Active learning for visual object detection

Yotam Abramson
Laboratory of Robotics
Ecole des Mines de Paris
60 bvd saint michel
75006 Paris, France
abramson@ensmp.fr

Yoav Freund
Center for Machine Learning.
Columbia University
New York, NY 10027
freund@cs.columbia.edu

abstract

One of the most labor intensive aspects of developing accurate visual object detectors using machine learning is to gather sufficient amount of labeled examples. We develop a selective sampling method, based on boosting, which dramatically reduces the amount of human labor required for this task. We apply this method to the problem of detecting pedestrians from a video camera mounted on a moving car. We demonstrate how combining boosting and active learning achieves high levels of detection accuracy in complex and variable backgrounds.

1. Introduction

The use of machine learning methods for visual object detection has become popular in recent years [7, 4]. It is now generally accepted that detection algorithms developed using machine learning are more accurate, more robust and faster than hand-crafted ones.

However, for these methods to work well, one needs large amounts of labeled training data. It is becoming increasingly evident that the manual work involved in hand labeling images has become a major factor in the time it takes to develop an accurate object detector. Levin et al. [3] used co-training in the context of visual car detection to improve the accuracy of a classifier using unlabeled examples. In this paper we use active learning in the context of visual pedestrian detection to concentrate the human feedback on the most informative and hard to classify examples.

An important observation in this context is that while it is relatively easy for a human to identify pedestrian in images, it is very hard to identify those areas of the image that are not pedestrians but look "almost" like pedestrians, however, in order for the learning algorithm to clearly identify the boundary between pedestrians and non-pedestrians it is critical to have examples that are close to the boundary. The location of the boundary between pedestrians and non-pedestrians in image space depends on the type of classification or scoring function that the learning algorithm uses and changes as learning proceeds. It thus makes common sense to select for labeling those examples that are close to the boundary defined by the detection function generated by the learning system *in its current state*. As the function gradually becomes more accurate the training examples should be chosen to be increasingly closer to the decision boundary and should gradually become harder to classify for the human. Thus making their labels increasingly informative to the learning process.

While this makes sense on an intuitive level, it is hard to justify it when one is working with a *generative* models for images as it creates a heavy bias in the data sampling process. It is easier to justify selective sampling when the goal is to learn a classification function, rather than a distribution.

In this paper we present the SEVILLE (SEmi automatic VIsuaL LEarning) system for active learning. This system is based on AdaBoost [2] and is similar to the work of Viola, Jones and Snow [7, 8]. The novelty of this work is in the substantial reduction of manual labeling achieved by active learning through selective sampling of the training set.

The paper is organized as follows. Section 2 describes the system and the input data. Section 3 presents the way we designed our experiments, followed by theoretical justifications in section 4. The experiments themselves are described in section 5.

2. The setup

We used quarter PAL (384x288 pixels) color video sequences, at 25 frames per second, that were taken from a forward-looking camera mounted on a car. The video was recorded as the car was driven on variety of rural and urban roads in and around Paris. The overall length of the recorded video is about 6 hours, from which we extracted sequences of variable duration ranging from several seconds to several minutes each. It will be noted that these sequences were cut from the parts containing pedestrians,

which is about 15% of the entire data.

The color images in our data were transformed to grey-level images in order to reduce computation. We define detection candidates to be images 48 pixels high and 24 pixels wide. We define a candidate to be a positive detection of a pedestrian if it contains the complete body (possibly partially occluded) with margins of approximately 20% of the body size around it.

The video data contains approximately 1500 different pedestrians. When using these images of these pedestrians for training, we usually took one image out of 3-4 consecutive frames, in order to avoid using examples that are too similar to each other.

In the images that contain pedestrians, usually 1-4 appear at the same time. Approximately third of the pedestrians are walking on sidewalks, parallel to our car, where the rest are crossing the road, mostly on crosswalks. It will be noted that crossing pedestrians are often using various diagonal crossing directions, so our database covers all viewing angles fairly well. The scenes contain various lighting conditions and various types of complex background.

After training we get a candidate scoring function. The system uses this scoring function to perform detection on new frames as follows. In each frame we consider around 400,000 candidate locations, starting with the minimal size (24x48) and scaling up by 10% in both dimensions 19 times.

We defined a horizontal line - the scene horizon - and use as candidates only rectangles that intersect this line. This reduces the number of candidates per image to around 170,000. We then identify all candidates whose score is higher than a preset threshold. Typically, there is a set of similar candidates that achieve high scores. We reduce these to single candidate by choosing the one that gets the maximal score.

2.1 The visual features

There are many types of visual features available for binary-classifying an image, such as the wavelet feature introduced by Viola and Jones [7]. We use the control-points features suggested by anonymous in *Anonymous*. These features have the advantage of being extremely simple and fast to compute. More details on these features are given in the attached supportive material.

2.2 The learning algorithm

The learning algorithm we used is Adaboost [2] based on single-feature decision stumps. We use the same learning procedure as Viola and Jones [7], with the only difference being our choice of features. Each feature f_i is a binary function of the candidate image and outputs +1 or -1. The total score generated by Adaboost is *normalized* using the sum of the feature weights, $s = \sum_i \alpha_i f_i / \sum_i |\alpha_i|$, so that the total score always ranges between -1 and +1. Normalization is important when one wants to use the boosting score as a measure of confidence as analyzed by Schapire et. al. in [5].

Each boosting iteration involves searching for a feature that has small error on the *weighted* training set. Since it is too computationally expensive to check all possible features, we use a heuristic method based on genetic programming. Our search procedure starts with a first "generation" of 100 random features, and uses various types of crossovers and mutations in order to improve the features to yield minimal weighted error on the training set. Our search stops when no improvement is obtained during 40 consecutive generations.

2.3 The interactive labeling system

Our experiments were carried out using a software system named SEVILLE which provides a graphical user interface for interactive labeling of training examples. The system is playing an input video sequence and displays on it those detections whose scores at within a specified range (more on the choice of range in Section 3).

These detections can be selected (individually or by groups) and be turned into learning examples by labeling them as negative or positive. The system allows the same treatment for rectangles which are marked manually (i.e. by dragging the mouse). Finally, if there is a section of the film with no pedestrians, the user can instruct the system to add all detections as negative examples. It is important to note the discrepancy between the time that it takes a typical user to generate an example using each of these modes. A manual marking of a rectangle by the mouse, exactly positioned on the pedestrian, takes in average approximately 20 seconds; a selection of a presented rectangle takes in average around 3 seconds; and an automatic mass-collection of negative examples from pedestrian-free sequence takes just about $\frac{1}{30}$ of a second.

The idea is that initially we build a rough and inaccurate detector, which is not good enough for our final goal but is sufficiently good to help in the example collection process. After collecting additional examples, we rerun the boosting algorithm and generate a more accurate detector. The more accurate the detector, the more effective it is in identifying those examples that are hard to separate. The role of the human is then to label borderline example while the vast majority of examples are eliminated from human consideration, thus drastically reducing the human work load.

3. The design of the experiment

Most machine learning procedures consist of four steps: data collection, labeling, training and testing. As we dis-

cussed earlier, in visual detection problems, labeling is a very labor intensive step. The goal of our experiment is to demonstrate how an iterative procedure which alternates between training and labeling can be used to substantially reduce the work involved in labeling.

In order to ensure an unbiased measure of the test accuracy of our detector we started by marking 215 ground-truth locations of 21 different pedestrians, chosen from a video sequence of 2:08 minutes. The marked images contain 37,064,791 non-pedestrian sub-windows. We later use this data-set to measure the ROC of the detectors generated by our process.

Each step in our procedure consists of two phases: collection of labeled examples and training. The training phase uses *all* of the labeled examples collected so far, separates them into 2/3 for training and 1/3 for validation and then runs the boosting algorithm on the training set until the ROC curve on the validation set stops improving.

Each labeling phase (other than the first) uses the detection scoring function generated by the training phase of the previous step. Each labeling phase uses a new, previously-unused video sequence. When applying the detector, the system is instructed to show detections whose boosting score (a number ranging between -1 and 1) is higher than μ^- and lower than μ^+. These thresholds are chosen manually, such that approximately 0.1% of the negative examples and around a half of the positive examples are shown.

As the movie segment is presented along with the detections in the chosen range, the user uses the interactive labeling system to label detections and to turn them into new training examples. Some detections can remain unlabeled if is not clear if they are positive or negative. All the labeled examples are added to the training set and the following training phase is executed.

4. Theoretical justification

The theoretical justification of considering the normalized boosting score $s = \sum_i \alpha_i f_i / \sum_i |\alpha_i|$ as a measure of prediction confidence was provided by Schapire et al in [5]. Intuitively, the normalized score value of a random test example varies only little if we select different training sets *from the same distribution*.

However, in our application we take one step further and assume that we get an accurate classifier even if we significantly alter the distribution of the examples by selecting mostly those examples that are close to the decision boundary. Note that if we were creating a generative model of the data, then such skewing of the distribution is likely to introduce significant bias into our classifier. However, our goal is to minimize classification error, not to maximize model likelihood, which is the reason that our learning algorithm can tolerate such skewing of the distribution of examples.

This is not true in general, but seems to hold when the initial unbiased sample is sufficient to restrict the set of good classifiers to a unique approximation of the optimal classifier. In other words, the distribution of the scores generated by Adaboost is such that all convex combination of the weak rules that are significantly different from the one we found have a much higher error. As a result, we can eliminate these significantly different classifiers from consideration and concentrate on making small adjustments to the classifier that we found. This in turn means that we only need to know the labels of examples that are close to the decision boundary, as only those will change label as a result of such small adjustments.

As we shall see in the experimental results. This condition indeed holds in our case and so the the method works.

Of course, the ultimate proof of the effectiveness of our method is given provided by considering the error of the final classifier on held out test data.

5. Experimental results

We ran ten steps of labeling and training, using the control-points features. We stopped after the tenth step because no further improvement was observed on an independent test set (see subsection 5.1). Until the final version of this paper is due, we hope to add more steps using motion features. We believe that motion features can continue where static features could not.

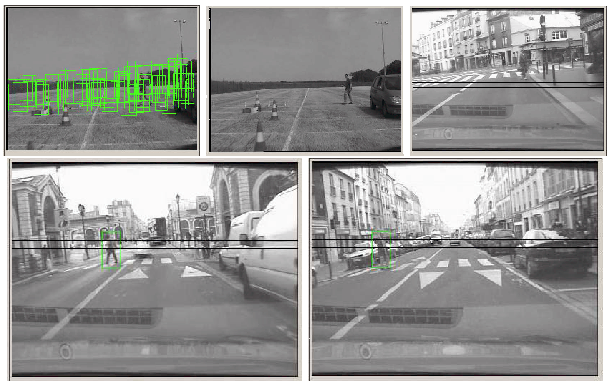

Figure 2: Various images from the process. Top left: massive false detections on step 2. Top middle and right: the simple and complex sequences used in the initial and mature steps, respectively. Bottom: results of the classifier in step 8 on two independent test sequences.

The development of the process in the different steps is shown in table 4. The most important column in the table is, perhaps, the amount of human labor. This can be seen by the large difference existing in the table between steps 6 and 7: while at step 6 we labeled more than 5000 examples in 1.5 hours, at step 7 we needed twice the time to label less

Step	total candidates	μ^-	μ^+	presented	labeled	human labor	positive	negative	training time	Weak rules
1	510 K	-	-	0	16	3m	6	10	2s	1
2	680 K	0	1	364	403	3m	36	374	6s	3
3	3,400 K	0.6	1	153	156	4m	46	520	22s	7
4	66,470 K	0.4	1	805	852	10m	86	1332	1m30s	30
5	37,910 K	0.1	0.8	1350	1439	10m	182	2675	8m	59
6	116,960 K	0	0.6	5150	5364	1h30m	417	7804	1h10m	270
7	24,140 K	-0.02	0.5	1320	863	3h	848	8236	7h30m	893
8	189,550 K	-0.02	0.5	8690	8707	3h	1178	16613	17h	1500
9	209,610 K	-0.02	0.5	2933	2933	3h	1486	19238	30h	2034
10	274,210 K	-0.02	0.5	3861	3861	4h	2046	22533	30h	3150

Table 1: The details of the training steps. Columns from left to right: Sequential number of step, total number of candidate images in the considered video input data (only frames which were actually used), lower and upper bound used for displaying the "hard" examples, number of such candidates presented to the user as "hard" (having a boosting value within the defined interval), number of candidates labeled by the user (including manually-collected rectangles that were not presented by the detector), time spent on the human labor of collecting these examples, *cumulative* number of positive and negative examples collected, computer time spent on the training session, number of AdaBoost cycles (weak rules) needed to produce a detector for the next step. Note that in the first step μ^- and μ^+ are not relevant because a detector is still not found.

than a thousands. A closer look in the tables reveals, that in step 7 we labeled more positive examples, and in step 6 we concentrated mainly on negative ones.

Additional insight into the active learning process can be gained from looking at the type of mistakes that are dominant after each step. In figure 1 we show the positive examples with smallest scores and the negative examples with largest scores as a function of the step number. During our experiments we identify three types of steps:

Steps 1-3 can be considered as "startup steps", where no valuable detector is available, positive examples are collected mostly manually, while negative examples are collected from the numerous false detections (see figure 2, top-left). In these steps we used simple and short sequences (see figure 2, top-middle), where the goal is only to obtain a stable detector. The mistakes made during the startup steps are trivial.

Steps 4-6 can be described as intermediate steps, where a valuable detector exists and where quality of the examples produced is already much higher than that of examples collected randomly by hand. Also harder sequences are used (see figure 2, top-right). However, these steps are not yet "mature" in the sense that trivial negative examples are still appearing and pedestrians in the new sequences are not always detected. The mistakes that are made during these steps are on image frames that contain complex backgrounds.

Steps 7-10 are mature steps, where μ^- and μ^+ are stabilizing around -0.02 and 0.5 respectively, examples are hard to separate and most of the new pedestrians are detected. The mistakes made by the detector on these steps are much more subtle. Some of the false positive examples in steps 6-10 are mistakes that might be made by a human (using this restricted view and no movement or color information). Some of these look pedestrians, and others are actual pedestrians but the image is not well positioned in the detection frame.

5.1 Quantitative results

We tested the detectors obtained by the different steps on the test data. The resulting ROC curve is shown in figure 4. One can observe that the detector improves as the process advances. Some successful detections from the test data are shown in figure 2. We can see that significant gains are made in the false positive rates until step 9, at that point the system detects 70% of the pedestrians with one false positive for approximately 10,000 candidates examined. In step 10 we get slightly worse results even though we added examples and the validation error keeps improving (see support material). A possible explanation is that we have reached the point where we have fully exploited our set of features and the differences that we see are a result of the limited span of the test set.

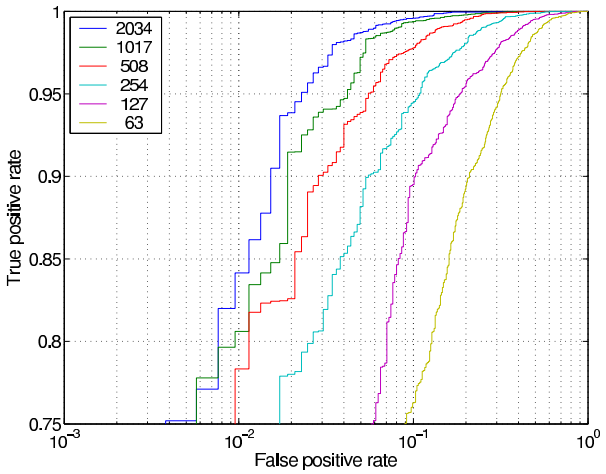

Figure 3: The validation-set ROC of the detector generated in step 9 as a function of the boosting cycle.

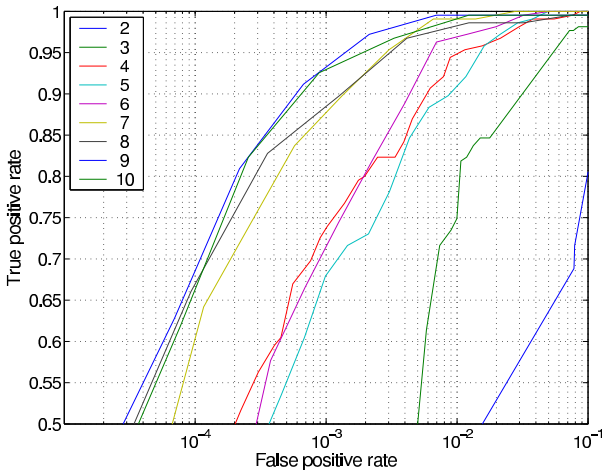

Figure 4: The test-set ROC of the detectors generated in each of the 10 steps.

Our results are comparable with previously obtained results for gray-scale static images [8] [4]. Obviously, for a practical system one must extend the input space and include features like color, movement and possibly context [6]. The process described in this paper makes it easier to select these additional features.

In figure 3, the ROC curves for the validation set show that boosting makes continual progress throughout the 2034 cycles of the 9th step. Note that the final validation ROC is much worse than the test ROC because the validation set, like the training set, is biased towards hard examples.

6. Conclusions and future work

We have presented a method for graduate collection of high quality training examples. In the results of our experiments, it is clearly seen that the examples we collect are becoming more and more hard to separate. In the experiments, we did not use all the input data which was available. It is interesting to observe the development of the process in further steps.

False positives generated in mature steps can guide us in choosing which features to add in order to further improve the detector's accuracy. These might be more complex features such as wavelet features as used by Viola and Jones [7], or motion features as used by Viola et al. [8]. While these features are more computationally expensive, we can use the cascading method of Viola and Jones to restrict their calculation to the rare cases where they can make a difference. In this way we believe we can add complex features to achieving high accuracy without significantly increasing the average detection time.

Relations between consecutive frames can be also used to improve performance by estimating camera movement and using it to smooth detections in consecutive frames, and using movement-based stereoscopy to estimate distance and therefor approximate size. We plan to include motion-based analysis in the final version of this paper.

References

[1] Anonymous. Illumination-independant visual features. submitted to CVPR '05.

[2] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37, 1995.

[3] Anat Levin, Paul Viola, and Yoav Freund. Unsupervised improvement of visual detectors using co-training. In *Proceedings of the International Conference on Computer Vision*, 2003.

[4] M. Oren, C. Papageorgiou, P. Sinha, E. Osuna, and T. Poggio. Pedestrian detection using wavelet templates, 1997.

[5] Robert E. Schapire, Yoav Freund, Peter Bartlett, and Wee Sun Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

[6] A. Shashua, Y. Gdalyahu, and G. Hayon. Pedestrian detection for driving assistance systems: Single-frame classification and system level performance. In *Proc. of the IEEE Intelligent Vehicles Symposium (IV2004)*, Parma, Italy, June 2004.

[7] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 2002.

[8] Paul Viola, Michael J. Jones, and Daniel Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE International Conference on Computer Vision*, pages 734–741, Nice, France, oct 2003.

Figure 1: Positive (left) and negative (right) examples for the different steps (each line is a step, from top to bottom). These examples received the minimal (for positive examples) or maximal (for negative examples) score during classification in their respective step.