Hidden-layer Ensemble Fusion of MLP Neural Networks for Pedestrian Detection

Kyaw Kyaw Htike
School of Information Technology, UCSI University, Kuala Lumpur, Malaysia
E-mail: ali.kyaw@gmail.com

Keywords: pedestrian detection, neural networks, fusion, ensembles, multi-layer perceptron

Received: January 6, 2017

Being able to detect pedestrians is a crucial task for intelligent agents especially for autonomous vehicles, robots navigating in cities, machine vision, automatic traffic control in smart cities, and public safety and security. Various sophisticated pedestrian detection systems have been presented in literature and most of the state-of-the-art systems have two main components: feature extraction and classification. Over the past decade, the majority of the attention has been paid to feature extraction. In this paper, we show that much can be gained by having a high-performing classification algorithm, and changing only the classification component of the detection pipeline while fixing the feature extraction mechanism constant, we show reduction in pedestrian detection error (in terms of log-average miss rate) by over 40%. To be specific, we propose a novel algorithm for generating a compact and efficient ensemble of Multi-layer Perceptron neural networks that is well-suited for pedestrian detection both in terms of detection accuracy and speed. We demonstrate the efficacy of our proposed method by comparing with several state-of-the-art pedestrian detection algorithms.

Povzetek: Razvit je nov algoritem z nevronskimi mrežami za zaznavanje pešcev v prometu npr. za avtonomno vozilo.

1 Introduction

Pedestrian detection is an important problem in Artificial Intelligence and computer vision. Many pedestrian detection systems have been proposed with different feature extraction and classification methods. One of the earliest general machine learning based pedestrian detection systems was proposed by Papageorgiou and Poggio [1]. They use Haar wavelets coefficients measuring (at multiple scales) differences in intensity levels of pixels as the feature representation, and a Support Vector Machine (SVM) with the quadratic kernel as the classifier.

A year later, this motivated a seminal work on object detection, namely the Viola-Jones face detector [2]. Viola and Jones use the idea of *integral images* to speed up the extraction of rectangular Haar basis functions to use as features which then serve as input to an adaboost classifier. The classifier, which is applied to an image in a sliding window fashion, is specifically designed to be an attentional cascade so that most of non-face examples can be rejected with relatively few feature extraction steps. This results in a fast face detector. Although Haar basis functions are well-suited for detecting frontal-faces, it was not very clear whether it is also good for detecting other categories of objects. Indeed, one of the critical cues human beings use to classify or detect objects is shape information (for which the bulding blocks are image gradients or edges); the set of Haar basis functions does not exploit this. In fact, after [2] came out, features that make effective use of image gradient information would soon be proposed in [3].

Leibe *et al.* [4] propose an *Implicit Shape Model* which learns, for each cluster of local image patches, a vote distribution on the centroid of the object class. At test time, Generalized Hough Transform [5] is used to detect objects; to be specific, each local image patch votes with the learnt distributions for the centroid in the hough voting space and then local maxima (or peaks) in the voting space correspond to object centroids in the test image. For each of these local maxima, local image patches that contributed (*i.e.* voted for) are identified through a process known as *backprojection*. From this, a rough segmentation (and consequently, bounding boxes) of the objects can be obtained. Their method requires segmentation of the training images which can be labor-intensive and costly, and identifying the local maxima and performing backprojection can be highly sensitive to many types of noise. Moreover, due to the use of image patches, the system is not robust to illumination and various image geometric transformations.

In 2005, a groundbreaking work on object detection was published by Dalal and Triggs [3] who propose Histograms of Oriented Gradients (HOG) as features for pedestrian detection. In HOG, a histogram of image gradients is constructed in each small local region (termed as a *cell*) in an image and these histograms are concatenated to form a high-dimensional feature vector. They carried out a large number of experiments to explore and evaluate the design space of the HOG feature extraction process. This includes highlighting the importance of *local contrast normaliza-*

tion whereby each histogram corresponding to a cell is redundantly normalized with respect to other nearby cells. It was shown that with HOG features, a linear SVM was sufficient to obtain state-of-the-art results and outperformed several techniques such as generalized haar wavelets [1], PCA-SIFT [6] and Shape Contexts [7].

Moreover, HOG is still highly influential to this day; the majority of current state-of-the-art research is based on either variations of HOG or building upon ideas outlined in HOG [8, 9, 10, 11]. For instance, the well-known part-based object detection system, Deformable Part Models (DPM) [12], use HOG features as building blocks. Many systems have extended DPMs (*e.g.* [13, 14, 15]). Schwartz *et al.* [16] use Partial Least Squares to reduce high dimensional feature vectors that combine edge, texture and color cues, and Quadratic Discriminant Analysis as the classifier.

Dollar *et al.* [17] propose Integral Channel Features (ICF) that can be considered as a generalization of HOG by including not only gradient when computing local histograms, but also other "channels" of information such as sums of grayscale and color pixel values, sums of outputs obtained by convolving the image with linear filters (such as Difference of Gaussian filters) and sums of outputs of nonlinear transformation of the image (*e.g.* gradient magnitude). An attempt was made in [18] to speed up features such as ICF by approximating some of the scales of the feature pyramid by interpolating from corresponding nearby scales during multi-scale sliding window object detection.

Benenson *et al.* [19] propose an alternative to speed up object detection by training a separate model for each of the nearby scales of the image pyramid. Although this increases the object detection speed at test time, it also considerably lengthens the training time.

Benenson *et al.* [20] extend HOG [3] by automatically selecting the HOG cell sizes and locations using boosting. Their approach is very expensive to train. However, their work shows that the plain HOG is powerful and with the right sizes and placements of HOG cells, with a single rigid component, it is possible to outperform various feature extraction methods researchers have proposed over the years after the invention of HOG.

For these reasons, in this work, we adopt HOG [3] as the feature extraction method. Furthermore, we focus on the classification component of the object detection pipeline. In order to isolate the performance of the classifier, we set the feature extraction mechanism constant for all the experiments of our proposed method. For the classification component of the pedestrian detection pipeline, we propose a powerful and efficient non-linear classifier ensemble that significantly increases the pedestrian detection performance (*i.e.* accuracy) while at the same time reducing the computational complexity at test time which is especially important for a task such as pedestrian detection. Although the proposed algorithm could also be potentially applied to other object detection tasks and general machine learning classification problems, we focus on pedestrian detection in this paper.

To the best of our knowledge, neural networks, or more accurately, Multi-layer Perceptrons (MLPs) have not been used for pedestrian detection. This observation is also true for ensembles of MLPs (EoMLPs): there have not been any major work using EoMLPs for object detection, let alone pedestrian detection. Although there can be many reasons behind this dearth of application of EoMLPs in pedestrian detection, one possible reason could be due to the highly computationally expensive nature of EoMLPs at test time and due to the popularity of linear Support Vector Machines (SVMs) and other regularized linear classifiers. We show in this paper that given the same feature extraction mechanism, using our proposed non-linear classifier gives much better pedestrian detection performance than the traditional classifiers commonly used for pedestrian detection.

Works such as [21], which apply EoMLPs to real-world problems, exist (although quite rare to find); however, they are for general pattern tasks rather than pedestrian detection or even object detection. Literature on EoMLPs have been published in the machine learning and Artificial Intelligence community; a few well-known ones are [22, 23, 24, 25]. They demonstrate the effectiveness of EoMLPs compared to individual MLPs on some standard statistical benchmarks; none of them have been applied to any object detection or even computer vision problems. Furthermore, although there are many different ensemble methods of numerous types of classifiers such as bagging [26], boosting [27], Random subspace [28], Random Forest [29] and Rulefit [30], the focus of this paper is on EoMLPs and pedestrian detection. Moreover, our proposed algorithm differs from [22, 23, 24, 25] in numerous ways including being suitable to be applied for pedestrian detection problems and our method outperforms state-of-the-art EoMLPs as will shown in the experimental results in Section 4.

We term our novel algorithm as *Hidden-layer Ensemble Fusion of Multi-layer Perceptrons* (HLEF-MLP).

2 Contribution

The contribution that we make in this paper is five-fold:

1. We propose a novel way, for the purpose of pedestrian detection, of training multiple individual MLPs in an ensemble and fusing the members of the ensemble at the hidden-feature level rather than at the output score level as done in existing EoMLPs [22, 23, 24, 25]. This has the benefit of being able to correct the mistakes of the ensemble in a major way since the final classifier still has access to hidden level features rather than only output scores or classification labels.

2. We use L1-regularization in a stacking fashion to jointly and efficiently select individual hidden feature units of all the members in the ensemble which has the effect of fusing the members of the ensemble. This results in only a few active projection units at test time,

which can be implemented as fast matrix projections on modern hardware for efficient pedestrian (or object) detection.

3. In HLEF-MLP, the decisions of the members are combined or fused in a systematic way using the given supervision labels such that the combination is optimal for the classification task. This is in contrast to ad-hoc class-agnostic nature of most fusion schemes (which turn to use techniques such as averaging).

4. We show that given the same feature extraction mechanism, HLEF-MLP gives much better pedestrian detection performance than the state-of-the-art classifiers commonly used for pedestrian detection.

5. HLEF-MLP is stable and robust to initialization conditions and local optima of individual member MLPs in the ensemble. Therefore, we do not need to be so careful about training each MLP for two main reasons: firstly, we are not relying solely on one MLP, and secondly, we are able to, during fusion, correct mistakes of the individual MLPs at the hidden features level as mentioned previously. In fact, each MLP falling in its own local optima should be seen as achieving diversity in the ensemble which is a desirable goal that can be obtained for "free". At the L1-regularized fusion step, the optimization function is convex, therefore it is guaranteed to achieve the global optimum.

3 Method

3.1 Overview

Let $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)\}$ be the labelled training dataset, where N is the number of training data points, $\mathbf{x}_i \in \mathbb{R}^k$ is the k-dimensional feature vector corresponding to the i-th training data point, $y_i \in \{1, 0\}$ is the supervision label associated with $\mathbf{x}_i$.

HLEF-MLP consists of two stages. $\mathcal{D}$ is split into $\mathcal{D} = \{\mathcal{D}^1, \mathcal{D}^2\}$, where

$$\mathcal{D}^1 = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{\frac{N}{2}}, y_{\frac{N}{2}})\}$$

and

$$\mathcal{D}^2 = \{(\mathbf{x}_{\frac{N}{2}+1}, y_{\frac{N}{2}+1}), \ldots, (\mathbf{x}_N, y_N)\}$$

In the first stage which we call "ensemble learning", we train each individual MLP on $\mathcal{D}^1$ and in the second stage termed "sparse fusion optimization", we use $\mathcal{D}^2$ to automatically learn how to fuse the decisions of the members of the ensemble. We now describe each stage.

3.2 Ensemble learning

The inputs to the first stage are $\mathcal{D}^1$ (obtained as described in Section 3.1) and the number of members M in the ensemble. The ensemble can be written as $[f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})]$ where each member $f_j(\mathbf{x})$ of the ensemble can be formulated as:

$$f_j(\mathbf{x}) = \text{logsig}(\mathbf{w_j} \tanh(\mathbf{W}_j\mathbf{x} + \mathbf{b}_j) + b_j) \qquad (1)$$

where $\mathbf{x}$ is the input feature vector, and $\mathbf{W}$ and $\mathbf{b}$ are the matrix and vector respectively corresponding to an affine transformation of $\mathbf{x}$. Each column of $\mathbf{W}$ corresponds to the weights of the incoming connections to a neuron. Therefore the number of hidden neurons in $f_j(\mathbf{x})$ is equal of the number of columns in $\mathbf{W}_j$ and the number of rows of $\mathbf{W}_j$ is k (recall that $\mathbf{x} \in \mathbb{R}^k$). Therefore, $\mathbf{W} \in \mathbb{R}^{k \times h}$, where h is the number of neurons in the hidden layer.

The architecture of this first stage of the ensemble is illustrated in Figure 1. In the figure, each ensemble member (*i.e.* a MLP neural network) is shown as a vertical chain of blocks of mathematical operations for a total of M chains. The j-th ensemble (chain) corresponds to the function f_j as defined in Equation 1 and it can be seen that the block chain diagram follows exactly the sequence of operations defined in Equation 1.

For example, for the first ensemble member f_1, the input data vector $\mathbf{x}$ is first matrix-multiplied with $\mathbf{W}_1$ (first block), and the resulting vector is then added with the vector $\mathbf{b}_1$ in the second block. The function $\tanh(\cdot)$ is then applied (third block). This is followed by matrix multiplication of the output of the previous block with $\mathbf{w_1}$ in the fourth block. In the last block, a scalar addition with b_1 is performed.

The functions $\tanh(\cdot)$ and $\text{logsig}(\cdot)$ (visualized in Figure 2) are non-linear activation functions (applied after respective affine transformations) independently acting on each dimension of the vector obtained after the affine transformation and are defined as follows:

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \qquad (2)$$

$$\text{logsig}(a) = \frac{e^a}{1 + e^a} \qquad (3)$$

The *latent* parameters of the members of the first stage of the ensemble need to be *learnt* from the training data $\mathcal{D}^1$ (which is a set of pairs of input feature vectors and output supervision labels) and this training (*i.e.* learning) process is depicted in Figure 3.

We set all $\mathbf{W}_j$ to be the same size (*i.e.* each MLP $f_j(\mathbf{x})$ in the ensemble has the same number of hidden neurons). For each member MLP $f_j(\mathbf{x})$ in the ensemble, the following loss function can be constructed:

$$\begin{aligned} &L(\mathcal{D}^1, \mathbf{W}_j, \mathbf{b}_j, \mathbf{w}_j, b_j) = \\ &-\frac{2}{N}\sum_{i=1}^{\frac{N}{2}} y_i \,\text{logsig}(\mathbf{w_j} \tanh(\mathbf{W}_j\mathbf{x}_i + \mathbf{b}_j) + b_j) + \\ &(1 - y_i)\,\text{logsig}(\mathbf{w_j} \tanh(\mathbf{W}_j\mathbf{x}_i + \mathbf{b}_j) + b_j) \end{aligned} \qquad (4)$$

where $\{\mathbf{x}_i, y_i\}_{i=1}^{\frac{N}{2}}$ are pairs of input feature vectors and output supervision labels from $\mathcal{D}^1$.

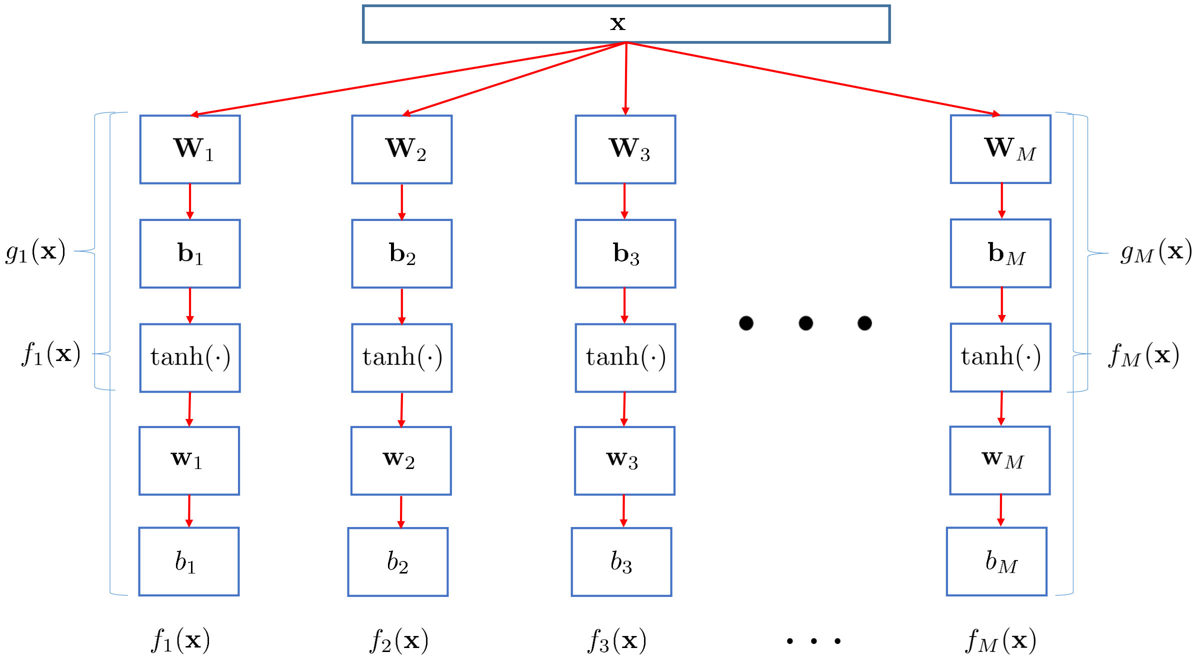

Figure 1: Architecture of the first stage of the ensemble.

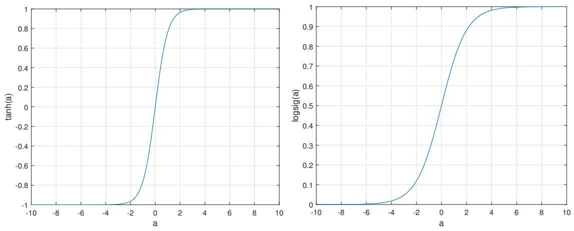

Figure 2: Non-linear activation functions; on the left is the curve for $\tanh(a)$ and on the right is $\text{logsig}(a)$, where a is the input signal to the activation function.

The loss function given in Equation 4 is a smooth and differential function and we optimize it using L-BFGS [31] since it is a type of efficient semi-newton method that does not require setting sensitive hyperparameters such as learning rate, momentum and mini-batch size. After the optimization, all the weights of each network:

$$\{\mathbf{W}_1, \mathbf{W}_2, \ldots, \mathbf{W}_M, \mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M, \mathbf{w}_1, \ldots, \\ \ldots, \mathbf{w}_M, b_1, b_2, \ldots, b_M\} \tag{5}$$

are obtained.

3.3 Sparse fusion optimization

In the second stage which is sparse fusion optimization, function $g_j(\mathbf{x})$ is used to project each data point $\mathbf{x}$ as given below:

$$g_j(\mathbf{x}) = \tanh(\mathbf{W}_j\mathbf{x} + \mathbf{b}_j) \tag{6}$$

Each data point $\mathbf{x} \in \mathcal{D}^2$ is projected using $\{g_j(\mathbf{x})\}_{j=1}^{M}$. That is, a new training dataset $\hat{\mathcal{D}}^2$ is constructed as follows:

$$\begin{bmatrix} g_1(\mathbf{x}_1) & g_2(\mathbf{x}_1) & \ldots & g_M(\mathbf{x}_1) \\ g_1(\mathbf{x}_2) & g_2(\mathbf{x}_2) & \ldots & g_M(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ g_1(\mathbf{x}_N) & g_2(\mathbf{x}_N) & \ldots & g_M(\mathbf{x}_N) \end{bmatrix} \tag{7}$$

where each row corresponds to one new data point in $\hat{\mathcal{D}}^2$. The architecture of this second stage of the ensemble is depicted in Figure 4.

It is important to note that the projection is done using the function $g(\cdot)$, and *not* $f(\cdot)$. In other words, this can be interpreted as projecting to the hidden layers of the individual MLP neural networks (which are members of the ensemble) and then learning to fuse in this hidden layer, hence the name of our proposed algorithm being *Hidden-layer Ensemble Fusion of Multi-layer Perceptrons* (HLEF-MLP).

This improves over the traditional (state-of-the-art) ensemble techniques in terms of both generating a much more compact ensemble (making it available to be applied to very time-critical applications such as pedestrian detection) and the final pedestrian detection performance (measured by log-average miss rate).

After constructing the projected dataset from $\hat{\mathcal{D}}^2$, a L1-regularized logistic regression is then trained on $\hat{\mathcal{D}}^2$ by optimizing:

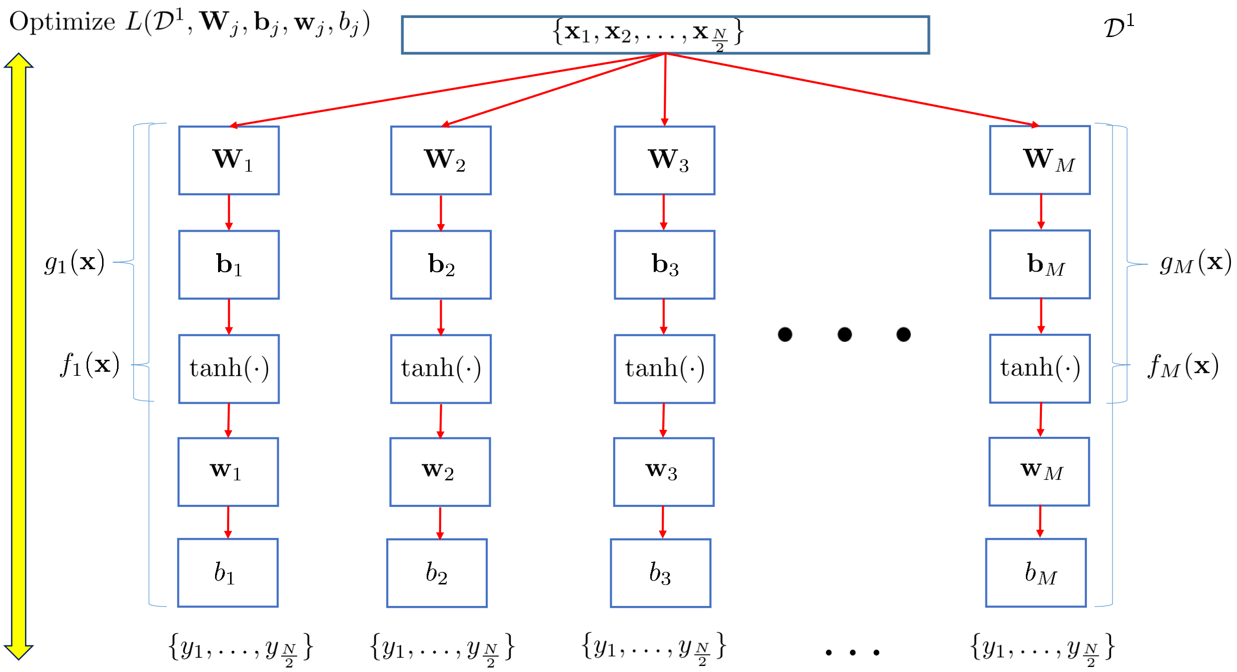

Figure 3: Independent ensemble member learning process.

$$\mathbf{m}_{\text{trained}} = \arg\min_{\mathbf{m}} \sum_{i=\frac{N}{2}+1}^{N} f_L(\mathbf{m}, [g_1(\mathbf{x}_i), g_2(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)], y_i) + \beta f_R(\mathbf{m}) \quad (8)$$

where $\mathbf{m}_{\text{trained}} \in \mathbb{R}^k$ is the trained classifier and is in fact a vector of weights. Moreover, f_L is the *loss function*, f_R is the *regularization* to encourage $\mathbf{m}$ to take small values (hence in a way, favoring simpler models), and β balances the regularization term and loss term. A higher value of β would result in a smoother solution (and less fit to the training data $\hat{\mathcal{D}}^2$) whereas lower β would result lower training error. The loss function is given by:

$$f_L(\mathbf{m}, [g_1(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)], y_i) = \log(1 + \exp(-y_i \mathbf{m}^T [g_1(\mathbf{x}_i), \ldots, g_M(\mathbf{x}_i)])) \quad (9)$$

In order to encourage sparse solutions, we use L1-regularization for which the regularization term is given by:

$$f_R = [1, \ldots, 1]^T \mathbf{m} \quad (10)$$

This effectively sets many of the components of the weight vector $\mathbf{m}$ to zero, resulting in a very compact ensemble (speeding up the pedestrian detection), while at the same time, retaining or even improving the ensemble performance for pedestrian detection. Figure 5 illustrates the learning process for the sparse fusion of the members in the ensemble at the hidden layer.

3.4 Prediction at test time

As illustrated in Figure 6, at test time, given a test feature vector $\mathbf{x}$, the prediction score s_{pred} is obtained by:

$$s_{\text{pred}} = \frac{1}{1 + \exp(-(\mathbf{m}_{\text{trained}})^T [g_1(\mathbf{x}), \ldots, g_M(\mathbf{x})])} \quad (11)$$

where $\{g_1, g_2, \ldots, g_M\}$ are the hidden-layer-projection functions (parts of the members of the ensemble) whose latent parameters have been obtained by the training process described in Section 3.2 and $\mathbf{m}_{\text{trained}}$ is the set of latent sparse weights that have been obtained as explained in Section 3.3.

Equation 11 can also be equivalently written as:

$$\begin{aligned} s_{\text{pred}} &= \frac{1}{1 + \exp(a)} \\ \text{where } a &= -(\mathbf{m}_{\text{trained}})^T [\tanh(\mathbf{W}_1 \mathbf{x} + \mathbf{b}_1), \ldots, \\ &\quad \tanh(\mathbf{W}_M \mathbf{x} + \mathbf{b}_M)] \end{aligned} \quad (12)$$

Since $\mathbf{m}_{\text{trained}}$ is a sparse vector (*i.e.* a vector where the majority of the elements are zeros), at test time, entire columns of the matrices $\{\mathbf{W}_1, \ldots, \mathbf{W}_M\}$ corresponding to the positions of the zero vector elements in $\mathbf{m}_{\text{trained}}$ can be omitted. This greatly speeds up the pedestrian detection process, while improving the performance (log-average miss rate), due to the sparse fusion technique at the hidden layers as described in Sections 3.2 and 3.3.

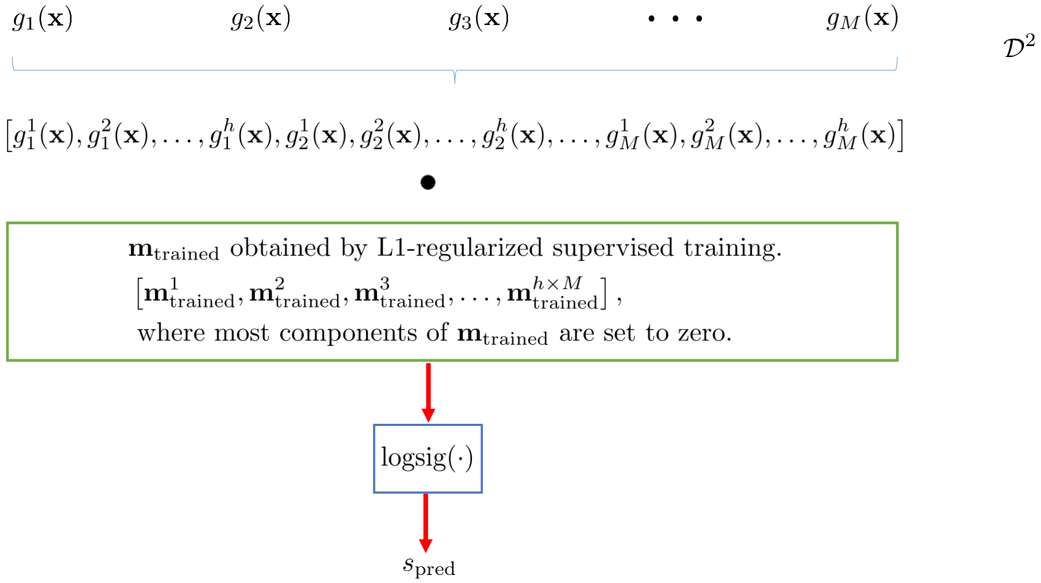
$g_1(\mathbf{x})$
$g_2(\mathbf{x})$
$g_3(\mathbf{x})$
$\cdots$
$g_M(\mathbf{x})$
$\mathcal{D}^2$
$[g_1^1(\mathbf{x}), g_1^2(\mathbf{x}), \ldots, g_1^h(\mathbf{x}), g_2^1(\mathbf{x}), g_2^2(\mathbf{x}), \ldots, g_2^h(\mathbf{x}), \ldots, g_M^1(\mathbf{x}), g_M^2(\mathbf{x}), \ldots, g_M^h(\mathbf{x})]$
$\mathbf{m}_{\text{trained}}$ obtained by L1-regularized supervised training.
$[\mathbf{m}_{\text{trained}}^1, \mathbf{m}_{\text{trained}}^2, \mathbf{m}_{\text{trained}}^3, \ldots, \mathbf{m}_{\text{trained}}^{h\times M}]$,
where most components of $\mathbf{m}_{\text{trained}}$ are set to zero.
logsig(·)
s_{pred}

Figure 4: Architecture of the second stage of the ensemble (fusion).

4 Results and discussion

4.1 Dataset

We use the INRIA Person dataset [3] for evaluating our algorithms. In the dataset, for training, there are 614 images containing pedestrians for which ground truth is available; after cropping and flipping each pedestrian, the total number of pedestrians come up to be 2474. There are also 1218 lage images that do not contain any pedestrians; from these, data corresponding to non-pedestrian class can be generated by a combination of initial sampling and hard negative mining as is common in object detection literature [32, 9].

4.2 Ensemble hyper-parameters

We set the size of the ensemble M (see Section 3.2) to 100 and the number of hidden neurons h in each hidden layer to 10 for all the experiments involving both our proposed method and baselines on various ensemble variations.

4.3 Experiment setup

We perform seven different experiments in order to compare our proposed algorithm with state-of-the-art pedestrian detection systems. The experiments are described below.

4.3.1 `VJ`

The Viola-Jones detector of [2] applied to pedestrian detection.

4.3.2 `HOG`

Pedestrian detector of the seminal work of Dalal and Triggs [3], using Histogram of Oriented Gradients (HOG) for feature extraction and linear SVM as the classifier.

4.3.3 `HikSvm`

The system proposed by Maji *et al.* [33] who use a variation of HOG (concatenation of histograms of oriented gradients for a few different cell sizes) and SVM with an approximation of the intersection kernel.

4.3.4 `Pls`

The pedestrian detection system proposed by Schwartz *et al.* [16] using Partial Least Squares (PLS) analysis to reduce the dimensions of very high dimensional features obtained by concatenating different sets of features obtained from edge, texture and color information derived from the original image. Then Quadratic Discriminant Analysis is used as the classifier.

4.3.5 `OursMLPEnsHidFuse`

Our proposed approach in this paper as described in Section 3. At test time, due to L1-regularized fusion at the hidden layer level, we can expect that the resulting ensemble will be very small, which will be suitable for time-critical applications such as pedestrian detection.

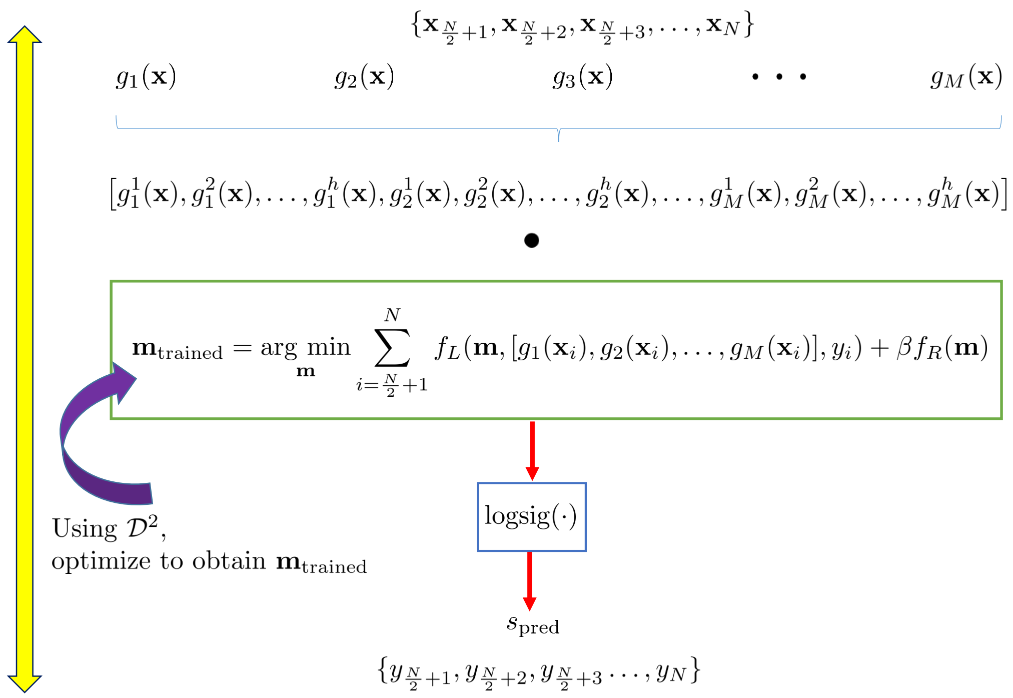
$\{\mathbf{x}_{\frac{N}{2}+1}, \mathbf{x}_{\frac{N}{2}+2}, \mathbf{x}_{\frac{N}{2}+3}, \dots, \mathbf{x}_N\}$
$g_1(\mathbf{x})$
$g_2(\mathbf{x})$
$g_3(\mathbf{x})$
$g_M(\mathbf{x})$
$[g_1^1(\mathbf{x}), g_1^2(\mathbf{x}), \dots, g_1^h(\mathbf{x}), g_2^1(\mathbf{x}), g_2^2(\mathbf{x}), \dots, g_2^h(\mathbf{x}), \dots, g_M^1(\mathbf{x}), g_M^2(\mathbf{x}), \dots, g_M^h(\mathbf{x})]$
$\mathbf{m}_{\text{trained}} = \arg\min_{\mathbf{m}} \sum_{i=\frac{N}{2}+1}^{N} f_L(\mathbf{m}, [g_1(\mathbf{x}_i), g_2(\mathbf{x}_i), \dots, g_M(\mathbf{x}_i)], y_i) + \beta f_R(\mathbf{m})$
Using $\mathcal{D}^2$,
optimize to obtain $\mathbf{m}_{\text{trained}}$
logsig(·)
s_{pred}
$\{y_{\frac{N}{2}+1}, y_{\frac{N}{2}+2}, y_{\frac{N}{2}+3} \dots, y_N\}$

Figure 5: Learning to fuse the members in the ensemble.

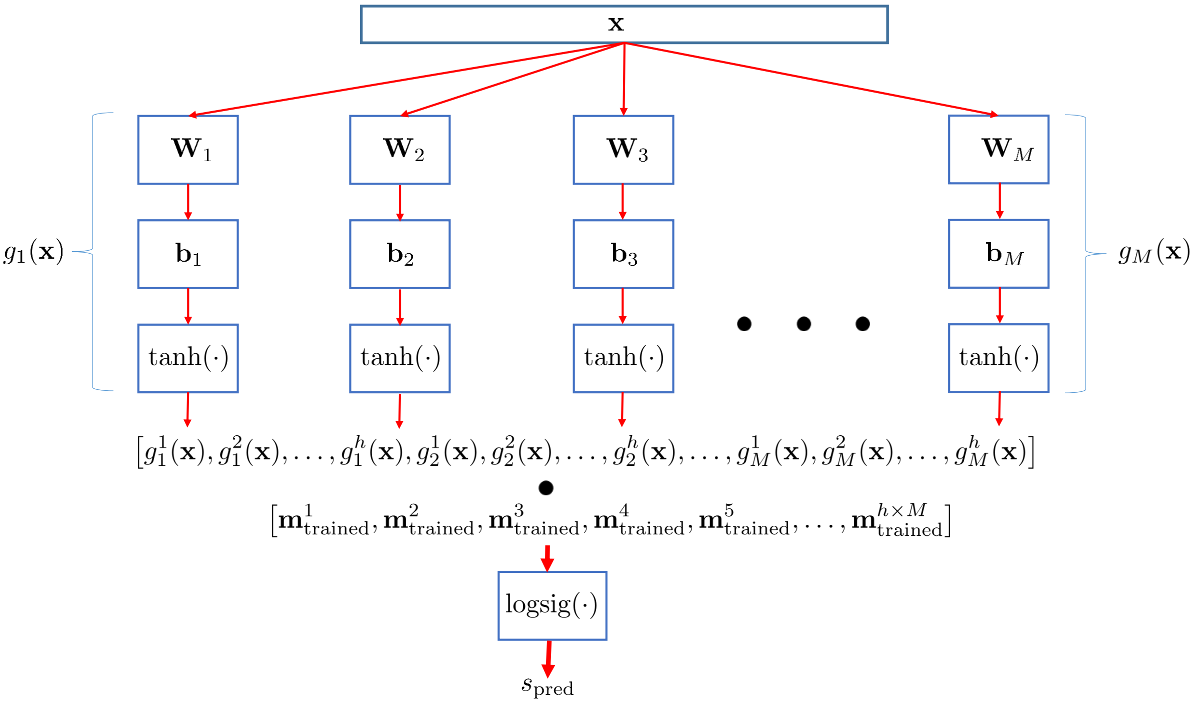
$\mathbf{x}$
$\mathbf{W}_1$
$\mathbf{W}_2$
$\mathbf{W}_3$
$\mathbf{W}_M$
$g_1(\mathbf{x})$
$\mathbf{b}_1$
$\mathbf{b}_2$
$\mathbf{b}_3$
$\mathbf{b}_M$
$g_M(\mathbf{x})$
tanh(·)
tanh(·)
tanh(·)
tanh(·)
$[g_1^1(\mathbf{x}), g_1^2(\mathbf{x}), \dots, g_1^h(\mathbf{x}), g_2^1(\mathbf{x}), g_2^2(\mathbf{x}), \dots, g_2^h(\mathbf{x}), \dots, g_M^1(\mathbf{x}), g_M^2(\mathbf{x}), \dots, g_M^h(\mathbf{x})]$
$[\mathbf{m}_{\text{trained}}^1, \mathbf{m}_{\text{trained}}^2, \mathbf{m}_{\text{trained}}^3, \mathbf{m}_{\text{trained}}^4, \mathbf{m}_{\text{trained}}^5, \dots, \mathbf{m}_{\text{trained}}^{h\times M}]$
logsig(·)
s_{pred}

Figure 6: Using the fused ensemble at test time.

4.3.6 OursMLPEnsScoreFuse

A variation of our proposed method OursMLPEnsHidFuse; instead of fusing at the level of hidden features, L1-regularized fusion takes place at the score level. This is also somewhat similar to classifier stacking [34] in literature; however most stacking ensembles in literature is using simple unregularized classifiers whereas OursMLPEnsScoreFuse is able to select the most efficient members of the ensemble jointly using supervised label information.

In terms of efficiency at test time, we can anticipate it to be worse than OursMLPEnsHidFuse because the supervised L1-regularized selection is being performed at the score level and also at test time, there is a need to multiple matrix projections (one for each member of the ensemble) and then combine them.

4.3.7 OursMLPEns

This is our implementation of the state-of-the-art MLP fusion by bagging. Here unlike OursMLPEnsHidFuse and OursMLPEns, there are no separation of the training dataset $\mathcal{D}$ into $\mathcal{D}^1$ and $\mathcal{D}^2$; independent training of the members of the ensemble takes place on the entire training data $\mathcal{D}$. Moreover, there is no sparse fusion optimization.

We can expect that compared to OursMLPEnsHidFuse and OursMLPEnsScoreFuse, this will have the worst efficiency at test time because the entire ensemble needs to be evaluated which is highly expensive.

4.4 Evaluation criteria

To evaluate the pedestrian detections from the aformentioned experiments, firstly there needs to be a measure on what a correct pedestrian detection is. Although this can be defined in many ways, we use the PASCAL 50% overlap criterion [35] which is the most widely-used one in state-of-the-art detection literature. Let a detected bounding box be $\mathbf{r}_d$ and the corresponding groundtruth bounding box be $\mathbf{r}_g$. In PASCAL 50% overlap criterion, in order to establish whether $\mathbf{r}_d$ is a correct detection, the overlap ratio, α_o, defined in Equation 13 is computed and then the detection $\mathbf{r}_d$ is deemed to be correct if $\alpha_o > 0.5$.

$$\alpha_o = \frac{\text{area}(\mathbf{r}_g \cap \mathbf{r}_d)}{\text{area}(\mathbf{r}_g \cup \mathbf{r}_d)} \tag{13}$$

The α_o can be understood as the ratio of the intersection of the detected bounding box with ground truth bounding box and the union of the detected box with ground truth box.

Graphs are an important tool to visualize the performance of pedestrian detectors since they show the performance at various levels of detection thresholds. To this end, we plot curves where miss rate is on the y-axis and false positives per image (FPPI) is on the x-axis.

This has been empirically proven in state-of-the-art pedestrian detection (*e.g.* by Benenson *et al.* [9]) to be a more accurate way of measuring the detection performance than the previously used false positives per window (FPPW).

In a miss rate versus FPPI plot, lower curves denote higher detection performance. Moreover, to summarize the performance of a detector with a single value, log-average miss rate is calculated which is approximately equal to the area under the miss rate versus FPPI curve.

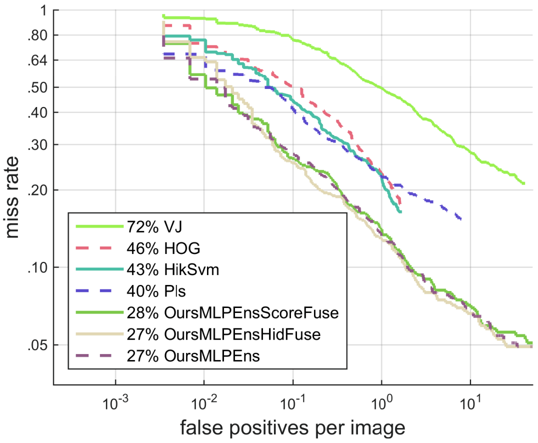

Figure 7: ROC curves for all experiments)

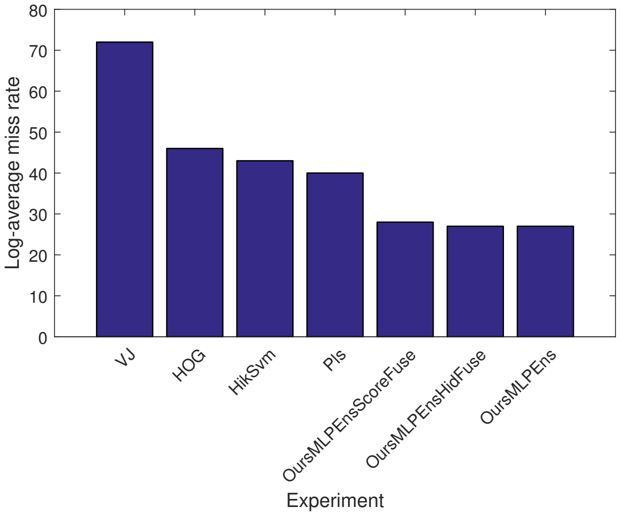

Figure 8: Comparison of log-average miss rate (lower is better).

4.5 Results

The miss rate versus FPPI plots for all the experiments are shown in Figure 7. In the figure, the curves are ordered in terms of log-average miss rate (LAMR) from worst to best. To better focus on the summarized performance, we also plot the LAMRs of each experiment in terms of a bar graph as illustrated in Figure 8. Finally, we depict in Figure 9 the comparison of average detection time taken by different algorithms for a 640 × 480 image. We also show in Table 1

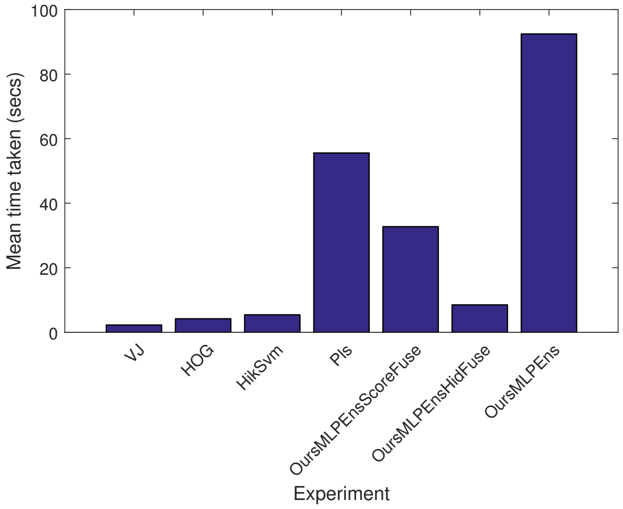

Figure 9: Comparison of average detection time for a 640 × 480 image.

Experiment	Mean time (secs)
VJ	2.23
HOG	4.18
HikSvm	5.41
Pls	55.56
OursMLPEnsScoreFuse	32.72
OursMLPEnsHidFuse	8.49
OursMLPEns	92.45

Table 1: Comparison in terms of detection time.

the raw detection time values for easier comparison.

From Figures 7, 8 and 9, the following observations can be made:

- VJ has the worst performance among all. This is to be expected since the Haar-features that it uses does not capture the object shape information well. Moreover, the seminal work HOG greatly improves over VJ for generic object detection.

- Although HikSvm requires complex modification in the feature extraction and the classifier compared to HOG, it only results in a modest detection performance gain (*i.e.* decrease in LAMR from 46% to 43%). Similar observation can be made for Pls although the improvement is relatively better.

- Our proposed algorithm OursMLPEnsHidFuse has the best performance among them, tied in the first place with OursMLPEns. Given that OursMLPEnsHidFuse is using the standard HOG features, we can see that just by using our proposed algorithm, the LAMR goes down from 46% and 27%. This is a significant improvement (corresponding to sover 40% reduction in LAMR). In comparison, HikSvm, despite proposing both new feature extraction and classification algorithms, only managed less than 0.1% reduction in LAMR.

- OursMLPEnsHidFuse has slightly higher detection performance than OursMLPEnsScoreFuse whereas in terms of speed at test time, OursMLPEnsHidFuse is almost 4 times faster. This shows the efficacy of our proposed algorithm OursMLPEnsHidFuse and also proves that fusion at the level of the hidden layers not only results in better detection performance and also very compact matrix projections (and hence much faster speed) due to the L1-regularized sparse fusion learning process having access to the hidden layer features.

- Despite OursMLPEnsHidFuse and OursMLPEns being tied at the first place in terms of detection performance, OursMLPEnsHidFuse is significantly (over 10 times) faster than OursMLPEns. This shows that our proposed OursMLPEnsHidFuse has the same detection accuracy as the very expensive state-of-the-art MLP bagging which is not practical for detection purposes. In other words, the novel method that we have presented in this paper OursMLPEnsHidFuse gives the same (top) performance as a MLP bagging-ensemble but with 10 times faster speed for pedestrian detection.

- OursMLPEnsScoreFuse is faster than OursMLPEns but lower in LAMR than OursMLPEnsHidFuse. This again shows that our proposed OursMLPEnsHidFuse not only results in a much smaller ensemble model (and consequently, faster detection speed), but is also more effective in bringing out the effectiveness of the ensemble compared to OursMLPEnsScoreFuse.

- OursMLPEnsHidFuse has better detection accuracy than PLS while being significantly faster using only standard HOG features. It shows that we have not saturated the performance of HOG features yet. There is still a lot of improvement that can made in the classification algorithm for pedestrian detection systems.

- Our experiments show for the first time that using the proposed algorithm OursMLPEnsHidFuse, it is possible to apply ensemble techniques for efficient object detection purposes.

4.5.1 Analysis of trained ensemble model complexity

Although the results and the discussion about detection performance and speeds in the previous section should be sufficient, for completeness, we theorize and analyze the model complexities and sizes for the three ensemble methods described in the previous section: OursMLPEns, OursMLPEnsScoreFuse and OursMLPEnsHidFuse. It is to be noted that as mentioned previously, there are $M = 100$ MLPs in the ensemble.

For OursMLPEns, given a test feature vector $\mathbf{x}$, there are 100 different matrix projections with each matrix having k rows and h columns (which is equal to the number of dimensions of the feature vector and the number of hidden neurons in each MLP respectively). After each projection, tansig$(\cdot)$ function must be applied, followed by a dot product between the resulting vector and a linear weight vector, which will produce a single score (for each member of the ensemble). Then the logsig$(\cdot)$ function needs to be performed on this score. Then 100 such scores will be averaged to give the final ensemble score. Therefore the ensemble complexity is quite high, especially for a highly computationally expensive task such as sliding window pedestrian detection.

For OursMLPEnsScoreFuse, after training the L1-regularized linear classifier on scores at the end of the model fusion step, 33 networks are discovered to be nonzero. This means that theoretically, it should be $\frac{100}{33} \approx 3$ times faster than OursMLPEns. In fact, this theoretical observation tallies with the experimental results presented in Table 1.

With regards to OursMLPEnsHidFuse, it was found that performing L1-regularized classifier optimization on hidden features resulted in highly sparse weight vector $\mathbf{m}_{\text{trained}}$ with the number of nonzero weight components equal to a mere 159. Therefore, at test time, there is only a need to perform a *single* matrix projection with a matrix having k rows and 159 columns. After that, $tansig$ function has to be applied followed by a dot product with a linear weight vector and finally a logsig$(\cdot)$ function. This means that the effective model complexity is much lower than either OursMLPEns or OursMLPEnsScoreFuse. This again matches with the empirical evidence.

5 Conclusion and future work

In this paper, we propose a novel algorithm to train a compact ensemble of Multi-layer Perceptron neural networks for pedestrian detection. The proposed algorithm integrates the members of the ensemble at the hidden feature layer level, resulting in a very small ensemble size (allowing for fast pedestrian detection) while at the same time outperforming existing neural network ensembling techniques and other pedestrian detection systems. We obtain very encouraging state-of-the-art results, and show for the first time that, using our proposed algorithm, it is indeed possible to apply neural network ensemble techniques for the task for pedestrian detection, something that was previously thought to be too inefficient due to the very large model size of ensembles.

There are several interesting directions of research based upon the work in this paper; firstly, there is an opportunity to apply the method presented in this paper to other object detection tasks and applications such as face and vehicle detection, and other general pattern recognition problems. Secondly, it will be highly beneficial to explore the effect of different feature extraction mechanisms on the resulting ensembles in terms of both detection performance and efficiency.

References

[1] Constantine Papageorgiou and Tomaso Poggio. A trainable system for object detection. *International Journal of Computer Vision*, 38(1):15–33, 2000.

[2] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1–511. IEEE, 2001.

[3] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, pages 886–893. IEEE, 2005.

[4] Bastian Leibe, Ales Leonardis, and Bernt Schiele. Combined object categorization and segmentation with an Implicit Shape Model. In *ECCV Workshop on Statistical Learning in Computer Vision*, volume 2, pages 7–14, 2004.

[5] Dana H Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2):111–122, 1981.

[6] Krystian Mikolajczyk and Cordelia Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[7] Serge Belongie, Jitendra Malik, and Jan Puzicha. Matching shapes. In *International Conference on Computer Vision*, volume 1, pages 454–461. IEEE, 2001.

[8] Kyaw Kyaw Htike and David Hogg. Adapting pedestrian detectors to new domains: a comprehensive review. *Engineering Applications of Artificial Intelligence*, 50:142–158, April 2016.

[9] Rodrigo Benenson, Mohamed Omran, Jan Hosang, and Bernt Schiele. *Ten Years of Pedestrian Detection, What Have We Learned?*, pages 613–627. Springer International Publishing, Cham, 2015.

[10] Kyaw Kyaw Htike and David Hogg. *Unsupervised Detector Adaptation by Joint Dataset Feature Learning*, pages 270–277. Springer International Publishing, Cham, 2014.

[11] Kyaw Kyaw Htike and David Hogg. Weakly supervised pedestrian detector training by unsupervised prior learning and cue fusion in videos. In *International Conference on Image Processing*, pages 2338–2342. IEEE, October 2014.

[12] Pedro Felzenszwalb, David McAllester, and Deva Ramanan. A discriminatively trained, multiscale, deformable part model. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, June 2008.

[13] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010.

[14] Pedro F Felzenszwalb, Ross B Girshick, and David McAllester. Cascade object detection with deformable part models. In *Conference on Computer Vision and Pattern Recognition*, pages 2241–2248. IEEE, 2010.

[15] Ross B Girshick, Pedro F Felzenszwalb, and David A Mcallester. Object detection with grammar models. In *Advances in Neural Information Processing Systems*, pages 442–450, 2011.

[16] William Robson Schwartz, Aniruddha Kembhavi, David Harwood, and Larry S Davis. Human detection using Partial Least Squares analysis. In *International Conference on Computer Vision*, pages 24–31. IEEE, 2009.

[17] Piotr Dollar, Zhuowen Tu, Pietro Perona, and Serge Belongie. Integral channel features. In *British Machine Vision Conference*, pages 91.1–91.11. BMVA Press, 2009.

[18] Piotr Dollár, Serge Belongie, and Pietro Perona. The fastest pedestrian detector in the West. In *British Machine Vision Conference*, volume 2, page 7, 2010.

[19] Rodrigo Benenson, Markus Mathias, Radu Timofte, and Luc Van Gool. Pedestrian detection at 100 frames per second. In *Conference on Computer Vision and Pattern Recognition*, pages 2903–2910. IEEE, 2012.

[20] Rodrigo Benenson, Markus Mathias, Tinne Tuytelaars, and Luc Gool. Seeking the strongest rigid detector. In *Conference on Computer Vision and Pattern Recognition*, pages 3666–3673. IEEE, 2013.

[21] E Filippi, M Costa, and E Pasero. Multi-layer perceptron ensembles for increased performance and fault-tolerance in pattern recognition tasks. In *International Conference on Neural Networks: IEEE World Congress on Computational Intelligence*, volume 5, pages 2901–2906. IEEE, 1994.

[22] Pablo M Granitto, Pablo F Verdes, and H Alejandro Ceccatto. Neural network ensembles: evaluation of aggregation algorithms. *Artificial Intelligence*, 163(2):139–162, 2005.

[23] Lars Kai Hansen and Peter Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (10):993–1001, 1990.

[24] Anders Krogh, Jesper Vedelsby, et al. Neural network ensembles, cross validation, and active learning. *Advances in Neural Information Processing Systems*, 7:231–238, 1995.

[25] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1):239–263, 2002.

[26] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996.

[27] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of Boosting. *Annals of Statistics*, 28(2):337–407, 1998.

[28] Tin Kam Ho. The random subspace method for constructing decision forests. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(8):832–844, 1998.

[29] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[30] Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, pages 916–954, 2008.

[31] Dong C. Liu, Jorge Nocedal, and Dong C. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528, 1989.

[32] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012.

[33] Subhransu Maji, Alexander C Berg, and Jitendra Malik. Classification using intersection kernel support vector machines is efficient. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[34] Saso Džeroski and Bernard Ženko. Is combining classifiers with stacking better than selecting the best one? *Machine learning*, 54(3):255–273, 2004.

[35] Mark Everingham, Luc J. Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.