Dataset Optimization for Real-Time Pedestrian Detection

Remi Trichet and Francois Bremond
INRIA Meditérannée
2004 route des Lucioles - BP 93 06902 Sophia Antipolis Cedex
remi.trichet@gmail.com, Francois.Bremond@inria.fr

***Abstract*—This paper tackles the problem of data selection for training set generation in the context of near real-time pedestrian detection through the introduction of a training methodology: FairTrain. After highlighting the impact of poorly chosen data on detector performance, we introduce a new data selection technique utilizing the expectation-maximisation algorithm for data weighting. FairTrain also features a version of the cascade-of-rejectors enhanced with data selection principles. Experiments on the INRIA and CALTECH datasets prove that, when finely trained, a simple HoG-based detector can outperform most of its near real-time competitors.**

***Index Terms*—Data Selection, Dataset Optimization, imbalanced datsets, Computer vision, Pedestrian Detection, Real-Time Application.**

I. Introduction

Human detection is the keystone of many computer vision applications that depend on the reliability of this important first step. These applications, ranging from surveillance, queue monitoring, retail data mining, and automatic pedestrian detection in the automotive industry, have fueled research during the last decade, leading to a soaring number of approaches on the topic.

In a recent study **[4]**, features, context, feature data, and the combination of approaches have been identified as the key performance elements for pedestrian detection. Roughly 30% of approaches focus on developing, combining or adapting features, and this direction has led to most of the breakthroughs over the past years, such as channel features **[13]**, or convolutional neural networks **[36]**. Context adaptation **[21]**, harnessing frequent geometry **[26][40]** or environment patterns **[38][53]**, also successfully received attention from the community, but this algorithm adaptation is often application or even dataset dependent. However, less progress has been stated concerning feature data. Training sets are still generated by applying the same techniques as a decade ago: Increasing the data variability is done by blindly mixing datasets **[4]**, and Bootstrapping **[11][17]** remains among the preferred choices for training set generation.

In this paper, we argue that data selection and adaptation is an important area that deserves more attention from the community. Training for pedestrian detection is, indeed, a peculiar task. It aims to differentiate a few positive samples with relatively low intra-class variation and a swarm of negative samples representing everything else present in the dataset. Consequently, the training set lacks discrimination and is highly imbalanced. Balancing these positive and negative instances is a rarely addressed issue in the literature due to the possible creation of noisy data while oversampling, and the likely loss of information when undersampling.

Consequently, this works advocates for a training procedure that bears two important data selection principles:

- Selecting more informative data.
- Balancing positive and negative instances all along the training process.

After motivating the need for proper data selection through experimentation, we will introduce *FairTrain*, a new training methodology that brings up a two-component contribution. First, it harnesses an expectation-maximisation scheme to weigh important training data for classification. Second, it improves the cascade-of-rejectors **[58][5]** classification by enforcing balanced train and validation sets every step of the way, and optimizing separately for recall and precision.

One key aspect of *FairTrain* is its genericity. It can easily be applied with other features or feature selection methods. Also, it differs from the state-of-the-art data selection methods by its ability to create balanced (training and validation) sets while avoiding over-generalisation. Experiments carried out on the INRIA **[11]** and Caltech-USA **[15]** datasets, demonstrate the effectiveness of the approach, leading a simple HoG-based detector to outperform its near real-time competitors.

The rest of this paper is organised as follows. Section 2 reviews the related state-of-the-art. Section 3 gives an overview of the system. Section 4 covers our training set generation technique whereas Section 5 details our classification scheme. Section 6 reports our experiments and Section 7 concludes on this work.

II. Related Work

Data selection from imbalanced datasets has been a concern for pedestrian detection since the birth of the field. **[28]** proved that disproportioned datasets degrade SVMs prediction accuracy, especially for non-linearly separable data. Subsequent research on these experiments **[54]** showed that best performance was obtained for approximately comparable class cardinalities when over-sampling the minority set. This principle was further applied todeep learning data **[22]**.

Limited work carries out these issues. Bootstrapping **[11][17]** is probably the most common adaptation to the problem. This data training mechanism improves accuracy

Fig. 1. *FairTrain* training pipeline. The initial training set generation selects data while balancing negative and positive sample cardinalities. A cascade classifiers is then trained on it, each independent classifier being learnt through bootstrapping. balanced positive and negative sets is sought all along the cascade. See text for full description.

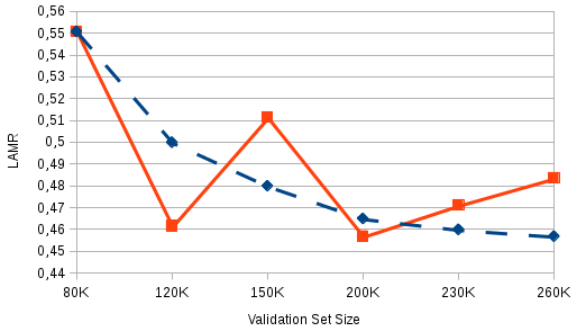

Fig. 2. Influence of the validation set size on a HoG classifier (Experimentation on the INRIA dataset). Expected performance is in dashed blue, actual results in plain red. The used metric is the log-average miss rate (the lower the better).

thanks to successive training set updates focusing on samples difficult to classify. It is first initialised on uniformly drawn negative samples. At each iteration, the negative training set is augmented with the false positives of the previous iteration. The process keeps repeating until the performance converges or a memory threshold is reached. **[47]** further showed that 2 iterations lead to optimal performance. **[20]** adopted a different practice for random forest. They first train t trees with uniformly-drawn samples, use them to obtain harder positive and negative sample to train the next t trees, and iteratively repeat this process. Each tree group bears its own bias, therefore improving the overall forest performance.

Generative techniques consider over-sampling the minority class through artificial data generation. This type of approach suffers from both, over-generalisation and the risk to create erroneous data. The SMOTE algorithm **[48]** is probably the most renowned one. It creates new data as the linear combination of a randomly selected data point and one of its (same class) k nearest neighbors. Borderline-SMOTE **[23]** subsequently proposed to improve SMOTE through clever selection and strengthening of weak pairs. ADASYN **[24]** considered limiting the over-generalisation through density-based data generation. In the same vein, **[29]** introduced cluster-based oversampling, simultaneously tackling the intra- and inter-class imbalance issue.

Cost sensitive learning methods **[16][44][31]** are an alternative to generative techniques that learn the cost of misclassifying each data sample. They optimise the classification performance by up-weighting important data, either directly or in the loss function, and provide a natural way to enhance the minority class.

Finally, **[27]** highlighted and tackled the imbalance issue with a two classifier cascade: The first one aims for a high recall whereas the second one enhances precision.

An alternate strategy to deal with the excess of negative data is to successively prune out easy-to-classify instances via the use of several consecutive classifiers. This cascade-of-rejectors **[58][5]** can also be considered to speed up the detection. The technique, inspired by Viola & Jones face detector **[46]**, builds up a classifier cascade that consists of successive rejection stages that get progressively more complex, therefore rejecting more difficult candidates as the classifiers get more specialised. **[43]** further upgraded the principle from features to data by performing hierarchical clustering on the dataset, leading to improved classification results. However, it requires, for each new datum, to find the corresponding set of clusters it belongs to. See **[25]** for more details on learning from imbalanced datasets.

III. *FairTrain* OVERVIEW

The training procedure that this paper advocates for unfolds as follows. After the initial data selection, we balance the negative and positive sample cardinalities. Then, a set of n negative data rejectors are trained and identified negative data are discarded. The validation set of negative samples are iteratively oversampled after each training to ensure a balanced set. The final classifier is learned after careful data selection. Figure 1 illustrates the process. The next two sections respectively detail our data selection mechanism and our classifier training methodology.

IV. TRAINING SET GENERATION

We conduct a simple experiment to illustrate the proposed main idea for the generation of the training set. Figure 2 plots the performance on the INRIA dataset of a simple HoG classifier in relation to the validation set size. The training set (of size 10K) is determined using bootstrapping **[11][17]**.

Performance should theoretically asymptotically increase with the size of the validation set. This assumption, depicted by the dashed curve on the Figure, corresponds to the common belief that the dataset informative power is directly correlated to its size. However, the results, plotted with the plain curve, are quite different from our expectations. They show significant irregularities as the dataset size increases; and the largest datasets do not even provide the best results. This encourages us to re-think about the way training data is considered: Apt data selection may be more important than their simple addition.

This is the principle underpinning our training set generation. The method is composed of several stages, detailed within

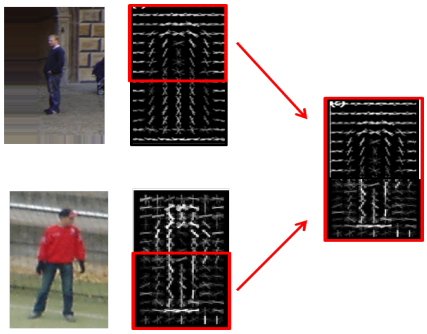

Fig. 3. data generation example.

the body of this section. We first select insightful positive and negative instances from the millions of extracted features. The minority class is then oversampled to create balanced training and validation sets. The third step generates an initial training set utilizing boostrapping **[11][17]**. Finally, we optimise this first set using an expectation-maximisation algorithm.

Throughout this paper, we will consider that a dataset is divided in training and testing sets and that the training set could further be divided in a training and validation set. The sizes of the training and validation partitions are algorithm dependent.

A. Instances selection

Ground truth positive examples are fed to the training set and near positives (i.e. samples with a ground truth coverage higher than 90%) are added to the validation set. For each image, data negative samples are sorted according to their similarity with positive examples and we then randomly select m instances among the most similar half (based on euclidean distance). A subset is added to the training set to match the positive sample cardinality and the rest is left to the validation set. This simple, yet efficient selection process focusses on hard data near the categories border. At this stage, we have constructed a tiny balanced training set and a large imbalanced validation set. The next steps oversample the validation set minority class and grows the training set.

B. Minority class oversampling

We augment the validation set in order for the two opposite classes to have the same cardinality. As the minority class is always the positive one, this comes down to augmenting the set of positive samples. For that purpose, we use a generative sampling algorithm that takes advantage of the possibility to geometrically locate the feature values within the bounding box. The method is then restricted to feature types that are geometrically structured, and have a precise location attached to their values, such as HoG **[11]**, DPM **[18]**, or Haar-inspired strategies **[13]**.

The new data generation function $f(.)$ unfolds as follows. We randomly select 2 instances $x_1 = [x_{11}, ...x_{1n}]$ and $x_2 = [x_{21}, ...x_{2n}]$ from the same class S_i and create a new sample with their lower and upper body features, while avoiding doublons. Uniform sampling with replacement is employed. More formally:

$$\forall x_1, x_2 \in S_i \quad f(x_1, x_2) = [x_{11}, ...x_{1\frac{n}{2}}, x_{2\frac{n}{2}+1}, ...x_{2n}] \quad (1)$$

So, a minority class of n samples can spawn a maximum of $n \times (n-1)$ new instances. This process doesn't suffer from the over-generalisation issue that mares other generative techniques, like SMOTE **[48]** as data pairs are not necessarily selected within the same neighborhood and the upper/lower body dichotomy rule guarantees a valid datum.

Of course, other geometrical dichotomies can be thought of as long as the final instance appearance, corresponding to the selected data feature combination, is credible. Moreover, vertical splits might be pointless, as lots of training data are often vertically mirrored during pre-processing. We do not normalise the final vector. In practice, this gives us slightly better results than a re-normalisation. The process is illustrated in the Figure 3.

C. Initial training set generation

We use bootstrapping **[11][17]** to augment the training set. At each iteration, we incorporate the most badly classified data from the validation set into the training set and retrain the classifier. We aim to repeat this process until the training set reaches its desired initial size T.

However, 2 iterations have been advised to avoid overfitting **[47]**. Therefore, we modified the algorithm to deal with this issue. Instead of simply selecting data from the validation set, we generate it the same way as detailed in the minority class oversampling section. Each new datum parent samples are selected among the misclassified data. Weighted sampling is utilised in this case, the probability of an instance to be drawn being set in relation to the extent of the error. We iterate until the desired training set size is reached or the validation set is perfectly classified.

D. Training set Optimisation

The proper selection of an apt training set is paramount for optimal classifier performance. Our purpose is to select the core training set data that will lead to improved classification. Our core idea is a typical machine learning one: Since we can hardly define what makes a good training set, we will let the algorithm learn it by itself. We utilise an expectation-maximisation algorithm for this purpose.

The algorithm unfolds as follows. Weights, initialised equal, are associated to the training data. At each iteration it, (M-step) we randomly select $D_{it}/2$ positive and $D_{it}/2$ negative data from the training set to respect the class equality rule **[54]**, the selection being guided by data weights. A new classifier C_{it} is then trained on these D_{it} data, and tested on the validation set. (E-step) Weights are updated according to the classification performance $score(C_{it})$ of the classifier C_{it}. A sample's weight is set according to the average classification performance over all the past runs for which it was selected.

$$W(x_i) = \left(\sum_{S_{x_i}=[it|x_i \in D_{it}]} score(C_{it}) \right) / |S_{x_i}| \quad (2)$$

To further improve this algorithm, we set up a global confidence value for each validation datum. This value is the average confidence value for each datum over all the past

Data: - The training set t of cardinality $|t|$.
- $|t|$ weights associated to training data.
- The validation set v of cardinality $|v|$.
- $|v|$ confidence values associated to validation data.
- $|v|$ global confidence values associated to validation data.
- The percentage D of data selected for bagging.
- The percentage P of badly classified data kept.
- The maximum number m of iterations.
Result: Optimal training set o with $|o| \leq |t|$
initialization;
Set weights to 1;
Set globalConfs to 1;
Set bestscore to minimal value;
for *it=0; it<m; it++* **do**
  // Randomly select $D/2$ positive and $D/2$ negative data from the training set.
  Bagging(weights, t, D);
  // Train a classifier C_{it} with the D data.
  C_{it} = Train(d);
  // Test the classifier on the validation set.
  $\{Score, confs\}$ = classify(C_{it}, v);
  // Update weights and global confidence values.
  Update(weights, score);
  Update(globalConfs, confs);
  // Saving best training set.
  if *Score > bestscore* **then**
    bestscore = Score;
    SaveClassifier(C_{it});
  end
  if *it%(m/10) == 0* **then**
    // Randomly generate P data among the most consistently badly classified.
    p = Generate(globalConfs, v, P);
    // and add them to the training set
    t += p;
  end
end

Algorithm 1: Training set generation algorithm.

iterations. Every $m/10$ iterations, this global confidence values are L1-normalised and utilised as probabilities to randomly generate new data based on the most badly classified data. In other words, we create additional data in areas of the feature space where extra information is required, like near the border between categories. This active learning mechanism improves the quality of the training set the same way bootstrapping **[11][17]** does. Consequently, the algorithm gives a better outcome when associated with a large validation set that will limit overfitting.

D is set low at the beginning (i.e 5% of the training set) and progressively increased as the iterations unfold until the desired training set size is reached. This allows the algorithm to focus on the core samples withholding the classifier first, before refining the set for optimal classification. Also, running the bulk of the iterations with small training sets speeds up the algorithm and allows us to run the numerous iterations necessary for its reliability. The algorithm pseudo-code is depicted in algorithm 1.

Obviously, a large number of iteration is necessary for the algorithm to produce good results. For our experiments, we set $m = 5000$ when the algorithm is a standalone, and $m = 1000$ when it is used as the cascade-of-rejectors final classifier (see section V). The training takes 16 hours to complete with 200K data using 12 cores; 6 hours for 10K data.

V. Cascade-of-Rejectors

The cascade-of-rejectors is an efficient strategy for sieving out a large proportion of false positives, and has been applied several times in the context of pedestrian detection **[10][58] [5][43][46]**. However, 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. However, this leads to progressively more imbalanced datasets. Fine tuning of the original positive versus negative instances ratio is usually necessary to obtain satisfactory results (a 2 to 4 times bigger negative set is the common practice). But this strategy is still under-optimal as balanced datasets are proved **[54]** to be the best performing ones.

To deal with these issues, we propose a new cascade-of-rejectors. Our approach borrows **[27]**'s main idea with a two-stage classification: The first one aims for a high recall whereas the second one focuses on precision.

The first stage embodies a cascade of $n-1$ rejectors. To avoid increasing the false negative rate, these $n-1$ initial classifiers enforce a high recall, moderately optimizing the performance on the precision. The metric formalises as:

$$M = (TN \times W_{TN} + TP)/(P + N) \tag{3}$$

with P, N, TP, TN, and W_{TN} being respectively positive, negative, true positive, true negative instances, and the weight associated to TN. W_{TN} is set at $W_{TN} = 0.5$. Its parametrisation has no influence on the results, when set low enough, which enforces a perfect or near perfect recall. Negative data, with a confidence below 0.5, are rejected at each stage. Negative data over-sampling is then undertaken to avoid emptying the validation set. When insufficient to match the positive instances cardinality, random positive data under-sampling is then considered. We employ Adaboost as classifiers for this part. These decision trees have demonstrated competitive performance whilst retaining computational efficiency **[51][34]**. During testing, typically over 95% of the data are safely discarded after 4-5 iterations. This cascade-of-rejectors guaranties a high recall, increased performance induced by balancing the datasets at each iteration, and few parameters to tune.

The second stage performs a finer classification, aiming for high precision results. Since the bulk of the data have been removed, more demanding computation can be performed at this stage without slowing down the detector. We employ a dense forest classifier, that has shown **[4]** giving slightly better performance than its counterparts on pedestrian detection. We

optimise for the log-average miss rate **[39]**.

The only parameter that requires careful tuning is the rejection threshold T value during the testing phase. Since the method optimises for the recall, the large majority of data around the separating border between the two classes are negative ones. It is then sensible to set $T > 0.5$. The two employed classifiers in this work are typical Adaboost **[52]** and random forest **[6]**. See the experiment section for a thorough evaluation of this parameter impact on performance.

VI. Experiments

This section, dedicated to our experimental validation, breaks down into four sub-parts: dataset presentation, experimental setup details, results and related comments, and finally parameter discussion.

A. Datasets

We experimented on the INRIA and Caltech-USA datasets. The INRIA **[11]** features high resolution pictures mostly gleaned from holidays photos. The training set consists of 2416 cropped positive instances from 614 images and 1218 images free of persons. The test set contains 1132 positive instances from 288 images and 453 person free images for testing purposes. This is among the most widely used dataset for person detector validation and comparative performance analysis.

The Caltech-USA **[15]** dataset consists of 2.3 hours of video recorded at 30fps from a vehicle driving various Los Angeles streets. It totals 350000 images and 1900 unique individual pedestrians, 300 large groups, and 110 hard to distinguish pedestrians. Despite some annotation errors **[57]**, its large size along with crowded environments, tiny pedestrians (as low as 20 pixels), and numerous occlusions probably make the Caltech-USA dataset the most widely used one. Annotations allow to experiment on 2 different sets. Contrarily to the "full set", the "reasonable set" restricts algorithms to pedestrians over 50 pixels in size and a maximum of 35% occlusion.

With respectively a large variability and tiny occluded detections, everyday pictures and automotive application, these two datasets offer complementary settings for our experiments.

B. Experimental setup

We experimented with HoG **[11]** and Haar-LBP **[9]** features. HoGs are configurated with 12×6 cells, 2×2 blocks and 12 angle orientations, for a total of 2640 values. Block as well as full histogram normalisation are performed. The LBP descriptor follows the same structure, features a value per channel for each cell, and non-uniform patterns are pruned out **[35]**. Candidates are selected using a multi-scale sliding window approach **[58]** 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.

We also clean the data based on Tomek links **[45]**. We employ euclidian distance and restrict the deletion to negative data to spare the scarce minority class. Approximately 0.1% to 0.2% of the negative data are removed this way. No generative resampling is performed. The model height is set to 100 pixels for the INRIA dataset, 50 pixels for Caltech.

We also implemented a cascade-of-rejectors baseline with the commonly used settings: Rejector optimisation is done according to the log-average miss rate metric. We tested this baseline with a typical initial validation set containing 4 times more negative samples than positive ones. To compare with the common multi-dataset training technique, we also used an augmented version with PETS2009 **[19]** positive samples. This set gathers 50K positive instances from both datasets and 110K negatives from the sole INRIA benchmark.

We utilised the openCV implementation for the Adaboost and random forest classifiers. Random forest parameter set includes the number of decision trees, the number of sampled feature dimensions and the max tree depth. They were selected by measuring out-of-bag errors (OOB) **[6]**. It was computed as the average of prediction errors for each decision tree, using the non-selected training data. Adaboost is initialised with a tree depth of 2 and 256 weak classifiers. Each extra run adds 64 weak classifiers. This is a low number of weak classifiers compared to typical settings (i.e. [1024, 2048]). However, in practice, increasing this value leads us to lower performance and speed.

We use a variant of the threshold non-maximum suppression (t-NMS) **[7]** that groups the detections according to the bounding boxes overlap with the group *d* best candidate, keeps the *d* candidates with highest confidence for each group, and builds the final candidate position with the group mean border positions. We set the overlap threshold to 0.6 and $d = 3$ for all our experiments. Log-Average Miss Rate (LAMR) **[39]** is employed as metric for all runs.

C. Results

Detailed method results on the INRIA dataset, including the influence of its various components and settings are reported in Table I. The expectation-maximisation (EM) training leads to an average 6% improvement over simple HoG-based classification, or 2.3% over the common cascade-of-rejectors. This demonstrates the importance of proper data selection. The main boost is obtained by our modified version of the cascade-of-rejectors, with a minimum 24% performance increment. Finally, the combined algorithms lead to a further 3% improvement. To sum up, *FairTrain* gives a simple HoG based detector an impressive 27% boost, therefore competing with state-of-the-art real-time references. It significantly outperforms the usual blend of datasets strategy (see Table I - run 3), scoring only 42.44%.

The main factor responsible for the cascade-of-rejectors significant results is the balanced generative dataset. Max-recall optimisation also has a slight, but steady influence on performance, with an average 0.51% better score. Similarly, we found random forest to outperform Adaboost as final classifier by approximately 0.4% on all EM runs. Finally, in accordance with the literature **[10][58][5]**, we observed that

Run	Validation Set Size	#Rejectors	Final Classifier	Optimization	LAMR Performance
1	**180K (INRIA)**	**0**	**A**	**n.a.**	**46.03%**
2	**180K (INRIA)**	**5**	**A**	**n.a.**	**43.38%**
3	**160K (INRIA+PETS)**	**5**	**A**	n.a.	**42.44%**
4	180K (INRIA)	0	EM training(A)	n.a.	39.72%
5	180K (INRIA)	0	EM training(RF)	n.a.	40.34%
6	180K (INRIA)	5	A	LAMR/Recall	21.64% / 21.09%
7	180K (INRIA)	5	RF	LAMR/Recall	21.34% / 20.98%
8	180K (INRIA)	5	EM training(RF)	LAMR/Recall	20.39% / 20.15%
9	220K (INRIA)	5	EM training(RF)	LAMR/Recall	20.25% / 19.68%
10	250K (INRIA)	5	EM training(RF)	LAMR/Recall	20.54% / 19.67%
11	300K (INRIA)	5	EM training(RF)	LAMR/Recall	19.52% / 19.01%
12	220K (INRIA)	6	EM training(RF)	LAMR/Recall	19.82% / 19.38%

TABLE I

LOG-AVERAGE MISS RATE RESULTS (THE LOWER THE BETTER) OF *FairTrain* WITH VARIOUS COMPONENTS AND SETTINGS ON THE INRIA DATASET WITH HOG FEATURES. BASELINES ARE IN BOLD. **A** - ADABOOST. **RF** - RANDOM FOREST. **EM** - EXPECTATION-MAXIMISATION. **LAMR** - LOG-AVERAGE MISS RATE. BASELINES ARE IN BOLD.

Method	INRIA	Speed(CPU/GPU)
HoG **[11]**	46%	0.5fps
HoG-LBP **[49]**	39%	Not provided
MultiFeatures **[50]**	36%	$<$ 1fps
FeatSynth **[1]**	31%	$<$ 1fps
MultiFeatures+CSS **[47]**	25%	No
Channel Features **[13]**	21%	0.5fps
FPDW **[12]**	21%	2-5fps
DPM **[18]**	20%	$<$ 1fps
RF local experts **[33]**	15.4%	3fps
PCA-CNN **[30]**	14.24%	$<$ 0.1fps
FairTrain - HoG	**19.01%**	**4/60fps**
VeryFast **[2]**	18%	8/135fps
WordChannels **[10]**	17%	0.5/8fps
crossTalk cascades **[14]**	17%	30-60fps
FairTrain - LBP	**17.28%**	**3.7/54fps**
FairTrain - HoG+LBP	**17.08%**	**2.6/39fps**
FRCNN **[42]**	13%	7fps
RPN+PF **[55]**	7%	6fps

Method	CALETCH	Speed(CPU/GPU)
HoG **[11]**	69%	0.5fps
DPM **[18]**	63.26%	$<$ 1fps
FeatSynth **[1]**	60.16%	$<$ 1fps
MultiFeatures+CSS **[47]**	60.89%	No
HoG Caltechx10	59.68%	0.5fps
FPDW **[12]**	57.4%	2-5fps
Channel Features **[13]**	56.34%	0.5fps
Roerei **[3]**	48.35%	1 fps
MOCO **[8]**	45.5%	$<$ 1fps
JointDeep **[37]**	39.32%	$<$ 1fps
InformedHaar **[56]**	34.6%	$<$ 0.63fps
katamari-v1 **[4]**	22.49%	$<$ 1fps
FRCNN **[42]**	56%	7fps
CrossTalk cascades **[14]**	53.88%	30-60fps
FairTrain - LBP	**51.12%**	**3.6/54fps**
FairTrain - HoG	**45.4%**	**3.9/58fps**
FairTrain - HoG+LBP	**45.2%**	**1.8/27fps**
WordChannels **[10]**	42.3%	0.5/8fps
SSD **[32]**	34%	56fps
RPN+PF **[55]**	10%	6fps

TABLE II

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. $G_x + G_y + L$ CHANNELS ARE EMPLOYED FOR THIS EXPERIMENT. THE USED METRIC IS THE LOG-AVERAGE MISS RATE (THE LOWER THE BETTER). BEST VIEWED IN COLOR.

optimal results are obtained for 5 iterations of the rejection cascade, the addition of extra iterations being impactless, as shown in Table I - run 12.

We have tested this algorithm for various validation set sizes (see Table I - runs 8 to 11) to replicate the section IV theoretical experiment (see Figure 2). Results show much more stable performance, therefore validating our primary assumption as well as the approach stemming from it.

Table II shows our method's ranking compared to the state-of-the-art, in terms of performance and speed. When comparing methods, note that some methods **[4], [47], [41]** get increased results thanks to optical flow features. T=0.58 is used for these runs. This work compares favorably to the state-of-the-art. While not being the best detector on the market, with respectively 17% and 45% log-average miss rate on the INRIA and Caltech datasets, *FairTrain* ranks on par or better compared to state-of-the-art pedestrian descriptors in terms of performance/speed ratio, except for **[55]**. For instance, it scores better than the famous DPM **[18]** and similarly to the integral channel features **[12]** while running 4 times faster. VeryFast **[2]** and the crosstalk cascades provide a viable alternative in terms of speed while displaying lower performance. Finally, deep learning techniques **[32], [42]** often perform better but are slower.

LBP is the best descriptor on the INRIA dataset, while HoGs show the best performance on Caltech. We assume that the numerous little pedestrians on the latter impact the texture descriptor more strongly than the gradients. As extra validation experiment, we also compared our method with the caltechx10 training set extracting 10 times more positives (i.e. every 3 frames) on the eponymous dataset **[12]**. This run, reported under the name of *HoG Caltechx10*, shows much lower performance than *FairTrain*. The HoG and LBP fusion yields little improvement.

Packed crowds, teensy pedestrians and awkward poses remain the main failure cases. For a near optimal parametrisation of T, pruning out approximately 70% of the data during the first iteration, the succession of classifiers overall multiplies by 1.4 the computation load. The end-to-end system processes 3 to 4 fps on an Intel Xeon 2.1GHz CPU, calculated over frames of 640×480 pixels in size. Further speed up is possible when using a GPU, leading our detector to perform in real-time. This makes *FairTrain* one of the best performing real-time pedestrian detector to date.

Figure 5 shows some examples taken among the most challenging ones. Packed crowds, teensy pedestrians and awkward poses remain the main failure cases.

D. Parameter Sensitivity

The rejection threshold T from the cascade-of-rejectors has a critical influence on both, results and computation time.

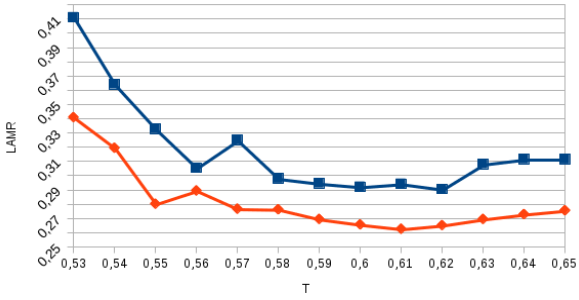

Fig. 4. Influence of the rejection threshold T on performance. The blue and red curves respectively depict versions with Adaboost and an EM trained random forest as final classifiers. Log-average miss rate is used as metric (the lower the better) and rejector optimization function.

Indeed, the more data are removed at each classification step, the faster the detector will be. Also, since there is a high imbalance of positive/negative instances during testing, it is sensible to yield better results for $T > 0.5$. Moreover, optimizing rejection cascade classifiers for a high recall further accentuates this bias. Figure 4 plots its influence on the results when optimizing for log-average miss rate with two different final classifiers: Adaboost and EM-optimised random forest. We observe that the results are more stable across the parameter range when the expectation-maximisation optimisation of the training set is employed. This further validates the use of this algorithm. The optimal testing confidence threshold is within the range $[0.58, 0.6]$ for all tests whether we are optimizing for log-average miss rate or to the recall. For all runs, a low standard deviation is observed within this optimal range (typically, .5%). This demonstrates the stability of this parameter. We explain the similar parameter setting (and performance) of the two optimisation techniques with their overall similarity. Indeed, log-average miss rate also intrinsically optimises for recall. Even though it provides lower performance in our case, an automatic threshold estimation of T for each separate classifier may theoretically lead to further improvement.

VII. Conclusion

This paper explored data selection mechanisms for near real-time pedestrian detection. Via *FairTrain*, we introduced an expectation-maximization data weighting scheme for enhanced training set generation. We also improved the cascade-of-rejectors by enforcing balanced datasets at every step of the classification and separately optimizing for recall and precision. The method showed competitive results compared to the state-of-the art and major performance boost when compared to original features.

By proving that blind data addition is not the best way to enhance a training set, this work brings up an important questions: What makes a good training set? Or even a good training datum? Future experimentation aims to provide responses to these core interrogations. Also, the addition of more recent features and further work on classifier combination is envisioned.

VIII. Acknowledgment

To appear in camera ready version.

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. Van Gool. Pedestrian detection at 100 frames per second. *CVPR*, 2013.
[3] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool. Seeking the strongest rigid detector. *CVPR*, 2013.
[4] R. Benenson, M. Omran, J. Hosang, and B. Schiele. Ten years of pedestrian detection, what have we learned? *ECCV, CVRSUAD workshop*, 2014.
[5] B. Berkin, B. K.P. Horn, and I. Masaki. Fast human detection with cascaded ensembles on the gpu. *IEEE Intelligent Vehicles Symposium*, 2010.
[6] L. Breiman. Random forests. *Machine Learning*, 2001.
[7] M. Diez Buil. Non-maximum suppression. *technical report ICG-TR-xxx*, 2011.
[8] G. Chen, Y. Ding, J. Xiao, and T. X. Han. Detection evolution with multi-order contextual co-occurrence. *CVPR*, 2013.
[9] E. Corvee and F. Bremond. Haar like and lbp based features for face, head and people detection in video sequences. *ICVS*, 2011.
[10] A. D. Costea and S. Nedevschi. Word channel based multiscale pedestrian detection without image resizing and using only one classifier. *CVPR*, 2014.
[11] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.
[12] P. Dollar, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. *BMVC*, 2010.
[13] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral channel features. *BMVC*, 2009.
[14] P. Dollr, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. *ECCV*, 2012.
[15] P. Dollr, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. *CVPR*, 2009.
[16] C. Elkan. The foundations of cost-sensitive learning. *Intl Joint Conf. Artificial Intelligence*, 2001.
[17] P. F. Felzenszwalb, R. B. Girshick, and D. McAllester. Cascade object detection with deformable part models. *CVPR*, 2010.
[18] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 32(9):1627–1645, 2009.
[19] J. Ferryman and A. Shahrokni. An overview of the pets2009 challenge. *PETS*, 2009.
[20] J. Gall and V. Lempitsky. Class-specific hough forests for object detection. *CVPR*, 2009.
[21] C. Galleguillos and S. Belongie. Context based object categorization: A critical survey. *CVIU*, 114:712–722, 2010.
[22] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.
[23] H. Han, W.Y. Wang, and B.H. Mao. Borderline-smote: A new over-sampling method in imbalanced data sets learning. *Intl Conf. Intelligent Computing*, 2005.
[24] H. He, Y. Bai, E.A. Garcia, and S. Li. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. *Intl J. Conf. Neural Networks*, 2008.
[25] H. He and E. A. Garcia. Learning from imbalanced data. *IEEE transactions on Knowledge and data engineering*, 21(9), 2009.
[26] A. Hoiem and M.H. Efros. Putting objects in perspective. *IJVC*, 2008.
[27] S. Hwang, T.-H. Oh, and I. S. Kweon. A two phase approach for pedestrian detection. *ACCV*, 2014.
[28] N. Japkowicz. Learning from imbalanced data sets: a comparison of various strategies. *AAAI Workshop on Learning from Imbalanced Data Sets*, 2000.
[29] T. Jo and N. Japkowicz. Class imbalances versus small disjuncts. *ACM SIGKDD Explorations Newsletter*, 6(1):40–49, 2004.
[30] W. Ke, Y. Zhang, P. Wei, Q. Ye, and J. Jiao. Pedestrian detection via pca filters based convolutional channel features. *ICASSP*, 2015.
[31] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollr. Focal loss for dense object detection. *ICCV*, 2017.
[32] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *ECCV*, 2016.
[33] J. Marin, David Vazquez, A. M. Lopez, J. Amores, and B. Leibe. Random forests of local experts for pedestrian detection. *ICCV*, 2013.
[34] W. Nam, P. Dollar, and J. H. Han. Local decorrelation for improved pedestrian detection. *NIPS*, 2014.

Fig. 5. Detection examples on the INRIA dataset. Green boxes represent our algorithm detections, the brightness being set in relation to their confidence. Red boxes depict missed detections. Best viewed in color.

[35] T. Ojala, M.Pietikinen, and T. Menp. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *PAMI*, 2002.

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

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

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

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

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

[41] D. Park, C.L. Zitnick, D. Ramanan, and P. Dollr. Exploring weak stabilization for motion feature extraction. *CVPR*, 2013.

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

[43] M. Souded and F. Bremond. Optimized cascade of classifiers for people detection using covariance features. *VISAPP*, 2013.

[44] K.M. Ting. An instance-weighting method to induce cost-sensitive trees,. *IEEE Trans. Knowledge and Data Eng.*, 14(3):659–665, 2002.

[45] I. Tomek. Two modifications of cnn. *IEEE Trans. System, Man, Cybernetics*, 6(11):769–772, 1976.

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

[47] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. *CVPR*, 2010.

[48] B.X. Wang and N. Japkowicz. Imbalanced data set learning with synthetic samples. *IRIS Machine Learning Workshop*, 2004.

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

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

[51] B. Wu and R. Nevatia. Cluster boosted tree classifier for multi-view, multi-pose object detection. *ICCV*, 2007.

[52] R. Shapire Y. Freund. A decision-theoric generalization of online learning and an application to boosting. *Journal of Computer and System Sciences*, 1997.

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

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

[55] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. Is faster r-cnn doing well for pedestrian detection? pages 443–457, 2016.

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

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

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