Cascade-Dispatched Classifier Ensemble and Regressor for Pedestrian Detection

Anonymous AVSS submission for Double Blind Review

Paper ID 1

Abstract

This paper focuses on ensemble classifiers for pedestrian detection. Ensemble learning is widely used in this field for context disambiguation or via a cascade-of-rejectors. However, applying the typical, parallel, instance of it remains disappointing in most cases. Our work studies the mechanisms that hinder the efficiency of ensemble classifiers for pedestrian detection, and, based on our findings, we introduce a structured classifier ensemble that improves performance without loss of speed. We also harness this principle for context disambiguation via the application of a regressor to pedestrian detection. Experiments on the INRIA and Caltech-USA datasets validate the approach.

1. Introduction

Despite the constant advent descriptor efficiency, single classifiers can only help distinguish pedestrians from the vast amount of possible negative samples to some extent. In particular when faced with small or blurry detections, and cluttered scenes, the amount of available information one can harness for this task gets scarce or the proportion of outliers is important. **[6, 39]** further confirmed the impact of this shortcoming on object detectors (for gradient features only).

Therefore, to compensate for the inability of individual classifiers under these circumstances, one can consider the usage of ensemble classifiers, each component classifiers dealing with a lower variability of data. So far, classifier ensembles have been used for pedestrian detection in two cases: cascade-of-rejectors and context enhancement.

The cascade-of-rejectors **[5, 38]** is a widely used technique that consists of successive classifiers that iteratively filter out more complex negative samples. The second type of method uses context information to disambiguate appearance inputs between pedestrian and the background. This term covers common sense arrangement of things that we all learned through experience. For instance, "the sky is above pedestrians" or "a pedestrian cannot have the size of a car tire". The computer vision learning of this objects and things coherent composition is based on interactions among objects **[24, 16]**, or scene statistics **[14, 28, 2]**.

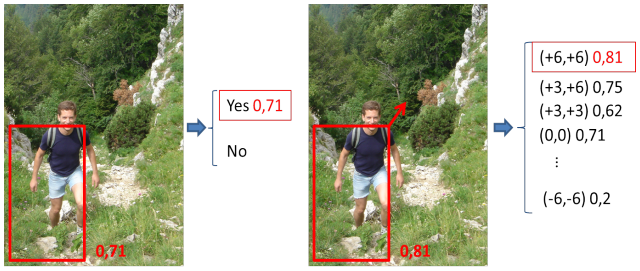

Figure 1: Difference between a typical classifier (left) and a classifier based on regressor (right): A regressor give an estimate of the detection position. The analyzed detection is in red. The confidence value associated to each detection is displayed next to the bounding box. Best viewed in color.

However, only few approaches **[6, 33, 34, 22]** learn it without use of pre-defined rules or database knowledge, and, despite the clear need for a more generic usage of classifier ensembles for pedestrian detection, most of them show a performance increase with simple features only and at the cost of significant extra computation.

This work gives a first answer to these moderate state-of-the-art results. We highlight that the necessary partitioning of the original training set on which the component classifiers are trained reduces their performance accordingly. As a consequence, we introduce a new structured ensemble learning to tackle this issue. Our algorithm expands each component classifier training set to deal with the data shortage issue and runs at the same speed as a typical cascade-of-rejectors.

We also propose an automatic context learning method based on a regressor **[41]**. A regressor is an ensemble of classifiers learnt with a predefined offset from the optimal object position. Therefore, each component classifier provides an estimate of the detection position instead of a simple binary Yes/No answer. As such, a bunch of them offer an automatic way to identify confusing candidates and incorporate them in the training set to optimize detector capabilities. Figure 1 illustrates this idea.

To summarize, our contribution is two-fold. First, the upgrade of the classifier cascade into a structured classifier ensemble that improves performance without loss of speed. Second, the use of a regressor for pedestrian detec-

tion within a Bayesian network to disambiguate false from true positives and give a better localization of the laters.

The rest of this paper is organized as follows. Section 2 reviews the related work while Section 3 gives an overview of our pedestrian detector pipeline. Section 4 and 5 respectively motivate and detail our classifier ensemble approach and the use of a regressor for pedestrian detection. Section 6 proves their efficacy through experimentation. Section 7 presents our conclusion and future directions.

2. Related Work

Ensemble classifiers can broadly be broadly categorized in three groups: *serial-* or *cascade-classifiers*, *parallel-classifiers*, and *mixed-* or *structured-classifiers*.

Serial classifiers are also called *cascade-of-rejectors* **[5, 38]**. The cascade-of-rejectors is an efficient strategy for sieving out a large proportion of false positives. Despite its improved performance, tuning this cascade of classifiers remains problematic. First, the classifiers are prone to errors that propagate along the cascade. Second, most authors employing this technique solely reject negative data after each classification step to avoid the creation of classifiers likely to reject positive data during the subsequent stages, therefore leading to progressively more imbalanced datasets. Fine tuning of the original positive versus negative instances ratio is usually necessary to obtain satisfactory results. But balancing the dataset is proved **[44]** to be the best performing strategy. To deal with these issues **[15]** proposes a two-stage classification: The first one aims for a high recall whereas the second one focuses on precision.

Parallel classifiers combine a set of classifiers, called *weak classifiers* into a more accurate *strong classifier*. They heavily rely on two factors: The selection of appropriate subsets of data on which to build the weak classifiers and the type of strong classifier utilized **[30, 29]**. Its use for pedestrian detection has so far mostly been restricted to context disambiguation via the integration of pre-defined rules or database knowledge. The first one consists of semi-supervised user-identification of disturbing elements, such as crowds **[24]** or cars **[16]** that are further integrated in the training process thanks to independent classifiers. The second type relies on perspective information like the integration of ground plane information, either through direct hyperplane estimation **[14, 28]**, the use of stixels **[2]**, or occlusion detection mechanism based on ensemble learning **[21]**. See **[13]** for a review of context based techniques for object detection.

So far, few methods **[22]** automatically learn the implicit relations between pedestrians and their various environments. **[6]** simply added some background pixels around the detection window. **[33]** proposed an efficient regression technique based on background patches to model probabilistic priors for pedestrian locations. **[34]** designed the Local Response Context (LRC) that concatenates detections around the window of interest in a single descriptor and utilizes partial least square to determine each component respective importance for classification. Therefore, parallel ensemble classifiers do not offer better performance than the cascade-of-rejectors whilst the necessary test of all classifier for any pedestrian proposal make them computationally slower.

Structured classifiers can be viewed as a combination of the previous two techniques. The pedestrian detector is decomposed of several successive stages, each one of them making use of several parallel classifiers. To the best of our knowledge, few papers fall into this category. **[43]** proposed a tree-structured set of classifiers, but their limited training data, outdated Haar features and home-made dataset doesn't make the comparison with current methods meaningful.

3. Method pipeline

Our training protocol, illustrated in figure 2, extends a cascade-of-rejectors of n classifiers into a structure ensemble of classifier featuring k classifiers at each stage of the cascade. In addition, each independent classifier is learnt through bootstrapping **[6, 12]** to improve performance. The issue of imbalance between the positive and negative data that stems from the use of the cascade is solved by integrating the *FairTrain* **[35]** training set generation protocol in our pipeline. Given a dataset D, and its subsets of positive and negative data *Pos(D)* and *Neg(D)*, this methodology aims to enforce the following constraint for all classifier training sets:

$$|Pos(D)| = |Neg(D)| \quad (1)$$

More practically, it decomposes in two distinct parts: The initial training set generation and the classifiers training. First, the initial training set generation carefully selects data from a set of images while balancing negative and positive sample cardinalities. And then, the key aspect being to respect the constraint (1) all along the cascade, the minority class is oversampled at each stage. See **[35, 36]** for details on *FairTrain*. The regressor is applied at the end to prune out the hardest false positives. Its use as final classifier greatly limits the extra computation cost it may induce.

4. Classifier ensemble

This section describes the ensemble of k classifiers utilized at each stage of the cascade during the training and testing phases.

The main hypothesis motivating the training of our parallel classifiers (and demonstrated in the results section) is that there are two important factors that drive the effectiveness E of parallel ensemble classifiers within the field of pedestrian detection: The number k of component classifiers and the average amount of training data $|S|$ associated to each one of them. Clustering the training set with a higher number of partitions k should theoretically improve the results,

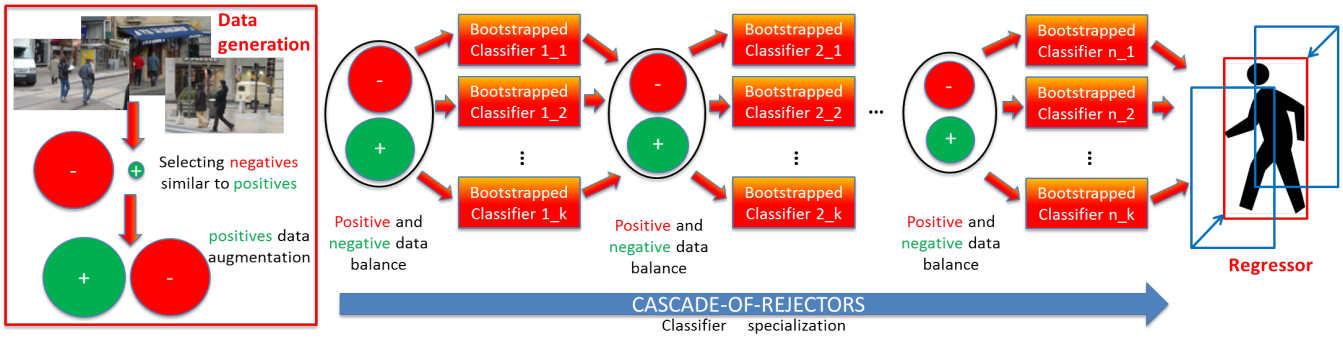

Figure 2: Ensemble classifier pipeline. Training pipeline. The initial training set generation selects data while balancing negative and positive sample cardinalities. A cascade of classifiers is then trained on it, each stage being composed of k parallel classifiers learnt through bootstrapping. balanced positive and negative sets is sought all along the cascade. Regressor classification is then applied. Each circle surface is proportional to the sets cardinality that it represents. Best viewed in color.

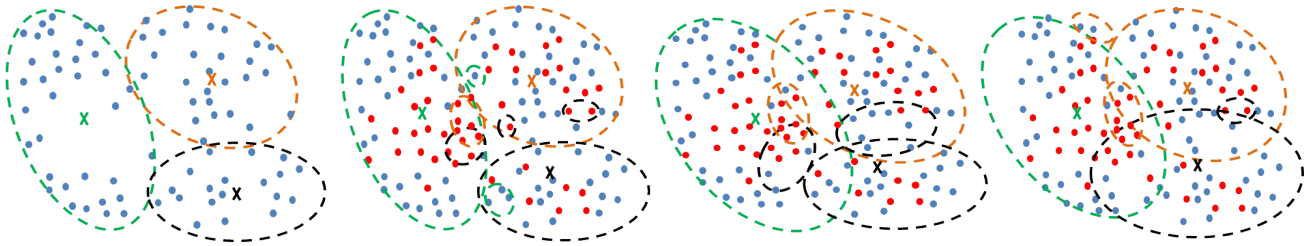

Figure 3: Toy example of data partitioning according to 3 clusters. Positive data are in red; negative ones in blue. The dashed circles represent the belonging to a given cluster and their centroid is depicted with a cross of the same color. Data points may belong to several clusters, leading to overlapping circles. Clusters may also be divided, leading to several circles of the same color. Best viewed in color.

but it also yields smaller partitions, and the lack of training data is detrimental to classifier performance. Therefore, this typical strategy leads to the following paradox: To increase E implies increasing k which implies reducing $|S|$ which diminishes E.

To tackle the issue, we augment the cluster cardinalities in several ways. We first adjust the parameters accordingly. We tune the initial dataset size up to 350K samples and limit the data loss from one stage of the cascade to another to 20%. Further expanding the initial dataset or reducing the shrinking factor beyond this point have no influence on the performance. Then, we adopt the subset creation strategy illustrated in figure 3:

1. The negative data are divided in k subsets according to a partitioning algorithm like k-means **[26]** or k-medoids **[17]**. Given a distance function d(.), the hard-coding variants of these methods typically partition data in k clusters represented with centroids μ and associates each datum x_i to one of them:

$$\mu = \frac{1}{|C|} \sum_{x_i \in C; i=1...|C|} x_i \tag{2}$$

$$\forall x \in C_i, \forall \mu_j, d(x, \mu_i) \leq d(x, \mu_j) \tag{3}$$

2. We comply with the rule (1): For each cluster, the closests positive data are included until a balance between positive and negative data is reached. The centroids are repositioned accordingly.

3. The Ω percent closest data points are added to the clusters to compensate for the training set shrink. The centroids are repositioned accordingly.

4. (Optionally) For each cluster, the closest data are added or removed to equalize the cardinality among them while respecting the constraint (1).

This strategy allows us to extract meaningful data partitions for training while keeping them populated enough to be discriminative. It first focusses on negative data in step 1 where the bulk of the data variability stands before to equalize the labels and inflate the clusters. It has the following noticeable properties:

1. One datum may belong to several clusters, leading to overlapping clusters.
2. Clusters may also be divided after step 2 depending on the position of the closest positive data.
3. The condition (3) is not valid anymore after step 2-3.
4. Setting $k = 1$ boils down to running a typical cascade-of-rejectors.
5. Only applying the step 1. on positive and negative data is the vanilla data clustering.

Let μ_{cl_i} be the centroids associated to a classifier cl_i and inversely cl_{μ_i} the classifier associated to the centroid μ_i.

During the testing phase, each new pedestrian proposal p is first associated to its closest centroid

$$\mu(p) = \mu_i | d(\mu_i, p) \leq d(\mu_j, p) \forall j \in [1...k] \tag{4}$$

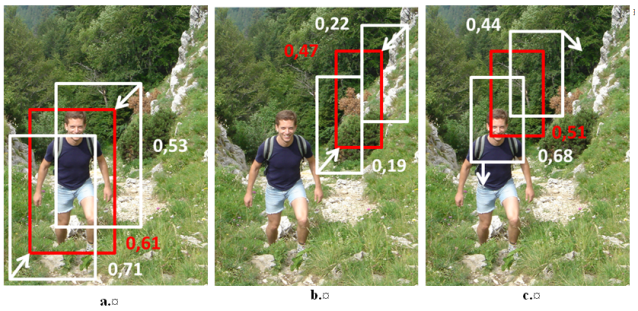

Figure 4: Regressor usage for pedestrian detection. The analyzed detection is in red, neighboring ones are in white. To each bounding box is associated its maximal confidence value and its corresponding regressor classifier (depicted by an arrow). Positive sample detections are reinforced by local detection displacement information (a.). Negative examples lead to low confidence scores (b.) or inconsistent positions (c.).

and only evaluated according to its associated classifier $P(p) = cl_{\mu_i}(p)$. This strategy leads to the use of only one classifier at each stage of the cascade no matter what the value of *k* is. Therefore, the computation time of our structured ensemble classifier remains the same as a standard cascade-of-rejectors.

5. Regressor

A regressor (not to be confused with mathematical regression) is a powerful tool for object detection related tasks. It was first introduced in **[41]**, and has mostly been used for object tracking **[41, 31]**. **[33]** employed a somewhat similar strategy to model pedestrian background and locate them in accordance to common landmarks (at the base of a building, on a road...) but they modelled background and pedestrian separately.

During regressor training, samples labelled with displacements from the target object $(\Delta x, \Delta y)$ are provided. During testing, unknown sample appearance is used to infer a possible nearby object position. Therefore, a regressor provides information about an object position instead of its sole occurrence. For example, given a regressor trained on examples labelled within the [(5, -5); (5, 5)] range, testing a sample located (-2,4) pixels away from the true target location will return (-2,4). This contrasts with typical classifiers outputting a binary Yes/No answer. This principle is illustrated in figure 1. The combination of local surrounding displacements may also indirectly model related deformations.

We apply this idea to pedestrian detection by considering the regressor as an ensemble of classifiers. Given *m* regressor classifiers trained on *m* surrounding positions of the ground truth detection, we treat each regressor classifier applied to one of the *m* positions as a weak classifier. The strong classifier is obtained by fusing their $m \times m$ scores through a bayesian network.

More specifically, training is achieved by learning the set S of *m* classifiers $C(\Delta x, \Delta y)$ within the range $[(-X, -Y); (X, Y)]$. Each classifier $C(\Delta x, \Delta y)$ estimates the confidence of a pedestrian to be located $(-\Delta x, -\Delta y)$ pixels away from any given location (x, y). They are trained with positive data shifted with a corresponding displacement $(\Delta x, \Delta y)$ from the ground truth location, and the closest negative samples augmented with the *m* possible displaced data. The height and width of these augmented data remain the same as their original data. Note that $S = \{(0, 0)\}$ trains a vanilla classifier. During testing, each candidate is augmented by a set *D* of *m* nearby locations within the predetermined range $[(-X, -Y); (X, Y)]$. Their local appearance descriptor is extracted and each classifier is tested on it. This yields a score matrix C of [*m* positions $\times$ *m* classifiers] for each candidate. We then utilize a Bayesian network to fuse the outcome of each classifier into one single confidence value. Be $C(\Delta x, \Delta y, x, y)$ the output of the classifier with displacement $(\Delta x, \Delta y)$ on position (x, y). We first calculate the probability of a pedestrian to be located at each position P(i,j) as the combination of the classifiers indicating this position:

$$P(i,j) = \prod_{\substack{(\Delta x, \Delta y) \in S \\ (x,y) \in D \\ (\Delta x, \Delta y) + (x,y) = (i,j)}} C(\Delta x, \Delta y, x, y) \quad (5)$$

This probability is eventually normalized to insure the same number of classifier involved at each position (i,j). We then relocate the analyzed candidate (x,y) to its most confident location (x_{max}, y_{max}) such that

$$P(x_{max}, y_{max}) = max_{(i,j) \in D} P(i,j) \quad (6)$$

with new confidence value:

$$C(x,y) = \frac{max_{(i,j) \in D} P(i,j)}{\sum_{(i,j) \in D} P(i,j)} \quad (7)$$

It is assumed that, in the occurrence of negative samples, either most of the classifiers will display a low confidence, either no clear consensus on its position will arise, as shown in figures 4.b. and 4.c.

To limit the computation cost, the regressor is only used as final ensemble-classifier of the cascade, to tell apart true from false positives. A key parameter is the displacement $(\Delta x, \Delta y)$ at which classifiers are learned. A too small translation will lead to classifier appearances similar to the main candidate one and further confusions. On the contrary, a big offset will create an unreliable appearance model, covering too much background. The number of classifiers and positions *m* also influence precision and processing time.

6. Experiments

This section presents the datasets and the experimental setup details, and compares our work to the state-of-the-art.

6.1. Evaluation protocol

Candidates are selected using a multi-scale sliding window approach **[48]** with a stride of 4 pixels. Approximately 60% are loosely filtered according to "edgeness" and symmetry. No background subtraction or motion features are employed on the Caltech dataset.

The training set size is up-bounded at 12K samples. Increasing this threshold doesn't improve performance. No generative resampling is performed. The validation set includes 350K samples for all experiments. The model height is set to 120 pixels for the INRIA dataset, 100 pixels for Caltech.

Following the standard parametrisation **[35, 36]**, we trained a cascade of 6 Adaboost classifiers, with a tree depth of 2, and a constant threshold of 0.54. We utilised the openCV implementation for the Adaboost classifiers. Best performance is obtained with 256 weak classifiers, each extra run adding 64 weak classifiers.

We use the threshold variant of non-maximum suppression (t-NMS) **[4]**. We set the overlap threshold to 0.5 and $d = 3$ for all our experiments. Log-Average Miss Rate (LAMR) **[25]** is employed as metric for all runs.

Pedestrian candidates are described with the LBP-Channel features **[37]**, using the *full* setting for all our-experiments: For each detection proposal window, the LBP descriptor blocks are spaced out over 8×16 positions along the x and y axis, leading to 128 different possible positions. The possible filter dimensions w_j and h_k is restricted to $2W/7$, with W the detection window width. In addition to that, filters covering the head, the torso, the legs and the entire body are extracted based on edge templates **[46]**. With this setting, 133 filters, totalling 1596 feature values, are extracted per channel. L_1 square block as well as L_2 full histogram normalisation are performed in every case. We employed $\{(1,1); (1,2); (2,1); (2,2); (1,3); (3,1); (4,4)\}$ as cell dimensions and the following 3 channels: graylevel intensities I, gradient magnitude M, gradient along x and y axis G_x and G_y, as well as the L color channel of the Luv color space.

We experimented on the INRIA **[6]** and Caltech-USA **[10]** datasets.

6.2. Comparison with the state-of-the-art

Comparison with the state-of-the-art is presented in table 1. For fair comparison, methods using optical flow features **[47]**, are reported without optical flow or left out of this evaluation. We used the $\{k = 9; \Omega = 100\%; \Delta = 50\%; m = 8\}$ parameter set for these experiments.

While running at a similar speed, our multi-classifiers method greatly outperforms the common classifier-ensemble pedestrian detectors **[24, 16, 21]**, scoring respectively 13.2% and 35.8% log average miss rate on the INRIA and Caltech datasets. Overall, our method ranks on par or better compared to state-of-the-art pedestrian descriptors in terms of performance/speed ratio, except for **[45]**. The improvement of using our structured classifiers and regressor method over the LBP-channels **[37]** is 1.1% on the INRIA dataset and 3.3% on the Caltech dataset. Deep learning technique **[19, 32]** often perform better but are slower. The regressor gives better results on the Caltech dataset, which is sensible since, compared to the higher variation INRIA dataset, this benchmark always deals with the same kind of scenery and always features the same contextual outliers (car, signs, newspaper boxes...).

As a reference, on the Caltech dataset, the classifier training takes 34 hours on one core to learn 18 classifiers ($k = 3$), and 51 hours for 30 classifiers ($k = 6$) but with the appropriate memory resources, the process can easily be parallelised. If the training of so many classifiers is long, the detector speed at testing time remains the same as for a cascade of single-classifiers. Our detector test speed is, of course, largely influenced by the descriptor parameters, which are the number of filters and channels.

7. Conclusion

This work stressed out an classifier ensemble paradox: To increase parallel classifier efficiency implies increasing the number of weak classfiers which implies reducing the cardinality of each weak classifier set which diminishes parallel classifier efficiency. Through the design of a new structured cascade-of-rejectors, and introducing the regressor to human classification, this work tackles this issue and demonstrates that ensemble classifiers can be a useful assistance to the community for the pedestrian detection task.

The next stage is, of course, the application of this methodology to CNN features. Also, more experiments are required to clearly study the appropriate selection of data for parallel classifiers.

References

[1] A. Bar-Hillel, D. Levi, E. Krupka, and C. Goldberg. Part-based feature synthesis for human detection. *ECCV*, 2010.
[2] R. Benenson, M. Mathias, R. Timofte, and L. V. Gool. Pedestrian detection at 100 frames per second. *CVPR*, 2012.
[3] R. Benenson, M. Mathias, R. Timofte, and L. V. Gool. Pedestrian detection at 100 frames per second. *CVPR*, 2013.
[4] M. D. Buil. Non-maximum suppression. *technical report ICG-TR-xxx*, 2011.
[5] A. D. Costea and S. Nedevschi. Word channel based multiscale pedestrian detection without image resizing and using only one classifier. *CVPR*, 2014.
[6] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.
[7] P. Dollar, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. *BMVC*, 2010.
[8] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral channel features. *BMVC*, 2009.
[9] P. Dollr, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. *ECCV*, 2012.

INRIA		
Method	LAMR	Speed(CPU/GPU)
HoG **[6]**	46%	0.5fps
HoG-LBP **[40]**	39%	Not provided
MultiFeatures **[42]**	36%	$<$ 1fps
FeatSynth **[1]**	31%	$<$ 1fps
Channel Features **[8]**	21%	0.5fps
FPDW **[7]**	21%	2-5fps
DPM **[11]**	20%	$<$ 1fps
Occlusion handling **[21]**	19.13%	Not provided
RF local experts **[20]**	15.4%	3fps
PCA-CNN **[18]**	14.24%	$<$ 0.1fps
CrossTalk cascades **[9]**	18.98%	30-60fps
VeryFast **[3]**	18%	8/135fps
FairTrain+LBP **[35]**	17.8%	3.7/54fps
WordChannels **[5]**	17%	0.5/8fps
SSD **[19]**	15%	56fps
LBP-channels **[37]**	14.3%	0.55/8fps
structured cascade regressor	**13.3%**	**0.55/8fps**
struct cascade+regressor	**13.9%**	**0.51/7.5fps**
FRCNN **[32]**	**13.2%**	**0.51/7.5fps**
RPN+PF **[45]**	13%	7fps
	7%	6fps

CALTECH		
Method	LAMR	Speed(CPU/GPU)
HoG **[6]**	69%	0.5fps
DPM **[11]**	63.26%	$<$ 1fps
FPDW **[7]**	57.4%	2-5fps
Channel Features **[8]**	56.34%	0.5fps
multi-classifier **[24]**	43%	$<$ 1fps
MT-DPM **[16]**	41%	$<$ 1fps
JointDeep **[23]**	39.32%	$<$ 1fps
MT-DPM + context **[16]**	38%	$<$ 1fps
InformedHaar **[46]**	34.6%	$<$ 0.63fps
Spatial pooling **[27]**	29.2%	$<$ 1fps
Checkboards **[47]**	24.4%	$<$ 1fps
FRCNN **[32]**	56%	7fps
CrossTalk cascades **[9]**	53.88%	30-60fps
FairTrain+HoG **[35]**	45.4%	3.9/58fps
WordChannels **[5]**	42.3%	0.5/8fps
LBP-channels **[37]**	39.1%	0.55/8fps
structured cascade regressor	**36.5%**	**0.55/8fps**
struct cascade+regressor	**37.7%**	**0.51/7.5fps**
SSD **[19]**	**35.8%**	**0.51/7.5fps**
RPN+PF **[45]**	34%	56fps
	10%	6fps

Table 1: Comparison with the state-of-the-art. Near real-time methods are separated from others. *Ours* is in bold. Deep learning techniques are in red. Computation times are calculated according to 640×480 resolution frames. The used metric is the log-average miss rate (the lower the better). Best viewed in color.

[10] P. Dollr, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. *CVPR*, 2009.

[11] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2009.

[12] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. *CVPR*, 2010.

[13] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *CVIU*, 114:712–722, 2010.

[14] A. Hoiem and M. Efros. Putting objects in perspective. *IJVC*, 2008.

[15] S. Hwang, T.-H. Oh, and I. S. Kweon. A two phase approach for pedestrian detection. *ACCV*, 2014.

[16] J. Jan. Robust multi-resolution pedestrian detection in traffic scenes. *CVPR*, 2013.

[17] L. Kaufman and P. Rousseeuw. Clustering by means of medoids, in statistical data analysis based on the l1 norm and related methods. *Y. Dodge*, pages 405–416, 1987.

[18] W. Ke, Y. Zhang, P. Wei, Q. Ye, and J. Jiao. Pedestrian detection via pca filters based convolutional channel features. *ICASSP*, 2015.

[19] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *ECCV*, 2016.

[20] J. Marin, D. Vazquez, A. M. Lopez, J. Amores, and B. Leibe. Random forests of local experts for pedestrian detection. *ICCV*, 2013.

[21] J. Marn, D. Vzquez, A. M. Lpez, J. Amores, and L. I. Kuncheva. Occlusion handling via random subspace classifiers for human detection. *IEEE Transactions on Cyberneticspart B: Cybernetics*, 44(3):342–354, 2014.

[22] L. Oliveira, U. Nunes, and P. Peixoto. On exploration of classifier ensemble synergism in pedestrian detection. *IEEE Transactions on Cyberneticspart B: Cybernetics*, 11(1):16–27, 2010.

[23] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. *ICCV*, 2013.

[24] W. Ouyang and X. Wang. Single-pedestrian detection aided by multi-pedestrian detection. *CVPR*, 2013.

[25] D. P, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 34(4):743–761, 2012.

[26] L. S. P. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, pages 129–137, 1982.

[27] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Strengthening the effectiveness of pedestrian detection with spatially pooled features. *ECCV*, 2014.

[28] D. Park, D. Ramanan, and C. Fowlkes. Multiresolution models for object detection. *ECCV*, 2010.

[29] R. Polikar. Ensemble learning. *Scholarpedia*, 4(1):2776, 2009.

[30] M. Ponti. Combining classifiers: from the creation of ensembles to the decision fusion. *SIBGRAPI-T*, 2011.

[31] J. Prokaj and G. Medioni. Persistent tracking for wide area aerial surveillance. *CVPR*, 2014.

[32] S. Ren, K. Hen, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

[33] M. Ristin, J. Gall, and L. V. Gool. Local context priors for object proposal generation. *ACCV*, 2012.

[34] W. Schwartz, L. Davis, and H. Pedrini. Local response context applied to pedestrian detection. *CIARP*, 2011.

[35] R. Trichet and F. Bremond. Dataset optimisation for real-time pedestrian detection. *rapport de recherche INRIA 9084*, 2017.

[36] R. Trichet and F. Bremond. How to train your dragon: best practices in pedestrian classifier training. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2017.

[37] R. Trichet and F. Bremond. Lbp channels for pedestrian detection. *WACV*, 2018.

[38] Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *CVPR*, 2001.

[39] C. Vondrick, A. Khosla, H. Pirsiavash, T. Malisiewicz, and A. Torralba. An efficient tree classifier ensemble-based approach for pedestrian detection. *IJCV*, 119(2):145–158, 2015.

[40] X. Wang, T. X. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. *ICCV*, 2009.

[41] O. Williams, A. Blake, and R. Cipolla. A sparse probabilistic learning algorithm for real-time tracking. *ICCV*, 2003.

[42] C. Wojek and B. Schiele. A performance evaluation of single and multi-feature people detection. *DAGM Symposium Pattern Recognition*, 2008.

[43] Y. Xu, X. Cao, and H. Qiao. An efficient tree classifier ensemble-based approach for pedestrian detection. *IEEE Transactions on Cyberneticspart B: Cybernetics*, 41(1):107–117, 2011.

[44] R. Yan, Y. Liu, R. Jin, and A. G. Hauptmann. On predicting rare classes with svm ensemble in scene classification. *ICASSP*, 2003.

[45] L. Zhang, L. Lin, X. Liang, and K. He. Is faster r-cnn doing well for pedestrian detection? pages 443–457, 2016.

[46] S. Zhang, C. Bauckhage, and A. B. Cremers. Informed haar-like features improve pedestrian detection. *CVPR*, 2014.

[47] S. Zhang, R. Benenson, and B. Schiele. Filtered channel features for pedestrian detection. *CVPR*, 2015.

[48] Q. Zhu, S. Avidan, M. Yeh, and K. Cheng. Fast human detection using a cascade of histograms of oriented gradients. *CVPR*, 2006.