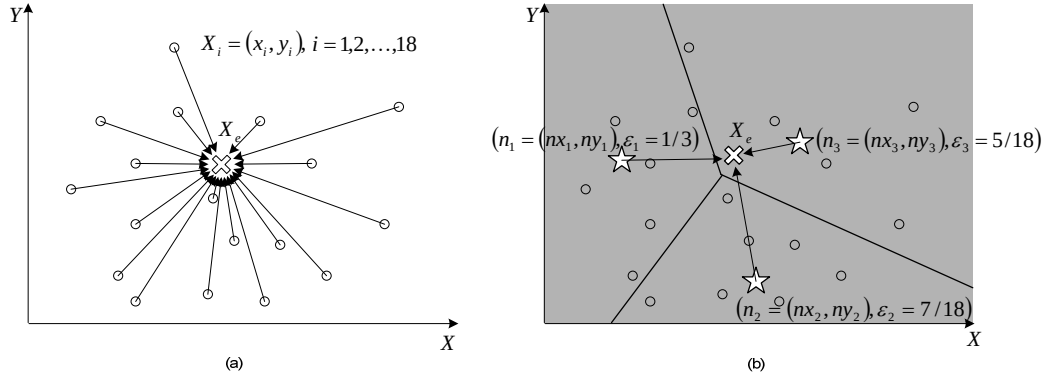


Fig. 8 (a) The traditional KDE and (b) the KDE Based on SOM

In this project, we consider one-dimensional input data, and thus the SOM is constructed as a one-dimensional grid. We use the trapezoidal rule to approximate the integral of the pdf. The proposed algorithm is summarized as follows.

[Algorithm: Network Intrusion Detection Algorithm]:

Input:

- A group of observations of the network activities. $X = \{X_k\}$, $X_k \in \mathfrak{R}$, $k = 1, 2, \dots, N$;
- Integer T specifying the number of iterations for training the SOM;
- Nonnegative numbers $\alpha, \beta \in \mathfrak{R}^+$ specifying the upper and the lower cutting probabilities, respectively;
- Positive number $\xi \in \mathfrak{R}^{++}$ specifying the interval used for searching the cutting limits.

Procedure:

1. Initialize a 1-d SOM as $\{n_i\}$, $i = 1, 2, \dots, \ell$, where ℓ is the total number of neurons in the SOM, and the weights ω_i associated with the neurons are initialized with values randomly sampled from the interval $[-0.1, 0.1]$.
2. Train the SOM. Do for $t = 1, 2, \dots, T$
 - a) Randomly pick a sample X_t from X .
 - b) Search the best-matching neuron (winning neuron) $n_{win}(t)$ using equation (21).
 - c) Update the synaptic weight vectors of all neurons using equations (22) and (23).
3. Search the upper cutting limit λ . Start from an arbitrary large value $\varphi_0 = X_{+\infty}$.
Estimate the pdf values $\{\hat{f}(\varphi_i)\}_{i=0,1,2,\dots}$ at a sequence of points $\{\varphi_i\}_{i=0,1,2,\dots}$ with

interval $-\xi$ as $\varphi_{k+1} - \varphi_k = -\xi$ using equation (24), and search $\lambda' = \varphi_u$ such that

$$P_{upper} = \frac{1}{2} \sum_{k=0}^u (\hat{f}(\varphi_{k+1}) + \hat{f}(\varphi_k)) \cdot \xi \geq \alpha. \text{ Return } \lambda = \min(\lambda', \max(X)).$$

4. Search the lower cutting limit μ . Start from an arbitrary small value $\varphi'_0 = X_{-\infty}$.

Estimate the pdf values $\{\hat{f}(\varphi'_i)\}_{i=0,1,2,\dots}$ at a sequence of points $\{\varphi'_i\}_{i=0,1,2,\dots}$ with interval ξ as $\varphi'_{k+1} - \varphi'_k = \xi$ using equation (24), and search $\mu' = \varphi'_l$ such that

$$P_{lower} = \frac{1}{2} \sum_{k=0}^l (\hat{f}(\varphi'_{k+1}) + \hat{f}(\varphi'_k)) \cdot \xi \geq \beta. \text{ Return } \mu = \max(\mu', \min(X)).$$

Output:

- Return λ and μ that are the upper and the lower cutting limits, respectively.
 - Report all the samples $X_{outlier}$ that lie in the outlier region (defined in equation (17)), i.e., $X_{outlier} \geq \lambda$ or $X_{outlier} \leq \mu$, as network intrusions, or network intrusion candidates for further analysis.
-

3. Results

In this section, we will demonstrate the applications of the proposed algorithms and models to various data mining applications. To provide a comprehensive assessment of the proposed approaches, we have investigated public available data sets and artificially synthetic data sets during this project period.

3.1 RAMOBoost – A Ranked minority over-sampling in Boosting approach (RAMOBoost) for learning from imbalanced data set

In order to gain a thorough insight in the competitiveness of the proposed RAMOBoost, we conduct various simulations of RAMOBoost and compare its performance with SMOTEBoost, SMOTE, ADASYN, AdaCost, BorderlineSMOTE, and SMOTE-tomek across different real-world data sets. In our current study, neural network with multi-layer perceptron (MLP) is employed as the base learner. The detailed configuration is as follows: The number of hidden layer neurons is set to be 4, and the number of input neurons is equal to the number of features for each data set. Similar to most of the existing imbalanced learning methods in literature, we also only consider two-class imbalanced problems in our current study. Therefore, the number of output neuron is set to be 2 for all the simulations. Sigmoid function is used as the activation function, and the inner training epochs is set to be 100 with a learning rate of 0.1.

Due to the concern that the scattered feature distribution of some data sets may hinder the neural network from converging enough fast for the parameter acceleration

process, before all data sets are presented to the comparative algorithms for learning, we employ the nonlinear normalization approach [85] to normalize the features of the data sets to reside in the interval [0, 1] first.

The performance of RAMOBoost is evaluated on 16 data sets from UCI machine learning repository [65] and ELENA project [68]. These data sets are varied in their sizes and class distributions to ensure a thorough assessment of the performance of RAMOBoost. Table 2 summarizes the characteristics of the data sets used in our simulation.

Table 2 Summary of the data sets characteristics (Sorted by imbalanced ratio)

Dataset	# feature	# data	# majority instances	# minority instances	Imbalanced Ratio
Sonar	60	208	97	111	0.47:0.53
Spambase	57	4601	1813	2788	0.39:0.61
Ionosphere	34	351	126	225	0.36:0.64
Pima-Indians-Diabetes	8	768	268	500	0.35:0.65
Wine	13	178	59	119	0.33:0.67
German	24	1000	300	700	0.30:0.70
Phoneme	5	5404	1586	3818	0.29:0.71
Vehicle	18	846	199	647	0.24:0.76
Texture	40	5500	1000	4500	0.18:0.82
Segment	18	2310	330	1980	0.14:0.86
Page_Blocks	10	5473	560	4913	0.10:0.90
Satimage	36	6435	626	5809	0.10:0.90
Vowel	10	990	90	900	0.09:0.91
Abalone	7	731	42	689	0.06:0.94
Glass	9	214	9	205	0.04:0.96
Yeast	8	483	20	463	0.04:0.96

Under the imbalanced learning scenario, the conventional assessment method of using a single criterion, such as overall accuracy, may not be able to provide a comprehensive assessment of the learning algorithm [16] [56] [60] [69] [70] [71] [72]. Considering a simple case of a given data set with 2% minority class examples and 98% majority class examples, a naive approach of classifying every example to be the majority class can at best provide an overall accuracy of 98% over the entire data set. However, in many real-world applications such as biomedical data analysis, such a classification performance would be unacceptable as it misclassifies all the minority cases, which generally are more important in such situations. As a result, the overall accuracy by itself may not be sufficient in evaluating the classification performance for imbalanced learning

problems. In our simulations, we adopt various assessment metrics related to the confusion matrix for analysis.

Let $\{p, n\}$ be the positive and negative testing examples and $\{Y, N\}$ be the classification results given by a learning algorithm for positive and negative predictions. A representation of classification performance can be formulated by a confusion matrix (contingency table) as illustrated in Fig. 9. Again, following the suggestion from [2] [14] [60], the minority class is used as the positive class and majority class is used as the negative class.

		Hypothesis output		Row counts:
		Y	N	
True class	p	TP (True Positives)	FN (False Negatives)	P_R
	n	FP (False Positives)	TN (True Negatives)	N_R

Fig. 9 Confusion matrix for performance evaluation

Based on Fig. 2, the metrics used to assess learning from imbalanced data sets in our simulations are defined as follows:

Overall Accuracy (OA):

$$OA = \frac{TP + TN}{TP + TN + FP + FN} \quad (24)$$

Precision:

$$Precision = \frac{TP}{TP + FP} \quad (25)$$

Recall:

$$Recall = \frac{TP}{TP + FN} \quad (26)$$

F-measure:

$$F - measure = \frac{(1 + \beta^2) \cdot Recall \cdot Precision}{\beta^2 \cdot Recall + Precision} \quad (27)$$

Where β is a coefficient to adjust the relative importance of *Precision* versus *Recall* (β is set to 1 in our simulation).

G-mean:

$$\begin{aligned} G - mean &= \sqrt{\text{positive accuracy} \times \text{negative accuracy}} \\ &= \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \end{aligned} \quad (28)$$

Another popular assessment method for imbalanced learning is the receiver operating characteristic (ROC) graph [7] [56] [69]. Based on the confusion matrix as defined in Fig. 9, one can calculate the tp_rate and fp_rate as follows:

$$tp_rate = \frac{TP}{P_R} \quad (29)$$

$$fp_rate = \frac{FP}{N_R} \quad (30)$$

ROC space is established by plotting tp_rate over fp_rate rate. Generally speaking, hard-type classifiers (those that only output discrete class labels) correspond to points in the ROC space: (fp_rate, tp_rate) . On the other hand, soft-type classifiers (those that output a likelihood of the degree to which an instance belongs to each class label) correspond to curves in the ROC space. Such curves are formulated by adjusting the decision threshold to generate a series of points in the ROC space. In order to assess different classifiers' performance in this case, one generally uses the area under curve (AUC) as an evaluation criterion. A detailed discussion of ROC analysis and its assessment for classifier performances can be found in [56] [69].

In our current simulation, we use 20 boosting iterations ($T = 20$ in the algorithm) as suggested in [73] for the ensemble learning. The number of synthetic data generated at each boosting iteration is set to be 200% of the number of the minority instances [2]. The parameter k_1 and k_2 is set to be 5 and 10, respectively, and the mapping coefficient α is equal to 0.3. For SMOTEBoost, SMOTE, ADASYN, BorderlineSMOTE, and SMOTE-tomek, the number of nearest neighbors is set to be 5. The cost factor C for AdaCost is set to be 3 according to the suggestion of [23] (C should be an integer between 2 and 9).

The simulation results are based on the average of 10 runs. At each run, we randomly draw half of the data as training data and use the remaining half as the testing data.

Fig. 10 gives several snapshots of the averaged ROC graphs of the RAMOBoost, SMOTEBoost, SMOTE, ADASYN, AdaCost, BorderlineSMOTE, and SMOTE-tomek methods. Here Fig. 10(a), 10(b), 10(c) and 10(d) represent the results for the German,

Ionosphere, Page Blocks, and Phoneme data sets, respectively. These figures indicate that RAMOBoost method is competitive when compared to other methods in the ROC space.

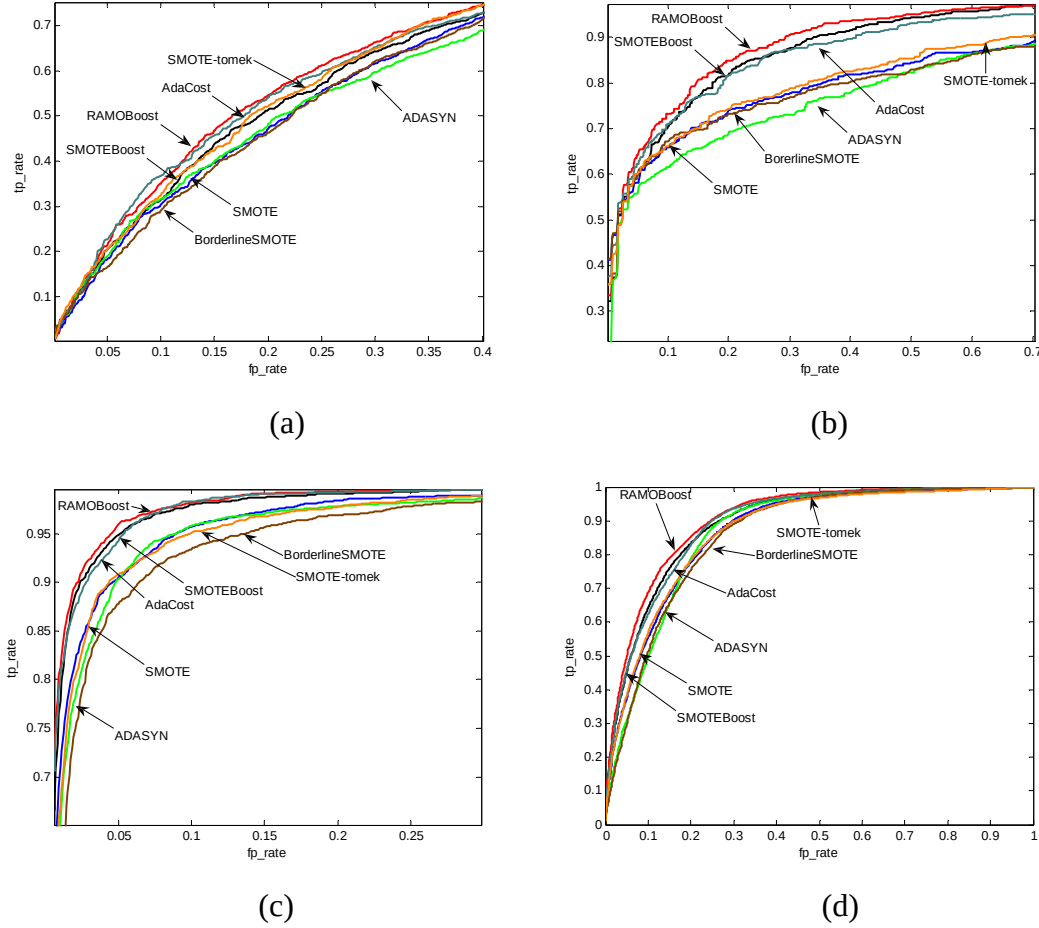


Fig. 10 The averaged ROC curves for RAMOBoost, SMOTEBoost, SMOTE, ADASYN, BorderlineSMOTE, and SMOTE-tomek methods

Table 3 summarizes the performance of the comparative algorithms, in which the best performance of each algorithm across each evaluation criteria is highlighted. Table 3 lists the AUC values for each method; the best performance is also highlighted.

Table 3 Evaluation metrics and performance comparison

Dataset	Methods	OA	Precision	Recall	F-measure	G-mean
Abalone	RAMOBoost	0.9405	0.4968	0.4889	0.4813	0.6808
	SMOTEBoost	0.943	0.5181	0.5348	0.5173	0.7134
	SMOTE	0.9477	0.5886	0.5328	0.5412	0.7166

	ADASYN	0.9101	0.361	0.4838	0.3892	0.6684
	AdaCost	0.952	0.241	0.4003	0.455	0.6156
	BorderlineSMOTE	0.9493	0.294	0.4855	0.554	0.686
	SMOTE-tomek	0.9441	0.261	0.4319	0.492	0.6433
Pima- Indians- Diabetes	RAMOBoost	0.724	0.5766	0.7467	0.6497	0.729
	SMOTEBoost	0.7229	0.5764	0.74	0.6466	0.7267
	SMOTE	0.7214	0.5746	0.74	0.6496	0.7281
	ADASYN	0.5539	0.4357	0.9709	0.5994	0.5702
	AdaCost	0.744	0.2816	0.61	0.3849	0.7043
	BorderlineSMOTE	0.7018	0.375	0.7656	0.5029	0.7154
	SMOTE-tomek	0.7039	0.3956	0.8102	0.5313	0.7248
Satimage	RAMOBoost	0.9195	0.5671	0.7127	0.6312	0.819
	SMOTEBoost	0.923	0.5867	0.6717	0.6276	0.7986
	SMOTE	0.8977	0.4791	0.606	0.5327	0.7465
	ADASYN	0.8422	0.3645	0.8431	0.5084	0.8424
	AdaCost	0.9217	0.552	0.5426	0.371	0.7118
	BorderlineSMOTE	0.8938	0.685	0.9652	0.3752	0.7598
	SMOTE-tomek	0.8957	0.701	0.9361	0.3679	0.773
Vehicle	RAMOBoost	0.9655	0.9142	0.9398	0.926	0.956
	SMOTEBoost	0.967	0.9137	0.946	0.929	0.9591
	SMOTE	0.9589	0.891	0.9373	0.9132	0.9511
	ADASYN	0.821	0.5665	0.9927	0.7206	0.8737
	AdaCost	0.9652	0.9132	0.9575	0.371	0.9623
	BorderlineSMOTE	0.961	0.913	0.9652	0.3752	0.9624
	SMOTE-tomek	0.9482	0.9091	0.9361	0.3679	0.9436
Vowel	RAMOBoost	0.999	0.9934	0.9931	0.9931	0.9962
	SMOTEBoost	0.9974	0.9842	0.9867	0.9853	0.9925
	SMOTE	0.9794	0.8569	0.9379	0.893	0.9599
	ADASYN	0.9101	0.5095	0.9488	0.6623	0.927
	AdaCost	0.9913	0.903	0.9696	0.9651	0.9813
	BorderlineSMOTE	0.9766	0.871	0.9222	0.9591	0.9515
	SMOTE-tomek	0.9747	0.889	0.9382	0.9624	0.9576
Yeast	RAMOBoost	0.9581	0.467	0.4341	0.4418	0.6405
	SMOTEBoost	0.9585	0.4941	0.4732	0.4687	0.6651
	SMOTE	0.9722	0.7557	0.5107	0.5761	0.703
	ADASYN	0.9552	0.5276	0.4891	0.4758	0.681
	AdaCost	0.9718	0.479	0.4524	0.344	0.6593
	BorderlineSMOTE	0.973	0.492	0.4882	0.368	0.6812

	SMOTE-tomek	0.9768	0.42	0.5107	0.384	0.7049
Phoneme	RAMOBoost	0.7921	0.5914	0.9068	0.7158	0.8222
	SMOTEBoost	0.8018	0.6131	0.8524	0.7128	0.8159
	SMOTE	0.786	0.5952	0.8248	0.6899	0.7942
	ADASYN	0.726	0.5137	0.9513	0.6671	0.777
	AdaCost	0.819	0.2473	0.702	0.3657	0.7797
	BorderlineSMOTE	0.7632	0.3308	0.8741	0.4799	0.7918
	SMOTE-tomek	0.7884	0.2985	0.8151	0.4369	0.7965
Texture	RAMOBoost	0.999	0.9986	0.9966	0.9976	0.9981
	SMOTEBoost	0.999	0.9976	0.997	0.9973	0.9982
	SMOTE	0.9949	0.9853	0.9863	0.9858	0.9916
	ADASYN	0.9156	0.6837	0.995	0.8101	0.9453
	AdaCost	0.9987	0.9798	0.9953	0.9946	0.9974
	BorderlineSMOTE	0.9928	0.9783	0.9811	0.9917	0.9881
	SMOTE-tomek	0.9976	0.9793	0.9913	0.9937	0.9951
Spambase	RAMOBoost	0.9448	0.9244	0.9387	0.9315	0.9438
	SMOTEBoost	0.9435	0.9191	0.9418	0.9302	0.9432
	SMOTE	0.9397	0.9194	0.9311	0.9251	0.9382
	ADASYN	0.7746	0.6424	0.9851	0.7776	0.7904
	AdaCost	0.947	0.8974	0.9413	0.8588	0.9462
	BorderlineSMOTE	0.9291	0.9028	0.936	0.8632	0.9302
	SMOTE-tomek	0.9376	0.9002	0.9384	0.8611	0.9377
Ionosphere	RAMOBoost	0.841	0.8572	0.6638	0.744	0.7874
	SMOTEBoost	0.8251	0.8244	0.6346	0.7156	0.7662
	SMOTE	0.8177	0.8026	0.6425	0.7106	0.7643
	ADASYN	0.6749	0.5263	0.7602	0.6198	0.6912
	AdaCost	0.8337	0.8237	0.6059	0.7352	0.7604
	BorderlineSMOTE	0.8206	0.8466	0.6516	0.7078	0.7698
	SMOTE-tomek	0.8166	0.8494	0.6539	0.711	0.7677
Wine	RAMOBoost	0.98	0.9525	0.9885	0.9696	0.9813
	SMOTEBoost	0.9787	0.9492	0.9885	0.9678	0.9805
	SMOTE	0.9787	0.9505	0.9885	0.9684	0.9804
	ADASYN	0.7933	0.6094	1	0.7536	0.8382
	AdaCost	0.9764	0.9319	0.9813	0.9648	0.9769
	BorderlineSMOTE	0.9753	0.9419	0.9885	0.9681	0.9778
	SMOTE-tomek	0.9551	0.9467	0.9853	0.9696	0.9629
Segment	RAMOBoost	0.997	0.9854	0.9907	0.988	0.9941
	SMOTEBoost	0.9965	0.9853	0.99	0.9876	0.9938

	SMOTE	0.9958	0.9835	0.9863	0.9848	0.9918
	ADASYN	0.9254	0.6253	1	0.798	0.9556
	AdaCost	0.9965	0.9845	0.9913	0.9843	0.994
	BorderlineSMOTE	0.9954	0.984	0.9869	0.9822	0.9918
	SMOTE-tomek	0.9953	0.984	0.9863	0.982	0.9915
German	RAMOBoost	0.7262	0.5602	0.527	0.5409	0.6547
	SMOTEBoost	0.7072	0.5258	0.5126	0.5176	0.6375
	SMOTE	0.685	0.4878	0.557	0.5192	0.642
	ADASYN	0.4918	0.3651	0.8762	0.5143	0.5282
	AdaCost	0.748	0.3963	0.4797	0.5283	0.6446
	BorderlineSMOTE	0.6846	0.4522	0.5754	0.5151	0.6492
	SMOTE-tomek	0.691	0.4777	0.6296	0.5148	0.6711
Glass	RAMOBoost	0.9748	0.6169	0.8464	0.7731	0.861
	SMOTEBoost	0.9748	0.648	0.9464	0.743	0.9596
	SMOTE	0.9897	0.894	0.9179	0.8874	0.9491
	ADASYN	0.9421	0.4552	0.7986	0.497	0.8555
	AdaCost	0.991	0.6377	0.9429	0.7722	0.9625
	BorderlineSMOTE	0.9907	0.6368	0.9262	0.704	0.9543
	SMOTE-tomek	0.9879	0.6359	0.9119	0.6988	0.9414
Page_Blocks	RAMOBoost	0.97	0.8326	0.8928	0.8614	0.9349
	SMOTEBoost	0.9696	0.834	0.8825	0.8573	0.9297
	SMOTE	0.9594	0.7781	0.8563	0.814	0.9118
	ADASYN	0.9251	0.5862	0.9414	0.7223	0.9322
	AdaCost	0.9704	0.7912	0.8559	0.8469	0.9175
	BorderlineSMOTE	0.9463	0.7853	0.8713	0.8171	0.912
	SMOTE-tomek	0.9576	0.7832	0.8627	0.8168	0.9139
Sonar	RAMOBoost	0.78	0.7566	0.7813	0.7672	0.7796
	SMOTEBoost	0.7702	0.7459	0.7748	0.7579	0.7697
	SMOTE	0.7606	0.733	0.7687	0.7485	0.7605
	ADASYN	0.5712	0.5184	0.9815	0.678	0.4624
	AdaCost	0.7721	0.7559	0.7644	0.7597	0.7711
	BorderlineSMOTE	0.7606	0.7364	0.771	0.7494	0.7607
	SMOTE-tomek	0.7442	0.7379	0.8073	0.7144	0.7448
Winning Times	RAMOBoost	7	10	1	12	9
	SMOTEBoost	2	3	3	1	2
	SMOTE	0	3	1	2	1
	ADASYN	0	0	10	0	1
	AdaCost	6	0	0	0	1

	BorderlineSMOTE	1	0	1	1	1
	SMOTE-tomek	0	0	0	0	1

From Table 3 and Table 4, we can say that the proposed RAMOBoost algorithm can provide competitive results compared to all comparative approaches in terms of *OA*, *Precision*, *F – measure*, and *G – mean*. Furthermore, the empirical results on RAMOBoost and SMOTEBoost validate that although RAMOBoost shares the same boosting procedure and the same data generation technique with SMOTEBoost, its adaptive ranking mechanism for determining the number of synthetic instance for each minority example make its performance superior to that of SMOTEBoost. For *Recall* performance, we see that ADASYN seems to provide a better *Recall* rate on these data sets. This is because ADASYN can learn very aggressively from the boundary since it generates synthetic data instances very close to the decision boundary (see Fig. 2(c)). This means that ADASYN may push the algorithm to focus on the minority (positive) class data to improve the *Recall* criteria (see definition in equation (26)), while the overall performance may not improve significantly. In other words, if one algorithm classifies all testing data as “positive” (minority class), its “*Recall*” rate will be maximized even if the overall performance is low. The results in Table 3 show that ADASYN performs better than other comparative algorithms in terms of *Recall*, which only stands for the number of correctly classified minority instances, but performs worse in all other assessment metrics, such as F-measure and G-mean which represent algorithm’s overall performance on imbalanced data sets.

Table 4 AUC performance characteristics

Dataset	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-tomek
Abalone	0.97609	0.92271	0.92291	0.89179	0.92395	0.90322	0.9039
Pima-Indians-Diabetes	0.79608	0.79825	0.80428	0.8144	0.81805	0.7947	0.81186
Satimage	0.9486	0.94678	0.89748	0.92234	0.93325	0.90189	0.90251
Vehicle	0.99487	0.99446	0.99314	0.97517	0.99511	0.99405	0.98948
Vowel	0.9999	0.99988	0.99615	0.98512	0.99906	0.99552	0.99344
Yeast	0.74512	0.74878	0.81603	0.77902	0.7792	0.8096	0.8241
Phoneme	0.90621	0.89472	0.87186	0.86497	0.89395	0.86103	0.87136
Texture	0.99999	0.99998	0.9992	0.99487	0.99991	0.99856	0.99967
Spambase	0.98379	0.98329	0.97942	0.96849	0.98552	0.97362	0.97649
Ionosphere	0.90138	0.88907	0.82093	0.79778	0.88186	0.81265	0.8265
Wine	0.9994	0.99937	0.99908	0.99607	0.99905	0.99796	0.99753
Segment	0.99976	0.99978	0.99959	0.99903	0.99974	0.9995	0.99961
German	0.74139	0.73357	0.71365	0.70182	0.71254	0.7105	0.73469
Glass	0.99478	0.99429	0.99801	0.97723	0.99741	0.99757	0.99736
Page_Blocks	0.98899	0.98772	0.97993	0.97621	0.98861	0.97063	0.97754
Sonar	0.86343	0.86176	0.84311	0.82832	0.76864	0.84205	0.82378

Winning Times	10	1	2	0	3	0	0
---------------	----	---	---	---	---	---	---

To evaluate the robustness of RAMOBoost against other comparative algorithms in different parameter configurations and scenarios, simulations on tuning the minority oversampling ratios and the imbalanced ratio are performed. For space consideration, we only present the results on “Abalone” data set here. Again, the neural network with MLP with the configuration described as aforementioned is used as the base learner. The simulation results are also based on the 10 random runs each of which divides the original data set evenly and randomly into training and testing data sets.

In reference [74], it is suggested that the over-sampling ratio could play a critical role for imbalanced learning problems, which motivates our simulation for evaluating the performance of RAMOBoost against other comparative algorithms under different over-sampling ratio. Specifically, the over-sampling ratio for the minority class is increased progressively from 100% to 500% with an interval of 100%. Table 5 displays the simulation results of 10 random runs of the averaged AUC of the comparative algorithms on learning from this data set, in which the best performance is highlighted, and the “Win-Lost-Tie (W-L-T)” information is also given.

Table 5 AUC under different over-sampling ratios

Over-Sampling ratio	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-tomek
100%	0.93202	0.91099	0.91676	0.89424	0.91908	0.90968	0.8867
200%	0.9308	0.90747	0.91655	0.89257	0.92351	0.90706	0.89791
300%	0.93244	0.91241	0.92108	0.88736	0.92208	0.90997	0.8977
400%	0.93146	0.90854	0.9221	0.88713	0.9227	0.90633	0.90664
500%	0.92374	0.90976	0.92317	0.87863	0.92206	0.91136	0.90995
W-L-T	5-0-0	0-5-0	0-5-0	0-5-0	0-5-0	0-5-0	0-5-0

The original “Abalone” data set has 28 classes and 4177 examples, in which we only employed two classes to evaluate the comparative algorithms, the results of which are shown in Table 3 and 4. In order to obtain versatile imbalanced ratio, we manipulate the classes’ combination of the original “Abalone” data set to form minority class and majority class. Table 6 summarizes the details for such combination policy and the corresponding imbalanced ratio. And Table 7 presents the simulation results on the simulation on tuning the imbalanced ratio, in which the best performance is highlighted.

The simulation results as shown in Table 5 and 7 illustrate the robustness of RAMOBoost as exposed to different internal (over-sampling ratio) and exterior (data set with different imbalanced class ratio) configurations. More importantly, from Table 7, it can be observed that RAMOBoost behaves competitive with all other comparative algorithms even when the data set is of very imbalanced class distributions.

Table 6 Combination of classes in “Abalone” data set

Index	Minority Combination	Majority Combination	# Minority	# Majority	Imbalanced ratio
I	1 ⊕ 2 ⊕ 22 ⊕ 24 ⊕ 25 ⊕ 26 ⊕ 27 ⊕ 28	8 ⊕ 9 ⊕ 10 ⊕ 11	15	2378	0.0063:0.9937
II	I ⊕ 23	8 ⊕ 9 ⊕ 10 ⊕ 11	24	2378	0.01:0.99
III	II ⊕ 21	8 ⊕ 9 ⊕ 10 ⊕ 11	38	2378	0.0157:0.9843
IV	III ⊕ 3	8 ⊕ 9 ⊕ 10 ⊕ 11	53	2378	0.0218:0.9782
V	IV ⊕ 20	8 ⊕ 9 ⊕ 10 ⊕ 11	79	2378	0.0322:0.9678
VI	V ⊕ 19	8 ⊕ 9 ⊕ 10 ⊕ 11	111	2378	0.0446:0.9554
VII	VI ⊕ 18 ⊕ 4	8 ⊕ 9 ⊕ 10 ⊕ 11	210	2378	0.0811:0.9189
VIII	VII ⊕ 17 ⊕ 15	8 ⊕ 9 ⊕ 10 ⊕ 11	371	2378	0.1350:0.8650
IX	VIII ⊕ 5	8 ⊕ 9 ⊕ 10 ⊕ 11	486	2378	0.1797:0.8303
X	IX ⊕ 6	8 ⊕ 9 ⊕ 10 ⊕ 11	745	2378	0.2386:0.7614

Table 7 AUC under different imbalanced ratio

Imbalanced Ratio	RAMOBoost	SMOTEBoost	SMOTE	ADASYN	AdaCost	BorderlineSMOTE	SMOTE-tomek
0.0063:0.9937	0.97887	0.97558	0.94373	0.94163	0.96915	0.9166	0.90582
0.01:0.99	0.90648	0.90557	0.90495	0.90412	0.9049	0.84195	0.8784
0.0157:0.9843	0.91886	0.91913	0.92122	0.92361	0.92921	0.91407	0.89492
0.0218:0.9782	0.95542	0.95072	0.93376	0.93219	0.95144	0.89655	0.92929
0.0322:0.9678	0.9584	0.9523	0.94371	0.93093	0.95562	0.93095	0.93487
0.0446:0.9554	0.94454	0.93652	0.91817	0.92162	0.94109	0.86302	0.90876
0.0811:0.9189	0.95025	0.94594	0.93479	0.9348	0.95068	0.92926	0.91392
0.1350:0.8650	0.91502	0.90501	0.89994	0.87255	0.91194	0.89588	0.8867
0.1797:0.8303	0.92274	0.91745	0.91241	0.90206	0.91994	0.90475	0.90704
0.2386:0.7614	0.89696	0.884	0.884	0.87725	0.89389	0.86997	0.86978
W-L-T	8-2-0	0-10-0	0-10-0	0-10-0	2-8-0	0-10-0	0-10-0

3.2 ADAIN – An adaptive incremental learning framework (ADAIN) for learning from the data flow

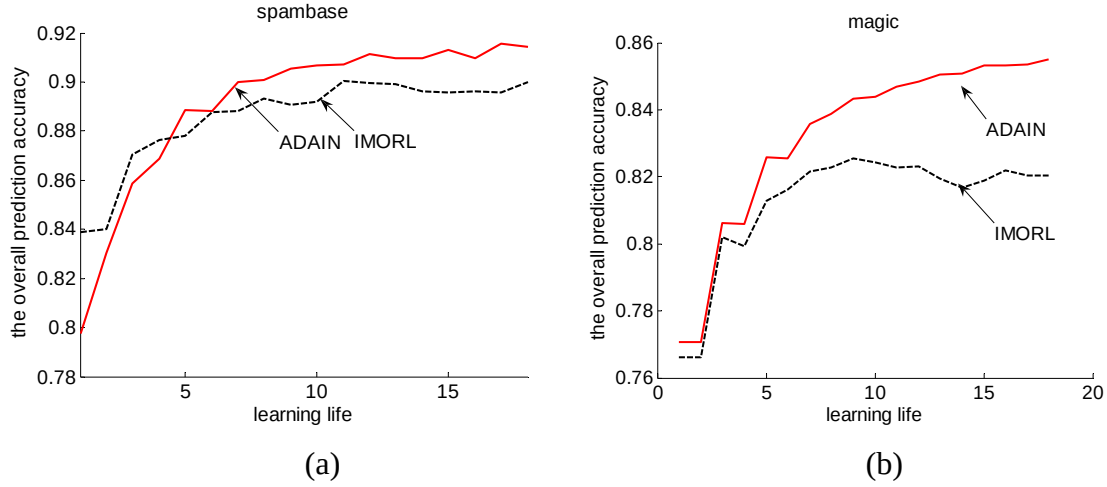
In order to validate the performance of the proposed ADAIN framework, 4 real-world data sets with varied size and number of classes from UCI machine learning repository [65] are employed to accomplish the comparative study in this research. The detailed information of these datasets can be found in Table 8.

Table 8 Information regarding the simulation data sets

Data set	# feature	# example	# class
spambase	57	4601	2
magic	10	19020	2
waveform	40	5000	3
sat	36	6435	6

In this simulation, each dataset is initially randomly sliced into 20 chunks with identical size. At each run, one chunk is randomly selected to be the testing data, and the remaining 19 chunks are sequentially fed to the classifier over time for incremental learning. All the simulation results for each datasets are averaged on 20 runs. CART is employed as the base learner in our current study. For the regression model based mapping function design, we adopted the MLP structure with 10 hidden layer neurons and 1 output neuron. The number of input neurons is set equal to the number of features for each data set.

Fig. 11 visualizes the prediction overall accuracy tendency over time of ADAIN as compared to [49], where Fig. 11(a), 11(b), 11(c), and 11(d) represents the data sets “spambase”, “magic”, “waveform”, and “sat”, respectively. From these figures, one can clearly see that the classifier's performance can increase over time, which means the system can adaptively learn over time, and accumulate knowledge to facilitate the future learning and decision-making processes. Table 9 shows the numerical accuracy for each individual class and the entire datasets. It can be intuitively figured that ADAIN can remarkably improve the learning performance compared to the method in [49].



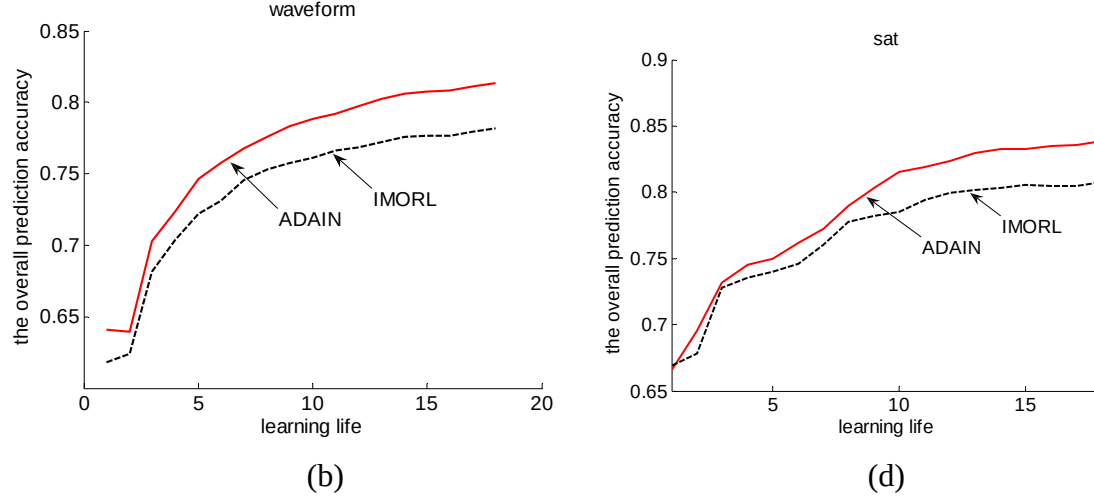


Fig. 11 Prediction overall accuracy

Table 9 The averaged prediction accuracy

Data sets	Methods	Prediction Accuracy						
		class 1	class 2	class 3	class 4	class 5	class 6	Overall
spambase	ADAIN	0.882	0.9352	--	--	--	--	0.9142
	IMORL	0.9106	0.8929	--	--	--	--	0.9
magic	ADAIN	0.9315	0.7137	--	--	--	--	0.8549
	IMORL	0.8404	0.7836	--	--	--	--	0.8205
waveform	ADAIN	0.7843	0.823	0.8193	--	--	--	0.8132
	IMORL	0.7575	0.8	0.8009	--	--	--	0.7814
sat	ADAIN	0.9602	0.9131	0.9169	0.4837	0.6417	0.8494	0.8387
	IMORL	0.9	0.8918	0.8566	0.5673	0.6841	0.7897	0.8079

We use the Hotelling's T-square statistic test, abbreviated as “ t -test”, to measure the statistical significance of the prediction accuracy between ADAIN and the approach in [49]. From Table 10, one can find that ADAIN can statistically outperform IMORL for all the simulation datasets.

Table 10 t -test for prediction accuracy

Data set	ADAIN		IMORL		$ Z $	Accept or reject H_0
	μ	σ	μ	σ		
spambase	0.9142	0.023	0.8999	0.0215	2.0312	Reject
magic	0.8549	0.0091	0.8205	0.0168	8.0576	Reject
waveform	0.8132	0.0102	0.7814	0.0117	9.1696	Reject
sat	0.8387	0.0493	0.8079	0.0351	2.2736	Reject

To have a more comprehensive analysis of the performance, we also employ Receiver Operating Characteristics (ROC) curve [72] to demonstrate the effectiveness of the proposed ADAIN framework. Fig. 12(a) and Fig. 12(b) represent the ROC curves for datasets “spambase” and “magic”, respectively.

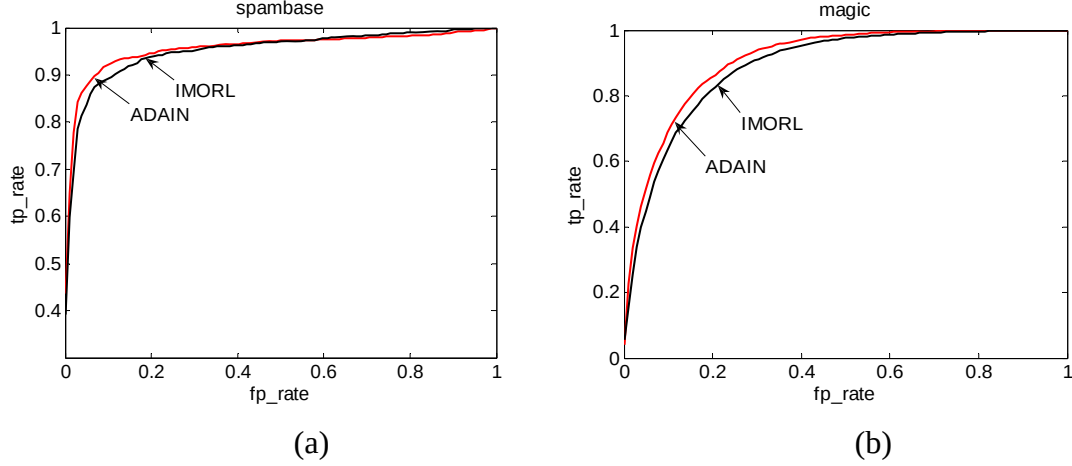


Fig. 12 ROC curves for selected simulation data sets

The Area Under ROC Curve (AUC) assessments as shown in Table 11 are also given for numerical understanding of the comparison between ADAIN and [49]. Note here in lieu of simply averaging the AUCs across the 20 runs, the averaged AUCs are derived by *vertical averaging* as suggested by [72]. For datasets with more than 2 classes, their averaged AUCs are calculated by summing the averaged AUC of the reference ROC curves weighted by the class ratio [75]. From Table 11, one can see that the proposed ADAIN framework is also very competitive against our previously proposed IMORL algorithm in terms of AUC evaluation metric.

Table 11 The averaged AUC **Error! Not a valid link.**

3.3 SERA – A Selectively Recursive Approach towards nonstationary imbalanced data classification

In this project period, we adopt the synthetic dataset to testify the effectiveness of our proposed algorithm. Despite the popularity of STAGGER [76] and the SEA [77] synthetic datasets in stream data mining, we use the synthetic dataset suggested in [61] which is more complex in classification boundary design and has more well-founded concept drifts mechanism embedded in. The generation of such synthetic dataset is shown as follows:

1. The classification boundary is designed as:

$$g(\mathbf{x}) = \sum_{i=1}^d a_i x_i x_{d-i+1} - a_0 \quad (31)$$

where x_i is the i th feature of instance $\mathbf{x}$, a_i is the i th feature coefficient assigned to x_i , and d is the number of dimensions of feature space. The class label of synthetic instances is decided by $\text{sgn}(g(\mathbf{x}))$.

2. The designed concept drifts occur both in the feature probability $p(\mathbf{x})$ and the conditional class label probability $p(y|\mathbf{x})$ [61], since the probability of target concept can be decomposed according to Bayes theory as follows:

$$p(y, \mathbf{x}) = p(y|\mathbf{x}) \cdot p(\mathbf{x}) \quad (32)$$

The concept drifts of $p(\mathbf{x})$ and $p(y|\mathbf{x})$ are reflected by the consistently varying mean of the features and the feature coefficients as $\mu_i s_i (1+t)$ and $a_i s_i (1+t)$, where μ_i and a_i are the i th feature of the feature mean and the i th feature coefficient initialized by a randomized real number between $[0, 1]$. To manipulate the concept drifts, at different timestamp, s_i is assigned an integer randomly alternating in $\{-1, 1\}$, and t is randomized as a real number in the interval $[0, 1]$.

Based on this mechanism, Fig. 13 gives some snapshots of the synthetic data sets over time. One can see that its feature distribution and the decision boundary are both varying over time.

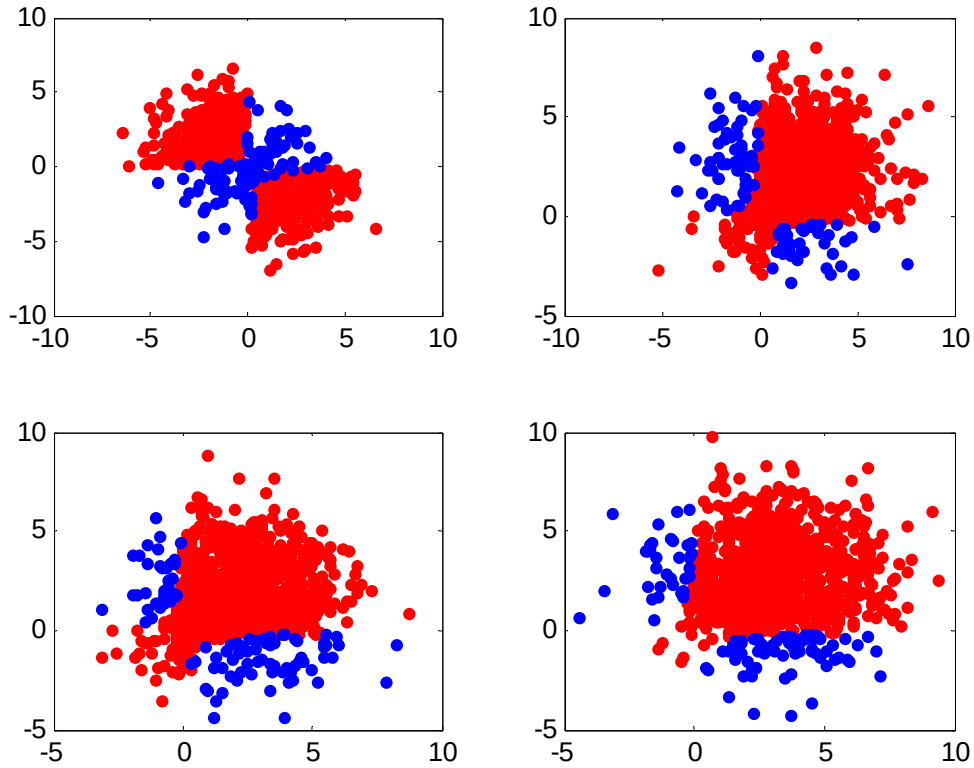


Fig. 13 Snapshots of the synthetic data set drifting over time

In our simulation, the number of the feature dimension is set to be 10, and the imbalanced ratio of each data chunk is set to be 1:100 globally.

-- Employ SMOTE [14] to balance the imbalanced ratio of current data chunk, and then apply a single classifier to learn from it, which is denoted as “SMOTE” in the display of simulation results.

-- Employ ADASYN [57] to balance the imbalanced ratio of current data chunk, and then apply a single classifier to learn from it, which is denoted as “ADASYN” in the display of simulation results.

-- Accommodate all previous minority examples to balance the imbalanced ratio of current data chunk, and then apply the “uncorrelated Bagging” [61] to learn from it, which is denoted as “uncorrelated Bagging” in the display of simulation results.

-- Selectively accommodate previous minority examples to balance the imbalanced ratio of current data chunk, and then apply one single classifier to learn from it, which is denoted as “Single” in the display of simulation results.

-- Selectively accommodate previous minority examples to balance the imbalanced ratio of current data chunk, and then apply BBagging with cost factor k being 1, 4 or 8 respectively, which are denoted as “BBagging(CF=1)”, “BBagging(CF=4)”, “BBagging(CF=8)” in the display of simulation results.

The uncorrelated Bagging cuts the majority data chunk into multiple distinct smaller parts to match the size of the set withholding all the minority examples accumulated insofar. When the number of accumulated minority examples is tantamount to or exceeds that of the majority examples, there is no need for uncorrelated Bagging to divide the majority data set, and thus only one hypothesis is generated; it cannot be regarded as an ensemble approach at this point.

We set the number of chunks in stream data to be 100. Each of the chunks carries 1000 examples for training purpose and 10000 instances for testing purpose. In our simulation, the post-balance ratio f is set to be 0.2 and 0.3 to observe its impact on the algorithm's performance. We install 3 observation points at chunks with timestamps $f \times 10$, 50, 80 to evaluate the performance trendline of the algorithms.

The neural network with multi perceptron (MLP) is used as the base classifier in our simulation. The number of hidden layer neurons is set to be 4. And the number of input neurons is set to be 10, i.e., the number of features of the synthetic dataset. Since there are two classes in our current simulation, the number of output neurons is set to be 2. Sigmoid function is used as the activation function, and the training epochs is set to be 100, with a learning rate being 0.1. For the BBagging, the number of iteration is set to be 10.

The evaluation metrics of the simulation results include *OA*, *Precision*, *Recall*, *F-measure* (when $\beta=1$) and *G-mean*, which are defined in equations (24) to (28). The simulation results when $f=0.2, 0.3$ at different observation points are given in Table 12 and Table 13, respectively.

Table 12 Simulation results on $f = 0.2$

$f = 0.2$						
Timestamp	Algorithm	OA	Precision	Recall	F-measure	G-mean
20	SMOTE	0.9843	0.087	0.06	0.071	0.2442
	ADASYN	0.9807	0.1092	0.13	0.1187	0.3586
	uncorrelated Bagging	0.9623	0.089	0.3	0.1373	0.5392
	Single	0.9623	0.089	0.3	0.1373	0.5393
	Bbagging(CF=1)	0.9688	0.0923	0.24	0.1333	0.484
	Bbagging(CF=4)	0.8981	0.0594	0.62	0.1085	0.7474
	Bbagging(CF=8)	0.8474	0.0499	0.79	0.0938	0.8185
50	SMOTE	0.9425	0.0297	0.15	0.0496	0.3776
	ADASYN	0.951	0.0267	0.11	0.043	0.3249
	uncorrelated Bagging	0.8778	0.0483	0.6	0.0894	0.7269
	Single	0.8811	0.0421	0.5	0.0776	0.6652
	Bbagging(CF=1)	0.8953	0.0537	0.57	0.0982	0.7157
	Bbagging(CF=4)	0.8416	0.0327	0.52	0.0616	0.6628
	Bbagging(CF=8)	0.7941	0.032	0.67	0.0611	0.73
80	SMOTE	0.984	0.1739	0.16	0.1667	0.3985
	ADASYN	0.9832	0.1304	0.12	0.125	0.345
	uncorrelated Bagging	0.9691	0.1057	0.28	0.1534	0.5228
	Single	0.9682	0.1135	0.32	0.1675	0.5585
	Bbagging(CF=1)	0.9746	0.1468	0.32	0.2013	0.5603
	Bbagging(CF=4)	0.9545	0.0993	0.44	0.1621	0.6498
	Bbagging(CF=8)	0.9428	0.0816	0.46	0.1386	0.6603

Table 13 Simulation results on $f = 0.3$

$f = 0.3$						
Timestamp	Algorithm	OA	Precision	Recall	F-measure	G-mean
20	SMOTE	0.9772	0.0429	0.06	0.05	0.2433
	ADASYN	0.9815	0.043	0.04	0.0415	0.1991
	uncorrelated Bagging	0.9153	0.058	0.49	0.1037	0.6712
	Single	0.9153	0.058	0.49	0.1037	0.6712
	Bbagging(CF=1)	0.9183	0.0717	0.6	0.1281	0.7436
	Bbagging(CF=4)	0.7835	0.0355	0.79	0.068	0.7867
	Bbagging(CF=8)	0.7024	0.0282	0.86	0.0546	0.7763

50	SMOTE	0.9445	0.0672	0.2	0.0672	0.4364
	ADASYN	0.9331	0.0404	0.25	0.0695	0.4848
	uncorrelated Bagging	0.821	0.0388	0.71	0.0735	0.7027
	Single	0.8511	0.0385	0.58	0.0723	0.7037
	Bbagging(CF=1)	0.8575	0.0484	0.71	0.0906	0.7809
	Bbagging(CF=4)	0.7361	0.0268	0.72	0.0517	0.7281
	Bbagging(CF=8)	0.6649	0.0226	0.77	0.0439	0.715
80	SMOTE	0.9839	0.1039	0.08	0.0904	0.2819
	ADASYN	0.9847	0.1045	0.07	0.0838	0.2638
	uncorrelated Bagging	0.9501	0.1018	0.51	0.1697	0.6977
	Single	0.9565	0.0905	0.37	0.1454	0.5967
	Bbagging(CF=1)	0.9668	0.1209	0.37	0.1823	0.6
	Bbagging(CF=4)	0.9335	0.0739	0.5	0.1284	0.6779
	Bbagging(CF=8)	0.9144	0.0605	0.52	0.1083	0.6911

As can be seen from Table 12 and Table 13, the over-sampling techniques are always best at “OA”. This is possibly because the synthetic instances are always generated by interpolating between the two spatially nearby minority examples that they don't risk so much as to undermine non-trivially the target concepts. Yet since the synthetic instances are now drawn from the same distribution as the real minority examples, they also cannot help much to improve the prediction performance on minority examples (significantly lower “*Recall*” than all other comparative algorithms). Furthermore, their overall performance is still below a fixed prediction, i.e., predict all instances belong to majority examples so as to achieve 99% “OA”. Comparing other metrics, our proposed algorithms (Single, BBagging(CF=1, 4, or 8)) demonstrate an improvement from $f = 0.2$ to $f = 0.3$ (nearly outperform the three other algorithms for comparison in all metrics except “OA”). Also note that our proposed algorithms perform remarkably better than other algorithms in “*Recall*” which represents the number of correctly predicted minority instances. This fact shows the superiority of our algorithms on learning the target concept of the minority examples.

Fig. 14 demonstrates the ROC curves on the 6 observation points, where Fig. 14(a), 11(b) and 14(c) correspond to the timestamp 20, 50 and 80 when $f = 0.2$; Fig. 14(d), 11(e) and 14(f) correspond to the timestamp 30, 50 and 80 when $f=0.3$. As can be seen from the ROC curves, the over-sampling techniques behave terribly worse than the other algorithms, which is consistent with our previous discussion.

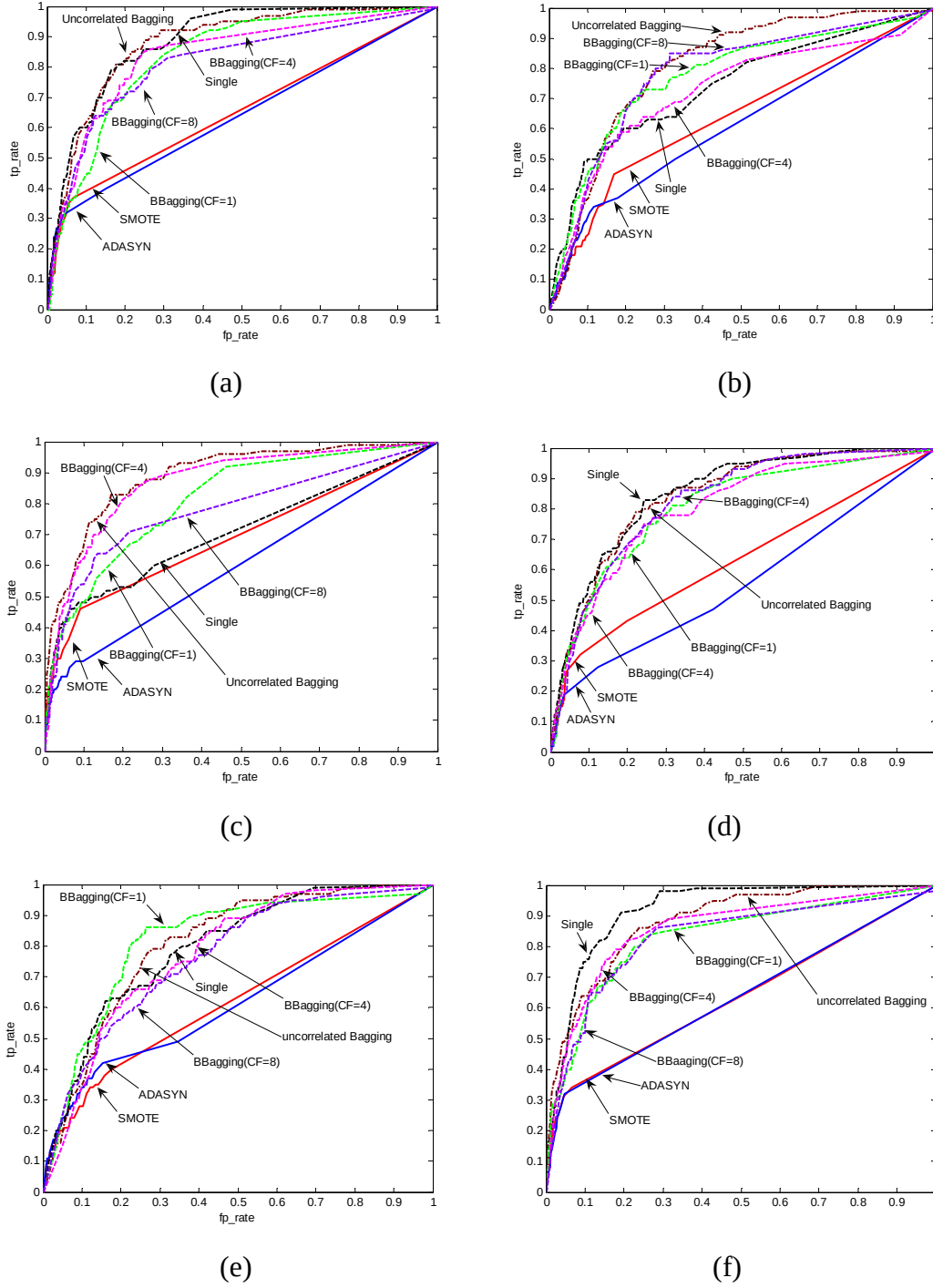


Fig. 14 ROC curves on the observation points in simulation

BBagging performs worse than uncorrelated Bagging when $f = 0.2$, however the “Single” classifier performs obviously better than uncorrelated Bagging. Both BBagging and “Single” classifier perform generally better than uncorrelated Bagging when $f = 0.3$. This is probably because more accommodated previous minority examples give rise to a

performance improvement. Also it is noted that the “Single” classifier outperforms all other algorithms in the observation point with timestamp 80 when $f = 0.3$. This result justifies our claim that ensemble approach is not necessarily better than a single hypothesis discussed previously.

3.4 Detection of Network Intrusions

3.4.1 System Model And Dataset

In this project period, we evaluate our algorithm with the real-world data collected from the integrated network based Ohio University’s network detective service (INBOUNDS) system. This system is a real-time based network IDS developed at Ohio University. We suggest interested readers refer to [105] for a detailed description of the system. Briefly speaking, the INBOUNDS system can be statistically described by six random variables. The parameters include: (1) INTER that describes the interactivity and defines the number of questions per second during a particular period; (2) ASOQ that is the average size of questions; (3) ASOA that is the average size of answer; (4) LQAIT that is log (base 10) of question-answer idle time (in seconds) that the server takes before responding to a question; (5) LAQIT that is log (base 10) of answer-question idle time that the client takes to ask another question; and (6) DOC that is the duration of connection (in seconds). All features are measured in a particular period T_M that is tunable and is set to 60 seconds for all experiments in this project.

We collected 7194 data samples in our current simulation. Table 14 summarizes several important statistics of the data, and Fig. 15 illustrates the histograms of the six features.

Table 14 Statistics of the Dataset

	Max	Min	Mean	Median	Standard Deviation
INTER	17	0	0.8295	1	0.7727
ASOQ	32120	0	589.1202	416	743.9734
ASOA	4344600	0	6802.3	1014	59464
LQAIT	0.729	-10	-1.383	-1.0850	0.8743
LAQIT	1.397	-10	-3.7143	-3.3605	3.3241
DOC	558.58	0	9.4632	0.4930	27.2444

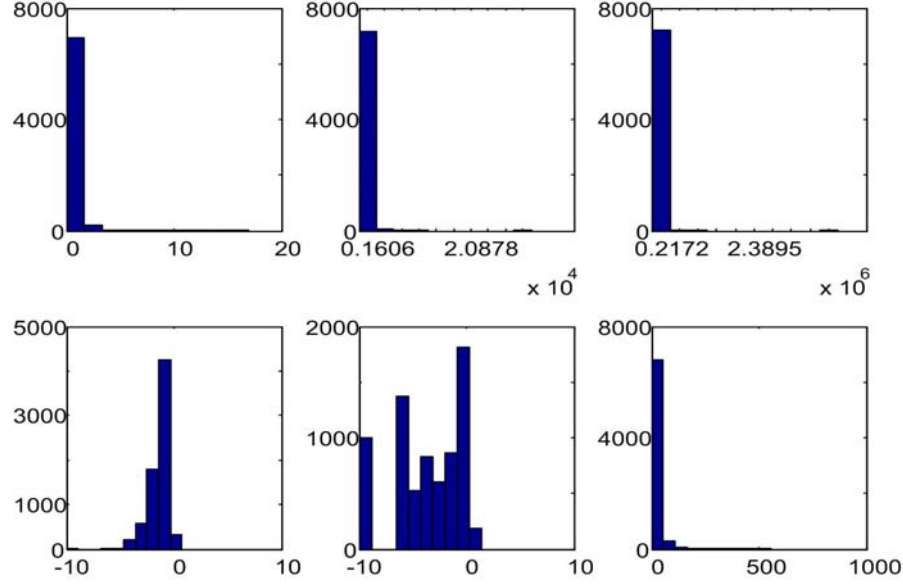


Fig. 15 Histograms of the features.

3.4.2 Simulation Results

We use the parameter settings for SOM and intrusion detection as follows. The number of neurons ℓ is 200, the initial learning rate η_0 is set to 3, the initial width of topological neighborhood σ_0 is set to 25, and the time constants in cooperative phase and adaptive phase are $\frac{1000}{\log \sigma_0} = 310.6675$ and 1000, respectively. The number of iteration of training is set to 300 times the number of neurons, *i.e.*, $T = 300 \cdot \ell = 60000$. The width of interval used for searching the cutting limits ξ is calculated as $\xi = \frac{\max(X) - \min(X)}{N_s}$, where N_s is a tunable parameter. Too small N_s will lead to less accuracy, whereas too large N_s will increase the computational load of the system. Here, we set N_s to 5×10^5 . The simulations are conducted using an Intel Duo Core CPU with 2.0GHz and 2 GB RAM memory under the MATLAB version 7.4.0.287 (R2007a) environment.

In Table 15, we present the experimental results using both methods, the traditional KDE and the KDE based on SOM, when the cutting probabilities α and β are both set to 0.25%, *i.e.*, the confidence level is 99.5%. For each random variable, the simulation results of the proposed algorithm are benchmarked against the traditional KDE. The first column of results presents the total time consumed by both methods. We further decompose the total computational time into the second and third columns, the time for SOM preprocessing that can be considered as an overhead and the time for KDE operation. The fourth and the sixth columns are the obtained upper limit and lower limit, *i.e.*, λ and μ , respectively. The fifth and the rightmost columns present the number of intrusions detected in the upper and the lower outlier regions, respectively. As can be noticed from table 15, the proposed algorithm can dramatically reduce the computational cost compared with the traditional KDE methods, while maintaining its detection accuracy. Fig. 16 shows the intrusions detected in the input data with respect to each random variable. The dotted

lines show the estimated pdfs for the input samples, and those that are surrounded by circles are the detected intrusions by our method.

We would also like to point out that different cutting probabilities may result different intrusion detection results. In others word, the cutting probabilities can be considered as the security level that reflect the risk tolerance of the IDS system. Large cutting probabilities indicates that the system is more risk averse, and attempts to reduce type II errors, or false negative errors by capturing all potential intrusions. But it may trigger much more frequent false alerts and report intrusions when in reality they are not. Moreover, large cutting probabilities may increase the computational load of the IDS system. On the contrary, small α and β indicates that the system is more speed seeking. This may reduce type I errors, or false positive errors, but it may not be able to capture someone misrepresenting a network activity that is intended to be malicious or intentionally harmful.

Table 15 Simulation results of intrusion detection when $\alpha = \beta = 0.25\%$

		Time (Total)	Time (SOM)	Time (KDE)	Upper Limit λ	# Intrusions in Θ_{upper}	Lower Limit μ	# Intrusions in Θ_{lower}
INTER	KDE	408.11	0	408.11	7.2529	18	0	0
	KDE +SOM	60.19	29.41	30.78	7.7431	16	0	0
ASOQ	KDE	396.53	0	396.53	3841.9	18	52.0487	15
	KDE +SOM	62.03	28.92	33.11	4061.4	14	51.856	15
ASOA	KDE	361.45	0	361.45	356330	18	0	0
	KDE +SOM	60.26	28.75	31.51	376110	14	0	0
LQAIT	KDE	549.33	0	549.33	0.1231	8	-4.921	17
	KDE +SOM	70.90	28.88	42.02	0.1165	8	-4.8797	20
LAQIT	KDE	1749.5	0	1749.5	1.397	0	-10	0
	KDE +SOM	126.66	29.10	97.55	1.397	0	-10	0
DOC	KDE	293.96	0	293.96	208.5155	18	0	0
	KDE +SOM	61.69	28.81	32.88	226.7989	17	0	0

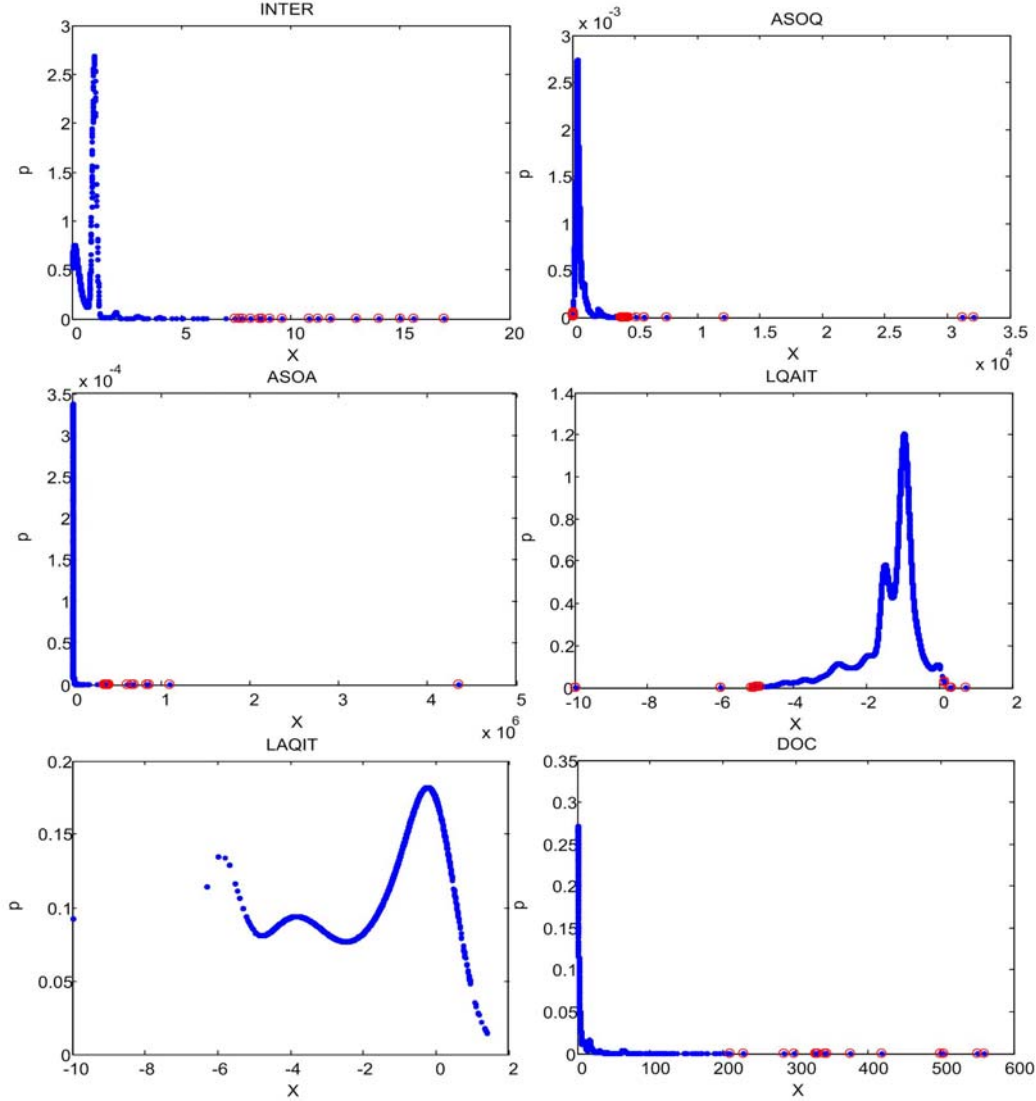


Fig. 16 Intrusions detected based on KDE and SOM by six observed features

4. FPGA-based Reconfigurable Platform for Video and Image Processing

In this project period, we have also developed a general-purpose, multi-task, and reconfigurable platform for video and image processing [109]. Generally speaking, a complex video application requires simultaneous data processing among different modules. The highly parallel data operation characteristic of FPGA provides a unique advantage of its application for such a purpose. According to different application requirements and specifications, different categories of FPGA chips can be used. In our current design, we use a low-cost high-end FPGA product, the Virtex-II Pro family (XC2VP30) as the prototype platform. Fabricated in 0.13 μ m process technology, the Virtex-II Pro family provides a good platform to meet different design requirements. For instance, the XC2VP30 FPGA includes dual Power-PC cores, over thirty thousand logic

elements and 2Mbits embedded RAM [110]. Compared to the conventional DSP based design, the XC2VP30 FPGA can efficiently implement the multiply and accumulate (MAC) operations in parallel, and the behavior of each processor or peripheral core can be customized. Fig. 2 provides a system level architecture of the proposed video processing platform.

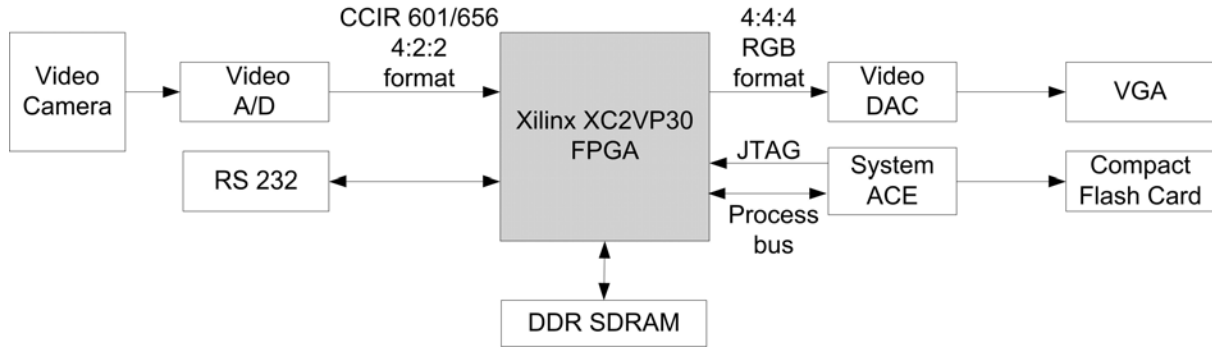


Fig. 17 The proposed system level architecture

In our system, a video analogue to digital conversion (ADC) board is used to capture the national television system committee (NTSC) signal and digitize it into CCIR 601/656 format. The architecture in Fig. 17 provides the flexibility of implementing different functional modules for video and image processing. In our current design, we have implemented three processing functions: zoom-in, zoom-out and edge-detection. Fig. 18 shows the data processing flow of the proposed system. One can easily extend this architecture to include more modules, or to test their own design concepts and algorithms based on this platform.

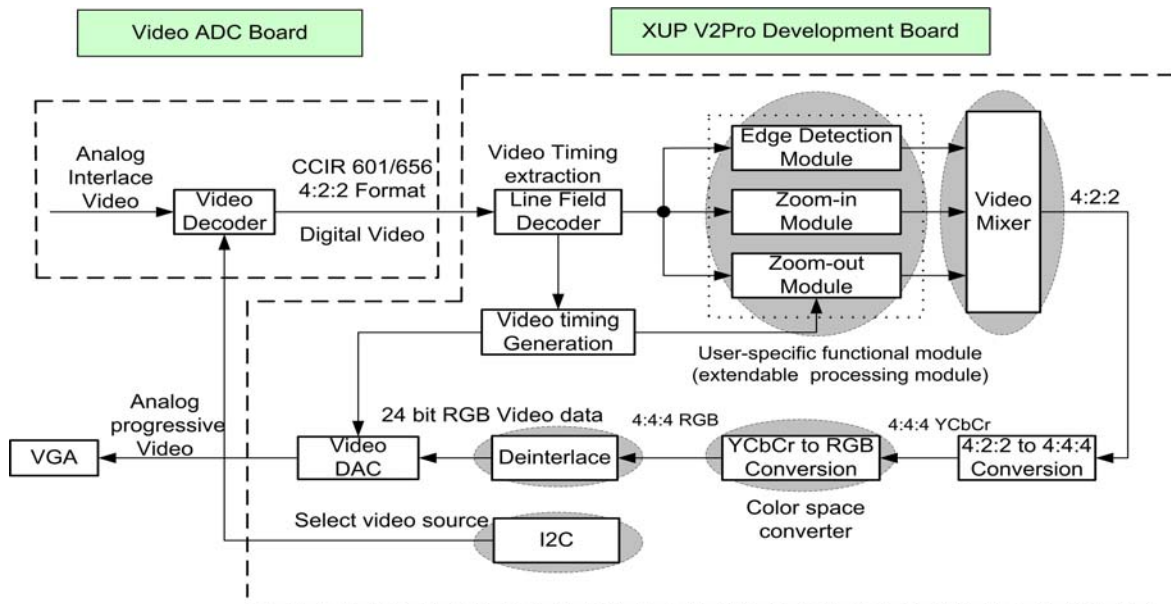


Fig. 18 Data processing flow of the proposed system

From Fig. 18 one can see, the FPGA implementation of the proposed system includes five major functional modules (the highlighted ellipses): the user-specific functional modules, the video mixer module, the color space converter module, the de-interlace module and the inter-integrated circuit (I2C) configuration module. The user-specific functional module implements most of the functionalities according to different video processing applications. This functional box can be extended in different application scenarios. The video mixer module can mix different video layers by the Alpha blending mixer function. This module supports both the picture-in-picture mixing and image blending. Each video layer can be independently displayed at running time. The color space converter module transforms the incoming video data between color spaces, which are specified by three coordinate values. This module supports the pre-defined conversions between standard color spaces, and allows user-specified coefficients to translate between any two three-valued color spaces. Interlaced video is commonly used in television standards such as phase alternation line (PAL) and NTSC. However, progressive video is required for LCD displays. Therefore, the de-interlace module converts interlaced video to progressive video. We use the embedded PowerPC405 microprocessor to achieve the I2C configuration function by programming the operational model of the analog device, the ADV7183B video decoder on the daughter card.

From Fig. 18 one can see, one of the advantages of the proposed system is that it provides an extendable module to implement different functionalities according to different application requirements. This provides the flexibility of using this system as a general-purpose video and image processing platform across different application domains. In our current research, we implement the edge detection and scaling (zoom-in and zoom-out) functions, which are important procedures in many complex video processing applications.

4.1 Four-direction Edge Detection

Edge detection is a fundamental and critical technique in most image processing applications to obtain useful information before feature extraction and object segmentation. This process detects outlines of an object and boundaries between objects and the background. In this research, we implement the four-direction Sobel operator [111] for edge detection. The detection resolutions and filter coefficients can be dynamically changed during the running time.

Generally speaking, the Sobel operator is based on a two-dimensional spatial gradient measurement on an image to detect the edges. This is implemented by calculating the convolutions of the image with a filter mask (convolution kernel) to calculate the approximate gradient magnitude [111]. Typically, the convolution kernel is moved pixel-by-pixel and line-by-line across the image, which can be defined as:

$$h[i, j] = f[i, j] * g[i, j] = \sum_{k=0}^{n-1} \sum_{l=0}^{m-1} f[k, l] g[i - k, j - l] \quad (33)$$

Where $g(i, j)$ represents the convolution kernel, n and m is the size of the convolution kernel at two dimensions, and $f(i, j)$ and $h(i, j)$ represents the original and filtered image, respectively.

A 3 by 3 kernel is used in our design to produce the map of intensity gradients. This is implemented by using the four-direction gradients calculated by convolving the source video frame with the four-direction kernels. Fig. 19 illustrates this idea.

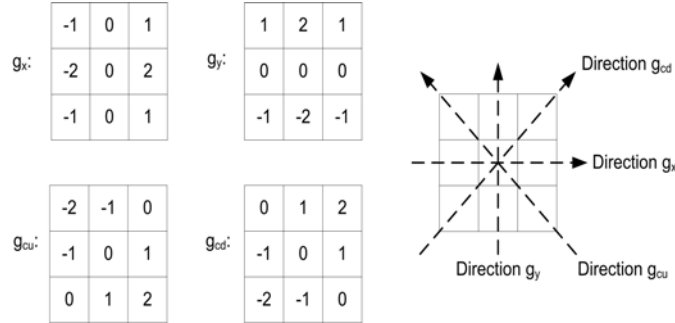


Fig. 19 Four-direction edge detection

In order to implement this four-direction edge detection, a generic 2-D image filter is proposed in Fig. 20. In this design, two line buffers and six registers are used to store the data flow and provide access to the neighborhood pixels. The incoming pixels are shifted through line buffers to create a delay line, which are sent to the filter array simultaneously with pixels from all the relevant video lines. At each filter node, the pixel is multiplied with the appropriate filter coefficients as indicated in Fig. 19. All the multiplier results are added together at the adder tree to produce the filter middle point output result. From Fig. 20 one can see, four additions and nine multiplications are needed to calculate the output value of the convolution.

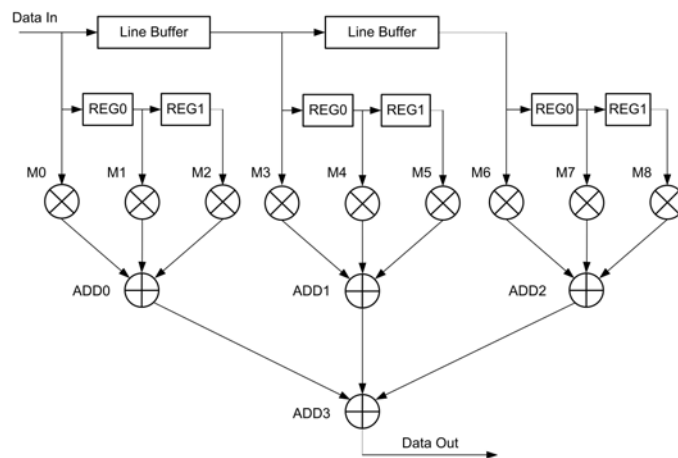


Fig. 20 Design of the four-direction edge detection

4.2 Scaling Functionalities: Image Zoom-in and Zoom-out

Scaling is another widely used technique in many video processing applications. In this research, we implement the zoom-in and zoom-out functions in the extendable functional module.

As far as the zoom-in function is concerned, there are several popular algorithms such as nearest neighbor method and bilinear interpolation method [112] [113]. Our current design supports both methods and can be configured to change resolutions and/or filter coefficients at running time. As an example, Fig. 21 gives a detailed design architecture of the bilinear interpolation method. Without loss of generality, we assume the upscale factor is two and one needs to zoom in as four times as the original image. Fig. 21 illustrates the method that is used to generate new pixels and new lines of the image. First, new pixels between line n and line $n+1$ are generated with a combination factor of $1/2$. Then, new pixels between the two vertical pixel lines are created. In our design, two video frame buffers are used: one is used to store the luminance signals and the other one is used to store the chroma signals.

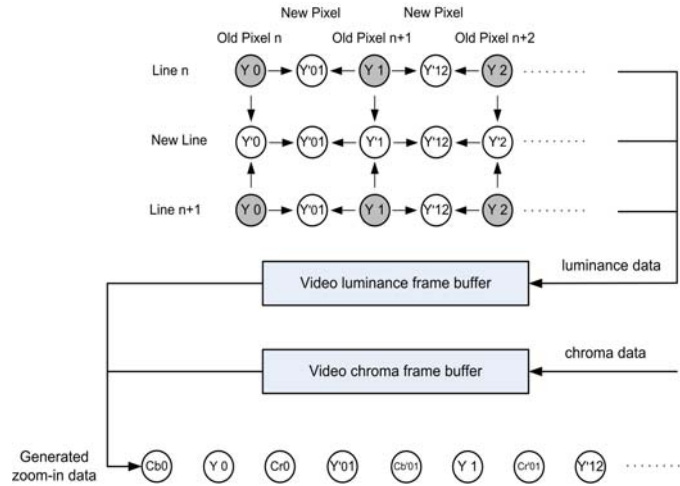


Fig. 21 Design of the zoom-in function for video processing

Fig. 22 illustrates the idea of implementing the zoom-out function. In order to eliminate the frequency mixing effect, the incoming images are first passed through a low-pass filter. The new pixels are then calculated by bilinear interpolation method. Assuming the zoom-out image is a quarter of the original image, Fig. 22 illustrates the data flow to implement this, where C_b and C_r represent video chroma data, and Y represents video luminance data. Since a digitalized NTSC video line includes 720 pixels, we need to calculate 360 Y signals and 180 C_b and C_r signals per line. Furthermore, in order to fulfill the zoom-out output timing, a CIF frame buffer is used to store the generated new pixels.

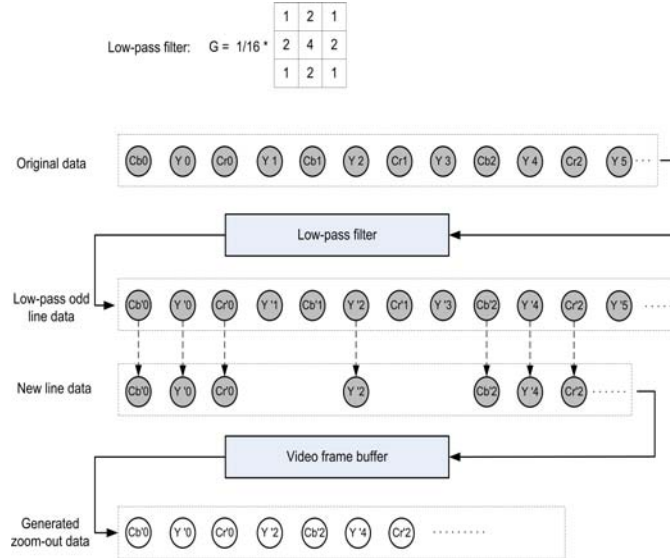


Fig. 22 Design of the Zoom-out function for video images

4.3 System Implementation and Experimental results

4.3.1 System Development

We implement the entire platform based on the Xilinx Virtex-II Pro development system [114]. Fig. 23 shows the hardware platform with major components. The on board XC2VP30 FPGA chip has about 30,816 logic cells, 136 18-bit multipliers, 2,448Kb of block RAM, and two PowerPC Processors. The DDR SDRAM DIMM can support up to 2Gbytes of RAM. This board also has many useful interface ports, such as the 10/100 Ethernet port, compact flash card slot, XSGA video port, RS-232 port, and others. It also has various expansion connectors to expand the usability of this board to meet the requirements of different video and image processing applications. Our major purpose of this system is to implement the entire hardware platform to provide a general solution for video and image processing, and demonstrate its effectiveness through various application scenarios.

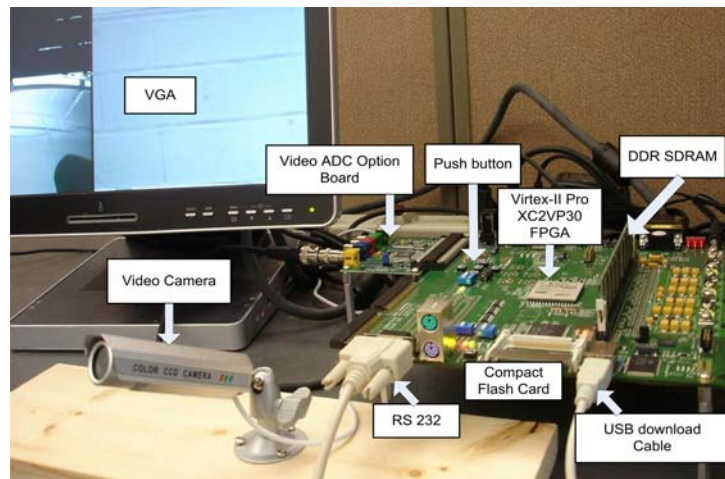


Fig. 23 The proposed FPGA platform.

In our design, we use four embedded block buffers to store the video data. Buffer1 and buffer3 are used to store odd field data, and buffer2 and buffer4 are used to store even field data. This arrangement can avoid odd field data displaying on even field. The detailed architecture is shown in Fig. 24.

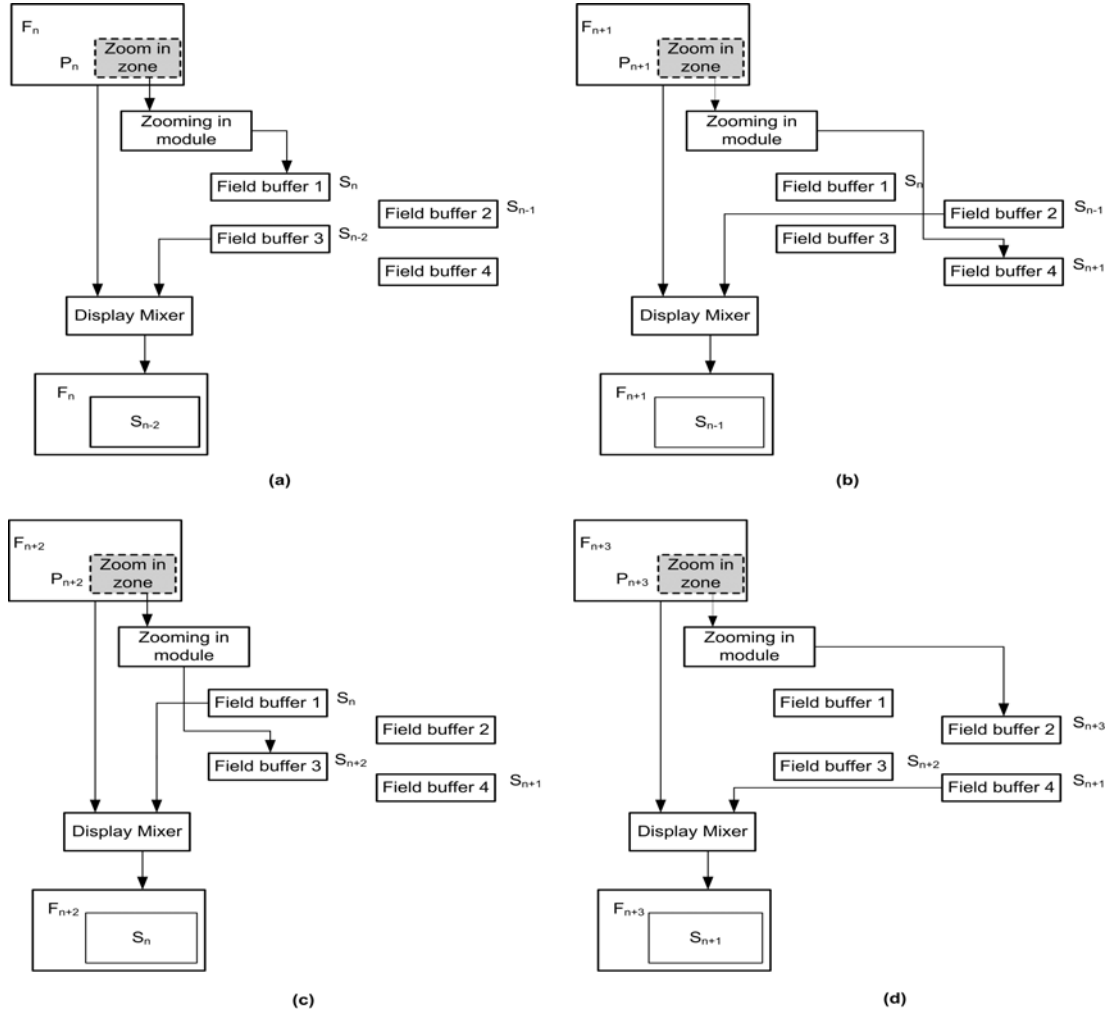


Fig. 24 The ping-pong architecture for video data processing

Considering the zoom-in module as an example, when the system processes the odd field data F_n , its output S_n will be written into buffer1. At the same time, the previously processed block S_{n-2} is mixed with the current frame F_n to generate the final output stream. The input-output order of these two buffers will change when the next odd field is presented. Symmetrically, when the even field F_{n+1} is presented, buffer 4 stores its output S_{n+1} and buffer 2 outputs the previously processed block S_{n-1} . The detailed timing and buffer operation diagram is illustrated in Fig. 25. This ping-pong architecture provides an

efficient way to avoid two different operations affecting the same buffer simultaneously. In order to increase the embedded memory resource for such operations, one can use the extended memory with the external DDR-SDRAM provided by the Virtex-II board.

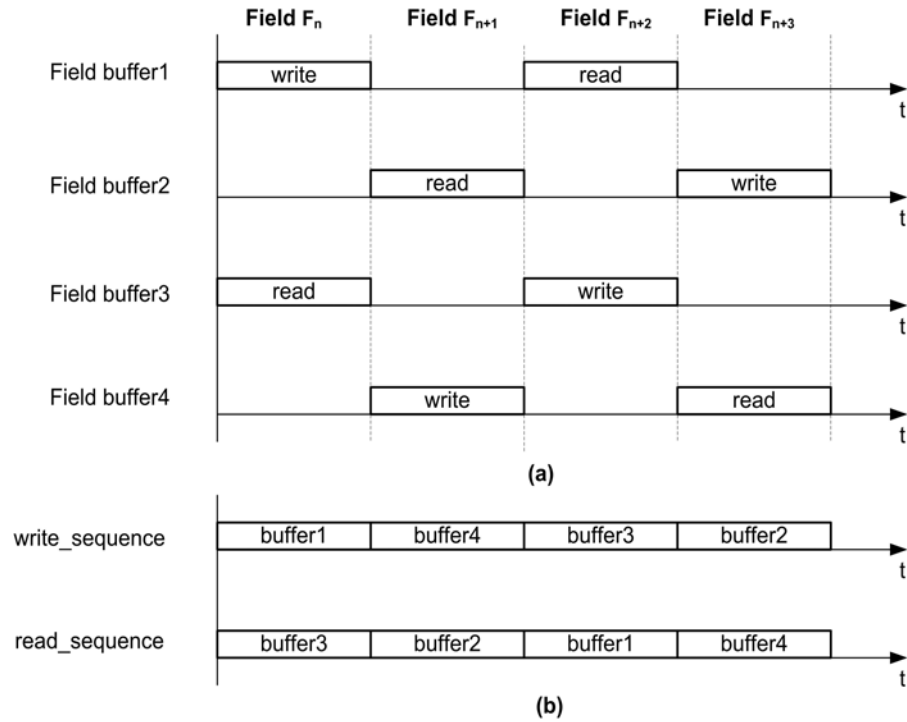


Fig. 25 The timing and buffer operation diagram

To verify the timing and logic functions, the entire system is simulated by the Xilinx Integrated Software Environment (ISE 9.1i) toolsets for extensive simulation and logic analysis. Fig. 26 shows a snapshot of the system logic and timing simulation results. The system operation clock is 27MHz (the clk_27 signal). The pcount signal counts the number of line pixels, and the firstline_data, secondline_data and thirdline_data represent the input video data of three lines. We operate the line buffer through fifo_wen and fifo_ren signal, which generates the background signal (background7) by delaying proper number of clocks from the original input video stream. By mixing the background signal and the processed video data (the f_data signal), we can get the final output signal (the SDI_O signal). From Fig. 26 one can see that a total of 10 clocks processing time is needed for one pixel operation.

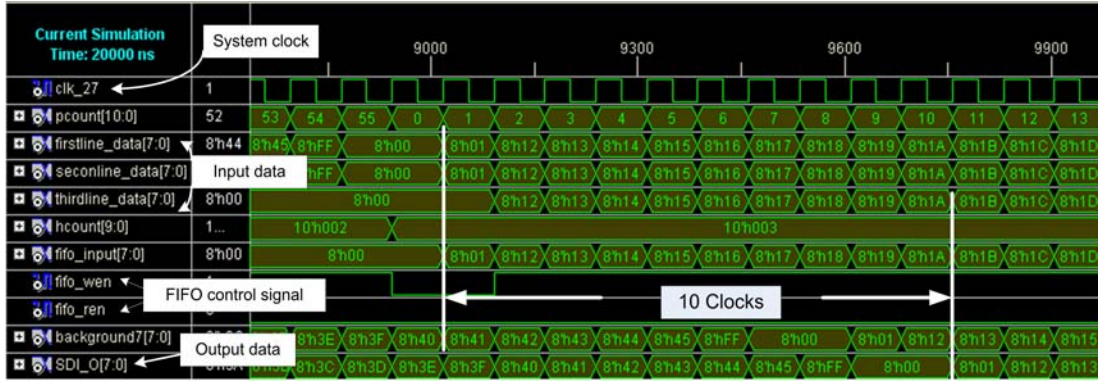


Fig. 26 A snapshot of the system logical simulation

The synthesized RTL level circuitry of the entire system is shown in Fig. 27, which includes the major components of the video processing module (edge detection, zoom-in and zoom-out), the two line buffers (see Fig. 20 for details), the color space converter, timing generation modules, and others. The final implementation includes a total NAND-equivalent gate counts of 5,154,734. Table 16 summarizes the major resource utilization characteristics of the final system, from which one can see the final system utilizes about 20% of logic resource, 50% of memory on chip, and has total power consumption around 203mw.

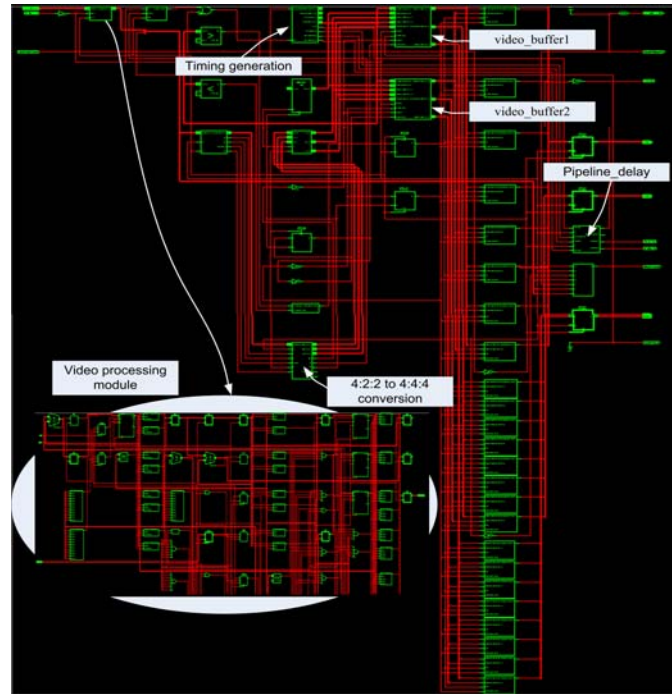


Fig. 27 Synthesized RTL level circuitry of the entire system

Table 16 Resource utilization of the entire system

Hardware resource	Availab	Us	Utilizati
-------------------	---------	----	-----------

	le	ed	on
Number of occupied Slices	13696	2869	20%
Total Number of 4 input LUTs	27392	5293	19%
Number of bonded IOBs	556	42	7%
Number of PPC405s	2	1	50%
Number of Block RAMs	136	70	51%
Number of MULT18X18s	136	5	3%
Number of GCLKs	16	2	12%

4.3.2 Simulation and Experimental Results

In this section, we demonstrate the effectiveness of the hardware system for different video processing applications. In the first experiment, we use the camera system to capture image data from different environments. Fig. 28 (a) shows the original image and Fig. 28 (b) illustrates the effects of the edge detection function. Fig. 28 (c) demonstrates all the functional modules in the same window, including the edge detection, zoom-in and zoom-out. All these functions can be controlled easily by the push buttons on the FPGA board (see Fig. 23 for system details).

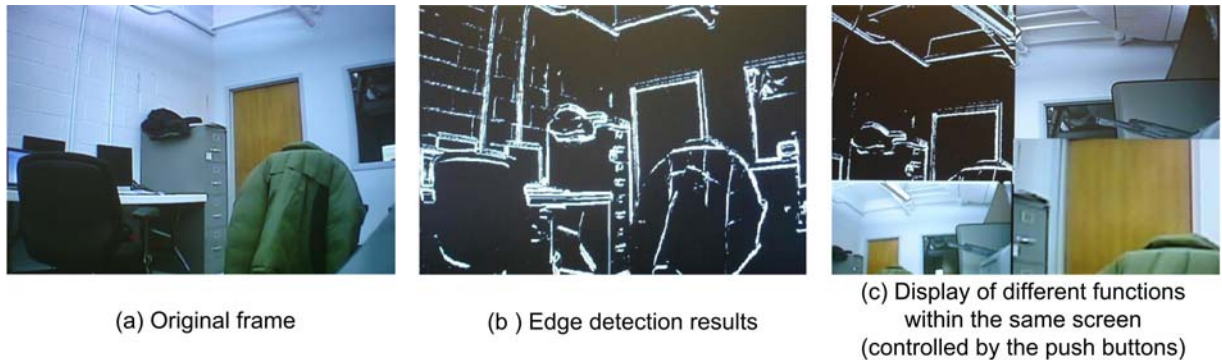


Fig. 28 System performance based on the camera input data

In the second experiment, we use a DVD player to provide the input video data for processing. Fig. 29 shows a snapshot of the processing results. As one can see, the proposed FPGA-based system can provide effective results in this situation.



Fig. 29 System performance based on the video data

5. Potential Applications

During this project period, we have developed various computational intelligent algorithms and models for data mining, tested their applications across different data sets, and evaluated their performances. Here we will briefly summarize the major research results and identify their potential applications.

In this project, we have proposed a ranked minority oversampling approach in boosting for imbalanced learning problems. The key characteristics of RAMOBoost are *adaptive learning* and *reduction of bias*. This is accomplished by adaptively shifting the decision boundary towards those difficult examples in both minority and majority examples, and systematically creating minority synthetic instances based on the distribution function. Simulation results on 16 datasets across various assessment metrics, including *OA*, *Precision*, *Recall*, *F-measure*, *G-mean*, and ROC curves, demonstrate the effectiveness of the proposed method.

As a new method to the imbalanced learning problems, there are several interesting future research directions. For instance, RAMOBoost in our current study is focused on handling the datasets with continuous features. It can be extended to deal with the datasets with nominal features by adopting the SMOTE-N method in [14]. Second, RAMOBoost in our current simulation is only evaluated on two-class imbalanced problems. It can be generalized to handle multi-class imbalanced learning problems to improve its applicability in practice. Finally, similar to many of the existing imbalanced learning algorithms, there are several parameters needed to be decided for RAMOBoost. We have shown some empirical results regarding this issue in this article, and we also would like to note that a systematical and adaptive way to adjust those parameters could be a challenging while important issue for this method to be applied across different application domains. Our group is currently investigating all these issues. Motivated by our initial results in this article, we believe that RAMOBoost may provide new insights to

the imbalanced learning problems, and have the potential to be a powerful tool in many application domains.

Based on the adaptive learning and ensemble learning methodology, the proposed adaptive incremental learning framework can automatically learn from data flow, accumulating experience, and use such knowledge to facilitate future learning and decision-making processes. We presented the system-level architecture and learning algorithm in detail. The effectiveness of our proposed approach is empirically verified on 4 real-world datasets in terms of prediction accuracy, ROC curve and the related AUC. The statistical test also validates the effectiveness of this framework.

There are a number of interesting topics for future research. For instance, it would be interesting to analyze the theoretical influence of the mapping function choice for the proposed method, and, if possible, to come up with some more effective approaches, e.g., density estimation, to further improve our incremental learning framework. Besides, in the incremental learning scenarios, it is not uncommon that new concepts may be unexpectedly introduced during the learning life, i.e., the concept drifting/shifting issue. The learning capability and characteristic of the proposed framework to adaptively adjust to such new concepts could be an important topic for future research. Motivated by the results in this article, we believe that the ADAIN framework may not only provide critical new insights into the adaptive incremental learning field, provide a powerful technique for different incremental learning applications, but it will also inspire and motivate future research opportunities in the community on this subject.

The proposed selectively ranked approach we proposed in this project period is aimed to learning from the imbalanced stream data with drifting target concepts. We argue that the accommodation of previously coming minority examples into the current training process is much more efficient and effective than the conventional stream data mining methods to learn from the imbalanced data stream. Rather than accepting all previously minority examples as existing approach, we propose to limit the number of accepted previous minority examples proportional to the size of the current majority set. The priority order of acceptance is decided by Mahalanobis distance. We also propose the BBagging to improve the learning performance on the minority instances by proportionally increasing their sampling weights. The simulation results show that our proposed algorithm is competitive with other algorithms and can significantly improve the prediction accuracy of the minority instances.

The proposed SERA framework is promising, and we have located several potential improvements path for this idea in the future. Our current simulation is based on a single run, and thus the statistical significance of SERA cannot be empirically proved at this point. In order to statistically justify the effectiveness of SERA, simulations on many trials are what we should explore in the future research. Besides, given the very limited volume of minority examples within the data chunk, better mechanisms, e.g. density estimation, need to be developed to measure the similarity between data chunks with different timestamps. Finally, there still exist some uncertainties about the post-balance ratio and the cost factor for BBagging in this work, which both require more adaptive and systematic mechanism to find the optimum values for them.

We also present an anomaly-based network intrusion detection algorithm based on KDE and SOM. We employ the KDE technique to estimate the probability density functions for the random variables used to describe an anomaly-based IDS and determine whether the network activities are normal or abnormal. However, huge volume of the observed data and the high computational cost of KDE can limit its usage in real-world applications. Therefore, we explore the learning and clustering capabilities of SOM, and use it to generate an approximation of the distribution of the input space in a compact manner. In this way, the number of kernels used in a kernel density estimator can be significantly reduced, and thus improve the efficiency for the intrusion detection. Our current algorithm focuses on univariate analysis. Therefore, in our future work, this algorithm will be extended to multivariate analysis of the network input parameters. Meanwhile, extensive experiments and simulations will be conducted over more real-world network intrusion data sets to evaluate the performance of the proposed algorithm.

Finally in this project period, we propose a FPGA-based prototype system for general purpose and multi-task video and image processing. System level hardware architecture and detailed design strategies are presented. The final system is implemented using the Xilinx Virtex-II Pro development system with an onboard XC2VP30 FPGA chip. Synthesized results indicate the overall system utilizes only about 20% of logic resource, 50% of memory on chip, and has total power consumption around 203 mw. This system provides a scalable and real-time reconfigurable platform to meet the requirements for many video processing applications. Furthermore, the reconfigurable and extendable characteristics of this system allow it to be easily modified to embed into different video and image processing scenarios. The effectiveness of the proposed prototype has been demonstrated by various experimental results.

In the future work, it would be interesting to integrate more complicated video processing modules into this platform. For instance, based on the edge detection function implemented in this research, it will be useful to implement a robust objects recognition algorithm into this system. In addition, since machine learning techniques have been extensively used for video and image processing, it would be interesting to develop various learning algorithms based on this prototype. For instance, we are currently designing a FPGA-based incremental learning system for video applications. The key idea is to develop an incremental learning architecture in hardware to learn and accumulate knowledge for multiple objects recognition and localization. Motivated by our research in this paper, we believe that such a FPGA-based system will provide a power platform for many real-world video and image processing applications.

Reference

- [1] F. Provost, "Learning with imbalanced data sets 101," in Learning from Imbalanced Data Sets, Papers from the AAAI Workshop, N. Japkowicz, Ed., Menlo Park, CA, 2000, technical Report WS-00-05.
- [2] N. Japkowicz, "(ed.)," Learning from Imbalanced Data Sets: Papers from the AAAI Workshop, 2000, technical Report WS-00-05.

- [3] N. V. Chawla, N. Japkowicz, and A. Kołcz, “(ed.),” in Proc. 12th Int. Conf. Machine Learning, Workshop on Learning from Imbalanced Data Sets II, 2003.
- [4] N. Chawla, N. Japkowicz, and A. Kołcz, “Editorial: special issue on learning from imbalanced data sets,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 1–6, 2004.
- [5] F. Provost and T. Fawcett, “Robust classification for imprecise environments,” Machine Learning, vol. 42, no. 3, pp. 203–231, 2001.
- [6] S. Clearwater and E. Stern, “A rule-learning program in high energy physics event classification,” Computer Physics Communications, vol. 67, pp. 159–182, 1991.
- [7] G. M. Weiss, “Mining with rarity: a unifying framework,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 7–19, 2004.
- [8] G. E. A. P. A. Batista, R. C. Prati, and M. C. Monard, “A study of the behavior of several methods for balancing machine learning training data,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 20–29, 2004.
- [9] N. V. Chawla, “C4.5 and imbalanced datasets: Investigating the effect of sampling method, probabilistic estimate, and decision tree structure,” in Proc. 12th Int. Conf. Machine Learning, Workshop on Learning from Imbalanced Data Sets II, 2003.
- [10] T. Jo and N. Japkowicz, “Class imbalances versus small disjuncts,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 40–49, 2004.
- [11] G. M. Weiss and F. Provost, “Learning when training data are costly: The effect of class distribution on tree induction,” J. Artificial Intelligence Research, vol. 19, pp. 315–354, 2003.
- [12] N. Japkowicz, “Class imbalances: Are we focusing on the right issue?” in Proc. 12th Int. Conf. Machine Learning, Workshop on Learning from Imbalanced Data Sets II, 2003.
- [13] R. C. Prati, G. E. A. P. A. Batista, and M. C. Monard, “Class imbalances versus class overlapping: An analysis of a learning system behavior,” in Proc. 3rd Mexican Int. Conf. Artificial Intelligence, Advances in Artificial Intelligence, 2004, pp. 312–321.
- [14] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “Smote: Synthetic minority over-sampling technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
- [15] H. Han, W. Wang, and B.-H. Mao, “Borderline-smote: A new oversampling method in imbalanced data sets learning,” in Proc. Int. Conf. Intelligent Computing, Advances in Intelligent Computing, 2005, pp. 878–887.
- [16] H. Guo and H. L. Viktor, “Learning from imbalanced data sets with boosting and data generation: the databoost-im approach,” ACM SIGKDD Explorations Newsletter, vol. 6, no. 1, pp. 30–39, 2004.
- [17] D. Mease, A. J. Wyner, and A. Buja, “Boosted classification trees and class probability/quantile estimation,” The Journal of Machine Learning Research, vol. 8, pp. 409–439, 2007.

- [18] I. Tomek, “Two modifications of cnn,” *IEEE Trans. System, Man and Communications*, vol. 6, pp. 769–772, 1976.
- [19] J. Yuan, J. Li, and B. Zhang, “Learning concepts from large scale imbalanced data sets using support cluster machines,” in *Proc. 14th Annual ACM Int. Conf. Multimedia*, 2006, pp. 441–450.
- [20] K. M. Ting, “An instance-weighting method to induce cost-sensitive trees,” *IEEE Trans. Knowledge and Data Engineering*, vol. 14, no. 3, pp. 659–665, 2002.
- [21] P. Viola and M. Jones, “Fast and robust classification using asymmetric adaboost and a detector cascade,” in *Advances in Neural Information Processing System 14*. MIT Press, 2002, pp. 1311–1318.
- [22] H. Masnadi-Shirazi and N. Vasconcelos, “Asymmetric boosting,” in *Proc. int. conf. Machine learning*. ACM, 2007, pp. 609–619.
- [23] W. Fan, S. J. Stolfo, J. Zhang, and P. K. Chan, “Adacost: misclassification cost-sensitive boosting,” in *Proc. 16th International Conf. Machine Learning*, 1999, pp. 97–105.
- [24] P. Domingos, “Metacost: a general method for making classifiers costsensitive,” in *Proc. 5th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, 1999, pp. 155–164.
- [25] X.-Y. Liu and Z.-H. Zhou, “Training cost-sensitive neural networks with methods addressing the class imbalance problem,” *IEEE Trans. Knowledge and Data Engineering*, vol. 18, no. 1, pp. 63–77, 2006.
- [26] Y. H. Liu and Y. T. Chen, “Face recognition using total margin-based adaptive fuzzy support vector machines,” *IEEE Trans. Neural Networks*, vol. 18, no. 1, pp. 178–192, 2007.
- [27] G. Wu and E. Y. Chang, “Aligning boundary in kernel space for learning imbalanced dataset,” in *Proc. 4th IEEE Int. Conf. Data Mining*, Brighton, UK, 2004, pp. 265–272.
- [28] G. Wu and E. Chang, “Kba: Kernel boundary alignment considering imbalanced data distribution,” *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 6, pp. 786–795, 2005.
- [29] X. Hong, S. Chen, and C. J. Harris, “A kernel-based two-class classifier for imbalanced data sets,” *IEEE Trans. Neural Networks*, vol. 18, no. 1, pp. 28–41, 2007.
- [30] S. Ertekin, J. Huang, L. Bottou, and L. Giles, “Learning on the border: active learning in imbalanced data classification,” in *Proc. 16th ACM Conf. Information and Knowledge Management*, Lisbon, Portugal, 2007, pp. 127–136.
- [31] S. Ertekin, J. Huang, and L. Giles, “Active learning for class imbalance problem,” in *Proc. 30th Annual Int. ACM SIGIR Conf. Research and Development in Information Retrieval*, Amsterdam, The Netherlands, 2007, pp. 823–824.
- [32] J. Zhu and E. Hovy, “Active learning for word sense disambiguation with methods for addressing the class imbalance problem,” in *Proc. Joint Conf. Empirical Methods in*

Natural Language Processing and Computational Natural Language Learning, Prague, Czech Republic, 2007, pp. 783–790.

[33] H. Zhao and P. C. Yuen, “Incremental linear discriminant analysis for face recognition,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 38, no. 1, pp. 210–221, 2008.

[34] J. R. Millan, “Rapid, safe, and incremental learning of navigation strategies,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 26, no. 3, pp. 408–420, 1996.

[35] G. Y. Chen and W. H. Tsai, “An incremental-learning-by-navigation approach to vision-based autonomous land vehicle guidance in indoor environments using vertical line information and multiweighted generalized hough transform technique,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 28, no. 5, pp. 740–748, 1998.

[36] H. He, S. Chen, Y. Cao, and J. A. Starzyk, “Incremental learning for machine intelligence,” in *Proc. Int. Conf. Cognitive and Neural Systems*, 2008.

[37] R. B. Segal and J. O. Kephart, “Incremental learning in swiftfile,” pp. 863–870, 2000.

[38] G. G. Yen and P. Meesad, “An effective neuro-fuzzy paradigm for machinery condition health monitoring,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 31, no. 4, pp. 523–536, 2001.

[39] J. Su, J. Wang, and Y. Xi, “Incremental learning with balanced update on receptive fields for multi-sensor data fusion,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 34, no. 1, pp. 659–665, 2004.

[40] S. U. Guan and F. Zhu, “An incremental approach to genetic-algorithmsbased classification,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 35, no. 2, pp. 227–239, 2005.

[41] Y. Cao and H. He, “Learning from testing data: A new view of incremental semi-supervised learning,” in *Proc. Int. Joint Conf. Neural Networks*, 2008.

[42] M. Pardowitz, S. Knoop, R. Dillmann, and R. D. Zollner, “Incremental learning of tasks from user demonstrations, past experiences, and vocal comments,” *IEEE Trans. on Systems, Man, and Cybernetics, part B*, vol. 37, no. 2, pp. 322–332, 2007.

[43] A. Sharma, “A note on batch and incremental learnability,” *J. Comput. Syst. Sci.*, vol. 56, no. 3, pp. 272–276, 1998.

[44] S. Lange and G. Grieser, “On the power of incremental learning,” *Theor. Comput. Sci.*, vol. 288, no. 2, pp. 277–307, 2002.

[45] Z.-H. Zhou and Z.-Q. Chen, “Abstract hybrid decision tree,” *Knowl.-Based Syst.*, vol. 15, no. 8, pp. 515–528, 2002.

[46] M. D. Muhlbaier, A. Topalis, and R. Polikar, “Learn++.nc: Combining ensemble of classifiers with dynamically weighted consult-and-vote for efficient incremental learning of new classes,” *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 152–168, 2009.

[47] S. Grossberg, “Adaptive resonance theory,” Technical Report, CAS/CNS TR-2000-024, Boston University, The Encyclopedia of Cognitive Science, 2003.

- [48] R. Polikar, L. Udpa, S. Udpa, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. on System, Man, and Cybernetics, part C*, vol. 31, no. 4, pp. 497–508, 2001.
- [49] H. He and S. Chen, "Imorl: Incremental multiple-object recognition and localization," *IEEE Trans. Neural Networks*, vol. 19, no. 10, pp. 1727–1738, 2008.
- [50] H. Wang, W. Fan, P. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in *Proc. 9th Int. Conf. Knowledge Discovery and Data Mining*. AAAI Press, 2003.
- [51] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Machine Learning*, vol. 23, no. 1, pp. 69–101, 1996.
- [52] R. Klinkenberg and T. Joachims, "Detecting concept drift with support vector machines," in *Proc. Int. Conf. Machine Learning*, 2000, pp. 487–494.
- [53] S. Grossberg, "Nonlinear neural network: Principles, mechanisms, and architectures," *Machine Learning*, vol. 1, no. 1, pp. 17–61, 1998.
- [54] M. Muhlbaier, A. Topali, and R. Polikar, "Learn++.nc: Combining ensemble of classifiers combined with dynamically weighted consultant-vote for efficient incremental learning of new classes", *IEEE Trans. Neural Networks*, vol. 20, no. 1, pp. 152-168, 2009.
- [55] J. Z. Kolter and M. A. Maloof, "Dynamic weighted majority: A new ensemble method for tracking concept drift," in *IEEE Int. Conf. Data Mining*, 2003.
- [56] N. V. Chawla, A. Lazarevic, L. O. Hall, and K. W. Bowyer, "Smoteboost: improving prediction of the minority class in boosting," in *Proc. Principles of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia*, 2003, pp. 107–119.
- [57] H. He, Y. Bai, E. A. Garcia, and S. Li, "Adasyn: Adaptive synthetic sampling approach for imbalanced learning," in *Proc. Int. Joint Conf. Neural Networks*, 2008, pp. 1323–1329.
- [58] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. Int. Conf. Machine Learning*, 1996, pp. 148–156.
- [59] Y. Freund and E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *J. Computer and System Sciences*, vol. 55, no. 1, pp. 119–139, 1997.
- [60] M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection," in *Proc. 14th Int. Conf. Machine Learning*, Nashville, Tennessee, USA, 1997, pp. 179–186.
- [61] J. Gao, W. Fan, J. Han, and P. S. Yu, "A general framework for mining concept-drifting streams with skewed distribution," in *SIAM Int. Conf. Data Mining*, 2007.
- [62] P. C. Mahalanobis, "On the generalized distance in statistics," in *Proc. National Institute of Science of India*, vol. 2, no. 1, pp. 49–55.

- [63] L. Breiman, "Bagging predictors," *Machine Learning*, vol. 24, no. 2, pp. 123–140, 1996.
- [64] J. Gao, W. Fan, and J. Han, "On appropriate assumption to mine data streams: Analysis and practice," in *Proc. Int. Conf. Data Mining*, 2007, pp. 143–152.
- [65] A. Asuncion and D. J. Newman, "Uci machine learning repository," online, [available]: <http://archive.ics.uci.edu/ml/datasets.html>.
- [66] J. Laurikkala, "Improving identification of difficult small classes by balancing class distribution," in *Proc. Conf. AI in Medicine in Europe: Artificial Intelligence Medicine*, 2001, pp. 63–66.
- [67] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Trans. Knowledge and Data Engineering*, vol. 21, No. 9, pp. 1263–1284, 2009.
- [68] "Elena project," online, [available]: <ftp://ftp.dice.ucl.ac.be/pub/neuralnets/ELENA/databases>.
- [69] F. Provost and T. Fawcett, "Analysis and visualization of classifier performance: Comparison under imprecise class and cost distributions," in *Proc. 3rd Int. Conf. Knowledge Discovery and Data Mining*, Newport Beach, CA, USA, 1997, pp. 43–38.
- [70] M. A. Maloof, "Learning when data sets are imbalanced and when costs are unequal and unknown," in *Proc. 20th Int. Conf. Machine Learning, Workshop on Learning from Imbalanced Data Sets II*, Washington, D.C., USA, 2003.
- [71] M. Kubat, R. C. Holte, and S. Matwin, "Machine learning for the detection of oil spills in satellite radar images," vol. 30, 1998, pp. 195–215.
- [72] T. Fawcett, "Roc graphs: Notes and practical considerations for data mining researchers," Technical Report HPL-2003-4, 2003, hP Labs.
- [73] D. Opitz and R. Maclin, "Popular ensemble methods: An empirical study," *J. Artificial Intelligence Research*, vol. 11, pp. 169–198, 1999.
- [74] N. V. Chawla, D. A. Cieslak, L. O. Hall, and A. Joshi, "Automatically countering imbalance and its empirical relationship to cost," *Data Min. Knowl. Discov.*, vol. 17, no. 2, pp. 225–252, 2008.
- [75] F. Provost and P. Domingos, "Well-trained pets: Improving probability estimation trees," in *CeDER Working Paper*, no. IS-00-04, Stern School of Business, New York University, NYC, NY 10012, 2001.
- [76] M. G. Elfekey, W. G. Aref, and A. K. Elmagarmid, "Stagger: Periodicity mining of data streams using expanding sliding windows," in *Proc. Int. Conf. Data Mining*, 2006, pp. 188–199.
- [77] W. N. Street and Y. Kim, "A streaming ensemble algorithm (sea) for large-scale classification," in *Proc. 7th Int. Conf. Knowledge Discovery and Data Mining*, 2001, pp. 377–382.
- [78] S. Smaha, "Haystack: An Intrusion Detection System," in *the Fourth Aerospace Computer Security Applications Conference*, pp. 37 – 44, Orlando, FL, 1988.

- [79] M. M. Sebring, E. Shellhouse, M. E. Hanna, and R. A. Whitehurst, "Expert Systems in Intrusion Detection: A Case Study," in *the 11th National Computer Security Conference*, pp 74-81, 1988.
- [80] D. Anderson, T. Frivold, and A. Valdes, "Next-Generation Intrusion Detection Expert System," *Technical Report*, SRI International, 1995.
- [81] G. White and V. Pooch, "Cooperating Security Managers: Distributed Intrusion Detection Systems," *Computers & Security*, vol. 15, no. 5, pp. 441-450, 1996.
- [82] T. Heberlein, G. Dias, K. Levitt, B. Mukherjee, J. Wood, and D. Wolber, "A Network Security Monitor," in *Proceedings of the 1990 IEEE symposium on Research in Security and Privacy*, pp. 296, 1990.
- [83] SNORT: The Open Source Intrusion Detection System. [Available] URL: <http://www.snort.org>.
- [84] P. Porras and P. Neumann, "EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances," in *the 20th National Information Systems Security Conference*, pp. 1-16, 1997.
- [85] V. Barnett and T. Lewis, *Outliers in Statistical Data*, New York, NY, John Wiley and Sons, 1994.
- [86] C. C. Aggarwal and P. Yu, "Outlier detection for high dimensional data," in *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001.
- [87] S. Kumar and E. Spafford. An Application of Pattern Matching in Intrusion Detection. *Technical Report 94-013*, Purdue University, Department of Computer Sciences, Mar. 1994.
- [88] S. Hawkins, H. He, G. Williams and R. Baxter, Outlier Detection Using Replicator Neural Networks, In Proc. of the 4th Int. Conf. on Data Warehousing and Knowledge Discovery (DaWaK02), Aix-en-Provence, France, pp. 170-180, 2002.
- [89] K. Narita and H. Kitagawa, "Outlier Detection for Transaction Databases Using Association Rules," in the Proceedings of the 2008 The Ninth International Conference on Web-Age Information Management, vol. 00, pp. 373-380 , 2008.
- [90] D. S. Moore and G. P. McCabe, *Introduction to the Practice of Statistics*, 4th edition. New York: W. H. Freeman, 2002.
- [91] B. W. Silverman, *Density Estimation for Statistics and Data Analysis*, 1st edition, Chapman & Hall/CRC, 1986.
- [92] J. D. F. Habbema, J. Hermans, and K. van der Broek, "A Stepwise Discrimination Program Using Density Estimation," In G. Bruckman, editor, *Compstat*, pp. 100-110. Vienna: Physica Verlag, 1974.
- [93] A. W. Bowman, "An Alternative Method of Cross-Validation for the Smoothing of Density Estimation," *Biometrika*, vol. 71, no. 2, pp. 353-360, 1984.
- [94] D. W. Scott and G. R. Terrell, "Biased and Unbiased Cross-validation in Density Estimation," *J. Amer. Statist. Assoc.*, vol. 82, no. 400, pp. 1131-1146, December 1987.

- [95] P. Hall, S. J. Sheather, M. C. Jones, and J. S. Marron, "On Optimal Data-based Bandwidth Selection in Kernel Density Estimation," *Biometrika*, vol. 78, no. 2, pp. 263-269, 1991.
- [96] B. U. Park and J. S. Marron, "Comparison of Data-driven Bandwidth Selectors," *J. Amer. Statist. Assoc.*, vol. 85, no. 409, pp. 66-72, 1990.
- [97] M. C. Jones and R. F. Kappenman, "On a class of kernel density estimate bandwidth selectors," *Scand. J. Statist.*, vol. 19, pp. 337-349, 1991.
- [98] M. C. Jones, J. S. Marron, S. J. Sheather, "A Brief Survey of Bandwidth Selection for Density Estimation," *J. of the American Statistical Association*, vol. 91, 1996.
- [99] V. C. Raykar and R. Duraiswami. Fast optimal bandwidth selection for kernel density estimation. *Proceedings of the sixth SIAM International Conference on Data Mining*, pages 524–528, 2006.
- [100] D. W. Scott and S. R. Rain. Multi-dimensional density estimation. *Data Mining and Computational Statistics*, 23, 2004.
- [101] T. Kohonen, "The Self-organizing map," *Neurocomputing*, Elsevier, 1998.
- [102] T. Kohonen, *Self-organizing Maps*, Springer, 2001.
- [103] T. Kohonen, "The Self-organizing Map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464-1480, September 1990.
- [104] S. Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd edition, pp. 446-454, Prentice Hall, Upper Saddle River, New Jersey 07458, 1999.
- [105] R. Balupari, *Real-Time Network-Based Anomaly Intrusion Detection*, Master's thesis, Ohio University, 2002.
- [106] A. Hämmäläinen, *Self-organizing Map and Reduced Kernel Density Estimation*, PhD thesis, University of Jyväskylä, Jyväskylä, Finland, 1995.
- [107] H. Yin and N. M. Allinson, "Self-organizing mixture networks for probability density estimation", *IEEE Trans. Neural Networks*, vol. 12, no. 2, pp. 405-411, 2001.
- [108] A. Elgammal, R. Duraiswami, and L. Davis, "The Fast Gauss Transform for efficient kernel density evaluation with applications in computer vision," *IEEE Trans. PAMI.*, vol. 25, pp. 1499- 1504, 2003.
- [109] J. Li, H. He, H. Man, and S. Desai, "A General-Purpose FPGA-based Reconfigurable Platform for Video and Image Processing," in *Proc. Int. Sym. Neural Networks (ISNN)*, *Lecture Notes in Computer Science (LNCS)*, vol. 5553, pp. 299-309, 2009.
- [110] Virtex-II Pro family (XC2VP30), Data sheet, [online], available: http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex_ii_pro_fpgas/index.htm
- [111] A. P. Dhawan, *Medical Image Analysis*, Wiley-interscience press, pp. 175-210, 2003.
- [112] B. Jahne, *Digital Image Processing*, 5th edition, Springer, Berlin, 427–440, 2002.

[113] F. Fekri, R. M. Mersereau, and R. W. Schafer, "A Generalized Interpolative Vector Quantization Method for Jointly Optimal Quantization, Interpolation, and Binarization of Text Images," *IEEE Transactions on Image Processing*, vol. 9, issue 7, pp. 1272 – 1281, 2000.

[114] Xilinx Virtex-II PRO(V2-Pro) development system, [online], available:
<http://www.digilentinc.com/Products/Detail.cfm?Prod=XUPV2P&Nav1=Products&Nav2=Programmable>

[115] S. Chen and H. He, "SERA: Selectively Recursive Approach towards Nonstationary Imbalanced Stream Data Mining," in *Proc. Int. Joint Conf. Neural Networks (IJCNN'09)*, pp. 522-529, 2009.

[116] Y. Cao, H. He, H. Man, X. Shen, "Integration of Self-organizing Map (SOM) and Kernel Density Estimation (KDE) for Network Intrusion Detection," in *Proc. SPIE*, 2009 (in press)