Learning to encode motion using spatio-temporal synchrony

Kishore Konda KONDA@INFORMATIK.UNI-FRANKFURT.DE
Goethe University Frankfurt, Frankfurt

Roland Memisevic ROLAND.MEMISEVIC@UMONTREAL.CA
University of Montreal, Montreal

Vincent Michalski VMICHALS@RZ.UNI-FRANKFURT.DE
Goethe University Frankfurt, Frankfurt

Abstract

We consider the task of learning to extract motion from videos. To this end, we show that the detection of spatial transformations can be viewed as the detection of synchrony between the image sequence and a sequence of features undergoing the motion we wish to detect. We show that learning about synchrony is possible using very fast, local learning rules, by introducing multiplicative "gating" interactions between hidden units across frames. This makes it possible to achieve competitive performance in a wide variety of motion estimation tasks, using a small fraction of the time required to learn features, and to outperform hand-crafted spatio-temporal features by a large margin. We also show how learning about synchrony can be viewed as performing greedy parameter estimation in the well-known motion energy model.

1. Introduction

The classic motion energy model turns the frames of a video into a representation of motion by summing over squares of Gabor filter responses (Adelson & Bergen, 1985; Watson & Albert J. Ahumada, 1985). One of the motivations for this computation is the fact that sums over squared filter responses allow us to detect "oriented" energies in spatio-temporal frequency bands. This, in turn, makes it possible to encode motion independently of phase information, and thus to represent motion to some degree independent of what is moving. Related models have been proposed for binocular disparity estimation (e.g. Fleet et al., 1996), which also involves the estimation of the displacement of local features across multiple views.

For many years, hand-crafted, Gabor-like filters have been used (see, e.g., Derpanis, 2012), but in recent years, unsupervised deep learning techniques have become popular which learn the features from videos (e.g. Taylor et al., 2010; Le et al., 2011; Ji et al., 2013; Memisevic & Hinton, 2007), The interest in learning-based models of motion is fueled in part by the observation that for activity recognition, hand-crafted features tend to not perform uniformly well across tasks (Wang et al., 2009), which suggests learning the features instead of designing them by hand.

In this work, we show that motion energy models may be thought of as combining two independent contributions to motion encoding, namely the detection of temporal "synchrony", and the encoding of invariance. We show how disentangling these two contributions provides a different perspective onto the energy model and suggests new approaches to learning. In particular, we show that local learning rules in combination with multiplicative "gating" interactions (e.g. Mel, 1994), are all that is required to learn energy models that can compete with the state-of-the-art in activity recognition. We also show that competitive motion features can be learned on conventional CPU-based hardware and in a small fraction of the time required by previous methods.

2. Motion from spatio-temporal synchrony

Consider the task of computing a representation of motion, given two frames $\vec{x}_1$ and $\vec{x}_2$ of a video. The classic energy model (Adelson & Bergen, 1985) solves this task by detecting subspace energy. This amounts to computing the sum of squared quadrature Fourier or Gabor coefficients across multiple frequencies and orientations (e.g. Hyvarinen et al., 2009). The motivation behind the energy model is that Fourier amplitudes are independent of stimulus phase, so they yield a representation of motion that is to some degree independent of image content. As we shall show below, this view confounds two independent contributions of the energy model, which may be disentangled in practice.

An alternative to computing the sum over *squares*, which has originally been proposed for stereopsis, is the cross-correlation model (Arndt et al., 1995; Fleet et al., 1996), which computes the sum over *products* of filter-responses across the two frames. It can be shown that the sum over products of filter responses in quadrature encodes angles in the invariant subspaces associated with the transformation. The representation of angles thereby also yields a phase-invariant representation of motion (e.g. Fleet et al., 1996; Cadieu & Olshausen, 2011; Memisevic, 2012). Like the energy model, it also confounds invariance and representing transformations as we shall show.

It can be shown that cross-correlation models and energy models are closely related, and that there is a canonical operation that turns one into the other (e.g. Fleet et al., 1996; Memisevic, 2012). We shall revisit the close relationship between these models in Section 3.3.

2.1. Motion estimation by synchrony detection

We shall now discuss how synchrony detection allows us to compute motion, and how content-invariance can be achieved by pooling afterwards, if desired. To this end, consider two filters $\vec{w}_1$ and $\vec{w}_2$ which shall encode the transformation between two images $\vec{x}_1$ and $\vec{x}_2$. We restrict our attention to transformations that can be represented as an orthogonal transformation in "pixel space", in other words, as an orthogonal image warp. As these include all permutations, they include, in particular, most common spatial transformations such as local translations and their combinations (see, e.g. Memisevic, 2012, for a recent discussion). The assumption of orthogonality of transformations is made implicitly also by the motion energy model.

To detect an orthogonal transformation, P, between the two images, we propose to use filters for which

$$\vec{w}_2 = P\vec{w}_1 \tag{1}$$

holds, and then to check whether the condition

$$\vec{w}_2^{\mathrm{T}}\vec{x}_2 = \vec{w}_1^{\mathrm{T}}\vec{x}_1 \tag{2}$$

is true. We shall call this the ”synchrony condition”. It amounts to choosing a filter pair, such that it is an example of the transformation we want to detect (Eq. 1), and to determine whether the two filters yield equal responses when applied in sequence to the two frames (Eq. 2). We shall later relax the exact equality to an approximate equality.

To see why the synchrony condition counts as evidence for the presence of the transformation, note first that $\vec{x}_2 = P\vec{x}_1$ implies $\vec{w}_2^{\mathrm{T}}\vec{x}_2 = \vec{w}_2^{\mathrm{T}}P\vec{x}_1$.

From this, we get:

$$\vec{x}_2 = P\vec{x}_1 \text{ (presence of } P)$$
$$\Rightarrow \quad \vec{w}_2^{\mathrm{T}}\vec{x}_2 \left(= \vec{w}_2^{\mathrm{T}}P\vec{x}_1 = (P^{\mathrm{T}}\vec{w}_2)^{\mathrm{T}}\vec{x}_1 = \right) \vec{w}_1^{\mathrm{T}}\vec{x}_1 \tag{3}$$

The last equation follows from $P^{\mathrm{T}} = P^{-1}$ (orthogonality of P). This shows that the presence of the transformation P implies synchrony (Eq.2) for any two filters which themselves are related through P, that is $\vec{w}_2 = P\vec{w}_1$. In order to detect the presence of P, we may thus look for the synchrony condition, using a set of filters transformed through P. This is an inductive (statistical) reasoning step, in that we can accumulate evidence for a transformation by looking for synchrony across multiple filters. The absence of the transformation implies that all filter pairs violate the synchrony condition.

It is interesting to note that for Gabor filters, phase shifts and position shifts are locally the same (e.g. Fleet et al., 1996). For global Fourier features, phase shifts and position shifts are exactly identical. Thus, synchrony (Eq. 1) between the inputs and a sequence of phase-shifted Fourier (or Gabor) features, for example, allows us to detect transformations which are *local translations*. We shall discuss learning of filters from video data in Section 3.

The synchrony condition can be extended to a sequence of more than two frames as follows: Let $\vec{x}_i, \vec{w}_i$ $(i = 1, \ldots, T)$ denote the input frames and corresponding filters. To detect a set of transformations P_i, each of which relates two adjacent frames $(\vec{x}_i, \vec{x}_{i+1})$, set $\vec{w}_{i+1} = P_i\vec{w}_i$ for all i. The condition for the presence of the sequence of transformations now turns into

$$\vec{w}_i^{\mathrm{T}}\vec{x}_i = \vec{w}_j^{\mathrm{T}}\vec{x}_j \quad \forall i, j = 1, \ldots, T \text{ and } i \neq j \tag{4}$$

2.2. The insufficiency of weighted summation

To check for the synchrony condition in practice, it is necessary to detect the equality of transformed filter responses across time (Eq. 2). Most current deep learning models are based on layers of weighted summation followed by a nonlinearity. The detection of synchrony, unfortunately, cannot be performed in a layer of weighted summation plus nonlinearity as we shall discuss now.

The fact that the sum of filter responses, $\vec{w}_1^{\mathrm{T}}\vec{x}_1 + \vec{w}_2^{\mathrm{T}}\vec{x}_2$, will attain its maximum for inputs that both match their filters seems to suggest that thresholding it would allow us to detect synchrony. This is not the case, however, because thresholding works well only for inputs which are very similar to the feature vectors themselves: Most inputs, in practice, will be normalized superpositions of *multiple* feature vectors. Thus, to detect synchrony with a thresholded sum, we would need to use a threshold small enough to represent features, $\vec{w}_1, \vec{w}_2$, that explain only a fraction of the variability in $\vec{x}_1, \vec{x}_2$. If we assume, for example, that the two features $\vec{w}_1$, $\vec{w}_2$ account for 50% of the variance in the inputs (an overly optimistic assumption), then we would have to reduce the threshold to be one half of the maximum attainable response to be able to detect synchrony. However, at this level, there is no way to distinguish between two stimuli which do satisfy the synchrony condition (*the motion in question is present*), and two stimuli where one image is a perfect match to its filter and the other has zero overlap with its filter (*the motion in question is not present*). The situation can only become worse for feature vectors that account for less than 50% of the variability.

2.3. Synchrony detection using multiplicative interactions

If one is willing to abandon weighted sums as the only allowable type of module for constructing deep networks, then a simple way to detect synchrony is by allowing for multiplicative ("gating") interactions between filter responses: The product

$$p = \vec{w}_2^{\mathrm{T}} \vec{x}_2 \cdot \vec{w}_1^{\mathrm{T}} \vec{x}_1 \tag{5}$$

will be large only if $\vec{w}_2^{\mathrm{T}} \vec{x}_2$ and $\vec{w}_1^{\mathrm{T}} \vec{x}_1$ both take on large (or both very negative) values. Any sufficiently small response of either $\vec{w}_2^{\mathrm{T}} \vec{x}_2$ or $\vec{w}_1^{\mathrm{T}} \vec{x}_1$ will shut off the response of p, regardless of the filter response on the other image. That way, even a low threshold on p will not sacrifice our ability to differentiate between the presence of some feature in one of the images vs. the *partial* presence of the transformed feature in both of the images (synchrony).

A related, less formal, argument for product interactions is that synchrony detection amounts to an operation akin to a logical "AND". This is at odds with the observation that weighted sums "accumulate" information and resemble a logical "OR" rather than an "AND" (e.g. Zetzsche & Nuding, 2005).

2.4. A locally learned gating module

It is important to note that multiplicative interactions will allow us to check for the synchrony condition using entirely *local* operations: Figure 1 illustrates how we may define a "neuron" that can detect synchrony by allowing for gating interactions within its "dendritic tree". A model consisting of multiple of these synchrony detector units will be a single-layer model, as there is no cross-talk required between the units. As we shall show, this fact allows us to use very fast local update rules for learning synchrony from data.

This is in stark contrast to the learning of energy models and bi-linear models (e.g. Grimes & Rao, 2005; Hyvärinen & Hoyer, 2000; Memisevic & Hinton, 2007; Bethge et al., 2007; Taylor et al., 2010), which rely on non-local computations, such as back-prop, for learning (see also, Section 2.5). Although multiplicative interactions have been a common ingredient in most of these models their motivation has been that they allow for the computation of subspace energies or subspace angles rather than synchrony (eg. Memisevic, 2012).

The usefulness of intra-dendritic gating has been discussed at lengths in the neuroscience literature, for example, in the work by Mel and colleagues (e.g. Archie & Mel, 2000; Mel, 1994). But besides multi-layer bilinear models discussed above, it has not received much attention in machine learning. Dendritic gating is reminiscent also of "Pi-Sigma" neurons (Shin & Ghosh, 1991), which have been applied to some supervised prediction tasks in the past.

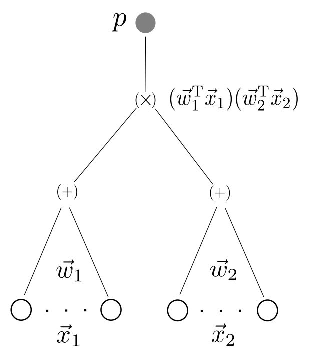
p
$(\times)$ $(\vec{w}_1^{\mathrm{T}}\vec{x}_1)(\vec{w}_2^{\mathrm{T}}\vec{x}_2)$
$(+)$
$(+)$
$\vec{w}_1$
$\vec{w}_2$
$\vec{x}_1$
$\vec{x}_2$

Figure 1. Gating within a "dendritic tree."

2.5. Pooling and energy models

Figure 2 shows an illustration of a product response using a 1-D example. The figure shows how the product of transformed filters and inputs yields a large response whenever (i) the input is well-represented by the filter and (ii) the input evolves over time in a similar way as the filter (second column in the figure). The figure also illustrates how failing to satisfy either (i) or (ii) will yield a small product response (two rightmost columns). The need to satisfy condition (i) makes the product response dependent on the input. This dependency can be alleviated by *pooling* over multiple products, involving multiple different filters, such that the top-level pooling unit fires, if any subset of the synchrony detectors fires. The classic energy model, for example, pools over filter pairs in quadrature to eliminate the dependence on *phase* (Adelson & Bergen, 1985; Fleet et al., 1996). In practice, however, it is not just phase but also frequency, position and orientation (or entirely different properties for non-Fourier features), which will determine whether an image is aligned with a filter or not. We investigate pooling with a separately trained pooling layer in Section 3.

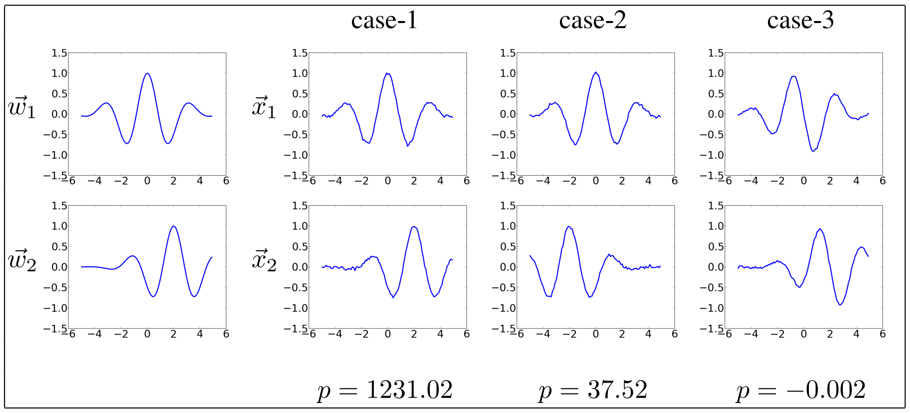

Figure 2. Demonstration of product responses p with two filters $\vec{w}_1, \vec{w}_2$ encoding a translation P. case-1: $\vec{x}_2 = P\vec{x}_1$; case-2: $\vec{x}_2 \neq P\vec{x}_1$; case-3: $\vec{x}_2 = P\vec{x}_1$ but $\vec{x}_1$ and $\vec{w}_1$ are out of phase by $\pi/2$.

3. Learning synchrony from data

We now discuss how to learn filters which allow us to detect the synchrony condition. There are in principle many ways to achieve this in practice, and we introduce two examples in the following. One is a variant of the contractive autoencoder (Rifai et al., 2011), the other is a temporal variant of the K-means clustering algorithm. In the following, we let $\vec{x}, \vec{y} \in \mathbb{R}^N$ denote images, and we let $\mathbf{W}^x, \mathbf{W}^y \in \mathbb{R}^{Q\times N}$ denote matrices whose rows contain Q feature vectors, which we will denote by $\vec{W}^x_q, \vec{W}^y_q \in \mathbb{R}^N$.

3.1. Synchrony autoencoder

First, we present an autoencoder which can encode motion across two frames and later extend it to longer sequences. Given two images, we first compute the linear filter responses $\vec{f}^x = \mathbf{W}^x\vec{x}$ and $\vec{f}^y = \mathbf{W}^y\vec{y}$. Given the derivations in Section 2, an encoding of the motion, $\vec{h} = \vec{h}(\vec{x}, \vec{y})$, inherent in the image sequence may then be defined as

$$\vec{h} = \sigma(\vec{f}^x \odot \vec{f}^y) \tag{6}$$

where $\odot$ is element-wise multiplication and σ is the sigmoid nonlinearity $(1+\exp(-x))^{-1}$. This definition makes sense only, if features vectors are related by the transformation we wish to detect. We shall now discuss how we can define a reconstruction criterion that enforces this criterion.

The standard way to train an autoencoder on images is to add a decoder and to minimize reconstruction error. In our case, because of the presence of multiplicative interactions in the encoder, the encoding loses information about the sign of the input. However, note that we may interpret the multiplicative interactions as gating as discussed in the previous section. This suggests defining the reconstruction error on one input, given the other. In the decoder we thus perform an element-wise multiplication of the hiddens and factors of one of the input to reconstruct the other. One may also view this as re-introducing the sign information at reconstruction time. Assuming an autoencoder with tied weights, the reconstructed inputs can then be defined as

$$\hat{x} = (\mathbf{W}^x)^{\mathrm{T}}(\vec{h} \odot \vec{f}^y) \tag{7}$$

$$\hat{y} = (\mathbf{W}^y)^{\mathrm{T}}(\vec{h} \odot \vec{f}^x) \tag{8}$$

We define the reconstruction error as the average squared difference between the two inputs and their respective reconstructions:

$$L((\vec{x}, \vec{y}), (\hat{\vec{x}}, \hat{\vec{y}})) = \|(\vec{x} - \hat{\vec{x}})\|^2 + \|(\vec{y} - \hat{\vec{y}})\|^2 \tag{9}$$

Learning amounts to minimizing the reconstruction error wrt. the filters $(\mathbf{W}^x)$ and $(\mathbf{W}^y)$. In contrast to bi-linear models, which may be trained using similar criteria (e.g. Memisevic, 2011; Taylor et al., 2010), the representation of motion in Eq. 6 will be dependent on the image content, such as Fourier phase for translational motion. But this dependence can be removed using a separately trained pooling layer as we shall show. The absence of pooling during feature learning allows for much more efficient learning as we show in Section 4. Note that, in practice, one may add bias terms to the definition of hiddens and reconstructions.

3.1.1. Contractive regularization

It is well-known that regularization is important to extract useful features and to learn sparse representations. Here, we use contraction as regularization (Rifai et al., 2011). This amounts to adding the Frobenius norm of the Jacobian of the extracted features, i.e., the sum of squares of all partial derivatives of $\vec{h}$ with respect to $\vec{x}, \vec{y}$

$$\|J_e(\vec{x}, \vec{y})\|^2_E = \sum_{ij}\left(\frac{\partial h_j(\vec{x}, \vec{y})}{\partial x_i}\right)^2 + \sum_{ij}\left(\frac{\partial h_j(\vec{x}, \vec{y})}{\partial y_i}\right)^2 \tag{10}$$

which for the sigmoid-of-square non-linearity becomes

$$\begin{aligned}\|J_e(\vec{x}, \vec{y})\|^2_E &= \textstyle\sum_j (h_j(1-h_j))^2 (f^x_j)^2 \sum_i (W^y_{ij})^2 \\ &\quad + \textstyle\sum_j (h_j(1-h_j))^2 (f^y_j)^2 \sum_i (W^x_{ij})^2 \end{aligned} \tag{11}$$

For training, we add the regularization term to the reconstruction cost, using a hyperparameter λ. Contractive regularization is not possible in (multi-layer) bi-linear models, due to the computational complexity of computing the contraction gradient for multiple layers (e.g. Memisevic, 2011). Being a single layer model, the synchrony autoencoder (SAE) makes the application of contractive regularization feasible.

As an alternative to contractive regularization, we also experimented with denoising regularization (Vincent et al., 2008), which amounts to corrupting the input $\vec{X}$ and training the model to reconstruct the actual input (which may be thought of as denoising the input). We used zero-mask noise (Vincent et al., 2008), which involves randomly setting a fraction of the components of the input to zero. The contraction parameter λ and noise fraction are set by cross-validation.

3.2. Synchrony K-means

We now describe a temporal variant of the (online) K-means algorithm to learn synchrony. Online K-means clustering has recently been shown to yield efficient, and highly competitive image features for objective recognition (Coates et al., 2011).

We first note that, given a set of Q cluster centers $\vec{W}_q^x$, performing online gradient-descent on the standard (not synchrony) K-means clustering objective is equivalent to updating the cluster centers using the local competitive learning rule (Rumelhart & Zipser, 1986)

$$\Delta \vec{W}_s^x = \eta(\vec{x} - \vec{W}_s^x) \tag{12}$$

where η is a step-size and s is the "winner-takes-all" assignment

$$s = \arg\min_q \|\vec{x} - \vec{W}_q^x\|^2 \tag{13}$$

When cluster-centers ("features") are contrast-normalized, the assignment function is equivalent to

$$s = \arg\max_q [(\vec{W}_q^x)^{\mathrm{T}} \vec{x}] \tag{14}$$

With the online K-means rule in mind, we now define a synchrony K-means (SK-means) model as follows. We define the synchrony condition by first introducing multiplicative interactions in the assignment function:

$$s = \arg\max_q [((\vec{W}_q^x)^{\mathrm{T}} \vec{x})((\vec{W}_q^y)^{\mathrm{T}} \vec{y})] \tag{15}$$

Note that computing the multiplication is equivalent to replacing the K-means winner-takes-all units by gating units (cf., Figure 1). This allows us to redefine the K-means objective function to be the reconstruction error between one input and the assigned prototype vector, which is gated (multiplied elementwise) with the projection of the other input:

$$L_x = (\vec{x} - \vec{W}_s^x((\vec{W}_s^y)^{\mathrm{T}}\vec{y}))^2 \tag{16}$$

The gradient of the reconstruction error is

$$\frac{\partial L_x}{\partial \vec{W}_s^x} = -4(\vec{x}(\vec{W}_s^y)^{\mathrm{T}}\vec{y} - \vec{W}_s^x((\vec{W}_s^y)^{\mathrm{T}}\vec{y})^2) \tag{17}$$

This allows us to define the *synchrony K-means learning rule*:

$$\Delta \vec{W}_s^x = \eta(\vec{x}(\vec{W}_s^y)^{\mathrm{T}}\vec{y} - \vec{W}_s^x((\vec{W}_s^y)^{\mathrm{T}}\vec{y})^2) \tag{18}$$

Similar to the online-kmeans rule (Rumelhart & Zipser, 1986), we obtain a Hebbian term $\vec{x}(\vec{W}_s^y)^{\mathrm{T}}\vec{y}$, and an "active forgetting" term $(\vec{W}_s^x((\vec{W}_s^y)^{\mathrm{T}}\vec{y})^2)$ which enforces competition among the hiddens. The Hebbian term, in contrast to standard K-means, is "gated", in that it involves both the "pre-synaptic" input $\vec{x}$, and the projected pre-synaptic input $(\vec{W}_s^y)^{\mathrm{T}}\vec{y}$ coming from the other input. Similarly the update rule for $\vec{W}_s^y$ is given by

$$\Delta \vec{W}_s^y = \eta(\vec{y}(\vec{W}_s^x)^{\mathrm{T}}\vec{x} - \vec{W}_s^y((\vec{W}_s^x)^{\mathrm{T}}\vec{x})^2) \tag{19}$$

3.3. Synchrony detection using even-symmetric non-linearities

As defined in Section 2.1, $\vec{x}_i, \vec{w}_i$ $(i = 1, \ldots, T)$ denote the input frames and corresponding filters. An even-symmetric nonlinearity with global minimum at zero, such as the square function, applied to $\sum_i \vec{w}_i^{\mathrm{T}} \vec{x}_i$, will be a detector of the synchrony condition, too. The reason is the binomial identity, which states that the square of the sum of terms contains the pair-wise products between all individual terms plus the squares of the individual terms. The latter do not change the preferred stimulus of the unit (Fleet et al., 1996; Memisevic, 2012). The value of $(\sum_i \vec{w}_i^{\mathrm{T}} \vec{x}_i)^2$ is high only when the individual terms are equal to each other and of high value, i.e, $\vec{w}_i^{\mathrm{T}} \vec{x}_i = \vec{w}_j^{\mathrm{T}} \vec{x}_j$ which is the synchrony condition in case of sequences (Equation 4). Squaring non-linearities applied to the sum of phase-shifted Gabor filter responses have been the cornerstone of the energy model (Adelson & Bergen, 1985; Watson & Albert J. Ahumada, 1985; Hyvarinen et al., 2009).

Even-symmetric non-linearities implicitly compute pair-wise products and they may be implemented using multiplicative interactions, too: Consider the unit in Figure 1, using "tied" inputs $\vec{x}_1 = \vec{x}_2$, and assume that they contain a video sequence rather than a single image. If we also use tied weights $\vec{w}_1 = \vec{w}_2$, then the output, p, of this unit will be equal to the square of $\vec{w}_1^{\mathrm{T}} \vec{x}_1$. In practice, the model can learn to tie weights, if required.

To enable the models from Sections 3.1 and 3.2 to encode motion across multiple frames, we may thus proceed as

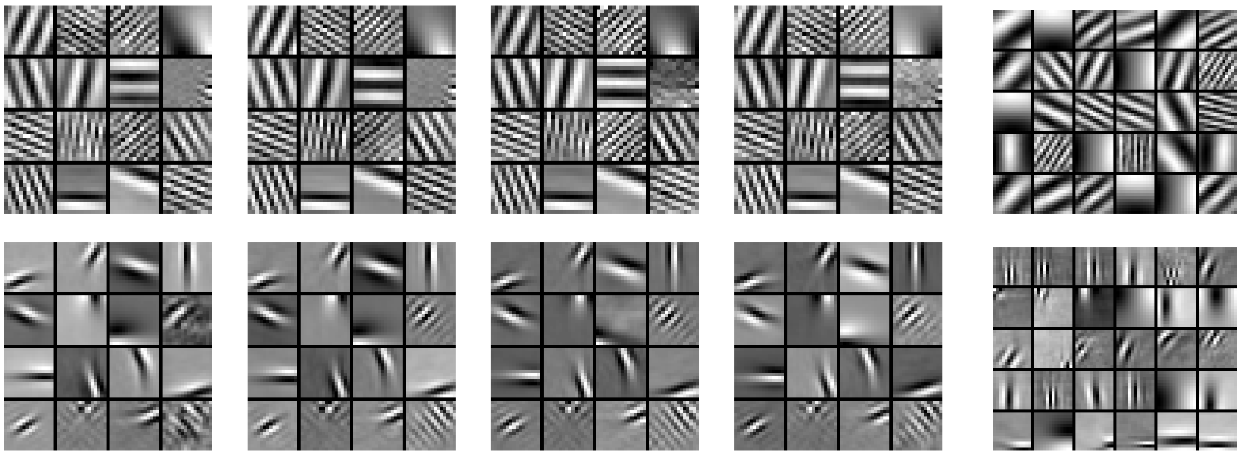

Figure 3. **Row 1:** Filters learned on synthetic translations of natural image patches. **Row 2:** Filters learned on natural videos. **Columns 1-4:** Frames 1-4 of the learned filters. **Column 5:** Filter groupings learned by a separate layer of K-means (only first frame filters shown). Each row in column 5 shows the six filters contributing the most to that cluster center.

follows: Let $\vec{X} \in \mathbb{R}^N$ be the concatenation of T frames $\vec{x}_t \in \mathbb{R}^M, t = 1, \ldots, T$, and let $\mathbf{W} \in \mathbb{R}^{Q \times N}$ denote the matrix containing the Q feature vectors $\vec{W}_q \in \mathbb{R}^N$ stacked row-wise. Each feature is composed of frame features $\vec{w}_{qt} \in \mathbb{R}^M$ where each $\vec{w}_{qt}$ spans one frame $\vec{x}_t$ from the input video.

The SAE can be adapted to sequences by replacing frames $\vec{x}, \vec{y}$ with a sequence $\vec{X}$ and tying the weights $\mathbf{W^x}, \mathbf{W^y}$ to $\mathbf{W}$. The representation of motion from Equation 6 can now be redefined as,

$$\begin{aligned} H_q &= \sigma(F_q^2), \quad \forall q = 1, \ldots, Q \\ &= \sigma\Big(((\vec{W}_q)^{\mathrm{T}} \vec{X})^2\Big) = \sigma\Big((\sum_t \vec{w}_{qt}^{\mathrm{T}} \vec{x}_t)^2\Big) \end{aligned} \tag{20}$$

Note that computing the square of $\sum_t \vec{w}_{qt}^{\mathrm{T}} \vec{x}_t$ above also accounts for synchrony as explained earlier. The reconstruction error and regularization term for this model can be derived by just replacing appropriate terms in Equations 9 and 11, respectively.

Similarly the update rule for the SK-means model becomes

$$\Delta \vec{W}_s = \eta(\vec{X}(\vec{W}_s^{\mathrm{T}} \vec{X}) - \vec{W}_s(\vec{W}_s^{\mathrm{T}} \vec{X})^2) \tag{21}$$

where the assignment function s is

$$s(\vec{X}) = \arg\max_q [(\vec{W}_q^{\mathrm{T}} \vec{X})^2] \tag{22}$$

For inference in case of the SK-means model, we use a sigmoid activation function on the squared features in our experiments instead of winner-takes-all (cf., Eq. 22). As in the case of object classification (Coates et al., 2011), relaxing the harsh sparsity induced by K-means tends to yield codes better suited for recognition tasks.

Example filters learned with the contractive SAE on sequences are shown in Figure 3. In the first row of the figure, columns 1 to 4 show filters learned on 50,000 synthetic movies generated by translating image patches from the natural image dataset in (Martin et al., 2001). Columns 1 to 4 of the second row show filters learned on blocks sampled from videos of a broadcast TV database in (Hateren & Schaaf, 1998). We obtained similar filters using the SK-means model.

3.4. Learning a separate pooling layer

To study the dependencies of features, we performed K-means clustering, using 500 centroids, on the hiddens extracted from the training sequences. Column 5 of Figure 3 shows, for the most active clusters across the training data, the six features which contribute the most to each of the cluster centers. It shows that the "pooling units" (cluster centers) group together features with similar orientation and position, and with arbitrary frequency and phase. This is to be expected, as translation in any direction will affect *all* frequencies and phase angles, and only "nearby" orientations and positions. Note in particular, that pooling across phase angles alone, as done by the classic energy model, would not be sufficient, and it is, in fact, not the solution found by pooling.

4. Application to activity recognition

Activity recognition is a common task for evaluating models of motion understanding. To allow for a fair comparison, we use the same pipeline as described in (Le et al., 2011; Wang et al., 2009), using the features learned by our models. We train our models on pca-whitened input

Table 1. Average accuracy on KTH.

ALGORITHM	PERFORMANCE(%)
SAE	93.5
SK-means	93.6
GRBM(TAYLOR ET AL., 2010)	90.0
ISA MODEL(LE ET AL., 2011)	93.9

patches of size $10 \times 16 \times 16$. The number of training samples is $200,000$. The number of product units are fixed at 300. For inference sub blocks of the same size as the patch size are cropped from "super blocks" of size $14 \times 20 \times 20$ (Le et al., 2011). The sub blocks are cropped with a stride of 4 on each axis giving 8 sub blocks per super block. The feature responses of sub blocks are concatenated and dimensionally reduced using PCA to form the local feature. Using a separate layer of K-means, a vocabulary of 3000 spatio-temporal words is learned with $500,000$ samples for training. In all our experiments the super blocks are cropped densely from the video with a 50% overlap. Finally, a χ^2-kernel SVM on the histogram of spatio-temporal words is used for classification.

4.1. Datasets

We evaluated our models on several popular activity recognition benchmark datasets:

KTH (Schuldt et al., 2004): Six actions performed by 25 subjects. Samples divided into train and test data according to the authors original split. The multi-class SVM is directly used for classification.

UCF sports(Rodriguez et al., 2008): Ten action classes. The total number of videos in the dataset is 150. To increase the data we add horizontally flipped version of each video to the dataset. Like in (Rodriguez et al., 2008) we train a multi-class SVM for classification, and we use leave-one-out for evaluation. That is, each original video is tested with all other videos as training set except the flipped version of the one being tested and itself.

Hollywood2 (Marszałek et al., 2009): Twelve activity classes. It consists of 884 test samples and 823 train samples with some of the video samples belonging to multiple classes. Hence, a binary SVM is used to compute the average precision (AP) of each class and the mean AP over all classes is reported (Marszałek et al., 2009).

YUPENN dynamic scenes (Derpanis, 2012): Fourteen scene categories with 30 videos for each category. We only use the gray-scale version of the videos in our experiments. Leave-one-out cross-validation is used for performance evaluation (Derpanis, 2012).

Table 2. Average accuracy on UCF sports.

ALGORITHM	PERFORMANCE(%)
SAE	86.0
SK-means	84.7
ISA MODEL(LE ET AL., 2011)	86.5

Table 3. Mean AP on Hollywood2.

ALGORITHM	PERFORMANCE(%)
SAE	51.8
SK-means	50.5
GRBM(TAYLOR ET AL., 2010)	46.6
ISA MODEL(LE ET AL., 2011)	53.3
COVAE (MEMISEVIC, 2011)	43.3

4.2. Results

The results are shown in Tables 1, 2, 3 and 4. They show that the SAE and SK-means are competitive with the state-of-the-art, although learning is simpler than for most existing methods. To evaluate the importance of element-wise products of hiddens and factors in the decoder (Equation 7 of Section 3), we also evaluated a model without multiplying the factors, that is, a standard autoencoder, on the Hollywood2 dataset. The model achieved an average precision of only 42.7 using the same configuration as that of experiments with the original model. The covariance auto-encoder (Memisevic, 2011) learns an additional mapping layer summing over the squared simple cell responses. Table 3 shows that the performance of this model is considerably lower than our approaches showing that learning the pooling-layer along with features does not help. We also experimented with noise as regularization, where we achieved 50.1 AP on Hollywood2, 92.5% and 85.3% accuracy on KTH and UCF sports datasets respectively. This shows that results from contraction and noise as regularizers are similar except that contraction is computationally less expensive.

Table 4. Average accuracy on YUPENN.

ALGORITHM	PERFORMANCE(%)
SAE (K-NN)	80.7
SAE (χ^2SVM)	96.0
SK-means (χ^2SVM)	95.2
SOE (DERPANIS, 2012)	79.0

Table 5. Performance on column dataset using SAE trained on row dataset.

DATASET	KTH	UCF	HOLLYWOOD2
KTH	93.5	85.3	44.7
UCF	92.9	86.0	48.9
HOLLYWOOD2	92.7	85.3	51.8

Table 6. Training time.

ALGORITHM	TIME
SK-means (GPU)	2 MINUTES
SK-means (CPU)	3 MINUTES
SAE (GPU)	$1-2$ HOURS
ISA (LE ET AL., 2011)	$1-2$ HOURS
GRBM (TAYLOR ET AL., 2010)	$2-3$ DAYS

4.3. Unsupervised learning and dataset bias

To show that our models learn features that can generalize across datasets ("self-taught learning" (Le et al., 2011)), we trained SAE on random samples from one of the datasets and used it for feature extraction to report performance on the others. The performances using the same metrics as before are shown in table 5. It can be seen that the performance gets reduced by only a fairly small fraction as compared to training on samples from the respective dataset. Only in the case where training on the KTH dataset, performance on Hollywood2 is considerably lower. This is probably due to the less diverse activities in KTH as compared to those in Hollywood2.

4.4. Computational efficiency

Training times for learning the motion features are shown in Table 6. They show that SK-means (trained on CPU) is orders of magnitude faster than all other models. For the GPU implementations, we used the theano library (Bergstra et al., 2010). We also calculated inference times using a similar metric as (Le et al., 2011) and computed the time required to extract descriptors for 30 videos from the Hollywood2 dataset with resolution 360×288 pixels (with "sigmoid-of-square" hiddens they are identical for SK-means or SAE). Average inference times (in seconds/frame) were 0.058 on CPU and 0.051 on GPU, making the models feasible in practical, and possibly real-time, applications. All experiments were performed on a system with a 3.20GHz CPU, 24GB RAM and a GTX 680 GPU.

5. Conclusions

Our work shows that learning about motion from videos can be simplified and significantly sped up by disentangling learning about the spatio-temporal evolution of the signal from learning about invariances in the inputs. This allows us to achieve competitive performance in activity recognition tasks at a fraction of the computational cost for learning motion features required by existing methods, such as the motion energy model (Le et al., 2011). We also showed how learning about motion is possible using entirely local learning rules.

Computing products by using "dendritic gating" within individual, but competing, units may be viewed as an efficient compromise between bi-linear models, that are expensive because they encode interactions between all pairs of pixels (Grimes & Rao, 2005; Memisevic & Hinton, 2007; Olshausen et al., 2007), and "factored" models (e.g. Cadieu & Olshausen, 2011; Taylor et al., 2010; Memisevic, 2012), which are multi-layer models that rely on more complicated training schemes such as back-prop and which do not work as well for recognition.

Acknowledgments: This work was supported by the German Federal Ministry of Education and Research (BMBF) in project 01GQ0841 (BFNT Frankfurt) as well as NSERC Discovery grant and a Google faculty research award.

References

Adelson, Edward H. and Bergen, James R. Spatiotemporal energy models for the perception of motion. *J. OPT. SOC. AM. A*, 2(2):284–299, 1985.

Archie, Kevin A. and Mel, Bartlett W. A model for intradendritic computation of binocular disparity. *Nature Neuroscience*, 3(1):54–63, January 2000.

Arndt, P.A., Mallot, H.A., and Bülthoff, H.H. Human stereovision without localized image features. *Biological cybernetics*, 72(4):279–293, 1995.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In *SciPy*, 2010.

Bethge, M, Gerwinn, S, and Macke, JH. Unsupervised learning of a steerable basis for invariant image representations. In *Human Vision and Electronic Imaging XII*. SPIE, 2007.

Cadieu, Charles F. and Olshausen, Bruno A. Learning Intermediate-Level Representations of Form and Motion from Natural Movies. *Neural Computation*, 24(4):827–866, December 2011.

Coates, Adam, Lee, Honglak, and Ng, A. Y. An analysis of single-layer networks in unsupervised feature learning. In *Artificial Intelligence and Statistics*, 2011.

Derpanis, Konstantinos G. Dynamic scene understanding: The role of orientation features in space and time in scene classification. In *CVPR*, 2012.

Fleet, D., Wagner, H., and Heeger, D. Neural encoding of binocular disparity: Energy models, position shifts and phase shifts. *Vision Research*, 36(12):1839–1857, June 1996.

Grimes, David and Rao, Rajesh. Bilinear sparse coding for invariant vision. *Neural Computation*, 17(1):47–73, 2005.

Hateren, J. H. van and Schaaf, A. van der. Independent component filters of natural images compared with simple cells in primary visual cortex. *Proceedings: Biological Sciences*, 265(1394):359–366, Mar 1998.

Hyvärinen, Aapo and Hoyer, Patrik. Emergence of phase- and shift-invariant features by decomposition of natural images into independent feature subspaces. *Neural Comput.*, 12:1705–1720, July 2000.

Hyvarinen, Aapo, Hurri, Jarmo, and Hoyer, Patrick O. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*. Springer Publishing Company, Incorporated, 2009.

Ji, Shuiwang, Xu, Wei, Yang, Ming, and Yu, Kai. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

Le, Q.V., Zou, W.Y., Yeung, S.Y., and Ng, A.Y. Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. In *CVPR*, 2011.

Marszałek, Marcin, Laptev, Ivan, and Schmid, Cordelia. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

Martin, D., Fowlkes, C., Tal, D., and Malik, J. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *ICCV*, 2001.

Mel, Bartlett W. Information processing in dendritic trees. *Neural Computation*, 6(6):1031–1085, 1994.

Memisevic, Roland. Gradient-based learning of higher-order image features. In *ICCV*, 2011.

Memisevic, Roland. On multi-view feature learning. In *ICML*, 2012.

Memisevic, Roland and Hinton, Geoffrey. Unsupervised learning of image transformations. In *CVPR*, 2007.

Olshausen, Bruno, Cadieu, Charles, Culpepper, Jack, and Warland, David. Bilinear models of natural images. In *SPIE Proceedings: Human Vision Electronic Imaging XII*, San Jose, 2007.

Rifai, Salah, Vincent, Pascal, Muller, Xavier, Glorot, Xavier, and Bengio, Yoshua. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*, 2011.

Rodriguez, Mikel D., Ahmed, Javed, and Shah, Mubarak. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.

Rumelhart, D. E. and Zipser, D. Parallel distributed processing: explorations in the microstructure of cognition, vol. 1. chapter Feature discovery by competitive learning, pp. 151–193. MIT Press, 1986.

Schuldt, C., Laptev, I., and Caputo, B. Recognizing human actions: a local svm approach. In *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, 2004.

Shin, Yoan and Ghosh, Joydeep. The pi-sigma network: An efficient higher-order neural network for pattern classification and function approximation. In *International Joint Conference on Neural Networks*, 1991.

Taylor, Graham W., Fergus, Rob, LeCun, Yann, and Bregler, Christoph. Convolutional learning of spatio-temporal features. In *Proceedings of the 11th European conference on Computer vision: Part VI*, ECCV'10, 2010.

Vincent, Pascal, Larochelle, Hugo, Bengio, Yoshua, and Manzagol, Pierre-Antoine. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 2008.

Wang, Heng, Ullah, Muhammad Muneeb, Kläser, Alexander, Laptev, Ivan, and Schmid, Cordelia. Evaluation of local spatio-temporal features for action recognition. In *University of Central Florida, U.S.A*, 2009.

Watson, Andrew B. and Albert J. Ahumada, Jr. Model of human visual-motion sensing. *J. Opt. Soc. Am. A*, 2(2): 322–341, Feb 1985.

Zetzsche, Christoph and Nuding, Ulrich. Nonlinear and higher-order approaches to the encoding of natural scenes. *Network (Bristol, England)*, 16(2-3):191–221, 2005.