University of Wollongong

Research Online

Faculty of Informatics - Papers (Archive)

Faculty of Engineering and Information Sciences

1-1-2009

Reduced training of convolutional neural networks for pedestrian detection

Giang Hoang Nguyen
University of Wollongong, giang_nguyen@uow.edu.au

Son Lam Phung
University of Wollongong, phung@uow.edu.au

Abdesselam Bouzerdoum
University of Wollongong, bouzer@uow.edu.au

Follow this and additional works at: https://ro.uow.edu.au/infopapers

 Part of the Physical Sciences and Mathematics Commons

Recommended Citation

Hoang Nguyen, Giang; Phung, Son Lam; and Bouzerdoum, Abdesselam: Reduced training of convolutional neural networks for pedestrian detection 2009, 61-66.
https://ro.uow.edu.au/infopapers/793

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Reduced training of convolutional neural networks for pedestrian detection

Abstract

Pedestrian detection is a vision task with many practical applications in video surveillance, road safety, autonomous driving and military. However, it is much more difficult compared to the detection of other visual objects, because of the tremendous variations in the inner region as well as the outer shape of the pedestrian pattern. In this paper, we propose a pedestrian detection approach that uses convolutional neural network (CNN) to differentiate pedestrian and non-pedestrian patterns. Among several advantages, the CNN integrates feature extraction and classification into one single, fully adaptive structure. It can extract two-dimensional features at increasing scales, and it is relatively tolerant to geometric, local distortions in the image. Although the CNN has good generalization performance, training CNN classifier is time-comsuming. Therefore, we present an efficient training approach for CNN. Through the experiments, we show that it is possible to design networks in a fraction of time taken by the standard learning approach.

Keywords

reduced, convolutional, training, neural, detection, networks, pedestrian

Disciplines

Physical Sciences and Mathematics

Publication Details

Hoang Nguyen, G., Phung, S. & Bouzerdoum, A. (2009). Reduced training of convolutional neural networks for pedestrian detection. In H. Huynh, D. Tien & G. Shi (Eds.), The 6th International Conference on Information Technology and Applications (ICITA 2009) (pp. 61-66). Hanoi, Vietnam: ICITA.

This conference paper is available at Research Online: https://ro.uow.edu.au/infopapers/793

Reduced Training of Convolutional Neural Networks for Pedestrian Detection

Giang Hoang Nguyen, Son Lam Phung *Member, IEEE*, and Abdesselam Bouzerdoum *Senior Member, IEEE*

***Abstract*— Pedestrian detection is a vision task with many practical applications in video surveillance, road safety, autonomous driving and military. However, it is much more difficult compared to the detection of other visual objects, because of the tremendous variations in the inner region as well as the outer shape of the pedestrian pattern. In this paper, we propose a pedestrian detection approach that uses convolutional neural network (CNN) to differentiate pedestrian and non-pedestrian patterns. Among several advantages, the CNN integrates feature extraction and classification into one single, fully adaptive structure. It can extract two-dimensional features at increasing scales, and it is relatively tolerant to geometric, local distortions in the image. Although the CNN has good generalization performance, training CNN classifier is time-comsuming. Therefore, we present an efficient training approach for CNN. Through the experiments, we show that it is possible to design networks in a fraction of time taken by the standard learning approach.**

Index Terms: pedestrian detection, convolutional neural networks, reduced training.

I. Introduction

Pedestrian detection aims to determine the presence and the location of people or pedestrians in images or video. It is a vision task that has important applications in video surveillance [1], road safety, autonomous driving [2], and many other areas [3]. Locating pedestrians in images or video is a challenging task because it is not easy to find robust visual characteristics that differentiate a pedestrian from all other objects. The body region of a pedestrian exhibits strong variations, e.g. simply by some change in clothing. The boundary of a pedestrian can change drastically depending on the walking, standing or running pose of the person. Furthermore, many practical applications of pedestrian detection are in outdoor environments where the lighting conditions vary greatly.

In this paper, we present a new approach that uses convolutional neural networks to locate pedestrians in images. Convolutional neural networks (CNNs) were proposed by LeCun *et al.* for the classification of two-dimensional (2-D) image patterns [4]. It is based on three architectural ideas: local receptive fields, weight sharing, and sub-sampling in the spatial domain. Convolutional neural networks have many strengths: (i) feature extraction and classification are integrated into one structure and fully adaptive; (ii) the network extract 2-D image features at increasing dyadic scales; (iii) it is relatively tolerant to geometric, local distortions in the image. CNNs have been used for in a number of applications including hand-written digit recognition[4], face detection [5], and face recognition [6].

Although CNNs are known to have good generalization capability, a main difficulty with using CNNs is that the memory requirement and training time increase rapidly with the number of 2-D training samples. In this paper, we introduce an efficient approach for training CNN for pedestrian detection. Our approach combines both unsupervised and supervised learning. First, a clustering techniques is applied to partition the training patterns into a smaller number of clusters. Second, a modified supervised learning algorithm is applied to train a CNN, using the weighted cluster centroids. This approach can be generalized to train neural networks or pattern classifiers for object detection tasks.

This paper is organized as follows. Section II presents a review of existing work on pedestrian detection. Section III describes the CNN architecture and its mathematical model. Section IV describes the proposed techniques for training the CNN pedestrian classifier. Section V presents experimental results and analysis, and Section VI gives the concluding remarks.

II. Background

There are two major strategies to detecting pedestrians. The early strategy locates people using heuristic visual cues such as motion [7], [8], background scene [9] and color [10], [11]. Pantil *et al.* found moving objects by calculating the difference between consecutive video frames [12]. Other authors constructed statistical models about color, intensity, or spatial/temporal variation of background pixels, and compared these models with a new video frame to segment foreground objects [3], [10]. These pre-processing steps are typically followed by techniques such as face detection [12] silhouette shape [13], Hausdorff-based shape comparison [14], or edge-based template matching [15]. These approaches rely on video sequences and have limited use when only a single image is available.

An alternative strategy for pedestrian detection is to scan rectangular regions or windows of the input image, and determine if each region resembles the human body, using a trained pattern classifier. This strategy is computation-intensive because windows at different image scales must be processed. Nevertheless, it can cope well with image variations and has been widely used in objection detection. Papageorgiou and Poggio introduced a pedestrian detection

Authors are with the School of Electrical, Computer and Telecommunication Engineering, University of Wollongong, Wollongong, NSW 2522, Australia. E-mails: {hgn94,phung,bouzer}@uow.edu.au).

© ICITA 2009 ISBN: 978-981-08-3029-8

method that extracts Haar wavelet features from each 128-by-64 window and uses support vector machines to classify the features [2]. Subsequently, Viola and Jones developed a fast object detection method that relies on a cascade of classifiers [16]. Each classifier uses one or more Haar-like features and is trained using an adaptive boosting algorithm (AdaBoost). Their method has been applied successfully to the face detection problem.

Viola and Jones method have been extended by several authors. Chen *et al.* proposed extra stages to improve classification performance of the AdaBoost [17]. In their method, a linear SVM is inserted after every two AdaBoost stages, and it accepts the classification scores of the previous two stages as inputs. At the same false positive rate (of 10^{-4} or below), Chen *et al.* method has only half the miss rate of the standard AdaBoost, evaluated on the INRIA pedestrian data set. Chen *et al.* also reported that combining intensity-based and gradient-based features improves detection performance. Cui *et al.* [18] proposed 3-D Haar features to describe both spatial and temporal differences between object regions, and the SVM to classify these features. Phung and Bouzerdoum introduced the edge density feature to differentiate between the pedestrian and non-pedestrian patterns [19].

Munder and Gavrila performed an experimental study on pedestrian classification on the DaimlerChrysler data set of $4,000$ pedestrian images and over $25,000$ non-pedestrian images [20]. They analyzed three main aspects in designing a classifier for pedestrian detection: (i) feature extraction methods such as Principal Component Analysis (PCA), Haar features and local receptive fields (LRFs); (ii) classifier architectures such as SVM, feed-forward neural network and k-nearest neighbor classifier; and (iii) incremental learning such as bootstrapping and adaptive boosting. They found that adaptive, trainable filters (local receptive fields) are more accurate than global features (PCA) and simple features (Haar). The SVM outperforms other tested classification architectures. Interestingly, Munder and Gavrila found that, among the tested classifiers, even the best structure has performance (classification rate of 89.75% for non-bootstrap classifier) that is still far from what is required for real-world applications.

In order to build a more accurate classifier, Alonso *et al.* proposed using features that are selected from a more diverse set that includes Canny edge image, Haar wavelets, gradient magnitude/orientation, co-occurrence matrix, histogram of intensity differences, histogram of normalized gradients, and number of texture units [21]. Chen *et al.* proposed using genetic algorithm to select shape and motion features for pedestrian detection [22]. Tivive and Bouzerdoum proposed a pedestrian classifier based on neural networks and the shunting inhibition mechanism [23].

III. CNN Network Architecture

Our approach to pedestrian detection involves scanning the input image at different scales. At each scale, all windows of a fixed size is processed by a classifier to determine whether or not each input window is a pedestrian pattern. The classifier is based on the convolutional neural network. This section will discuss the network architecture.

A CNN consists of three main types of layers: (i) convolution layers, (ii) sub-sampling layers, and (iii) an output layer. Network layers are arranged in a feed-forward structure: each convolution layer is followed by a sub-sampling layer, and the last convolution layer is followed by the output layer (see Fig. 1). The convolution and sub-sampling layers are 2-D layers, whereas the output layer is considered as a 1-D layer. In CNN, each 2-D layer is made up of several planes, a plane is a 2-D array of neurons. The output of a plane is called a feature map.

- In a *convolutional layer*, each plane is connected to one or more feature maps of the preceding layer. A connection is associated with a convolution mask, which is a 2-D matrix of adjustable entries called *weights*. Each plane first computes the convolution between its 2-D inputs and its convolution masks. The convolution outputs are summed together and then added to an adjustable scalar, known as a *bias* term. Finally, an activation function is applied on the result to obtain the plane output. The plane output is a 2-D matrix called a *feature map*. This name arises from the fact that the convolution output indicates the presence of a visual feature at a given pixel location. A convolution layer produces one or more feature maps. Each feature map is then connected to exactly one plane in the next sub-sampling layer.
- A *sub-sampling* layer has the same number of planes as the preceding convolution layer. A sub-sampling plane divides its 2-D input into non-overlapping blocks of size 2×2 pixels. For each block, the sum of four pixels is calculated; this sum is multiplied by an adjustable weight before being added to a bias term. The result is passed through an activation function to produce an output for the 2×2 block. Clearly, each sub-sampling plane reduces its input size by half along each dimension. A feature map in a sub-sampling layer is connected to one or more planes in the next convolution layer.
- In the *last convolution layer*, each plane is connected to exactly one preceding feature map. This layer uses convolution masks that have the same size as the input feature maps. Therefore, each plane in the last convolution layer will produce one scalar output. The outputs from all planes in this layer are then connected to the output layer.
- The *output* layer, in general, can be constructed from sigmoidal neurons or radial-basis-function (RBF) neurons. Here, we use sigmoidal neurons for the output layer. The outputs of this layer are considered as the network outputs. In a pattern classification application, these outputs indicate the category of the input image.

Let l denote the index of a network layer. The layer index l goes from 1 to L, where L is the number of network layers. Here, we assume that $L = 2a + 2$

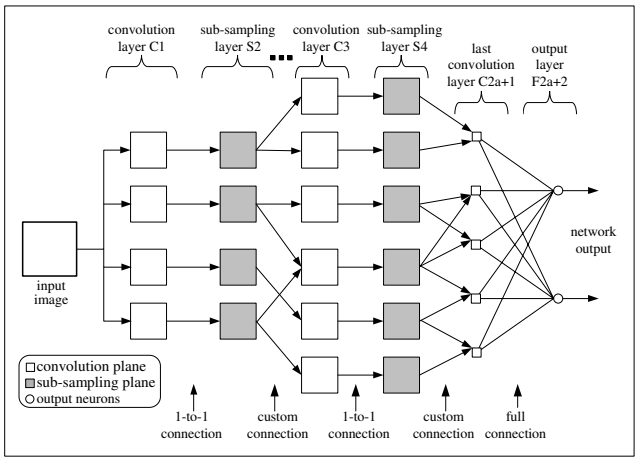

Fig. 1. Layers in a convolutional neural network.

where a is a positive integer. Let N_l be the number of feature maps in layer l, and $f_l(.)$ be the activation function of layer l. Let $\mathbf{y}_n^l$ be the n-th feature map or output of layer l.

Convolution layer

Consider a convolution layer l. In this structure, l is an odd integer, $l = 1, 3, ..., 2a + 1$. Let $r_l \times c_l$ denote the size of convolution mask for layer l. For feature map n in layer C_l, let

- $\mathbf{w}_{m,n}^l = \{w_{m,n}^l(i,j)\}$ be the convolution mask from feature map m in layer $\{l-1\}$ to feature map n in layer l,
- b_n^l be the bias term associated with feature map n,
- $\mathbf{V}_n^l$ denote the list of all planes in layer $\{l-1\}$ that are connected to feature map n. For example, $\mathbf{V}_4^l = \{2, 3, 5\}$ means that feature maps 2, 3 and 5 of layer $\{l-1\}$ are connected to feature map 4 of layer l.

Feature map n of convolution layer l is calculated as

$$\mathbf{y}_n^l = f_l(\sum_{m \in \mathbf{V}_n^l} \mathbf{y}_m^{l-1} \otimes \mathbf{w}_{m,n}^l + b_n^l) \quad (1)$$

where $\otimes$ denotes the 2-D convolution operator. Suppose that the size of input feature maps $\mathbf{y}_m^{l-1}$ is $H_{l-1} \times W_{l-1}$ pixels, and the size of convolution masks $\mathbf{w}_{m,n}^l$ is $r_l \times c_l$ pixels. Because we perform convolution without zero-padding the inputs, the size of output feature map $\mathbf{y}_n^l$ is $(H_{l-1} - r_l + 1) \times (W_{l-1} - c_l + 1)$ pixels.

Sub-sampling layer

Now, we consider a sub-sampling layer l, where l is an even integer, $l = 2, 4, ..., 2a$. For feature map n, let w_n^l be the weight and b_n^l be the bias term. We divide feature map n of convolution layer $\{l-1\}$ into non-overlapping blocks of size 2×2 pixels. Let $\mathbf{z}_n^{l-1}$ be a matrix obtained by summing the four pixels in each block. That is,

$$\begin{aligned} z_n^{l-1}(i,j) &= y_n^{l-1}(2i-1, 2j-1) + y_n^{l-1}(2i-1, 2j) \\ &\quad + y_n^{l-1}(2i, 2j-1) + y_n^{l-1}(2i, 2j) \end{aligned} \quad (2)$$

Feature map n of sub-sampling layer l is now calculated as

$$\mathbf{y}_n^l = f_l(\mathbf{z}_n^{l-1} \times w_n^l + b_n^l) \quad (3)$$

A feature map $\mathbf{y}_n^l$ in sub-sampling layer l will have a size of $H_l \times W_l$, where $H_l = H_{l-1}/2$ and $W_l = W_{l-1}/2$.

Output layer

In this study, we consider output layer L that consists of sigmoidal neurons. Let N_L be the number of output sigmoidal neurons. Let $w_{m,n}^L$ denote the weight from feature map m of the last convolutional layer, to neuron n of the output layer. Let b_n^L be the bias term associated with neuron n of layer L. The output of sigmoidal neuron n is calculated as

$$y_n^L = f^L(\sum_{m=1}^{N_{L-1}} y_m^{L-1} w_{m,n}^L + b_n^L) \quad (4)$$

The outputs of all sigmoidal neurons form the network outputs.

IV. New approach for training CNN classifier

Efficient network training is a major challenge in using the CNN for image classification. Our new approach to CNN training combines three techniques: (i) batch training with RPROP, (ii) partial evaluation of error gradient, (iii) reduced training via clustering.

A. Batch training with RPROP

Most existing CNN applications use stochastic online training, where the network parameters are updated after each training sample is presented. While this technique reduces memory requirement, it leads to slow training and causes significant oscillation in the overall training error: By the time the last training examples are presented, the network tends to *forget* the earlier training examples. In contrast, batch training updates network parameters only after accumulating the contributions from all training examples. Compared to stochastic online training, this technique is more stable.

Given a fixed set of M input samples $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_M\}$, the CNN is trained to produce target outputs $d_1, d_2, ..., d_M$, where d_i is set to $+0.9$ if $\mathbf{x}_i$ is a pedestrian; otherwise d_i is set to -0.9. Training is done iteratively to reduce the mean-square-error function

$$E(\mathbf{w}) = \frac{1}{M} \sum_{i=1}^{M} (y_i - d_i)^2, \quad (5)$$

where $\mathbf{w}$ represents all weights and biases of the network. A technique to compute the error gradient $g(\mathbf{w}) = \nabla E(\mathbf{w})$ is described in [24].

Once the error gradient is computed, various training algorithms can be used. For the CNN, we use the resilient back-propagation algorithm (RPROP [25]). RPROP is one of the fastest first-order training algorithms; it works even when the gradient becomes very small [19]. RPROP is suitable for CNN training because it requires much less memory compared to other methods such as conjugate gradient or Levenberg-Marquardt.

The RPROP method can be described briefly as follows. Each weight w_i is associated with an adaptive step $\Delta_i(t)$, and is updated as

$$w_i(t+1) = w_i(t) + \Delta_i(t) \;\; \text{sign}\{g(w_i,t)\}. \tag{6}$$

where $g(w_i,t)$ denotes the partial derivative for weight w_i at time t. The adaptive step is increased if the gradient keeps its sign, and reduced if the gradient changes sign:

$$\Delta_i(t) = \begin{cases} \eta_{\text{inc}}\ \Delta_i(t-1), & \text{if} \;\; g(w_i,t)\ g(w_i,t-1) > 0 \\ \eta_{\text{dec}}\ \Delta_i(t-1), & \text{if} \;\; g(w_i,t)\ g(w_i,t-1) < 0 \\ \Delta_i(t-1), & \text{otherwise} \end{cases}. \tag{7}$$

where $\eta_{\text{inc}} > 1$ and $\eta_{\text{dec}} < 1$ are two scalar factors.

B. Partial evaluation of error gradient

Even with the RPROP algorithm, memory requirement for batch computation of the CNN error gradient $g(\mathbf{w})$ becomes prohibitive for large training sets. For example, in our experiments, memory is not sufficient on a PC with 3GB RAM for batch training with 5000 input images, each having a size of 56×20 pixels.

To tackle this memory problem, we propose another technique. For a large training set $\mathbf{X}$, we partition it into subsets $\mathbf{X} = \mathbf{X}_1 \bigcup \mathbf{X}_2 \bigcup ... \bigcup \mathbf{X}_N$, and compute the error gradient for each separate subset: $g_1(\mathbf{w})$, $g_2(\mathbf{w})$, ..., $g_N(\mathbf{w})$. The overall error gradient is then computed as

$$g(\mathbf{w}) = \sum_{n=1}^{N} g_m(\mathbf{w}) \tag{8}$$

With this technique, we are able to handle training sets of more than $20,000$ images on 3GB RAM. When disk storage is used, it can deal with an even larger training set.

C. Reduced training via clustering

We propose another efficient technique for training the CNN, in which a pre-processing step is introduced to reduce the number of training samples. To this end, unsupervised clustering is applied to the original data set $\mathbf{X}$ to extract cluster centers $\{\mathbf{c}_k\}$ that yield a compact representation of the original data. Here, clustering is performed on only training samples belonging to the same class. It is assumed that each cluster contain samples from a single class, but a class may have several clusters.

Through clustering, the data set $\mathbf{X}$ is reduced to K clusters, where K is much smaller than M. Each cluster is represented by a cluster centroid $\mathbf{c}_k$ and cluster size z_k; z_k is simply the number of training samples in the cluster. After clustering, most existing techniques use only cluster centroids and discard information about cluster sizes. The novelty in our technique is that we take into account the cluster sizes to compensate for the information lost during clustering. To this end, a modified error function is defined as

$$E_p(\mathbf{w}) = \sum_{k=1}^{K} (\widehat{y}_k - \widehat{d}_k)^2 p(\mathbf{c}_k), \tag{9}$$

where $\widehat{y}_k$ is the network output, $\widehat{d}_k$ is the target output, and $p(\mathbf{c}_k)$ is the weight for cluster centroid $\mathbf{c}_k$.

By changing the way the cluster weight $p(\mathbf{c}_k)$ is defined, different training algorithms can be created to handle large or unbalanced data sets [26]. In this work, we define the cluster weight as follows

$$p(\mathbf{c}_k) = \frac{z_k}{n_k}, \tag{10}$$

where n_k is the number of original training samples that belong to the same class as $\mathbf{c}_k$. To implement this technique, we modify the computation of the error gradient to take into account the weights $p(\mathbf{c}_k)$.

V. Experiments and Analysis

In this section, we analyze the proposed approach for training the CNN pedestrian classifier. We also discuss the design of a pedestrian versus non-pedestrian classifier, and compare the standard CNN training and the proposed reduced training technique for CNN.

A. Data preparation

To support this study, we have collected 5600 images and manually prepared the ground-truth of pedestrian coordinates in these images. These images were taken by us in indoor and outdoor environments, of different people; the pedestrian poses vary to include walking, running and standing. We also collected $10,000$ photos with no pedestrian, from which non-pedestrian patterns can be extracted. In this paper, we used the first 4500 images for designing and evaluating individual CNN classifiers. We reserved the remaining 1100 images for future testing of the full pedestrian detector. The data sets used in our experiments are summarized in Table I.

TABLE I
PEDESTRIAN DATA SET.

	Training set	Validation set	Test set
Pedestrian	5760	1920	1920
Non-pedestrian	5760	1920	1920
Total	11520	3840	3840

B. Selecting network input size

The CNN has a total of six layers and a receptive field size of 5×5 pixels; it uses linear function for the output layer. We investigated 60 network structures for the CNN classifier.

- Network input size (5 choices): 60×20, 56×20, 52×24, 40×20 and 36×20.
- Activation function (2 choices): 'linear for convolution layer, sigmoidal for sub-sampling layer', or 'sigmoidal for convolution layer, linear for sub-sampling layer'.
- Number of feature maps in layers $C1$ (2 choices): 3 or 4.
- Connection between layer $S2$ and layer $C3$ (3 choices): '1-to-2', '1-to-2,1-to-1', '1-to-2,2-to-1'. In a '1-to-2' scheme, each feature map is connected to two feature maps in the next layer. In a '2-to-1' scheme, two feature

maps are connected to one feature map in the next layer. It is also possible to combine different connection schemes.

For each network structure, the CNN was trained, using techniques presented in IV-A and IV-B, for 2000 epoches and 200 networks are created. Each network structure took about 48 hours of training on one PC. After training, the 200 networks were evaluated on the validation set, and the best network was selected. The CR of the selected network on the test set is used to indicate the performance of the network structure. The experiments were run on a cluster of 45 computers.

For each of the five input sizes, 12 different network structures were evaluated. The mean, maximum and standard deviation of the classification rates on the test set are shown in Table II. For each input size, CR_{mean} and CR_{max} is the average and the maximum CR respectively, computed across all network structures having the same input size. We found that the input size of 56×20 pixels has the highest CR_{mean} of 93.7%, the second highest CR_{max} of 95.1%, the second lowest standard deviation for CR ($\sigma = 0.9\%$). This result indicates that input size of 56×20 is suitable for pedestrian versus non-pedestrian classification.

TABLE II
CLASSIFICATION RATES (%) ON THE TEST SET FOR DIFFERENT INPUT SIZES.

Input size	60×20	56×20	52×24	40×20	36×20
CR_{mean}	93.3	93.7	92.3	92.2	93.0
CR_{max}	94.6	95.1	94.7	94.1	95.2
STD of CR	0.8	0.9	2.4	1.3	1.4

Comparison results also indicate that the following network structure performs better among the 60 structures:

- Input size is 56×20 pixels.
- The activation function is sigmoidal for convolution layers, and linear for sub-sampling layers.
- Layer $C1$ has three feature maps.
- The connection between layer $S2$ and $C3$ is '1-to-2, 1-to-1'.

C. Comparison of standard versus reduced training

We compare the following three training approaches.

- *Random-CNN*: Training samples were selected randomly from the original training set. The size of the reduced set is 3456 samples.
- *RT-CNN*: Representative training examples were found via clustering and the proposed training algorithm was applied, as described in Section IV-C. The size of the reduced set is 3456 samples. Clustering is done through the k-means algorithm because it requires little parameter tuning and is quite effective in handling large data sets. [27].
- *All-CNN*: The entire original training set is used to train the network. The size of the original training set is 11520 samples.

Each approach is applied to train a network of the same initial weights for 4000 epochs. The networks produced at different numbers of epochs were run on the validation set to select the best network, which was then evaluated on the test set. The role of the validation set is to prevent over-training.

1) Learning speed: Figure 2 shows the classification rate on the original training set versus the training time, for the RT-CNN and All-CNN approach. The RT-CNN reached a training error rate of 0.3% in 360 minutes, whereas the All-CNN took 550 minutes to reach to the same training error rate. Furthermore, to reach a classification rate of 91.6% on the test set, the RT-CNN took 13.5 hours, whereas the standard one took nearly 20 hours. The RT-CNN took 29 hours to learn the entire training set, while the All-CNN needed 43 hours. Note that, for the RT-CNN there is a one-off cost of finding clusters, which depends on the clustering algorithm and the number of clusters. In the pedestrian detection task, it took only eight minutes to form 3456 clusters from a data set of $11,520$ samples in 56×20 dimensional space.

These results indicate that the RT-CNN is more efficient than the All-CNN in terms of training time.

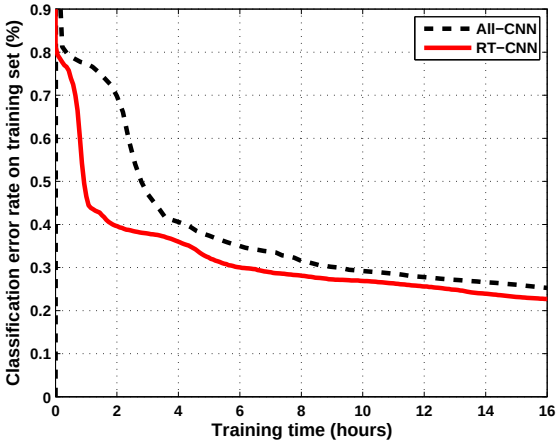

Fig. 2. Classification rates on the training set versus the training time for RT-CNN and All-CNN techniques.

2) Generalization capability: Table III shows the performance of Random-CNN, RT-CNN and All-CNN approaches. The classification rates of Random-CNN, RT-CNN, and All-CNN are 92.53%, 93.62%, and 95.1%, respectively. Using unsupervised clustering to select training samples (RT-CNN) achieves a higher classification rate, compared to selecting training samples randomly (Random-CNN).

Compared to All-CNN technique, the reduced training technique (RT-CNN) has a 1.6% lower CR, but it uses only 30% of the training samples and takes 33% less training time.

The receiver operating characteristics (ROC) curves for the RT-CNN and All-CNN are shown in Fig. 3. The results show that with the proposed training technique (RT-CNN), it is possible to lower training time significantly while maintaining the classification rate.

TABLE III

COMPARISON OF THREE TRAINING TECHNIQUES ON TEST SET.

	CR at $\theta_{valid}(\%)$	CR$_{max}(\%)$	Training time (h)
Random-CNN	92.0	92.5	29
RT-CNN	93.5	93.6	29
All-CNN	95.1	95.2	43

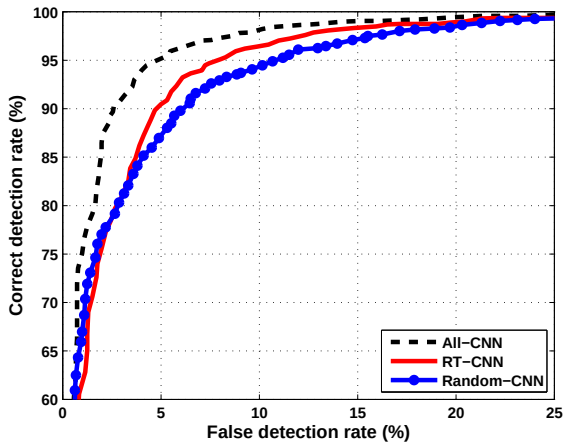

Fig. 3. ROC on test set for All-CNN, RT-CNN, and Random-CNN training techniques.

VI. CONCLUSIONS

This paper applies convolutional neural network to detect pedestrians in images. Our work is motivated by the fact that convolutional neural networks have many strengths in image classification tasks but their training is time-consuming. In this paper, we present three techniques for efficient training of CNN on large data sets. We show that it is possible to train a CNN classifier in less time while maintaining good generalization performance. Our reduced training approach uses unsupervised clustering to find representative training examples, and then applies weighted supervised learning that incorporates cluster sizes into the error function.

REFERENCES

[1] R. Collins, A. Lipton, H. Fujiyoshi, and T. Kanade, "Algorithms for cooperative multisensor surveillance," *Proceedings of the IEEE*, vol. 89, no. 10, pp. 1456–1477, 2001.

[2] C. Papageorgiou and T. Poggio, "Trainable pedestrian detection," in *International Conference on Image Processing*, vol. 4, 1999, pp. 35–39.

[3] I. Haritaoglu and M. Flickner, "Attentive billboards," in *International Conference on Image Analysis and Processing*, 2001, pp. 162–167.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] 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*, vol. 26, no. 11, pp. 1408–1423, 2004.

[6] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back, "Face recognition: a convolutional neural-network approach," *IEEE Transactions on Neural Networks*, vol. 8, no. 1, pp. 98 –113, 1997.

[7] A. Branca, M. Leo, G. Attolico, and A. Distante, "People detection in dynamic images," in *International Joint Conference on Neural Networks*, vol. 3, 2002, pp. 2428–2432.

[8] M.-T. Yang, Y.-C. Shih, and S.-C. Wang, "People tracking by integrating multiple features," in *International Conference on Pattern Recognition*, vol. 4, 2004, pp. 929–932.

[9] N. M. Oliver, B. Rosario, and A. P. Pentland, "A bayesian computer vision system for modeling human interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 831–843, 2000.

[10] Y. Rachlin, J. Dolan, and P. Khosla, "Learning to detect partially labeled people," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2003, pp. 1536–1541.

[11] S. Harasse, L. Bonnaud, and M. Desvignes, "Human model for people detection in dynamic scenes," in *International Conference on Pattern Recognition*, vol. 1, 2006, pp. 335–354.

[12] R. Patil, P. Rybski, T. Kanade, and M. Veloso, "People detection and tracking in high resolution panoramic video mosaic," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, 2004, pp. 1323–1328.

[13] I. Haritaoglu, D. Harwood, and L. Davis, "W4: real-time surveillance of people and their activities," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 809–830, 2000.

[14] S. M. Yoon and H. Kim, "Real-time multiple people detection using skin color, motion and appearance information," in *13th IEEE International Workshop on Robot and Human Interactive Communication*, 2004, pp. 331–334.

[15] Z. Zhang and K. Kodagoda, "Multi-sensor approach for people detection," in *International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2005, pp. 355–360.

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

[17] Y. T. Chen and C. S. Chen, "Fast human detection using a novel boosted cascading structure with meta stages," *IEEE Transactions on Image Processing*, vol. 17, no. 8, pp. 1452–1464, 2008.

[18] X. Cui, Y. Liu, S. Shan, X. Chen, and W. Gao, "3d haar-like features for pedestrian detection," in *IEEE International Conference on Multimedia and Expo*, 2007, pp. 1263–1266.

[19] S. L. Phung and A. Bouzerdoum, "A pyramidal neural network for visual pattern recognition," *IEEE Transactions on Neural Networks*, vol. 27, no. 1, pp. 329–343, 2007.

[20] S. Munder and D. M. Gavrila, "An experimental study on pedestrian classification," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 11, pp. 1863–1868, 2006.

[21] I. Alonso, D. Llorca, M. Sotelo, L. Bergasa, P. Toro, M. Ocana, and M. Garrido, "Combination of feature extraction methods for SVM pedestrian detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 292–307, 2007.

[22] D. Chen, X. B. Cao, Y. W. Xu, H. Qiao, and F. Y. Wang, "A SVM-based classifier with shape and motion features for a pedestrian detection system," in *IEEE Intelligent Vehicles Symposium*, 2006, pp. 331–335.

[23] F. H. C. Tivive and A. Bouzerdoum, "A biologically inspired visual pedestrian detection system," in *2008 International Joint Conference on Neural Networks*, Hong Kong, China, 2008, pp. 704–710.

[24] S. L. Phung and A. Bouzerdoum, "MATLAB library for convolutional neural networks," University of Wollongong, Tech. Rep., URL: http://www.elec.uow.edu.au/staff/sphung, 2009.

[25] M. Riedmiller and H. Braun, "A direct adaptive method of faster backpropagation learning: The rprop algorithm," in *IEEE International Conference on Neural Networks*, San Francisco, 1993, pp. 586–591.

[26] A. B. G. H. Nguyen and S. L. Phung, "A supervised learning approach for imbalanced data sets," in *19th International Conference on Pattern Recognition*, Hawaii, USA, 2008, pp. 1–4.

[27] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An efficient k-means clustering algorithm: analysis and implemetation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 881 – 892, 2002.