MITSUBISHI ELECTRIC RESEARCH LABORATORIES
http://www.merl.com

Unsupervised Network Pretraining via Encoding Human Design

Liu, M.-Y.; Mallya, A.; Tuzel, C.O.; Chen, X.

TR2016-022 March 2016

Abstract

Over the years, computer vision researchers have spent an immense amount of effort on designing image features for the visual object recognition task. We propose to incorporate this valuable experience to guide the task of training deep neural networks. Our idea is to pretrain the network through the task of replicating the process of hand-designed feature extraction. By learning to replicate the process, the neural 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 pretrain two convolutional neural networks where one replicates the process of histogram of oriented gradients feature extraction, and the other replicates the process of region covariance feature extraction. After finetuning, we achieve substantially better performance than the baseline methods.

IEEE Winter Conference on Applications of Computer Vision (WACV)

Unsupervised Network Pretraining via Encoding Human Design

Ming-Yu Liu*[1], Arun Mallya[2], Oncel Tuzel[1], and Xi Chen[3]

[1]Mitsubishi Electric Research Labs (MERL), Cambridge MA, USA
[2]University of Illinois, Urbana-Champaign, IL, USA
[3]University of Maryland, College Park, MD, USA

Abstract

Over the years, computer vision researchers have spent an immense amount of effort on designing image features for the visual object recognition task. We propose to incorporate this valuable experience to guide the task of training deep neural networks. Our idea is to pretrain the network through the task of replicating the process of hand-designed feature extraction. By learning to replicate the process, the neural 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 pretrain two convolutional neural networks where one replicates the process of histogram of oriented gradients feature extraction, and the other replicates the process of region covariance feature extraction. After finetuning, we achieve substantially better performance than the baseline methods.

1. Introduction

Deep learning methods are revolutionizing the field of visual object recognition. Starting from the breakthrough in image classification [16], similar successes were achieved for several other computer vision tasks [10, 27]. These have demonstrated that powerful feature representations can be learned from data automatically, out-dating 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 techniques developed for deep network optimization is pretraining, which helps when the amount of labeled data is limited.

Pretraining refers to the technique of initializing the 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 [12, 3, 22]. 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 the 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].

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

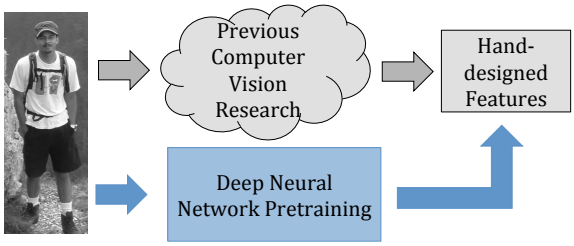

Figure 1: **Network pretraining via replicating hand-designed features:** We propose an unsupervised pretraining method based on hand-designed feature replication for deep learning. Through learning to replicate the features, the deep network utilizes the past computer vision research knowledge and learns a way to model the visual object structure. A subsequent supervised finetuning step further optimizes the deep network to achieve better recognition performance.

*mliu@merl.com

(HOG) [4] and region covariance (COV) [28], etc., are the results of years of research effort and human intelligence, and are established tools for modeling visual objects appearance. Our method is developed based on the insight that if the 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.

For certain visual object recognition tasks, the discriminative information is quite sparse as compared to the available information contained in the image. For example, shape is an important cue but color is not for pedestrian detection. The reconstruction-based pretraining methods have the tendency of favoring encoding information required for reconstructing the image in the network, which dilutes the discriminative information critical for the task in the network. When such discriminative information is present in the human-designed feature, our method encourages the network to encode this discriminative information primarily.

We verify the proposed pretraining method on the pedestrian detection task, for which many hand-designed features have been developed. In particular, the HOG and region covariance features have achieved remarkable performance and extensive use. We apply the proposed method to pretrain two convolutional neural networks (CNNs), where one replicates the HOG feature while the other replicates the region covariance feature. Our feature replication networks are then finetuned with class label information. We evaluate the performance on two public datasets. The experimental results show that the proposed method achieves better performance than the baseline methods.

1.1. Contributions

The contributions of the paper are listed below:

- We propose an unsupervised pretraining method based on replicating the process of hand-designed feature extraction for training a neural network.
- We apply the proposed method to pretrain two neural networks, one replicating the process of HOG feature extraction and the other replicating the process of region covariance feature extraction, for the pedestrian detection task. Experimental results show that our pretraining method achieves favorable performance compared to several autoencoder-based pretraining methods. The resulting network achieves near state-of-the-art performance on pedestrian detection problem.

2. 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 minima or flat regions, starting from random initializations. First effective strategies for unsupervised pretraining of deep networks were based on greedy layer-wise optimization such as Restricted Boltzmann Machines (RBMs) [12], and Stacked Autoencoders [23, 3]. RBMs operate based on an energy-minimization criterion over the training set. The autoencoder tries to reconstruct the output 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 less than the number of inputs and layer sizes are decreasing. Autoencoders have also been shown to lead to networks with good generalization performance when the layer sizes are non-decreasing [3].

Unsupervised methods have also been developed for pretraining CNNs, involving the use of convolutional RBMs [17], sparse-coding autoencoders [24], and sparse-coding based non-linear transforms [15, 14, 25].

We verify our proposed pretraining method on the pedestrian detection task. Here, 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 [31], SIFT [30], 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 [32]. More recently, Dollár *et al.* [5] proposed Integral Channel Features based on gradient histograms, gradients, and the LUV color space. The current state-of-the-art *Spatial Pooling* [21] uses covariance features and LBP along with spatial pooling. Some other examples of hand-designed features for pedestrian detection include [2, 18, 19].

There were several deep learning-based methods for pedestrian detection. Sermanet *et al.* [25] use a sparse-coding based method [15] to learn a dictionary of filters in an unsupervised manner and [20] learns a deep-network with added deformation and occlusion handling. The success of human-designed features over deep learning methods for pedestrian detection further supports our belief that past human experience in the form of hand-designed features can guide learning deep networks.

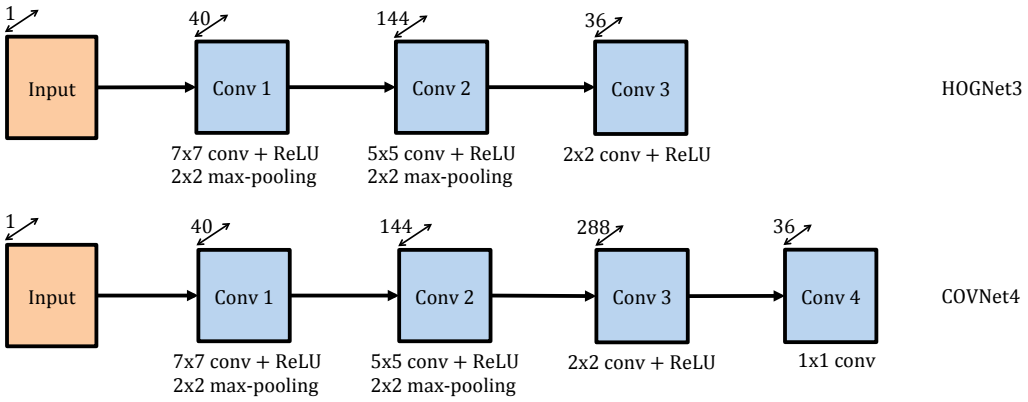

Figure 2: **Experimental network structures:** The HOGNet3 & COVNet4 networks are pretrained through Euclidean loss minimization between the network output and the HOG & COV features of the input image, respectively.

3. 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).

3.1. HOG Feature Replication

We use a three-layer CNN shown in Fig. 2 in our experiments and pretrain it using the HOG feature [4]. We refer to this network as HOGNet3. 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 [9]. The orientation binning and histogram normalization are the two major sources that contribute to the HOG features' nonlinear dependency on the input image.

We extract HOG features from grayscale image patches using the following parameters: image patch size of 16×16, 9 orientation bins, cell-size of 8×8, and a block-size of 16×16. 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 7×7, 5×5, and 2×2, 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 2×2. 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.

3.2. Region Covariance Feature Replication

In the region covariance feature representation [29], 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 [29] and use an eight-dimensional feature vector for each pixel. The eight dimensions are

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

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 $\mathbf{q}$ extracted from the pixels in an image region. Each of the elements in C encodes the correlation between two dimensions of $\mathbf{q}$.

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 Σ is the same as C and is 8×8. Since Σ is symmetric, we concatenate the upper triangular part of Σ 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 16×16.

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 7×7 and 5×5 resp., followed by ReLUs and 2×2 non-overlapping max-pooling. The third layer consists of 288 filters of size 2×2, while the fourth layer has 36 filters of size 1×1. 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.

4. Pedestrian Detection

We verify the proposed pretraining method on the pedestrian detection task. We first pretrain the HOGNet3 and COVNet4 to replicate the HOG and region covariance features respectively. They are then finetuned using labeled data for pedestrian detection.

We use the scanning window approach proposed in [31], 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 ouputs 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 64×128, which is also the choice reported in the original hand-designed feature works [4, 29]. 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 [25] 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 16×16 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 64×128. 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 64×128 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 finetune 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, 25]. 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^j) to map them to a predicted ground-truth box ($\hat{G}^j$). 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.

5. Experiments

We first analyze the proposed pretraining method for replicating two hand-designed features. Next, we finetune the networks for the pedestrian detection task and compared them with the existing algorithms. All the experiments were conducted using a PC with an Intel i7 multi-core processor and a Nvidia K40 Tesla GPGPU card.

5.1. Experimental Results for Pretraining

We used the publicly available convolutional neural network library *Caffe* [13] 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 inverse law for adjusting the learning rate for pretraining. The networks were regularized by the L2 loss on the network parameters with a weight decay of 0.0005. As described in the section 4, the training data was obtained from the INRIA training set.

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 16×16 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

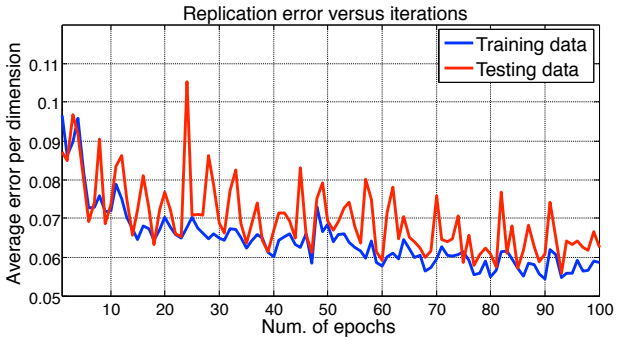

Figure 3: **Replication of the HOG features:** The figure plots the training and test error of the HOGNet3.

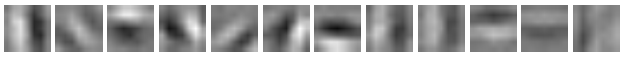

Figure 4: **Visualization of HOGNet3:** The figure shows some of the first layer convolutional filters in the HOGNet3. The filters resemble edge detectors of different orientations, capturing the behavior of the HOG feature extractor.

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. 3, we plot the pretraining performance of the HOGNet3 where the y-axis is the average estimation error per feature dimension and the x-axis denotes the number of epochs. As shown in the graph, both of the training errors decreased gradually with some fluctuations. We did not observe significant overfitting. After 100 epochs, the average error was reduced to 0.065 on the test set, where the average magnitude of the HOG features was 0.14.

Fig. 4 visualizes 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 [25].

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.

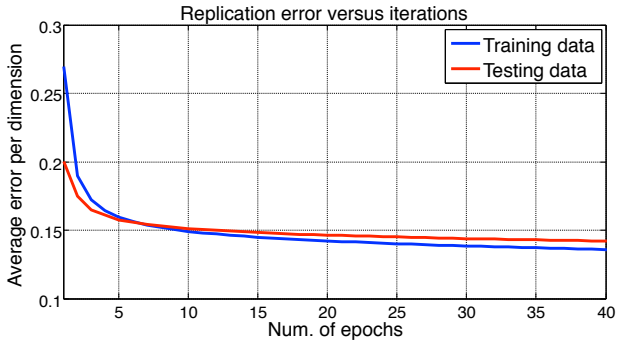

Figure 5: **Replication of the region covariance features:** The figure plots the training and test error of the COVNet4.

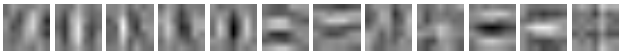

Figure 6: **Visualization of COVNet4:** The figure shows some of the first layer convolutional filters in the COVNet4. The filters resemble 1^{st} and 2^{nd} order image derivative operators, capturing the behavior of the region covariance feature extractor.

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 features.

5.2. 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 4. 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. A dropout layer [26] was added to the last feature layer for better generalization performance. After finetuning, we applied the sliding window technique for hard-data mining using the INRIA training set as described in Section 4.

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

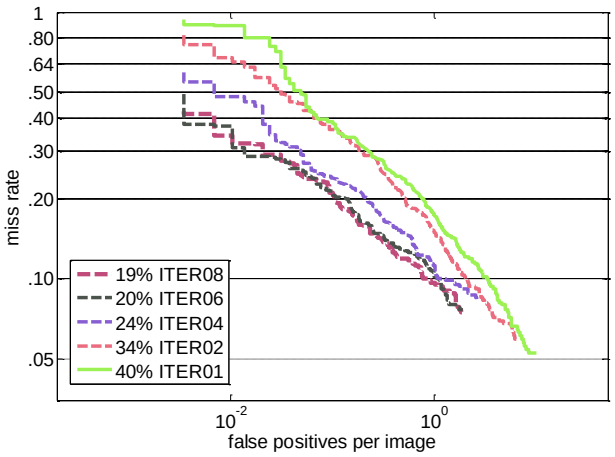

Figure 7: **Effect of hard-data mining:** The figure illustrates the detection performance improvement on the INRIA test dataset, from the hard-data mining step performed during finetuning the HOGNet3.

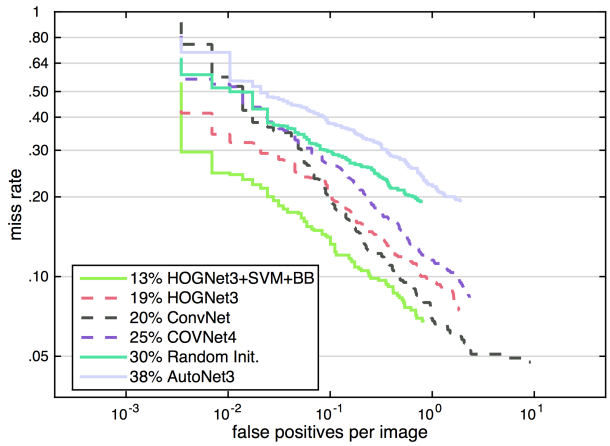

Figure 8: **Pretraining method comparison:** The figure compares the performance of the networks trained with different pretraining methods on the INRIA test dataset. Our HOGNet3 achieves the best performance, outperforming the ConvNet [25], which is trained based on a sparse-coding based pretraining method [15].

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. Further, we also finetuned a network with the same architecture as HOGNet3 but initialized with random weights to show the usefulness of our pretraining method.

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 using the INRIA training set.

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 networks trained using different pretraining techniques in Fig. 8. It includes the HOGNet3, COVNet4, AutoNet3, Random Init., and ConvNet [25]. The ConvNet had a very different structure, using YUV images as input, and absolute value rectification and contrast normalization for nonlinearity. It used a sparse-coding based pretraining method.

From Fig. 8, we observe that the HOGNet3 obtained an average miss rate of 19%, outperforming the 20% of the the ConvNet [25], the 25% of the the COVNet4, the 38% of the AutoNet3 and the 30% of the randomly initialized network. When utilizing the SVM classifier and the bounding box prediction technique, our performance was further improved to 13%. The AutoNet3 did not perform well since it tried to encode the information required for reconstruction, which dilutes the critical information for pedestrian detection. The improvement over random initialization of the network clearly shows the usefulness of our pretraining scheme. 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. 9 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

Figure 11: Example pedestrian detection outputs of the HOGNet3 on the INRIA and Daimler test datasets.

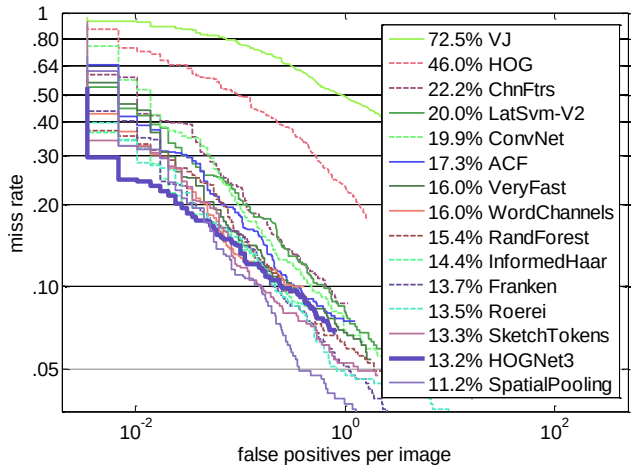

Figure 9: **Results on the INRIA test dataset:**

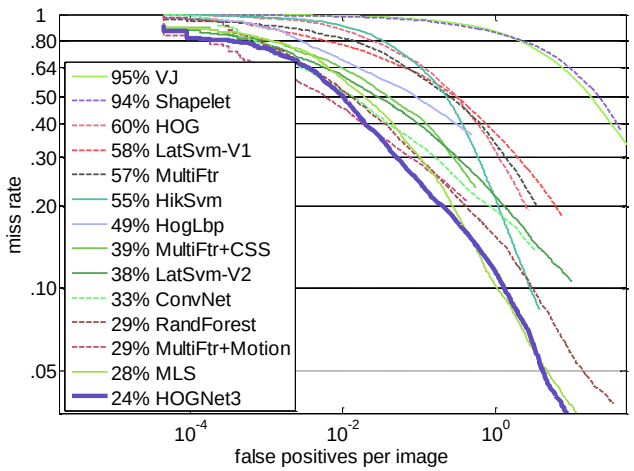

Figure 10: **Results on the Daimler test dataset:**

HOGNet3 obtained an average miss rate of 13.2%, which was second only to the *Spatial Pooling* algorithm [21]. It significantly outperformed the baseline methods including the HOG detector (46.0%) [4] and ConvNet (19.9%) [25].

Fig. 10 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 [25] to obtain the first place. In Fig. 11, we visualize some of the pedestrian detection outputs on the INRIA and Daimler datasets.

6. Conclusion

Through the example of pedestrian detection, we have shown that hand-designed features can be used for pretraining deep neural networks in a simple but effective way. Our method was based on the insight that discriminative information encoded in the hand-designed feature can be transferred to the network via pretraining. It can be later integrated with class specific information in the finetuning stage to boost the performance. Our method is useful for the recognition task where only a small amount of training data are available but a good feature can be hand engineered.

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. 4

[2] R. Benenson, M. Mathias, T. Tuytelaars, and L. V. Gool. Seeking the strongest rigid detector. In *Conference on Computer Vision and Pattern Recognition*, 2013. 2

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Neural Information Processing Systems (NIPS)*, 2007. 1, 2

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Conference on Computer Vision and Pattern Recognition*, 2005. 2, 3, 4, 8

[5] P. Dollar, Z. Tu, P. Perona, and S. Belongie. Integral channel features. In *British Machine Vision Conference (BMVC)*, 2009. 2

[6] P. Dollar, C. Wojek, B. Schiele, and P. Perona. Pedestrian detection: an evaluation of the state of the art. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 34(4):743–761, 2012. 4, 7

[7] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, and S. Bengio. Why does unsupervised pre-training help deep learning? *Journal on Machine Learning Research*, 11:625–660, 2010. 1

[8] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: a library for large linear classification. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 9:1871–1874, 2008. 5

[9] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. 3

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Conference on Computer Vision and Pattern Recognition*, 2014. 1, 5

[11] G. Hinton. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

[12] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006. 1, 2

[13] 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. 5

[14] K. Kavukcuoglu, M. Ranzato, R. Fergus, and Y. Le-Cun. Learning invariant features through topographic filter maps. In *Conference on Computer Vision and Pattern Recognition*, 2009. 2

[15] K. Kavukcuoglu, P. Sermanet, Y. lan Boureau, K. Gregor, M. Mathieu, and Y. L. Cun. Learning convolutional feature hierarchies for visual recognition. In *Neural Information Processing Systems (NIPS)*. 2010. 2, 7

[16] A. Krizhevshky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural netowrks. In *Neural Information Processing Systems (NIPS)*, 2012. 1

[17] H. Lee, R. Grosse, R. Ranganath, and A. Y. Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *International Conference on Machine Learning*, 2009. 2

[18] J. J. Lim, C. L. Zitnick, and P. Dollar. Sketch tokens: a learned mid-level representation for contour and object detection. In *Conference on Computer Vision and Pattern Recognition*, 2013. 2

[19] M. Mathias, R. Benenson, R. Timofte, and L. V. Gool. Handling occlusions with franken-classifiers. In *International Conference on Computer Vision*, 2013. 2

[20] W. Ouyang and X. Wang. Joint deep learning for pedestrian detection. In *International Conference on Computer Vision*, 2013. 2

[21] S. Paisitkriangkrai, C. Shen, and A. van den Hengel. Strengthening the effectiveness of pedestrian detection with spatially pooled features. In *European Conference on Computer Vision*. 2014. 2, 8

[22] M. Ranzato, C. Poultney, S. Chopra, Y. L. Cun, et al. Efficient learning of sparse representations with an energy-based model. In *Neural Information Processing Systems (NIPS)*, pages 1137–1144, 2006. 1

[23] M. A. Ranzato, Y. lan Boureau, and Y. L. Cun. Sparse feature learning for deep belief networks. In *Neural Information Processing Systems (NIPS)*. 2008. 2

[24] M. A. Ranzato, C. Poultney, S. Chopra, and Y. L. Cun. Efficient learning of sparse representations with an energy-based model. In *Neural Information Processing Systems (NIPS)*. 2007. 2

[25] P. Sermanet, K. Kavukcuoglu, and S. Chintala. Pedestrian detection with unsupervised multi-stage feature learning. In *Conference on Computer Vision and Pattern Recognition*, 2013. 2, 4, 5, 6, 7, 8

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal on Machine Learning Research*, 15(1):1929–1958, 2014. 6

[27] Y. Taigman, M. Yang, M. A. Ranzato, and L. Wolf. Deepface: closing the gap to human-level performance in face verification. In *Conference on Computer Vision and Pattern Recognition*, 2014. 1

[28] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *European Conference on Computer Vision*. 2006. 2

[29] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 2008. 4

[30] A. Vedaldi, V. Gulshan, M. Varma, and A. Zisserman. Multiple kernels for object detection. In *International Conference on Computer Vision*, 2009. 2

[31] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *International Conference on Computer Vision*, 2003. 2, 4

[32] X. Wang, T. Han, and S. Yan. An hog-lbp human detector with partial occlusion handling. In *International Conference on Computer Vision*, 2009. 2