Pedestrian Detection with
Unsupervised Multi-Stage Feature Learning

Pierre Sermanet Koray Kavukcuoglu Soumith Chintala
Yann LeCun
Courant Institute of Mathematical Sciences, New York University
sermanet,koray,soumith,yann@cs.nyu.edu

Abstract

Pedestrian detection is a problem of considerable practical interest. Adding to the list of successful applications of deep learning methods to vision, we report state-of-the-art and competitive results on all major pedestrian datasets with a convolutional network model. The model uses a few new twists, such as multi-stage features, connections that skip layers to integrate global shape information with local distinctive motif information, and an unsupervised method based on convolutional sparse coding to pre-train the filters at each stage.

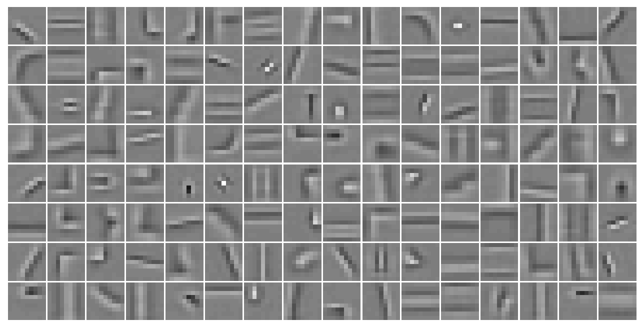

Figure 1: 128 9×9 **filters trained on grayscale INRIA images using Algorithm 1**. It can be seen that in addition to edge detectors at multiple orientations, our systems also learns more complicated features such as corner and junction detectors.

1. Introduction

Pedestrian detection is a key problem for surveillance, automotive safety and robotics applications. The wide variety of appearances of pedestrians due to body pose, occlusions, clothing, lighting and backgrounds makes this task challenging.

All existing state-of-the-art methods use a combination of hand-crafted features such as *Integral Channel Features* [9], HoG [5] and their variations [13, 33] and combinations [38], followed by a trainable classifier such as SVM [13, 28], boosted classifiers [9] or random forests [7]. While low-level features can be designed by hand with good success, mid-level features that combine low-level features are difficult to engineer without the help of some sort of learning procedure. Multi-stage recognizers that learn hierarchies of features tuned to the task at hand can be trained end-to-end with little prior knowledge. Convolutional Networks (ConvNets) [23] are examples of such hierarchical systems with end-to-end feature learning that are trained in a supervised fashion. Recent works have demonstrated the usefulness of unsupervised pre-training for end-to-end training of deep multi-stage architectures using a variety of techniques such as stacked restricted Boltzmann machines [16], stacked auto-encoders [4] and stacked sparse auto-encoders [32], and using new types of non-linear transforms at each layer [17, 20].

Supervised ConvNets have been used by a number of authors for such applications as face, hand detection [37, 29, 15, 31, 14, 36]. More recently, a large ConvNet by [21] achieved a breakthrough on a 1000-class ImageNet detection task. The main contribution of this paper is to show that the ConvNet model, with a few important twists, consistently yields state of the art and competitive results on all major pedestrian detection benchmarks. The system uses unsupervised convolutional sparse auto-encoders to pre-train features at all levels from the relatively small INRIA dataset [5], and end-to-end supervised training to train the classifier and fine-tune the features in an integrated fashion. Additionally, multi-stage features with layer-skipping connections enable output stages to combine global shape detectors with local motif detectors.

Processing speed in pedestrian detection has recently seen great progress, enabling real-time operation without sacrificing quality. [3] manage to entirely avoid image rescaling for detection while observing quality improvements. While processing speed is not the focus of this paper, features and classifier approximations introduced by [8] and [3] may be applicable to deep learning models for faster detection, in addition to GPU optimizations.

2. Learning Feature Hierarchies

Much of the work on pedestrian detection have focused on designing representative and powerful features [5, 9, 8, 38]. In this work, we show that generic feature learning algorithms can produce successful feature extractors that can achieve state-of-the-art results.

Supervised learning of end-to-end systems on images have been shown to work well when there is abundant labeled samples [23], including for detection tasks [37, 29, 15, 31, 14, 36]. However, for many input domains, it is hard to find adequate number of labeled data. In this case, one can resort to designing useful features by using domain knowledge, or an alternative way is to use unsupervised learning algorithms. Recently unsupervised learning algorithms have been demonstrated to produce good features for generic object recognition problems [24, 25, 18, 20].

In [16], it was shown that unsupervised learning can be used to train deep hierarchical models and the final representation achieved is actually useful for a variety of different tasks [32, 24, 4]. In this work, we also follow a similar approach and train a generic unsupervised model at each layer using the output representation from the layer before. This process is then followed by supervised updates to the whole hierarchical system using label information.

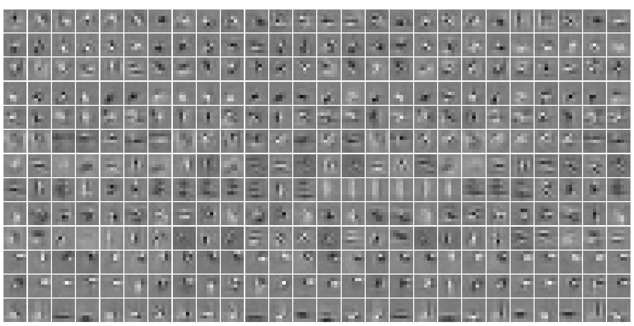

Figure 2: **A subset of 7×7 second layer filters** trained on grayscale INRIA images using Algorithm 2. Each row in the figure shows filters that connect to a common output feature map. It can be seen that they extract features at similar locations and shapes, e.g. the bottom row tends to aggregate horizontal features towards the bottom of the filters.

2.1. Hierarchical Model

A hierarchical feature extraction system consists of multiple levels of feature extractors that perform the same filtering and non-linear transformation functions in successive layers. Using a particular generic parametrized function one can then map the inputs into gradually more higher level (or abstract) representations [23, 16, 4, 32, 24]. In this work we use sparse convolutional feature hierarchies as proposed in [20]. Each layer of the unsupervised model contains a convolutional sparse coding algorithm and a predictor function that can be used for fast inference. After the last layer a classifier is used to map the feature representations into class labels. Both the sparse coding dictionary and the predictor function do not contain any hard-coded parameter and are trained from the input data.

The training procedure for this model is similar to [16]. Each layer is trained in an unsupervised manner using the representation from previous layer (or the input image for the initial layer) separately. After the whole multi-stage system is trained in a layer-wise fashion, the complete architecture followed by a classifier is fine-tuned using labeled data.

2.2. Unsupervised Learning

Recently sparse coding has seen much interest in many fields due to its ability to extract useful feature representations from data, The general formulation of sparse coding is a linear reconstruction model using an overcomplete dictionary $\mathcal{D} \in \mathbb{R}^{m \times n}$ where $m > n$ and a regularization penalty on the mixing coefficients $z \in \mathbb{R}^n$.

$$z^* = \arg\min_z \|x - \mathcal{D}z\|_2^2 + \lambda s(z) \tag{1}$$

The aim is to minimize equation 1 with respect to z to obtain the optimal sparse representation z^* that correspond to input $x \in \mathbb{R}^m$. The exact form of $s(z)$ depends on the particular sparse coding algorithm that is used, here, we use the $\|.\|_1$ norm penalty, which is the sum of the absolute values of all elements of z. It is immediately clear that the solution of this system requires an optimization process. Many efficient algorithms for solving the above convex system has been proposed in recent years [1, 6, 2, 26]. However, our aim is to also learn generic feature extractors. For that reason we minimize equation 1 wrt $\mathcal{D}$ too.

$$z^*, \mathcal{D}^* = \arg\min_{z,\mathcal{D}} \|x - \mathcal{D}z\|_2^2 + \lambda \|z\|_1 \tag{2}$$

This resulting equation is non-convex in $\mathcal{D}$ and z at the same time, however keeping one fixed, the problem is still convex wrt to the other variable. All sparse modeling algorithms that adopt the dictionary matrix $\mathcal{D}$ exploit this property and perform a coordinate descent like minimization process where each variable is updated in succession. Following [30] many authors have used sparse dictionary learning to represent images [27, 1, 19]. However, most of the sparse coding models use small image patches as input x to learn the dictionary $\mathcal{D}$ and then apply the resulting model to every overlapping patch location on the full image. This approach assumes that the sparse representation for two neighboring patches with a single pixel shift is completely independent, thus produces very redundant representations. [20, 39] have introduced convolutional sparse modeling formulations for feature learning and object recognition and we use the Convolutional Predictive Sparse Decomposition (CPSD) model proposed in [20] since it is the only convolutional sparse coding model providing a fast predictor function that is suitable for building multi-stage feature representations. The particular predictor function we use is similar to a single layer ConvNet of the following form:

$$f(x; g, k, b) = \tilde{z} = \{\tilde{z}_j\}_{j=1..n} \quad (3)$$
$$\tilde{z}_j = g_j \times \tanh(x \otimes k_j + b_j) \quad (4)$$

where $\otimes$ operator represents convolution operator that applies on a single input and single filter. In this formulation x is a $p \times p$ grayscale input image, $k \in \mathbb{R}^{n \times m \times m}$ is a set of 2D filters where each filter is $k_j \in \mathbb{R}^{m \times m}$, $g \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$ are vectors with n elements, the predictor output $\tilde{z} \in \mathbb{R}^{n \times p-m+1 \times p-m+1}$ is a set of feature maps where each of $\tilde{z}_j$ is of size $p-m+1 \times p-m+1$. Considering this general predictor function, the final form of the convolutional unsupervised energy for grayscale inputs is as follows:

$$\mathbb{E}_{CPSD} = \mathbb{E}_{ConvSC} + \beta \mathbb{E}_{Pred} \quad (5)$$
$$\mathbb{E}_{ConvSC} = \left\| x - \sum_j \mathcal{D}_j \otimes z_j \right\|_2^2 + \lambda \|z\|_1 \quad (6)$$
$$\mathbb{E}_{Pred} = \|z^* - f(x; g, k, b)\|_2^2 \quad (7)$$

where $\mathcal{D}$ is a dictionary of filters the same size as k and β is a hyper-parameter. The unsupervised learning procedure is a two step coordinate descent process. At each iteration, **(1) Inference:** The parameters $W = \{\mathcal{D}, g, k, b\}$ are kept fixed and equation 6 is minimized to obtain the optimal sparse representation z^*, **(2) Update:** Keeping z^* fixed, the parameters W updated using a stochastic gradient step: $W \leftarrow W - \eta \frac{\partial \mathbb{E}_{CPSD}}{\partial W}$ where η is the learning rate parameter. The inference procedure requires us to carry out the sparse coding problem solution. For this step we use the FISTA method proposed in [2]. This method is an extension of the original iterative shrinkage and thresholding algorithm [6] using an improved step size calculation with a momentum-like term. We apply the FISTA algorithm in the image domain adopting the convolutional formulation.

For color images or other multi-modal feature representations, the input x is a set of feature maps indexed by i and the representation z is a set of feature maps indexed by j for each input map i. We define a map of connections P from input x to features z. A j^{th} output feature map is connected to a set P_j of input feature maps. Thus, the predictor function in Algorithm 1 is defined as:

$$\tilde{z}_j = g_j \times \tanh\left(\sum_{i \in P_j} (x_i \otimes k_{j,i}) + b_j \right) \quad (8)$$

and the reconstruction is computed using the inverse map $\bar{P}$:

$$\mathbb{E}_{ConvSC} = \sum_i \|x_i - \sum_{j \in \bar{P}_i} \mathcal{D}_{i,j} \otimes z_j\|_2^2 + \lambda \|z\|_1 \quad (9)$$

For a fully connected layer, all the input features are connected to all the output features, however it is also common to use sparse connection maps to reduce the number of parameters. The online training algorithm for unsupervised training of a single layer is:

Algorithm 1 Single layer unsupervised training.

function Unsup$(x, \mathcal{D}, P, \{\lambda, \beta\}, \{g, k, b\}, \eta)$
  Set: $f(x; g, k, b)$ from eqn 8, $W^p = \{g, k, b\}$.
  Initialize: $z = \emptyset$, $\mathcal{D}$ and W^p randomly.
  repeat
    Perform **inference**, minimize equation 9 wrt z using FISTA [2]
    Do a stochastic **update** on $\mathcal{D}$ and W^p. $\mathcal{D} \leftarrow \mathcal{D} - \eta \frac{\partial \mathbb{E}_{ConvSC}}{\partial \mathcal{D}}$ and $W^p \leftarrow W^p - \eta \frac{\partial \mathbb{E}_{Pred}}{\partial W^p}$
  until convergence
  Return: $\{\mathcal{D}, g, k, b\}$
end function

2.3. Non-Linear Transformations

Once the unsupervised learning for a single stage is completed, the next stage is trained on the feature representation from the previous one. In order to obtain the feature representation for the next stage, we use the predictor function $f(x)$ followed by non-linear transformations and pooling. Following the multi-stage framework used in [20], we apply absolute value rectification, local contrast normalization and average down-sampling operations.

Absolute Value Rectification is applied component-wise to the whole feature output from $f(x)$ in order to avoid cancellation problems in contrast normalization and pooling steps.

Local Contrast Normalization is a non-linear process that enhances the most active feature and suppresses the other ones. The exact form of the operation is as follows:

$$v_i = x_i - x_i \otimes w \;, \quad \sigma = \sqrt{\sum_i w \otimes v_i^2} \quad (10)$$
$$y_i = \frac{v_i}{\max(c, \sigma)} \quad (11)$$

where i is the feature map index and w is a 9×9 Gaussian weighting function with normalized weights so that $\sum_{ipq} w_{pq} = 1$. For each sample, the constant c is set to $mean(\sigma)$ in the experiments.

Average Down-Sampling operation is performed using a fixed size boxcar kernel with a certain step size. The size of the kernel and the stride are given for each experiment in the following sections.

Once a single layer of the network is trained, the features for training a successive layer is extracted using the predictor function followed by non-linear transformations. Detailed procedure of training an N layer hierarchical model is explained in Algorithm 2.

The first layer features can be easily displayed in the parameter space since the parameter space and the input space is same, however visualizing the second and higher level features in the input space can only be possible when only

Algorithm 2 Multi-layer unsupervised training.

function **HierarUnsup**$(x, n_i, m_i, P_i, \{\lambda_i, \beta_i\}, \{w_i, s_i\}, i = \{1..N\}, \eta_i)$

  Set: $i = 1$, $X_1 = x$, $lcn(x)$ using equations 10-11, $ds(X, w, s)$ as the down-sampling operator using boxcar kernel of size $w \times w$ and stride of size s in both directions.

  repeat

    Set: $\mathcal{D}_i, k_i \in \mathbb{R}^{n_i \times m_i \times m_i}$, $g_i, b_i \in \mathbb{R}^{n_i}$.

    $\{\mathcal{D}_i, k_i, g_i, k_i, b_i\} = Unsup(X_i, \mathcal{D}_i, P_i, \{\lambda_i, \beta_i\}, \{g_i, k_i, b_i\}, \eta_i)$

    $\tilde{z} = f(X_i; g_i, k_i, b_i)$ using equation 8.

    $\tilde{z} = |\tilde{z}|$

    $\tilde{z} = lcn(\tilde{z})$

    $X_{i+1} = ds(\tilde{z}, w_i, s_i)$

    $i = i + 1$

  until $i = N$

end function

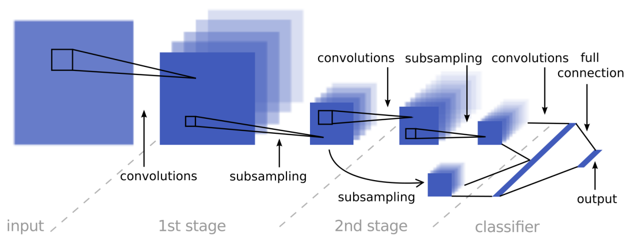

Figure 3: **A multi-scale convolutional network.** The top row of maps constitute a regular ConvNet [17]. The bottom row in which the 1st stage output is branched, subsampled again and merged into the classifier input provides a multi-stage component to the classifier stage. The multi-stage features coming out of the 2nd stage extracts a global structure as well as local details.

invertible operations are used in between layers. However, since we use absolute value rectification and local contrast normalization operations mapping the second layer features onto input space is not possible. In Figure 2 we show a subset of 1664 second layer features in the parameter space.

2.4. Supervised Training

After the unsupervised learning of the hierarchical feature extraction system is completed using Algorithm 2, we append a classifier function, usually in the form of a linear logistic regression, and perform stochastic online training using labeled data.

2.5. Multi-Stage Features

ConvNets are usually organized in a strictly feed-forward manner where one layer only takes the output of the previous layer as input. Features extracted this way tend to be high level features after a few stages of convolutions and subsampling. By branching lower levels' outputs into the top classifier (Fig. 3), one produces features that extract both global shapes and structures and local details, such as a global silhouette and face components in the case of human detection. Contrary to [12], the output of the first stage is branched after the non-linear transformations and pooling/subsampling operations rather than before.

We also use color information on the training data. For this purpose we convert all images into YUV image space and subsample the UV features by 3 since color information is in much lower resolution. Then at the first stage, we keep feature extraction systems for Y and UV channels separate. On the Y channel, we use 32 7×7 features followed by absolute value rectification, contrast normalization and 3×3 subsampling. On the subsampled UV channels, we extract 6 5×5 features followed by absolute value rectification and contrast normalization, skipping the usual subsampling step since it was performed beforehand. These features are then concatanated to produce 38 feature maps that are input to the first layer. The second layer feature extraction takes 38 feature maps and produces 68 output features using 2040 9×9 features. A randomly selected 20% of the connections in mapping from input features to output features is removed to limit the computational requirements and break the symmetry [23]. The output of the second layer features are then transformed using absolute value rectification and contrast normalization followed by 2×2 subsampling. This results in 17824 dimensional feature vector for each sample which is then fed into a linear classifier.

In Table 1, we show that multi-stage features improve accuracy for different tasks, with different magnitudes. Greatest improvements are obtained for pedestrian detection and traffic-sign classification while only minimal gains are obtained for house numbers classification, a less complex task.

2.6. Bootstrapping

Bootstrapping is typically used in detection settings by multiple phases of extracting the most offending negative answers and adding these samples to the existing dataset while training. For this purpose, we extract 3000 negative samples per bootstrapping pass and limit the number of most offending answers to 5 for each image. We perform 3 bootstrapping passes in addition to the original training phase (i.e. 4 training passes in total).

2.7. Non-Maximum Suppression

Non-maximum suppression (NMS) is used to resolve conflicts when several bounding boxes overlap. For both INRIA and Caltech experiments we use the widely accepted PASCAL overlap criteria to determine a matching score between two bounding boxes ($\frac{intersection}{union}$) and if two boxes overlap by more than 60%, only the one with the highest score is kept. In [10]'s addendum, the matching criteria is modified by replacing the union of the two boxes with the minimum of the two. Therefore, if a box is fully contained in another one the small box is selected. The goal for this

Task	Single-Stage features	Multi-Stage features	Improvement %
Pedestrians detection (INRIA) (Fig. 4)	23.39%	17.29%	26.1%
Traffic Signs classification (GTSRB) [35]	1.80%	0.83%	54%
House Numbers classification (SVHN) [34]	5.54%	5.36%	3.2%

Table 1: **Error rates improvements of multi-stage features over single-stage features** for different types of objects detection and classification. Improvements are significant for multi-scale and textured objects such as traffic signs and pedestrians but minimal for house numbers.

modification is to avoid false positives that are due to pedestrian body parts. However, a drawback to this approach is that it always disregards one of the overlapping pedestrians from detection. Instead of changing the criteria, we actively modify our training set before each bootstrapping phase. We include body part images that cause false positive detection into our bootstrapping image set. Our model can then learn to suppress such responses within a positive window and still detect pedestrians within bigger windows more reliably.

3. Experiments

We evaluate our system on 5 standard pedestrian detection datasets. However, like most other systems, we only train on the INRIA dataset. We also demonstrate improvements brought by unsupervised training and multi-stage features. In the following we name our model **ConvNet** with variants of unsupervised (Convnet-U) and fully-supervised training (Convnet-F) and multi-stage features (Convnet-U-MS and ConvNet-F-MS).

3.1. Data Preparation

The ConvNet is trained on the INRIA pedestrian dataset [5]. Pedestrians are extracted into windows of 126 pixels in height and 78 pixels in width. The context ratio is 1.4, i.e. pedestrians are 90 pixels high and the remaining 36 pixels correspond to the background. Each pedestrian image is mirrored along the horizontal axis to expand the dataset. Similarly, we add 5 variations of each original sample using 5 random deformations such as translations and scale. Translations range from -2 to 2 pixels and scale ratios from 0.95 to 1.05. These deformations enforce invariance to small deformations in the input. The range of each deformation determines the trade-off between recognition and localization accuracy during detection. An equal amount of background samples are extracted at random from the negative images and taking approximately 10% of the extracted samples for validation yields a validation set with 2000 samples and training set with 21845 samples. Note that the unsupervised training phase is performed on this initial data before the bootstrapping phase.

3.2. Evaluation Protocol

During testing and bootstrapping phases using the INRIA dataset, the images are both up-sampled and sub-sampled. The up-sampling ratio is 1.3 while the sub-sampling ratio is limited by 0.75 times the network's minimum input (126×78). We use a scale stride of 1.10 between each scale, while other methods typically use either 1.05 or 1.20 [11]. A higher scale stride is desirable as it implies less computations.

For evaluation we use the bounding boxes files published on the Caltech Pedestrian website [1] and the evaluation software provided by Piotr Dollar (version 3.0.1). In an effort to provide a more accurate evaluation, we improved on both the evaluation formula and the INRIA annotations as follows. The evaluation software was slightly modified to compute the continuous area under curve (AUC) in the entire [0, 1] range rather than from 9 discrete points only (0.01, 0.0178, 0.0316, 0.0562, 0.1, 0.1778, 0.3162, 0.5623 and 1.0 in version 3.0.1). Instead, we compute the entire area under the curve by summing the areas under the piecewise linear interpolation of the curve, between each pair of points. In addition, we also report a 'fixed' version of the annotations for INRIA dataset, which has missing positive labels. The added labels are only used to avoid counting false errors and wrongly penalizing algorithms. The modified code and extra INRIA labels are available at [2]. Table 2 reports results for both original and fixed INRIA datasets. Notice that the continuous AUC and fixed INRIA annotations both yield a reordering of the results (see supplementary material for further evidence that the impact of these modifications is significant enough to be used). To avoid ambiguity, all results with the original discrete AUC are reported in the supplementary paper.

To ensure a fair comparison, we separated systems trained on INRIA (the majority) from systems trained on TUD-MotionPairs and the only system trained on Caltech in table 2. For clarity, only systems trained on INRIA were represented in Figure 5, however all results for all systems are still reported in table 2.

3.3. Results

In Figure 4, we plot DET curves, i.e. miss rate versus false positives per image (FPPI), on the fixed INRIA dataset and rank algorithms along two measures: the error rate at 1 FPPI and the area under curve (AUC) rate in the [0, 1] FPPI range. This graph shows the indi-

[1] http://www.vision.caltech.edu/Image_Datasets/CaltechPedestrians
[2] http://cs.nyu.edu/~sermanet/data.html#inria

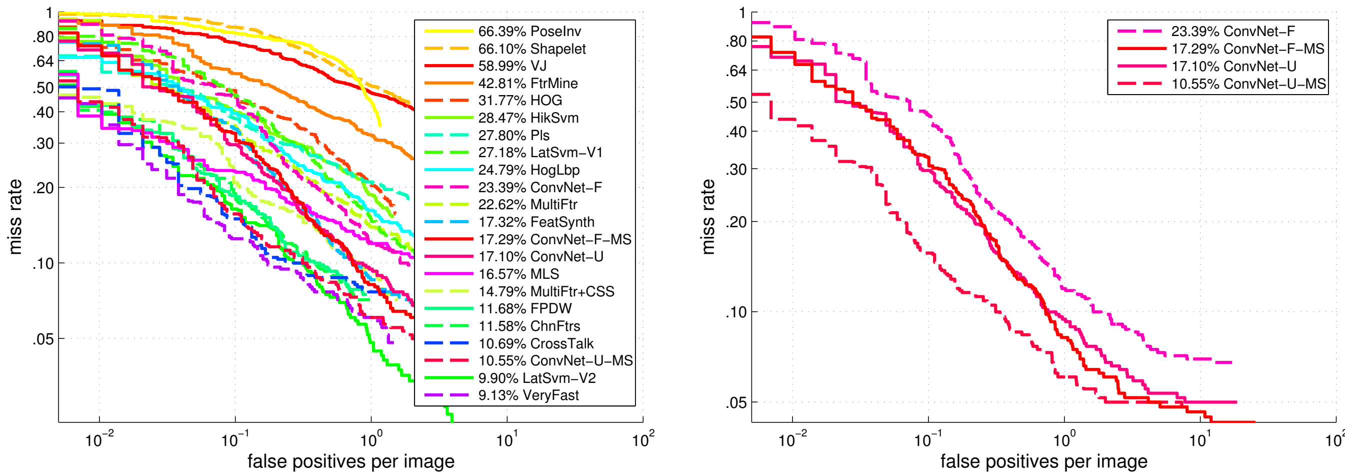

Figure 4: **DET curves on the fixed-INRIA dataset for large pedestrians measure** report false positives per image (FPPI) against miss rate. Algorithms are sorted from top to bottom using the proposed continuous area under curve measure between 0 and 1 FPPI. **On the right, only the ConvNet variants are displayed to highlight the individual contributions of unsupervised learning (ConvNet-U) and multi-stage features learning (ConvNet-F-MS) and their combination (ConvNet-U-MS) compared to the fully-supervised system without multi-stage features (ConvNet-F).**

vidual contributions of unsupervised learning (ConvNet-U) and multi-stage features learning (ConvNet-F-MS) and their combination (ConvNet-U-MS) compared to the fully-supervised system without multi-stage features (ConvNet-F). With 17.1% error rate, unsupervised learning exhibits the most improvements compared to the baseline ConvNet-F (23.39%). Multi-stage features without unsupervised learning reach 17.29% error while their combination yields the competitive error rate of 10.55%.

Extensive results comparison of all major pedestrian datasets and published systems is provided in Table 2. Multiple types of measures proposed by [10] are reported. For clarity, we also plot in Figure 5 two of these measures, 'reasonable' and 'large', for INRIA-trained systems. The 'large' plot shows that the ConvNet results in state-of-the-art performance with some margin on the ETH, Caltech and TudBrussels datasets and is closely behind LatSvm-V2 and VeryFast for INRIA and Daimler datasets. In the 'reasonable' plot, the ConvNet yields competitive results for INRIA, Daimler and ETH datasets but performs poorly on the Caltech dataset. We suspect the ConvNet with multi-stage features trained at high-resolution is more sensitive to resolution loss than other methods. In future work, a ConvNet trained at multiple resolution will likely learn to use appropriate cues for each resolution regime.

4. Discussion

We have introduced a new feature learning model with an application to pedestrian detection. Contrary to popular models where the low-level features are hand-designed, our model learns all the features at all levels in a hierarchy. We used the method of [20] as a baseline, and extended it by combining high and low resolution features in the model, and by learning features on the color channels of the input. Using the INRIA dataset, we have shown that these improvements provide clear performance benefits. The resulting model provides state of the art or competitive results on most measures of all publicly available datasets. Small-scale pedestrian measures can be improved in future work by training multiple scale models relying less on high-resolution details. While computational speed was not the focus and hence was not reported here, our model was successfully used with near real-time speed in a haptic belt system [22] using parallel hardware. In future work, models designed for speed combined to highly optimized parallel computing on graphics cards is expected to yield competitive computational performance.

References

[1] M. Aharon, M. Elad, and A. M. Bruckstein. K-SVD and its non-negative variant for dictionary design. In M. Papadakis, A. F. Laine, and M. A. Unser, editors, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 5914, pages 327–339, Aug. 2005. 2

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Img. Sci.*, 2(1):183–202, 2009. 2, 3

[3] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Computer*

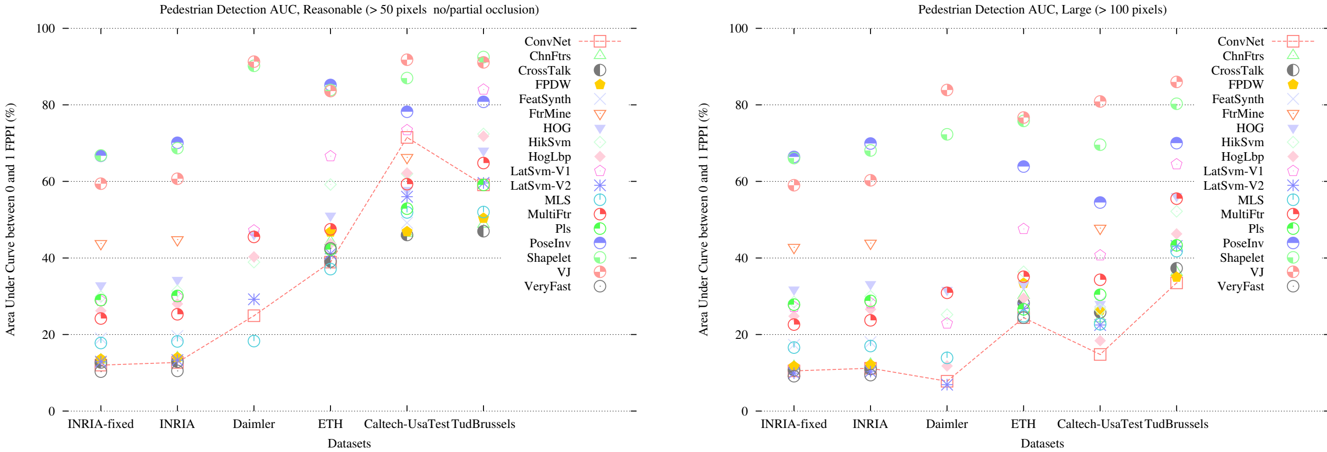

Figure 5: **Reasonable and Large measures for all INRIA-trained systems on all major datasets, using the proposed continuous AUC percentage.** The AUC is computed from DET curves (smaller AUC means more accuracy and less false positives). For a clearer overall performance, each ConvNet point is connected by dotted lines. While only the 'reasonable' and 'large' measures are plotted here, all measures are reported in table 2. The ConvNet system yields state-of-the-art or competitive results on most datasets and measures, except for the low resolutions measures on the Caltech dataset because of higher reliance on high-resolution cues than other methods.

Trained on	INRIA																	
	ConvNet	**ChnFtrs**	**CrossTalk**	**FPDW**	**FeatSynth**	**FtrMine**	**HOG**	**HikSvm**	**HogLbp**	**LatSvm-V1**	**LatSvm-V2**	**MLS**	**MultiFtr**	**Pls**	**PoseInv**	**Shapelet**	**VJ**	**VeryFast**
All - AUC %																		
INRIA-fixed	12.0	13.3	12.7	13.6	19.0	43.8	32.9	29.9	26.2	28.8	12.9	17.8	24.2	29.0	66.7	66.8	59.4	**10.3**
INRIA	12.7	13.9	12.9	14.0	19.6	44.8	34.3	31.4	28.0	29.8	13.3	18.2	25.3	30.1	70.1	68.7	60.7	**10.5**
Daimler	58.6	-	-	-	-	-	67.9	62.4	69.8	64.2	62.3	**51.8**	68.8	-	-	94.9	94.8	-
ETH	47.1	48.7	43.8	51.5	-	-	54.9	61.6	51.1	69.1	49.3	**42.8**	51.7	47.4	86.5	85.6	84.5	46.9
Caltech-UsaTest	90.9	**77.1**	77.8	78.1	78.1	86.7	85.5	86.8	87.9	91.7	84.2	83.4	83.4	81.2	92.6	95.4	99.1	-
TudBrussels	66.8	57.6	**55.0**	59.0	-	-	73.6	76.4	77.2	85.7	67.2	59.2	70.5	66.1	83.8	93.8	92.7	-
Reasonable - AUC % - >50 pixels & no/partial occlusion																		
INRIA-fixed	12.0	13.3	12.7	13.6	19.0	43.8	32.9	29.9	26.2	28.8	12.9	17.8	24.2	29.0	66.7	66.8	59.4	**10.3**
INRIA	12.7	13.9	12.9	14.0	19.6	44.8	34.3	31.4	28.0	29.8	13.3	18.2	25.3	30.1	70.1	68.7	60.7	**10.5**
Daimler	24.9	-	-	-	-	-	46.2	38.9	40.3	47.2	29.2	**18.3**	45.5	-	-	90.2	91.3	-
ETH	38.9	44.2	39.1	46.8	-	-	51.1	59.2	43.7	66.6	41.1	**37.1**	47.5	42.2	85.2	83.9	83.6	42.5
Caltech-UsaTest	71.5	46.4	**46.0**	46.9	49.2	66.3	57.8	62.0	62.2	73.4	56.0	51.9	59.3	52.9	78.2	87.0	91.8	-
TudBrussels	59.1	48.8	**47.0**	50.4	-	-	68.1	72.4	71.8	84.0	59.6	52.0	64.8	59.1	80.8	92.5	91.1	-
Large - AUC % - >100 pixels																		
INRIA-fixed	10.5	11.6	10.7	11.7	17.3	42.8	31.8	28.5	24.8	27.2	9.9	16.6	22.6	27.8	66.4	66.1	59.0	**9.1**
INRIA	11.2	12.2	11.0	12.1	18.0	43.9	33.2	30.0	26.6	28.2	10.3	17.0	23.7	28.8	69.9	68.1	60.3	**9.4**
Daimler	7.8	-	-	-	-	-	31.7	25.2	11.8	22.9	**6.9**	13.9	30.9	-	-	72.3	83.9	-
ETH	**24.4**	30.2	28.2	33.4	-	-	33.1	36.4	29.5	47.6	26.8	24.8	35.1	26.6	63.9	75.8	76.7	24.4
Caltech-UsaTest	**14.8**	24.1	25.8	26.4	28.6	47.8	28.0	26.5	18.4	40.7	22.5	22.7	34.3	30.4	54.5	69.6	80.9	-
TudBrussels	**33.5**	36.2	37.3	35.0	-	-	56.2	52.2	46.3	64.5	43.1	41.8	55.5	43.3	70.0	80.3	86.0	-
Near - AUC % - >80 pixels																		
INRIA-fixed	11.3	11.6	11.0	11.9	17.3	42.6	31.5	28.5	24.7	27.5	11.1	16.5	22.7	27.7	66.0	66.1	58.7	**9.7**
INRIA	11.9	12.2	11.2	12.3	17.9	43.7	32.9	30.0	26.5	28.5	11.5	16.8	23.8	28.8	69.4	68.1	60.0	**9.9**
Daimler	**10.0**	-	-	-	-	-	36.8	30.4	10.9	27.6	10.8	14.7	33.7	-	-	78.3	86.3	-
ETH	**28.9**	35.2	30.9	37.5	-	-	40.5	45.6	31.7	52.2	31.4	29.5	39.4	34.1	80.6	79.9	80.0	29.8
Caltech-UsaTest	27.3	27.4	28.9	28.4	29.5	48.9	33.1	34.3	**24.7**	47.2	26.7	29.1	40.8	31.2	66.8	75.7	85.3	-
TudBrussels	40.4	39.5	40.3	**38.8**	-	-	61.1	58.7	50.5	70.9	47.1	45.3	57.2	49.6	80.0	85.6	89.0	-
Medium - AUC % - 30-80 pixels																		
INRIA-fixed	33.1	100.0	99.7	100.0	100.0	100.0	100.0	100.0	85.3	85.3	99.7	100.0	86.1	100.0	99.7	99.7	91.5	**27.9**
INRIA	33.1	100.0	99.7	100.0	100.0	100.0	100.0	100.0	85.3	85.3	99.7	100.0	86.1	100.0	99.7	99.7	91.5	**27.9**
Daimler	54.2	-	-	-	-	-	62.1	54.4	70.7	58.5	60.0	**44.7**	63.2	-	-	95.2	93.7	-
ETH	55.4	42.9	**42.1**	45.4	-	-	49.9	54.7	61.2	71.5	57.3	43.9	47.3	45.0	73.9	74.5	71.2	48.3
Caltech-UsaTest	92.2	**69.5**	70.6	70.6	70.2	82.1	81.4	82.6	91.5	91.1	80.8	80.6	77.8	75.8	88.8	94.7	98.7	-
TudBrussels	67.8	57.4	**55.5**	59.7	-	-	71.4	74.9	82.9	85.5	68.2	59.1	68.7	65.0	79.4	94.1	91.7	-

Table 2: **Performance of all systems on all datasets using the proposed continuous AUC percentage** over the range [0,1] from DET curves. The top performing results (among INRIA-trained models) are highlighted in bold for each row. DET curves plot false positives per image (FPPI) against miss rate. Hence a smaller AUC% means a more accurate system with lower amount of false positives. The ConvNet model (ConvNet-U-MS here) holds several state-of-the-art or competitive scores. We report the multiple measures introduced by [10] for all major pedestrian datasets. For readibility, not all measures are reported nor are models not trained on INRIA. All results however are reported in the supplementary paper.

Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, pages 2903–2910. IEEE, 2012. 1

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances*

in Neural Information Processing Systems 19, pages 153–160. MIT Press, 2007. 1, 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In C. Schmid, S. Soatto, and C. Tomasi, editors, *CVPR'05*, volume 2, pages 886–893, June 2005. 1, 2, 5

[6] I. Daubechies, M. Defrise, and C. De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on Pure and Applied Mathematics*, 57(11):1413–1457, 2004. 2, 3

[7] P. Dollár, R. Appel, and W. Kienzle. Crosstalk cascades for frame-rate pedestrian detection. 1

[8] P. Dollár, S. Belongie, and P. Perona. The fastest pedestrian detector in the west. In *BMVC 2010, Aberystwyth, UK.* 1, 2

[9] P. Dollár, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *BMVC 2009, London, England.* 1, 2

[10] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: A benchmark. In *CVPR'09*. IEEE, June 2009. 4, 6, 7

[11] P. Dollár, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: An evaluation of the state of the art. *PAMI*, 99, 2011. 5

[12] J. Fan, W. Xu, Y. Wu, and Y. Gong. Human tracking using convolutional neural networks. *Neural Networks, IEEE Transactions on*, 21(10):1610 –1623, 2010. 4

[13] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. In *PAMI 2010.* 1

[14] A. Frome, G. Cheung, A. Abdulkader, M. Zennaro, B. Wu, A. Bissacco, H. Adam, H. Neven, and L. Vincent. Large-scale privacy protection in street-level imagery. In *ICCV'09*. 1, 2

[15] C. Garcia and M. Delakis. Convolutional face finder: A neural architecture for fast and robust face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2004. 1, 2

[16] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006. 1, 2

[17] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun. What is the best multi-stage architecture for object recognition? In *ICCV'09*. IEEE, 2009. 1, 4

[18] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. LeCun. Learning invariant features through topographic filter maps. In *CVPR'09*. IEEE, 2009. 2

[19] K. Kavukcuoglu, M. Ranzato, and Y. LeCun. Fast inference in sparse coding algorithms with applications to object recognition. Technical report, CBLL, Courant Institute, NYU, 2008. CBLL-TR-2008-12-01. 2

[20] K. Kavukcuoglu, P. Sermanet, Y. Boureau, K. Gregor, M. Mathieu, and Y. LeCun. Learning convolutional feature hierachies for visual recognition. In *Advances in Neural Information Processing Systems (NIPS 2010)*, 2010. 1, 2, 3, 6

[21] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS 2012: Neural Information Processing Systems*. 1

[22] Q. Le, M. Quigley, J. Feng, J. Chen, Y. Zou, W, M. Rasi, T. Low, and A. Ng. Haptic belt with pedestrian detection. In *NIPS, 2011 (Demonstrations)*. 6

[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, November 1998. 1, 2, 4

[24] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 801–808. MIT Press, Cambridge, MA, 2007. 2

[25] H. Lee, R. Grosse, R. Ranganath, and A. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *ICML'09*, pages 609–616. ACM, 2009. 2

[26] Y. Li and S. Osher. Coordinate Descent Optimization for l1 Minimization with Application to Compressed Sensing; a Greedy Algorithm. *CAM Report*, pages 09–17. 2

[27] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman. Discriminative learned dictionaries for local image analysis. *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8, June 2008. 2

[28] S. Maji, A. C. Berg, and J. Malik. Classification using intersection kernel support vector machines is efficient. volume 0, pages 1–8, Los Alamitos, CA, USA, 2008. IEEE Computer Society. 1

[29] S. Nowlan and J. Platt. A convolutional neural network hand tracker. pages 901–908, San Mateo, CA, 1995. Morgan Kaufmann. 1, 2

[30] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: a strategy employed by v1? *Vision Research*, 37(23):3311–3325, 1997. 2

[31] M. Osadchy, Y. LeCun, and M. Miller. Synergistic face detection and pose estimation with energy-based models. *Journal of Machine Learning Research*, 8:1197–1215, May 2007. 1, 2

[32] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In *NIPS'07*. MIT Press, 2007. 1, 2

[33] W. R. Schwartz, A. Kembhavi, D. Harwood, and L. S. Davis. Human detection using partial least squares analysis. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 24 –31, 29 2009-oct. 2 2009. 1

[34] P. Sermanet, S. Chintala, and Y. LeCun. Convolutional neural networks applied to house numbers digit classification. In *Proceedings of International Conference on Pattern Recognition*, 2012. 5

[35] P. Sermanet and Y. LeCun. Traffic sign recognition with multi-scale convolutional networks. In *Proceedings of International Joint Conference on Neural Networks*, 2011. 5

[36] G. Taylor, R. Fergus, G. Williams, I. Spiro, and C. Bregler. Pose-sensitive embedding by nonlinear nca regression. In *Advances in Neural Information Processing Systems NIPS 23*, 2010. 1, 2

[37] R. Vaillant, C. Monrocq, and Y. LeCun. Original approach for the localisation of objects in images. *IEE Proc on Vision, Image, and Signal Processing*, 141(4):245–250, August 1994. 1, 2

[38] S. Walk, N. Majer, K. Schindler, and B. Schiele. New features and insights for pedestrian detection. In *CVPR 2010, San Francisco, California.* 1, 2

[39] M. Zeiler, D. Krishnan, G. Taylor, and R. Fergus. Deconvolutional Networks. In *CVPR'10*. IEEE, 2010. 2