Convolutional Learning of Spatio-temporal Features

Graham W. Taylor, Rob Fergus, Yann LeCun, and Christoph Bregler

Courant Institute of Mathematical Sciences, New York University
New York, USA
{gwtaylor,fergus,yann,bregler}@cs.nyu.edu

Abstract. We address the problem of learning good features for understanding video data. We introduce a model that learns latent representations of image sequences from pairs of successive images. The convolutional architecture of our model allows it to scale to realistic image sizes whilst using a compact parametrization. In experiments on the NORB dataset, we show our model extracts latent "flow fields" which correspond to the transformation between the pair of input frames. We also use our model to extract low-level motion features in a multi-stage architecture for action recognition, demonstrating competitive performance on both the KTH and Hollywood2 datasets.

Key words: unsupervised learning, restricted Boltzmann machines, convolutional nets, optical flow, video analysis, activity recognition

1 Introduction

While the dominant methodology for visual recognition from images and video relies on hand-crafted features, there has been a growing interest in methods that learn low-level and mid-level features, either in supervised [1], unsupervised [2–4], or semi-supervised settings [5]. In recent years, feature-learning methods have focused on learning multiple layers of feature hierarchies to extract increasingly abstract representations at each stage. This has been generally done by composing modules of the same architecture such as Restricted Boltzmann Machines (RBM) [2], autoencoders [3], or various forms of encoder-decoder networks [4, 6, 7] each of which are trained unsupervised and therefore can take advantage of large amounts of unlabeled image data. The resulting "deep architectures" are then globally trained discriminatively, with the idea that the first phase of unsupervised feature learning has provided an initialization that is much more salient for high-level tasks than the usual random initialization.

Most of the above methods do not exploit the pictorial nature of the input, and have been applied to relatively small image patches (typically less than 64×64 pixels), because they do not scale well with the size of the input. This can be addressed by using a *convolutional architecture* [1], which exploits the fact that salient motifs can appear anywhere in the image. This idea has been recently used in the context of RBMs [8, 9]. By employing successive stages of weight-sharing and feature-pooling, deep convolutional architectures can achieve stable latent representations at each layer, that preserve locality, provide invariance to small variations of the input, and drastically reduce the number of free parameters.

To date, most of the work on unsupervised feature extraction has focused on static images but little attention has been given to learning about the way that images from videos change over time. The few works that address the problem (e.g. [10, 6]) are trained on isolated patches (not convolutionally), and suffer from the same limitations as static methods. In this paper, we propose a model that can extract motion-sensitive features from pairs of images (i.e. neighbouring frames of video). The features can capture both static and dynamic content. Our model is trained convolutionally which enables it to work on high-resolution images. We first apply it to synthetic data and show that it learns to represent flow-like features when the type of transformations are restricted. We then use it to extract useful features for human activity recognition in a multi-stage architecture that achieves state-of-the-art performance on the KTH actions dataset. Results are also shown on the challenging Hollywood2 action recognition dataset.

2 Related work

Our work extends the Gated RBM (GRBM) model proposed by Memisevic and Hinton [10]. The GRBM is able to extract distributed, domain-specific representations of image patch transformations. Due to its tensor parameterization, it is not practical to apply this model to patches larger than about $(N = 32) \times 32$ since the number of parameters grows as $O(N^4)$. Therefore, it has only been applied to low-resolution synthetic images of shifting pixels or PCA-reduced samples of low-resolution video. While the model has been shown to improve digit classification by learning the types of transformations to which the classifier should remain invariant, we are not aware of is application to a discriminative task on real video. Memisevic and Hinton have recently proposed a factored form of the GRBM [11] that drastically reduces the number of free parameters by replacing the three-way weight tensor with three low-rank matrices. In the present work, we take an alternative convolutional approach to scaling up the model, which achieves the additional benefit of translation invariance. Sutskever and Hinton [12] proposed a type of temporal RBM for video. Using synthetic videos of bouncing balls, they trained a model which was then able to generate similar videos, but did not apply their work to discriminative tasks. The signal from the past only provides a type of "temporal bias" to the hidden variables, which is fundamentally different from our third-order RBM, where past inputs modulate the interactions between the current input and the latent feature representation.

Building on the rapidly growing literature on sparse over-complete decompositions of image patches [13], Cadieu and Olshausen [6] have proposed a two-layer probabilistic model that learns complex motion features from video. In contrast to our model, they explicitly separate static amplitude and dynamic phase at the first layer. The second layer then learns high-order dependencies among the phase variables. Dean et al. [14] have recently proposed learning spatio-temporal descriptors by recursively applying the feature-sign sparse coding algorithm [15] to 3D patches of videos extracted at detected interest points. Like our work, their descriptors are adaptive, but their method is trained at the patch level.

State-of-the-art methods for activity recognition use engineered motion and texture descriptors extracted around interest points detected by spatio-temporal corner detectors. The descriptors are then vector-quantized, pooled over time and space into a "bag", and fed to an SVM classifier. Among the best performing methods are 1) Laptev et al.'s spatio-temporal interest points (STIP) [16] used in conjunction with the "HOG/HOF" descriptor that computes histograms of spatial gradients and optic flow accumulated in local space-time neighbourhoods [17]; 2) Dollar et al.'s "Cuboids" approach [18] used in conjunction with several different descriptor types; and 3) Willems et al.'s approach [19] which uses the determinant of the Hessian as a saliency measure and computes a weighted sum of Haar wavelet responses within local rectangular sub-volumes.

In contrast to these approaches, we perform a type of implicit, rather than explicit interest point detection and focus on learning descriptors rather than hand-crafting them. We also bypass the quantization step in favor of several additional layers of feature extraction that provide a distributed representation of each video. Jhuang et al. [20] propose an approach similar in spirit to ours, using multiple levels of feature detectors at increasing spatio-temporal scale. However, like [17–19], they forgo learning until the very last stage: low and mid-level features are engineered.

3 Unsupervised learning of spatio-temporal features

We first describe a related approach, the gated Restricted Boltzmann Machine, which models image patches but does not scale to realistic-sized images or video. We then describe our convolutional model.

3.1 The gated Restricted Boltzmann Machine (GRBM)

The gated Restricted Boltzmann Machine [10] differs from other conditional RBM architectures (e.g. [21, 12]) in that its inputs change the effective *weights* of the model instead of simply adjusting the effective *biases* of visible or latent variables (see Figure 1(left)). This is achieved by defining an energy function that captures third-order interactions among three types of binary stochastic variables: inputs, $\mathbf{x}$, outputs, $\mathbf{y}$, and latents, $\mathbf{z}$:

$$E\left(\mathbf{y}, \mathbf{z}; \mathbf{x}\right) = -\sum_{ijk} W_{ijk} x_i y_j z_k - \sum_k b_k z_k - \sum_j c_j y_j \tag{1}$$

where W_{ijk} are the components of a parameter tensor, $\mathbf{W}$, which is learned. To model affine and not just linear dependencies, biases $\mathbf{b}$ and $\mathbf{c}$ are included.

When learning from video, $\mathbf{x}$ and $\mathbf{y}$ are image patches (expressed as vectors) at identical spatial locations in sequential frames, and $\mathbf{z}$ is a latent representation

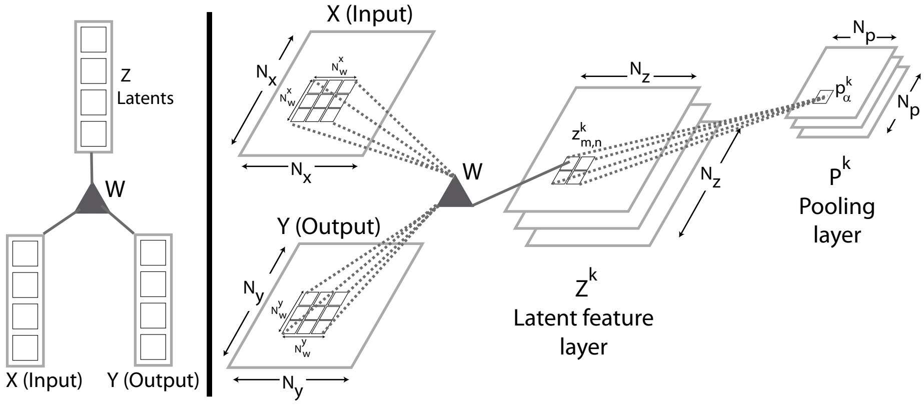

Fig. 1. Left: A gated RBM. Right: A convolutional gated RBM using probabilistic max-pooling.

of the transformation between $\mathbf{x}$ and $\mathbf{y}$. The energy of any joint configuration $\{\mathbf{y}, \mathbf{z}; \mathbf{x}\}$ is converted to a conditional probability by normalizing:

$$p(\mathbf{y}, \mathbf{z}|\mathbf{x}) = \exp\left(-E(\mathbf{y}, \mathbf{z}; \mathbf{x})\right)/Z(\mathbf{x}) \tag{2}$$

where the "partition function", $Z(\mathbf{x}) = \sum_{\mathbf{y},\mathbf{z}} \exp\left(-E(\mathbf{y}, \mathbf{z}; \mathbf{x})\right)$ is intractable to compute exactly since it involves a sum over all possible configurations of the output and latent variables. However, we do not need to compute this quantity to perform either inference or learning. Given an input-output pair of image patches, $\{\mathbf{x}, \mathbf{y}\}$, it follows from Eq. 1 and 2 that

$$p(z_k = 1|\mathbf{x}, \mathbf{y}) = \sigma(\sum_{ij} W_{ijk} x_i y_j + b_k) \tag{3}$$

where $\sigma(z) = 1/(1 + \exp(-z))$ is the logistic.

Maximizing the marginal conditional likelihood, $p(\mathbf{y}|\mathbf{x})$, over parameters $\theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ is difficult for all but the smallest models due to the intractability of computing Z. Learning, however, still works well if we approximately follow the gradient of another function called the contrastive divergence (CD) [22].

3.2 The convolutional gated Restricted Boltzmann Machine (convGRBM)

GRBMs represent the input and output as a vector, and thus ignore the pictorial structure of images. Weights that encode a particular local transformation must be re-learned to detect that same transformation at multiple locations. We now describe a form of GRBM that shares weights at all locations in an image. Inference is performed efficiently through convolution, so we refer to the model as a convolutional GRBM (convGRBM). The model is illustrated in Figure 1(right).

In our description of the GRBM, we suggested that $\mathbf{x}$ and $\mathbf{y}$ were time-adjacent patches from video, but they could have been any arbitrary vectors.

Here, we assume that $\mathbf{x}$ is a $N_x \times N_x$ binary image and $\mathbf{y}$ is a $N_y \times N_y$ binary image. We assume square, binary images to simplify the presentation but provide details of using real-valued images in the supplemental material. In the GRBM we had K binary latent variables. Now we have K $N_z \times N_z$ binary latent feature maps ($\mathbf{z} = \{\mathbf{z}^k\}_{k=1}^K$). Let m and n be spatial indices to each 2D feature map, such that a single feature is described as $z^k_{m,n}$. The indices m and n not only index a particular 2D feature, but they also define 1) an $N^y_w \times N^y_w$ local region in $\mathbf{y}$ from which this feature receives input, and 2) a $N^x_w \times N^x_w$ region of $\mathbf{x}$ which modulates the interaction between all K features at location m, n and the $N^y_w \times N^y_w$ local region in $\mathbf{y}$. Alternatively, we can think of each of the K features at index m, n as contributing a local log-linear patch model between the $N^x_w \times N^x_w$ pixels in $\mathbf{x}$ and the $N^y_w \times N^y_w$ pixels in $\mathbf{y}$ where the location of these local regions is specified by m, n. The number of local autoregressive models that can be "blended" is exponential in the number of feature maps.

For the remainder of our discussion, we will make two assumptions: 1) the input and output images are the same dimensions, $N_x = N_y$ (this holds true for neighbouring frames in video); and 2) the filter dimensions in the input and the output are the same, $N^x_w = N^y_w$. These assumptions are not necessary, but they greatly simplify bookkeeping and therefore the presentation that follows.

The convGRBM has the following energy function:

$$\begin{aligned} E(\mathbf{y}, \mathbf{z}; \mathbf{x}) = &- \sum_{k=1}^{K} \sum_{m,n=1}^{N_z} \sum_{r,s=1}^{N^y_w} z^k_{m,n} \gamma(\mathbf{x})^k_{r,s,m,n} y_{m+r-1,n+s-1} \\ &- \sum_{k=1}^{K} b_k \sum_{m,n=1}^{N_z} z^k_{m,n} - c \sum_{i,j=1}^{N_y} y_{i,j} \end{aligned} \tag{4}$$

where we use a per-map bias, b_k, for the latent variables and single output bias, c. Eq. 4 is similar to the energy function of a convolutional RBM [8], except that what was previously a filter weight with 3 indices: r, s, k has been replaced by a *conditional* filter weight, $\gamma(\mathbf{x})^k_{r,s,m,n} = \sum_{u,v}^{N^x_w} W^k_{r,s,u,v} x_{m+u-1,n+v-1}$, with 5 indices. The additional indices m, n denote the local region in $\mathbf{x}$ which modulates the filter. Note that while m, n index the entire feature map, u, v and r, s index within the local regions of $\mathbf{x}$ and $\mathbf{y}$, respectively.

As in the GRBM, the probability of jointly observing $\mathbf{y}$ and $\mathbf{z}$ given $\mathbf{x}$ is given by Eq. 2. The conditional distributions for $\mathbf{z}|\mathbf{y}, \mathbf{x}$ and $\mathbf{y}|\mathbf{z}, \mathbf{x}$ naturally follow:

$$p(z^k_{m,n} = 1 | \mathbf{x}, \mathbf{y}) = \sigma(\sum_{r,s=1}^{N^y_w} \gamma(\mathbf{x})^k_{r,s,m,n} y_{m+r-1,n+s-1} + b_k) \tag{5}$$

$$p(y_{i,j} = 1 | \mathbf{x}, \mathbf{z}) = \sigma(\sum_{k=1}^{K} \sum_{r,s=1}^{N^y_w} \hat{\gamma}(\mathbf{x})^k_{r',s',i+r-1,j+s-1} \hat{z}^k_{i+r-1,j+s-1} + c) \tag{6}$$

where $r' = N^y_w - r + 1$ and $s' = N^y_w - s + 1$ represent a "flipping" of the filter indices (i.e. correlation rather than convolution), and $\hat{\mathbf{z}}$ is the result of zero-padding $\mathbf{z}$ such that its first $N^y_w - 1$ rows and columns are zero. Note that in

Eq. 5 an output unit $y_{i,j}$ makes a bottom-up contribution to several elements (m, n) in all K feature maps. Therefore, in top-down reconstruction (Eq. 6) we must ensure that each output unit receives input from all feature map elements to which it has contributed, through the same conditional filter weight that was used bottom-up. To account for border effects, it is convenient to define $\hat{\gamma}(\mathbf{x})$ as a zero-padded version of $\gamma(\mathbf{x})$ whose dimensions are $N_w^y \times N_w^y \times N_y \times N_y \times K$.

As with convolutional RBMs, we can express both inference (Eq. 5) and reconstruction (Eq. 6) in terms of convolution operations (see the supplemental material for details). While inference in a convolutional RBM requires a single 2D convolution of the data with the filters, inference in the convGRBM requires a 2D convolution of the output and data for each element of the conditioning window: i.e. $N_w^x \times N_w^x$ convolutions. The same holds true for reconstruction (replacing data with feature maps). Note, however, that a fully-connected (i.e. non-convolutional) GRBM requires $N_x \times N_x$ more operations during inference than a standard RBM. Restricting connections to be local clearly makes a huge difference in efficiency, especially when the ratio of pixels to filter size is high.

Probabilistic max pooling Most object recognition systems use a pooling operation that combines nearby values in input or feature space through a max, average or histogram operator. This provides the system with some invariance to small local distortions and reduces the computational burden. Traditional pooling layers, however, are designed for feed-forward architectures like convolutional nets and do not support generative models such as RBMs that include top-down feedback. Lee et al. [8] thus introduced probabilistic max-pooling in the context of convolutional RBMs. We adopt their approach, and summarize it here.

Recall that we have K feature maps connected to the visible input and output. We introduce a layer on top of the feature maps, called the pooling layer, which also has K maps, each connected 1-1 to a feature map. However, the maps of the pooling layer are reduced in spatial resolution by a constant factor C in each dimension (e.g. 2 or 4). More precisely, each feature map $\mathbf{z}^k$ is partitioned into non-overlapping $C \times C$ blocks, and each block is connected to exactly one binary unit, p_α^k, in the pooling layer (i.e. $N_p = N_z/C$). Here, we have adopted the notation of [8] where α indexes the pooling units and also define a block formally as $B_\alpha \triangleq \{(m, n) : z_{m,n} \text{ belongs to the block } \alpha\}$.

The connection between pooling unit p_α and the features in block B_α is constrained such that *at most* one of the features in a block is on, and if any of the features in block B_α is on, then p_α must be on, otherwise p_α is off. This leads to a modified, constrained, energy function:

$$E(\mathbf{y}, \mathbf{z}; \mathbf{x}) = -\sum_{k=1}^{K} \sum_{\alpha} \sum_{(m,n) \in B_\alpha} \sum_{r,s=1}^{N_w^y} z_{m,n}^k \gamma(\mathbf{x})_{r,s,m,n}^k y_{m+r-1,n+s-1}$$
$$- \sum_{k=1}^{K} b_k \sum_{m,n=1}^{N_z} z_{m,n}^k - c \sum_{i,j=1}^{N_y} y_{i,j} \quad \text{subject to:} \sum_{(m,n) \in B_\alpha} z_{m,n}^k \leqslant 1, \forall k, \alpha. \qquad (7)$$

Changing the energy function results in a change to the inference procedure. Note that each unit in feature map k receives the following bottom-up signal from the input and output:

$$I(z^k_{m,n}) \triangleq \sum_{r,s=1}^{N^y_w} \gamma(\mathbf{x})^k_{r,s,m,n} y_{m+r-1,n+s-1} + b_k. \tag{8}$$

Due to the factorial form of Eq. 7, we can sample each of the blocks independently as a multinomial function of their inputs:

$$p(z^k_{m,n} = 1|\mathbf{x}, \mathbf{y}) = \Omega^{-1} \exp\left(I(z^k_{m,n})\right), \qquad p(p^k_\alpha = 0|\mathbf{x}, \mathbf{y}) = \Omega^{-1} \tag{9}$$

where the normalization constant is $\Omega = 1 + \sum_{(m',n') \in B_\alpha} \exp\left(I(z^k_{m',n'})\right)$.

4 Experiments on synthetic data: NORB

One way to evaluate third-order RBMs is by experimenting in a domain where optical flow is controlled and regular (e.g. the "shifting pixels" experiments of [10]). In this section, we describe a domain for experimentation that is of increased complexity yet still controlled. The "Small NORB" dataset [23] has 5 object categories (humans, airplanes, cards, trucks, animals), and 5 different object instances for each training and test. Each object instance has 18 azimuths, 9 camera-elevations, and 6 illuminations, for a total of 24300 training samples and 24300 test samples. Traditionally NORB has been used to evaluate object recognition. Since our goal is to extract useful "transformation" features from pairs of images we use the dataset differently than intended.

The azimuth, elevation, and illumination changes in the NORB dataset are at fixed intervals and corresponding labels for each image are available. Therefore, we created synthetic "videos" where an object underwent forward or reverse transformation in one of the dimensions while the others were held fixed. Before generating the videos, we downsampled each image to 32×32 pixels, and preprocessed it using local contrast normalization (LCN) as described in [24]. The LCN operation involves a 9×9 smoothing filter, so each resulting image is 24×24.

We then trained a convGRBM with real-valued outputs and 20 binary feature maps. The filter dimensions were $N^x_w = N^y_w = 9$. The model trained on all azimuth changes of $\pm 20°$, and all camera elevation changes of $\pm 10°$. It was trained for 30 complete passes through the training set, using standard CD(1) learning. Figure 2 shows the result of performing 10 "image analogies". Each analogy is represented by a group of six small greyscale images and one larger "optical flow' image. To perform an analogy, the model is presented with a pair of images each from an object instance it has never seen before, and asked to apply the same inferred transformation to a random target image, also which it has never seen before. We can also visualize the "flow" implicit in the hidden units and conditional on the pair, by drawing, for each input pixel, an arrow to the output pixel to which it is most strongly connected according to the learned filters, W (marginalized over the binary feature maps). Much information is potentially lost in this representation [10]: the transformation encoded by the feature maps can be much richer than what is expressed by optical flow alone.

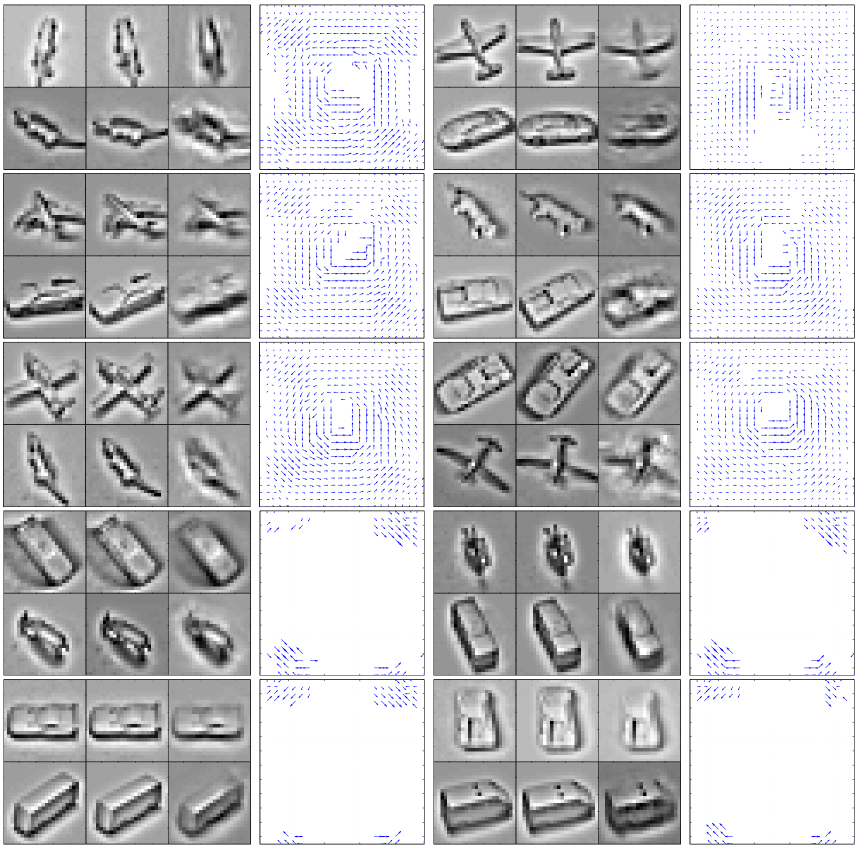

Fig. 2. Analogies. Each group of six greyscale images from left to right, top to bottom represent: input image; output image; model's reconstruction of output; random target image; ground truth of random target (i.e. by searching for the example that corresponds to the transformation between image and output); inferred transformation applied to targets. Examples 1-6 show changes in azimuth; 7-10 show changes in camera elevation. A representation of inferred "max" flow fields is shown for each example.

5 Experiments on human activity recognition

Recognition of human activity from video is a challenging problem that has received an increasing amount of attention from the computer vision community in recent years. The ability to parse high-level visual information has wide-ranging applications that include surveillance and security, the aid of people with special needs and the understanding and interpretation of non-verbal communication.

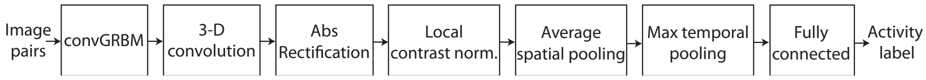

Fig. 3. An overview of our multi-stage architecture for human activity recognition. See text for a description of each stage.

We approach the problem with a multi-stage architecture (see Figure 3) that combines convolutional and fully-connected layers. At the lowest layer, a convolutional GRBM extracts features from every successive pair of frames. We observe that most features are motion-sensitive, but others capture static information. This is particularly useful in providing context in more challenging datasets [25] and will aid in applying our method to other tasks, such as scene recognition from video. A subset of the feature maps inferred from the KTH actions dataset are shown in Figure 4. The features are extremely diverse: many capture limb movement, others capture edge content, and one seems particularly apt at segmenting person from background (we note that the background is generally uniformly textured in KTH).

To capture mid-level spatio-temporal cues, we apply a traditional (i.e. feed-forward) convolutional layer that uses 3D spatio-temporal filters. A connectivity table indicates which of the 3D convolutional layer output maps are connected to each convGRBM pooling map. Our convolutional layer is a 3D extension of the architecture advocated by Jarrett et al. [7]: filtering, followed by a tanh nonlinearity, followed by absolute value rectification, followed by a local contrast normalization layer, followed by average pooling and subsampling. Both the abs(·) and tanh(·) are performed element-wise, so their extension to 3D is straightforward. The LCN and pooling/subsampling layers each employ a filtering operation, which we perform in 3D instead of 2D.

The output of the second convolutional layer is a series of 3D feature maps. To cope with variable-length sequences, we perform an additional max pooling in the temporal dimension. This ensures that the mid-level features can be reduced to a vector of consistent size. This representation is followed by one or more fully-connected layers (we use 1 or 2 in our experiments). The topmost layer is a softmax (multinomial) layer corresponding to discrete activity labels, and intermediate layers use a tanh nonlinearity. The convGRBM is trained unsupervised using CD, while the upper layers are trained by backpropagation. We do not backpropagate through the first layer following unsupervised training, though this could be done to make the low-level features more discriminative.

5.1 KTH actions dataset

The KTH actions dataset [26] is the most commonly used dataset in evaluating human action recognition. It consists of 25 subjects performing six actions: walking, jogging, running, boxing, hand waving, and hand clapping under 4 scenarios (outdoors, outdoors with scale variation, outdoors with different clothes and indoors). Each sequence is further divided into shorter "clips" for a total of 2391 sequences. We use the original evaluation methodology: assigning 8 subjects to a training set, 8 to a validation set, and the remaining 9 subjects to a test set so that our results are directly comparable to the recent survey by Wang et al. [27].

Preprocessing We maintained the original frame rate (25fps) and spatial resolution 160×120 in all of our experiments. All videos then underwent 3D local contrast normalization (an extension of [24]).

Unsupervised Learning. We trained a convGRBM with $N_z = 32$ feature maps and a pooling factor of $C = 4$. Filter sizes were $N_x^x = N_x^y = 16$. We chose 16 as it was a number amenable to GPU-based computing, and it was close to the minimal patch size (18×18) suggested by Wang et al. [27]. We have not tried other patch sizes. Weights were updated in "mini-batches" of 128 pairs of subsequent frames (the order of pairs was randomly permuted as to balance the mini-batches). We made 30 complete passes over all videos in the training set.

Supervised Learning We trained a convolutional net with 128 $9 \times 9 \times 9$ filters (randomly initialized) on top of the features extracted by the convGRBM. Each feature map of the convolutional net received input from 4 randomly chosen pooling maps from the first layer. Architectural choices were motivated by a desire to extract mid-level spatio-temporal features; the local connectivity used is standard practice [1]. The nonlinearities we used were identical to those in [7] with the exception of extending contrast normalization and downsampling to 3D: LCN was performed using a 9×9×9 smoothing filter, followed by 4×4×4 average downsampling. We also tried a more traditional network architecture which did not use absolute value rectification and LCN. We found that it slightly decreased accuracy (by about 1%; less drastic than reported in [7] for static object recognition). The pooling layer was then subjected to a further max-pooling over time, the output was vectorized and connected to one or two fully-connected layers. All layers (except the convGRBM) used online backpropagation[1]. We made 30 complete passes through the training set.

Table 1 compares our approach to the prior art using dense sampling (i.e. no interest-point detection) and K-means quantization. We report mean accuracy over all six actions. Our method, to the best of our knowledge, gives the best mean accuracy on KTH amongst methods that do not use interest-point detection. The currently best performing method [17] uses the STIP interest-point detector and HOG/HOF or HOF descriptors (91.8 and 92.1%, respectively). Due to the high ratio of background pixels to subject pixels in KTH, and the limited number of actions (that don't require context information), interest-point methods tend to perform extremely well on KTH. Evidence already indicates that dense-sampling outperforms interest-points on more challenging datasets [27].

To demonstrate the advantage of low-level feature extraction with convGRBMs, we have replaced the first layer with a standard 3D convolutional layer ($32\mathrm{F}_{\mathrm{CSG}}^{16\times16\times2}$

[1] The choice of using online learning here was simply a matter of convenience due to variable sequence lengths. Since the convGRBM is trained on pairs of frames (rather than whole sequences) it is easier to train in mini-batches.

Table 1. KTH action dataset: classification performance using dense sampling. Integers preceding a module indicate the number of feature maps in that module. Superscripts indicate filter sizes or downsampling ratio (chosen by context). convGRBM is our proposed method, trained unsupervised. F_{CSG} is a standard convolutional layer: a set of convolution filters (C) followed by a sigmoid/tanh nonlinearity (S), and gain coefficients (G). R/N/P_A is abs rectification, followed by local contrast normalization, followed by average pooling. The number of fully-connected layers are either 1 which corresponds to logistic regression (log_reg) or 2, which corresponds to a multi-layer perceptron (mlp).

Prior Art	Accuracy	Convolutional architectures	Accuracy
HOG3D-KM-SVM	85.3	32convGRBM$^{16\times16}$-128$F_{CSG}^{9\times9\times9}$-R/N/$P_A^{4\times4\times4}$-log_reg	88.9
HOG/HOF-KM-SVM	86.1	32convGRBM$^{16\times16}$-128$F_{CSG}^{9\times9\times9}$-R/N/$P_A^{4\times4\times4}$-mlp	**90.0**
HOG-KM-SVM	79.0	32$F_{CSG}^{16\times16\times2}$-R/N/$P_A^{4\times4\times4}-128F_{CSG}^{9\times9\times9}$-R/N/$P_A^{4\times4\times4}$-log_reg	79.4
HOF-KM-SVM	88.0	32$F_{CSG}^{16\times16\times2}$-R/N/$P_A^{4\times4\times4}-128F_{CSG}^{9\times9\times9}$-R/N/$P_A^{4\times4\times4}$-mlp	79.5

- see Table 1). By using filters of size $16\times16\times2$ and a $4\times4\times4$ pooling layer, we have matched the architecture of the convGRBM as best as possible to perform this comparison. The entire network is trained by backpropagation. We note that this fully feed-forward approach performs considerably worse.

5.2 Hollywood2 dataset

The Hollywood2 dataset [25] consists of a collection of video clips containing 12 classes of human action extracted from 69 movies. It totals approximately 20.1 hours of video and contains approximately 150 samples per action. It provides a more realistic and challenging environment for human action recognition by containing varying spatial resolution, camera zoom, scene cuts and compression artifacts.

Table 2. Hollywood2 dataset: average precision (AP) using dense sampling.

Method	AP
Prior Art [27]:	
HOG3D+KM+SVM	45.3
HOG/HOF+KM+SVM	**47.4**
HOG+KM+SVM	39.4
HOF+KM+SVM	45.5
convGRBM+SC+SVM	**46.6**

Performance is evaluated as suggested by Marszalek et al. [25]: by computing the average precision (AP) for each of the action classes and reporting mean AP over all actions. Following [27], we downsampled the spatial resolution of every video clip (which varies between clips) by a factor of 2. Videos were then zero-padded to have a constant spatial resolution. We did no temporal downsampling. All videos then underwent 3D local contrast normalization.

Similar to the KTH dataset, we trained a convGRBM with $N_z = 32$ feature maps and a pooling factor of $C = 4$. Filter sizes were $N_x^x = N_x^y = 16$. The convGRBM was trained for 50 complete passes over all videos in the training dataset and used a sparsity regularization term in the CD updates [28] that encouraged the hidden units to have a mean activation of 0.1.

Instead of applying a convolutional network to extract mid-level features, we sampled the feature maps of the convGRBM with a stride of 8 pixels in each direction, and formed local temporal groups of 10 frames. We then used the method described in [29] to learn a dictionary of 4000 basis vectors, and encode the temporal groups as sparse linear coefficients of the bases. Each video then yielded a varying number of sparse vectors (given different lengths) so we applied max-pooling over the temporal dimension. A SVM (with RBF kernel) was then trained (per-activity) on the top-level representation. Since Hollywood2 videos may contain more than one activity, this approach allowed us to avoid training a separate 3D convolutional net per-activity.

We achieve a mean AP of 46.6% using dense sampling, learned convGRBM low-level features and sparse coding with 4000 elements. To the best of our knowledge, the only superior published result is 47.4% which uses dense sampling with HOG/HOF features and quantization [27]. However, our result outperforms other popular methods such as Cuboids (45.0%) and Willems et al. (38.2%) (published in [27]). We also expect that an approach that combined our learned features with HOG/HOF descriptors could perform well.

6 Conclusion

Gated RBMs extract latent representations that are useful for video understanding tasks. However, they do not scale well to realistic resolutions and must learn separate feature detectors at all locations in a frame. In the spirit of recent work exploring convolutional deep architectures, we have introduced the convolutional gated RBM. We showed that it learned to represent optical flow and performed image analogies in a controlled, synthetic environment. In a more challenging setting, human activity recognition, it extracted useful motion-sensitive features, as well as segmentation and edge-detection operators that allowed it to perform competitively against the state-of-the-art as part of a multi-stage architecture.

Acknowledgments. We thank the Office Of Naval Research (ONR N000140910789, ONR N000140910076), and Google for supporting this research.

References

1. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86** (1998) 2278–2324
2. Hinton, G., Osindero, S., Teh, Y.: A fast learning algorithm for deep belief nets. Neural Comput **18** (2006) 1527–1554
3. Larochelle, H., Erhan, D., Courville, A., Bergstra, J., Bengio, Y.: An empirical evaluation of deep architectures on problems with many factors of variation. In: ICML. (2007) 473–480
4. Ranzato, M., Poultney, C., Chopra, S., LeCun, Y.: Efficient learning of sparse representations with an energy-based model. In: NIPS. (2006) 1137–1144
5. Nair, V., Hinton, G.: 3D object recognition with deep belief nets. In: NIPS. (2009) 1339–1347

6. Cadieu, C., Olshausen, B.: Learning transformational invariants from natural movies. In: NIPS. (2009) 209–216
7. Jarrett, K., Kavukcuoglu, K., Ranzato, M., LeCun, Y.: What is the best multi-stage architecture for object recognition? In: ICCV. (2009) 2146–2153
8. Lee, H., Grosse, R., Ranganath, R., Ng, A.Y.: Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In: ICML. (2009) 609–616
9. Norouzi, M., Ranjbar, M., Mori, G.: Stacks of convolutional restricted Boltzmann machines for shift-invariant feature learning. In: CVPR. (2009)
10. Memisevic, R., Hinton, G.: Unsupervised learning of image transformations. In: CVPR. (2007)
11. Memisevic, R., Hinton, G.: Learning to represent spatial transformations with factored higher-order Boltzmann machines. Neural Comput (2010)
12. Sutskever, I., Hinton, G.: Learning multilevel distributed representations for high-dimensional sequences. In: AISTATS. (2007)
13. Olshausen, B., Field, D.: Sparse coding with an overcomplete basis set: A strategy employed by V1? Vision Res **37** (1997) 3311–3325
14. Dean, T., Corrado, G., Washington, R.: Recursive sparse spatiotemporal coding. In: Proc. IEEE Int. Workshop on Mult. Inf. Proc. and Retr. (2009)
15. Lee, H., Battle, A., Raina, R., Ng, A.Y.: Efficient sparse coding algorithms. In: NIPS. (2007) 801–808
16. Laptev, I., Lindeberg, T.: Space-time interest points. In: ICCV. (2003) 432–439
17. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: CVPR. (2008)
18. Dollár, P., Rabaud, V., Cottrell, G., Belongie, S.: Behavior recognition via sparse spatio-temporal features. In: VS-PETS. (2005)
19. Willems, G., Tuytelaars, T., Gool, L.: An efficient dense and scale-invariant spatio-temporal interest point detector. In: ECCV. (2008) 650–663
20. Jhuang, H., Serre, T., Wolf, L., Poggio, T.: A biologically inspired system for action recognition. In: ICCV. (2007)
21. He, X., Zemel, R., Carreira-Perpiñán, M.: Multiscale conditional random fields for image labeling. In: CVPR. (2004) 695–702
22. Hinton, G.: Training products of experts by minimizing contrastive divergence. Neural Comput **14** (2002) 1771–1800
23. LeCun, Y., Huang, F., Bottou, L.: Learning methods for generic object recognition with invariance to pose and lighting. In: CVPR. (2004)
24. Pinto, N., Cox, D., DiCarlo, J.: Why is real-world visual object recognition hard? PLoS Comput Biol **4** (2008)
25. Marszalek, M., Laptev, I., Schmid, C.: Actions in context. In: CVPR. (2009) 2929–2936
26. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local SVM approach. In: ICPR. (2004) 32–36
27. Wang, H., Ullah, M., Kläser, A., Laptev, I., Schmid, C.: Evaluation of local spatio-temporal features for action recognition. In: BMVC. (2009) 127–138
28. Lee, H., Ekanadham, C., Ng., A.: Sparse deep belief net model for visual area V2. In: NIPS. (2008) 873–880
29. Mairal, J., Bach, F., Ponce, J., Sapiro, G.: Online dictionary learning for sparse coding. In: ICML. (2009) 689–696
30. Freund, Y., Haussler, D.: Unsupervised learning of distributions of binary vectors using 2-layer networks. In: Proc. NIPS 4. (1992)

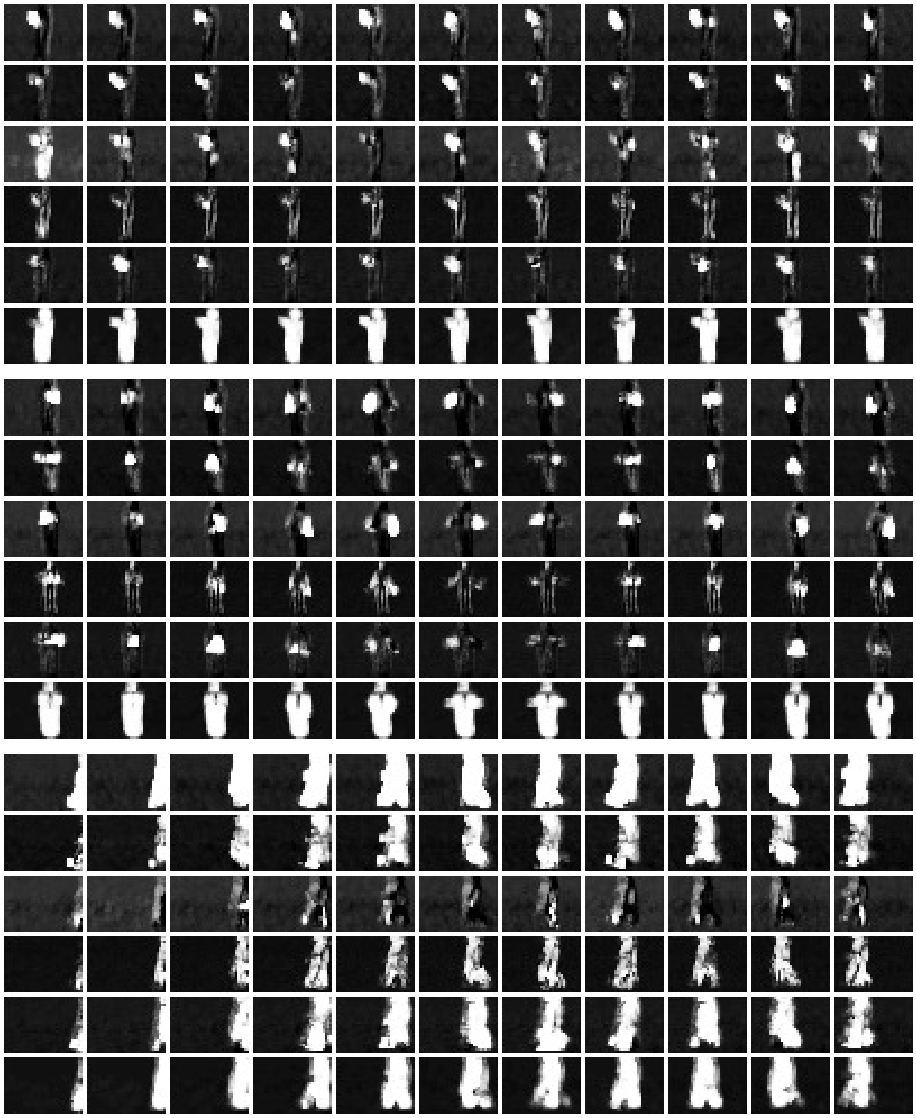

Fig. 4. Feature maps inferred from the KTH actions dataset. A subset of 6 (4×4 max-pooled) feature maps (of 32 total) inferred from sequences of the same subject performing different activities: boxing (rows 1-6), hand-clapping (rows 7-12) and walking (rows 13-18). Rows correspond to features, columns correspond to frames. We show person 1, scenario 1 and sequence 1. We display real-valued probabilities of activation rather than stochastic choices. We also downsample the frame rate by a factor of 4 for display. From the hand-clapping example, we see that features 1 and 3 are sensitive to motion in opposite directions (note how features 1 and 3 localize opposite hands), feature 4 seems to be sensitive to edges, and feature 6 learns to segment the subject from the background. Remaining activities are shown in the supplemental material.