Unsupervised Deep Network Pretraining via Human Design

Ming-Yu Liu
Mitsubishi Electric Research Labs (MERL)
Cambridge, MA, USA
mliu@merl.com

Arun Mallya
University of Illinois
Urbana-Champaign, IL, USA
arun.mallya@gmail.com

Oncel C. Tuzel
Mitsubishi Electric Research Labs (MERL)
Cambridge, MA, USA
oncel@merl.com

Xi Chen
University of Maryland
College Park, MD, USA
chenxistephen@gmail.com

Abstract

Over the years, computer vision researchers have spent an immense amount of effort on designing image features for visual object recognition tasks. We propose to leverage this valuable experience to guide the task of training deep neural networks. Specifically, we propose a new unsupervised pretraining method for deep neural networks. Our idea is to pretrain the network through the task of replicating classic hand-designed features. By learning to replicate these features, the deep network integrates previous research knowledge and learns to model visual objects in a way similar to the hand-designed features. In the succeeding finetuning step, it further learns object-specific representations from labeled data and this boosts its classification power. We demonstrate the application of our method on the pedestrian detection task. We pretrain two deep convolutional neural networks where one replicates the histogram of oriented gradients feature, and the other replicates the region covariance feature. After finetuning, we achieve substantially better performance than those networks pretrained using existing methods.

1. Introduction

Deep learning methods are revolutionizing the field of visual object recognition. Starting from the breakthrough in the ImageNet challenge [15], similar successes were achieved for several other image classification tasks [10, 27, 23]. These have demonstrated that powerful feature representations can be learned from data automatically, outdating traditional approaches based on hand-designed features. Following these successes, the focus of visual object recognition research has shifted from feature engineering to deep network design and optimization. One of the popular techniques developed for deep network optimization is pretraining, which is particularly important when the amount of labeled data is limited or when the network capacity is very large.

Pretraining refers to the technique of initializing the deep network parameters with the ones learned from applying the network to solve a different task. Most of the existing pretraining approaches are based on unsupervised learning [11, 3, 23]. These works pretrain the deep network in a layer-wise manner where a layer is pretrained after the preceding layer is pretrained. Each layer learns a non-linear transformation from the input to an intermediate representation which can be used to reproduce the input data. The reproducing capability suggests that the network encodes the main variation of the input data. Once the pretraining process is completed, the resulting network parameters are used.
to initialize a deep network, which is further optimized using labeled data; a process known as finetuning. It has been shown that the unsupervised pretraining guides the learning towards basins of attraction of minima that support better generalization capability [7]. A few supervised pretraining procedures have also been developed including the recent work by Girshick et al. [10] in which they pretrain a deep network by applying it to solve a different object recognition task where a large amount of labeled data is available.

In this paper, we propose a new unsupervised pretraining method. We propose to pretrain a deep network by solving a regression task that replicates the classic hand-designed feature extraction process (Fig. 1). Classic hand-designed features such as the Histogram of Oriented Gradients (HOG) [4]. Scale-Invariant Feature Transform (SIFT) [18], Local Binary Pattern (LBP) [20], region covariance (COV) [29, 30], etc., are the results of years of research effort and human intelligence, and are established tools for modeling visual objects for a variety of tasks. Our method is developed based on the insight that if the deep network can learn to replicate the hand-designed features, then it learns a way to capture object structure, similar to the hand-designed features. The following finetuning step can further boost the recognition performance by incorporating class specific information. Although using hand-designed features, our method is in line with the feature learning paradigm since it learns representations from data automatically. The hand-designed features are only used as “guidance” for initializing the deep network. Furthermore, our proposed pretraining method is unsupervised since the hand-designed features are extracted and used for pretraining without label information.

We illustrate the application of the proposed pretraining method on the pedestrian detection task, for which many hand-designed features have been actively developed. In particular, the HOG and region covariance features have achieved remarkable performance and extensive use. We apply the proposed method to pretrain two deep convolutional neural networks (CNNs), where one replicates the HOG feature while the other replicates the region covariance feature. Our feature replication deep networks are then finetuned with class label information. We benchmark the performance of our deep networks on public pedestrian detection datasets and achieve better performance than those trained using previously existing pretraining methods.

1.1. Related Work

For a long time, deep neural networks were believed to be too hard to train due to the tendency of gradient-based backpropagation methods to get stuck in local minimas, starting from random initializations. Effective strategies for training deep networks based on greedy layer-wise unsupervised pretraining through the use of Restricted Boltzmann Machines (RBMs) [11], and Stacked Auto-encoders [24, 3] were first developed around 2006. RBMs operate based on an energy-minimization criterion over the training set and are often trained using contrastive divergence algorithm. The auto-encoder tries to reconstruct the input using the activations of the hidden layer, often with the aim of learning a compressed representation of the input where the number of hidden units is lesser than the number of inputs and layer sizes are decreasing. Auto-encoders have also been shown to lead to networks with good generalization performance when then layer sizes are non-decreasing [3].

Supervised training methods have become popular for pretraining large CNNs [10,26]. Pretraining in these cases is performed by learning a classification task on a large labeled dataset (such as ImageNet), followed by finetuning on the limited dataset relevant to the task (such as PASCAL). Useful unsupervised methods have also been developed for pretraining CNNs, involving the use of RBMs [16], sparse-coding auto-encoders [25], and sparse-coding based nonlinear transforms [14, 13, 27].

As we demonstrate the use of our proposed pretraining method on the pedestrian detection task, we briefly review popular techniques in this area. Hand-designed features have been extensively researched and used for pedestrian detection. Gradient-based features such as Haar-like features [32], SIFT [31], and the very popular HOG [4] have been successfully used for the task. Texture-based features such as the Local Binary Pattern (LBP) have been used along with HOG in [33]. Even more recently proposed methods such as the one by Dollár et al. [5] use hand-designed Channel Features based on gradient histograms, gradients, and the LUV color space, while the state-of-the-art Spatial Pooling [22] uses covariance features and LBP along with spatial pooling. Multiple other methods which have at some point of time held the position of state-of-the-art on the INRIA or Caltech Pedestrian Detection datasets such as Roerei [2], Sketch-Tokens [17], Franken [19] have all relied on channel features.

There were several attempts at using deep learned features for pedestrian detection. Sermanet et al. [27] use a sparse-coding based method presented in [14] to learn a dictionary of filters in an unsupervised manner and [21] learns a deep-network with added deformation and occlusion handling. The success of human-designed features over deep-learned features for pedestrian detection further supports our belief that human supervision in the design of features is relevant and potentially better than generic unsupervised methods.

1.2. Contributions

The contributions of the paper are listed below:

- We propose a novel unsupervised pretraining method
based on replicating hand-designed features for training a deep network.

- We apply the proposed pretraining method to replicate the HOG and region covariance features using deep networks.
- We illustrate the application of the hand-designed feature replication deep networks for pedestrian application and achieve favorable performance compared to the results form the existing pretraining methods.

The paper is organized as follows: In Section 2, we present the proposed pretraining method. We illustrate its application to the pedestrian detection task in Section 3. Experimental results are shown in Section 4 and Section 5 concludes the paper.

2. Pretraining via Feature Replication

Our pretraining method is based on the replication of hand-designed features. While various hand-designed features can be used for the task, in this paper, we consider two examples: the HOG feature and the region covariance feature. For pretraining by replicating the HOG feature, we use a three-layer convolutional neural network, and for the region covariance feature, we use a four-layer convolutional neural network. Note that different network architectures can be used to replicate these features. The only requirements are that the number of output units should match the feature dimension, and the range of the output units should cover the range of the features (non-negative or real values).

2.1. HOG Feature Replication

We use a three-layer CNN shown in Fig. 2 in our experiments and pretrain it using the HOG feature. We refer to this network as HOGNet3 in the remainder of this paper. HOG feature computation utilizes a hierarchy of primitives of different levels of granularity. From high to low, they are image, block, cell, and pixel. An image contains a set of overlapping blocks, each block contains several cells, and each cell contains several pixels. The feature computation consists of three steps: 1) pixel-wise gradient computation, 2) cell-wise oriented gradient histogram computation, and 3) block-wise histogram normalization and concatenation. Specifically, the image is first convolved with derivative masks in the \( x \) and \( y \) directions for obtaining pixel-wise gradient magnitude and orientation. This is followed by quantizing gradient orientation to a set of bins, evenly spaced over 0 to 180 degrees. Pixels in a cell then distribute their gradient magnitudes to the orientation bins based on the gradient orientation. This constructs a histogram of oriented gradients for each cell. Histograms of the cells are individually normalized by the L2-norm of element-wise sum of cell histograms in the block. The normalized cell histograms are concatenated in a vector, which is the HOG feature for the block. The HOG features for the overlapping blocks in the image are further concatenated in a vector, which is the HOG feature for the image.

The power of the HOG feature is many-fold. The use of gradient orientation captures structural information of the object. The histogram binning avoids modeling unreliable fine details. The histogram normalization improves its invariance to illumination change. The HOG feature is generic in the sense that it does not contain class-specific information although better recognition performance can be achieved by using class-specific parameterization. The orientation binning and histogram normalization are the two major sources that contribute to the HOG features’ non-linear dependency on the input image.

We extract HOG features from grayscale image patches using the following parameters: image patch size of \( 16 \times 16 \), 9 orientation bins, cell-size of \( 8 \times 8 \), and a block-size of \( 16 \times 16 \). An image patch is mapped to a 36-dimensional HOG vector. The numbers of filters in the CNN layers are 40, 144, and 36 with sizes \( 7 \times 7 \), \( 5 \times 5 \), and \( 2 \times 2 \), respectively. The number of filters in the last layer is 36 for matching the dimension of HOG features. In HOGNet3, each of the convolutional filter banks is followed by a bank of ReLUs for increasing the network’s non-linearity, while keeping the optimization tractable. We apply non-overlapping max-pooling after the ReLU banks of the first and second CNN layers, with a pooling kernel size of \( 2 \times 2 \). Due to the ReLU, outputs of HOGNet3 are non-negative, the same as the HOG feature.

Our HOGNet3 is trained via solving a multi-dimensional regression problem. The image patches are sampled from natural images and Euclidean loss is used as the regression objective function. We use back propagation and stochastic gradient descent with momentum to update the network parameters for minimizing the Euclidean loss.

2.2. Region Covariance Feature Replication

In the region covariance feature representation, an image patch is represented by the covariance matrix of some pixel-wise features of the patch. While various pixel features can be used, we follow the original work and use an 8-dimensional feature vector for each pixel. The 8 dimensions are

\[
q = [x, y, |I_x|, |I_y|, \sqrt{I_x^2 + I_y^2}, |I_{xx}|, |I_{yy}|, \tan^{-1}\frac{I_x}{I_y}]^T
\]

(1)

where \( x \) and \( y \) are the pixel coordinates, \( I_x \) and \( I_y \) are the first derivatives in \( x \) and \( y \) directions, and \( I_{xx} \) and \( I_{yy} \) are the second derivatives in \( x \) and \( y \) directions. Let \( C \) denote the covariance matrix of \( q \) extracted from the pixels in an image region. Each of the elements in \( C \) encodes the correlation between two dimensions of \( q \).
The covariance matrix $C$ has interesting properties that are useful for modeling the visual object appearance in an image region. Firstly, it is invariant to global illumination shift. Increasing or decreasing the image intensity value by a constant does not affect $C$. One difficulty in working with covariance features is that the space of positive definite (covariance) matrices is not a Euclidean space, but its structure can be better analyzed using elements of Riemannian geometry. To this end, we use the log-Euclidean geometry proposed in [1] and embed the manifold to a Euclidean space (tangent space) via the log transform:

$$
\Sigma = U \log(S) U^T
$$

where $C = USU^T$ is the singular value decomposition of $C$. Note that the size of $\Sigma$ is the same as $C$ and is $8 \times 8$. Since $\Sigma$ is symmetric, we concatenate the upper triangular part of $\Sigma$ to form a compact vector representation for the region covariance feature, which is 36-dimensional. The region covariance feature can be extracted from an image region of any size. In this paper, we use a region size of $16 \times 16$.

In our experiments, we use a four-layer CNN shown in Fig. 2 referred to as COVNet4, and pretrain it using the region covariance feature. The first two layers of the COVNet4 are the same as the HOGNet3. They are 40 and 144 convolutional filter banks of size $7 \times 7$ and $5 \times 5$ resp., followed by ReLUs and $2 \times 2$ non-overlapping max-pooling. The third layer consists of 288 filters of size $2 \times 2$, while the fourth layer has 36 filters of size $1 \times 1$. We insert a bank of ReLUs after the third convolutional filter bank but not after the fourth one. This is because the fourth layer is the output layer for replicating the region covariance feature, which is a real number vector. Similar to the HOGNet3, the COVNet4 is trained via solving a multi-dimensional regression problem by minimizing the Euclidean loss.

### 3. Application to Pedestrian Detection

We demonstrate the application of the proposed pretraining method for the pedestrian detection task. We first pretrain the HOGNet3 and COVNet4 to replicate the HOG and region covariance features respectively. They are then fine-tuned using labeled data for pedestrian detection.

We use the scanning window approach proposed in [32], where a window is defined as a rectangular image region. For detecting and localizing pedestrians in an image, a classification model based on the window size is repeatedly applied to different image locations. It declares a detection at an image location if the classifier outputs a score that is greater than a preset threshold. The image is scaled to form an image pyramid for detecting pedestrians of different sizes in the original image. The window size for our pedestrian hypothesis is set to $64 \times 128$, which is also the choice reported in the original hand-designed feature works [4][50]. For this window size, both HOGNet3 and COVNet4 produce an output feature map of size $13 \times 29 \times 36$ when the stride of the convolution filters is set to 1.

We use the training images in the INRIA dataset [4] for training the deep networks. The training set contains 614 images with humans and 1218 images without humans. In the original INRIA dataset, only some of the human locations are provided. We follow the approach described in [27] and label all the human locations in the training set. This enables us to utilize the images with humans for hard-data mining, which is important as many false positive instances occur from parts of human bodies. We convert all the images to grayscale since both of the hand-designed features considered in the paper are designed for grayscale images. We thus only use the grayscale images for learning the deep networks.

We randomly sample a set of $16 \times 16$ image patches from the training images for pretraining. We extract the
HOG and region covariance features using their respective feature extractors and train the HOGNet3 and COVNet4 to learn the relationships between the input image patch and the extracted features. The number of image patches used for pretraining is about 15 million.

After pretraining, we finetune the networks using labeled data. We extract the pedestrian windows that are not heavily occluded from the training set for creating the positive sample set. In order to increase the variation, we perform data augmentation. We scale up and down the pedestrian windows by a factor of 1.05 and 0.95, respectively. We also randomly shift the pedestrian windows in the $x$ and $y$ directions by 2% of their height. The pedestrian windows obtained from the training images are of different sizes. We scale the height to 128 pixels and use the method suggested in [6] to normalize the windows. The normalized windows are then scaled to a fixed size of $64 \times 128$. These constitute a set of positive samples which are around $90K$ in number. Also, we sample a set of windows from the training images that do not contain humans to construct the negative sample set. The number of samples of the negative set is around $50K$. These windows are resized to $64 \times 128$ to match the positive windows. We add a softmax output layer on top of the HOGNet3 and COVNet4 respectively and train the networks by minimizing the softmax loss. Below, we discuss some commonly used techniques for optimizing performance in object detection, which we integrate into our system.

Hard-data mining is a practical technique that is known to boost detection performance of the classifiers. After finetuning, we apply the sliding window technique and use the learned networks to detect pedestrians in all the images in the INRIA training set. We store the scores of the false positive windows and add the top scoring $20K$ false positive windows to the negative training set. We then fine-tune HOGNet3 and COVNet4, where the network parameters are initialized using the parameters learned from the previous finetuning step. This completes a run of hard-data mining. We repeat the hard-data mining procedure for several runs until the number of false positive samples falls below $2K$. Overall, we perform 8 hard-data mining runs for the HOGNet3 and 6 runs for COVNet4.

After hard-data mining, we remove the softmax output layer from the HOGNet3 and COVNet4 and use the networks as a feature extractor. The extracted features are then fed to a linear SVM for learning the maximum margin classifier. This approach has been adopted in several previous works including [10, 27]. Instead of using the L1-Hinge Loss SVM, we use the L2-Hinge Loss SVM as the quadratic cost allows for faster optimization. To train the SVM, we use LIBLINEAR, a publicly available large-scale linear SVM implementation [8]. We use the training data obtained from the last hard-data mining iteration for training the SVM, where the negative data is weighted 20 times more than the positive data and the L2-Hinge Loss is weighted by 1.

The sliding window detection approach often declares several detections around a pedestrian hypothesis. In order to combine the multiple detections, we apply non-maximum suppression. We use the pairwise window matching method for non-maximum suppression. Let $W_1$ and $W_2$ be two window hypotheses and assume that the detection score of $W_1$ is larger than that of $W_2$. We suppress $W_2$ if it has a large overlap with $W_1$. Two popular measures are available for measuring the overlap. One is based on the intersection over union measure (IoU), which is the ratio of the size of the intersection of $W_1$ and $W_2$ divided by the size of their union. The other measure is based on the ratio of the size of the intersection of $W_1$ and $W_2$ over the size of $W_2$, called Io2. We found that applying the two measures in sequence renders slightly better performance than applying either of them alone for our detector networks. The threshold we use for IoU is 0.4, while the threshold for Io2 is 0.6.

In order to reduce localization errors, we train a linear regression model to improve the bounding boxes of the pedestrian detections obtained from the SVM classifier. Following the methodology of [10], we use the features outputted from the last layer of the network to learn four functions that map the $x$, $y$ locations, height, and width of the proposed detections ($P^i$) to their corresponding ground truth locations ($G^i$) using pairs ($P^i, G^i$). The ground truth pairs are collected from applying the SVM classifier to scan through the pedestrian hypothesis in the training set. During test time, we apply the learned functions on the proposed detections ($P^i$) to map them to a predicted ground-truth box ($G^i$). The training data ($P^i, G^i$) is selected such that $P^i$ has an SVM score of at least 0.5 and intersects $G^i$ with an IoU measure of at least 0.6.

4. Experiments

We first analyze the proposed pretraining method and then present results on the pedestrian detection task, comparing it with the competing approaches. All the experiments were conducted using a PC with an Intel i7 multi-core processor and a Nvidia K40 Tesla GPGPU card.

4.1. Experimental Results for Pretraining

We used the publicly available convolutional neural network library Caffe [12] for training the deep networks. It utilizes the GPU architecture to parallelize the feedforward and gradient computations, allowing efficient training of the network. The mini-batch size used for stochastic gradient descent was $10.5K$, with a base learning rate of 0.001 and momentum weight of 0.9. As the magnitude of the region covariance feature was larger than the HOG feature, we used a smaller base learning rate of 0.00001. We used
The average error per dimension was 0.05.

For the INRIA training set, the average error was reduced to 0.065 on the test set, where the average magnitude of the HOG features was 0.14.

In Fig. 3, we plot the pretraining performance of replicating the HOG features. The figure shows some of the first layer convolutional filters in the HOGNet3. The filters resemble edge detectors of different orientations, capturing the behavior of the HOG feature extractor.

In Fig. 4, we visualize the filters learned in the first convolutional layer of the HOGNet3. We found that the filters resembled edge filters of different orientations. As the HOG feature is based on oriented gradients, these filters captured gradient magnitudes at different orientations from the input image and passed the results to the succeeding layers for mimicking the histogram operation. The remaining filters not shown either resembled edge filters of small gradients or some small magnitude random patterns. We did not observe filters resembling texture filters as observed in [27].

In Fig. 5, we plot the pretraining performance of replicating the region covariance features using the COVNet4. Unlike the replication of the HOG features, we found that both of the training and test errors decreased smoothly, probably due to the smaller learning rate. After 40 epochs, the testing error per feature dimension was about 0.14, while the average magnitude of the region covariance feature was 0.26.

Fig. 6 displays the first layer convolutional filters learned by replicating the region covariance features. Unlike those in HOGNet3, the filters in COVNet4 were more similar to the first-order and second-order gradient operators in the $x$ and $y$ directions, which are used to compute the region covariance feature in Equation (1). Similar to the HOG case, the remaining filters were not texture filters but random patterns of small magnitude.

**4.2. Experimental Results for Pedestrian detection**

We finetuned the HOGNet3 and COVNet4 using the labeled data in the INRIA training set for pedestrian detection as described in Section 3. As the pedestrian window size was several times larger than the image patch used in the

In order to evaluate the pretraining performance, we also sampled a set of image patches from the INRIA test dataset. Similar to the INRIA training dataset, the INRIA test dataset consists of two sets of images: one set with humans and the other without humans. The number of images in the two sets are 287 and 453, respectively. We converted all the images in the dataset to grayscale and randomly sampled 10 million $16 \times 16$ image patches from both the sets and used the HOG and region covariance feature extractors to construct the test set.

We pretrained the HOGNet3 and COVNet4 by minimizing the Euclidean loss between the network outputs and the extracted hand-designed features. We trained the HOGNet3 for 100 epochs but trained the COVNet4 for 40 epochs since the training loss decreased very slowly after 30 epochs for COVNet4. After each epoch, we stored the parameters of the networks and evaluated their performance on the test set.

In Fig. 5 we plot the pretraining performance of replicating the region covariance features using the COVNet4. Unlike the replication of the HOG features, we found that both of the training and test errors decreased smoothly, probably due to the smaller learning rate. After 40 epochs, the testing error per feature dimension was about 0.14, while the average magnitude of the region covariance feature was 0.26.

Fig. 6 displays the first layer convolutional filters learned by replicating the region covariance features. Unlike those in HOGNet3, the filters in COVNet4 were more similar to the first-order and second-order gradient operators in the $x$ and $y$ directions, which are used to compute the region covariance feature in Equation (1). Similar to the HOG case, the remaining filters were not texture filters but random patterns of small magnitude.

### 4.2. Experimental Results for Pedestrian detection

We finetuned the HOGNet3 and COVNet4 using the labeled data in the INRIA training set for pedestrian detection as described in Section 3. As the pedestrian window size was several times larger than the image patch used in the
pretraining, we decreased the the mini-batch size to 100 in order to fit the data into the memory. The base learning rate was set to 0.01 for both networks, while the other parameters were kept the same as those used in the pretraining step. We initialized the softmax-appended HOGNet3 and COVNet4 by using the network parameters learned in the respective pretraining steps and finetuned them for 10 epochs. After finetuning, we applied the sliding window technique for hard-data mining using the INRIA training set as described in Section 3.

For facilitating the comparison, we also applied the autoencoder technique to pretrain a deep network using a similar architecture. We used an autoencoder which tries to reconstruct the half-sized input through a fully-connected layer after three convolutional layers of dimensions same as the HOGNet3 network. We trained the network to reconstruct the input in the end-to-end fashion instead of using the greedy layerwise pretraining. This network is referred to as AutoNet3.

We evaluated the performance of the trained networks for pedestrian detection using the INRIA test dataset and Daimler test dataset. The INRIA test dataset, composed of personal photographs, contains 287 color images with humans, which we converted to grayscale. The Daimler test dataset was captured using a camera mounted on a vehicle driving around in a city and the captured images are in grayscale. The Daimler dataset, which contains 21787 images is several magnitudes larger than the INRIA dataset. To detect pedestrians at different scales, we searched through three octaves with a base scale of 1.07. Although the Daimler dataset also includes a training set, we did not retrain our networks using the set. Our networks were trained only using the INRIA training set. There are several other datasets proposed for pedestrian detection as summarized in [6], all of which are taken from street views in the color image format, except for the INRIA dataset and Daimler dataset. Our network does not utilize color information, which is a valuable cue for pedestrian detection since the colors of buildings, trees, and traffic signs are usually very different to a pedestrian’s appearance.

The Caltech toolbox described in [6] was used for performance evaluation. The detection performance of each algorithm was visualized using a Receiver Operating Characteristic (ROC) curve where the y axis is the miss rate and the x axis is the false positive per image (FPPI). The curve is plotted in the log scale. It is difficult to compare two algorithms based on the ROC curves as the toolbox uses the average miss rate for summarizing the ROC curve. Specifically, the average of the miss rates for the 9 points evenly spread between 0.01 to 1 FPPI in the log scale is used to summarize the performance of an algorithm. The miss rate from the region where FPPI is less than 0.01 is not used in the comparison.

In Fig. 7, we show the performance improvement resulting from the hard-data mining step during finetuning of the HOGNet3. Similar behavior was displayed by the COVNet4. While more iterations helped, the improvements also diminished with iterations.

We compare the performance of the deep networks trained using different pretraining techniques in Fig. 8. It includes the HOGNet3, COVNet4, AutoNet3, and ConvNet [27]. All networks except COVNet4 were based on three
convolutional layers. The ConvNet [27] had a very different structure, using YUV images as input, and absolute value rectification and contrast normalization for nonlinearity. It also randomly selected 20% of the neuron connections for breaking the symmetry and used an SVM for the output layer, along with a sparse-coding based pretraining.

From Fig. 8 we observe that the HOGNet3 obtained an average miss rate of 19%, outperforming the 20% of the ConvNet [27], the 25% of the COVNet4, and 38% of the AutoNet3. When utilizing the SVM classifier and the bounding box prediction technique, the performance was further improved to 13%. The AutoNet3 did not perform well, probably due to the pretraining. The autoencoder was not trained in a stage-wise manner and the end-to-end training probably led the network to a poor local minima. In the remaining part of the section, we only report the performance of the HOGNet3 when used with SVM and the bounding box regression technique.

Fig. 10 compares the HOGNet3 with the state-of-the-art pedestrian detection methods on the INRIA test set. The performance data of the other algorithms were obtained from the Caltech pedestrian detection website. The HOGNet3 obtained an average miss rate of 13.2%, which was second only to the Spatial Pooling algorithm [22]. The Caltech toolbox only measured the detection performance between 0.01 to 1.0 FPPI. It excludes the low FPPI region which is of great interest for autonomous driving. When extending the FPPI range included for computing the average miss rate to the leftmost point of the curves, we found that the HOGNet3 achieved the best average miss rate of 16.2%, which was better than the average miss rate of 16.9% obtained by [22].

Fig. 11 compares the performance of the HOGNet3 with the state-of-the-art pedestrian detection methods on the Daimler test set. The HOGNet3 performed better than the ConvNet [27] to obtain the first place. In Fig. 9 we visualize some of the pedestrian detection outputs on the INRIA and Daimler datasets.
5. Conclusion

We have shown that hand-designed features can be used for pretraining neural networks in a very simple, but powerful manner to achieve comparable or better performance than traditional and complicated pretraining methods. We believe that this method can be extended for greedy pretraining of the layers of a much deeper neural network through a procedure similar to that of an autoencoder. While this work serves as a proof of concept, using multiple features such as HOG, COV, color descriptors together will undoubtedly result in a more robust and diverse range of well-initialized neurons.

References

[1] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache. Geometric means in a novel vector space structure on symmetric positive-definite matrices. SIAM journal on matrix analysis and applications, 29(1):328–347, 2007.
[2] R. Benenson, M. Mathias, T. Tuytelaars, and L. V. Gool. Seeking the strongest rigid detector. In CVPR, 2013.
[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. NIPS, 2007.
[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005.
[5] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In BMVC, 2009.
[6] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: an evaluation of the state of the art. TPAMI, 34(4):743–761, 2012.
[7] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? JMLR. 11:625–660, 2010.
[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: a library for large linear classification. TPAMI, 9:1871–1874, 2008.
[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. TPAMI, 32(9):1627–1645, 2010.
[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.
[11] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. Neural computation, 18(7):1527–1554, 2006.
[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. arXiv preprint arXiv:1408.5093, 2014.
[13] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. Le-Cun. Learning invariant features through topographic filter maps. In CVPR, 2009.
[14] K. Kavukcuoglu, P. Sermanet, Y. Ian Boureau, K. Gregor, M. Mathieu, and Y. L. Cun. Learning convolutional feature hierarchies for visual recognition. In NIPS, 2010.
[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.
[16] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In ICML, 2009.
[17] J. J. Lim, C. L. Zitnick, and P. Dollar. Sketch tokens: a learned mid-level representation for contour and object detection. In CVPR, 2013.
[18] D. G. Lowe. Object recognition from local scale-invariant features. In ICCV, 1999.
[19] M. Mathias, R. Benenson, R. Timofte, and L. V. Gool. Handling occlusions with franken-classifiers. In ICCV, 2013.
[20] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. TPAMI, 24(7):971–987, 2002.
[21] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In ICCV, 2013.
[22] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Strengthening the effectiveness of pedestrian detection with spatially pooled features. In ECCV, 2014.
[23] M. Ranzato, C. Poultney, S. Chopra, Y. L. Cun, et al. Efficient learning of sparse representations with an energy-based model. In NIPS, pages 1137–1144, 2006.
[24] M. A. Ranzato, Y. Ian Boureau, and Y. L. Cun. Sparse feature learning for deep belief networks. In NIPS, 2008.
[25] M. A. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun. Efficient learning of sparse representations with an energy-based model. In NIPS, 2007.
[26] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. Cnn features off-the-shelf: an astounding baseline for recognition. arXiv preprint arXiv:1403.6822, 2014.
[27] P. Sermanet, K. Kavukcuoglu, and S. Chintala. Pedestrian detection with unsupervised multi-stage feature learning. In CVPR, 2013.
[28] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. Deepface: closing the gap to human-level performance in face verification. In CVPR, 2014.
[29] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In ECCV, 2006.
[30] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. TPAMI, 2008.
[31] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In ICCV, 2009.
[32] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In ICCV, 2003.
[33] X. Wang, T. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In ICCV, 2009.