Pedestrian Detection with Multichannel Convolutional Neural Networks

David José Lopes de Brito Duarte Ribeiro
Instituto Superior Técnico

***Abstract*—This work proposes an innovative method to approach the PD problem, based on the combination of heterogeneous input channel features obtained with Convolutional Neural Networks (CNNs). More specifically, a pre-trained CNN model is fine-tuned for each single input channel (e.g. RGB or LUV) and high level features are extracted from the penultimate layer (right before the classification layer). Finally, a multichannel input CNN model is trained (partially or fully) with these high level features. During test, the full images are pre-processed with the Aggregated Channels Features (ACF) detector in order to generate pedestrian candidate windows (i.e., windows potentially containing pedestrians). Next, these candidate windows are entered in the multichannel input CNN model, being effectively classified as pedestrians or non pedestrians.**

The developed method is competitive with other state of the art approaches, when evaluated on the INRIA dataset, achieving improvements over the baseline ACF method. Two experimental setups were adopted, namely, the full INRIA dataset with higher resolution and the partial INRIA dataset with lower resolution. Furthermore, the devised methodology can be successfully applied to the low resolution PD problem, and promisingly extended to other areas by integrating information from several inputs.

***Index Terms*—Pedestrian Detection, Convolutional Neural Networks, single input channel, CNN multichannel input model, high level features**

I. Introduction

THE ability to detect people is an important and challenging component of human-machine interaction. The substantial advances observed in Pedestrian Detection (PD) in the last few years, were motivated by its wide range of applications, e.g. automotive safety, surveillance, robotics and Advanced Driver Assistance Systems, to quote a few.

The PD main challenges are due to the high variability in the pedestrian appearance and the similarities with other classes of objects resembling pedestrians. The appearance of the pedestrians is influenced by the pose, clothing, lighting conditions, backgrounds, scales and even by the atmospheric conditions. Besides that, occlusions, background clutter, deformations, viewpoint variations, low resolution and the resemblances between pedestrians and other classes of objects are other complexities inherent to this task (illustrated in Figure 1).

The Deep Learning (DL) framework, including Convolutional Neural Networks (CNN), have successfully been applied to several vision tasks such as general object classification [1], general object detection [2] and even PD [3], [4], [5]. As a result, this framework can be utilized to further explore the PD problem.

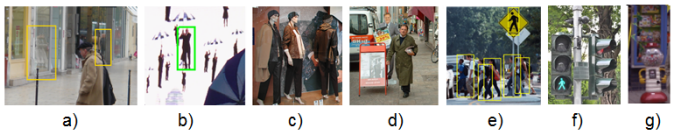

Fig. 1. Some examples of PD challenges: a) reflections of pedestrians in windows; b) drawings of people; c) mannequins; d) a pedestrian sized poster; e) a pedestrian traffic sign and crowd related occlusions; f) pedestrian traffic lights and g) a gumball machine resembling a pedestrian. a), b), c) and d) were adapted from [6], e) was obtained from [7], f) was obtained from [8] and g) was obtained from the INRIA dataset [9].

I-A *Related Work*

The PD problem has been studied for more than a decade, with several datasets, benchmarks, evaluation metrics and over 40 methods being developed and established in the process.

However, these methodologies can be categorized into three main families according to [10]: Decision Forests (DF) (possibly using boosting techniques and the most common approach), Deformable Part Models (DPM) variants and Deep Networks (DN). Additionally, some works consider the combination of diverse families, such as [11], which integrates DPM and DN to tackle the PD occlusion problem.

In general terms, most detectors contemplate three main components: feature representation, classification schemes and training methodology. The features can be obtained from different input channels, transformations or pre-processing (e.g. HOG, RGB and LUV). Regarding the classifiers, it is possible to differentiate between two main frameworks: monolithic and part-based. While in the monolithic case the entire image is taken into account to detect the pedestrians (DF family, e.g. [9]), in the part-based case, deformation models and partial detections contribute to determine the pedestrian existence probability (DPM family, e.g. [12]). Training can be performed with a linear (more frequently) or non linear Support Vector Machine (SVM) (e.g. [9]), boosting, among more complex methodologies.

Accordingly, the work of [13] intends to unify the framework of several PD methods (such as [14], [15], [16], [17] and [18]), by proposing a detector's model where a boosted decision forest uses filtered low-level features.

In the last years, the improvement experienced in the PD methodologies performance relied mainly on the use of better features [10], as noticed when comparing the prominent benchmarks of Viola and Jones [19], HOG [9], Integral Channels Features [14], Squares Channels Features [16] and Spatial Pooling [20].

The sliding window based PD methodologies are the most successful and commonly found in the literature, in contrast to keypoints [21] or segmentation [22] based approaches. Among the pioneers of PD sliding window methods is [23], which resorted to the use of the Haar wavelet transform in conjunction with SVMs to build an overcomplete dictionary of features at multiple scales. This work was extended by [19], which developed a faster technique (named Integral Images) to compute the features, applied AdaBoost to sparsely extract the most important visual attributes, and proposed a cascade of classifiers to constrain the computational effort to the most promising regions. Recently, these core principles are still applied in most of the detectors (mainly in the DF and DPM families, but also in combination with DN approaches), to some degree or variation.

In [14], the Integral Images technique proposed in [19] was applied to different image channels (namely, gradient histograms, grayscale, RGB, HSV, LUV, and gradient magnitude), in order to extract features, which were subsequently processed by a boosted classifier. This method was improved in [15], by changing the features from local sums in image channels to pixel lookups in aggregated channels (resulting in the Aggregated Channels Features (ACF) method). Building on the ACF framework [15], in [18] a methodology to locally decorrelate the channel features was created.

The complementarity of different types of features has been proven to yield improved results for the combinations: motion and color self-similarity (CSS) [24], and optical flow information, context information (i.e., interactions among two people) and channel augmentation using a discrete cosine transform [10].

The emergence of real time applications (e.g. surveillance and robotics) demanded higher detection rates (measured in frames per second(fps)), besides accuracy. As a result, new methodologies were proposed (e.g. [25], [15]) to approximate the feature computation at multiple scales, resulting in substantial runtime improvements with slight accuracy reductions. For example, it is possible to achieve detection rates of approximately 5 fps on 640x480 images [25], superior to 30 fps [15], from 35 to 65 fps (on 640x480 images) [26], 50 fps for monocular images (using GPU) and 135 fps for stereo images (using CPU and GPU) [27].

In the scope of deformable part models, an unsupervised method was devised in [28] to learn scale invariant part constellations and, more recently, [12], [29] applied root filters, part filters and the respective deformation models (named star models) to images at multiple scales and positions. The resulting star models were combined and a score was assigned to each position and scale of the images.

The detection of pedestrians far away (i.e., at low and medium resolutions) from the camera guarantees a safety margin in automotive applications and allows to obtain adequate results in surveillance, without requiring expensive equipment. However, the detection with low resolution images is disappointing [30], making it an important and challenging problem in PD. Consequently, to address this problem, the work of [31] and [32] introduced improvements to the DPM approach proposed in [12].

Further considerations in parts, shape, pose and articulation representations are present in the literature. For example, edgelet features [33], shapelet mid-level features [34], poselets [35], and sketch tokens mid-level features [36]. Additionally, [17] explores the prior knowledge of the up-right human body shape, and [37], [20] uses spatially pooled low-level features in conjunction with the optimization of the area under the receiver operating characteristic curve only in the range of the detector evaluation.

The Deep Learning (DL) framework (i.e., DN family) has recently been applied to the PD problem. Particularly, techniques to address the occlusion problem comprise the modelling of: the visibility relationships [11], [38], the pedestrian interactions [39] and the interaction among feature extraction, deformation handling, occlusion handling and classification [3]. The background clutter and the pedestrian intra-class variability are tackled in [40] by using a model with a convolutional layer to capture lower level features, and Switchable Restricted Boltzmann Machine based layers to obtain higher level feature combinations and body salience maps.

The work of [4] initializes a CNN with the convolutional sparse coding unsupervised learning methodology. Conversely, [41] and [5] use CNNs pre-trained with the Imagenet dataset [42].

Some methods (mainly in the DN family) might not comprise the entire sliding window detector architecture, lacking the scanning of the image with detection windows at multiple scales and the non maximal suppression. As a result, they require the use of other methods to output candidate windows (with potential pedestrians), in order to reduce the computational effort and lead to runtime speed ups. Accordingly, the works published in [3], [43], [40] adopt this procedure by applying a HOG, CSS and a linear SVM detector to generate candidate images and only afterwards, entering this candidate windows into the deep network. Additionally, [5] uses the ACF detector [15], and the Squares Channel Features detector [16] as the candidate windows selection methods.

Conversely, although applied to the object detection task, [2] convolves a network with full images, being able to not only classify but also locate objects, without requiring the use of other methods to select candidate windows.

Although not being present in the PD Deep Learning literature, the combination of multiple inputs is a promising idea that has the potential to be successful in PD. In [44], the learning settings: multimodal fusion, cross modality learning and shared representation learning are used to learn audio and video information. Furthermore, the work of [41] addresses multiview mammogram analysis, by extracting features from different mammogram views CNN models and combining them in a joint CNN model. This methodology [41] leads to significant improvements, which shows the relevance of combining multiple input channels.

I-B Proposed Methodology

The proposed methodology uses a combination of the Aggregated Channel Features (ACF) [15] detector and Convolutional Neural Networks to perform the PD task.

The ACF detector performs the tasks of a regular sliding window detector, by generating candidate windows (i.e., potentially containing pedestrians) and the associated scores. Subsequently, these windows are entered in the CNN model in order to refine the classification.

Besides the regular CNN use (single channel input), an innovative method is implemented based on the combination of features extracted from top level layers of CNN models obtained with different inputs (multichannel combination). Particularly, this method is applied to the challenging low resolution PD problem.

II. Background

II-A Introduction to CNNs

CNNs are variants of Neural Networks (NN), specially designed to exploit structured input data with lattice-like topology (e.g. sound, images and video) [45]. They have the ability to handle high dimensional inputs, attain invariance to slight changes and reduce the number of connections and parameters, being more robust to overfitting.

More formally, a CNN consists in a multi-layer processing architecture that can be regarded as a function f assigning input data, denoted by $\mathbf{x}_0 \in \mathbb{R}^{H_0 \times W_0 \times D_0}$ (where the subscript 0 denotes the raw input data before any layer computation), to an output, denoted by $\hat{y} \in \{1, \ldots, C\}$ (where C is the number of classes) [46], as follows: $f(\mathbf{x}_0, \overline{\mathbf{w}}) = f_L(\ldots f_2(f_1(\mathbf{x}_0, \mathbf{w}_1, \mathbf{b}_1), \mathbf{w}_2, \mathbf{b}_2) \ldots, \mathbf{w}_L, \mathbf{b}_L)$, where $\overline{\mathbf{w}} = [\mathbf{w}, \mathbf{b}] = [\overline{\mathbf{w}}_1, \ldots, \overline{\mathbf{w}}_L] = [\mathbf{w}_1, \mathbf{b}_1, \ldots, \mathbf{w}_L, \mathbf{b}_L]$ represents the parameters, which include the weights (denoted by $\mathbf{w}$) and the biases (denoted by $\mathbf{b}$).

Starting from the input $\mathbf{x}_0 \in \mathbb{R}^{H_0 \times W_0 \times D_0}$, the output of each network module is respectively denoted by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L$, where $\mathbf{x}_l \in \mathbb{R}^{H_l \times W_l \times D_l}$, $\mathbf{w}_l \in \mathbb{R}^{H'_l \times W'_l \times D_l \times Q_l}$ and $\mathbf{b}_l \in \mathbb{R}^{Q_l}$, for an arbitrary module l (as shown in Figure 2) [46]. H_l, W_l, D_l denote the height, width and depth of the feature map (respectively), H'_l and W'_l denote the height and width of the weights (or filters), respectively, and Q_l denotes the number of weights (or filters) and biases.

The most common CNN modules are: (i) convolution layers, (ii) fully connected layers, (iii) pooling layers (e.g. max or average pooling), (iv) activation functions (e.g. hyperbolic tangent, sigmoid function and rectified linear unit (ReLU)), (v) different types of normalization (e.g. contrast normalization or cross-channel normalization), (vi) classification layers (e.g. softmax classification layer) and (vii) loss functions (which are appended to the core architecture of the CNN in order to train the network, e.g. cross entropy loss or hinge loss).

Convolutional layer: allows to explore the properties of data with lattice-like topology in a computationally viable manner, by employing the principles of parameter sharing, feature reusability, local connectivity and equivariance to translation.

The convolution of the input $\mathbf{x}_k$ at module k, with the weigths $\mathbf{w}_k$ and bias $\mathbf{b}_k$ is given by:

$$\mathbf{x}_k(j) = \sum_{i \in \Omega(j)} \mathbf{x}_{k-1}(i) * \mathbf{w}_k(i,j) + \mathbf{b}_k(j), \qquad (1)$$

where the output location is j, the input location is i, $*$ denotes the convolution operation and $\Omega(j)$ represents the input region addresses [41].

Fully connected layer: in this case, the neurons in adjacent layers are fully connected, similarly to the case of NNs layers.

Pooling: reduces the dimensionality of the feature representations by collecting statistics about them, using pooling functions, typically in non-overlapping input patches [47]. A common choice is the max pooling function given by: $\mathbf{x}_k(j) = \max(\mathbf{x}_{k-1}(j))$ [46], [41].

Activation functions: enrich the network representational power with non-linearity and are typically placed after the convolutional layers (similar to the layers in the NN case) [47]. A popular activation function is the Rectified Linear Unit (ReLU), defined by $\mathbf{x}_k(j) = \max(0, \mathbf{x}_{k-1}(j))$ [46].

Normalization: comprises various methodologies, such as local contrast normalization and cross-channel normalization. The latter is given by: $y_i = x_i(\kappa + \alpha \sum_{t \in G(d)} x_i^2)^{-\beta}$, where $G(t) \subset \{1, \ldots, D\}$ represents a subset of the input channels, for each output channel t [46].

Classification: assigns a class to the inputs, based on scores or probabilities. In the context of CNNs, the most common choice is the softmax function given by $y_k = \frac{e^{x_k}}{\sum_{t=1}^{T} e^{x_t}}$, where $k = 1, \ldots, T$; T represents the total number of classes and y_k denotes the probability of x_k belonging to the class k, out of all the possible T classes [46].

Loss function: is applied during the training process for optimization purposes. A common choice is the softmax logarithmic loss (also known as the cross entropy loss), given by: $y = -x_c + \log \sum_{t=1}^{T} e^{x_t}$, where the ground truth class is represented by x_c and T is the total number of classes [46].

II-B CNNs training and testing formulation

The process of training a CNN aims to learn the parameters $[\mathbf{w}_1, \mathbf{b}_1, \ldots, \mathbf{w}_L, \mathbf{b}_L]$, given a set of training data $\mathcal{D} = \{(\mathbf{x}_0, y)^i\}_{i=1}^N$ (N is the number of training data samples), in order to obtain a representative model, capable of generalizing the information obtained during training, and correctly classifying new unseen inputs during the test phase. The superscript i was added to the previous $\mathbf{x}_0$ and y notation, but will be suppressed if found unnecessary. Consequently, the training problem can be cast as the following optimization problem, $\arg\min_{\overline{\mathbf{w}}} J$, detailed as:

$$\arg\min_{\mathbf{w},\mathbf{b}} \frac{1}{N} \sum_{i=1}^{N} l(f(\mathbf{x}_0^i; \mathbf{w}, \mathbf{b}, y^i) + \frac{\lambda}{2}(\|\mathbf{w}\|^2 + \|\mathbf{b}\|^2), \qquad (2)$$

where l is a loss function [1] $l : f(\mathbf{x}_0^i, \overline{\mathbf{w}}) \rightarrow \mathbb{R}$, which penalizes the errors associated with wrong class predictions (i.e., a prediction $f(\mathbf{x}_0^i; \overline{\mathbf{w}})$ is wrong if it is different from its associated label y^i, and correct otherwise) [46]. The squared Euclidean norm of the weights and biases acts as a regularization penalty, in order to reduce their size and contribute to mitigate overfitting. The parameter λ controls the relevance of the regularization, when compared with the loss term [47].

[1] The distinction between referring to the loss function or the network layer should be clarified by the context.

For training purposes, the loss function can be appended to the end of the network (as depicted in Figure 2), by composing it with the model of the CNN (i.e., $f(\mathbf{x}_0, \overline{\mathbf{w}})$), resulting in the expanded CNN model, denoted by $z = l \circ f(\mathbf{x}_0, \overline{\mathbf{w}})$ [46].

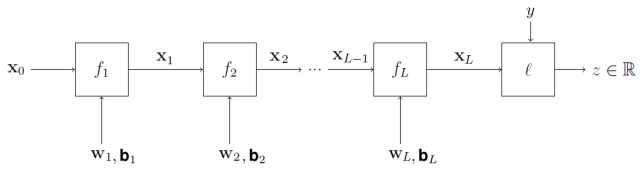

Fig. 2. Diagram of the expanded CNN model (obtained from [46]).

The optimization problem described in Equation 2 can be solved by using backpropagation with stochastic gradient descent (SGD),or its variant mini-batch SGD with momentum, since it typically processes substantial amounts of training data [46].

Indeed, resorting to mini-batch SGD with momentum [48], the weights (and similarly to the biases) of the objective function J are updated according to:

$$\mathbf{m}_{t+1} = \mu_t \mathbf{m}_t + \eta_t \frac{\partial J}{\partial \mathbf{w}_t}, \tag{3a}$$

$$\mathbf{w}_{t+1} = \mathbf{w}_t - \mathbf{m}_{t+1}, \tag{3b}$$

where $\mu_t \in]0, 1]$ is the momentum value at the t-th iteration, η_t is the learning rate at the t-th iteration, $\mathbf{m}_t$ is the momentum term at the t-h iteration and $\mathbf{w}_t$ are the weights at the t-th iteration [49], [48] (the module subscript was temporarily replaced by the iteration subscript for simplicity reasons, since the equations are valid for every layer). Other hyperparameters to select are the number of epochs (one epoch corresponds to a complete sweep across the training set) and the batch size (the number of training examples used at a certain iteration occurring in each epoch) [49], [48]. The parameters are initialized randomly or transferred from a learned representation (as discussed in Subsection II-C). The hyperparameters can be obtained through cross-validation or from reference values and procedures [47].

The derivative of the regularization term is simple to calculate. However, to compute the partial derivative of the loss function term of the objective function J with respect to the parameters of an arbitrary layer l, it is necessary to perform a forward pass in the expanded CNN model and apply the chain rule in conjunction with backpropagation from the end of the network to the beginning, as follows:

$$\frac{\partial z}{\partial (\text{vec } \mathbf{w}_l)^T} = \frac{\partial z}{\partial (\text{vec } \mathbf{x}_L)^T} \frac{\partial \mathbf{x}_L}{\partial (\text{vec } \mathbf{x}_{L-1})^T} \cdots \frac{\partial \mathbf{x}_{l+1}}{\partial (\text{vec } \mathbf{x}_l)^T} \frac{\partial \mathbf{x}_l}{\partial (\text{vec } \mathbf{w}_l)^T}, \tag{4}$$

where the vec function reshapes its argument to a column vector and the derivatives are computed at the working point, set when the input was forwarded through the network [46].

The intermediate matrices involved in these calculations lead to expensive and even unfeasible computations [46]. This problem can be solved by recursively backpropagating the derivate $\frac{\partial z}{\partial (\text{vec } \mathbf{x}_l)^T}$ from the current layer l, without computing these matrices explicitly (as detailed in [46]).

In the test phase, the input to test is forwarded through the previously trained original CNN model (without the loss function), outputting a prediction, which is compared with the corresponding label to determine the test error.

II-C Transfer Learning

In image trained CNNs, the features are organized according to a hierarchy, with the lower level features being more general and the high level features more class specific [47]. Therefore, the features of CNNs trained with a certain base task dataset (starting from random initialization), can be transferred to another network devoted to a distinct task. Subsequently, this network denoted as pre-trained, can be trained with another dataset (the task specific dataset), in a process called fine-tuning [50], [47].

CNNs are able to learn rich feature representations from raw inputs (such as image pixels). These features can be extracted from the top-level layers of a (pre-trained and fine-tuned) network and used to train a classifier. Images from the task specific dataset can be forwarded through the same network, followed by the extraction of features (from the same layer as before) and testing on the previously trained classifier [51].

III. Proposed Methodology

The proposed methodology uses each one of the several input channels to fine-tune a pre-trained CNN model, and then extracts high level features from the layer right before the classifier. These high level features are combined (partially or entirely) to train another CNN model (multichannel input CNN model), which classifies the test candidate windows generated by the ACF method.

III-A Datasets and Input channels

Two types of datasets are considered: the dataset containing pedestrian images (which is the INRIA dataset [9]) and the dataset used to pre-train the CNN model (a subset of the Imagenet dataset [42], comprising 1000 object categories with approximately 1,2 million training images, 50 thousand validation images and 100 thousand test images, and whose images range from birds to vehicles).

Assuming the following notation for the pedestrian dataset: $\mathcal{D} = \{\mathbf{m}^{(i)}, \mathbf{x}^{(i)}, \mathbf{y}^{(i)}\}_{i=1}^{N}$ where $\mathbf{m}$ denotes an image, i.e., $\mathbf{m} : \Omega \rightarrow \mathbb{R}$ with Ω denoting the image lattice; $\mathbf{x}$ are the input channels of $\mathbf{m}$, where $\mathbf{x} = \{\mathbf{x}_{\text{RGB}}, \mathbf{x}_{\text{GM}}, \mathbf{x}_{\text{Gx}}, \mathbf{x}_{\text{Gy}}, \mathbf{x}_{\text{Gs}}, \mathbf{x}_{\text{YUV}}, \mathbf{x}_{\text{LUV}}\}$. The above notation stands for different input channels obtained from $\mathbf{m}$ as follows: RGB, Gradient Magnitude, Horizontal derivative (along all 3 dimensions of the depth), Vertical derivative (along all 3 dimensions of the depth), Grayscale, YUV and LUV, respectively; $y \in \mathcal{Y} = \{1, 2\}$ denotes the class, that is, if $y = 1$, $\mathbf{m}$ does not contain a pedestrian and if $y = 2$, $\mathbf{m}$ contains a pedestrian; i indexes the ith image out of N, used for training. All the input channels are three dimensional, except for $\mathbf{x}_{\text{GM}}$ and $\mathbf{x}_{\text{Gs}}$ which are two dimensional. Figure 3 illustrates the input channels used.

Similar notation is adopted to denote the Imagenet dataset used to pre-train the CNN model: $\widetilde{\mathcal{D}} = \{(\widetilde{\mathbf{x}})^{(n)}, (\widetilde{y})^{(n)}\}_n$, with $\widetilde{\mathbf{x}} : \Omega \rightarrow \mathbb{R}$ and $\widetilde{y} \in \mathcal{Y} = \{1, \ldots, 1000\}$.

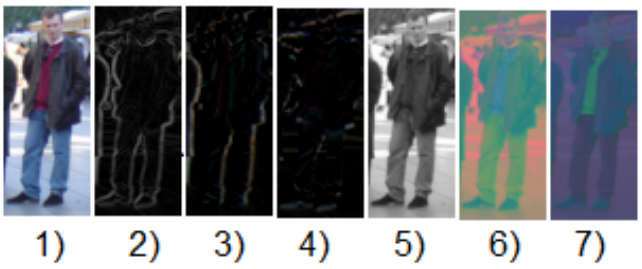

Fig. 3. Example images of the input channels: 1) RGB colorspace ($\mathbf{x}_{RGB}$); 2) gradient magnitude ($\mathbf{x}_{GM}$); 3) horizontal derivative ($\mathbf{x}_{Gx}$); 4) vertical derivative ($\mathbf{x}_{Gy}$); 5) grayscale ($\mathbf{x}_{Gs}$); 6) YUV colorspace ($\mathbf{x}_{YUV}$); 7) LUV colorspace ($\mathbf{x}_{LUV}$).

III-B Pre-trained CNN model

The publicly available model CNN-F [52], pre-trained with Imagenet, and represented by $f(\widetilde{\mathbf{x}}, \widetilde{\mathbf{w}})$ with parameters $\widetilde{\mathbf{w}} = [\widetilde{\mathbf{w}}_{cn}, \widetilde{\mathbf{w}}_{fc}, \widetilde{\mathbf{w}}_{cl}]$, was used. Its architecture is depicted in Figure 4 and contains a total of 8 layers: 5 being convolutional (including non-linear sub-sampling and parameters $\widetilde{\mathbf{w}}_{cn}$), 2 being fully connected (with parameters $\widetilde{\mathbf{w}}_{fc}$) and the last 1 being a classification layer (also fully connected with parameters $\widetilde{\mathbf{w}}_{cl}$).

More specifically, the CNN input size is 224x224x3 and the first convolutional layer comprises: 64 filters with a 11x11 receptive field, convolutional stride 4 and no zero spatial padding, followed by local response normalization and max-pooling with a downsampling factor of 2 and zero padding on the bottom and on the right. The second convolutional layer contains: 256 filters with a 5x5 receptive field, convolutional stride 1 and two levels of zero spatial padding, followed by local response normalization and max-pooling with a downsampling factor of 2. The third, fourth and fifth convolutional layers comprise: 256 filters with a 3x3 receptive field, convolutional stride 1 and one level of zero spatial padding. The remaining of the fifth layer contains max-pooling with a downsampling factor of 2. The sixth and seventh layers are fully connected with size 4096. The eight and last layer is a softmax classifier of size 1000. All layers containing parameters (except for the last one) have the Rectified Linear Unit as the activation function.

III-C CNN single input model

To construct the CNN model for the target task of pedestrian detection, the parameters, $\widetilde{\mathbf{w}}_{cn}$ and $\widetilde{\mathbf{w}}_{fc}$, from the pre-trained model (from layers 1 to 7) were transferred to the pedestrian detection task CNN model.

The architecture of the pedestrian detection task CNN model is equal to the pre-trained model's architecture, with the exception of the last layer, which was replaced by a new softmax classification layer (having parameters $\mathbf{w}_{cl}$ and randomly initialized using a Gaussian distribution with zero mean and variance equal to 0,01), designed for two classes (i.e., pedestrian and non pedestrian).

Each of the seven previously mentioned input channels was used to fine-tune a CNN (with the logarithmic loss function), resulting in a total of seven single input CNN models: 1) for RGB color space, $f(\mathbf{x}_{RGB}, \mathbf{w}_{RGB})$; 2) for Gradient Magnitude, $f(\mathbf{x}_{GM}, \mathbf{w}_{GM})$; 3) for the Horizontal derivative, $f(\mathbf{x}_{Gx}, \mathbf{w}_{Gx})$; 4) for the Vertical derivative, $f(\mathbf{x}_{Gy}, \mathbf{w}_{Gy})$; 5) for grayscale, $f(\mathbf{x}_{Gs}, \mathbf{w}_{Gs})$; 6) for YUV color space, $f(\mathbf{x}_{YUV}, \mathbf{w}_{YUV})$; 7) for LUV color space, $f(\mathbf{x}_{LUV}, \mathbf{w}_{LUV})$. This process is illustrated in Figure 5 a).

For the two dimensional inputs, the image was replicated to fill the third dimension. Before entering the CNN, all images were resized to the size 224x224x3 using cubic interpolation. Prior to these two operations, the mean of the training images was computed and removed from all train,validation and test images, as a normalization approach.

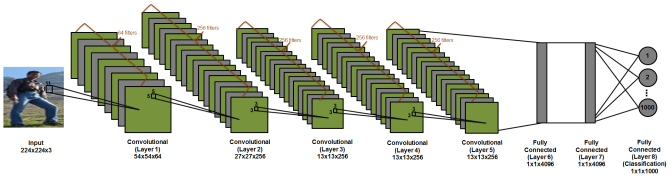

Fig. 4. Pre-trained model architecture (obtained from [48]).

III-D CNN multichannel input combination model

To build the multichannel combination CNN model, the features from the seventh layer (designated by the $L-1$th layer) of each of the seven single input CNN models are extracted and combined among themselves (entirely or partially). These features are then utilized to train (by minimizing the logarithmic loss function) a softmax classification layer (randomly initialized using a Gaussian distribution with zero mean and variance equal to 0,01), originating the multichannel input combination CNN model represented by: $f(\mathbf{x}_{RGB,L-1}, \mathbf{x}_{GM,L-1}, \mathbf{x}_{Gx,L-1}, \mathbf{x}_{Gy,L-1}, \mathbf{x}_{Gs,L-1}, \mathbf{x}_{YUV,L-1}, \mathbf{x}_{LUV,L-1}; \mathbf{w}_{cl})$ (similarly to the procedure in [41]) and depicted in Figure 5 b).

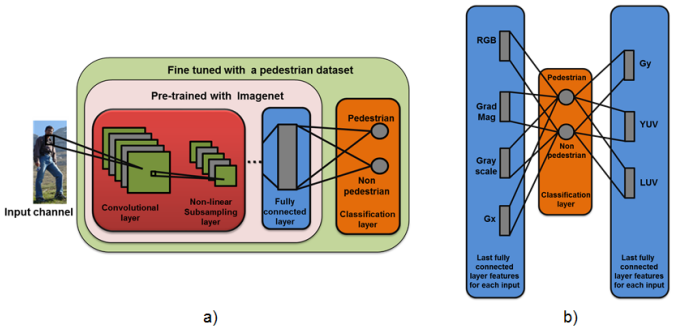

Fig. 5. Scheme of the a) single input and b) multiple input combination CNN models.

III-E Overall detection methodology

The overall detection process comprises two parts, as shown in Figure 6. The ACF detector [15] is applied to the test images to generate pedestrian candidate windows (i.e., windows potentially containing pedestrians), outputted in the form

of bounding boxes that surround each detected person, with corresponding confidence scores.

Next, each of the candidate windows is extracted from the full image and passed through the multiple input combination CNN model previously described, being classified as pedestrian or non pedestrian. The candidate windows classified as non pedestrians are discarded and the ones classified as pedestrians maintain the bounding box and confidence score obtained with the ACF detector. Then, the bounding box and the confidence score are utilized to perform the per-image evaluation of the overall detector (described in Subsection IV-A of Section IV).

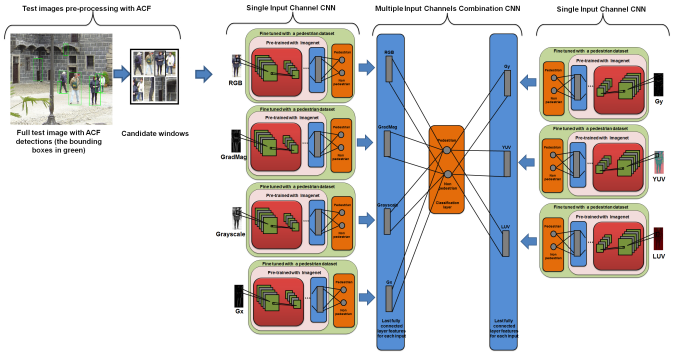

Fig. 6. Scheme of the overall detection methodology.

III-F Implementation

The developed method was implemented in Matlab and the MatConvNet Matlab toolbox [46] was used to implement the CNN framework. The Piotr's Computer Vision Matlab Toolbox [53] (2014, version 3.40) was utilized for the ACF detection, to assess the performance and for benchmarking.

IV. Experiments

IV-A Performance evaluation methodology

The per-image evaluation metric was used to evaluate the performance of the proposed method, since it was proved to be more suitable than the per-window metric [30] and is used for the most recent benchmarks in PD. In the selected evaluation methodology, a detection window is slided on a grid of locations throughout a full image and a bounding box and a confidence score are outputted for each multiscale detection, followed by non-maximal suppression (NMS) and the application of the PASCAL measure [30], expressed as:

$$a_o = \frac{area(BB_{dt} \cap BB_{gt})}{area(BB_{dt} \cup BB_{gt})} > 0,5, \quad (5)$$

meaning that the overlapping area (a_o) among the two bounding boxes must be greater than 50% [30]. Only one match between each BB_{dt} and BB_{gt} is allowed [30]. The first matches occur between the detection bounding boxes (BBs_{dt}) having the highest confidence scores and the corresponding ground truth bounding boxes (BBs_{gt}) (according to the PASCAL measure described previously) [30]. After the previous matches, a single BB_{dt} might have been matched to several BBs_{gt}. This constitutes an ambiguity that is solved by selecting the highest overlapping match (ties are solved by arbitrary selection) [30]. False positives correspond to unmatched BB_{dt} and false negatives correspond to unmatched BB_{gt} [30].

The detectors performance can be illustrated with a curve, portraying the variation of the miss rate (MR) with the false positives per image (FPPI), when the threshold established for the detection confidence score is changed [30]. These curves have logarithmic scales and are particularly useful for detector benchmarking purposes [30]. In order to concisely express the information contained in the MR against FPPI performance curve, the log-average miss rate was introduced (henceforth, referred to simply as miss rate) [30], which is the average of nine miss rate values selected for nine FPPI rates belonging to the interval $[10^{-2}, 10^{0}]$ [30]. More specifically, each of these FPPI rates corresponds to a equally distributed point (in the logarithmic domain) for the mentioned interval [30]. In the special case where a certain miss rate value cannot be computed for a certain FPPI rate, because the curve ended before sweeping through the entire FPPI rates interval (i.e., $[10^{-2}, 10^{0}]$), it is replaced by the minimum miss rate obtained [30].

IV-B Experimental setup

The experiments were performed on the INRIA dataset [9]. Two main experimental setups were considered: one including the entire INRIA training set with higher image resolution and the second one using only a small portion of the INRIA training set with lower image resolution. The test sets have the same number of images, but different resolutions, i.e., higher resolution for the higher resolution case and lower resolution for the lower resolution case.

In the first experimental setup, designated by full INRIA dataset, 17128 images were used for training (90%) and validation (10%). In this set of 17128 images, 12180 are negative (i.e., without pedestrians) and the remaining 4948 images include: 1237 positive images (i.e., containing pedestrians), its horizontal flipping or mirroring (another 1237 images) and random deformations applied to the previous 2474 images (positive images and its horizontal flipping).The deformations consist in performing cubic interpolation between randomly chosen starting and end values for the width and height of the image (obtained from a uniform distribution on the interval $]0, 15[$), but preserving the original size. The size of the images is 100 for the height and 41 for the width, for all input channels, corresponding to the higher resolution images.

In the second experimental setup, designated by partial INRIA dataset, 400 images were used for training (75%) and validation (25%). In this set of 400 images, 300 are negative and the remaining 100 images are positive, without data augmentation (no horizontal flipping and no random deformations), which were randomly chosen from the entire set of positive images. The RGB images result from resizing the ACF detections from 100x41x3 to 25x10x3. The gradient magnitude and the gradient histogram result from the ACF features computation applied to the 100x41x3 RGB images and then shrinked by a factor of 4 (i.e., with size 25x10).

The Felzenszwalb's histogram of oriented gradients [12] was applied to the 100x41x3 RGB images, having size 12x5x32 and was then reshaped to the size 32x20x3. These are the lower resolution images.

The negative images (i.e., without pedestrians) result from randomly extracting windows of size 100x41x3 from the INRIA negative training images (i.e., 10 windows per negative full image).

To obtain the positive images (for training and validation), the ACF detector is applied to the INRIA positive training dataset and the bounding boxes corresponding to the true positives are selected. The original false negatives bounding boxes are extracted as well. The true positives and the false negatives constitute the total set of positive images.

The test set contains 1835 images which correspond to candidate windows obtained by running the ACF detector in the INRIA 288 positive full test images.

IV-C Results

Training and testing were performed using the Matlab-based MatConvNet toolbox [46] running on CPU mode (no GPU was used) on 2,50 GHz Intel Core i7-4710 HQ with 12 GB of RAM and 64 bit architecture. No cross validation technique was used.

IV-C1 PD using the full INRIA dataset with 7 input channels: Each single input CNN model spent approximately 4,5 hours to train and 3 minutes to test. The training process only concerns the fine-tuning of the entire network with the INRIA [9] pedestrian dataset. The time spent pre-training the initial CNN-F model [52] with a subset of the Imagenet [42] dataset was not taken into account. In the multichannel combination CNN model, the feature extraction per input channel took approximately from 2 hours to 3 hours and the test time was under 1 minute. However, estimating the runtime of the entire method (in frames per second) requires the inclusion of the feature extraction time, besides the classification time (calculated previously for the test phase).

The optimization algorithm used for training (included in the MatConvNet toolbox [46]) was the mini-batch stochastic gradient descent with momentum, having the following parameters: batch size equal to 100, number of epochs equal to 10, learning rate equal to 0,001 and momentum equal to 0,9.

Table I presents the results for the single input channels: RGB color space (denoted by RGB), gradient magnitude (denoted by GradMag), horizontal derivative across all RGB channels (denoted by Gx), vertical derivative across all RGB channels (denoted by Gy), grayscale (denoted by Grayscale), YUV color space (denoted by YUV) and LUV color space (denoted by LUV). The input channels with original size 100x41, which are the gradient magnitude and grayscale, are replicated in order to achieve the size 100x41x3. The other input channels having original size 100x41x3, do not undergo this operation. Before entering the CNN, all inputs are resized to the size 224x224x3 using cubic interpolation (since this is the network expected input dimensions).

Table II depicts the performance when the following 4 input channels are combined: gradient magnitude (GradMag), horizontal derivative (Gx), vertical derivative (Gy) and LUV color space (LUV) (according to the methodology described in Figure 5 of Section III). Table III shows the performance for the combination of all the 7 input channels (according to the methodology described in Figure 5 of Section III). The combination of the inputs: gradient magnitude, horizontal derivative and LUV color space provides the best result, which is further improved by changing the training parameters, only for the multichannel combination, as follows: the learning rate was changed to 0,0001, the number of epochs was changed to 80 and the batch size was changed to 2000, resulting in a miss rate % of 14,64% (not shown in Tables II and III, but presented in Figure 7).

The comparison of the results obtained with the developed method for the combination of different input channels, using the full INRIA dataset, are shown in Figure 8. In order to perform a fair comparison with the best result, the training parameters of each combination presented in Figure 8 were changed to have the same values as the best method training parameters in the multichannel combination.

The comparison of the developed method best result, which occurs for the combination of the inputs (using the full INRIA dataset): gradient magnitude (GradMag), horizontal derivative (Gx) and LUV color space (LUV), with other PD benchmarks is presented in Figure 7 (denoted by Multichannel CNN in the box).

TABLE I
MISS RATE % USING SINGLE CHANNELS AS INPUT AND WITHOUT FEATURE COMBINATIONS FOR THE FULL INRIA DATASET.

Channel	Miss Rate%
RGB	16,27
GradMag	15,24
Gx	16,42
Gy	16,31
Grayscale	15,75
YUV	16,03
LUV	15,97

TABLE II
MISS RATE% USING 4 FEATURES COMBINATIONS FOR THE FULL INRIA DATASET (THE ONES IN THE TABLE MEAN THAT THE FEATURE OF THAT CHANNEL IS PRESENT IN THE COMBINATION AND THE ZEROS REPRESENT ITS ABSENCE).

GradMag	Gx	Gy	LUV	Miss Rate%
0	0	1	1	15,97
0	1	0	1	15,94
1	0	0	1	15,09
0	1	1	0	16,12
1	0	1	0	16,04
1	1	0	0	15,62
0	1	1	1	16,02
1	0	1	1	14,91
1	1	0	1	14,82
1	1	1	0	15,99
1	1	1	1	15,77

IV-C2 PD using the partial INRIA dataset with 4 input channels: Each single input CNN model spent approximately 14 minutes to train and 1,5 minutes to test. The training

TABLE III
MISS RATE% USING 7 FEATURES COMBINATIONS FOR THE FULL INRIA DATASET (THE ONES IN THE TABLE MEAN THAT THE FEATURE OF THAT CHANNEL IS PRESENT IN THE COMBINATION AND THE ZEROS REPRESENT ITS ABSENCE).

RGB	GradMag	Grayscale	Gx	Gy	YUV	LUV	Miss Rate%
1	1	1	1	1	1	1	16,70
1	1	0	1	0	0	1	16,55
1	1	0	1	1	0	1	16,70
0	0	1	1	1	1	1	15,95
1	1	0	1	1	1	1	16,00
1	0	1	0	1	1	1	16,75
0	1	0	1	0	0	1	14,82
0	1	0	1	1	0	1	15,77
0	1	1	1	1	0	1	15,92
0	1	0	1	1	1	1	15,89
0	1	0	0	1	0	1	14,91
0	1	0	1	0	0	0	15,62
0	0	0	1	0	0	1	15,94
1	1	0	1	0	0	0	16,46
0	1	1	1	0	0	1	16,22

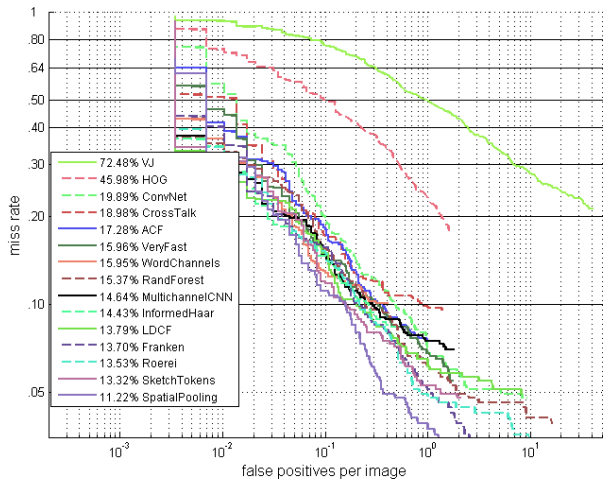

Fig. 7. Comparison of the developed method best result, denoted by Multichannel CNN, with other PD benchmarks for the full INRIA dataset. The box contains the log-average miss rate % for each method.

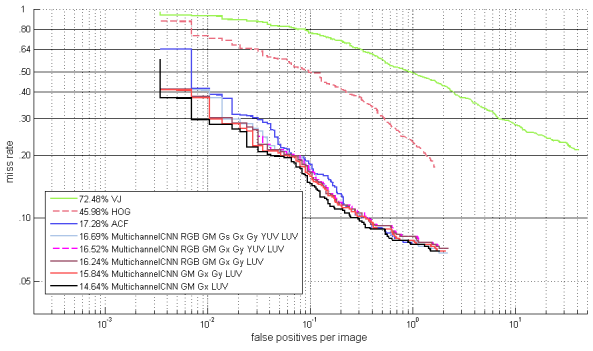

Fig. 8. Comparison of the results obtained with the developed method for the combination of different input channels for the full INRIA dataset. The box contains the log-average miss rate % for each method

process only concerns the fine-tuning of the entire network with the INRIA [9] pedestrian dataset. The time spent pre-training the initial CNN-F model [52] with a subset of the Imagenet [42] dataset was not taken into account. In the multichannel combination CNN model, the feature extraction per input channel took approximately 10 minutes and the test time was under 1 minute. However, estimating the runtime of the entire method (in frames per second) requires the inclusion of the feature extraction time, besides the classification time (calculated previously for the test phase).

The optimization algorithm used for training was the mini-batch stochastic gradient descent with momentum (included in the MatConvNet toolbox [46]), having the following parameters: batch size equal to 10, number of epochs equal to 10, learning rate equal to 0,001 and momentum equal to 0,9.

Table IV presents the results for the single input channels: RGB color space with size 25x10x3 (denoted by RGB), gradient magnitude with size 25x10 (denoted by GradMag), gradient histogram in the orientation range from 150 degrees to 180 degrees with size 25x10 (denoted by GradHist6) and the reshaped Felzenszwalb's histogram of oriented gradients [12] with size 32x20x3 (denoted by FHOG). The performance for the combination of these input channels is depicted in Table V. For the two dimensional input channels, the image is replicated in order to fill the third dimension. Before entering the CNN, all inputs are resized to the size 224x224x3 using cubic interpolation (since this is the network expected input dimensions).

TABLE IV
MISS RATE % USING SINGLE CHANNELS AS INPUT AND WITHOUT FEATURE COMBINATIONS FOR THE PARTIAL INRIA DATASET.

Channel	Miss Rate%
RGB	21,05
GradMag	24,23
GradHist6	21,83
FHOG	22,34

TABLE V
MISS RATE% USING 4 FEATURES COMBINATIONS FOR THE PARTIAL INRIA DATASET (THE ONES IN THE TABLE MEAN THAT THE FEATURE OF THAT CHANNEL IS PRESENT IN THE COMBINATION AND THE ZEROS REPRESENT ITS ABSENCE).

RGB	GradMag	GradHist6	FHOG	Miss Rate%
0	0	1	1	25,62
0	1	0	1	25,17
1	0	0	1	21,04
0	1	1	0	23,52
1	0	1	0	23,46
1	1	0	0	23,82
0	1	1	1	23,40
1	0	1	1	18,68
1	1	0	1	18,44
1	1	1	0	21,76
1	1	1	1	19,95

IV-C3 Discussion: By analyzing Table I, Table II and Table III, it is possible to observe that the combination of multiple input channels is advantageous and can lead to improved results. In fact, the best result is 14,64 % miss rate

and was obtained for the combination of GradMag, Gx and LUV input channels (individually these channels have worst performances, i.e., the miss rate of GradMag is 15,24%, the miss rate of Gx is 16,42%, and the miss rate of LUV is 15,97%).

However, some combinations of the input channels lead to miss rates that are worse than the ones of its individual input channels alone. For example, the Gx miss rate (equal to 16,42%) is better than the 7 input channels combination miss rate in Table III (equal to 16,70%), but worst than the 4 input channels combination miss rate in Table II (equal to 15,77%).

Comparing the results from Tables I, II, III, IV and V, it is relevant conjecturing that, when combining several input channels, an higher performance improvement can be obtained if each single channel has reduced quality (e.g. in terms of resolution or dimensionality), whereas higher quality single input channels seem to lead to less improvement when combined (although still existent). Concordantly, the multichannel combination maximum improvement, in the lower resolution single input channels case (shown in Tables IV and V, where images have lower dimensions than in the Tables I, II and III), is 5,79% (resulting from the difference between GradMag miss rate% and RGB, GradMag, FHOG combination miss rate%) and, in the higher resolution single input channels case (shown in Tables I, II and III), is 1,78% (resulting from the difference between Gx miss rate% and GradMag, Gx and LUV combination miss rate%). Despite the resizing to 224x224x3 underwent by all images before entering the CNN, the scarce image resolution affects the resize operation causing loss of image quality (from the higher resolution to the lower resolution case, the height and width were reduced to approximately one fourth of the initial size, except for the case of FHOG, in which the height was reduced to approximately one third and the width to one half of the initial RGB image size, before being transformed into FHOG).

According to the previous information, the proposed PD method is suited for the low resolution PD problem. However, the low resolution images were only used in the CNN based approach, since the ACF used high resolution images to generate the candidate windows. In fact, the performance for the CNN multichannel combination case cannot be compared with the performance of the ACF method alone (i.e., the baseline), because the image resolution was not reduced before the generation of the candidate windows. Nevertheless, within the scope of the CNN framework, it is still possible to compare the performance of the single input channels with the performance of the multiple input channels combinations (as done previously).

The channels combination performance improvements do not assume more significant proportions, possibly due to the lack of heterogeneity among the channels. For instance, if the input channels represented different views of the pedestrians, more noticeable differences would be expected. As a result, a better and more heterogeneous selection of the input channels may enhance the performance of their combination. Indeed, the combination of the RGB, GradMag, Gx and LUV input channels produces a miss rate (equal to 16,55%) 1,91% worst than the miss rate of the combination of GradMag, Gx and LUV (equal to 14,64%), possibly due to the redundancy existent between the colorspaces RGB and LUV.

Regarding the sensibility of the model, when the hyperparameter batch size increases, the performance tends to improve in the single channel case (mainly when the training dataset size is small), not showing significant differences in the multichannel combination case. By increasing the number of epochs over 10, there are no substantial changes in the performance (specially when the training dataset is large), since the training and validation errors are not able to substantially decrease more. Indeed, increasing the number of epochs has no substantial effect in the multichannel combination case.

Another methodology could be used to obtain positive images by cropping and resizing the INRIA positive dataset (having dimensions 100x41x3). However, the results reached with the adopted methodology (i.e., with ACF pre-processing) are superior, by less than 1% (approximately), to the ones obtained with the cropped and resized INRIA positive training methodology (whose results are not detailed herein).

The selected experimental setups, namely, the full INRIA dataset with higher resolution images and the partial INRIA dataset with lower resolution images, intend to test two extreme cases. More data and higher resolution images, allow the network to undergo better training, leading to the best result (e.g. 14,64 % miss rate for the combination of the inputs: GradMag, Gx and LUV). Conversely, when the images have lower resolution and the amount of data utilized is substantially more scarce, the training of the network is not as good, leading to the worst situation regarding the resolution and data quantity (e.g. 18,44 % miss rate for the combination of the inputs: RGB, GradMag and FHOG).

The best result achieved with the developed method occurs for the combination of the inputs (using the full INRIA dataset): GradMag, Gx and LUV (as depicted in Figures 7 and 8). When compared with other PD benchmarks, as shown in Figure 7, it is possible to conclude that the proposed approach is competitive with the state of the art, and introduces an improvement of 2,64% when compared with the ACF method alone.

V. Conclusions

An innovative PD methodology was proposed, based on the Deep Learning framework. Specifically, a CNN model pre-trained with Imagenet [42] was fine-tuned with several transformations (e.g. LUV, gradient magnitude and the original RGB) of the INRIA pedestrian dataset [9]. The final CNN multichannel input model was built by combining the penultimate layer features of each input CNN model.

Two experimental setups were adopted, namely, the full INRIA dataset with higher resolution and the partial INRIA dataset with lower resolution. Particularly, the performed experiments motivate the application of this method to low resolution PD. The experimental results obtained using the full INRIA dataset are competitive with the PD state of the art approaches. Moreover, when multiple input channels are available, the developed method can be abstractly applied in other areas beyond the scope of PD, to integrate information from several inputs.

Possible extensions consist in selecting more heterogeneous input channels (such as pedestrian body parts, which can be obtained with [35], or different views of the pedestrians), deeper architectures for the pre-trained CNN models, a distinct detector for the pedestrian candidate windows generation (e.g. Square Channels Features or the Roerei detectors [16]) and integrating the multiscale sliding window task and the pedestrian bounding box prediction task in the CNN (similarly to [2]), avoiding the use of a detector to generate candidate windows. Additionally, the low resolution results can be extended to the entire detection system, by reducing the image resolution before the application of the ACF method.

REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *NIPS*, 2012, pp. 1106–1114.

[2] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," in *ICLR*. CBLS, 2014.

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

[4] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," *CVPR*, 2013.

[5] J. H. Hosang, M. Omran, R. Benenson, and B. Schiele, "Taking a Deeper Look at Pedestrians," *CoRR*, vol. abs/1501.05790, 2015.

[6] M. Taiana, J. C. Nascimento, and A. Bernardino, "On the purity of training and testing data for learning: The case of Pedestrian Detection," *Neurocomputing*, vol. vol. 150, Part A, pp. 214–226, 2015.

[7] "Pedestrian traffic sign image," last Access: July, 2015. [Online]. Available: http://iica.de/pd/index.py

[8] "Pedestrian traffic light image," last Access: July, 2015. [Online]. Available: http://www.novuslight.com/transportation-solution-for-pedestrians-in-cologne_N1232.html

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *International Conference on Computer Vision & Pattern Recognition*, vol. 2, 2005, pp. 886–893.

[10] R. Benenson, M. Omran, J. H. Hosang, and B. Schiele, "Ten Years of Pedestrian Detection, What Have We Learned?" *CoRR*, vol. abs/1411.4304, 2014.

[11] W. Ouyang and X. Wang, "A discriminative deep model for pedestrian detection with occlusion handling," *CVPR*, 2012.

[12] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[13] S. Zhang, R. Benenson, and B. Schiele, "Filtered channel features for pedestrian detection," *CoRR*, vol. abs/1501.05759, 2015.

[14] P. Dollár, Z. Tu, P. Perona, and S. Belongie, "Integral Channel Features," in *BMVC*, 2009.

[15] P. Dollár, R. Appel, S. Belongie, and P. Perona, "Fast Feature Pyramids for Object Detection," *PAMI*, 2014.

[16] R. Benenson, M. Mathias, T. Tuytelaars, and L. Van Gool, "Seeking the strongest rigid detector," in *CVPR*, 2013.

[17] S. Zhang, C. Bauckhage, and A. B. Cremers, "Informed haar-like features improve pedestrian detection," in *CVPR*, 2014, pp. 947–954.

[18] W. Nam, P. Dollár, and J. H. Han, "Local decorrelation for improved pedestrian detection," in *NIPS*, 2014, pp. 424–432.

[19] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2004.

[20] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Pedestrian detection with spatially pooled features and structured ensemble learning," *CoRR*, vol. abs/1409.5209, 2014.

[21] E. Seemann, B. Leibe, K. Mikolajczyk, and B. Schiele, "An evaluation of local shape-based features for pedestrian detection," in *Proceedings of the British Machine Vision Conference*, 2005.

[22] C. Gu, J. J. Lim, P. Arbelaez, and J. Malik, "Recognition using regions," in *CVPR*, 2009, pp. 1030–1037.

[23] C. Papageorgiou and T. Poggio, "A trainable system for object detection," *International Journal of Computer Vision*, vol. 38, no. 1, pp. 15–33, 2000.

[24] S. Walk, N. Majer, K. Schindler, and B. Schiele, "New features and insights for pedestrian detection," in *CVPR*, 2010, pp. 1030–1037.

[25] P. Dollár, S. Belongie, and P. Perona, "The fastest pedestrian detector in the west," in *BMVC*, 2010.

[26] P. Dollár, R. Appel, and W. Kienzle, "Crosstalk Cascades for Frame-Rate Pedestrian Detection," in *ECCV*, 2012.

[27] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool, "Pedestrian detection at 100 frames per second," in *CVPR*, 2012.

[28] R. Fergus, P. Perona, and A. Zisserman, "Object class recognition by unsupervised scale-invariant learning," in *CVPR*, 2003, pp. 264–271.

[29] P. F. Felzenszwalb, D. A. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *(CVPR*, 2008.

[30] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: An Evaluation of the State of the Art," *PAMI*, vol. 34, 2012.

[31] D. Park, D. Ramanan, and C. Fowlkes, "Multiresolution models for object detection," in *ECCV*, 2010, pp. 241–254.

[32] J. Yan, X. Zhang, Z. Lei, S. Liao, and S. Z. Li, "Robust multi-resolution pedestrian detection in traffic scenes," in *CVPR*, 2013, pp. 3033–3040.

[33] B. Wu and R. Nevatia, "Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors," in *ICCV*, 2005, pp. 90–97.

[34] P. Sabzmeydani and G. Mori, "Detecting pedestrians by learning shapelet features," in *CVPR*, 2007.

[35] L. Bourdev and J. Malik, "Poselets: Body part detectors trained using 3d human pose annotations," in *ICCV*, 2009.

[36] J. J. Lim, C. L. Zitnick, and P. Dollár, "Sketch tokens: A learned mid-level representation for contour and object detection," in *CVPR*, 2013, pp. 3158–3165.

[37] S. Paisitkriangkrai, C. Shen, and A. van den Hengel, "Strengthening the effectiveness of pedestrian detection with spatially pooled features," *CoRR*, vol. abs/1407.0786, 2014.

[38] W. Ouyang, X. Zeng, and X. Wang, "Modeling mutual visibility relationship in pedestrian detection," in *CVPR*, 2013, pp. 3222–3229.

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

[40] P. Luo, Y. Tian, X. Wang, and X. Tang, "Switchable deep network for pedestrian detection," *CVPR*, 2014.

[41] G. Carneiro, J. Nascimento, and A. Bradley, "Unregistered multiview mammogram analysis with pre-trained deep learning models," *To appear in MICCAI*, 2015.

[42] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR*, 2009.

[43] W. O. X. Zeng and X. Wang, "Multi-stage contextual deep learning for pedestrian detection," *ICCV*, 2013.

[44] J. Ngiam, A. Khosla, M. Kim, J. Nam, H. Lee, and A. Y. Ng, "Multimodal deep learning," in *ICML*, 2011, pp. 689–696.

[45] Y. Bengio, I. J. Goodfellow, and A. Courville, "Deep learning," 2015, book in preparation for MIT Press. [Online]. Available: http://www.iro.umontreal.ca/~bengioy/dlbook

[46] A. Vedaldi and K. Lenc, "MatConvNet – Convolutional Neural Networks for MATLAB (including the manual)," *CoRR*, vol. abs/1412.4564, 2014.

[47] L. Fei-Fei and A. Karpathy, "Notes about Convolutional Neural Networks from the course CS231n: Convolutional Neural Networks for Visual Recognition lectured at Standford University," Winter quarter, 2015, last Access: July, 2015. [Online]. Available: http://cs231n.stanford.edu/

[48] A. Vedaldi, "AIMS Big Data, Lecture 3: Deep Learning," 2015, last Access: July, 2015. [Online]. Available: http://www.robots.ox.ac.uk/~vedaldi/assets/teach/2015/vedaldi15aims-bigdata-lecture-4-deep-learning-handout.pdf

[49] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, and S. Tandon, "Deep Learning Tutorial," last Access: July, 2015. [Online]. Available: http://uflDl.stanford.edu/tutorial

[50] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" *CoRR*, vol. abs/1411.1792, 2014.

[51] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[52] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman, "Return of the Devil in the Details: Delving Deep into Convolutional Nets," in *British Machine Vision Conference*, 2014.

[53] P. Dollár, "Piotr's Computer Vision Matlab Toolbox (PMT)," http://vision.ucsd.edu/~pdollar/toolbox/doc/index.html.