Large-scale Learning with SVM and Convolutional Nets for Generic Object Categorization

Fu Jie Huang, Yann LeCun
The Courant Institute of Mathematical Sciences
New York University, New York, NY, USA
{jhuangfu,yann}@cs.nyu.edu

Abstract

The detection and recognition of generic object categories with invariance to viewpoint, illumination, and clutter requires the combination of a feature extractor and a classifier. We show that architectures such as convolutional networks are good at learning invariant features, but not always optimal for classification, while Support Vector Machines are good at producing decision surfaces from well-behaved feature vectors, but cannot learn complicated invariances. We present a hybrid system where a convolutional network is trained to detect and recognize generic objects, and a Gaussian-kernel SVM is trained from the features learned by the convolutional network.

Results are given on a large generic object recognition task with six categories (human figures, four-legged animals, airplanes, trucks, cars, and "none of the above"), with multiple instances of each object category under various poses, illuminations, and backgrounds. On the test set, which contains different object instances than the training set, an SVM alone yields a 43.3% error rate, a convolutional net alone yields 7.2% and an SVM on top of features produced by the convolutional net yields 5.9%.

1 Introduction

The detection and recognition of generic object categories with invariance to pose, illumination, clutter, and occlusions has motivated several research efforts in the last few years. It is a considerably more challenging problem than detection alone, or recognition alone.

While model-based specific object recognition systems have had a long history, the recognition of broad, generic categories with wide variability of form such as "animals", or "airplanes" is still a very challenging problem. The task of detecting and classifying objects in such high-level categories is an ideal challenge for machine learning methods. Discriminative methods can be tested in situations where the number of categories is small while the intra-class variations can be very large due to shape variations within a category, or due to variations in viewpoint, lighting conditions, and background clutter.

In this paper, we investigate two types of popular discriminative learning methods: Support Vector Machines (SVM) and Convolutional Nets. SVMs [2] have become a standard tool in the classification toolbox in the last decade. They are theoretically appealing because [32]: (1) with an appropriate choice of the kernel parameters, they can in principle learn any training set perfectly; (2) the loss function minimized by the learning algorithm is convex; (3) the maximum margin criterion gives raise to "sparse" solutions with theoretical bounds on the generalization error; (4) once the kernel function is chosen, the algorithm is relatively free of adjustable parameters, except the maximum cost of misclassified samples. In practice, however, the SVM training is very computationally intensive, and scales badly with the size of the data sets. Large-scale experiments (10^5 - 10^6 samples) on high-dimensional data (10^4 variables) have so far been limited. Another limitation of SVMs is that the "standard" kernels, such as the Gaussian kernel, are woefully inadequate for image recognition from raw pixels. With a Gaussian kernel, the SVM architecture essentially reduces to performing pixel-level global template matching with stored training samples, and linearly combining the matching scores. As we will show, such an architecture cannot handle the wide variability of appearance in pixel images that result from variations in pose, illumination, and clutter, unless an impractically large number of templates (support vectors) are used. Ad-hoc preprocessing and feature extraction can, of course, be used to mitigate the problem, but this concentrates on methods that can be fed with raw pixel data and that integrate feature extraction as part of the learning process.

The other method used in this paper, the Convolutional Network [14], is a multi-layer architecture trained in supervised mode using a gradient-based algorithm that minimizes a loss function. The convolutional net architecture is specifically designed for image processing, containing multiple alternated layers of trainable filters, point-wise non-linearities, and spatial subsampling. The overall function is non-linear in the parameters (the filter coefficients), and the loss function is non-convex. There is no guarantee that a global minimum will be found, but this does not seem to be a problem in practice. Empirically, the training time scales sub-linearly with dataset size. Convolutional net have been shown empirically to automatically learn salient image features and yield good recognition accuracy. However, no theoretical bounds exist, beyond the most generic ones that apply to all reasonable learning algorithms with finite capacity.

The contribution of this paper is two-fold. First, we test the SVM and convolutional net methods on the publicly available NORB dataset [15] which is intended as a benchmark dataset for large-scale invariant object categorization experiments. The dataset contains images of 50 objects belonging to 5 generic categories: human figures, four-legged animals, airplanes, trucks, and cars. Five instances of each category are used for training, and the remaining five for testing. Each object instance has 162 different views distributed on a viewing hemisphere, under 6 illuminations, and captured by a camera pair. A large dataset is available in which the objects are slapped on complex backgrounds, and randomly jittered for position, scale, in-plane rotation, contrast, and brightness. Previously reported results in the literature [15] have indicated convergence failures when training an SVM on the full dataset. The present paper reports successful results obtained after running systematic experiments with a highly-parallelized efficient SVM implementation [10]. Systematic experiments with convolutional nets on the same dataset were also performed.

The main point of the paper is to combine the advantages of convolutional nets and SVM. It is shown that the convolutional net can be trained to extract features that are relatively invariant to irrelevant variations of the input. Then it is shown that an SVM trained on feature vectors produced by the convolutional net can attain superior performance. It is easy to see that integrating a feature extraction process $c(\mathbf{x})$ with a regular kernel $K(c(\mathbf{x}), c(\mathbf{x}'))$ is equivalent to constructing a new kernel $K_c(\mathbf{x}, \mathbf{x}')$ on raw images. The Mercer condition over the new kernel K_c is automatically preserved. We show that this hybrid method yields significant performance improvement over the stand-alone systems on the large NORB dataset.

2 Previous Work

Object recognition is a very active research area. Many different approaches have been proposed over the years. The use of color, texture, and contours have been advocated in [16], the use of global appearance templates in [19, 18, 25], silhouettes and edge information [22, 29, 16, 5, 25], pose-invariant feature histograms [17, 6, 1], distinctive local features [26, 35, 33, 12], and image fragments [30].

Learning-based methods operating on raw pixels or low-level local features have been quite successful for such applications as face detection [31, 24, 33, 27]. SVM has been used first for OCR [7], and later for face detection [20], object recognition [23] on the COIL [19] dataset. In these reports, SVMs were used directly on raw pixels, while [6] used histogram features. More recent work concentrated on the use of more sophisticated kernel that use local features [11, 3, 34].

Several hierarchical, feed-forward multi-layer architectures have been proposed for invariant object recognition and detection, including convolutional nets [13, 31, 14, 15], hierarchies of features detectors based on image fragments [8], and variants of the HMAX architecture [28]. These architectures are all based on stacking multiple layers of the following modules [9, 13]: feature extraction through multiple convolutional filters, point-wise non-linearities, and spatial subsampling. These architectures are inspired by Hubel and Wiesel's classic simple cell/complex cell model of the early visual cortex. The spatial subsampling modules are designed to provide some level of shift and distortion invariance.

In [9], the filters are learned with an unsupervised method that produces sparse features, and the non-linearities are piecewise linear rectifications. In [28], the first layer is simply a set of fixed Gabor filters, and the non-linearity/subsampling is a max over local filter outputs. A large number of Gabor filters are necessary to cover the orientation/scale space. In [8], the filters are image fragments selected from training images using a mutual information criterion with the objects labels. Since the filters are relatively selective and class-specific, a large number of them is required. In convolutional nets [15], the features are all learned in supervised mode using gradient descent, and the non-linarities are sigmoid functions. The filter produced are quite broadly tuned, and a relatively small number of them is required for good performance (typically 8 at the first layer). The supervised, gradient-based optimization produces good results with far more compact networks than other methods, allowing for real-time applications.

3 SVM, Convolutional Nets, and Combining the two methods

In this section, we first briefly describe the SVM method and the Convolutional networks. The focus is to inspect their internal structures to provide insights into their respective strengths and weaknesses on the present vision task. This analysis will lead us to propose a method that combines the strengths of the two methods.

As we will show, SVMs can produce good decision surfaces if the input representation is reasonably well-behaved. But, with their fixed architecture, they cannot learn to extract relevant features so as to handle complicated invariances. Conversely, convolutional nets can learn invariant local features that are appropriate for image recognition, but the top layers seem to produce suboptimal classification surfaces.

3.1 SVM with Gaussian Kernel

SVMs with Gaussian kernels have two-layers. The first layer can be viewed as a set of *template matchers* that measure the similarity of the input pattern $\mathbf{x}$ to each of the training samples $\mathbf{x}_i$. The second layer computes the discriminant function as a linear combination of the similarity scores with learned weights α_i (many of which may be zero):

$$f(\mathbf{x}) = sgn(\sum_i y_i \alpha_i K(\mathbf{x}_i, \mathbf{x}) + b)$$

where the kernel function $K(\mathbf{x}_i, \mathbf{x})$ measures the similarity between the input pattern $\mathbf{x}$ and the training sample $\mathbf{x}_i$. The samples $\mathbf{x}_i$ for which the corresponding α_i are non-zero are the support vectors.

The supervised learning algorithm only affects the coefficients α_i. They are obtained by maximizing a margin criterion. In the dual formulation, the following quadratic loss function is minimized:

$$\mathcal{L}_D(\alpha) = \frac{1}{2}\alpha^T \mathbf{G} \alpha - \mathbf{1}^T \alpha$$

subject to the constraints $0 \le \alpha_i \le C, i = 1, \ldots, n$ and $\sum_{i=1}^n \alpha_i y_i = 0$.

$\mathbf{G}_{i,j} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. When the kernel function K satisfies the Mercer condition, the matrix $\mathbf{G}$ is positive semidefinite, and therefore the loss function is convex, which guarantees the existence of a single global minimum. For small problems (10^3 samples), this constrained quadratic optimization problem can be easily solved with standard methods like primal-dual interior-point methods with less than 100 lines of plain Matlab code. Unfortunately, the scaling properties of standard optimization algorithms in terms of memory usage and CPU time makes them impractical for large and complex datasets: the kernel matrix memory usage scales like $O(n^2)$, and the CPU time scales like $O(n^3)$ at best.

For large problems, the sparse nature of the learned coefficients α_i must be exploited. The sparsity comes from the fact that many data points are redundant, and can be discarded at early stages of the optimization procedure to make the algorithm efficient. Popular SVM algorithms, such as SMO and its variants [21], consider small batches of samples at a time (with two samples in the case of the original SMO), and use a caching mechanism to avoid storing the full kernel matrix.

The experiments reported here use the latest version of a fast implementation of SMO-type algorithm [10], running on a 32-node Linux cluster, with each node having dual AMD Athlon 1.5GHz CPUs and 2GB memory. The program automatically splits the training samples and distributes them to different nodes. The kernel matrix can be distributed over multiple nodes and fully cached in the cluster's large memory pool, to speed up the optimization process. The communication overhead is negligible (less than 5%).

3.2 Convolutional Network

Convolutional nets are multi-layer architectures where the successive layers are designed to learn progressively higher-level features, until the last layer which produces categories. All the layers are trained simultaneously to minimize an overall objective function. the feature extraction is therefore an integral part of the classification system, rather than a separate module, and is entirely trained from data, rather than designed.

However, once training is complete, one can view the last layer as a linear classifier operating on features extracted by the previous layers. Since those features are the result of the integrated training procedure, one could hope that they have been optimized to satisfy the requirements and limitations of the last layer.

The feature extraction front-end can be seen as composed of a stack of convolution and subsampling layers. The convolution (C-)layers compute convolutions over the previous layers $\mathbf{x}_{in}$ with some small trainable convolution kernels k:

$$\mathbf{x}_{out} = S(\sum_i \mathbf{x}_{in} \circledast k_i + b)$$

where S is a non-linear function (a hyperbolic tangent sigmoid), and b is a scalar bias. Depending on the values of the kernel coefficients k, the convolution operation can implement a local edge detector, a low-pass filter, or something entirely different. On each C-layer, multiple convolution kernels can be used, creating several different feature maps.

The spatial subsampling (S-)layers take the average of a $n \times n$ pixel block, multiply it by a trainable scalar β, add a bias, and pass the result through a sigmoid:

$$\mathbf{x}_{out} = S(\beta \sum \mathbf{x}_{in}^{n \times n} + b)$$

The result is a feature map of lower resolution where some position information about features has been eliminated, thereby building some level of distortion invariance in the representation.

Alternated layers of convolution and subsampling can extract features from increasingly large receptive fields, with increasing robustness to irrelevant variabilities of the inputs. The overall effect of these layers is to extract a feature vector v from the input $\mathbf{x}$, written as $v = c(\mathbf{x})$.

The last layer of a convolutional network can be seen as a linear classifier operating on the feature representation extracted by the previous layers. This last layer computes the product of the feature vector v with a weight matrix W, adds a bias vector, and passes the result through sigmoid functions. For training, the Euclidean distance between the output vector and a target output vector T^i is used as the loss function to be minimized:

$$\mathcal{L} = \|S(W.v + b) - T^i\|^2$$

where W is a trainable weight matrix of the last layer. T^i can be a traditional place code (one unit corresponding to

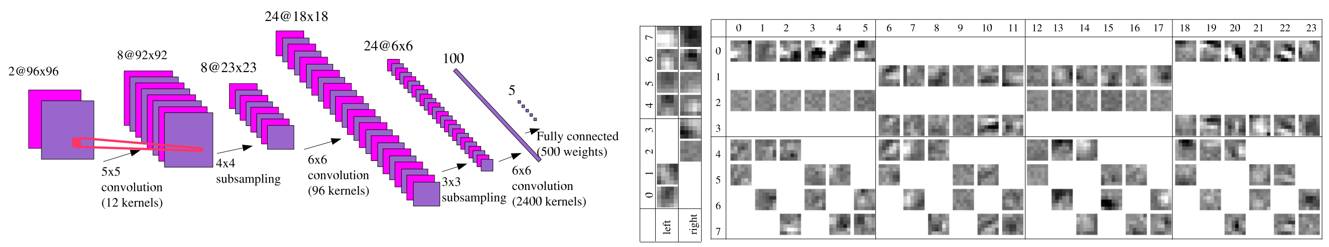
2@96x96
8@92x92
8@23x23
24@18x18
24@6x6
100
5
5x5 convolution (12 kernels)
4x4 subsampling
6x6 convolution (96 kernels)
3x3 subsampling
6x6 convolution (2400 kernels)
Fully connected (500 weights)
left
right

Figure 1. Left: architecture of the convolutional net. The input is an image pair, the system extracts 8 feature maps of size 92×92, 8 maps 23×23, 24 maps 18×18, 24 maps 6×6, and 100 features, followed by the 5 outputs. Middle: The learned convolution kernels of the C1 layer. The rows correspond to the feature maps, the columns are the input images. The first two maps takes input from the left image, the next two from the right, and others from both images. Right: The 96 learned kernels of the C3 layer. The columns correspond to the 24 feature maps output by C3, and the rows correspond to the 8 feature maps output by the S2 layer. Each feature map draw from 2 monocular maps and 2 binocular maps of S2.

the i-th element set to active, other units inactive), i being the class label of the input $\mathbf{x}$. The network is trained by minimizing $\mathcal{L}$. Gradient descent based algorithms can be used for the optimization since all the layers are differentiable.

A six-layer net, shown in figure 1, was used in the experiments reported here. The layers are respectively named C1, S2, C3, S4, C5, and output (numbers indicate the sequential position of the layers). The input of this example network is a pair of 96×96 gray scale images.

C1 uses 12 5×5 convolution kernels to generate 8 feature maps. As shown in figure 1, the first 2 maps take input from the left image, the next two from the right image, and the last 4 from both. There are 308 (300 on k_i and 8 on b) trainable parameters in this layer. The output of C1 is a 8×92×92 3-dimensional array. S2 is a 4×4 subsampling layer with 16 parameters, with output dimensions 8×23×23.

C3 uses 96 convolution kernels of size 6×6 to output 24 feature maps. Each C3 map takes input from 2 monocular maps and 2 binocular maps from S2, each with a different combination, as shown in figure 1. This configuration is used to combine features from the stereo image pairs. The C3 layer contains 3,480 trainable parameters, with output dimensions 24×18×18. S4 is a 3×3 subsampling layer which outputs feature maps of size 24×6×6.

C5 has a variable number of maps (80 and 100 in the reported results) that combine inputs from all map in S4 through 6×6 kernels. A C5 layer outputting 100 feature maps has 86,500 trainable parameters, about 95% of the whole system's parameter set. The output of the C5 layer is a 100-dimensional (or 80-dimensional) vector. This vector is the feature extracted by the sequence of C1 to C5 layers.

The output layer takes inputs from all C5 maps, transform them into a 5-dimensional vector, and compute the mean squared error with a set of target outputs (5-dimensional vectors that has one element corresponding to one class set to 1.5, all other elements set to -1.5). For 6-class detection/recognition tasks, we added a 6-th target output with all units set to -1.5 for the background class. For a C5 layer with 100-dimensional output, this layer has 505 free parameters. In this case, the whole network has a total of 90,857 trainable parameters.

To minimize the loss function a stochastic version of the Levenberg-Marquardt algorithm with diagonal approximation of the Hessian was used.

3.3 Combining Convolutional net with SVM

Architectures such as convolutional networks are quite good at learning invariant features, but not always optimal for classification (most of the trainable parameters are in the middle layers). On the other hand, Support Vector Machines with their fixed kernel function cannot learn complicated invariances, but produce good decision surfaces when applied to well-behaved feature vectors. It is interesting to investigate a hybrid system in which the convolutional net is trained to extract features that are relatively invariant to irrelevant variations of the input, so that an SVM with a simple Gaussian kernel can do a good job at separating the categories in the learned feature space.

This idea is appealing also because integrating the feature extraction process $c(\mathbf{x})$ with a regular kernel $K(c(\mathbf{x}), c(\mathbf{x}'))$ is equivalent to constructing a new kernel $K_c(\mathbf{x}, \mathbf{x}')$ on raw images. If K is a Mercer kernel, K_c also satisfies the Mercer condition. This can be shown by noting that the Mercer condition guarantees that the kernel function can be written as an inner product in a transformed space ϕ:

$$\begin{aligned} K(c(\mathbf{x}), c(\mathbf{y})) &= \langle \phi(c(\mathbf{x})), \phi(c(\mathbf{y})) \rangle \\ &= \langle \phi_c(\mathbf{x}), \phi_c(\mathbf{y}) \rangle \\ &= K_c(\mathbf{x}, \mathbf{y}) \end{aligned}$$

where ϕ_c is the composition of ϕ and c.

The equivalence is important since it means that a kernel K_c can be learned by cascading a learned feature extraction c with a fixed kernel function K.

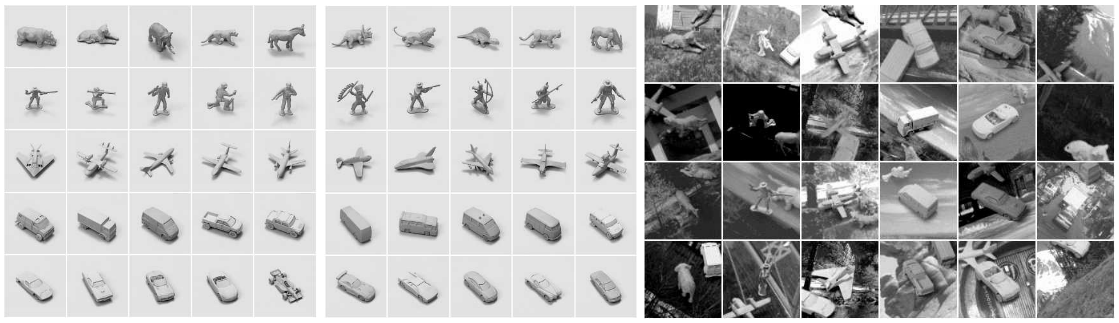

Figure 2. Left: The 25 objects used for training. Middle: the 25 objects used for testing. Each object is captured by a camera pair from 18 azimuth, 9 elevations and under 6 different illuminations. Right: a few of the 291,600 training examples from the *jittered-cluttered* set. Each column shows images from one category (including a background category).

In the following experiments, we use a convolutional net as the feature extractor c. The training and testing samples are fed through the trained network, and the output of the C5 layer are extracted as the features. The feature sets from the training samples are then used as input to train a Gaussian SVMs in the usual way. We show that this hybrid system not only improve the recognition performance significantly, but also are computationally very efficient.

Local feature extraction from images can be carried out in various ways. Popular strategies include using a bank of fixed, hardwired filters such as Gabor wavelets [28], or selecting patches or fragments extracted from training images [35, 30, 12]. Both approaches are quite expensive computationally, because they require a large number of filters to cover the space of possible inputs. Convolutional nets can get away with a small number of filters by training them from data in supervised mode.

4 Data Sets

The experiments in this paper used the NORB dataset [15] which is publicly available. This data set is a collection of images of 50 different toys, with 10 toys in each of the 5 generic categories: four-legged animals, human figures, airplanes, trucks, and cars. The 50 objects are split into a training set as shown in figure 2 and a testing set as in figure 2.

Each object is captured by a camera pair with 162 different views (9 elevations from 30° to 70° every 5°, 18 azimuths sampled every 20° along the horizontal viewing circle) and under 6 different illuminations.

Two datasets derived from NORB are used. The first dataset, called the *normalized-uniform* set, are images of a single object with a normalized size placed at the center of images with uniform background. The training set has 24,300 stereo image pairs of size 96×96, and another 24,300 for testing (from different object instances).

The second set, the *jittered-cluttered* set, contains objects with randomly perturbed positions, scales, in-plane rotation, brightness, and contrast. The objects are placed on highly cluttered backgrounds and other NORB objects placed on the periphery. A 6-th category of images is included: background images containing no objects. Some examples images of this set are shown in figure 2.

Each image is randomly perturbed so that the objects are at different positions ([-3, +3] pixels horizontally and vertically), and of different scales (ratio in [0.8, 1.1]), image-plane angles ($[-5^\circ, 5^\circ]$), brightness ([-20, 20] shifts of gray scale), contrast ([0.8, 1.3] gain). The objects were placed on randomly chosen natural scene images as backgrounds. A randomly chosen "distractor" object from the same set is placed at the periphery of the image. The central object could be occluded by the distractor.

To generate the training set, each image was perturbed with 10 different configurations of the above parameters, which makes up 291,600 image pairs of size 108×108. The testing set has 2 drawings of perturbations, and have 58,320 pairs.

In the NORB datasets, the only useful and reliable clue is the shape of the object, while all the other parameters that affect the appearance are subject to variation, or are designed to contain no useful clue. Parameters that are subject to variation are: viewing angles (pose), lighting conditions. Potential clues whose impact was eliminated include: color (all images are grayscale), and object texture. For specific object recognition tasks, the color and texture information may be helpful, but for generic recognition tasks the color and texture information are distractions rather than useful clues. By preserving natural variabilities and eliminating irrelevant clues and systematic biases, NORB can serve as a benchmark dataset in which no hidden regularity that would unfairly advantage some methods over others can be used.

5 Results and Discussions

Results with the three methods described on both the *normalized-uniform* set and the *jittered-cluttered* set are reported, with the focus on the later set which is more challenging and closer to real-world scenarios. Running times for testing and training are also reported to draw attention to the scalabilities of the methods.

5.1 Results on the *jittered-cluttered* set

The results on this set are shown in table 1. To classify the 6 categories, 6 binary ("one vs. others") SVM subclassifiers are trained independently, each with the full set of 291,600 samples. The training samples are raw 108×108 pixel image pairs cascaded into a 23,328-dimensional vector, with values between 0 to 255.

	SVM	Conv Net			SVM/Conv
test error	43.3%	16.38%	7.5%	7.2%	5.9%
train time (min*GHz)	10,944	420	2,100	5,880	330+
test time per sample (sec*GHz)	2.2	0.04			0.06+
#SV	5%				2%
parameters	σ=10^4 C=40	step size = 2×10^{-5} - 1×10^{-6}			dim=100 σ=5 C=1

Table 1. Testing error rates and training/testing timings on the *jittered-cluttered* dataset of different methods. The timing is normalized to hypothetical 1GHz single CPU. The convolutional nets have multiple results with different training passes due to its iterative training.

SVMs have relatively few free parameters to tune. In the case of Gaussian kernels, one can choose σ (Gaussian kernel sizes) and C (penalty coefficient) that yield best results by grid tuning. A rather disappointing test error rate of 43.3% is obtained on this set, as shown in the first column of table 1.

The training time depends heavily on the value of σ for Gaussian kernel SVMs. The experiments are run on a 64-CPU (1.5GHz) cluster, and the timing information is normalized into a hypothetical 1GHz single CPU to make the measurement meaningful. (The training time of the program is inversely proportional to the number of CPU's in the cluster). The testing time is per sample and in normalized seconds (for a single 1GHz CPU). The program is optimized for unsigned byte input vectors.

The convolutional net uses a online stochastic gradient descent for training, where the whole data set can be used for multiple passes to get progressively better results. We listed results after different number of passes (1, 5, 14) and their timing informations. The test error rate flattens out at 7.2% after about 10 passes. No significant over-training was observed, and no early stopping was performed. One parameter controlling the training procedure must be heuristically chosen: the global step size of the stochastic gradient procedure. Best results are obtained by adopting a schedule in which this step size is progressively decreased from 2×10^{-5} to 1×10^{-6}.

A full propagation of one data sample through the network requires about 4 million multiply-add operations. Parallelizing the convolutional net is relatively simple since multiple convolution can be performed simultaneously, and each convolutions can be performed independently on subregions of the layers. The experiments here are run on a single CPU (AMD Opteron at 2GHz) with 4GB of memory. Again, the training time is normalized to a hypothetical 1GHz CPU. The convolutional nets are computationally very efficient. The training time scales sublinearly with dataset size in practice, and the testing can be done in real-time with a few frames per second.

The third column shows the result with the hybrid system. The training and testing features are extracted with the convolutional net trained after 14 passes. The C5 layer of the network has 100 outputs, therefore the features are 100-dimensional, with values ranged between -1.7 and 1.7 (decided by the sigmoid function of the C5 layer).

The SVMs on features extracted from the convolutional net yields error rate of 5.9%, a significant improvement over either method alone. By incorporating a learned feature extractor into the kernel function, the SVM was indeed able to leverage both the ability to use low-level local features and at the same time keep all the advantages of a large margin classifier.

The hybrid system is also very efficient in the running times. Its training time includes 3 parts: the time to train a convolution net, the time to extract the features, and the time to train the SVM with the feature set. The time shown in the table is only the training time of the SVM with the feature sets. With the significant dimension reduction and the decrease of the percentage of support vectors, this training is 30 times faster than the pure SVM, and is a small portion of the convolutional net training time. The testing times include 2 parts: the time of a full propagation through the convolutional network, and the time of testing the trained SVMs on the extracted feature shown in the table, which is about 40 times faster than the pure SVM.

The poor performance of SVM with Gaussian kernels on raw pixel is not unexpected. With a Gaussian kernel, the template matching layer of the SVM is merely calculating Euclidean-distance-based similarities between the incoming pattern and the support vectors. The resulting similarity numbers are linearly combined to produce a score for a category. This architecture has previously been reported to obtain good results on both MNIST [14] and COIL [19] data sets. But those data sets contain images with little variation in registration and background, and so do not expose the SVMs' limitations. The MNIST data set has mostly black or white pixels, with size-normalized and registered isolated digit images on a white background. The COIL set's simplicity is demonstrated by the perfect recognition rate even with rather naive classification methods.

	SVM	Conv Net			SVM/Conv
test error	11.6%	10.4%	6.0%	6.2%	5.9%
train time (min*GHz)	480	64	448	3,200	50+
test time per sample (sec*GHz)	0.95	0.03			0.04+
#SV	28%				28%
parameters	σ=2,000 C=40	step size = 2×10^{-5} - 2×10^{-7}			dim=80 σ=5 C=0.01

Table 2. Testing error rates and training/testing timings on the *normalized-uniform* dataset of different methods. The timing is normalized to hypothetical 1GHz single CPU. The convolutional nets have multiple results with different training passes due to its iterative training.

5.2 Results on the *normalized-uniform* set

Table 2 shows the results on the smaller NORB dataset with uniform background. This dataset simulates a scenario in which objects can be perfectly segmented from the background, and is therefore rather unrealistic.

The experiments are done in the same way as on the full set. The SVMs for this set has 5 sub-classifiers, each trained on the 24,300 samples of 96×96 pixel image pairs. The convolutional net trained on this set has a smaller C5 layer with 80 outputs. The input features to the SVM of the hybrid system are accordingly 80-dimensional vectors.

The results of the convolutional net trained after 2, 14, 100 passes are listed in the table. The network is slightly overtrained with more than 30 passes. No regularization term was used in the experiment. The SVM in the hybrid system is trained over the features extracted from the network trained with 100 passes. The improvement of the combination is marginal over the convolutional net alone.

6 Conclusion and Outlook

The tremendous increase of the computing power has opened doors to large-scale data-driven learning methods with application to tasks such as computer vision. In this paper, convolutional nets and SVM are investigated with results on a generic object categorization dataset. to combine the advantages of the two methods we proposed a two-step learning process: first, train a convolutional net and view the first $N-1$ layers as a feature extractor c. Second, train an SVM on the features produced by the convolutional net. Based on experiments on the NORB data set, we have shown that this system yields a dramatic performance improvements over the SVM alone, and a significant performance improvement over the the convolutional net alone on the large *jittered-cluttered* NORB dataset. Experiments with the smaller *normalized-uniform* NORB dataset did not yield a significant improvement in accuracy over a plain convolutional network.

This suggests that large convolutional nets trained complex tasks may suffer from *undertraining*. It is possible that gradient-based optimization is not able to find the best use of the parameters. The results point to the fact that the *normalized-uniform* task is simple enough for the convolutional net to generate features in the penultimate layers in which the categories are linearly separable. The fact that the convolutional net trained on the large datasets can be improved by substituting an SVM in lieu of the last layer suggests that convolutional nets fall victim to *undertraining*.

With a trained object categorization system, tasks such as the detection of objects in large images or videos are a straightforward extension: the convolutional net can be replicated over the entire image at multiple scales. On modern mid-range processors (e.g. Pentium M 1.7GHz), convolutions can be performed at a rate of roughly 1.10^9 operations per second. This allowed us to build a real-time system for generic object detection and recognition that can run at several frames per second at full video resolution on a laptop.

While the scalability problems of SVMs with standard kernels is being addressed by numerous research efforts, including the use of online and active learning [4], their performance for invariant image recognition without prior feature extraction is limited.

Convolutional networks, on the other hand, yield good recognition accuracy with low computational complexity. Future work will attempt to build larger architectures so as to further improve the recognition accuracy, while taking advantage of the constant increase of available computational power.

Acknowledgments

We wish to thank Hans-Peter Graf and Eric Cosatto and their collaborators for letting us use their parallel implementation of SVM. We thank Leon Bottou for helpful discussions. This project was supported by NSF Grants No. 0535166 and No. 0325463.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Matching shapes. In *Proc. of ICCV*, IEEE, 2001.

[2] C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, 1998

[3] A. Barla, F. Odone, and A. Verri. "Hausdorff kernel for 3D object acquisition and detection," *ECCV*, 2002

[4] A. Bordes, S. Ertekin, J. Weston and L. Bottou Fast Kernel Classifiers with Online and Active Learning *Journal of Machine Learning Research*, vol. 6, 2005

[5] O. Carmichael, M. Hebert Object Recognition by a Cascade of Edge Probes. *Proc. British Mach. Vision Conf.*, 2002

[6] O. Chapelle, P. Haffner, and V. Vapnik, SVMs for Histogram-Based Image Classification, IEEE Trans. Neural Networks, 1999.

[7] C. Cortes, V. Vapnik, Support vector networks, *Machine Learning*, vol. 20, 1995

[8] B. Epshtein, S. Ullman, Feature Hierarchies for Object Classification *Proc. of ICCV*, Beijing, 2005.

[9] K. Fukushima, and S. Miyake, Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position *Pattern Recognition*, 15, pp 455-469, 1982.

[10] H. Graf, E. Cosatto, L. Bottou, I. Dourdanovic, V. Vapnik Parallel Support Vector Machines: The Cascade SVM *NIPS*, vol. 17, 2004

[11] K. Grauman and T. Darrell. The Pyramid Match Kernel: Discriminative Classification with Sets of Image Features. *Proc. of ICCV*, IEEE, 2005.

[12] S. Lazebnik, C. Schmid, and J. Ponce. Semi-Local Affine Parts for Object Recognition. *Proc. British Machine Vision Conference*, vol. 2, pp. 959-968, September 2004.

[13] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, L Jackel, Handwritten digit recognition with a back-propagation network in D. Touretzky (ed) *NIPS*, Vol. 2, Morgan Kaufman, 1990.

Gradient-Based Learning Applied to Document Recognition *Proceedings of the IEEE*, Nov 1998.

[14] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio. Gradient-Based Learning Applied to Document Recognition *Proceedings of the IEEE*, Nov 1998.

[15] Y. LeCun, F.-J. Huang, L. Bottou. Learning Methods for Generic Object Recognition with Invariance to Pose and Lighting. *Proc. CVPR*, 2004.

[16] J. Malik, S. Belongie, T. Leung, and J. Shi Contour and Texture Analysis for Image Segmentation. *Int. J. of Comp. Vision*, 2001.

[17] B. Mel SEEMORE:Combining color, shape, and texture histogramming in a neuraly-inspired approach to visual object recognition. *Neural Computation*, 9:777-804, 1997.

[18] B. Moghaddam, A. Pentland. Probabilistic Visual Learning for Object Detection. *Proc. of ICCV*, IEEE, June 1995.

[19] H. Murase and S. Nayar. Visual learning and recognition of 3D objects from appearance. *Int. J. of Comp. Vision*, 14(1):5–24, 1995.

[20] E. Osuna, R. Freund , F. Girosi. Training Support Vector Machines: an Application to Face Detection. *Proc. of CVPR*, Puerto Rico. IEEE, 1997.

[21] J. Platt Fast Training of Support Vector Machines using Sequential Minimal Optimization. *Advances in kernel methods: support vector learning*. MIT Press, 1999.

[22] J. Ponce, M. Cepeda, S. Pae, S. Sullivan. "Shape models and object recognition." In D.A. Forsyth et al., editor, *Shape, Contour and Grouping in Computer Vision*. Springer, 1999.

[23] M. Pontil, A. Verri. "Support Vector Machines for 3-D Object Recognition," *IEEE Trans. Patt. Anal. Machine Intell.* Vol. 20, 637-646, 1998.

[24] H.A. Rowley, S. Baluja, T. Kanade. Neural network-based face detection. *IEEE Trans. Patt. Anal. Mach. Intell.*, 20(1):23–38, January 1998.

[25] B. Leibe, and B. Schiele. "Analyzing Appearance and Contour Based Methods for Object Categorization.", *Proc. of CVPR*, IEEE, 2003.

[26] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Trans. Patt. Anal. Mach. Intell.*, 19(5):530–535, May 1997.

[27] H. Schneiderman and T. Kanade. A statistical method for 3d object detection applied to faces and cars. In *Proc. of CVPR*, IEEE, 2000.

[28] , T. L. Serre, L. Wolf and T. Poggio. Object Recognition with Features Inspired by Visual Cortex. *Proc. of CVPR*, IEEE, 2005.

[29] A. Selinger, R. Nelson. "Appearance-Based Object Recognition Using Multiple Views," *Proc. of CVPR*, IEEE, 2001.

[30] S. Ullman, M. Vidal-Naquet, and E. Sali. "Visual features of intermediate complexity and their use in classification", *Nature Neuroscience*, 5(7), 2002.

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

[32] V. Vapnik *Statistical Learning Theory*. John Wiley and sons, New York, 1998

[33] P. Viola, M. Jones. Rapid Object Detection using a Boosted Cascade of Simple Features. *Proc. of CVPR*, IEEE, 2001.

[34] C. Wallraven, B. Caputo, A. Graf, "Recognition with Local Features: the Kernel Recipe," *Proc. of ICCV*, IEEE, 2003

[35] M. Weber, M. Welling, and P. Perona. Towards automatic discovery of object categories. In *Proc. of CVPR*, IEEE 2000.