Learning to relate images

Roland Memisevic

Abstract—A fundamental operation in many vision tasks, including motion understanding, stereopsis, visual odometry, or invariant recognition, is establishing correspondences between images or between images and data from other modalities. Recently, there has been an increasing interest in learning to infer correspondences from data using relational, spatio-temporal, and bi-linear variants of deep learning methods. These methods use multiplicative interactions between pixels or between features to represent correlation patterns across multiple images. In this paper we review the recent work on relational feature learning, and we provide an analysis of the role that multiplicative interactions play in learning to encode relations. We also discuss how square-pooling and complex cell models can be viewed as a way to represent multiplicative interactions and thereby as a way to encode relations.

Index Terms—Learning Image Relations, Spatio-temporal Features, Mapping Units, Energy Models, Complex Cells.

1 Introduction

Correspondence is arguably the most ubiquitous computational primitive in vision: **Motion estimation, action recognition** and **tracking** amount to establishing correspondences between frames; **geometric inference** and **stereo vision** between multiple views of a scene; **invariant recognition** between images and invariant templates in memory; **visual odometry** between images and estimates of motion; etc. In these and many other tasks, the relationship *between* images not the content of a single image carries the relevant information. Representing structures within a single image, such as contours, can be also considered as an instance of a correspondence problem, namely between areas, or pixels, within an image. The fact that correspondence is such a common operation across vision suggests that the task of *representing relations* may have to be kept in mind when trying to build autonomous vision systems and when trying to understand biological vision.

A lot of progress has been made recently in building models that learn to solve tasks like object recognition from independent, static images. One of the reasons for the recent progress is the use of *local features*, which help deal with occlusions and small invariances. A central finding is that the right choice of features not the choice of high-level classifier or computational pipeline are what typically makes a system work well. Interestingly, some of the best performing recognition models are highly biologically consistent, in that they are based on features that are learned unsupervised from data. Besides being biological plausible, feature learning comes with various benefits, such as helping overcome tedious engineering, helping adapt to new domains and allowing for some degree of *end-to-end learning* in place of constructing, and then combining, a large number of modules to solve a task. The fact that tasks like object recognition can be solved using biologically consistent, learning based methods raises the question whether understanding *relations* can be amenable to learning in the same way. If so, this may open up the road to learning based and/or biologically consistent approaches to a much larger variety of problems than static object recognition, and perhaps also beyond vision.

In this paper, we review a variety of recent methods that address correspondence tasks by learning local features. We discuss how these methods are fundamentally based on *multiplicative interactions* between pixels or between filter responses. The idea of using multiplicative interactions in vision was introduced about 30 years ago under the terms "mapping units" [1] and "dynamic mappings" [2]. An illustration of mapping units is shown in Figure 1: The three variables shown in the figure interact multiplicatively, and as a result, each variable (say, z_k) can be thought of as dynamically *modulating* the connections between other variables in the model (x_i and y_j). Likewise, the value of any variable (for example, y_j) can be thought of as depending on the product of the other variables (x_i, z_k) [1]. This is in contrast to common feature learning models like ICA, Restricted Boltzmann Machines, auto-encoder networks and many others, all of which are based on bi-partite networks, that do not involve any three-way multiplicative interactions. In these models, independent hidden variables interact with independent observable variables, such that the value of any variable depends on a weighted *sum* not product of the other variables. Closely related to models of mapping units are energy models (for example, [3]), which may be thought of as a way to "emulate" multiplicative interactions by computing squares.

We shall show how both mapping units and energy models can be viewed as ways to learn and detect rotations in a set of *shared invariant subspaces of a set of commuting matrices*. Our analysis may help understand why action recognition methods seem to profit from using squaring non-linearities and it predicts that the use of squaring non-linearities or multiplicative interactions

• *R. Memisevic is with Department of Computer Science and Operations Research, University of Montreal. E-mail: memisevr@iro.umontreal.ca*

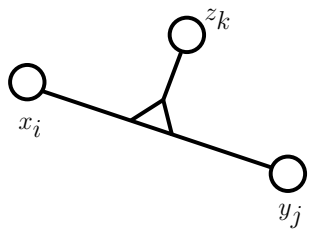

Fig. 1. A *mapping unit* [1]. The triangle symbolizes multiplicative interactions between the three variables z_k, x_i and y_j. The value of any one of the three variables is a function of the product of the others.

will be essential in any tasks that involve representing relations.

2 Multiview feature learning

2.1 Feature learning

We briefly review standard feature learning models in this section and we discuss relational feature learning in Section 2.2. We discuss extensions of relational models and how they relate to complex cells and to energy models in Section 3.

Practically all standard feature learning models can be represented by a graphical model like the one shown in Figure 2 (top). The model is a bi-partite network that connects a set of unobserved, latent variables z_k with a set of observable variables (for example, pixels) y_j. The weights w_{jk}, which connect pixel y_j with hidden unit z_k, are learned from a set of training images $\{\boldsymbol{y}^\alpha\}_{\alpha=1,\ldots,N}$. The vector of latent variables $\boldsymbol{z} = (z_k)_{k=1\ldots K}$ in Figure 2 (top) is considered to be unobserved, so one has to infer it, separately for each training case, along with the model parameters for training. The graphical model shown in the figure represents how the dependencies between components y_i and z_k are parameterized, but it does not define a model or learning algorithm. A wide variety of models and learning algorithms can be parameterized as in the figure, including principal components analysis (PCA), mixture models, k-means clustering, or restricted Boltzmann machines [4]. Each of these can in principle be used as a feature learning method (see, for example, [5] for a recent quantitative comparison of several models in a recognition task).

For the hidden variables to extract useful structure from the images, their capacity needs to be constrained. The simplest form of constraining it is to let the dimensionality K of the hidden variables be smaller than the dimensionality J of the images. Learning in this case amounts to performing dimensionality reduction. It has become increasingly obvious recently that it is more useful in most applications to use an *over-complete* representation, that is, $K > J$, and to constrain the capacity of the latent variables instead by forcing the hidden unit activities to be *sparse*. In Figure 2, and in

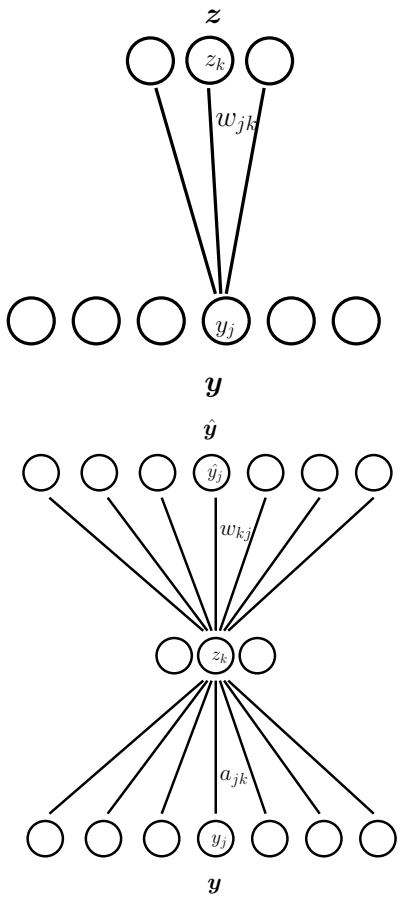

Fig. 2. **Top:** Feature learning graphical model. **Bottom:** Auto-encoder network.

what follows, we use $K < J$ to symbolize the fact that $\boldsymbol{z}$ is capacity-constrained, but it should be kept in mind that capacity can be (and often is) constrained in other ways. The most common operations performed by a trained model are: Inference (or "Analysis"): Given image $\boldsymbol{y}$, compute $\boldsymbol{z}$; and Generation (or "Synthesis"): Invent a latent vector $\boldsymbol{z}$, then compute $\boldsymbol{y}$.

A simple way to train this type of model, given training images, is by minimizing the squared reconstruction error combined with a sparsity term for the hidden variables (for example, [6]):

$$\sum_\alpha \sum_j (y_j^\alpha - \sum_k w_{jk} z_k^\alpha)^2 + \lambda \sum_k |z_k^\alpha| \qquad (1)$$

Optimization is with respect to both $W = (w_{jk})_{j=1\ldots J, k=1\ldots K}$ and all $\boldsymbol{z}^\alpha$. It is common to alternate between optimizing W and optimizing all $\boldsymbol{z}^\alpha$. After training, inference then amounts to minimizing the same expression wrt. $\boldsymbol{z}$ for test images (with W fixed).

To avoid iterative optimization during inference, one can eliminate $\boldsymbol{z}$ from Eq. 1 by defining it implicitly as a function of $\boldsymbol{y}$. A common choice of function is $\boldsymbol{z} = \mathrm{sigmoid}\,(A\boldsymbol{y})$ where A is a matrix and $\mathrm{sigmoid}(a) = (1+\exp(-a))^{-1}$ is a squashing non-linearity which confines the values of $\boldsymbol{z}$ to reside in a fixed interval. This

model is the well-known auto-encoder (for example, [7]) and it is depicted in Figure 2 (bottom). Learning amounts to minimizing reconstruction error with respect to both A and W, with gradients that are usually computed using back-prop. In practice, it is common to define the auto-encoder in a symmetric fashion by setting $A = W^{\mathrm{T}}$ in order to reduce the number of parameters and for consistency with other feature learning models.

In practice, it is common to encourage sparse hidden activities by adding an appropriate sparsity penalty during training. Alternatively, it has been shown that a similar effect can be achieved by training the auto-encoder to de-noise corrupted versions of its inputs [7]. To this end, one feeds in noisy inputs during training (for example, by adding Gaussian noise to the input, or by randomly setting individual dimensions of the input to zero) and minimizes reconstruction error with respect to the original (not noisy) data. This turns the auto-encoder into a "de-noising auto-encoder" which shows properties similar to common sparse coding methods, but inference, like in a standard auto-encoder, is a simple feed-forward mapping [7]. In the rest of the paper, we shall use the term auto-encoder to refer to de-noising auto-encoders, in other words, we shall always assume that inputs are corrupted for training.

A technique similar to the auto-encoder is the Restricted Boltzmann machine (RBM) [4], [8]: RBMs define the joint probability distribution

$$p(\boldsymbol{y}, \boldsymbol{z}) = \frac{1}{Z} \exp\big(-E(\boldsymbol{y}, \boldsymbol{z})\big), \tag{2}$$

$$\text{with } E(\boldsymbol{y}, \boldsymbol{z}) = -\sum_{jk} w_{jk} y_j z_k \tag{3}$$

$$\text{and } Z = \sum_{\boldsymbol{y}, \boldsymbol{z}} \exp\big(-E(\boldsymbol{y}, \boldsymbol{z})\big) \tag{4}$$

From the joint one can derive

$$p(z_k|\boldsymbol{y}) = \text{sigmoid}\big(\sum_j w_{jk} y_j\big) \tag{5}$$

$$p(y_j|\boldsymbol{z}) = \text{sigmoid}\big(\sum_k w_{jk} z_k\big) \tag{6}$$

This shows that inference, again, amounts to a linear mapping plus non-linearity. Learning amounts to maximizing the average log-probability $\frac{1}{N}\sum_\alpha \log p(\boldsymbol{y}^\alpha)$ of the training data. Since the derivatives with respect to the parameters are not tractable (due to the normalizing constant Z in Eq. 2), it is common to use approximate Gibbs sampling in order to approximate them. This leads to a Hebbian-like learning rule known as contrastive divergence training [8]. As with auto-encoders, it is common to enforce sparsity of the hiddens during training (for example, [9]).

Another common feature learning method is independent components analysis (ICA) (for example, [10]). One way to train an ICA-model that is complete (that is, where the dimensionality of $\boldsymbol{z}$ is the same as that of $\boldsymbol{y}$) is by encouraging latent responses to be sparse, while preventing weights from becoming degenerate [10]:

$$\min_W \|W^{\mathrm{T}} \boldsymbol{y}\|_1 \tag{7}$$

$$\text{s.t.} \quad W^{\mathrm{T}} W = I \tag{8}$$

The constraint can be inconvenient in practice, where it is commonly enforced by repeated orthogonalization using an eigen decomposition.

For most feature learning models, inference and generation are variations of the two linear functions:

$$z_k = \sum_j w_{jk} y_j \tag{9}$$

$$y_j = \sum_k w_{jk} z_k \tag{10}$$

The set of model parameters $W_{\cdot k}$ for any k are typically referred to as "features" or "filters" (although a more appropriate term would be "basis functions"; we shall use these interchangeably). Practically all methods yield Gabor-like features when trained on natural images. An advantage of non-linear models, such as RBMs and auto-encoders, is that stacking them makes it possible to learn feature hierarchies (deep learning) [11].

In practice, it is common to add bias terms, such that inference and generation (Eqs. 9 and 10) are affine not linear functions, for example, $y_j = \sum_k w_{jk} z_k + b_j$ for some parameter b_j. We shall refrain from adding bias terms to avoid clutter, noting that, alternatively, one may think of $\boldsymbol{y}$ and $\boldsymbol{z}$ as being in "homogeneous" coordinates, containing an extra, constant 1-dimension.

Feature learning is typically performed on small images patches of size between around 5×5 and 50×50 pixels. One reason for this is that training and inference can be computationally demanding. More important, local features make it possible to deal with images of different size, and to deal with occlusions and local object variations. Given a trained model, two common ways to perform invariant recognition on test images are:

"Bag-Of-Features": Crop patches around interest points (such as SIFT or Harris corners), compute latent representation $\boldsymbol{z}$ for each patch, collapse (add up) all representations to obtain a single vector $\boldsymbol{z}^{\text{Image}}$, classify $\boldsymbol{z}^{\text{Image}}$ using a standard classifier. There are several variations of this scheme, including using an extra clustering-step before collapsing features, or using a histogram-similarity in place of Euclidean distance for the collapsed representation.

"Convolutional": Crop patches from the image along a regular grid; compute $\boldsymbol{z}$ for each patch; concatenate all descriptors into a very large vector $\boldsymbol{z}^{\text{Image}}$; classify $\boldsymbol{z}^{\text{Image}}$ using a standard classifier. One can also use combinations of the two schemes (see, for example [5]).

Local features yield highly competitive performance in object recognition tasks [5]. In the next section we discuss recent approaches to extending feature learning to encode relations between, as opposed to content within, images.

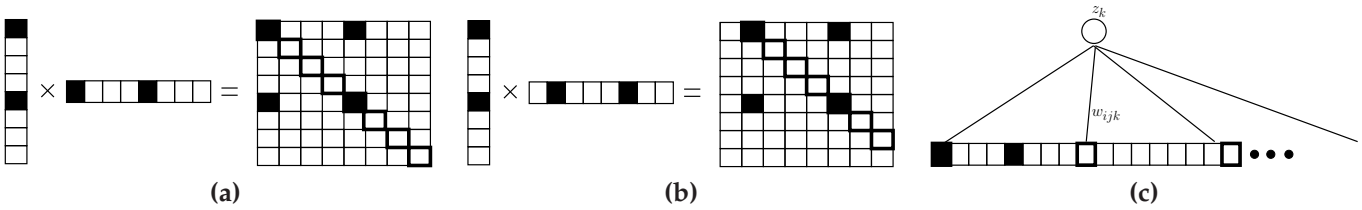

Fig. 3. **(a)** The diagonal of $L := \boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}$ contains evidence for the identity transformation. **(b)** The secondary diagonals contain evidence for shifts. **(c)** A hidden unit that pools over one of the diagonals can detect transformations. Such a hidden unit would need to compute a *sum over products* of pixels.

2.2 Learning relations

We now consider the task of learning relations between two images $\boldsymbol{x}$ and $\boldsymbol{y}$ as illustrated[1] in Figure 4, and we discuss the role of multiplicative interactions when learning relations.

2.2.1 The need for multiplicative interactions

A naive approach to modeling relations between two images would be to perform standard feature learning on the concatenation of the images. Obviously, a hidden unit in such a model would receive as input the sum of two projections, one from each image. To detect a particular transformation, the two receptive fields would need to be defined, such that one receptive field is the other modified by the transformation that the hidden unit is supposed to detect. The net input that the hidden unit receives would then tend to be high for image pairs showing the transformation. Unfortunately, however, the net input would be equally dependent on the images themselves not just the transformation: If both images change but not the transformation between them, the hidden activity would also change. Another way to see this is by noting that hidden variables act like logical "OR"-gates, which can only accumulate the information from their receptive fields [13].

It is straightforward to build a content-independent detector, however, if we allow for *multiplicative interactions* between the variables. In particular, consider the outer product $L := \boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}$ between two one-dimensional, binary images, as shown in Figure 3. Every component L_{ij} of this matrix constitutes evidence for exactly one type of transformation (translation, in the example). The components L_{ij} act like AND-gates, that can detect coincidences. Since a component L_{ij} is equal to 1 only when both corresponding pixels are equal to 1, a hidden unit that pools over multiple components (Figure 3 (c)) is much less likely to receive spurious activity that depends on the image content rather than on the transformation. Note that pooling over the components of L amounts to computing the *correlation* of the output image with a transformed version of the input image. The same would be true for real-valued images.

When a variable, z, depends linearly on the product of two other variables, that is, $z = a(xy)$, then both x and y can be thought of as *gating* the connection between the other variable and z, because $z = (ax)y = (ay)x$ (cf. Figure 1). In other words, commutativity allows us to think of x as modulating the parameter a that connects z and y (and vice versa).

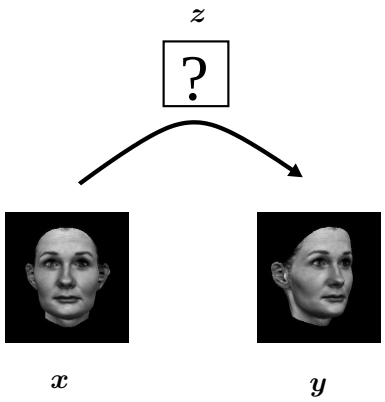

Fig. 4. Learning to encode relations: We consider the task of learning latent variables $\boldsymbol{z}$ that encode the relationship between images $\boldsymbol{x}$ and $\boldsymbol{y}$, independently of their content.

Based on this observation, a variety of feature learning models that encode transformations have been suggested (see, for example, [14], [15], [16]). The idea is to let the pixels in one image gate the parameters of a feature learning model applied to another image. This is equivalent to letting hidden variables encode the *product* of pixel x_i in one image and pixel y_j in the other image. If every pixel in the first image is allowed to independently gate every parameter in the model of the other image, then the total number of parameters is (number of hidden variables) $\times$ (number of input-pixels) $\times$ (number of output pixels). It is common to think of the parameters as populating a 3-way-tensor W with components w_{ijk}.

Figure 5 shows two illustrations of this type of model (adapted from [16]). The left sub-figure shows a feature learning model whose parameters are modulated by another image. Each input pixel of that other image can be thought of as blending in a slice $w_{i..}$ of the parameter tensor. This turns the model into a kind of *predictive* or *conditional* feature learning model [14], [16].

1. Face images are taken from the data-base described in [12]

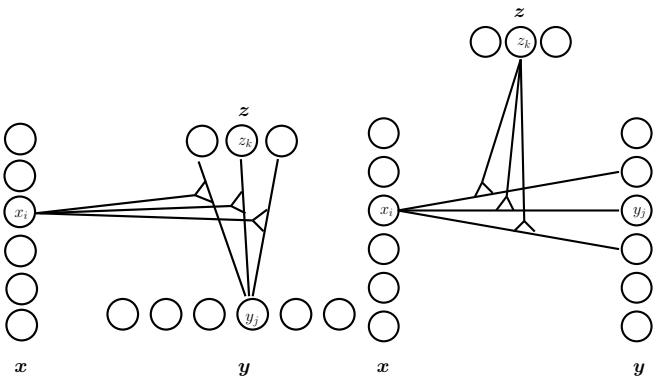

Fig. 5. Relating images using multiplicative interactions. Two views of the same model. **Left:** Input-modulated feature learning, **Right:** Mixture of warps.

Figure 5 (right) shows an alternative visualization of the same type of model, where each hidden variable can blend in a slice $w_{..k}$ of the parameter tensor. Each slice, in turn, is a matrix connecting an "input" pixel from image $\boldsymbol{x}$ to an "output"-pixel from image $\boldsymbol{y}$. We can think of this matrix as performing linear regression in the space of stacked gray-value intensities. A linear transformation in "pixel-space" is commonly known as a "warp". Although linear in pixel-space, a warp can be a highly non-linear transformation in image coordinates. Note also that a warp has a very large number of parameters in comparison to an affine image transformation. The model in Figure 5 (right) may be thought of as defining a *mixture of warps*.

In both cases, hidden variables take on the roles of dynamic mapping units [1], [2] which encode the relationship not the content of the images. Each unit in the model can gate connections between other variables in the model. We shall refer to this type of model as "gated feature learning", "cross-correlation", or "multiview feature learning" model in the following.

Like in a standard feature learning model one needs to include biases in practice. The set of model parameters thus consists of the three-way parameters w_{ijk}, as well as of single-node parameters w_i, w_j and w_k. One could also include "higher-order-biases" [16] like w_{ik}, which connect two groups of variables, but it is not common to do so. Like before, we shall drop all bias terms in what follows in order to avoid clutter. Both simple biases and higher-order biases can be implemented by introducing appropriate constant-1 dimensions to the images or the hidden variables.

2.3 Inference

The graphical model for gated feature learning is tripartite. That of a standard feature learning model is bipartite. As a result, inference can be performed in almost the same way as in a standard feature learning model, whenever two out of three groups of variables have been observed, as we show now.

Consider the task of inferring $\boldsymbol{z}$, given $\boldsymbol{x}$ and $\boldsymbol{y}$. Recall that for a standard feature learning model we have: $z_k = \sum_j w_{jk} y_j$ (up to component-wise non-linearities like the sigmoid). Formally, we may think of the gated feature learning model as turning the weights into a *linear function* of $\boldsymbol{x}$:

$$w_{jk}(\boldsymbol{x}) = \sum_{\boldsymbol{i}} w_{\boldsymbol{i}jk} x_{\boldsymbol{i}} \tag{11}$$

so that the inference equation becomes

$$z_k = \sum_j w_{jk}(\boldsymbol{x}) y_j = \sum_j \big(\sum_i w_{ijk} x_i \big) y_j = \sum_{ij} w_{ijk} x_i y_j \tag{12}$$

which amounts to computing, for each hidden z_k, a quadratic form in $\boldsymbol{x}$ and $\boldsymbol{y}$. The quadratic form is defined by the weight tensor $w_{..k}$. Thus, we can think of inference in a gated feature learning model either as computing a (quadratic) function of two images, or as standard inference for a single image, $\boldsymbol{y}$, where parameters are linearly dependent on another image, $\boldsymbol{x}$. We shall refer to the latter as *"predictive coding"*, because we can think of the model as predicting $\boldsymbol{y}$ from $\boldsymbol{x}$ via $\boldsymbol{z}$. The fact that inference may be interpreted in multiple ways is common in models with bi-linear dependencies [17].

Note that Eq. 12 is symmetric in $\boldsymbol{x}$ and $\boldsymbol{y}$. We could therefore, in principle, switch their roles in Eqs. 11 and 12 and define the linear parameters (Eq. 11) as a function of $\boldsymbol{y}$ rather than $\boldsymbol{x}$. During training, however, it has been common to drop this symmetry and to declare one of the two images as the gating "input" and the other as the "output", as we shall show.

The *meaning* of the hidden variables differs from that in standard feature learning models despite the similarity of inference: In standard feature learning, $\boldsymbol{z}$ constitutes a representation of the input image, in gated feature learning $\boldsymbol{z}$ represents the *transformation* that takes $\boldsymbol{x}$ to $\boldsymbol{y}$.

Inferring $\boldsymbol{y}$, given $\boldsymbol{x}$ and $\boldsymbol{z}$ yields the analogous expression:

$$y_j = \sum_k w_{jk}(\boldsymbol{x}) z_k = \sum_k \big(\sum_i w_{ijk} x_i \big) z_k = \sum_{ik} w_{ijk} x_i z_k \tag{13}$$

which again amounts to computing a quadratic form. The meaning of $\boldsymbol{y}$ is now "$\boldsymbol{x}$ transformed according to the known transformation $\boldsymbol{z}$". Likewise, we could compute $\boldsymbol{x}$ given $\boldsymbol{z}$ and $\boldsymbol{y}$ using an analogous equation, but, again, this would not be common if the model was trained asymmetrically (cf., Section 2.4.1).

It is important to note that for any given transformation $\boldsymbol{z}$, $\boldsymbol{y}$ is a linear function of $\boldsymbol{x}$, so it can be written

$$\boldsymbol{y} = L(\boldsymbol{z})\boldsymbol{x} \tag{14}$$

From Eq. 13 it follows that the entries of matrix $L(\boldsymbol{z})$ are given by

$$L_{ij}(\boldsymbol{z}) = \sum_k w_{ijk} z_k \tag{15}$$

When $\boldsymbol{x}$ and $\boldsymbol{y}$ are images, the linear function is a *warp*. So the hidden variables can be thought of as composing a warp from "basis warps" $w_{..k}$, in exactly the same way that feature learning models can be thought of as composing an image from basis images (features) $w_{.k}$ (cf., Eq 10). And whereas inferring a component z_k in a standard feature learning model amounts to computing the inner product between image $\boldsymbol{y}$ and feature $w_{.k}$, for gated feature learning it amounts to computing the inner product between $\boldsymbol{x}\boldsymbol{y}^T$ and the basis warp $w_{..k}$ (cf., Eqs. 9 and 12).

Besides computing hidden unit activities or generating fantasy data, a third common operation in many feature learning models is to compute a confidence value for new input data, which quantifies how well that data can be represented by the model. For this number to be useful, it has to be "calibrated", which is typically achieved by using a probabilistic model. In contrast to standard feature learning, training a probabilistic gated feature learning model can be slightly more complicated, because of the dependencies between $\boldsymbol{x}$ and $\boldsymbol{y}$ conditioned on $\boldsymbol{z}$. We shall discuss this issue in more detail in Section 2.4.2.

2.4 Learning

2.4.1 Predictive training

The training data for a multiview feature learning model consists of pairs of data-points $(\boldsymbol{x}^\alpha, \boldsymbol{y}^\alpha)$. Training is similar to standard feature learning, but there are some important differences. In particular, recall that the gated model may be viewed as a feature learning model whose input is the vectorized outer product $\boldsymbol{x}\boldsymbol{y}^{\mathrm{T}}$. Standard learning criteria, such as squared reconstruction error are obviously not appropriate for the outer product.

It is, however, possible to use squared error for training, albeit not on the products. To this end, consider the perspective from predictive coding, where the parameters of a feature learning model on $\boldsymbol{y}$ are modulated by an input image $\boldsymbol{x}$. This suggests deriving the learning criterion from the task of predicting $\boldsymbol{y}$ from $\boldsymbol{x}$ [15], [14], [16]. We shall first discuss this learning approach from the "inference-free" perspective (Eq. 1), and we shall subsequently discuss how using the linear inference equations (cf., Section 2.1) can simplify learning in direct analogy to standard feature learning.

The modulation of parameters in Eq. 11 is *case-dependent*, that is, each input example leads to a different model for $\boldsymbol{y}$. Learning can therefore be viewed as feature learning with case-dependent weights. In analogy to Eq. 1, we can write the reconstruction error that data-case $(\boldsymbol{x}^\alpha, \boldsymbol{y}^\alpha)$ contributes as

$$\sum_j \big(y_j^\alpha - \sum_{ik} w_{ijk} x_i^\alpha z_k^\alpha\big)^2 \tag{16}$$

By using Eq. 11 and by adding a sparsity penalty for $\boldsymbol{z}$, we can write the cost over the whole data-set also as

$$\sum_\alpha \sum_j \big(y_j^\alpha - \sum_k w_{jk}(\boldsymbol{x}^\alpha) z_k^\alpha\big)^2 + \lambda \sum_k |z_k^\alpha| \tag{17}$$

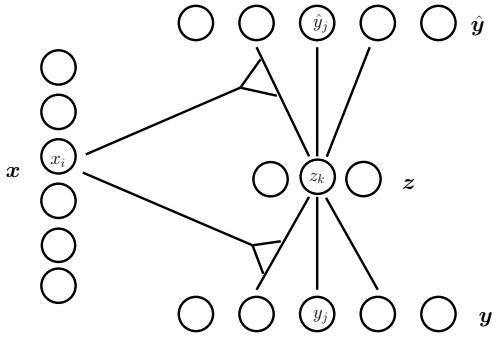

Fig. 6. A gated auto-encoder is an auto-encoder that learns to represent an image, $\boldsymbol{y}$, using parameters that are modulated by another image, $\boldsymbol{x}$. This makes it possible to learn relationships between $\boldsymbol{x}$ and $\boldsymbol{y}$ with gradient-based learning and back-prop.

which highlights the similarity with Eq. 1. Differentiating Eq. 16 with respect to w_{ijk} is the same as in a standard feature learning model. In particular, the model is still linear wrt. the parameters. Predictive learning is therefore possible with gradient-based optimization similar to standard feature learning (cf. Section 2.1).

However, in analogy to standard feature learning, it can be useful to use a feed-forward inference function to simplify inference and learning. A simple way to eliminate the hidden variables is by using an encoder-network as defined in Eq. 12 (possibly followed by a sigmoid nonlinearity) and by using a set of decoder parameters a_{ijk} to compute reconstructions as defined in Eq. 13. Like in standard feature learning one may tie decoder and encoder parameters by setting $a_{ijk} = w_{ijk}$.

This type of model is known as *gated autoencoder* [18], [19], and it is depicted in Figure 6. Learning is similar to learning a standard auto-encoder. This becomes obvious by plugging Eq. 12 into Eq. 16 and by noting that $\boldsymbol{x}$ is *fixed* in each training example. It is also possible to add multiple layers and to apply non-linearities to the hidden layers (in which case, of course, the derivatives will no longer be linear wrt. the parameters). Conditioning on $\boldsymbol{x}$ always ensures that the model is a directed acyclic graph, so one can use standard back-prop to compute derivatives.

As a second example of a gated feature learning model, we obtain the *gated Boltzmann machine* (GBM) by changing the energy function into the three-way energy [16]:

$$E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}) = -\sum_{ijk} w_{ijk} x_i y_j z_k \tag{18}$$

Exponentiating and normalizing yields the conditional probability over image $\boldsymbol{y}$ given $\boldsymbol{x}$:

$$p(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\big(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big), \tag{19}$$

$$Z(\boldsymbol{x}) = \sum_{\boldsymbol{y}, \boldsymbol{z}} \exp\big(-E(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\big) \tag{20}$$

Note that the normalization is over $\boldsymbol{y}$ and $\boldsymbol{z}$ only, which ensures that we obtain a conditional model of the output image $\boldsymbol{y}$. While it is possible to define a joint model over both images, this makes training more difficult (cf., Section 2.4.2).

Like in a standard RBM, maximum likelihood and contrastive divergence training involve sampling $\boldsymbol{z}$ and $\boldsymbol{y}$. In the gated Boltzmann machine, samples are drawn from the *conditional* distributions $p(\boldsymbol{y}|\boldsymbol{z},\boldsymbol{x})$ and $p(\boldsymbol{z}|\boldsymbol{y},\boldsymbol{x})$. Training the gated Boltzmann machine is like training a standard RBM, if we again utilize the fact that every input training pair defines a standard RBM whose parameters are defined as a (case-dependent) function of the input. For both the GBM and the gated autoencoder, one can include weight-decay terms and penalty terms to encourage hidden variable responses to be sparse.

2.4.2 Relational training

Modeling the *joint* distribution over two images, rather than the conditional distribution of one image given the other, can make it possible to perform image matching, by allowing us to quantify how compatible any two images are under to the trained model [20].

Formally, modeling the joint amounts simply to changing the normalization constant of the GBM to $Z = \sum_{\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}} \exp\big(-E(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\big)$ (cf. Eq. 20). Learning is more complicated as a result, however, because the view of input-dependent parameters no longer holds. Susskind et al. [20] show that it is possible to use a "three-way" version of contrastive divergence learning, where each iteration involves sampling $\boldsymbol{x}$ from $p(\boldsymbol{x}|\boldsymbol{h},\boldsymbol{y})$ and sampling $\boldsymbol{y}$ from $p(\boldsymbol{y}|\boldsymbol{h},\boldsymbol{x})$.

Another potential advantage of learning a symmetric model is that it allows us to model higher-order features for a single image, in other words, features that encode products of pixel intensities within the image. See, for example, [21] who train second-order features by using a joint version of a GBM with $\boldsymbol{x} = \boldsymbol{y}$. In contrast to [20], they use hybrid Monte Carlo for learning.

Alternatively, a gated auto-encoder can be turned into a symmetric model by defining the cost as the sum of two symmetric reconstruction costs:

$$\sum_j \big(y_j^\alpha - \sum_{ik} w_{ijk} x_i^\alpha z_k^\alpha\big)^2 + \sum_i \big(x_i^\alpha - \sum_{jk} w_{ijk} y_j^\alpha z_k^\alpha\big)^2 \quad (21)$$

This makes it possible to learn higher-order features in a non-probabilistic way using gradient descent [19]. For further approaches to learning higher-order within-image features see [22] and [23].

2.5 Toy example: Motion extraction and analogy making

An example of a gated Boltzmann machine applied to a motion inference task is shown in Figure 7. We trained a GBM on binary image pairs containing random dots, such that the output image $\boldsymbol{y}$ is a translated copy of the input image $\boldsymbol{x}$. Some example image pairs are shown in the two left-most columns of Figure 7 (a). The center column of Figure 7 (a) visualizes the corresponding inferred transformations as vector fields. To generate the vector-field, we first infer the linear warp from an image pair using Eqs. 12 and 15. Subsequently, we find for each input-pixel the output-pixel to which it is most strongly connected according to the inferred linear transformation, and we draw an arrow pointing from the input pixel to the output pixel. The plot shows that up to unpredictable edge effects, the model can correctly infer the translations inherent in the image pairs after being trained on shifts.

The two right-most columns in Figure 7 show how the inferred transformation can be applied to new images not seen during training *by analogy*. To this end, we apply the inferred linear transformation to the input test image using Eq. 13.

Figure 7 (b) shows a variation of this task, where the transformations are "split-screen" translations, that is, translations which are independent in the top half vs. the bottom half of the image. This example demonstrates how the model is able to decompose transformations into independent constituting transformations. This ability of the model is crucial for encoding natural videos, which contain a multitude of transformations as a result of a combination of many local transformations. We shall discuss the learning of natural video data in more details below.

2.6 A brief history of gating

Shortly after mapping units were introduced in 1981, energy models [3] received a lot of attention. Energy models apply *squaring non-linearities* to features and have therefore also been referred to as "square-pooling" models. Energy models have also been common as models of *complex cells* [10].

Since the square of a multiview (for example, binocular) feature can be shown to implicitly encode cross-products between simple (monocular) features, energy models are closely related to multiplicative feature learning models. In fact, energy models can be used to implement mapping units and vice versa. We discuss this relationship in detail in Sections 3 and 4. Early work on energy models suggested these as a way to encode motion by relating time frames in a video [3], and to perform stereo vision by relating images from different viewpoints [24], [25], [26]. An approach to learning-based disparity estimation was introduced later by [27].

In the early work on energy models, hard-wired Gabor features were used as the linear receptive fields instead of features that are learned from data [26], [25], [28]. The focus on Gabor features has somewhat biased the analysis of energy models to focus on the *Fourier-spectrum* as the main object of interest (see, for example, [28], [26]). As we shall discuss in Section 3, Fourier-components arise just as the special case of one transformation class, namely *translation*, and many important properties of these models apply also to other transformation classes.

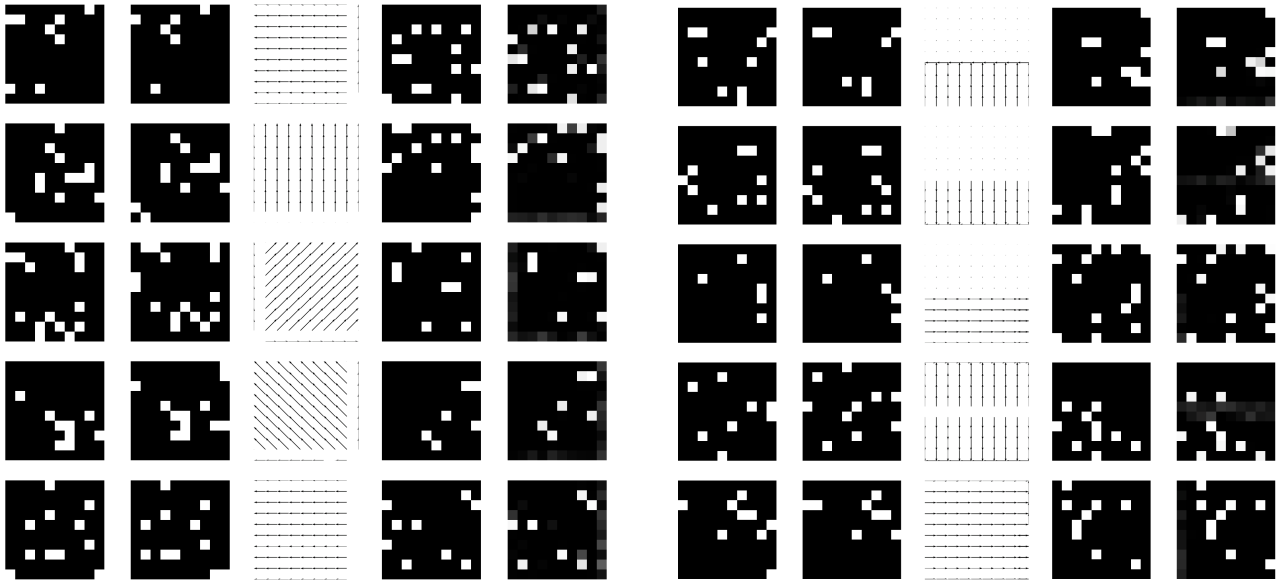

Fig. 7. Inferring transformations from test data. **(a)** Coherent motion across the whole image. **(b)** "Factorial motion" that is independent in different image regions. In both plots, the meaning of the five columns is as follows (left-to-right): Random test images x, random test images y, inferred flow-field, new test-image $\hat{x}$, inferred output $\hat{y}$.

Energy models based on Gabor features have also been applied to a single image. In this case they encode features independently of the Fourier-phase of the input and as a result, their responses are invariant to small translations as well as to contrast variations of the input (see, for example, [10]).

Shortly after energy and cross-correlation models emerged, there has been some interest in learning invariances with higher-order neural networks, which are neural networks trained on polynomial basis expansions of their inputs [29]. Higher-order neural networks can be composed of computational units that compute sums of products. These units are sometimes referred to as "Sigma-Pi-units" [30] (where "Pi" stands for product and "Sigma" for sum). At the same time, multiplicative interactions have been explored also as an approach to building distributed representations of symbolic data (for example, [31], [32]).

In 1995, Kohonen introduced the "Adaptive Subspace Self-Organizing Map" (ASSOM) [33], which computes sums over squared filter responses to represent data. Like the energy model, the ASSOM is based on the idea that the sum of squared responses is invariant to various properties of its inputs. In contrast to the early energy models, the ASSOM is trained from data. Inspired by the ASSOM, "Independent Subspace Analysis" (ISA) was introduced by [23], who place the idea in the context of more conventional feature learning models. Shortly thereafter, extensions of this work showed how the grouping of squared filter responses can be used to learn topographic feature maps [34], [35].

In a parallel line of work, bi-linear models have been proposed at approximately the same time as an approach to learning in the presence of multiplicative interactions [17]. The early work on bi-linear models used these as global models trained on whole images rather than using local receptive fields. In contrast to more recent approaches to learning with multiplicative interactions, training involved filling a two-dimensional grid with data that shows two types of variability (referred to as "style" and "content"). The purpose of bi-linear models is then to untangle the two degrees of freedom in the data. More recent work does not make this distinction, and the purpose of multiplicative hidden variables is merely to capture the multiple ways in which two images can be related. The work by [15], [14] or [16], for example, shows how multiplicative interactions make it possible to model the multitude of relationships between frames in natural videos, or between artificially transformed images [16]. An earlier multiplicative interaction model, that is also related to bi-linear models, is the "routing-circuit" [36].

Multiplicative interactions have also been used to model structure within static images, which can be thought of as modeling higher-order relations, and, in particular, pair-wise products, between pixel intensities (for example, [22], [23], [37], [38], [39], [40]).

3 Factorization and Energy Models

In the following, we discuss the close relationship between gated feature learning models and energy models. To this end, we first describe how parameter factorization makes it possible to pre-process input images and thereby reduce the number of parameters.

3.1 Factorizing the gating parameters

The number of gating parameters is roughly cubic in the number of pixels, if we assume that the number of constituting transformations is about the same as the number of pixels. It can easily be more for highly over-complete hiddens. One way to reduce that number is by factorizing the parameter tensor W into three matrices, such that each component w_{ijk} is given by a "three-way inner product" [41]:

$$w_{ijk} = \sum_{ijk} \sum_{f=1}^{F} w^x_{if} w^y_{jf} w^z_{kf} \tag{22}$$

Here, F is a number of hidden "factors", which, like the number K of hidden units, has to be chosen by hand or by cross-validation. The matrices w^x, w^y and w^z are $I \times F$, $J \times F$ and $K \times F$, respectively.

An illustration of this factorization is given in Figure 8 (top). It is interesting to note that, under this factorization, the activity of output-variable y_j, by using the distributive law, can be written:

$$\begin{aligned} y_j &= \sum_{ik} w_{ijk} x_i z_k = \sum_{ik} (\sum_f w^x_{if} w^y_{jf} w^z_{kf}) x_i z_k \\ &= \sum_f w^y_{jf} (\sum_i w^x_{if} x_i)(\sum_k w^z_{kf} z_k) \end{aligned} \tag{23}$$

Similarly, for z_k we have

$$\begin{aligned} z_k &= \sum_{ij} w_{ijk} x_i y_j = \sum_{ij} (\sum_f w^x_{if} w^y_{jf} w^z_{kf}) x_i y_j \\ &= \sum_f w^z_{kf} (\sum_i w^x_{if} x_i)(\sum_j w^y_{jf} y_j) \end{aligned} \tag{24}$$

One can obtain a similar expression for the energy in a gated Boltzmann machine. Eq. 24 shows that factorization can be viewed as *filter matching*: For inference, each group of variables $\boldsymbol{x}$, $\boldsymbol{y}$ and $\boldsymbol{z}$ are projected onto linear basis functions which are subsequently multiplied, as illustrated in Figure 8 (bottom).

It is important to note that the way factorization reduces parameters is *not* necessarily by projecting data into a lower-dimensional space before computing the multiplicative interactions – a claim that can be found frequently in the literature. In fact, frequently, F is chosen to be *larger* than I and/or J. The way that factorization reduces the number of parameters is by restricting the *three-way connectivity*: That is, with factorization, the number of pair-wise products is equal to the number of factors rather than equal to the number of pixels squared. Learning then amounts to finding basis functions that can deal with this restriction optimally.

All gated feature learning models can be subjected to this factorization. Training is similar to training an unfactored model, which can be seen by using the chain rule and differentiating Eq. 22. An example of a factored gated auto-encoder is described in [19]. Virtually all factored models that were introduced use the restriction of single multiplicative interactions (Eq. 22). An open research question is to what degree a less restrictive connectivity – equivalently, using a non-diagonal core-tensor in the factorization – would be advantageous.

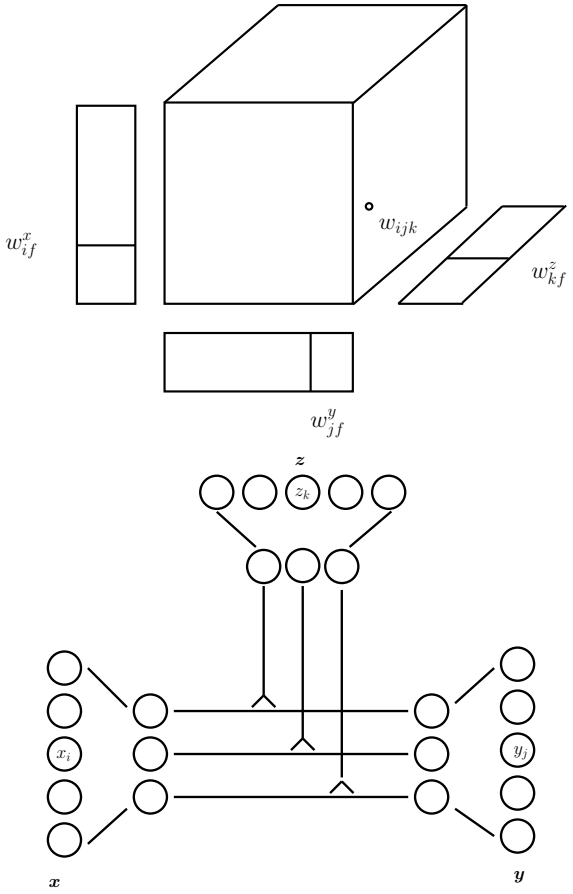

Fig. 8. **Top:** Factorizing the parameter tensor. **Bottom:** Interpreting factorization as filter matching.

Factored models have empirically been shown to learn filter-pairs that optimally represent transformation classes, such as Fourier-components for translations and a polar variant of Fourier-components for rotations [41]. In contrast to the dictionaries learned with standard feature learning methods, the filters always come in pairs. These may be referred to as "predictionary" as they are often learned using predictive training (cf., Section 2.4).

Figures 9 and 10 show examples of predictionaries learned from translations, affine transformations, split-screen translations, which are independent translations in the top and bottom half of the image, and natural video. For training the filters in the top rows and on the bottom right, we used data-sets described in [41] and [19] and the model described in [19]. The filters resemble receptive fields found in various cells in visual cortex [42]. To obtain split-screen filters (bottom left) we generated a data-set of split-screen translations and trained the model described in [41]. In Section 4, we provide an analysis that sheds some light onto why the filters take on this form.

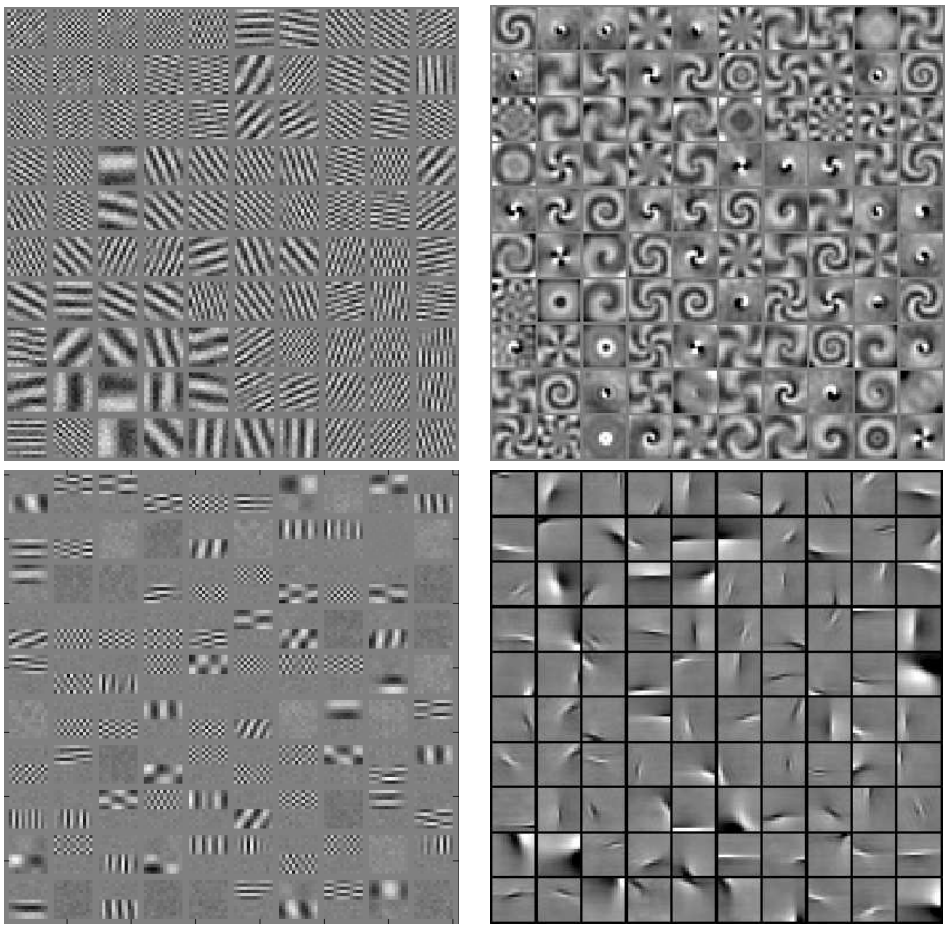

Fig. 9. Input filters learned from various types of transformation. **Top-left:** Translation, **Top-right:** Rotation, **Bottom-left:** split-screen translation, **Bottom-right:** Natural videos. See figure 10 on the next page for corresponding output filters.

In practice, it is common to utilize a set of tricks to improve stability and efficiency of learning. It is common, for example, to normalize output filter matrices w^x and w^y during learning, such that all filters $w^x_{\cdot f}$ and $w^y_{\cdot f}$ grow slowly and maintain roughly the same length as learning progresses. This is typically achieved by re-normalizing filters after each parameter update (see, for example, [20], [39]). It is also common to connect hidden units locally to the factors, rather than using full connectivity. A slightly more complicated approach is to let all hidden units populate a virtual "grid" in a low-dimensional space (for example, 2-D) and to connect hidden units to factors, such that neighboring hidden units are connected to the same or to overlapping sets of factors. This typically leads to topographic organization of filters, an example of which is the set of shift filters shown in Figures 9 and 10. This approach is also common for learning energy models on still images (for example, [34], [35]). Finally, it is common to train the models using image patches that are DC centered and contrast normalized, and usually also whitened. For a quantitative comparisons of several variations of gated feature learning models see [43].

3.2 Energy models

Energy models [3], [24] are an alternative approach to modeling image motion and disparities, and they have been deployed monocularly, too. A main application of energy models has been the detection of small translational motion in image pairs. This makes them suitable as biologically plausible mechanisms of both local motion estimation and binocular disparity estimation. Energy models detect motion by projecting two images onto two phase-shifted Gabor functions each (for a total of four basis function responses). The two responses *across* the images are added and squared. The sum of these two squared, spatio-temporal responses then yields the response of the energy model.

The rationale behind the energy model is that, since each within-image Gabor filter pair can be thought of as a localized spatio-temporal Fourier component, the sum of the squared components yields an estimate of spectral energy, which is not dependent of the *phase* – and thus to some degree not dependent on the *content* – of the input images. The two filters within each image need to be sine/cosine pairs, which is commonly referred to as being "in quadrature".

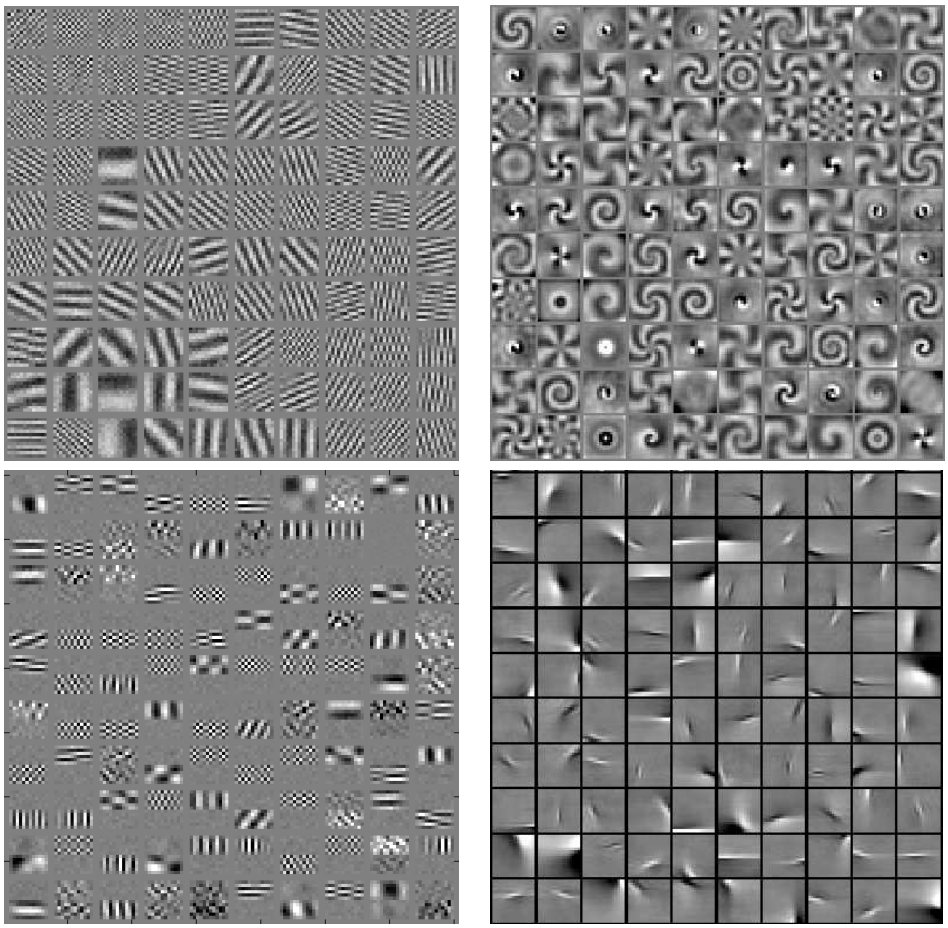

Fig. 10. Output filters learned from various types of transformation. **Top-left:** Translation, **Top-right:** Rotation, **Bottom-left:** split-screen translation, **Bottom-right:** Natural videos. See figure 9 on the previous page for corresponding input filters.

A detector of local shift can be built by using a set of energy models tuned to different frequencies. To turn a set of energy models into an estimate of local translation, one can, for example, read off the shift from the model with the strongest response [25], [26], or use pooling to get a more stable estimate [28].

In their early approach to *learning* energy models from training data, Hyvarinen and Hoyer [23] suggest extending a standard feature learning model by introducing an elementwise squaring operation and adding a linear pooling layer. For learning, they suggest adapting ICA by forcing latent variable responses (which are now sums of squared basis function responses) to be sparse, while keeping the filters orthogonal to avoid degenerate solutions. This approach is known as "Independent Subspace Analysis" (ISA) [23]. ISA was introduced initially to model single images not pairs, but it is possible to apply it to the concatenation of multiple images, like the early versions of the energy model ([3], [24]). In contrast to the early models, it is common to use ISA models that pool over more than two filters, and pooling weights can be learned along with the filters, instead of being fixed to one. Figure 11 shows an illustration of ISA applied to an image pair. As the figure shows, the model can be viewed as a two-layer network, with a hidden layer that uses an elementwise squaring nonlinearity. The first hidden layer of an energy model model is closely to the latent "factors" of a factored GBM as we shall show. Both ISA and factored gated Boltzmann machines were recently shown independently to yield state-of-the-art performance in various motion recognition tasks [44], [45].

3.3 Relationship between gated feature learning and energy models

Learning energy models, such as ISA, on the concatenation of two inputs $\boldsymbol{x}$ and $\boldsymbol{y}$ is closely related to learning gated feature learning models. Let $w^x_{\cdot f}$ ($w^y_{\cdot f}$) denote the set of weights connecting part $\boldsymbol{x}$ ($\boldsymbol{y}$) of the concatenated input with factor f (cf. Figure 11). The activity of hidden

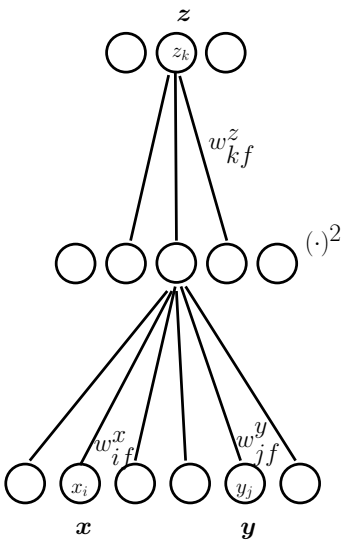

Fig. 11. Illustration of Independent Subspace Analysis (ISA) applied to an image pair $(\boldsymbol{x}, \boldsymbol{y})$.

unit z_k in the energy model is given by

$$\begin{aligned} z_k &= \sum_f w^z_{kf} \left({w^x_{\cdot f}}^{\mathrm{T}} \boldsymbol{x} + {w^y_{\cdot f}}^{\mathrm{T}} \boldsymbol{y} \right)^2 \\ &= \sum_f w^z_{kf} \left(2({w^x_{\cdot f}}^{\mathrm{T}} \boldsymbol{x})({w^y_{\cdot f}}^{\mathrm{T}} \boldsymbol{y}) + ({w^x_{\cdot f}}^{\mathrm{T}} \boldsymbol{x})^2 + ({w^y_{\cdot f}}^{\mathrm{T}} \boldsymbol{y})^2 \right) \end{aligned} \tag{25}$$

Up to the quadratic terms in Eq. 25, hidden unit activities are the same as in a gated feature learning model (cf., Eq. 24). As we shall discuss in detail below, the quadratic terms do not have a significant effect on the meaning of the hidden units. The hidden units in an energy model may therefore be interpreted as a way to implement mapping units which encode relations. See also [28], for a discussion of this relationship in the context of the traditional energy models.

4 Relational codes and simultaneous eigenspaces

We now show that hidden variables learn to detect subspace-rotations when they are trained on transformed image pairs. In Section 2.3 (Eq. 14) we showed that transformation codes $\boldsymbol{z}$ represent linear transformations, L, that is $\boldsymbol{y} = L\boldsymbol{x}$. We shall restrict our attention in the following to transformations that are orthogonal, that is, $L^{\mathrm{T}}L = LL^{\mathrm{T}} = I$, where I is the identity matrix. In other words, $L^{-1} = L^{\mathrm{T}}$. Note that practically all relevant spatial transformations, like translation, rotation or local shifts, can be expressed approximately as an orthogonal warp, because orthogonal transformations subsume, in particular, all *permutations* ("shuffling pixels").

An important fact about orthogonal matrices is that the eigen-decomposition $L = UDU^{\mathrm{T}}$ is complex, where eigenvalues (diagonal of D) have absolute value 1 [46]. Multiplying by a complex number with absolute value 1 amounts to performing a rotation in the complex plane, as illustrated in Figure 12 (left). Each eigenspace associated with L is also referred to as *invariant subspace* of L (as the application of L will keep the eigenvectors within the subspace).

Applying an orthogonal warp is thus equivalent to (i) projecting the image onto *filter pairs* (the real and imaginary parts of each eigenvector), (ii) performing a rotation within each invariant subspace, and (iii) projecting back into the image-space. In other words, we can decompose an orthogonal transformation into a set of independent, 2-dimensional rotations. The most well-known examples are translations: A 1D-translation matrix contains ones along one of its secondary diagonals, and it is zero elsewhere.[2] The eigenvectors of this matrix are Fourier-components [47], and the rotation in each invariant subspace amounts to a phase-shift of the corresponding Fourier-feature. This leaves the norm of the projections onto the Fourier-components (the power spectrum of the signal) constant, which is a well known property of translation.

It is interesting to note that the imaginary and real parts of the eigenvectors of a translation matrix correspond to sine and cosine features, respectively, reflecting the fact that Fourier components naturally come in *pairs*. These are commonly referred to as *quadrature pairs* in the literature. In the special case of Gabor features, the importance of quadrature pairs is that they allow us to detect translations independently of the local content of the images [26], [28]. However, the property that eigenvectors come in *pairs* is not specific to translations. It is shared by all transformations that can be represented by an orthogonal matrix, so that they can be composed from 2-dimensional rotations. Bethge et al. [48] use the term *"generalized quadrature pair"* to refer to the eigenfeatures of these transformations.

4.1 Commuting warps share eigenspaces

An observation that is central to our analysis is that eigenspaces can be *shared* among transformations. When eigenspaces are shared, then the only way in which two transformations differ, is in the angles of rotation within the eigenspaces. That way, shared eigenspaces allow us to represent *multiple transformations with a single set of features*. An example of a shared eigenspace is the Fourier-basis, which is shared among translations. This well-known observation follows from the fact that the set of all circulant matrices (which are 1-D translation-matrices) of the same size have the Fourier-basis as eigen-basis [47]. However, eigenspaces can be shared between other transformations. An obvious generalization is local translation, which may be considered the constituting transformations of natural videos. Another, less obvious generalization is spatial rotation. Formally,

2. To be exactly orthogonal it has to contain an additional one in another place, so that it performs a rotation with wrap-around.

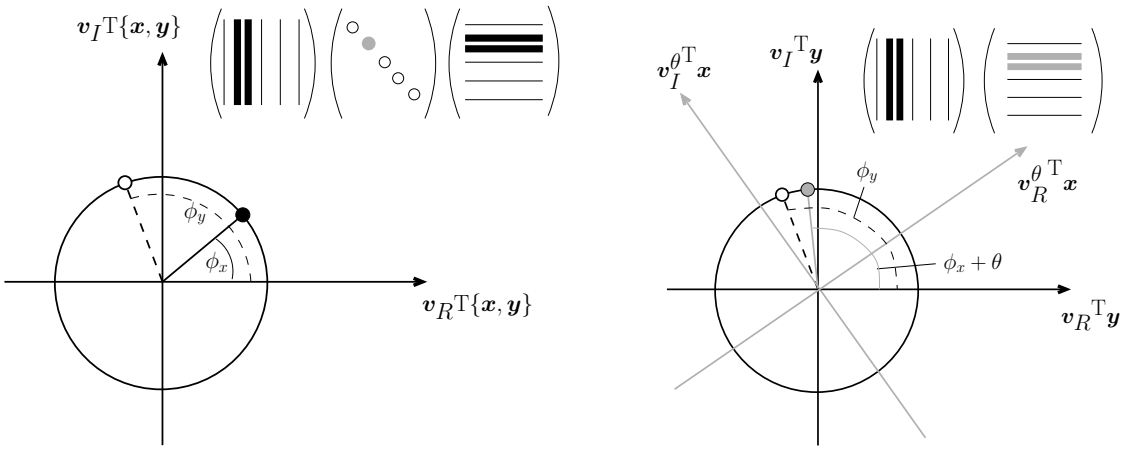

Fig. 12. **Left:** Inference in a gated feature learning model is equivalent to extracting rotation angles from two-dimensional invariant subspaces. **Right:** By absorbing eigenvalues into the eigenvectors, a mapping unit can learn to detect rotations by a particular angle (its preferred angle): the inner product between the projections of the two images $\boldsymbol{x}$ and $\boldsymbol{y}$ in the figure will be maximal when $\phi_y = \phi_x + \theta$, that is, when the rotation that the detector applies to $\boldsymbol{x}$ has the effect of aligning $\boldsymbol{x}$ with $\boldsymbol{y}$.

two matrices A, B share eigenvectors if they *commute*, that is if $AB = BA$ holds [46].[3] As an example, consider translations: translating an image by a pixels to the left and then by b pixels upwards yields the same result as first translating it upwards and then to the left, showing that translations commute.

The importance of commuting transformations for our analysis is that, since these transformations share an eigen-basis, they differ only wrt. the angle of rotation in the joint eigenspace. As a result, we may extract a particular transformation from a given image pair $(\boldsymbol{x}, \boldsymbol{y})$ by recovering the angles of rotation between the projections of $\boldsymbol{x}$ and $\boldsymbol{y}$ onto the eigenspaces. To this end, consider the real and complex parts $\boldsymbol{v}_{\mathrm{R}}$ and $\boldsymbol{v}_{\mathrm{I}}$ of some eigen-feature $\boldsymbol{v}$. That is, $\boldsymbol{v} = \boldsymbol{v}_{\mathrm{R}} + i\boldsymbol{v}_{\mathrm{I}}$, where $i = \sqrt{-1}$. The real and imaginary coordinates of the projection of $\boldsymbol{x}$ onto the invariant subspace associated with $\boldsymbol{v}$ are given by $\boldsymbol{v}_{\mathrm{R}}^{\mathrm{T}}\boldsymbol{x}$ and $\boldsymbol{v}_{\mathrm{I}}^{\mathrm{T}}\boldsymbol{x}$, respectively. For the output image, they are $\boldsymbol{v}_{\mathrm{R}}^{\mathrm{T}}\boldsymbol{y}$ and $\boldsymbol{v}_{\mathrm{I}}^{\mathrm{T}}\boldsymbol{y}$.

Let ϕ_x and ϕ_y denote the angles of the projections of $\boldsymbol{x}$ and $\boldsymbol{y}$ with the real axis in the complex plane. If we normalize the projections to have unit norm, then the cosine of the angle between the projections, $\phi_y - \phi_x$, may be written

$$\cos(\phi_y - \phi_x) = \cos\phi_y \cos\phi_x + \sin\phi_y \sin\phi_x$$

by trigonometric identity. This is equivalent to computing the inner product between two normalized projections (cf. Figure 12 (left)). In other words, to estimate the (cosine of) the angle of rotation between the projections of $\boldsymbol{x}$ and $\boldsymbol{y}$, we need to *sum over the product of two filter responses*.

3. This can be seen by considering any two matrices A and B with $AB = BA$ and with λ, v an eigenvalue/eigenvector pair of B with multiplicity one. It holds that $BAv = ABv = \lambda Av$. Therefore, Av is also an eigenvector of B with the same eigenvalue.

Note, however, that normalizing each projection to 1 amounts to dividing by the sum of squared filter responses, an operation that is highly unstable if a projection is close to zero. This will be the case, whenever one of the images is almost orthogonal to the invariant subspace. This, in turn, means that the rotation angle *cannot be recovered from the given image*, because the image is too close to the axis of rotation. One may view this as a subspace-generalization of the well-known *aperture problem* beyond translation, to the set of orthogonal transformations. Normalization would ignore this problem and provide the illusion of a recovered angle even when the aperture problem makes the detection of the transformation component impossible. In the next section we discuss how gated feature learning overcomes this problem, by allowing us to treat the problem as a *rotation detection* task instead.

4.2 Representing transformations by detecting subspace rotations

For each eigenvector, $\boldsymbol{v}$, and rotation angle, θ, define the complex filter

$$\boldsymbol{v}^{\theta} = \exp(i\theta)\boldsymbol{v}$$

which represents a projection and simultaneous rotation by θ. This amounts to *absorbing* the rotation, as given by some eigenvalue, into the eigenvector itself, allowing us to define a subspace rotation-detector with preferred angle θ as follows:

$$r^{\theta} = (\boldsymbol{v}_R^{\mathrm{T}}\boldsymbol{y})(\boldsymbol{v}_R^{\theta\,\mathrm{T}}\boldsymbol{x}) + (\boldsymbol{v}_I^{\mathrm{T}}\boldsymbol{y})(\boldsymbol{v}_I^{\theta\,\mathrm{T}}\boldsymbol{x}) \tag{26}$$

Like before, if projections would be normalized to length 1, we would have

$$r^{\theta} = \cos\phi_y \cos(\phi_x+\theta) + \sin\phi_y \sin(\phi_x+\theta) = \cos(\phi_y - \phi_x - \theta),$$

which would be maximal whenever $\phi_y - \phi_x = \theta$, thus when the observed angle of rotation, $\phi_y - \phi_x$, is equal to the preferred angle of rotation, θ. An illustration of such a rotation detector is given in Figure 12 (right).

Although normalizing projections is not a good idea due to the subspace aperture problem, normalization turns out not to be necessary, if the task is defined as a detection task. In particular, note that if features and data are contrast normalized, then the subspace inner product (Eq. 26) will yield a strong response if the following two conditions are met:

- the angle between the projections of $\boldsymbol{x}$ and $\boldsymbol{y}$ matches the detector's preferred angle, θ,
- the projections of the images onto the invariant subspace are large enough (in other words, the images are sufficiently well-aligned with the subspace).

The second condition implies that the output of the detector defined in Eq. 26 factors in not only the presence of a transformation but also its ability to discern it. In other words, when the detector fires, we know its preferred transformation is present. When it does not fire, then either the transformation is not present, or it is present but invisible to the detector because the images are not well-aligned with its invariant subspace.

However, when a transformation that is present is invisible to a detector, then a different detector defined over a different invariant subspace may still be able to observe the transformation. Thus, a population of detectors can yield a robust representation of transformations. As an an example, consider two images $\boldsymbol{x}$ and $\boldsymbol{y}$ related through vertical translation. The invariant subspaces of translations are spanned by Fourier components. Vertical translation, in particular, is represented by phase-shifts of horizontal Fourier components. Now consider a rotation detector defined in a horizontal Fourier component of a particular frequency. If the images happen to lack this horizontal frequency, then the detector will be silent, even if its transformation is present. Since vertical translation, however, is detectable not only in one but in many subspaces (such as in the other horizontal Fourier components of different frequencies), detectors defined over those subspaces may still be able to detect the transformation. In fact, if we assume that detectors for all subspaces (horizontal frequencies) are present, then the only way that no detector fires, would be if the transformation is not present.

One way to combine the information from multiple detectors into a single representation of a transformation is by *pooling*, because a sum (or weighted sum) over multiple detectors will be able to represent the event that any of the detectors fires. That way, a weighted sum over rotation detectors can yield a representation of transformations that is not dependent of the content of the images (such as the frequency content in the case of translation).

Formally, if we stack imaginary and real eigenvector pairs for the input and output images, $\boldsymbol{v}$ and $\boldsymbol{v}^\theta$, column-wise in matrices V and U, respectively, we may define the representation $\boldsymbol{t}$ of a transformation, given two images $\boldsymbol{x}$ and $\boldsymbol{y}$, as:

$$\boldsymbol{t} = W^{\mathrm{T}} P \left(U^{\mathrm{T}}\boldsymbol{x}\right) \cdot \left(V^{\mathrm{T}}\boldsymbol{y}\right) \tag{27}$$

where W is an appropriate "across-subspace" pooling matrix, and P is a band-diagonal "within-subspace" pooling matrix that defines the two-dimensional inner product in Equation 26.[4]

Note that Eq. 27 takes exactly the same form as inference in a factored gated feature learning model (cf., Eq. 24), if we absorb the within-subspace pooling matrix P into W.

This shows that we may interpret mapping units in a gated feature learning model as a way to encode transformations by *representing rotation angles in invariant subspaces*.

4.3 Learning as simultaneous diagonalization

The interpretation of mapping units as encodings of rotations relies on several assumptions:

- Images $\boldsymbol{x}$ and $\boldsymbol{y}$ are contrast-normalized.
- The columns of both U and V come in pairs, each of which spans a two-dimensional invariant subspace of a transformation class. (Formally, the pairs represent the real and imaginary components of some complex eigen-vector of the transformation class.).
- Corresponding filter pairs in U and V are related through rotations only. In other words, for each filter pair $\boldsymbol{v}_f$ in V there exists θ such that the corresponding filter pair in U can be written $\boldsymbol{v}_f^\theta = \exp(i\theta)\boldsymbol{v}_f$.

While it would be possible in principle to define filter pairs with these properties by hand in order to represent a given transformation class, model parameters can be learned from image pairs as we discussed in Sections 2 and 3. In particular, if we interpret the inference equations in a factored model (Eqs. 24, 27) as constraints under which we learn to represent the training image pairs, then it becomes clear that learning may be viewed as a way to find appropriate sets of filter pairs that are able to detect the rotations.

Learning a factored gated feature learning model, thus, has the effect of performing an *approximate simultaneous diagonalization of a set of transformations*. As we showed in Section 3, training on translations, for example, yields Fourier features, which indeed represent the invariant subspaces of translation. In addition to finding representations of the invariant subspaces, learning involves finding an appropriate across-subspace pooling matrix W.

In [49] it is shown empirically that, when a data-set contains more than one transformation class, learning can involve partitioning the set of observed transformations into commutative subsets and simultaneously diagonalizing each subset.

4. To this end, P has to contain exactly 2 ones along each row and has to be zero elsewhere.

It is important to note that, although complex arithmetic provides a convenient notational framework in the analysis, complex numbers are not typically used in any actual implementations. Learning yields filters which, implicitly, span the two-dimensional invariant subspaces of a transformation class. However, these subspaces do not need to be predefined as the real/imaginary components of a complex vector, nor do filter pairs need to be exactly in quadrature, as long as the learning can adapt them to represent two-dimensional rotations. As an analogy, consider training a linear auto-encoder to perform PCA: While the auto-encoder features will span the same subspace as the eigen-vectors of the data covariance matrix, they will typically not be exactly the same (in particular, the features will typically not be orthogonal). By making use of Eq. 27 rather than the more common Eq. 24 to define mapping unit activations (and thereby keeping within-subspace pooling and across-subspace pooling separate), it is nevertheless possible to visualize the learned "real" and "imaginary" components of the learned filters [49].

It is interesting to note that diagonalizing a single transformation would amount to performing a kind of canonical correlations analysis (CCA), so training a multiview feature learning model may be thought of as performing multiple (possibly nonlinear) canonical correlation analyzes with tied features. Note that learning such a "mixture" of related CCA models, rather than a single CCA model, is crucial in practical applications, because typically any image can potentially be transformed by many different transformations from the given class and not just by a single instance from that class. Similarly, modeling within-image structure by setting $\boldsymbol{x} = \boldsymbol{y}$ (cf., Section 2.4.2) is like learning a PCA mixture with tied weights.

4.4 Relation to energy models

By concatenating images $\boldsymbol{x}$ and $\boldsymbol{y}$, as well as filters $\boldsymbol{v}$ and $\boldsymbol{v}^\theta$, we may approximate the subspace rotation detector (Eq. 26) with the response of an energy detector:

$$\begin{aligned} r^\theta &= \big((\boldsymbol{v}_R{}^{\mathrm{T}}\boldsymbol{y}) + (\boldsymbol{v}_R^\theta{}^{\mathrm{T}}\boldsymbol{x})\big)^2 + \big((\boldsymbol{v}_I{}^{\mathrm{T}}\boldsymbol{y}) + (\boldsymbol{v}_I^\theta{}^{\mathrm{T}}\boldsymbol{x})\big)^2 \\ &= 2\big((\boldsymbol{v}_R{}^{\mathrm{T}}\boldsymbol{y})(\boldsymbol{v}_R^\theta{}^{\mathrm{T}}\boldsymbol{x}) + (\boldsymbol{v}_I{}^{\mathrm{T}}\boldsymbol{y})(\boldsymbol{v}_I^\theta{}^{\mathrm{T}}\boldsymbol{x})\big) \\ &\quad + (\boldsymbol{v}_R{}^{\mathrm{T}}\boldsymbol{y})^2 + (\boldsymbol{v}_R^\theta{}^{\mathrm{T}}\boldsymbol{x})^2 + (\boldsymbol{v}_I{}^{\mathrm{T}}\boldsymbol{y})^2 + (\boldsymbol{v}_I^\theta{}^{\mathrm{T}}\boldsymbol{x})^2 \end{aligned} \tag{28}$$

Eq. 28 is equivalent to Eq. 26 up to the four quadratic terms, which represent the squared norms of the projections of $\boldsymbol{x}$ and $\boldsymbol{y}$ onto the invariant subspace. Thus, like the norm of the projections, they contribute information about the discernibility of transformations. By adding the square terms, the detector will have a different tuning behavior than the inner product detector defined in Eq. 26. However, the energy detector still attains its peak response exactly when both images reside within its invariant subspace and when their projections are rotated by the detector's preferred angle θ, so it is tuned to the same transformation as the inner product detector. This shows that energy models applied to the concatenation of two images are well-suited to modeling transformations, too. The inner product detector (Eq. 26) is commonly referred to as cross-correlation model in the literature (for example, [28]).

4.5 Representing videos

Both energy models and cross-correlation models can be applied to sequences of more than two images. In a gated feature learning model, Eq. 26 may be modified to contain pair-wise products across all time steps, or all those that are deemed relevant (for example, products between adjacent frames).

In the energy model, Eq. 28 may be modified to compute the square of the concatenation of more than two images. In particular, consider the concatenation ("vectorization") $\boldsymbol{X}$ of a sequence of T frames $\boldsymbol{x}_t,\ t = 1, \ldots, T$, projected onto the concatenation of T corresponding feature vectors $\boldsymbol{v}_R^t$ and $\boldsymbol{v}_I^t,\ t = 1, \ldots, T$. The sum of squares of the projections onto these filters will yield an energy response of the form

$$\begin{aligned} r &= \big(\sum_t \boldsymbol{v}_R^t{}^{\mathrm{T}}\boldsymbol{x}_t\big)^2 + \big(\sum_t \boldsymbol{v}_I^t{}^{\mathrm{T}}\boldsymbol{x}_t\big)^2 \\ &= \Omega + \sum_{st}\Big[\big(\boldsymbol{v}_R^s{}^{\mathrm{T}}\boldsymbol{x}_s\big)\big(\boldsymbol{v}_R^t{}^{\mathrm{T}}\boldsymbol{x}_t\big) + \big(\boldsymbol{v}_I^{s\mathrm{T}}\boldsymbol{x}_s\big)\big(\boldsymbol{v}_I^t{}^{\mathrm{T}}\boldsymbol{x}_t\big)\Big] \end{aligned} \tag{29}$$

where we use Ω to denote all square terms, which like before represent subspace norms. Equation 29 shows that the energy response will implicitly contain the sum over *all* pair-wise products between projected frames, or equivalently, the sum over inner products between all pairs of two-dimensional projections of the data. Thus, for r to take on a meaningful value, all angles implicit in these inner products have to be consistent, in other words, they have to be multiples of the first angle.

A limitation of the energy detector for modeling video data is therefore that it can only detect the repeated application of a single transformation.

4.5.1 Example: Implementing an energy model via a cross-correlation model

The close relation between energy models and gated feature learning models makes it possible to implement one via the other. Figure 13 shows example filters from an energy model trained on concatenated frames from videos showing moving random dots.[5] We trained a gated auto-encoder with $F = 256$ factors and $K = 128$ mapping units, where $\boldsymbol{x} = \boldsymbol{y}$ is given by the concatenation of 10 frames. Filters are constrained, such that $w_{if}^x = w_{if}^y$. Each 10-frame input shows random dots moving at a constant speed. Speed and direction vary across movies.

Since the gated auto-encoder, a cross-correlation model, multiplies two sets of filter responses which are

5. Data and code are available at http://learning.cs.toronto.edu/~rfm/relational

the same, it effectively computes a square, and thus implements an energy model. In the absence of any within-image structure, all filters learn to represent only across-image correlations. Thus, as predicted by Eq. 25 the energy model, in turn, effectively implements a cross-correlation model.

Figure 13 depicts, separately, the 10 sets of 256 filters corresponding to the 10 time-frames. It shows that the model learns spatio-temporal Fourier features which are selective for speed, frequency and orientation.

5 Discussion

While energy and cross-correlation models have been well-known for almost twenty years, they have only recently begun to be applied sucessfully in many vision tasks, such as motion understanding and action recognition, where they now achieve state-of-the-art performance. We shall discuss some selected applications in the next section. Most of the applications make use of multi-layer or deep learning extensions of multiview feature learning. We shall discuss the relationship between gating and deep learning in detail in Section 5.2, and we shall point to some open research problems in Section 5.3.

5.1 A tour of recent applications

Action recognition is a prototypical example of a feature learning application that requires the ability to extract *inter-frame* representations to extract motion from videos. The state-of-the-art approaches to action recognition are based on multiview feature learning, either by making use of gating connections [45] or of energy models [44]. To yield state-of-the-art performance both types of model utilize multiple layers of features, so they combine the use of deep architectures with multiplicative gating. An empirical investigation of the usefulness of multiple layers is presented in [44].

Learning invariant features from images: It is interesting to note that invariant object recognition itself can be viewed as a correspondence problem, where the goal is to match an input observation to an invariant template in memory. An approach to invariant recognition based on a variation of a multiview feature learning model is proposed in [50]. The model can be considered as an approach to invariant recognition through modeling mappings that take images to class labels. The input to this model is an image, the output is an orthogonal encoding of a class label, and prediction amounts to marginalizing over the set of possible mappings. The model transforms an input into a canonical pose, so that it can be matched with a template, which itself represents the object in some canonical pose. As shown in [50], "swirly features" similar to the rotation features in Figures 9 and 10 emerge when learning to perform rotationally invariant recognition. The model was shown to achieve state-of-the-art performance in a variety of invariant recognition tasks.

Learning invariant features from videos: Common object recognition systems differ from the way humans learn about objects in that they are trained on static views instead of movies, which show objects move around or change their pose. A model that computes squares or cross-products can automatically learn to associate object identity with 3-D structure or other object properties, simply by being trained on multiple, concatenated frames. A recent application of this idea is the learning invariant features from videos [51]. In that work, a deep auto-encoder is combined with an energy model, which makes it possible to learn image features whose subspace energies stay constant across frames (see also [52] for a similar, probabilistic formulation of the same idea). A multi-layer version of the model was shown to yield state-of-the-art results in object recognition by using subspace energies as features. A similar approach to learning invariances was proposed in [49]. In contrast to [51], the work in [49] uses phase-differences rather than subspace energies to represent objects.

Tracking and pose estimation: While in this paper we focus on the use of gating units to learn about image relations, there has been some recent work on applying multiplicative interactions in other tasks and in other domains. Examples include [53], [54] who use multiplicative interactions to gate hidden state-transitions for tracking. The model by [54] is based on using three-way interactions to model the relationship between image regions [55]. Factored gating units have been applied also to the task of learning human pose dynamics from MOCAP data [56], and an extension of that work has been applied to tracking [57].

There exist several further recent applications of multiplicative interactions in a variety of domains. An example is the work by [58] who use gating connections to model occlusion with applications in de-noising. Another example is the work by [59] who use factored gating connections to let input data modulate the hidden state transitions in a recurrent neural network. They report strong improvements in character-level language modeling, both over conventional recurrent networks and over other state-of-the-art models on these tasks.

A general explanation of these recent successes of multiplicative interactions, which goes beyond our analysis on image relations, may be that they provide a simple way to increase model capacity: When a very large amount of training data is available relative to the size of the models that may be trained within a reasonable amount of time, multiplication and unconventional non-linearities can provide a computationally and memory-efficient way to increase the expressiveness of models.

5.2 Multiview features and deep learning

A gated feature learning module is inherently a multi-layer architecture, because it uses pooling over multiplicative combinations (or squares) of features. Building

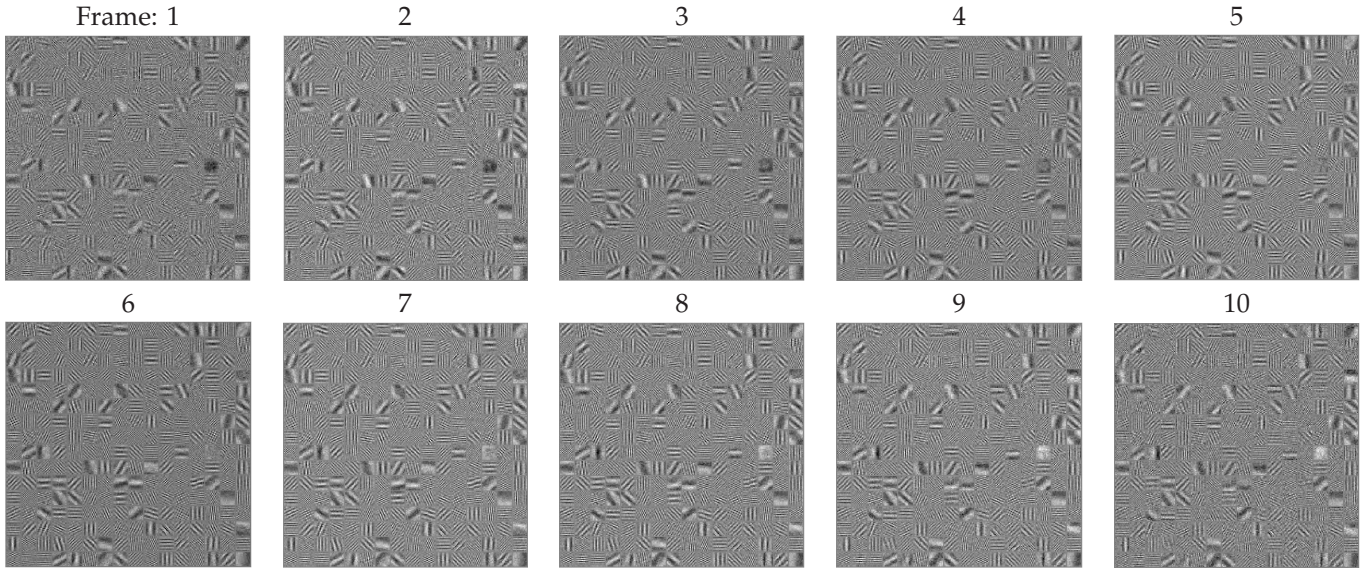

Fig. 13. Implementing a *cross-correlation model* via an *energy model* via a *cross-correlation model*. Sequence of filters learned from the concatenation of 10 frames of moving random dots.

deep models by stacking multiple layers of this type of module has shown to further benefit applications, similar to the learning of image features, as we discussed in Section 5.1. There are several subtle but important differences in the role of deep learning for learning multiview representations, however, and we shall discuss these in the following.

Our analysis in Section 4 raises the question to what degree our ability to learn about transformations is hampered by deviating from exactly commuting transformations and exactly orthogonal matrices. Existing experiments (for example in [16], [41]) suggest that there is some robustness wrt. deviations from exact orthogonality, but there has been no quantitative analysis. It is conceivable that one could pre-process input examples, such that they can be related through approximately orthogonal matrices to make them amenable to gating or energy models. This would obviously require additional processing layers. In that way, deep learning can play an enabling role for building certain relational feature learning models. What the exact requirements of extra layers should be in order to improve the learnability of transformations is an important research question. It seems conceivable that high-dimensional and sparse representations as well as topographic filter organization may play an important role, as as they may make it easier to express relations in terms of *sets of local translations*.

A direct implication for learning of (non-linear) visual feature hierarchies can be derived for spatial (for example, geometric) transformations of images: The fact that commutativity and orthogonality of the set of learned transformations must be preserved throughout the hierarchy imposes strong constraints on the learning of that hierarchy. One condition which seems to be well-suited (if not necessary) to preserving this structure is *retinotopy*. In a feature hierarchy that is composed of retinotopic maps, a local shift in the image, for example, will map to a corresponding local shift in a higher layer feature map. While retinotopic organization of features is well-known to hold in visual cortex, it has been much less common to explicitily enforce it in deep learning. Yet it may play a crucial role in keeping spatial relations intact.

A further implication of our analysis for the learning of deep architectures is that it suggests unconventional non-linearities, such as squaring or rectification, to be useful for building models and for extending their applicability beyond still images. From a biological point of view, multiplicative interactions may also be viewed as a conceptually simple approximation to more complex *dendritic computations* [60] than the common neuron abstraction used in practically all deep learning models. The use of more elaborate models of dendritic computation in deep learning is a wide-open field, although uncommon non-linearities, such as rectification, have begun to get an increasing amount of attention recently (for example, [61]).

5.3 Directions for further research

The current standard approach to solving correspondence tasks in vision is by matching similar image regions and by using robust statistics to deal with the typically large number of false positives [62]. Multiview feature learning differs from this type of approach in that it learns dense correspondences, albeit only in small image regions. The advantage of learning is that noise and false matches are dealt with automatically, since hidden units jointly clean up and make sense of imperfect correspondences in the data. The disadvantage is that, for computational reasons, learning has to be restricted to very small image regions. An interesting

research question is whether there is a bridge between multiview feature learning and solving sparse, global correspondence tasks in vision. Two ways in which multiview feature learning may be brought to use in such tasks are the following: (1) It may be possible to improve sparse matching by using similarity metrics derived from gated feature learning. This could help improve matching accuracy, and it could make it possible to deal with geometrical and other nuisance transforms through learning rather than hand-coding [63]. (2) It may be possible to increase the efficiency of gated feature learning on the scale of larger images by combining local receptive fields (factors) with mapping units that pool across factors defined at widely different image locations. To further reduce the connectivity one could use prior knowledge about likely transformations.

Using multiplicative interactions can also be related to analogy making [41]. It can be argued that analogy making is at the heart of many cognitive capabilities [64]. An interesting question is, in what way an analogy-making module may be a useful building block to solve tasks that are too difficult for the common deep architectures, which are composed of bi-partite feature learning modules. Since gated feature learning and energy models can be trained with standard, even Hebbian-like, learning (cf., Section 2.2), analogy-making does not require any uncommon or unusual machinery besides multiplicative interactions.

Squaring can be approximated using other non-linearities (for example, [13]). A possible research question is, what type of approximations of computing squares or cross-products may be advantageous computationally and/or more plausible biologically. Of course, squares could be approximated using a feed-forward network with sigmoid hidden unit activations [65]. However, the abundance of matching and correspondence tasks in vision does seem to provide an inductive bias in favor of genuine multiplicative interactions and/or squaring non-linearities.

Acknowledgements: This work was supported in part by the German Federal Ministry of Education and Research (BMBF) in the project 01GQ0841 (BFNT Frankfurt). We also thank the reviewers for several useful suggestions.

REFERENCES

[1] G. F. Hinton, "A parallel computation that assigns canonical object-based frames of reference," in *Proceedings of the 7th international joint conference on Artificial intelligence - Volume 2*, San Francisco, CA, USA, 1981, pp. 683–685.

[2] C. von der Malsburg, "The correlation theory of brain function," Max-Planck-Institut für Biophysikalische Chemie, Postfach 2841, 3400 Göttingen, FRG, Internal Report, 81-2, 1981, reprinted in E. Domany, J. L. van Hemmen, and K. Schulten, editors, *Models of Neural Networks II*, chapter 2, pages 95–119. Springer-Verlag, Berlin, 1994.

[3] E. Adelson and J. Bergen, "Spatiotemporal energy models for the perception of motion," *J. Opt. Soc. Am. A*, vol. 2, no. 2, pp. 284–299, 1985.

[4] P. Smolensky, "Parallel distributed processing: explorations in the microstructure of cognition, vol. 1," D. E. Rumelhart, J. L. McClelland, and C. PDP Research Group, Eds. Cambridge, MA, USA: MIT Press, 1986, ch. Information processing in dynamical systems: foundations of harmony theory, pp. 194–281.

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

[6] B. Olshausen and D. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images." *Nature*, vol. 381, no. 6583, pp. 607–609, June 1996.

[7] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," ser. ICML '08. New York, NY, USA: ACM, 2008.

[8] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Computation*, vol. 14, no. 8, pp. 1771–1800, 2002.

[9] G. Hinton, "A Practical Guide to Training Restricted Boltzmann Machines," Tech. Rep., 2010.

[10] A. Hyvarinen, J. Hurri, , and P. O. Hoyer, *Natural Image Statistics: A Probabilistic Approach to Early Computational Vision*. Springer Verlag, 2009.

[11] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, pp. 1527–1554, July 2006.

[12] N. Troje and H. Bülthoff, "Face recognition under varying poses: The role of texture and shape," *Vision Research*, vol. 36, no. 12, pp. 1761–1771, 6 1996.

[13] C. Zetzsche and U. Nuding, "Nonlinear and higher-order approaches to the encoding of natural scenes." *Network (Bristol, England)*, vol. 16, no. 2-3, pp. 191–221, 2005.

[14] 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.

[15] D. Grimes and R. Rao, "Bilinear sparse coding for invariant vision," *Neural Computation*, vol. 17, no. 1, pp. 47–73, 2005.

[16] R. Memisevic and G. Hinton, "Unsupervised learning of image transformations," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[17] J. Tenenbaum and W. Freeman, "Separating style and content with bilinear models," *Neural Computation*, vol. 12, no. 6, pp. 1247–1283, 2000.

[18] R. Memisevic, "Non-linear latent factor models for revealing structure in high-dimensional data," dissertation, University of Toronto, 2008.

[19] ——, "Gradient-based learning of higher-order image features." in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011.

[20] J. Susskind, R. Memisevic, G. Hinton, and M. Pollefeys, "Modeling the joint density of two images under a variety of transformations," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[21] M. Ranzato, A. Krizhevsky, and G. E. Hinton, "Factored 3-Way Restricted Boltzmann Machines For Modeling Natural Images," in *Proc. Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

[22] Y. Karklin and M. S. Lewicki, "Is early vision optimized for extracting higher-order dependencies?" in *Advances in Neural Information Processing Systems 18*. MIT Press, 2006.

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

[24] I. Ohzawa, G. C. Deangelis, and R. D. Freeman, "Stereoscopic Depth Discrimination in the Visual Cortex: Neurons Ideally Suited as Disparity Detectors," *Science (New York, N.Y.)*, vol. 249, no. 4972, pp. 1037–1041, Aug. 1990.

[25] T. Sanger, "Stereo disparity computation using Gabor filters," *Biological Cybernetics*, vol. 59, pp. 405–418, 1988, 10.1007/BF00336114.

[26] N. Qian, "Computing stereo disparity and motion with known binocular cell properties," *Neural Comput.*, vol. 6, pp. 390–404, May 1994.

[27] S. Becker and G. E. Hinton, "A self-organizing neural network that discovers surfaces in random-dot stereograms." *Nature*, vol. 355, pp. 161–163, 1992.