The role of spatio-temporal synchrony in the encoding of motion

Kishore Reddy Konda
Goethe University Frankfurt
konda@informatik.uni-frankfurt.de

Roland Memisevic
University of Montreal
roland.memisevic@umontreal.ca

Vincent Michalski
Goethe University Frankfurt
vmichals@rz.uni-frankfurt.de

Abstract

We consider the task of inferring motion from an image sequence. We show that the detection of a spatial transformation can be viewed as the detection of synchrony between the image sequence and a sequence of features undergoing that transformation. The classic motion energy model can be derived from this view by introducing phase invariance via pooling. The view from synchrony therefore allows us to disentangle the contributions of invariance and of motion estimation in the energy model. It also makes it possible to derive local learning rules for learning motion representations unsupervised from data. We show that a model based on local learning can achieve competitive performance in a wide variety of motion understanding tasks and works much better than hand-crafted spatio-temporal features.

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 [1, 31]. One of the motivations for this computation is the fact that computing the sum of squares allows us to detect "oriented" energy in a spatio-temporal frequency band. This, in turn, makes it possible to encode motion independently of stimulus phase and, thus, to represent motion to some degree independently from what is moving. Related models have been proposed for binocular disparity estimation (e.g., [9]), 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., [8]), but in recent years, unsupervised learning techniques have become popular which learn the features from videos (e.g. [28, 15, 14, 19]). 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 [30], 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, possibly Hebb-type, learning in combination with dendritic "gating" interactions (e.g., [18]), 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 $\vec{x}_2$ in a video. The classic energy model [1] 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 [13]. 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 [3, 9], 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 (eg., [9, 6, 21]). 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 (eg., [9, 21]). We shall revisit the close relationship between these models in Section 2.4.

2.1 Motion estimation by synchrony detection

We shall now discuss how the products inherent in a cross-correlation model allow 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 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., [21] for a recent discussion). The assumption of orthogonality of transformations is made implicitly also by the motion energy model.

One way to detect an orthogonal transformation, P, between the two images is to define the filters such that

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

holds, and to check for the condition

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

We shall call this "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 an equal response 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:

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

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., [9]). For global Fourier features, phase shifts and position shifts are exactly identical. Thus, synchrony (Eq. 1) between the stimulus and a sequence of phase-shifted Fourier (or Gabor) features, would allow us to detect transformations from the transformation class *translation*.

2.2 Synchrony detection using multiplicative interactions

To check for the synchrony condition in practice, it is necessary to detect the equality of transformed filter responses across time. Such a detection of coincidence cannot be modeled with synaptic summation, because it requires a response which is large, if and only if both stimuli match their respective filters, as we show now. Synaptic summation is unsuitable, even though the sum, $\vec{w}_1^{\mathrm{T}}\vec{x}_1 + \vec{w}_2^{\mathrm{T}}\vec{x}_2$, does attain its maximum for stimuli that both match their filters – which seems to suggest that thresholding the sum would allow us to detect synchrony.

Unfortunately, thresholding works well only for stimuli which are very similar to the feature vectors themselves. Assume that features form an orthonormal basis[1]. Most stimuli, in practice, will be normalized superpositions of *multiple* basis vectors. Thus, to detect synchrony with synaptic summation, 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 feature $\vec{w}_1$, $\vec{w}_2$ account for 50% of the variance in their respective stimuli (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.

However, if one is willing to abandon synaptic summation as the only allowed computational device, then a simple way to detect synchrony is by allowing for multiplicative ("gating") interactions between filter responses. In particular, the product $p = \vec{w}_2^{\mathrm{T}}\vec{x}_2 \cdot \vec{w}_1^{\mathrm{T}}\vec{x}_1$ 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 synaptic summation accumulates information and thus computes an operation more akin to a logical "OR" (e.g., [32]). Multiplicative interactions are a common ingredient in motion encoding models [10, 19, 22, 6, 28, 21, 5].

Figure 1 illustrates how we may define a model neuron that can detect synchrony by allowing for gating interactions within the dendritic tree of the neuron. A model consisting of multiple of these motion detector units will be a single-layer model, and there is no cross-talk required between the units. As we shall show, this fact allows us to use efficient 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., [10, 12, 19, 5, 28]), which rely on non-local computations, such as back-prop, for learning (see also, Section 2.3). The relevance of intra-dendritic gating is discussed in detail in the work by Mel and colleagues (e.g. [2, 18]). Dendritic gating is reminiscent also of "Pi-Sigma" neurons [27], which have been applied to some supervised prediction tasks in the past.

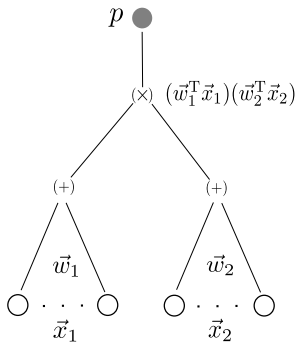

Figure 1: Gating in a dendritic tree.

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

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

[1] In practice, it is more common to use overcomplete features, but this does not change the validity of the argument.

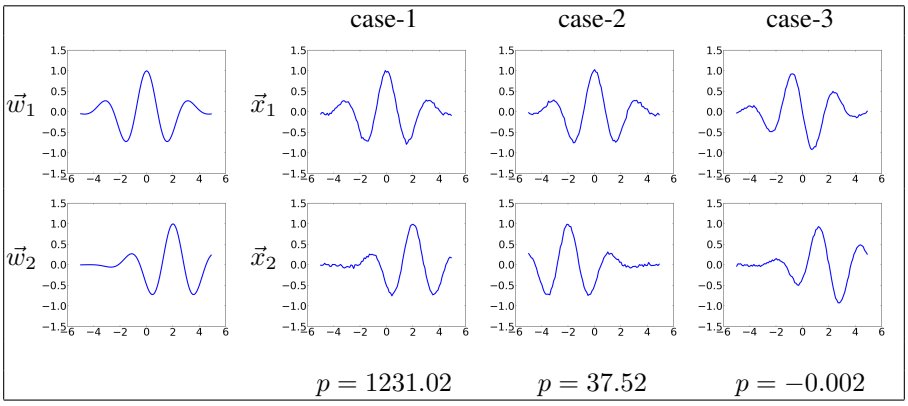

Figure 2: Demonstration of product responses p with two filters $\vec{w}_1, \vec{w}_2$ encoding a translation P. case-1: $\vec{x}_2 = P\vec{x}_1$; case-2: $\vec{x}_2 \neq P\vec{x}_1$; case-3: $\vec{x}_2 = P\vec{x}_1$ but $\vec{x}_1$ and $\vec{w}_1$ are out of phase by $\pi/2$.

2.3 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 stimuli yields a large response whenever (i) the stimulus is well-represented by the filter and (ii) the stimulus 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 stimulus. 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* [1, 9]. 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.

2.4 Synchrony detection using even-symmetric non-linearities

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 products between all individual terms plus the squares of the individual terms. The latter do not change the preferred stimulus of the unit [9, 21]. Squaring non-linearities applied to the sum of phase-shifted Gabor filter responses have been the cornerstone of the energy model [1, 31, 13].

Even-symmetric non-linearities provide an alternative to computing pair-wise products. But they may be implemented using pair-wise products, too: Consider the neuron 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 the neuron 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. We discuss learning of the weights of this model in the next section.

3 Learning synchrony from data

Feature learning involves learning of filters which allow us to detect the synchrony condition. There are many ways to achieve this in practice, and we introduce two exemplary ones in the following. One is a variant of K-means clustering, the other is a variant of a contractive autoencoder. In the remainder of this section, we let $\vec{X} \in \mathbb{R}^N$ be the concatenation of T frames $\vec{x}_t \in \mathbb{R}^M, t = 1, \ldots, T$.

3.1 Synchrony K-means

We first note that, given a set of Q cluster centers $\vec{W}_q$, performing online gradient-descent on the *standard K-means clustering* objective is equivalent to updating the cluster centers using the local

competitive learning rule $\vec{W}_{s(\vec{X})} = \vec{W}_{s(\vec{X})} + \eta(\vec{X} - \vec{W}_{s(\vec{X})})$, where η is a step-size and $s(\vec{X})$ is a "winner-takes-all" assignment [25]:

$$s(\vec{X}) = \text{arg min}_q \|\vec{X} - \vec{W}_q\|^2 \tag{5}$$

When cluster-centers ("features") are contrast-normalized, the assignment function is equivalent to $s(\vec{X}) = \text{arg max}_q[\vec{W}_q^{\mathrm{T}}\vec{X}]$.

With the online competitive K-means rule in mind, we now define a synchrony K-means (SK-means) model. To this end, we introduce the synchrony condition by first introducing a squaring nonlinearity in the assignment function:

$$s(\vec{X}) = \text{arg max}_q[(\vec{W}_q^{\mathrm{T}}\vec{X})^2] = \text{arg max}_q[(\vec{W}_q^{\mathrm{T}}\vec{X})(\vec{W}_q^{\mathrm{T}}\vec{X})] \tag{6}$$

Note that computing the square is equivalent to replacing the K-means "winner-takes-all" neurons by gating neurons (cf., Figure 1), whose two inputs are two copies of a single video sequence, and whose two corresponding features are tied to be the same. 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" (multiply elementwise) with the projection of the other input:

$$L = (\vec{X} - \vec{W}_{s(\vec{X})}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X}))^2 \tag{7}$$

The gradient can be shown to be

$$\frac{\partial L}{\partial \vec{W}_{s(\vec{X})}} = -4(\vec{X}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X}) - \vec{W}_{s(\vec{X})}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X})^2) \tag{8}$$

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

$$\vec{W}_{s(\vec{X})} = \vec{W}_{s(\vec{X})} + \eta(\vec{X}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X}) - \vec{W}_{s(\vec{X})}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X})^2) \tag{9}$$

Similar to the online-kmeans rule [25], we obtain a Hebbian term $\vec{X}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X})$, and an "active forgetting" term $(-\vec{W}_{s(\vec{X})}(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X})^2)$ 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 $(\vec{W}_{s(\vec{X})}^{\mathrm{T}}\vec{X})$ coming from the other input branch.

For inference, we use a sigmoid activation function on the squared features in our experiments instead of winner-takes-all (cf., Eq. 6). Like in the case of object classification [7], relaxing the harsh sparsity induced by K-means yields better codes for recognition.

3.2 Synchrony autoencoder

As a second example, we now show how we may define a synchrony autoencoder (SAE) to learn motion. 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 latent representation, which we call *factors* in analogy to factored bi-linear models (e.g., [28, 20]), can be written $\vec{F} = \mathbf{W}\vec{X}$. Thus, the components of $\vec{F}$ are $F_q = \sum_t \vec{w}_{qt}^{\mathrm{T}}\vec{x}_t$. We define the representation of motion as the vector $\vec{H}$ whose components are the squares of $\vec{F}$. In practice, we also apply a saturating non-linearity $\sigma(x) = (1 + \exp(-x))^{-1}$ because we found it to be more stable during learning. Thus, the encoding of input video sequence $\vec{X}$ may be defined as

$$H_q = \sigma(F_q^2) = \sigma\Big(\big(\sum_t \vec{w}_{qt}^{\mathrm{T}}\vec{x}_t\big)^2\Big) \quad \forall q = 1, \ldots, Q \tag{10}$$

The standard way to train an autoencoder is to add a decoder and to minimize reconstruction error. In our case, because of the presence of a symmetric non-linearity in the encoder, the encoding loses information about the sign of the input. However, like in the previous section, we may interpret the squaring nonlinearity as the application of an element-wise product of two identical projections of the input. This suggests defining the reconstruction error on *one* copy of the input, given the other. In the decoder we thus perform an element-wise multiplication of the hiddens and factors to

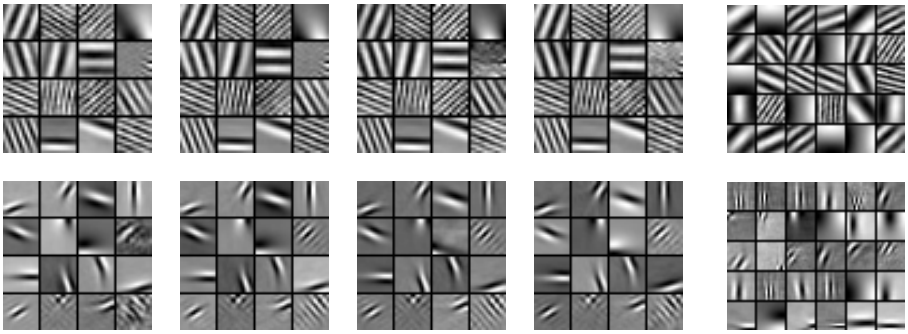

Figure 3: **Row 1:** Filters learned from synthetic translations of natural image patches. **Row 2:** Filters learned from natural videos. **Columns 1-4:** Frames 1-4 of the learned filters. **Column 5:** Filter groupings learned by a separate layer of K-means (only first frame filters shown). Each row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

reconstruct the input. One may also view this as re-introducing the sign information at reconstruction time. Assuming an autoencoder with tied weights, the reconstructed input can then be defined as

$$\vec{\hat{X}} = \mathbf{W}^{\mathrm{T}}(\vec{F} \odot \vec{H}) \tag{11}$$

where $\odot$ denotes element-wise multiplication. The reconstruction cost, L, is the squared error $L(\vec{X}, \vec{\hat{X}}) = \|(\vec{X} - \vec{\hat{X}})\|^2$, which can be optimized using gradient descent. In practice, one may add bias terms to the definitions of hiddens and reconstruction.

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 [23]. 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 $\vec{X}$.

$$\|J_e(\vec{X})\|_E^2 = \sum_{ij} \left(\frac{\partial H_j(\vec{X})}{\partial X_i} \right)^2 \tag{12}$$

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

$$\|J_e(\vec{X})\|_E^2 = \sum_i (H_i(1 - H_i))^2 {F_i}^2 \sum_j {W_{ij}}^2 \tag{13}$$

For training, we add the regularization term to the reconstruction cost, using a hyperparameter λ. Contractive regularization is not possible in (multi-layer) energy models due to the computational complexity of computing the contraction gradient for multiple layers (e.g., [20]). Being a single layer model, the SAE makes the application of contractive regularization feasible. Example filters learned with the contractive SAE 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 [17]. Columns 1 to 4 of the second row show filters learned on blocks sampled from videos of a broadcast TV database in [11]. We obtained similar filters using the SK-means model.

As an alternative to contractive regularization, we also experimented with denoising regularization [29], 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 [29], 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.

3.3 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

Algorithm	Performance(%)
SAE	93.5
SK-means	93.6
GRBM[28]	90.0
ISA model[15]	93.9

Table 1: Average accuracy on KTH.

Algorithm	Performance(%)
SAE	51.8
SK-means	50.5
GRBM[28]	46.6
ISA model[15]	53.3
covAE [20]	43.3

Table 3: Mean AP on Hollywood2.

Algorithm	Performance(%)
SAE	86.0
SK-means	84.7
ISA model[15]	86.5

Table 2: Average accuracy on UCF sports.

Algorithm	Performance(%)
SAE (k-NN)	80.7
SAE (χ^2svm)	96.0
SK-means (χ^2svm)	95.2
SOE [8]	79.0

Table 4: Average accuracy on YUPENN.

clusters across the training data, the six features which contribute 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 clustering.

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 [15, 30], using the features learned by our models. We train our models on pca-whitened input 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 $14 \times 20 \times 20$ [15]. 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.

Datasets

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

KTH [26]: 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[24]: 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 [24] 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.

Hollywood2 [16]: 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 [16].

YUPENN dynamic scenes [8]: 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 [8].

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

Dataset	KTH	UCF	Hollywood2
KTH	93.7	85.3	44.7
UCF sports	92.9	86.0	48.9
Hollywood2	92.7	85.3	51.8

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

Algorithm	**Time**
SK-means (GPU)	2 minutes
SK-means (CPU)	3 minutes
SAE (GPU)	$1-2$ hours
ISA [15]	$1-2$ hours
GRBM [28](from [15])	$2-3$ days

Table 6: Training time.

importance of element-wise products of hiddens and factors in the decoder (Equation 11 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 [20] 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.

Unsupervised learning and dataset bias: To show that our models learn features that can generalize across datasets ("self-taught learning" [15]), 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.

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 [4]. We also calculated inference times using a similar metric as [15] 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/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

We presented a view of motion energy that disentangles its two main (but separate) contributions: Detection of synchrony over time and invariance to stimulus content. We showed how disentangling these contributions makes it possible to train motion energy features efficiently using local, Hebb-type learning rules, and how this makes it possible to achieve competitive performance in activity recognition at a fraction of the computational cost for learning motion features required by existing methods.

This shows how computing products by using dendritic gating within individual, but competing, neurons may be viewed as an efficient compromise between bi-linear models [10, 19, 22, 5] (which are expensive because they encode interactions between all pairs of pixels), and "factored" complex cell models (e.g., [6, 28, 21]) (which are multi-layer models that require more complex training schemes, and which do not work as well for recognition).

Acknowledgments

This work was supported in part by the German Federal Ministy of Education and Research (BMBF) in project 01GQ0841 (BFNT Frankfurt).

References

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

[2] K. A. Archie and B. W. Mel. A model for intradendritic computation of binocular disparity. *Nature Neuroscience*, 3(1):54–63, Jan. 2000.

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

[4] J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, and Y. Bengio. Theano: a CPU and GPU math expression compiler. In *SciPy*, 2010.

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

[6] C. F. Cadieu and B. A. Olshausen. Learning Intermediate-Level Representations of Form and Motion from Natural Movies. *Neural Computation*, 24(4):827–866, Dec. 2011.

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

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

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

[10] D. Grimes and R. Rao. Bilinear sparse coding for invariant vision. *Neural Computation*, 17(1):47–73, 2005.

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

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

[13] A. Hyvarinen, J. Hurri, and P. O. Hoyer. *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision.* Springer Publishing Company, Incorporated, 1st edition, 2009.

[14] S. Ji, W. Xu, M. Yang, and K. Yu. 3D convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013.

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

[16] M. Marszałek, I. Laptev, and C. Schmid. Actions in context. In *IEEE Conference on Computer Vision & Pattern Recognition*, 2009.

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

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

[19] R. Memisevic. Unsupervised learning of image transformations. In *CVPR*, 2007.

[20] R. Memisevic. Gradient-based learning of higher-order image features. In *ICCV*, 2011.

[21] R. Memisevic. On multi-view feature learning. In *ICML*, 2012.

[22] B. Olshausen, C. Cadieu, J. Culpepper, and D. Warland. Bilinear models of natural images. In *SPIE Proceedings: Human Vision Electronic Imaging XII*, San Jose, 2007.

[23] S. Rifai, P. Vincent, X. Muller, X. Glorot, and Y. Bengio. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *ICML*, 2011.

[24] M. D. Rodriguez, J. Ahmed, and M. Shah. Action mach: a spatio-temporal maximum average correlation height filter for action recognition. In *CVPR*, 2008.

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

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

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

[28] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Proceedings of the 11th European conference on Computer vision: Part VI*, ECCV'10, 2010.

[29] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, 2008.

[30] H. Wang, M. M. Ullah, A. Kläser, I. Laptev, and C. Schmid. Evaluation of local spatio-temporal features for action recognition. In *University of Central Florida, U.S.A*, 2009.

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

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