M.S. THESIS

Concept-Aware Ensemble System for Pedestrian Detection

보행자 인식을 위한 상황 인지 앙상블 시스템

BY

Helin Lin

November, 2013

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

Concept-Aware Ensemble System for Pedestrian Detection

보행자 인식을 위한 상황 인지 앙상블 시스템

지도교수 최 기 영

이 논문을 공학석사 학위논문으로 제출함

2013년 11월

서울대학교 대학원

전기컴퓨터 공학부

림 학 림

림학림의 공학석사 학위논문을 인준함

2013년 11월

위 원 장 : 채 수 익______________

부위원장 : 최 기 영______________

위　　원 : 최 진 영______________

Abstract

Concept-Aware Ensemble System for Pedestrian Detection

Helin Lin
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
COLLEGE OF ENGINEERING
SEOUL NATIONAL UNIVERSITY

In ADAS, multiple classifier system on pedestrian detection is occupying important position because of its merit that each classifier can be able to create synergistic approaches to compensate the other member classifier’s inability. On the other hand, according to different poses of pedestrians and variable background, once trained pedestrian detector needs to be tuned dynamically along the variation of real-world environment, thus the system is requested to incrementally accept new information and retain the old one at the same time.

This thesis presents an incremental learning, environment-adaptive ensemble system for pedestrian detection by combining pedestrian detector constituted by multiple classifiers with front-end concept recognizer that can turn off inefficient member classifiers adaptively. Through adopting incremental learning algorithm, newly added batch dataset is trained by learning algorithm and the newly generated classifier is united to the existing ensemble along with the update of the voting weight. As the update of voting weight is only taken when the new training is carried out and focuses on the performance on current environment, temporal trade-off on performance between current and old environment is inevitable. This problem is addressed by applying concept recognizer in front of the ensemble thus turning off ineffective classifiers and selecting the most efficient voting weight vector adaptive to each pedestrian candidate. With the intervention of the front-end concept recognizer, the system can retain good performance on old environment while does not lose focus on current environment.

Keywords: Incremental learning, Multiple classifier system, Pedestrian detection, ADAS

Student Number: 2012-22568

Contents

Abstract **I**

Contents **III**

List of Figures **V**

Chapter 1 Introduction **1**

Chapter 2 Pedestrian Detection Basics **4**

2.1 Detection Flow 4

2.2 HOG Feature Descriptor 5

2.3 SVM Learning Algorithm 7

Chapter 3 Related Work **9**

3.1 Incremental Learning 9

Chapter 4 Proposed Approach **11**

4.1 Incremental Learning on Pedestrian Detector 12

4.2 Incremental Learning on Concept Recognizer 14

4.3 Cooperation between Pedestrian Detector and Concept Recognizer 15

Chapter 5 Experimental Results **20**

5.1 Experimental Setup 20

5.2 Performance Analysis 23

Chapter 6 Conclusion and Future Work 33

Bibliography 35

Abstract in Korean 37

List of Figures

Figure 1. Pedestrian detection flow. 5

Figure 2. Histogram of Oriented Gradients. 6

Figure 3. Support Vector Machine. 7

Figure 4. Concept-aware ensemble system for pedestrian detection 12

Figure 5. Basic structure of concept recognizer. 14

Figure 6. Contents of lookup table. 15

Figure 7. Performance analysis of a classifier on various datasets. 21

Figure 8. The order of concept drift. 22

Figure 9. Accuracy on total test. 25

Figure 10. Performance gap when the last training dataset is CVC-01. 26

Figure 11. Performance gap when the last training dataset is NICTA. 27

Figure 12. Average performance on already learned data. 28

Figure 13. Average performance on unfamiliar data. 29

Figure 14. Performance variation on Daimler test dataset. 30

Figure 15. Performance variation on CVC-01 test dataset. 31

Figure 16. Performance variation on NICTA test dataset. 32

Chapter 1

Introduction

Due to increasing popularity of automobiles as well as road accidents, the importance of driver assistance systems is daily growing. Car-embedded system for pedestrian detection is gradually becoming a mainstream of protection systems for improving traffic safety. Following this trend, multiple classifier systems [10] for pedestrian detection are getting more and more importance these days because of their merit that a classifier can be able to compensate inabilities of other member classifiers, thereby achieving a significant synergistic effect.

Such a pedestrian detection system, once have been trained, needs to accept new information along the variation of real-world environment or newly acknowledge poses of pedestrians, which requires incremental learning. There are some important reasons why the subsequent incremental learning cannot be included in the initial training: 1) there is no representative dataset that can cover all the cases for the variation of poses and background from the beginning, 2) the training of a big volume of data at once is time-costly, 3) storage space can be limited to hold all the data, and 4) accidental loss of data already used for training can happen.

The best scenario is that the system has no need to refer to old data when it learns new information while still free from catastrophic forgetting. Such an approach was taken by an algorithm called Learn++.NSE [2]. It is also based on ensemble of classifiers and its main contribution is that it can closely follow the changing environment by incrementally learning new information by only using new data but without accessing already used ones. Unlike its original version [11], which has the problem of outvoting by the new environment due to the fixed voting weights, the new version can put more weight on the new environment by dynamically determining time-adjusted voting weights of member classifiers according to their performance on latest environments.

Because of the particularity of pedestrian detection, the system should also consider some additional issue called concept drift. In line with the variation of pedestrian's pose (front/rear, right/left) as well as the diversity of background (illumination, occlusion), the pedestrian detection problem in general has a very high rate of concept drift. Frequent concept drift is best covered by frequent retraining and voting weight update according to Learn++.NSE, and the training should be done with correctly labeled training data, which is hard to be met in real world. I overcome this shortcoming by adding a *concept recognizer* that automatically detects the concept for the input data, turns off inefficient classifiers, and chooses the most suitable voting weight vector adaptively for the remaining member classifiers.

I still apply Learn++.NSE incremental learning algorithm to implement a pedestrian detector that can follow the change of environment through retraining. However, instead of retraining the system very often, a concept recognizer, which is the main contribution of this paper, is introduced in front of the pedestrian detector as mentioned above. I call the system a *concept-aware ensemble system*.

The remainder of this paper is organized as follows. Chapter 2 is for the basic knowledge of pedestrian detection. Chapter 3 discusses related work, followed by Chapter 4 which explains the details of our approach. The experimental results are provided in Chapter 5, and conclusions with remaining work for further improvement are given in Chapter 6.

Chapter 2

Pedestrian Detection Basics

3.1 Detection Flow

Pedestrian detection generally covers preprocessing, foreground segmentation, object classification and tracking as shown in Figure 1. Usually the pedestrian image is taken by either monocular-based or stereo-based method. Tasks like camera calibration, fine adjustment as well as exposure time are managed in preprocessing. Foreground segmentation has significant effect on reducing calculation time. It is the stage of extracting *region of interest* (ROI) or generating candidates for the object classification stage. Without explicit segmentation, like in exhaustive scanning, many background regions can cost computation time and it is fatal to this real time system. And any misses of

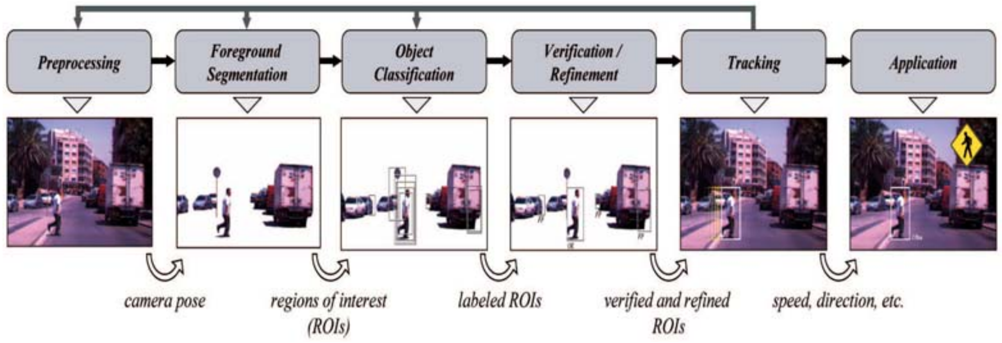


Figure 1. Pedestrian detection flow[12].

pedestrians in this stage will never be recovered by the subsequent stages. These forwarded ROIs are classified into "pedestrian" or "not-pedestrian" at the object classification stage. The main goal in this stage is to maximize hit rate while minimize false positive. Variable features like *Histogram of Oriented Gradients* (HOG) [5], LBP [13], Haar wavelet as well as shape context descriptor are extracted and fed to learning algorithms involving *Support Vector Machine* (SVM), *Multi-layer Perceptron* (MLP), AdaBoost and etc. The combination of HOG-SVM has superior performance on pedestrian detection and is widely used. As the tracking module, it makes the system to follow detected pedestrian. This stage can help to predict the direction to which the pedestrian is heading and also, can assist the foreground segmentation to decide pedestrian candidate.

3.2 HOG Feature Descriptor

Histogram of Oriented Gradients (HOG) are feature descriptors which are believed the one of the best features for pedestrian detection. The main method of HOG is that in an image, the appearance and shape of local object can be well

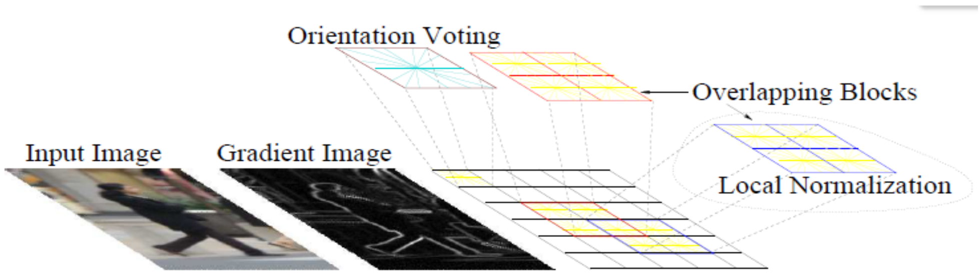


Figure 2. Histogram of Oriented Gradients[5].

described by gradient or edge direction. The essence is the statistic information of gradients and they are usually exist near edge.

It divides the image into small cells, for example, each cell consists 8x8 pixels. The shape of the cell can be radial or rectangular. Then the gradient of each pixel is calculated using horizontal and vertical two masks, [-1, 0, -1] and $[-1, 0, -1]^T$. Thus every pixel has its gradient and direction. It is mainly to capture the silhouette information, at the same time weaken the interference from light further. Then every pixel casts a weighted vote for an orientation-based histogram channel with its gradient value and direction. The voting weight can be the gradient itself or some return values from various kinds of functions utilizing this gradient. The channel is composed by nine bins involves from 0~π. For example, a pixel has direction degree among 60~80, and the gradient is about 7, then it votes the fourth bin with its original voting weight 7. Now a cell histogram with 9 bins is created and this is the basic cell descriptor. Several cells, for example 2x2, are combined to compose a block and one cell can belong to different blocks. Thus a block descriptor is constituted by connected four cell histograms. The last step is to connect all the overlapping block descriptors

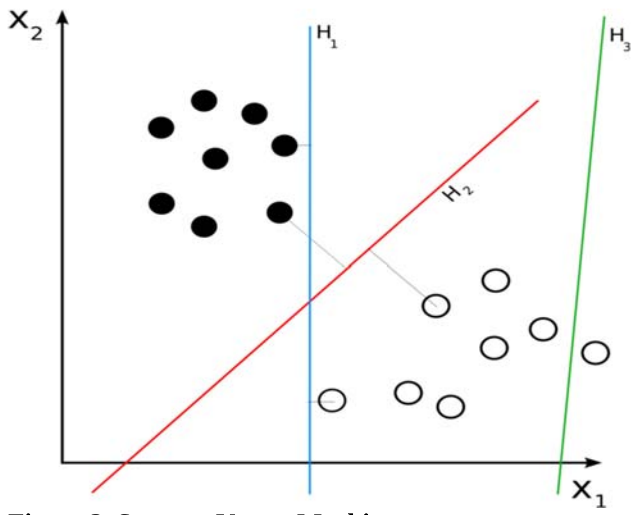


Figure 3. Support Vector Machine.

serially and the HOG feature descriptor for the image is created as high dimension vector.

3.3 SVM Learning Algorithm

Support Vector Machine is a widely used machine learning algorithm on pattern recognition or other detection application. The main theory of it can be summarized as two points: (1) it aims to analyze linearly separable problem. To those cannot be separated linearly, through using the nonlinear mapping algorithm, it maps these linearly inseparable low-dimensional space to high-dimensional feature space. Thus in the higher dimension space, the sample can be separated linearly by hyper-plane. (2) Based on structural risk minimization theory, it constructs the optimal partitioning hyper-plane in the feature space and makes the expected risk in the whole sample space to satisfy an upper bound restriction.

The SVM learning can be interpreted as convex optimization problem, thus utilizing already known effective algorithm to find global minimization of target function. Unlike the other learning algorithms (rule-based classifier and artificial neural networks) those search hypothesis space based on greedy tactics and so mostly can only get local optimal. SVM controls its ability through maximizing the distance between the decision hyper-plane with boundary vector (support vector). Nevertheless, parameters like kernel function and soft margin have important effect to the performance. Different kernel functions make SVM variable. Common kernel functions are as below.

(1) Linear: $K(x, y) = x \cdot y$

(2) Polynomial: $K(x, y) = (x \cdot y + 1)^d$

(3) Radian Basis Function: $K(x, y) = \exp(-\gamma \| x - y \|^2)$

(4) Hyperbolic tangent: $K(x, y) = \tanh(\kappa x \cdot y + c)$

SVM is supervised learning model thus needs labeled training data. In the training process, it tries to select the support vectors as well as the hyper-plane from which, these support vectors have max margins. In Figure 3, the hyper-plane H_2 is the optimal one. SVM performs best on binary classification problem, rather than on multi-class problem.

Chapter 3

Related Work

3.1 Incremental Learning

There have been previous approaches to incremental learning, and Learn++.NSE and Learn++.NC [1] are representative ones for multi-classifier systems. Learn++.NSE is efficient in closely following changing environment and Learn++.NC is efficient in adding or deleting classes.

Learn++.NSE generates a member classifier every time there happens concept drift and the existing ensemble is no longer suit for the current environment. New classifier is generated using the training data collected from recent environment. Every member classifier including the newly generated one is evaluated on the newly added dataset. For each member classifier, all the error rates, one for each newly added dataset, are saved. The error rates collected thus are normalized, weighted, and used to calculate the final voting weight for that classifier. The weight (penalty) for the normalized error rate is set larger on more recent environment than on less recent one and each classifier gets its new time-adjusted voting weight. In such a method, the voting weight vector of the ensemble is better adapted to more recent environments. Then the final hypothesis is calculated using max voting. Since the total number of member classifiers cannot be increased infinitely, some classifiers should be pruned based either on age or on error.

The main frame of Learn++.NC is similar to Learn++.NSE. If the new dataset introduces some new classes, the existing classifiers that are trained without such classes inevitably make wrong decisions. To reduce such “outvoting” negative impact, Learn++.NC utilizes dynamically weighted consult and vote (DW-CAV) as its voting method. The special feature of this voting method is that the voting weight of each member classifier is varied according to the response of each classifier to the input instance. Thus it can keep balance of voting weight among classifiers and address outvoting problem.

Chapter 4

Proposed Approach

The proposed concept-aware ensemble system for pedestrian detection is a combination of the concept recognizer and the conventional ensemble pedestrian detector as shown in Figure 4. For an input instance, the concept recognizer at the front-end determines the concept and chooses an appropriate set of classifiers and the corresponding voting weights in the lookup table according to the recognized concept. The selected set of classifiers and the voting weights make up the ensemble pedestrian detector at the back-end which makes a final decision on whether it has detected a pedestrian or not. As mentioned in Chapter 1, I use Learn++.NSE for the conventional ensemble pedestrian detector at the back-end. For the concept recognizer at the front-end, I also use incremental learning, but in that case, I use Learn++.NC. To avoid confusion, I call the classifiers used in the

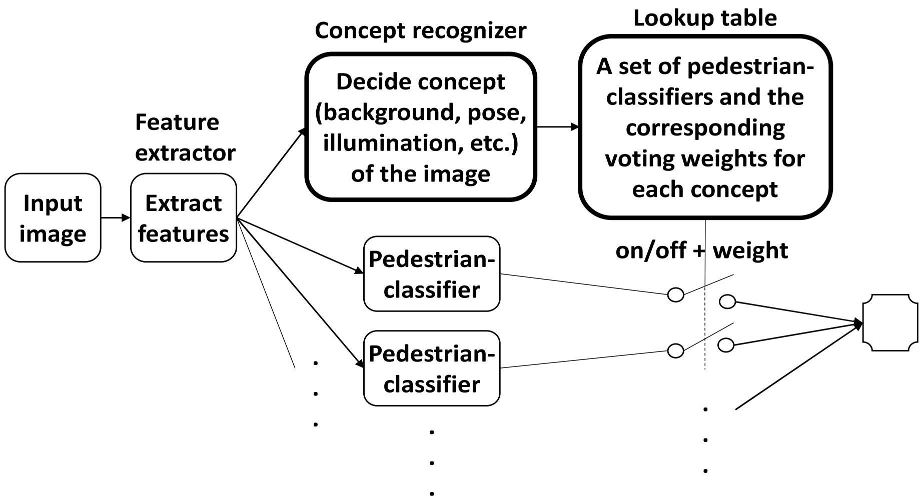

Concept recognizer
Lookup table
Decide concept (background, pose, illumination, etc.) of the image
A set of pedestrian-classifiers and the corresponding voting weights for each concept
Feature extractor
Input image
Extract features
on/off + weight
Pedestrian-classifier
Pedestrian-classifier

Figure 4. Concept-aware ensemble system for pedestrian detection.

concept recognizer as *concept-classifiers* (CCs) and the ones in the back-end pedestrian detector as *pedestrian-classifiers* (PCs).

4.1 Incremental Learning on Pedestrian Detector

Many state-of-the-art pedestrian detectors consisting of feature extractors and classifiers are trained once with a specific pedestrian dataset. This cannot be an issue when the detector is used by some invariant environment like surveillance systems. However, in the case of a detection system embedded in a moving car, the background as well as the poses of pedestrians are always varying, thus easily resulting in maladjustment. In order to avoid the problem, I implement a tuned version of Learn++.NSE to make it cooperate with the concept recognizer in the front-end, while not altering any key aspect of the original version. Following are the major changes that I make.

First, whereas the original algorithm sets the error rate threshold to 1/2, I suggest a much higher restriction as the error rate threshold since pedestrian detection is a binary classification problem and thus fifty percent error rate can be achieved even with a random prediction. If the newly generated classifier has error rate bigger than this newly suggested error threshold, it is regenerated. For the existing classifiers whose error rate exceeds the threshold, their voting weights for the specific dataset are set to zero.

In addition, not all the member classifiers participate in making final decisions; less efficient classifiers are blocked by the front-end concept recognizer. In the same perspective, the voting weight update for a member classifier is conducted only when the classifier is judged to be efficient on the current training dataset (i.e., the current concept). Also, a classifier can has multiple voting weights if it is involved in multiple different concepts. Further details are discussed in 4.3.

It is undesirable to have too many different concepts for several reasons. First of all, it will make the problem too complicated. Most of input instances can be classified into a limited set of concepts. Increasing the number of concepts does not guarantee sufficient increase in the accuracy. Thus the change in the concept of input data tends to have cycle property. To take advantage of this, Learn++.NSE does not delete old classifiers that do not join the current hypothesis and resumes them when their evaluation on latest training dataset get high score. The limitation of this approach is that the old classifiers are resumed only through a new training. It is never reused otherwise, even if it encounters with an already learned concept. To automatically recognize previously encountered concepts and just reuse the corresponding classifiers and training

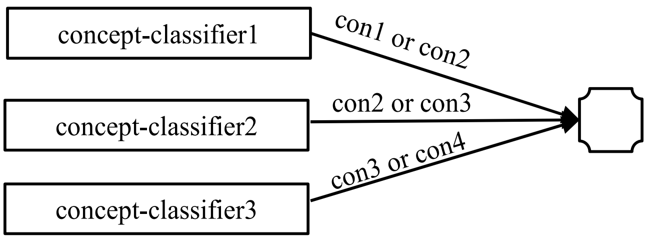


Figure 5. Basic structure of concept recognizer

results (voting weights), the proposed approach introduces the concept recognizer into the system.

4.2 Incremental Learning on Concept Recognizer

The concept recognizer should also be trained incrementally since all the training data sets are not available at the same time. Unlike pedestrian detection, which is just considering binary classification, there can be many different concepts and thus it is much closer to multi-classification problem. To efficiently add new concepts as new classes, I utilize Learn++.NC for the construction of the concept recognizer. Since it is not known *a priori* how much concepts are there totally at the beginning, every time a new concept is found, it should be learned incrementally one by one.

Basically, the concept recognizer is relying on multiple binary CCs. For example, the first CC is trained to classify concept1 and concept2 as shown in Figure 5. When the system encounters an unfamiliar one, e.g., concept3, a new CC is generated. However, since it requires two different concepts to train itself for binary classification, it takes the most recently created concept from the existing set of concepts to match with the new concept. Thus concept2 is selected in our example and the second CC is trained to classify input instances into

con1	PC1 (0.72)	**con1**	PC1 (0.62) PC3 (0.85)	**con1**	PC1 (0.72)
con2	PC2 (0.93)	**con2**	PC2 (0.93)	**con2**	PC2 (0.93)
con3	PC3 (0.82)			**con3**	PC3 (0.82) PC1 (0.55) PC2 (0.57)
	(a)		(b)		(c)

Figure 6. Contents of the lookup table after concept generation and independent voting weight update on the third dataset. (a) Neither of PC1 or PC2 performs well on the dataset. (b) Just PC2 performs well on the dataset. (c) Both PC1 and PC2 perform well on the dataset. (The number in the parentheses is the voting weight)

concept2 and concept3. Continuing one step further, I obtain the concept recognizer shown in Figure 5.

So the three supposed CCs in Figure 5 can only classify the input instance as 1) CC1, CC2, and CC3 respectively take care of concept1/concept2, concept2/concept3, and concept3/concept4 pairs. When the input instance belongs to concept4, for example, the CC1 and CC2 will give an incorrect decision because the case is out of their coverage. Such a problem is successfully resolved by the voting method used in DW-CAV.

4.3 Cooperation between Pedestrian Detector and Concept Recognizer

The intervention of the concept recognizer to the original pedestrian detector constructed by Learn++.NSE incurs significant changes in the update of voting weight vector as well as the final hypothesis. The decision on whether to add a new concept to the concept recognizer is also decided through the evaluation by the PCs on the current training data.

A dataset is considered to represent an environment or a concept here. The training starts with only one PC, and the input instances are assumed to belong to concept1, which is the only existing concept and thus no concept recognizer is needed. Therefore, there is no difference between the proposed algorithm and the original Learn++.NSE.

When a second dataset is used for training, then a second PC is created and trained with the dataset. Also, the first PC is evaluated with the dataset to see if it performs well enough (hit rate exceeds 90% while false positive is kept below 10%; the threshold values are determined empirically). If the first PC performs well, then no new concept is created and the two PCs are used to make a final decision (their decisions are combined with proper weights according to their hit rates). Otherwise, it is interpreted as a concept drift and a second concept (concept2) is created and the second PC is included in the PC set of concept2; it is unknown if the second PC can perform well on concept1. Since there are two different concepts now, the concept recognizer can come in handy. The role of the first CC is to classify input instances into one of the two concepts: concept1 and concept2.

Training a CC requires a set of input data labeled concept1 or concept2. Data for concept2 is extracted from the current PC training data. It is composed of the data on which PC1 has made a wrong decision. To support the training for concept1 as the anti-pole class of concept2, I save a small part (100 true instances and 100 false instances) of the previous dataset. This part of data is not retained permanently, but will be deleted as soon as it is used for the training of the next CC. Then a part of the current dataset for concept2 is saved by the same reason.

When the third training dataset is fed, a new PC is generated and both of the existing PCs are evaluated on this dataset. There are many different cases. First, consider having only one existing concept (concept1) and thus the two existing PCs have been used for that concept. If the two existing PCs perform well, then they are combined with the new PC for concept1 and no new concept is created. If only one PC performs well, then the PC is combined with the new PC to comprise a PC set for a newly created concept. If no existing PC performs well, then only the new PC is included in the PC set of the newly created concept. Now, consider having two existing concepts (concept1 and concept2). If only one existing PC performs well on this new dataset, then the concept of the new dataset is assumed to be the same as that having the PC in its PC set. Thus no new concept is created and the PC is combined with the new PC to make a new PC set for the existing concept (see the example in Figure 6(b)). If both of them have bad efficiency on this dataset, a new concept is created, which includes only the new PC in its PC set. Then a new CC that classifies input data into concept2 and concept3 is added as shown in Figure 6(a). If both of the existing PCs have good performance, the third concept is created to use all three PCs with the update of time-adjusted voting weight for the three PCs. In this case, there are three concepts as shown in Figure 6(c) and both PC1 and PC2 have two different voting weights, one for each different concept.

The concept drift may either introduce a new concept or cycle back to the existing one. In both cases, a new CC is generated (even though no new concept is generated in the case of cycling back, still a new CC is added to strengthen the classification of the new dataset; if there is no concept drift, even if a new PC is generated, there will be no generation of a new CC). In the examples of Figure 6, concept drift occurs from concep2 to concept3 on the third dataset for cases (a) and (c), and from concept2 back to concept1 for case (b). In all the three cases, a new CC (concept2/concept3 or concept2/concept1) is generated. The case where PC3 is assigned to concept2 together with PC2 is considered as no concept drift. Voting weight for a PC is updated in the same way as the original Learn++.NSE. However, the update is conducted only when there comes a new PC into the PC set of a concept.

The accuracy threshold for making decision on whether an existing PC can be selected for the composition of the current concept should be set higher than the error rate threshold that decides the voting weight of a PC on the current training dataset. If the two thresholds are the same, for example, then there will not be much performance improvement due to the concept recognizer, because each newly generated classifier will likely be included in the same concept as that of the existing member PCs. For an extreme example, if both of the two thresholds are set to 1/2 (when the error rate of a PC is over 1/2, it will be set to 1/2 to make its voting weight to zero on that concept), the concept recognizer will totally lose its role, and the system will behave like the one having only one concept.

The distribution calculation* for the instances in a training dataset, which is used in Learn++.NSE to determine the voting weight of each PC, is also adopted in the proposed system. Note, however, that it works together with the concept recognizer. Thus, in contrast to the original system where all the PCs take part in the calculation for an input image, in the proposed system, each instance is classified first according to its concept and then only the PCs involved in that

----

*In Learn++.NSE, every time a new dataset is fed, the distribution of each instance in the dataset is calculated by the existing ensemble of classifiers. By doing so, an instance on which the existing ensemble has worse performance can get higher distribution. Then a PC that does not perform well on instances with high distributions is assigned with a low voting weight.

concept join the calculation.

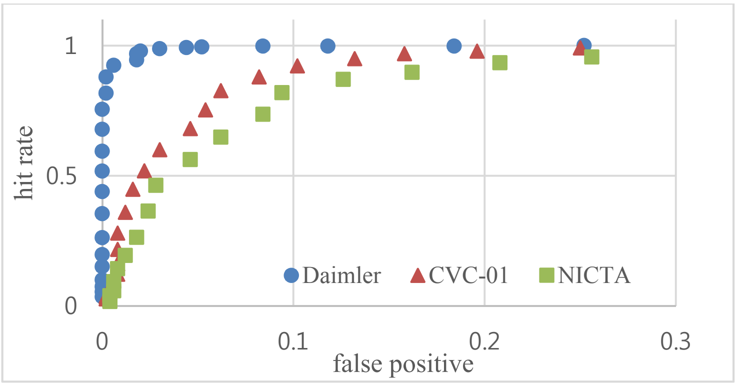
In summary, all the existing PCs are evaluated on the newly added dataset and only PCs those perform well on the new dataset are selected to implement a new concept. If it turns out that all the selected old classifiers are exactly the same as those in the PC set of an existing concept, the newly generated PC is just added to that concept instead of generating a new one. A CC is generated only when there happens concept drift, and the update of voting weight is proceeded independently in each concept, and thus the instance distribution calculation of training data on PC generation as well as the final hypothesis is also conducted with the intervention of concept recognizer.

Chapter 5

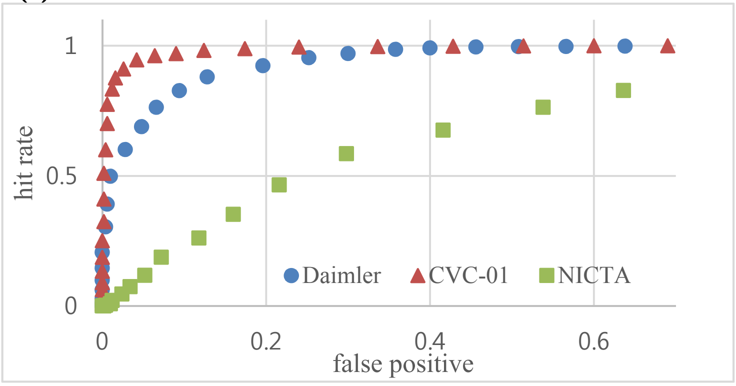
Experimental Results

5.1 Experimental Setup

To show the effectiveness of the proposed system described above, I experimented with various pedestrian datasets. The experiment was performed under the assumption that different pedestrian datasets represented different concepts as they were collected by different places with variable methods. This assumption was proved reasonable by the fact that a classifier trained using "Daimler" [6] performed much worse on other pedestrian datasets such as "CVC-01" [7] and "NICTA" [8], and so did the classifiers trained by "CVC-01" and "NICTA" respectively as shown in Figure 7(a) ~ (c). For this experiment, I set the order of concept drifts like the one in Figure 8. For this experiment, I



(a) Performance of the classifier trained with Daimler dataset



(b) Performance of the classifier trained with CVC-01 dataset

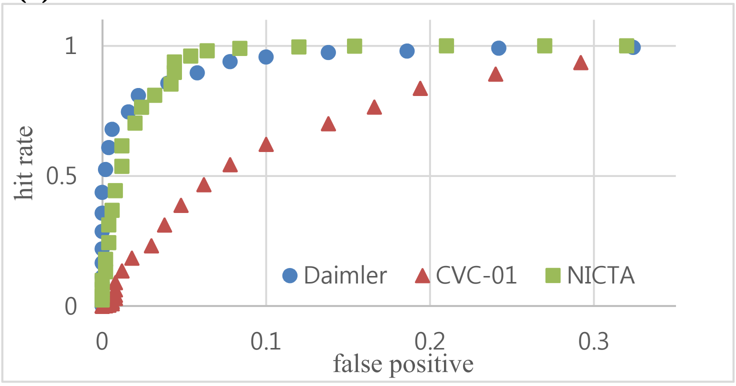


Figure 7. Performance analysis of a classifier on various datasets. (a) Classifier is trained using Daimler. (b) Classifier is trained using CVC-01. (c) Classifier is trained using NICTA.

extracted three training sets from each of the original datasets. For example, I extracted Daimler1, Daimler2, and Daimler3 from the Daimler dataset. I also

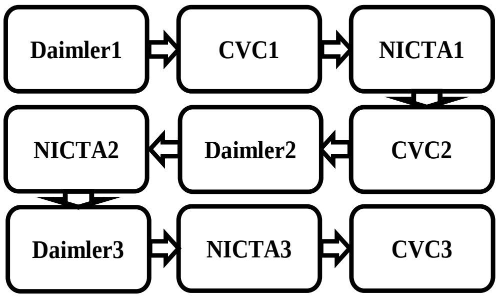


Figure 8. The order of concept drift

extracted CVC1, CVC2, and CVC3 from CVC-01, but in this case, some instances were imported from other CVC series of datasets because of the shortage of positive images in CVC-01.

For the feature, I selected rectangular Histogram of Oriented Gradients (HOG) which was considered as the most efficient for pedestrian objects. The size of HOG was set to window size = 48x96 pixels, block size = 2x2 cells, and cell size = 8x8 pixels. The sliding step of the block was set to one cell size both in horizontal and vertical directions. The extracted HOG features were labeled as “pedestrian” or “not-pedestrian”, and then fed to a linear SVM [3] as a PC to be trained. For the CC, I utilized the same HOG features considering the calculation cost, but this time, these HOG features were labeled differently with the corresponding concepts. The maximum number of PCs was set to nine. All the parameters used in the original Learn++.NSE remain in our implementation except for the error rate threshold, which was changed from 1/2 to 3/4. For the classifier selection and training, I utilized the OpenCV library.

I tested two different versions of the system: a traditional pedestrian detector and the proposed combination of pedestrian detector and concept recognizer. The traditional version was to see if the system could successfully follow the concept drift with retraining at right time. And the result was also used for the comparison with the other version. The comparison was conducted mainly focusing on three points: 1) trade-off between the performance for the current environment and that of old environments, 2) performance on already learned datasets (a part of a dataset is used for training and the rest is used for testing), and 3) performance on unfamiliar datasets. The third point was considered by testing the system on pedestrian datasets of MIT [9] and INRIA [5] after training the system with pedestrian datasets of Daimler, CVC-01, and NICTA.

5.2 Performance Analysis

As shown in Figure 10 ~ 13, the proposed version with concept recognizer outperforms the one with pure pedestrian detector in both performance trade-off (between current and old concepts) and average performance on data from already learned or unfamiliar datasets. It is clear that there exists a synergy effect between the front-end (concept recognizer) and back-end (pedestrian detector) ensembles. The concept recognizer incrementally learns new concepts and keeps efficient voting weight vector, while the pedestrian detector learns new information of pedestrians incrementally and closely follows the change at the same time.

Figure 12 shows the average performance on the nine datasets used for training. Every time the system was retrained with a new dataset in the order shown in Figure 8, it was tested for performance on all the already learned datasets. Thus, Daimler1, the first dataset, was tested nine times and all the results were averaged to obtain the data in the figure. On the other hand, cvc3 was tested only once.

Figure 13 shows the average performance on the never learned data. The performance improvement in Daimler, CVC-01, and NICTA test datasets is not strange because of the similarity between the training and test datasets. While the most encouraging result is that the performance on the unfamiliar datasets like INRIA and MIT is also improved.

Figure 10 ~ 11 show the performance gap between the two different versions of systems: without and with concept recognizer. Without the concept recognizer, it performs well only on the data similar to last trained concept. With the concept recognizer, the performance is a little decreased on the last trained concept but the performance on old concepts significantly outperforms. The loss of performance on data with the same concept is natural since without the interference of the concept recognizer, the system focuses only on the currently learned concept.

In Figure 9, I tested the system on a total test set to see if it conducts incremental learning well. The total test set was composed of 8,000 positive and negative images. The proportions of positive and negative images are the same and same number of images were extracted from each of Daimler, CVC-01, NICTA, and INRIA. The figure shows that the version with concept recognizer outperforms in the whole training process and that it successfully does incremental learning.

Figure 14 ~ 16 show the performance trend in Daimler, CVC-01, and NICTA test datasets during the nine trainings. In every case, when the system gets trained with the data from the same pedestrian datasets, it performs better. The difference is that when there is no concept recognizer, the amplitude of performance change is much bigger than the one with concept recognizer. This is

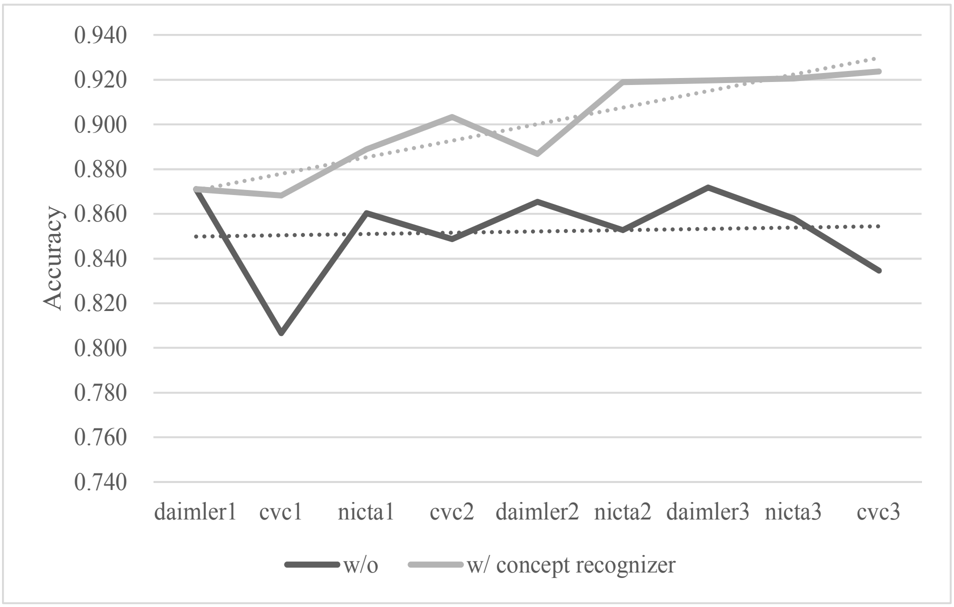
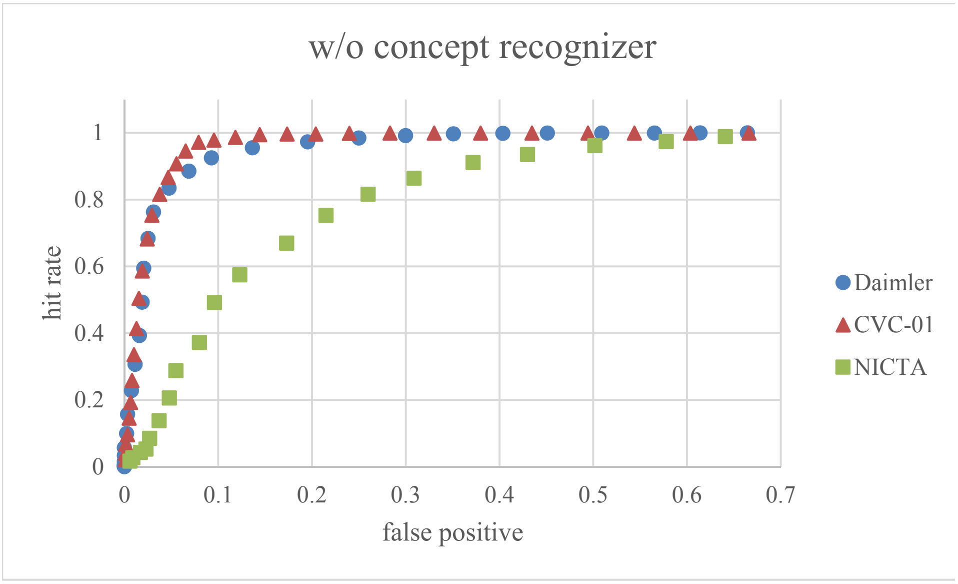


Figure 9. Accuracy on total test (positive + negative).

because the proposed system focuses on the current concept while not losing attention to the previously learned concepts.



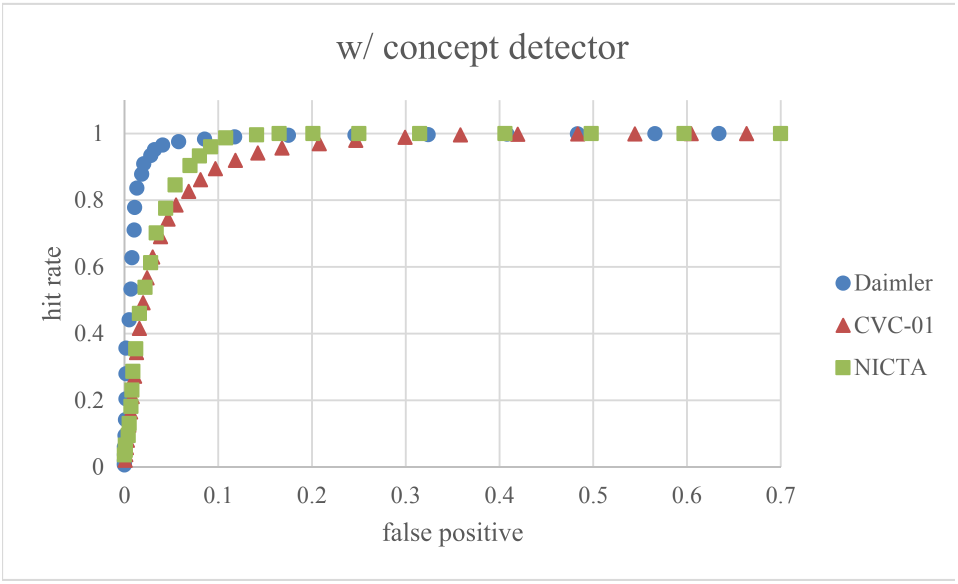


Figure 10. Performance gap when the last training dataset is CVC-01.

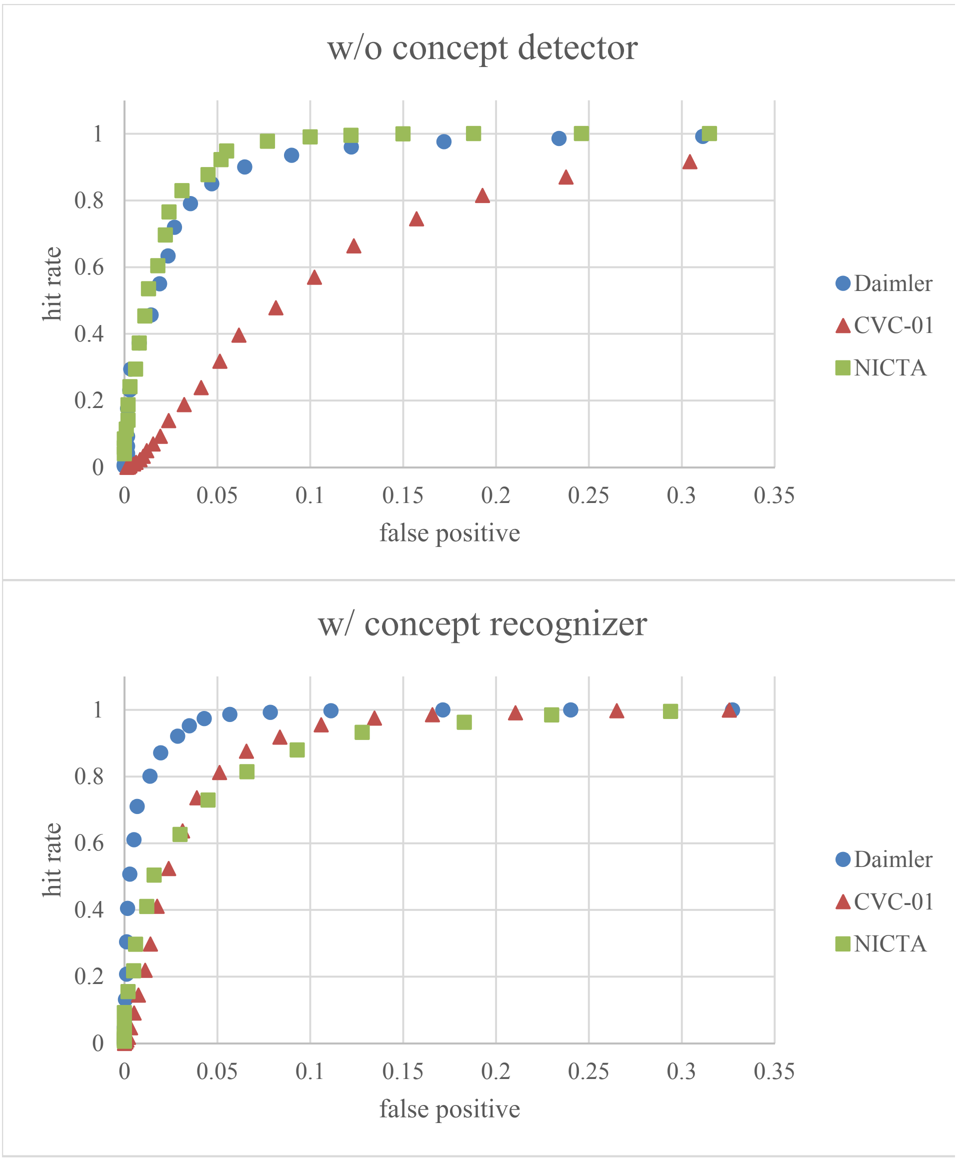
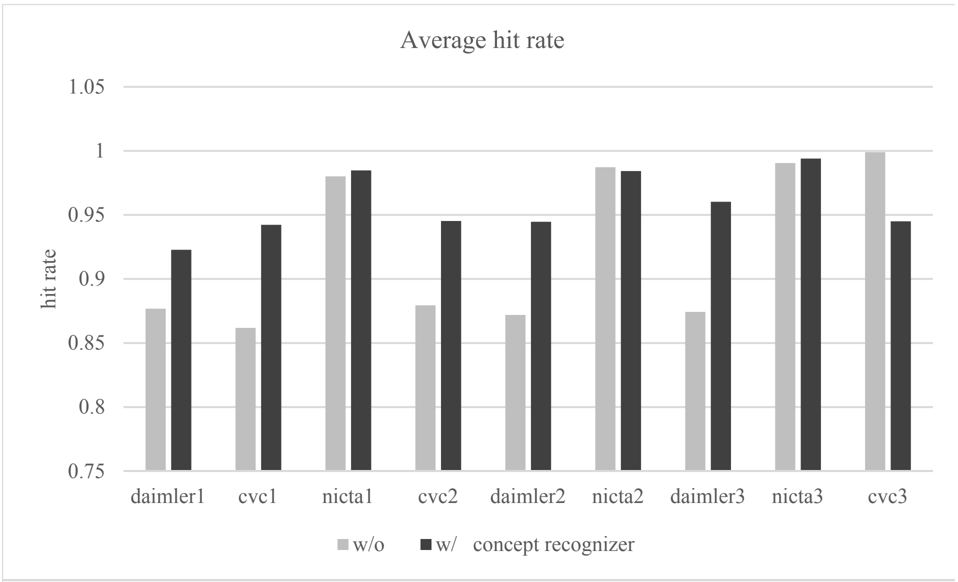


Figure 11. Performance gap when the last training dataset is NICTA.



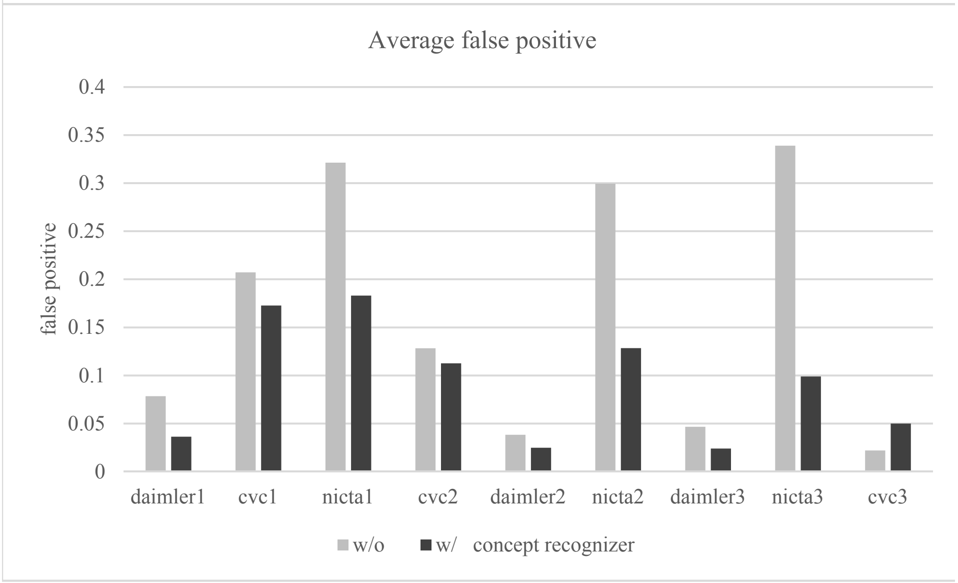
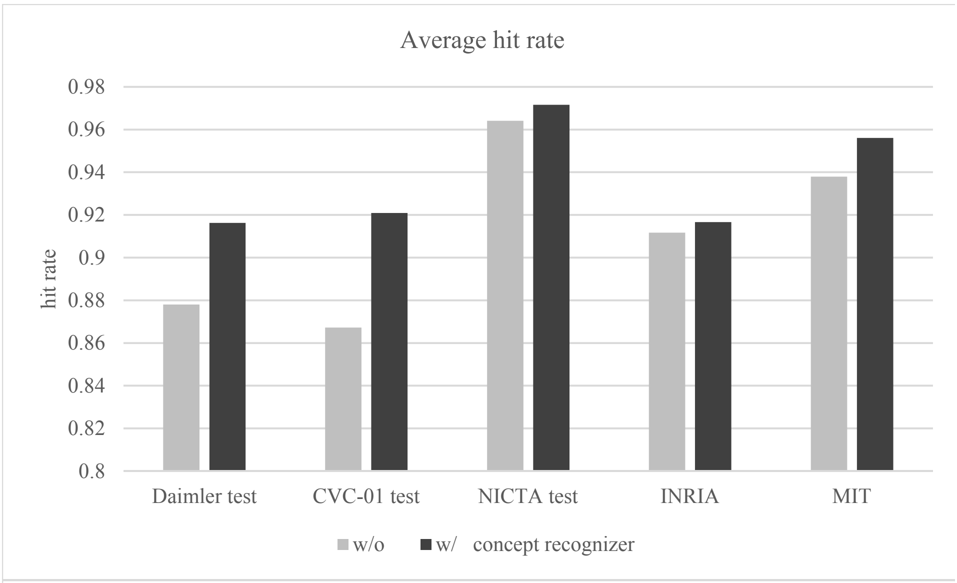


Figure 12. Average performance on already learned data.



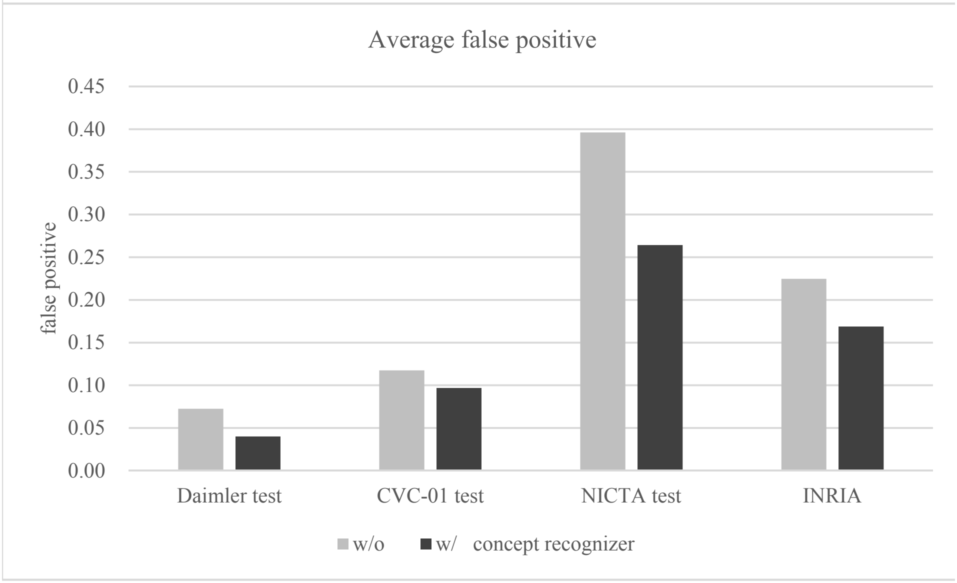


Figure 13. Average performance on unfamiliar data.

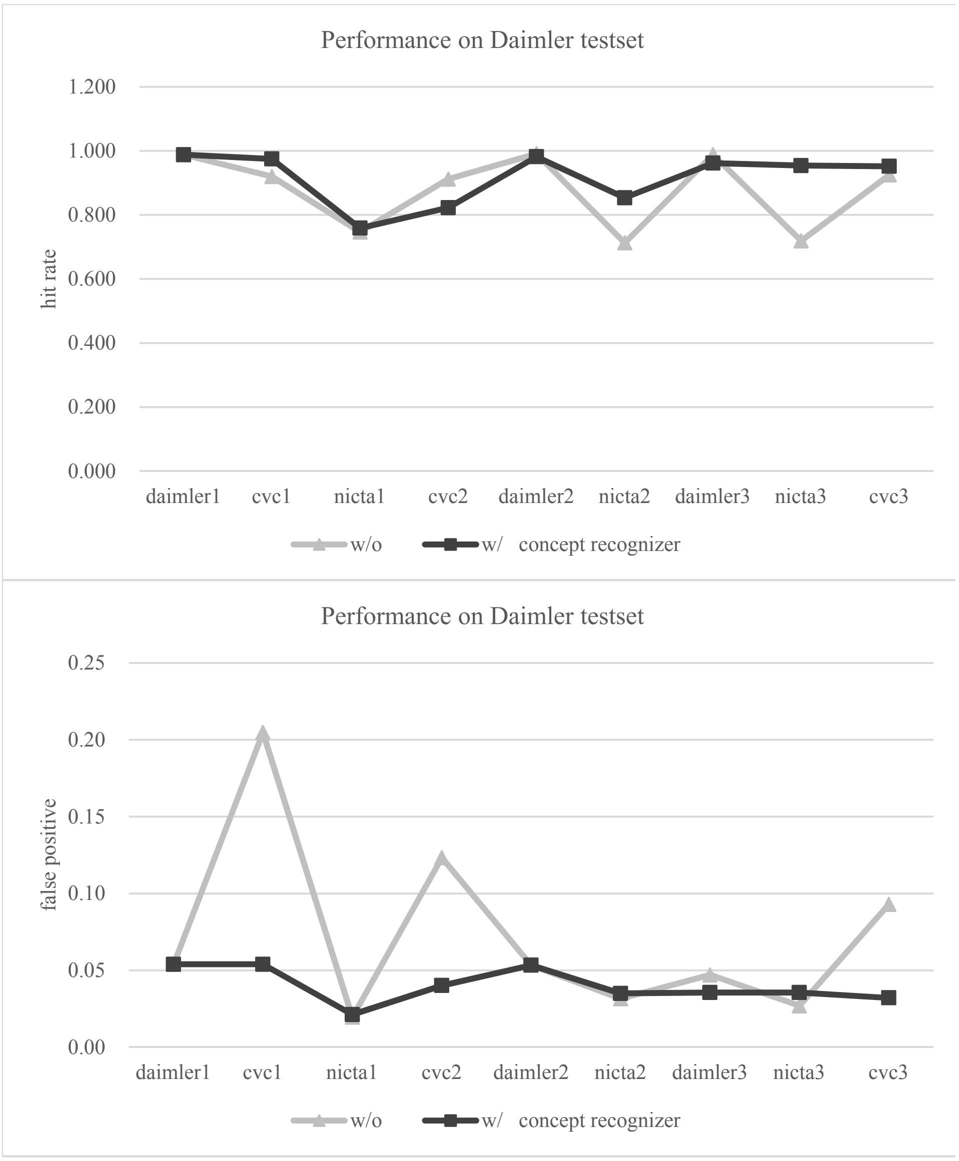


Figure 14. Performance variation on Daimler test dataset.

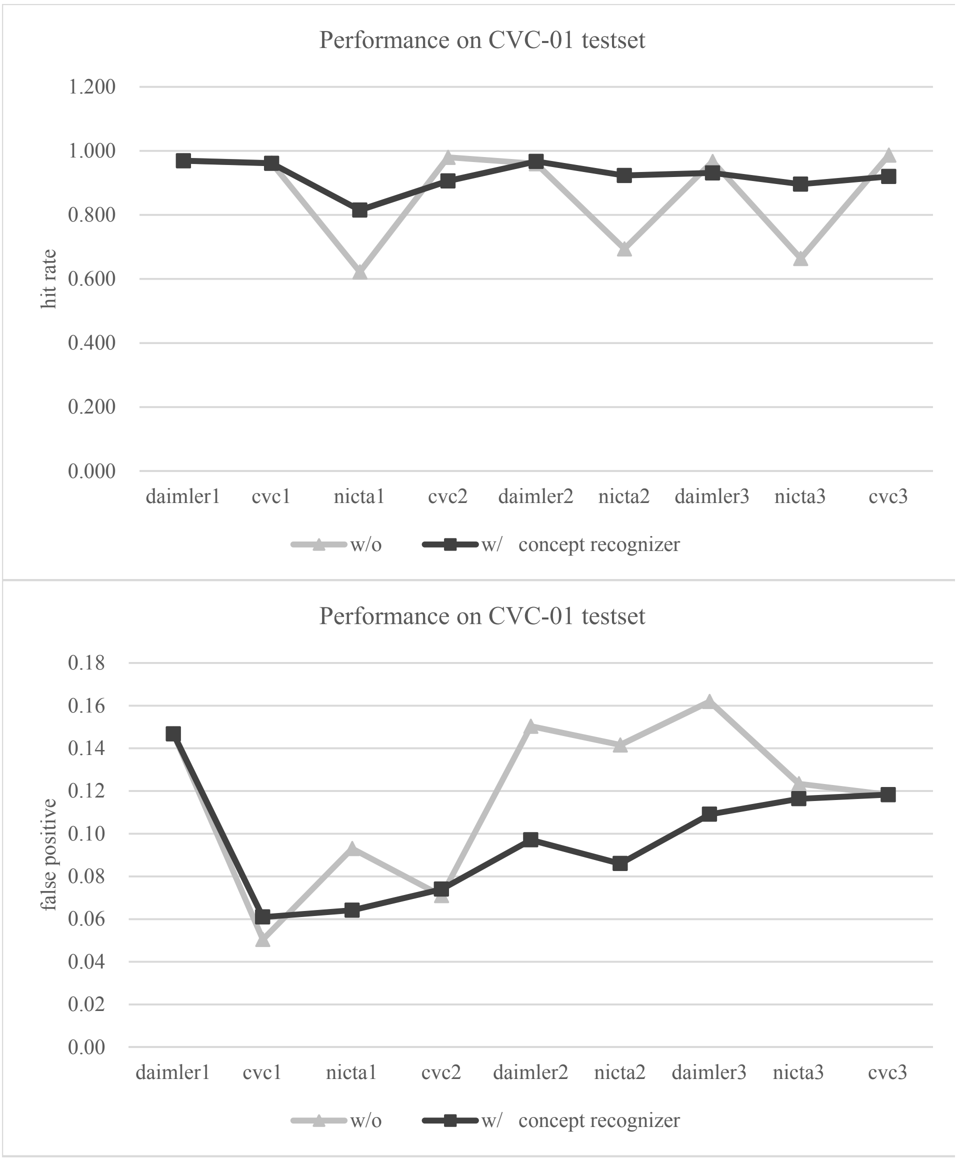
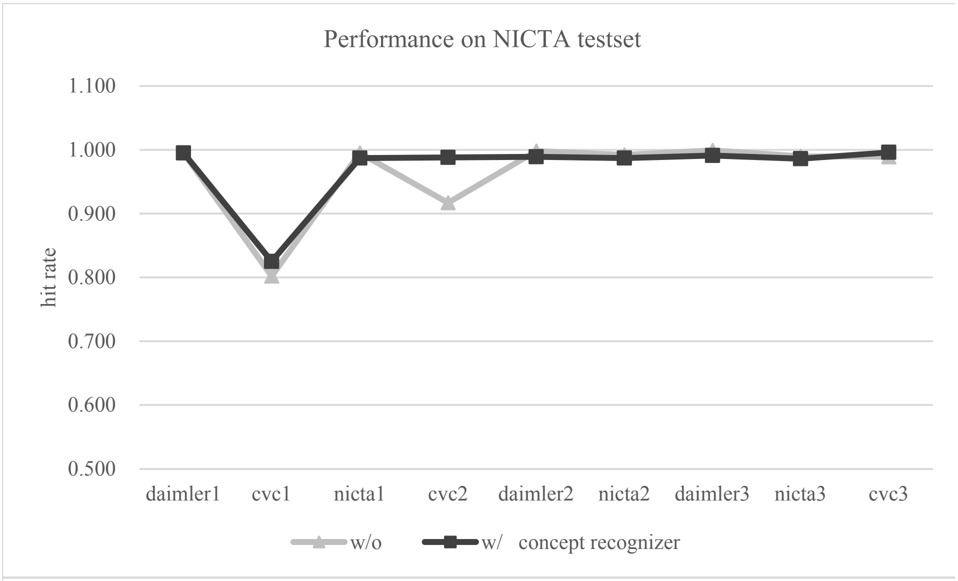


Figure 15. Performance variation on CVC-01 test dataset.



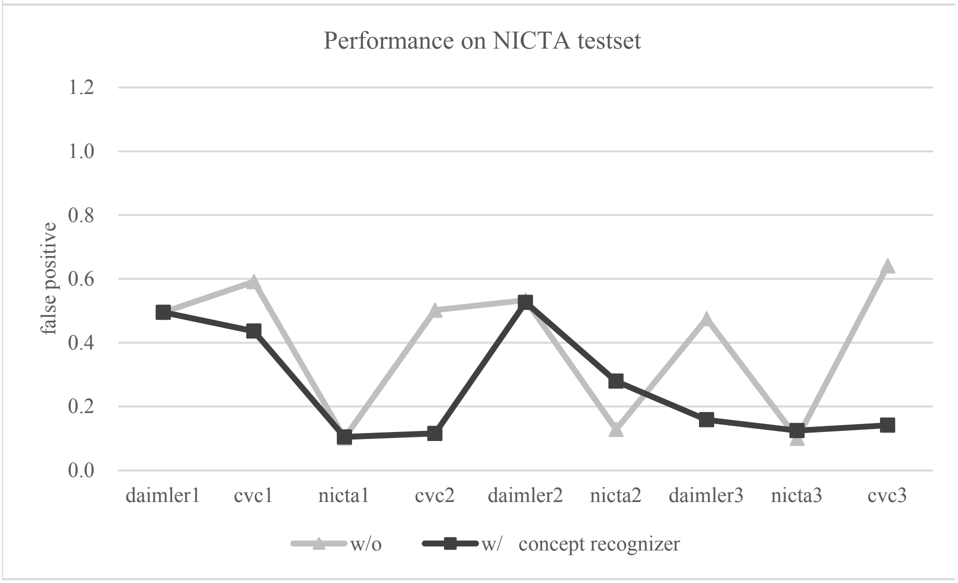


Figure 16. Performance variation on NICTA test dataset.

Chapter 6

Conclusion and Future Work

In this paper, I proposed an incremental learning and selective ensemble system for pedestrian detection. This system consists of two separate modules: front-end concept recognizer and back-end pedestrian detector. In the proposed approach, I adopt Learn++.NSE algorithm, which is efficient in non-stationary environment, to make the pedestrian detector to do incremental learning as well as follow the change of the environment. And I also adopt Learn++.NC, which is superior in adding new classes to make the concept recognizer to effectively learn new concepts. The limitation of the pedestrian detector based on pure Learn++.NSE is that it cannot adapt to several environments simultaneously. Through complementing the weakness by adding a concept recognizer to detect the concept before the pedestrian detection, the system retains not only good performance on old environments, but also the focus on the current one.

Although the system can incrementally learn new information, the training still needs manual intervention because it requires correctly labeled training datasets. Utilizing unsupervised learning for the bagging of these images can be a solution, but it also needs to be complemented by the method for learning with uncertainly labeled data. In addition, to train the concept recognizer, it needs data from at least two different concepts. In our work, I assume the environment does not change so suddenly thus there are still some data that represent the former concept and these data are labeled with our intervention. In the future work, if the system can be made to automatically divide training data into different concepts, it will make the system totally free from manual intervention.

Bibliography

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

[2] R. Elwell, and R. Polikar, “Incremental learning of concept drift in nonstationary environments,” *IEEE Trans. Neural Netw.,* vol. 22, no. 10, pp. 1517-1531, Oct.2011.

[3] V. Vapnik, *The Nature of Statistical Learning Theory,* Springer, 1995.

[4] Y. Freund and R. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *J.Computer and System Sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[5] N. Dalal and B. Triggs, “Histogram of oriented gradients for human detection,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, vol. 1, pp. 886-893, 2005.

[6] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *PAMI*, pages 1863-1868, 2006.

[7] D. Geronimo, A. Sappa, A. Lopez, and D. Ponsa, "Adaptive image sampling and windows classification for on-board pedestrian detection," *Proc. Inter. Conf. on Computer Vision Systems*, 2005.

[8] G. Overett, L. Petersson, N. Brewer, L. Andersson, and N. Pettersson, "A new pedestrian dataset for supervised learning," *Proc. IEEE Intelligent Vehicles Symposium*, 2008.

[9] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *Proc. IJCV*, 38(1):15-33, 2000.

[10] L. Oliveira, U. Nunes and P. Peixoto, “On exploration of classifier ensemble synergism in pedestrian detection,” *IEEE Trans. Intelligent Transportation Systems,* vol. 11, no. 1, 2010.

[11] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, “Learn++: An incremental learning algorithm for supervised neural networks,” *IEEE Trans. Syst., Man Cybern. Part C: Appl. Rev.,* vol. 31, no. 4, pp. 497-508, Nov. 2001.

[12] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, “Survey of Pedestrian Detection for Advanced Driver Assistance Systems,” *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 32, no. 7, 2010.

[13] T. Ahonen, A. Hadid, and M. Pietikinen. Face recognition with local binary patterns. In *ECCV,* pages 469-481, 2004.

한글 초록

고급운전자보조시스템에서 여러 개의 분류기로 구성된 앙상블 시스템은 각각의 분류기가 다른 분류기의 부족 점을 보완해주는 그 시너지 효과로 인하여 각광을 받고 있다. 다른 한 방면으로는 보행자의 다양한 포즈와 변화가 많은 배경 때문에 한번 학습을 통한 보행자 탐지기는 현실세계 환경 변화에 따라 지속적인 조율을 필요로 한다. 때문에 이러한 보행자 인식 시스템은 기존에 학습한 지식을 잃지 않는 정황하에 새로운 지식을 추가적으로 습득할 수 있는 능력을 요구한다.

이 논문에서는 여러 개의 분류기로 구성된 보행자 탐지기와 시스템의 앞 단에 위치한, 환경의 변화에 따라 비효율적인 분류기를 자동으로 끄는 역할을 하는 환경 탐지기를 결합하여 증진적으로 학습하고 환경의 변화에 더욱 동적으로 적응하는 보행자 인식 시스템을 제시하였다. 검증 된 증진학습 알고리즘을 적용하여 새로 추가되는 데이터를 분류 알고리즘으로 학습시키고 새롭게 생성된 분류기는 기존의 앙상블에 결합되며 각 분류기는 기존까지의 성능에 근거하여 현재의 환경에 가장 적합한 투표 권한을 가지게 된다. 이러한 투표권한 갱신은 매번 새로운 학습으로 분류기가 생성될 때마다 이루어지며 현재 환경에서의 성능에 최적화 되도록 갱신되기 때문에 현재환경과 과거환경 사이에서의 성능 트레이드 오프는 불가피하다. 이러한 문제는 시스템의 앞 단에 환경 탐지기를 추가하여 보행자 후보마다 적합한 분류기를

선별하고 탐지된 환경에 적합한 투표권한을 부여함으로써 시스템의 기동력을 높임으로써 해결하였다. 이러한 앞 단의 환경 탐지기의 개입으로 인하여 시스템은 현재 환경에 집중하면서도 과거 환경에 대한 좋은 성능을 잃지 않게 되었다.

주요어: 증진적 학습, 앙상블 분류기 시스템, 보행자 인식, 고급운전자 보조시스템.

학번: 2012-22568