Journal of Machine Learning Research 12 (2011) 1025-1068 Submitted 7/10; Revised 1/11; Published 3/11

Two Distributed-State Models For Generating High-Dimensional Time Series[*]

Graham W. Taylor GWTAYLOR@CS.NYU.EDU
Courant Institute of Mathematical Sciences
New York University
New York, NY 10003, USA

Geoffrey E. Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
Toronto, ON M5S 3G1, Canada

Sam T. Roweis ROWEIS@CS.NYU.EDU
Courant Institute of Mathematical Sciences
New York University
New York, NY 10003, USA

Editor: Yoshua Bengio

Abstract

In this paper we develop a class of nonlinear generative models for high-dimensional time series. We first propose a model based on the restricted Boltzmann machine (RBM) that uses an undirected model with binary latent variables and real-valued "visible" variables. The latent and visible variables at each time step receive directed connections from the visible variables at the last few time-steps. This "conditional" RBM (CRBM) makes on-line inference efficient and allows us to use a simple approximate learning procedure. We demonstrate the power of our approach by synthesizing various sequences from a model trained on motion capture data and by performing on-line filling in of data lost during capture.

We extend the CRBM in a way that preserves its most important computational properties and introduces multiplicative three-way interactions that allow the effective interaction weight between two variables to be modulated by the dynamic state of a third variable. We introduce a factoring of the implied three-way weight tensor to permit a more compact parameterization. The resulting model can capture diverse styles of motion with a single set of parameters, and the three-way interactions greatly improve its ability to blend motion styles or to transition smoothly among them.

Videos and source code can be found at `http://www.cs.nyu.edu/~gwtaylor/publications/jmlr2011`.

Keywords: unsupervised learning, restricted Boltzmann machines, time series, generative models, motion capture

1. Introduction

The simplest time series models, and the earliest studied, contain no hidden variables. Two members of this class of "fully-observed" models are the vector autoregressive model and the N^{th} order

∗. This article is dedicated to the memory of the third author who unexpectedly passed away on January 12, 2010.

Markov model. Though elegant in their construction, these models are limited by their lack of memory. To capture long-range structure they must maintain explicit links to observations in the distant past, which results in a blow-up in the number of parameters. The strong regularities present in many time series suggest that a more efficient parameterization is possible.

More powerful models, such as the popular hidden Markov model (HMM), introduce a hidden (or latent) state variable that controls the dependence of the current observation on the history of observations. HMMs, however, cannot efficiently model data that is a result of multiple underlying influences since they rely on a single, discrete K-state multinomial to represent the entire history of observations. To model N bits of information about the past, they require 2^N hidden states.

In this paper, we propose an alternative class of time series models that have three key properties which distinguish them from the prior art. The first property is distributed (i.e., componential) hidden state. Mixture models such as HMMs generate each observation from a single category. Distributed state models (e.g., products) generate each object from a set of features that each contain some aspect of that object's description. linear dynamical systems (LDS) have a continuous, and therefore componential hidden state, but in order to make inference in these models tractable, the relationship between latent and visible variables is constrained to be linear. We show that by carefully choosing the right form of nonlinear observation model it is possible to attain tractable, exact inference, yet retain a rich representational capacity that is linear in the number of components.

Directed acyclic graphical models (or Bayes nets) are a dominant paradigm in models of static data. Their temporal counterparts, dynamic Bayes nets (Ghahramani, 1998), generalize many existing models such as the HMM and its various extensions. In all but the simplest directed models, inference is made difficult due to a phenomenon known as "explaining away" where observing a child node renders its parents dependent (Pearl, 1988). To perform inference in these networks, typically one resorts to approximate techniques such as variational inference (Neal and Hinton, 1998) or Monte Carlo methods which have a significant number of disadvantages (Ghahramani, 1998; Murphy, 2002).

An alternative to directed models is to abandon the causal relationship between variables, and instead focus on *undirected* models. One such model, the restricted Boltzmann machine (RBM) (Smolensky, 1986), has garnered recent interest due to its desirable property of permitting efficient exact inference. Unfortunately this comes at a cost: Exact maximum likelihood learning is no longer possible due to the existence of an intractable normalizing constant called the partition function. However, the RBM has an efficient, approximate learning algorithm, contrastive divergence (CD) (Hinton, 2002), that has been shown to scale well to large problems. RBMs have been used in a variety of applications (Welling et al., 2005; Gehler et al., 2006; Hinton and Salakhutdinov, 2006; Larochelle et al., 2007; Salakhutdinov et al., 2007) and over the last few years their properties have become better understood (Bengio and Delalleau, 2008; Salakhutdinov and Murray, 2008; Sutskever and Hinton, 2008). The CD learning procedure has also been improved (Carreira-Perpinan and Hinton, 2005; Tieleman, 2008; Tieleman and Hinton, 2009). With a few exceptions (Hinton and Brown, 2000; Sutskever and Hinton, 2007) the literature on RBMs is confined to modeling static data. In this paper, we leverage the desirable properties of an undirected architecture, the RBM, and extend it to model time series. This brings us to the second key property of the models we propose: their observation or emission distribution is an undirected, bipartite graph. This makes inference in our models simple and efficient.

The final key property of our proposed models is that they can form the building blocks of deep networks by incrementally learning one layer of feature extractors at a time. One motivation for promoting deep architectures is biological plausibility. Experimental evidence supports the belief that the brain uses multiple layers of feature-detecting neurons to process rich sensory input such as speech or visual signals (Hinton, 2007). There is also a practical argument for deep learning. In capturing more abstract, high-level structure from the data, the higher layers provide a more statistically salient representation for tasks such as discrimination. These ideas are not new, but until recently, the problem of how to efficiently train deep networks remained open. The backpropagation algorithm requires a large amount of labeled data and has difficulties with poor local minima and vanishing gradients. A resurgence in the study of deep architectures has been sparked by the discovery that deep networks can be initialized by greedy unsupervised learning of each layer (Hinton et al., 2006). RBMs were originally proposed for this task, but autoencoders (Bengio et al., 2007) and sparse encoder-decoder networks (Ranzato et al., 2006) have also been shown to work. After a pre-training stage, the entire network can be fine-tuned with either a generative or discriminative objective.

2. Modeling Human Motion

Motion capture (mocap) is the process of recording the movement of a subject as a time series of 3D cartesian coordinates corresponding to real or virtual points on the body. Most modern systems use a series of synchronized high-speed cameras to capture the location of strategically-placed physical markers attached to the subject (so called “marker-based” systems) or use image features to infer points of interest (so-called “markerless” systems). Marker-based systems are much more common but necessitate the use of a laboratory setting. Markerless systems permit motion capture in more natural environments (e.g., outdoors) but in general require more time to post-process the data. Recent advances in motion capture technology have fueled interest in the analysis and synthesis of motion data for computer animation and tracking.

Given its high-dimensional nature, nonlinearities, and long-range dependencies, mocap data is ideal for both studying the limitations of time series models and demonstrating their effectiveness. Several large motion capture data repositories are available, and people are very good at detecting anomalies in data that is generated from a model, so it is easy to judge the relative generative ability of two models. While focusing on a particular domain has greatly facilitated model development and comparison, there is nothing motion-specific to any of the models discussed herein. Therefore, there is no reason to believe that they cannot be applied to other high-dimensional, highly-structured time series data. In the following discussion, we briefly review related work in mocap-driven motion synthesis.

2.1 Motion Synthesis for Computer Animation

A dominant approach in computer animation is “keyframing” whereby an animator employs software to manually configure the “key” body poses over time, and these frames are interpolated to form smooth trajectories. This process, however, is time and labor intensive. It is therefore common to use mocap data to supplement or replace keyframing. A variety of methods have been developed to exploit the plethora of high-quality motion sequences available for animation. These approaches can be loosely divided into a handful of categories which we describe below.

2.1.1 CONCATENATION METHODS

Perhaps the simplest way to generate new motion sequences based on data is to sensibly concatenate short examples from a motion database to meet sparse user-specified constraints (Tanco and Hilton, 2000; Arikan and Forsyth, 2002; Kovar et al., 2002; Lee et al., 2002; Arikan et al., 2003). Pullen and Bregler (2002) propose a hybrid approach where low-frequency components are retained from user input and high-frequency components, called "texture", are added from the database. The obvious benefit of concatenation approaches is the high-quality motion that is produced. However, the "synthesized" motions are restricted to content already in the database and therefore many resources must be devoted to capture all desired content.

2.1.2 BLENDING AND INTERPOLATION METHODS

Many methods produce new motions by interpolating or blending existing content from a database (Rose et al., 1998; Park et al., 2002; Kovar and Gleicher, 2004; Mukai and Kuriyama, 2005). Unfortunately, these methods typically require extensive pre-processing which generally involves some type of time-warping to align the original sequences. Furthermore, the resulting motions often grossly violate dynamics, resulting in artifacts such as "footskate" and thereby requiring extensive clean-up using inverse kinematics.

2.1.3 TRANSFORMING EXISTING MOTION

Another method is to transform motion in the training data to new sequences by learning to adjust its style or other characteristics (Urtasun et al., 2004; Hsu et al., 2005; Torresani et al., 2007). Such approaches have produced impressive results given user-supplied motion content but we seek more powerful methods that can synthesize both style and content.

2.1.4 PHYSICS-BASED METHODS

Models based on the physics of masses and springs have produced some impressive results by using sophisticated "energy-based" learning methods (LeCun et al., 1998) to estimate physical parameters from motion capture data (Liu et al., 2005). However, if we want to generate realistic human motion, we need to model all the complexities of the real dynamics which is extremely difficult to do analytically. In this paper we focus on model driven analysis and synthesis but avoid the complexities involved in imposing physics-based constraints, relying instead on a "pure" learning approach in which all the knowledge in the model comes from the data.

2.1.5 GENERATIVE MODELS

Data from modern motion capture systems is high-dimensional and contains complex nonlinear relationships among the components of each observation, which is typically a series of joint angles with respect to some skeletal structure. This is a challenge for existing approaches to sequence modeling. However, there are examples of successes in the literature. Brand and Hertzmann (2000) model style and content of human motion with hidden Markov models (HMMs) whose emission distributions depend on stylistic parameters learned directly from the data. Their approach permits sampling of novel sequences from the model and applying new styles to existing content. HMMs, however, cannot efficiently model mocap data due to their simple, discrete state. Linear dynamical systems, on the other hand, have a more powerful hidden state but they cannot model the complex nonlinear dynamics created by the nonlinear properties of muscles, contact forces of the foot on the ground and myriad other factors. This problem has been addressed by applying piecewise-linear models to synthesize motion (Pavlovic et al., 2001; Li et al., 2002; Bissacco, 2005). In general, exact inference and learning is intractable in such models and approximations are costly and difficult to evaluate.

2.1.6 GAUSSIAN PROCESS MODELS

Models based on Gaussian processes (GPs) have received a great deal of recent attention, especially in the tracking literature. The Gaussian process dynamical model (Wang et al., 2008) extends the Gaussian process latent variable model (GP-LVM) (Lawrence, 2004) with a GP-based dynamical model over the latent representations. This model has been shown to discover interesting structure in motion data and permit synthesis of simple actions. However, the main concern with GP-based approaches is their computational expense (cubic in the number of training examples for learning, quadratic in the number of training examples for prediction or generation). This problem may be alleviated by sparse methods but this remains to be seen. Another downside of the GPDM is that a single model cannot synthesize multiple types of motion, a limitation of the simple manifold structure and unimodal dynamics learned by these models. Recently proposed models such as the multifactor GP (Wang et al., 2007) and hierarchical GP-LVMs (Lawrence and Moore, 2007) address this limitation.

3. Conditional Restricted Boltzmann Machines

We have emphasized that models with distributed hidden state are necessary for efficiently modeling complex time series. But using distributed representations for hidden state in directed models of time series (Bayes nets) makes inference difficult in all but the simplest models (HMMs and linear dynamical systems). If, however, we use a restricted Boltzmann machine (RBM) to model the probability distribution of the observation vector at each time frame, the posterior over latent variables factorizes completely, making inference easy. In this section, we first review the RBM and then propose a simple extension to capture temporal dependencies yet maintain its most important computational properties: simple, exact inference and efficient approximate learning using the contrastive divergence algorithm.

3.1 Restricted Boltzmann Machines

The restricted Boltzmann machine (Smolensky, 1986) is a Boltzmann machine with a special structure (Figure 1c). It has a layer of visible units fully connected to a layer of hidden units but no connections within a layer. This bi-partite structure ensures that the hidden units are conditionally independent given a setting of the visible units and vice-versa. Simplicity and exactness of inference are the main advantages to using an RBM compared to a fully connected Boltzmann machine.

To make the distinction between visible and hidden units clear, we use v_i to denote the state of visible unit i and h_j to denote the state of hidden unit j. We also distinguish biases on the visible units, a_i from biases on the hidden units, b_j. The RBM assigns a probability to any joint setting of the visible units, $\mathbf{v}$ and hidden units, $\mathbf{h}$:

$$p(\mathbf{v},\mathbf{h}) = \frac{\exp\left(-E(\mathbf{v},\mathbf{h})\right)}{Z} \tag{1}$$

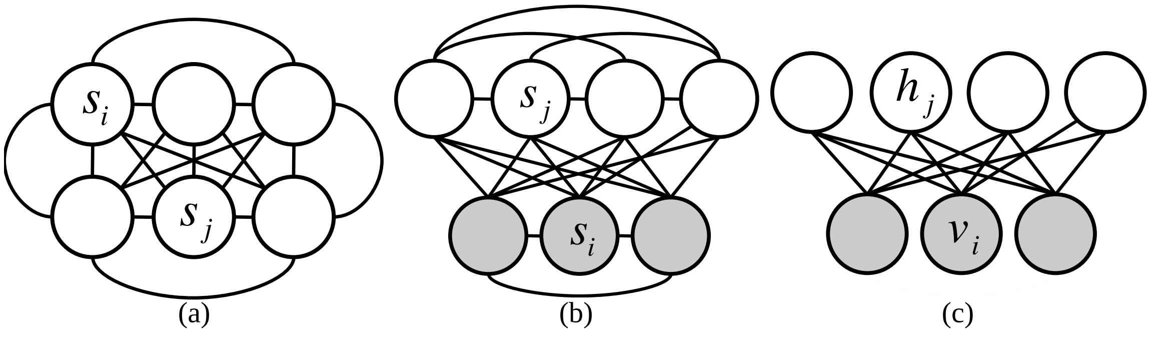

Figure 1: a) A Boltzmann machine. b) A Boltzmann machine partitioned into visible (shaded) and hidden units. c) A restricted Boltzmann machine.

where $E(\mathbf{v},\mathbf{h})$ is an energy function. When both the visible and the hidden units are binary with states 1 and 0, the energy function is

$$E(\mathbf{v},\mathbf{h}) = -\sum_{ij} W_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j$$

where Z is a normalization constant called the partition function, whose name comes from statistical physics. The partition function is intractable to compute exactly as it involves a sum over the (exponential) number of possible joint configurations:

$$Z = \sum_{v',h'} E(\mathbf{v}',\mathbf{h}').$$

Marginalizing over the hidden units in Equation 1 and maximizing the likelihood leads to a very simple maximum likelihood weight update rule:

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{model}}. \tag{2}$$

where $\langle \cdot \rangle_{\text{data}}$ is an expectation with respect to the data distribution and $\langle \cdot \rangle_{\text{model}}$ is an expectation with respect to the model's equilibrium distribution. Because of the conditional independence properties of the RBM, we can easily obtain an unbiased sample of $\langle v_i h_j \rangle_{\text{data}}$ by clamping the visible units to a vector in the training data set, and sampling the hidden units in parallel according to

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1+\exp(-b_j - \sum_i W_{ij} v_i)}. \tag{3}$$

This is repeated for each vector in a representative "mini-batch" from the training set to obtain an empirical estimate of $\langle v_i h_j \rangle_{\text{data}}$ To compute $\langle v_i h_j \rangle_{\text{model}}$ requires us to obtain unbiased samples from the joint distribution $p(\mathbf{v},\mathbf{h})$. However, there is no known algorithm to draw samples from this distribution in a practical amount of time. We can perform alternating Gibbs sampling by iterating between sampling from $p(\mathbf{h}|\mathbf{v})$ using Equation 3 and sampling from $p(\mathbf{v}|\mathbf{h})$ using

$$p(v_i = 1|\mathbf{h}) = \frac{1}{1+\exp(-a_i - \sum_j W_{ij} h_j)}. \tag{4}$$

However, Gibbs sampling in high-dimensional spaces typically takes too long to converge. Empirical evidence suggests that rather than running the Gibbs sampler to convergence, learning works well if we replace Equation 2 with

$$\Delta W_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}}, \tag{5}$$

where the second expectation is with respect to the distribution of "reconstructed" data. The reconstruction is obtained by starting with a data vector on the visible units and alternating between sampling all of the hidden units using Equation 3 and all of the visible units using Equation 4 K times. The learning rules for the biases are just simplified versions of Equation 5:

$$\begin{aligned} \Delta a_i &\propto \langle v_i \rangle_{\text{data}} - \langle v_i \rangle_{\text{recon}}, \\ \Delta b_j &\propto \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{recon}}. \end{aligned} \tag{6}$$

The above procedure is not maximum likelihood learning but it corresponds to approximately following the gradient of another function called the contrastive divergence (Hinton, 2002). We use the notation CD-K to denote contrastive divergence using K full steps of alternating Gibbs sampling after first inferring the states of the hidden units for a datavector from the training set. Typically K is set to 1, but recent results show that gradually increasing K with learning can significantly improve performance at an additional computational cost that is roughly linear in K (Carreira-Perpinan and Hinton, 2005).

3.2 RBMs with Real-Valued Observations

Typically, RBMs use stochastic binary units for both the visible data and hidden variables, but for many applications the observed data is non-binary. For some domains (e.g., modeling handwritten digits) we can normalize the data and use the real-valued probabilities of the binary visible units in place of their activations. When we use mean-field logistic units to model data that is very non-binary (e.g., modeling patches of natural images), it is difficult to obtain sharp predictions for intermediate values and so it is more desirable to use units that match the distribution of the data.

Fortunately, the stochastic binary units of RBMs can be generalized to any distribution that falls in the exponential family (Welling et al., 2005). This includes multinomial units, Poisson units and linear, real-valued units that have Gaussian noise (Freund and Haussler, 1992). To model real-valued data (e.g., mocap), we use a modified RBM with binary logistic hidden units and real-valued Gaussian visible units. The joint probability of $\mathbf{v}$ and $\mathbf{h}$ follows the form of Equation 1 where the energy function is now

$$E(\mathbf{v},\mathbf{h}) = \sum_i \frac{(v_i - a_i)^2}{2\sigma_i^2} - \sum_{ij} W_{ij} \frac{v_i}{\sigma_i} h_j - \sum_j b_j h_j.$$

where a_i is the bias of visible unit i, b_j is the bias of hidden unit j and σ_i is the standard deviation of the Gaussian noise for visible unit i. The symmetric weight, W_{ij}, connects visible unit i to hidden unit j.

Any setting of the hidden units makes a linear contribution to the mean of each visible unit:

$$p(v_i|\mathbf{h}) = \mathcal{N}\left(a_i + \sigma_i \sum_j W_{ij} h_j, \sigma_i^2\right). \tag{7}$$

Inference simply uses a scaled form of Equation 3:

$$p(h_j = 1|\mathbf{v}) = \frac{1}{1 + \exp(-b_j - \sum_i W_{ij} \frac{v_i}{\sigma_i})}.$$

Given the hidden units, the distribution of each visible unit is defined by a parabolic log likelihood function that makes extreme values very improbable. For any setting of the parameters, the gradient of the quadratic term with respect to a visible unit will always overwhelm the gradient due to the weighted input from the binary hidden units provided the value v_i of a visible unit is far enough from its bias, a_i. Conveniently, the contrastive divergence learning rules remain the same as in an RBM with binary visible units.

Finally, a brief note about σ_i: it is possible to learn, but this is difficult using CD-1 (Hinton, 2010). In practice, we simply rescale our data to have zero mean and unit variance and fix σ_i to be 1. Provided no noise is added to the mean reconstructions given by Equation 7, this makes the learning work well even though we would expect a good model to predict the data with much higher precision. For the remainder of the paper, we will assume $\sigma_i = 1$, but that no noise is added to the reconstructions used for learning.

3.3 The Conditional RBM

The RBM models static frames of data, but does not incorporate any temporal information. We can model temporal dependencies by treating the visible variables in the previous time slice(s) as additional fixed inputs. We add two types of directed connections: autoregressive connections from the past N configurations (time steps) of the visible units to the current visible configuration, and connections from the past M configurations of the visible units to the current hidden configuration. The addition of these directed connections turns the RBM into a conditional RBM (Figure 2). The autoregressive weights can model linear, temporally local structure very well, leaving the hidden units to model nonlinear, higher-level structure.

N and M are tunable parameters and need not be the same for both types of directed connections. To simplify discussion, we will assume $N = M$ and refer to N as the order of the model. Typically, in our experiments, we use a small number such as $N = 3$. In modeling motion capture with higher frame rates, we have found that a good rule of thumb is to set $N = F/10$ where F is the frame rate of the data (in frames per second).

To simplify the presentation, we will assume the data at $t-1,\ldots,t-N$ is concatenated into a “history” vector which we call $\mathbf{v}_{<t}$. So if $\mathbf{v}_t$ is of dimension D, then $\mathbf{v}_{<t}$ is of dimension $N \cdot D$. We will use k to index the individual, scalar components of $\mathbf{v}_{<t}$. The autoregressive parameters are summarized by an $N \cdot D \times D$ weight matrix called A and the directed “past to hidden” parameters are summarized by an $N \cdot D \times H$ matrix B where H is the number of binary hidden units. This does not change the computation, but allows us to simplify the presentation of the following equations as we can avoid explicitly summing over past frames.

3.3.1 INFERENCE AND LEARNING

Fortunately, inference in the CRBM is no more difficult than in the standard RBM. The states of the hidden units are determined by both the input they receive from the current observation and the input they receive from the recent past. Given $\mathbf{v}_t$ and $\mathbf{v}_{<t}$, the hidden units at time t are conditionally

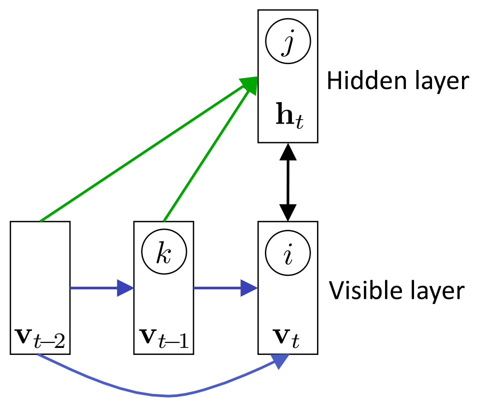

Figure 2: Architecture of the CRBM. In this figure we show $N = 2$ but in our experiments, we typically use a slightly higher order. There are no connections between the hidden units at different time steps (see Section 3.4.3).

independent. The effect of the past on each hidden unit can be viewed as a dynamic bias:

$$\hat{b}_{j,t} = b_j + \sum_k B_{kj} v_{k,<t}$$

which includes the static bias, b_j, and the contribution from the past. This slightly modifies the factorial distribution over hidden units: b_j in Equation 3 is replaced with $\hat{b}_{j,t}$ to obtain

$$p(h_{j,t} = 1 | \mathbf{v}_t, \mathbf{v}_{<t}) = \frac{1}{1 + \exp(-\hat{b}_{j,t} - \sum_i W_{ij} v_{i,t})}. \tag{8}$$

Note that we are now conditioning on $\mathbf{v}_{<t}$. Figure 3 shows an example of frame-by-frame inference in a trained CRBM.

The past has a similar effect on the visible units. The reconstruction distribution becomes

$$p(v_{i,t} | \mathbf{h_t}, \mathbf{v}_{<t}) = \mathcal{N}\left(\hat{a}_{i,t} + \sum_j W_{ij} h_{j,t}, 1\right) \tag{9}$$

where $\hat{a}_{i,t}$ is also a dynamically changing bias that is an affine function of the past:

$$\hat{a}_{i,t} = a_i + \sum_k A_{ki} v_{k,<t}.$$

We can still use contrastive divergence for training the CRBM. The updates for the symmetric weights, W, as well as the static biases, $\mathbf{a}$ and $\mathbf{b}$, have the same form as Equation 5 and Equation 6 but have a different effect because the states of the hidden units are now influenced by the previous visible units. The updates for the directed weights are also based on simple pairwise products. The

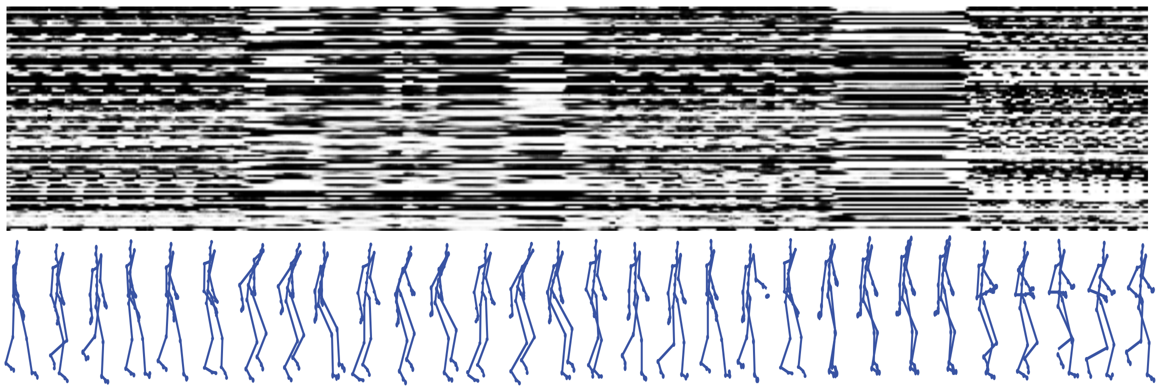

Figure 3: In a trained model, probabilities of each feature being "on", conditional on the data at the visible units. Shown is a 100-hidden unit model and a sequence which contains (in order) walking, sitting/standing (three times), walking, crouching, and running. Rows represent features, columns represent sequential frames.

gradients are now summed over all time steps:

$$\Delta W_{ij} \propto \sum_t \left(\langle v_{i,t} h_{j,t} \rangle_{\text{data}} - \langle v_{i,t} h_{j,t} \rangle_{\text{recon}} \right), \tag{10}$$

$$\Delta A_{ki} \propto \sum_t \left(\langle v_{i,t} v_{k,<t} \rangle_{\text{data}} - \langle v_{i,t} v_{k,<t} \rangle_{\text{recon}} \right), \tag{11}$$

$$\Delta B_{kj} \propto \sum_t \left(\langle h_{j,t} v_{k,<t} \rangle_{\text{data}} - \langle h_{j,t} v_{k,<t} \rangle_{\text{recon}} \right), \tag{12}$$

$$\Delta a_i \propto \sum_t \left(\langle v_{i,t} \rangle_{\text{data}} - \langle v_{i,t} \rangle_{\text{recon}} \right), \tag{13}$$

$$\Delta b_j \propto \sum_t \left(\langle h_{j,t} \rangle_{\text{data}} - \langle h_{j,t} \rangle_{\text{recon}} \right) \tag{14}$$

where $\langle \cdot \rangle_{\text{data}}$ is an expectation with respect to the data distribution, and $\langle \cdot \rangle_{\text{recon}}$ is the K-step reconstruction distribution as obtained by alternating Gibbs sampling, starting with the visible units clamped to the training data.

While learning a CRBM, we do not need to proceed sequentially through the training data sequences. The updates are only conditional on the past N time steps, not the entire sequence. As long as we isolate "chunks" of $N+1$ frames (the size depending on the order of the directed connections), these small windows can be mixed and formed into mini-batches. To speed up the learning, we assemble these chunks of frames into "balanced" mini-batches of size 100.

We randomly assign chunks to different mini-batches so that the chunks in each mini-batch are as uncorrelated as possible. To save computer memory, time frames are not actually replicated in mini-batches; we simply use indexing to simulate the "chunking" of frames.

3.3.2 SCORING OBSERVATIONS

The CRBM defines a joint probability distribution over a data vector, $\mathbf{v}_t$, and a vector of hidden states, $\mathbf{h}_t$, conditional on the recent past, $\mathbf{v}_{<t}$:

$$p(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = \frac{\exp\left(-E\left(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}\right)\right)}{Z(\mathbf{v}_{<t})} \tag{15}$$

where the partition function, Z, is constant with respect to $\mathbf{v}_t$ and $\mathbf{h}_t$ but depends on $\mathbf{v}_{<t}$. As in the RBM, it is intractable to compute exactly because it involves an integration over all possible settings of the visible and hidden units:

$$Z(\mathbf{v}_{<t}) = \sum_{\mathbf{h}'_t}\int_{\mathbf{v}'_t} \exp\left(\left(-E(\mathbf{v}'_t,\mathbf{h}'_t|\mathbf{v}_{<t}\right)\right) d\mathbf{v}'_t.$$

The energy function is given by

$$E(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = \frac{1}{2}\sum_i (v_{i,t}-\hat{a}_{i,t})^2 - \sum_{ij} W_{ij} v_{i,t} h_{j,t} - \sum_j \hat{b}_{j,t} h_{j,t} \tag{16}$$

where we have assumed $\sigma_i = 1$. The probability of observing $\mathbf{v}_t$ can be expressed by marginalizing out the binary hidden units:

$$p(\mathbf{v}_t|\mathbf{v}_{<t}) = \sum_{\mathbf{h}_t} p(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}) = \frac{\sum_{\mathbf{h}_t}\exp\left(-E\left(\mathbf{v}_t,\mathbf{h}_t|\mathbf{v}_{<t}\right)\right)}{Z(\mathbf{v}_{<t})}. \tag{17}$$

Under the CRBM, the probability of observing a *sequence*, $\mathbf{v}_{(N+1):T}$, given $\mathbf{v}_{1:N}$, is just the product of all the local conditional probabilities:

$$p(\mathbf{v}_{(N+1):T}|\mathbf{v}_{1:N}) = \prod_{t=N+1}^{T} p(\mathbf{v}_t|\mathbf{v}_{<t}). \tag{18}$$

We do not attempt to model the first N frames of each sequence, though a separate set of biases could be learned for this purpose.

Although the partition function makes Equation 17 and Equation 18 intractable to compute exactly, we can exploit the fact that the hidden units are binary and integrate them out to arrive at the "free energy":

$$F(\mathbf{v}_t|\mathbf{v}_{<t}) = \frac{1}{2}\sum_i (v_{i,t}-\hat{a}_{i,t})^2 - \sum_j \log\left(1+\exp(\sum_i W_{ij} v_{i,t} + \hat{b}_{j,t})\right), \tag{19}$$

which is a function of the model parameters and recent past. It is the negative log probability of an observation plus $\log Z$ (see Equation 15). Given a history, the free energy allows us to score a single temporal frame of observations under a fixed setting of the parameters, but unlike a probability it does not let us compare between models.[1] It can still be useful, however, in making deterministic forward predictions (as described in the following section). Freund and Haussler (1992) give details on deriving the free energy for an RBM.

1. Different models will have different partition functions.

3.3.3 GENERATION

Generation from a learned CRBM can be done on-line. The visible states at the last few time steps determine the effective biases of the visible and hidden units at the current time step. We always keep the previous visible states fixed and perform alternating Gibbs sampling to obtain a joint sample from the CRBM. This picks new hidden and visible states that are compatible with each other and with the recent (visible) history. To start alternating Gibbs sampling, we need to initialize with either $\mathbf{v}_t$ or $\mathbf{h}_t$. For time-series data that is smooth (e.g., mocap), a good choice is to initially set $\mathbf{v}_t = \mathbf{v}_{t-1}$. In practice, we alternate 30 to 100 times, though the quality of generated data does not seem to be sensitive to this parameter.

Generation does not require us to retain the training data set, but it does require initialization with N observations. Typically we use randomly drawn consecutive frames from the training data as an initial configuration.

A trained CRBM has the ability to fill in missing data (complete or partial observations), regardless of where the dropouts occur in a sequence. To be strictly correct, we would need to use smoothing (i.e., conditioning on future as well as past observations) in order to take into account the effect of a filled-in value on the probability of future observed values. As in the learning procedure, we ignore smoothing and this approximation allows us to fill in missing data on-line. Filling in missing data with the CRBM is very similar to generation. We simply clamp the known data to the visible units, initialize the missing data to something reasonable (for example, the value at the previous frame), and alternate between stochastically updating the hidden and visible units, *with the known visible states held fixed*.

The noise in sampling may be an asset when using the CRBM to generate sequences, but when using the CRBM to fill in missing data, or in a predictive setting it may be undesirable. Rather than obtaining a sample, we may want the model's "best guess". Given the model parameters, and past history, we can follow the negative gradient of the free energy (Equation 19) with respect to either a complete or partial setting of the visible variables, $\mathbf{v}_t$:

$$\frac{\partial F(\mathbf{v}_t|\mathbf{v}_{<t})}{\partial v_{k,t}} = v_{k,t} - \left(\hat{a}_{i,t} + \sum_j W_{ij} f\left(-\sum_i W_{ij} v_{i,t} - \hat{b}_{j,t}\right)\right)$$

where $f(\cdot)$ is the logistic function. The gradient at a unit has an intuitive form: it is the difference between its current value and the value that would be obtained by mean-field reconstruction. We use conjugate-gradient optimization, but any general purpose gradient-based optimizer is suitable.

3.4 Higher Level Models: The Conditional Deep Belief Network

Once we have trained the model, we can add layers in the same way as a deep belief network (DBN) (Hinton et al., 2006). The previous layer CRBM is kept, and the sequence of hidden state vectors, while driven by the data, is treated as a new kind of "fully observed" data. The next level CRBM has the same architecture as the first (though we can alter the number of its units) and is trained in the exact same way. Upper levels of the network can then model higher-order structure.

Figure 4a shows a CRBM whose symmetric, undirected weights have been represented explicitly by two sets of directed weights: top-down "generative" weights W_0, and bottom-up "recognition" weights, W_0^T. This representation is purely illustrative: it does not at all change the model. The use of the zero subscripts and superscripts simply indicates that the CRBM is first in a series of layers which we will introduce shortly.

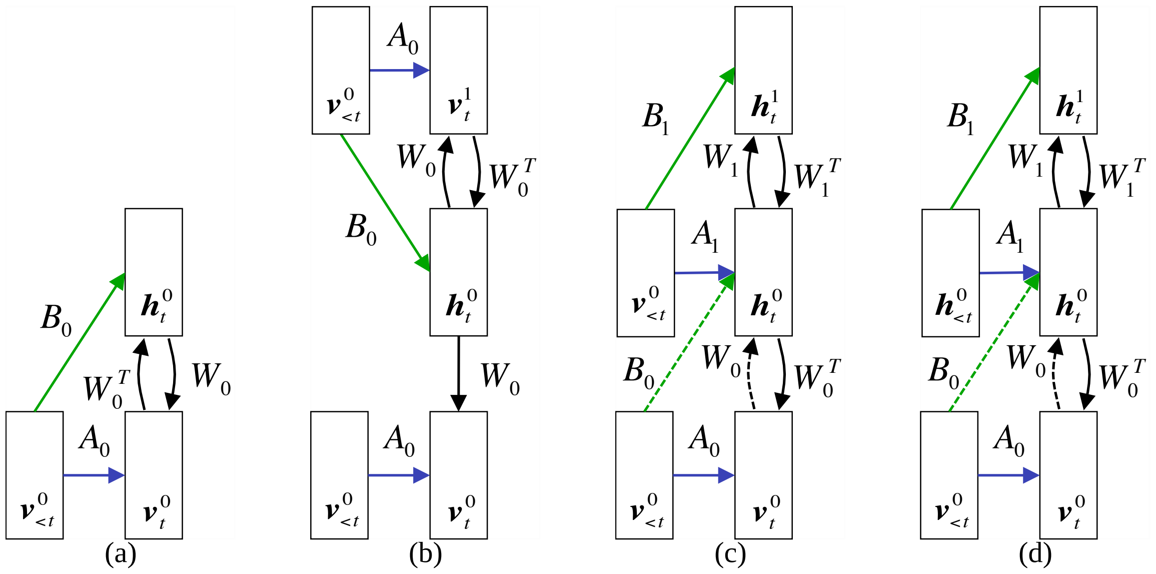

Figure 4: Building a conditional deep belief network. (a) The CRBM. (b) A generative model whose weights between layers are tied. It defines the same joint distribution over $\mathbf{v}_t^0$ and $\mathbf{h}_t^0$. The top two layers interact using symmetric connections while all other connections are directed. (c) Improving the model by untying the weights; holding W_0, A_0 and B_0 fixed and greedily training W_1, A_1 and B_1. Note that the "dashed" directed, bottom-up weights are not part of the generative model. They are used to infer factorial, approximate posterior distributions over $\mathbf{h}_t^0$ when $\mathbf{v}_t^0$ is clamped to the data. (d) The model we use in practice. Note the change from $\mathbf{v}_{<t}^0$ to $\mathbf{h}_{<t}^0$. We ignore uncertainty in the past hidden states.

Figure 4b shows a generative model that is equivalent to the original CRBM in the sense that their joint distributions over $\mathbf{v}_t^0$ and $\mathbf{h}_t^0$, conditional on $\mathbf{v}_{<t}^0$ are the same. We have added a second set of visible units, $\mathbf{v}_t^1$, identical to the first, and ensured that the undirected, symmetric weights between $\mathbf{v}_t^1$ and $\mathbf{h}_t^0$ are equal to the weights used in the original CRBM. Furthermore, we introduce a copy of $\mathbf{v}_{<t}^0$ and the autoregressive connections, A_0. The weights are therefore "tied" between the two layers. Additionally, the bottom-up weights between $\mathbf{v}_t^0$ and $\mathbf{h}_t^0$, W_0^T, are no longer part of the generative model in Figure 4b. Although the model defines the same joint distribution, its semantics are very different than the CRBM. To generate an observation, $\mathbf{v}_t^0$, conditional on $\mathbf{v}_{<t}^0$, we must reach equilibrium in the conditional associative memory formed by the top two layers and then perform a single down-pass using directed weights W_0 and A_0. The CRBM generates observations as explained in Section 3.3.3.

Note that if we observe $\mathbf{v}_t^0$, the units $\mathbf{h}_t^0$ are no longer conditionally independent because the undirected connections between $\mathbf{v}_t^0$ and $\mathbf{h}_t^0$ have been replaced by directed connections. The new model is therefore subject to the effects of "explaining away". However, because of the tied weights, the CRBM at the top two layers becomes a "complementary prior" (Hinton et al., 2006): meaning that when we multiply the likelihood term by the prior, the posterior is factorial. Researchers who are used to using directed models often assume that $W_0\mathbf{v}_t^0 + B_0\mathbf{v}_{<t}^0$ is computing a likelihood term.

This is incorrect. It is computing the product of the likelihood term and the prior term (i.e., the posterior). Both the likelihood term and the prior term are much more complicated since they are each far from being factorial.

If we hold W_0, A_0 and B_0 fixed, but "untie"[2] the weights between the top two layers (Figure 4c) we can improve the generative model by greedily learning W_1, A_1 and B_1, treating the activations of $\mathbf{h}_t^0$ while driven by the training data as a kind of "fully-observed" data. When the weights are untied, units in the topmost layer no longer represent the visible units, but another layer of latent features, $\mathbf{h}_t^1$. We can still use W_0^T and B_0 (which are not part of the generative model) to infer factorial *approximate* posterior distributions over the states of $\mathbf{h}_t^0$.

The joint distribution defined by the original CRBM, $p(\mathbf{v}_t^0, \mathbf{h}_t^0 | \mathbf{v}_{<t}^0)$, decomposes into a mapping from features to data, $p(\mathbf{v}_t^0 | \mathbf{h}_t^0, \mathbf{v}_{<t}^0)$, and an implicit prior over the features, $p(\mathbf{h}_t^0 | \mathbf{v}_{<t}^0)$ which is also determined by W_0. We can think of training the next layer as a means of improving the prior model. By fixing W_0, A_0, the distribution $p(\mathbf{v}_t^0 | \mathbf{h}_t^0, \mathbf{v}_{<t}^0)$ is unchanged. The gain from building a better model of $p(\mathbf{h}_t^0 | \mathbf{v}_{<t}^0)$ more than offsets the loss from having to perform approximate inference. This greedy learning algorithm can be applied recursively to any number of higher layers and is guaranteed to never decrease a variational lower bound on the log probability of the data under the full generative model (Hinton et al., 2006).

In practice, greedily training multiple layers of representation works well. However, there are a number of small changes we make to gain flexibility and improve the computational cost of performing inference and learning. Bending the rules as follows breaks the above guarantee:

1. We replace maximum likelihood learning with contrastive divergence (for obvious computational reasons).

2. The guarantee relies on initializing the weights of each successive layer with the weights in the layer below. This assumes that all odd layers are of equal size and all even layers of equal size. In practice, however, we typically violate this constraint and initialize the weights to small random values.

3. Rather than train each layer conditional on $\mathbf{v}_{<t}^0$ (which we assume to be the fully-observed recent past of the visible units), we train each layer using its own recent past as the conditioning input. $\mathbf{v}_{<t}^0, \mathbf{h}_{<t}^0, \ldots, \mathbf{h}_{<t}^{H-1}$ (where H is the number of hidden layers), always treating the past as fully-observed.

The model that we use in practice is shown in Figure 4d. It is a conditional deep belief network (CDBN). The inference we perform in this model, conditional on past visible states, is approximate because it ignores the future (it does not do smoothing). Because of the directed connections, exact inference within the model should include both a forward and backward pass through each sequence. We perform only a forward pass because smoothing is intractable in the multi-layer model. Effectively, at each layer we replace the full posterior by an approximate filtering distribution. However, there is no guarantee that this is a good approximation. Compared with an HMM, the lack of smoothing is a loss. But the deep model is still exponentially more powerful at using its hidden state to represent data.

2. A note on our naming convention: the "untied" A_0 becomes B_1 since it now represents a visible-to-hidden connection. The "untied" B_0 becomes A_1 since it will ultimately be a "visible-visible" connection when the hidden units are treated as observed during greedy learning.

3.4.1 ON-LINE GENERATION WITH HIGHER-LEVEL MODELS

The generative model for a conditional DBN consists of a top-level conditional associative memory (with symmetric weights and dynamic biases) and any number of directed lower layers (with top-down generative weights and dynamic generative biases). We also maintain bottom-up connections that are used in approximate inference. Like in a DBN, to generate a sample, $\mathbf{v}_t$, the associative memory must settle on a joint setting of the units in the top two hidden layers and then the top-down weights are used to generate the lower layers. Since the model is conditional, each layer must also consider the effect of the past via the dynamic biases. Note that in a deep network, all but the topmost hidden layer will have two sets of dynamic biases: recognition biases from when it was greedily trained as a hidden layer, and generative biases from when it was subsequently trained as a "visible" layer. During generation, we must be careful not to double-count the input to each layer (i.e., by including both types of biases when computing the total input to each unit); we use the recognition biases during inference and generative biases during generation.

As a concrete example, let us consider generating an observation from a conditional DBN built by greedily training two CRBMs (the same network shown in Figure 4d).

1. If the first CRBM is order N and the second CRBM is order M then we must initialize with $N+M$ frames, $\mathbf{v}_{1:(N+M)}$ (Figure 5a).

2. Next, we initialize M frames of the first hidden layer using a mean-field up-pass through the first CRBM (Figure 5b).

3. Then we initialize the first layer hidden units at $t = N+M+1$ to be a copy of the real-valued probabilities we have just inferred at $t = N+M$. We perform alternating Gibbs sampling in the 2nd layer CRBM. At each step, we stochastically activate the top-level hidden units, but on the final step, we suppress noise by using the real-valued probabilities of the top layer to obtain the real-valued probabilities of the first layer hidden units (Figure 5c).

4. We do a mean-field down-pass in the first layer CRBM to obtain the visible states at time $t = N+M+1$ (Figure 5d).

Again, we copy the real-valued probabilities of the first layer hidden units to initialize Gibbs sampling for the next frame, and repeat steps 3 and 4 above for as many frames as desired.

3.4.2 FINE-TUNING

Following greedy learning, both the weights and the simple inference procedure are suboptimal in all but the top layer of the network, as the weights have not changed in the lower layers since their respective stage of greedy training. We can, however, use a contrastive form of the "wake-sleep" algorithm (Hinton et al., 1995) called the "up-down" algorithm (Hinton et al., 2006) to fine-tune the generative model. In our experiments, we have observed that fine-tuning improves the visual quality of generated sequences at a modest additional computational cost.

3.4.3 TEMPORAL LINKS BETWEEN HIDDEN UNITS

In a conditional restricted Boltzmann machine the hidden state and visible state depend only on past instances of the visible variables. The CRBM is a special case of the temporal restricted Boltzmann machine (TRBM) (Sutskever and Hinton, 2007) in which there are no temporal connections

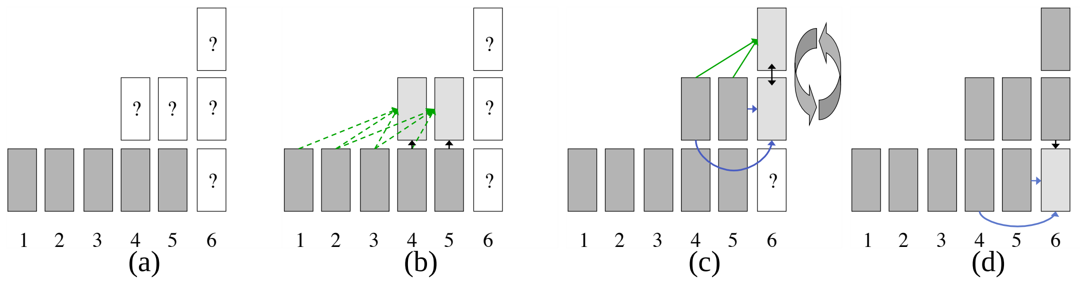

Figure 5: Generating from a conditional deep belief network with two hidden layers. For this example, we assume the first layer CRBM is third order and the second layer CRBM is second order. We provide five frames to initialize the model.

between hidden units. This makes filtering in the CRBM exact, and "mini-batch" learning possible, as training does not have to be done sequentially. This latter property can greatly speed up learning as well as smooth the learning signal, as the order of data vectors presented to the network can be randomized. This ensures that the training cases in each mini-batch are as uncorrelated as possible.

As soon as we introduce connections between hidden units, we must resort to approximate filtering or deterministic methods (Sutskever et al., 2009) even in a single layer model. In training higher-level models using CRBMs, we gain hidden-to-hidden links via the autoregressive connections of the higher layers. At each stage of greedy learning, filtering is exact within each CRBM. However, filtering in the overall multi-layer model is approximate.

3.5 Experiments

We have carried out a series of experiments training CRBM models on motion capture data from publicly available repositories. After learning a model using the updates described in Section 3.3, we can demonstrate in several ways what it has learned about the structure of human motion. Perhaps the most direct demonstration, which exploits the fact that it is a probability density model of sequences, is to use the model to generate *de-novo* a number of synthetic motion sequences. Supplemental video files of these sequences are available on the website mentioned in the abstract; these motions have not been retouched by hand in any motion editing software. Note that we also do not have to keep a reservoir of training data sequences for generation - we only need the weights of the trained model and N valid frames for initialization. Our model is, therefore, suitable for low-memory devices.[3] More importantly, we believe that compact models are likely to be better at generalization.

3.5.1 DATA SOURCE AND REPRESENTATION

The first data set used in these experiments was obtained from `http://mocap.cs.cmu.edu`. It will be hereafter referred to as the CMU data set. The second data set used in these experiments was released by Hsu et al. (2005). We obtained it from from `http://people.csail.mit.edu/ehsu/work/sig05stf/`. It will be hereafter referred to as the MIT data set. The data consisted of 3D joint angles derived from 30 (CMU) or 17 (MIT) markers plus a root orientation and displacement.

3. The level of compression obtained will of course vary with the number of free parameters and size of the data set.

Data was represented with the encoding described in Appendix A. The final dimensionality of our data vectors was 62 (CMU) and 49 (MIT).

One advantage of the CRBM is the fact that the data does not need to be heavily preprocessed or dimensionality reduced before learning. Other generative approaches (Brand and Hertzmann, 2000; Li et al., 2002) apply PCA to reduce noise and dimensionality. However, dimensionality reduction becomes problematic when a wider range of motions is to be modeled. The autoregressive connections can be thought of as doing a kind of "whitening" of the data.

3.5.2 DETAILS OF LEARNING

Except where noted, all CRBM models were trained as follows: Each training case was a window of $N+1$ consecutive frames and the order of the training cases was randomly permuted. The training cases were presented to the model as "mini-batches" of size 100 and the weights were updated after each mini-batch. Models were trained using CD-1 (see Section 3.1) for a fixed number of epochs (complete passes through the data). All parameters used a learning rate of 10^{-3}, except for the autoregressive weights which used a learning rate of 10^{-5}. A momentum term was also used: 0.9 of the previous accumulated gradient was added to the current gradient. All parameters (excluding biases) used L2 weight decay of 0.0002.

3.5.3 GENERATION OF WALKING AND RUNNING SEQUENCES FROM A SINGLE MODEL

In our first demonstration, we train a single CRBM on data containing both walking and running motions; we then use the learned model to generate both types of motion, depending on how it is initialized. We extracted 23 sequences of walking and 10 sequences of running from subject 35 in the CMU data set. After downsampling to 30Hz, the training data consisted of 2813 frames. We trained a 200 hidden-unit CRBM for 4000 passes through the training data, using a third-order model (for directed connections). The order of the sequences was randomly permuted such that walking and running sequences were distributed throughout the training data.

Figure 6 shows a walking sequence and a running sequence generated by the same model, using alternating Gibbs sampling (with the probability of hidden units being "on" conditional on the current and previous three visible vectors). Since the training data does not contain any transitions between walking and running (and *vice-versa*), the model will continue to generate walking or running motions depending on where it is initialized.

3.5.4 LEARNING TRANSITIONS BETWEEN WALKING AND RUNNING

In our second demonstration, we show that our model is capable of learning not only several types of homogeneous motion content but also the transitions between them when the training data itself contains examples of such transitions. We trained on 9 sequences (from the MIT database, file `Jog1_M`) containing long examples of walking and running, as well as a few transitions between the two gaits. After downsampling to 30Hz, this provided us with 2515 frames. Training was done as before, but after the model was trained, an identical 200 hidden-unit model was trained on top of the first model (see Section 3.4). The resulting two-level model was used to generate data. A video available on the website demonstrates our model's ability to stochastically transition between various types of motion during a single generated sequence.

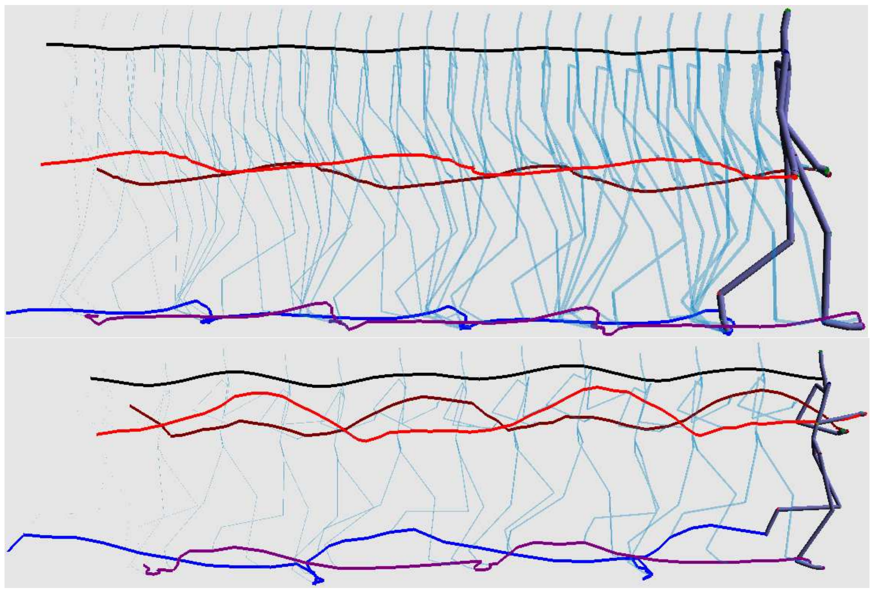
Figure 6: After training, the same model can generate walking (top) and running (bottom) motion (see supplemental videos). Each skeleton is 4 frames apart.

3.5.5 INTRODUCING TRANSITIONS USING NOISE

In our third demonstration, we show how transitions between different types of motion content can be generated even when such transitions are absent in the data. We use the same model and data as described in Section 3.5.3, where we have learned on separate sequences of walking and running. To generate, we use the same sampling procedure as before, except that at each time we stochastically choose the hidden states (given the current and previous three visible vectors) we add a small amount of Gaussian noise to the hidden state biases. This encourages the model to explore more of the hidden state space without deviating too far from the current motion. Applying this "noisy" sampling approach, we see that the generated motion occasionally transitions between learned gaits. These transitions appear natural (see the supplemental video).

3.5.6 LEARNING MOTION STYLE

We have demonstrated that the CRBM can generate and transition between different gaits, but what about its ability to capture more subtle stylistic variation within a particular gait? We also seek to show the CRBM's ability to learn on data at a higher frame-rate (60Hz), and from a much larger training corpus. Finally, we incorporate label information into our training procedure.

From the CMU data set, we extracted a series of 10 stylized walk sequences performed by subject 137. The walks were labeled as *cat, chicken, dinosaur, drunk, gangly, graceful, normal, old-man, sexy* and *strong*. We balanced the data set by repeating the sequences three to six times (depending on the original length) so that our final data set contained approximately 3000 frames of each style at 60fps.

In general, we used the same training procedure as above, but made a few important changes:

- At each iteration of CD learning, we performed 10 steps of alternating Gibbs sampling (CD-10)
- We added a sparsity term to the energy function to gently encourage the hidden units, while driven by the data, to have an average activation of 0.2 (details below)
- At each iteration of CD learning, we added Gaussian noise with $\sigma = 1$ to each dimension of the past history, $\mathbf{v}_{<t}$. This ensures that the generative model can cope with the type of noisy history that is produced when generating from the model. For the linear autoregressive parameters, A, this is equivalent to L2 regularization (Matsuoka, 1992). For the parameters which involve the binary hidden units it is not quite equivalent but has a very similar effect.

These experiments were carried out considerably later than the experiments described in Section 3.5.3 to 3.5.5 and so represent a refinement to our learning method. This allows us to cope with the higher frame rate and larger degree of variability in the training set. Recent work on estimating the partition functions of RBMs and evaluating the log probability of held-out sets has shown that models trained with CD$>$1 , although more computationally demanding to train, are significantly better generative models (Salakhutdinov and Murray, 2008). We have chosen CD-10 as a compromise between closely approximating maximum likelihood learning and minimizing computational cost.

The recent popularity of sparse, overcomplete latent representations has highlighted both the theoretical and practical motivations for their use in unsupervised learning (Olshausen and Field, 1997; Lee and Seung, 1999; Ranzato et al., 2006; Lee et al., 2008). Sparse representations are often more easy to interpret, and also more robust to noise. Furthermore, evidence suggests that they may be used in biological systems. Recent sparse “energy-based methods” (Ranzato et al., 2006, 2007, 2008) have proposed the use of sparsity as an alternative to contrastive divergence learning. The “contrastive term” in CD (which represents the derivative of the log partition function) corresponds to pulling up on the energy (or pushing down on the probability) of points outside the training set. Another way to ensure that the energy surface is low only around the training set is to eliminate the partition function and replace it with a term that limits the volume of the input space over which the energy surface can take low value (Ranzato et al., 2008). Using sparse overcomplete latent representations is a means of limiting this volume by minimizing the information content of the latent representation. Using both a contrastive term and sparsity, as we have done here, is a two-fold approach to sculpting energy surfaces.

To implement sparsity, we maintained a damped “average activation” estimate for each hidden unit. Each element of this vector was initialized to the target activation, 0.2. Every time we presented a mini-batch, we updated the estimate to be 0.9 times its current value plus 0.1 times the average activation of the hidden units while the visible units were clamped to the data. The average was taken over the mini-batch. After we calculated the positive-phase (data) and negative-phase (after K steps of Gibbs sampling) statistics for each parameter, we added, to the original gradient, the gradient of the cross-entropy error between the updated activity estimate and the target, 0.2, with respect to that parameter. Note that updates for visible-only parameters (e.g., autoregressive weights and visible biases) were unaffected by the sparsity term. Our sparsity term is similar to the one used by Lee et al. (2008). However, they used a squared-error penalty between average activation and target while we used cross-entropy error (Nair and Hinton, 2009) which is more appropriate for logistic units.

1-layer Model. A single-layer CRBM with 1200 hidden units and $N = 12$ was trained for 200 epochs on data for 10 different walking styles, with the parameters being updated after every 100 training cases. Each training case was a window of 13 consecutive frames and the order of the training cases was randomly permuted. In addition to the real-valued mocap data, the hidden units received additive input from a "one-hot" encoding of the matching style label through another matrix of weights. Respecting the conditional nature of our application (generation of stylized motion, as opposed to, say classification) this label was not reconstructed during learning. After training the model, we generated motion by initializing with 12 frames of training data and holding the label units clamped to the style matching the initialization.

With a single layer we could generate high-quality motion of 9/10 styles (see the supplemental videos), however, the model failed to produce good generation of the *old-man* style. We believe that this relates to the subtle nature of this particular motion. In examining the activity of the hidden units over time while clamped to training data, we observed that the model devotes most of its hidden capacity to capturing the more "active" styles as it pays a higher cost for failing to model more pronounced frame-to-frame changes.

2-layer Model. We also learned a deeper network by first training a CRBM with 600 binary hidden units and real-valued visible units and then training a higher-level CRBM with 600 binary hidden and 600 binary visible units. Both models used $N = 12$. The data for training the higher-level CRBM consisted of the activation probabilities of the hidden units of the first CRBM while driven by the training data. Style labels were only connected to the top-layer of this network, while training the second level CRBM. The first-level model was trained, without style labels, for 300 epochs and the second-level model was trained for 120 epochs.

After training, the 2-hidden-layer network was able to generate high-quality walks of all styles, including *old-man* (see Figure 7 and the supplemental videos). The second level CRBM layer effectively replaces the prior over the first layer of hidden units, $p(\mathbf{h}_t|\mathbf{v}_{<t})$, that is implicitly defined by the parameters of the first CRBM. This provides a better model of the subtle correlations between the features that the first-level CRBM extracts from the motion. The superiority of the second layer may indeed be a result of its ability to capture longer-term dependencies in the data. Learning the *old-man* style is conditional on capturing longer-term dependencies since the signal (representing joint angles) changes more slowly. The 2-layer network has access to a wider temporal context and therefore is better able to model this particular style. We thank one of the anonymous reviewers for suggesting this explanation.

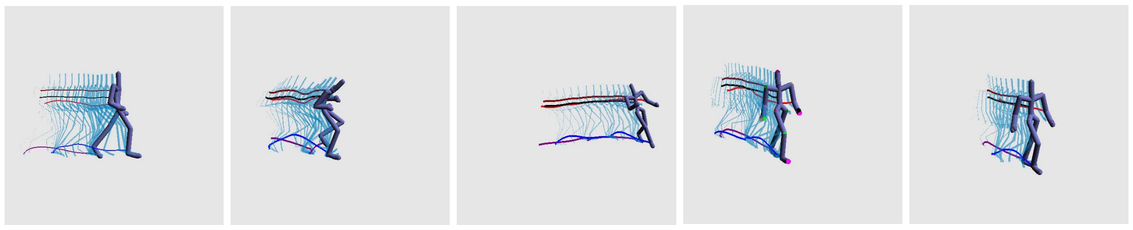
(a) Cat (b) Dinosaur (c) Graceful (d) Sexy (e) Strong

Figure 7: Generating different walking styles from the same conditional deep belief network with two hidden layers.

Qualitative Comparison to the GPDM. The Gaussian process dynamical model (GPDM) (Wang et al., 2008) was proposed for human motion synthesis and for use as a prior in tracking (Urtasun et al., 2006). It extends the Gaussian process latent variable model (GP-LVM) (Lawrence, 2004) with a GP-based dynamical model over the latent representations. However, as we demonstrate in this section, the model has difficulty in capturing multiple styles of motion due to its simple manifold structure and unimodal dynamics.

We used two publicly available GPDM implementations, each with its own suggested hyperparameters and structural settings. The first implementation was provided by Neil Lawrence's FGPLVM toolbox: `http://staffwww.dcs.shef.ac.uk/people/N.Lawrence/fgplvm`. Lawrence recommends fixing by hand, rather than optimizing, the hyperparameters of the dynamics GP (Lawrence, 2006). This ensures a strong preference for smooth trajectories. For the dynamics, we used the default compound (RBF + white noise) kernel with recommended hyperparameter settings of $\bar{\alpha} = [0.2, 0.001, 1 \times 10^{-6}]^T$. The observation model used a compound (RBF + bias + white noise) kernel whose hyperparameters were optimized.

The second implementation we employed was provided by Jack Wang: `http://www.dgp.toronto.edu/~jmwang/gpdm/`. This implementation differed from the first in a number of respects. First, it used a compound (linear + RBF + white noise) dynamics kernel whose hyperparameters were optimized rather than set by hand. The observation model used a compound (RBF + white noise) kernel whose hyperparameters were optimized. This GPDM also "balanced" the objective function by reweighting the dynamics term by the ratio of observed dimensions to latent dimensions (Wang et al., 2008). Similar to fixing hyperparameters, this encourages smoothness of the latent trajectories.

With each implementation we trained both a single model on the complete 10 walking styles data set as well as 10 style-specific models. The data was preprocessed identically to the CRBM experiments, however, we did not balance the data set by repeating sequences. It would have taken several weeks to train the GPDM on a corpora of approximately 30,000 frames. We tried each of the sparse approximations provided by the FGPLVM toolbox to reduce the computational complexity. In our experience, though drastically improving training time, all of the approximations led to far worse synthesized motion quality. In all results shown, we use the recommended 3 latent dimensions (Lawrence, 2006; Urtasun et al., 2006; Wang et al., 2008). We also experimented with 8 and 16 latent dimensions but found that this caused quality to decrease.

Similar to the online process used for drawing samples from a CRBM, we simulated the dynamical process one frame at a time, starting from training data (mapped to latent space). At each time step, we set the latent position to the mean latent position conditioned on the previous step. The latent trajectory then induces a per-frame Gaussian distribution over (normalized) poses (i.e., the reconstruction distribution). We take the mean of this distribution for each frame. Wang et al. (2008) recommend drawing fair samples of entire trajectories using hybrid Monte Carlo (HMC), using the simulated latent trajectory as an initialization. We did not observe any significant improvement in the quality of synthesized motion when using HMC. Moreover, it increased simulation time by an order of magnitude.

The supplemental videos show the result of synthesizing motion styles from the GPDM. For each model and style we show three sequences: 1) a sequence generated from the same initialization as we used for the CRBM; 2) the best sequence, as determined by visual inspection, over ten different initializations spaced uniformly over the training data; and 3) reconstructing the training data using the latent representation. We observed that when trained per-style, both implementations of the GPDM could generate reasonable-looking motion, though not of the same quality as a 1 or 2-layer CRBM trained on all styles. Regardless of the implementation, a single GPDM trained on all styles failed to generate satisfactory motion. More recent extensions of the GP-LVM, such as Topologically-constrained GP-LVMs (Urtasun et al., 2008), multifactor GPs (Wang et al., 2007) and hierarchical GP-LVMs (Lawrence and Moore, 2007) may perform better at this task.

3.5.7 FILLING IN MISSING DATA

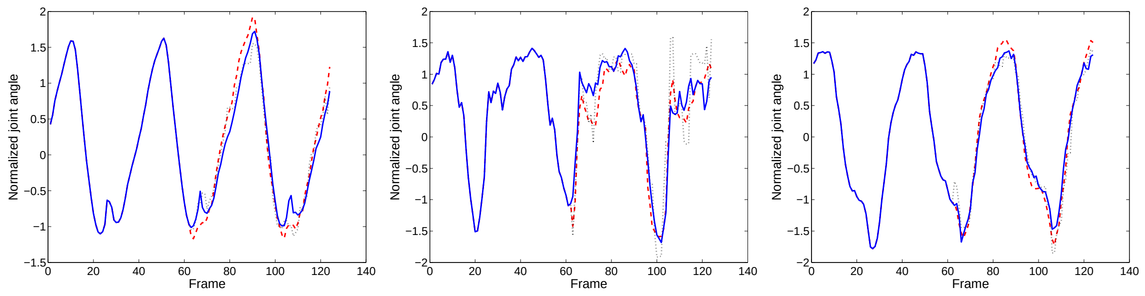

Figure 8: The model successfully fills in missing data using only the previous values of the joint angles (through the temporal connections) and the current angles of other joints (through the symmetric connections). Shown are the three angles of rotation for the left hip joint. The original data is shown as a solid line, the model's prediction is shown as a dashed line, and the results of nearest neighbour interpolation are shown as a dotted line.

Due to the nature of the motion capture process, which can be adversely affected by lighting and environmental effects, as well as noise during recording, motion capture data often contains missing or unusable data. Some markers may disappear ("dropout") for long periods of time due to sensor failure or occlusion. The majority of motion editing software packages contain interpolation methods to fill in missing data, but this leaves the data unnaturally smooth. These methods also rely on the starting and end points of the missing data. Hence, if a marker goes missing until the end of a sequence, naïve interpolation will not work. Such methods often only use the past and future data from the single missing marker to fill in that marker's missing values. Since joint angles are highly correlated, substantial information about the placement of one marker can be gained from the others. To demonstrate filling in, we trained a model exactly as described in Section 3.5.3, holding out one walking and one running sequence from the training data to be used as test data. For each of these walking and running test sequences, we erased two different sets of joint angles, starting halfway through the test sequence. These sets were the joints in (1) the left leg, and (2) the entire upper body. As seen in the supplemental video, the quality of the filled-in data is excellent and is hardly distinguishable from the original ground truth of the test sequence. Figure 8 demonstrates the model's ability to predict the three angles of rotation of the left hip.

We report results on the held-out walking sequence, of length 124 frames. We compared our model's performance to nearest neighbour interpolation, a simple method where for each frame, the values on known dimensions are compared to each example in the training set to find the closest match (measured by Euclidean distance in the normalized angle space). The unknown dimensions are then filled in using the matched example. As reconstruction from our model is stochastic, we repeated the experiment 100 times and report the mean. For the missing leg, mean squared reconstruction error per joint using our model was 8.78, measured in normalized joint angle space, and summed over the 62 frames of interest. Using nearest neighbour interpolation, the error was greater: 11.68. For the missing upper body, mean squared reconstruction error per joint using our model was 20.52. Using nearest neighbour interpolation, again the error was greater: 22.20.

We note that by adding additional neighbouring points, the nearest neighbour prediction can be significantly improved. For filling in the left leg, we found that $K = 8$ neighbours gave minimal error (8.63), while for the missing upper body, using $K = 6$ neighbours gave minimal error (12.67). These scores, especially in the case of the missing upper body, are, in fact, an improvement over using the CRBM for prediction. However, we note that in practice we would not be able to fine-tune the number of nearest neighbours nor could we be expected to have access to a large database of extremely similar training data. In more realistic missing-data scenarios, we would expect the model-based approach to generalize much better. Furthermore, we have not optimized other tunable parameters such as the model order, number of Gibbs steps per CD iteration, and number of hidden units; all of which are expected to have an impact on the prediction error.

4. Factored Conditional Restricted Boltzmann Machines

In this section we present a different model, based on the CRBM, that explicitly preserves the CRBM's most important computational properties but includes multiplicative three-way interactions that allow the effective interaction weight between two units to be modulated by the dynamic state of a third unit. We factor the three-way weight tensor implied by the multiplicative model, greatly reducing the number of parameters.

4.1 Multiplicative Interactions

A major motivation for the use of RBMs is that they can be used as the building blocks of deep belief networks (DBN), which are learned efficiently by training greedily, layer-by-layer (see Section 3.4). DBNs have been shown to learn very good generative models of handwritten digits (Hinton et al., 2006), but they have difficulty modeling patches of natural images. This is because RBMs have no simple way to capture the smoothness constraint in natural images: a single pixel can usually be predicted very accurately by simply interpolating its neighbours.

To address this concern, Osindero and Hinton (2008) introduced the semi-restricted Boltzmann machine (SRBM). In an SRBM, the constraints on the connectivity of the RBM are relaxed to allow lateral connections between the *visible* units in order to model the pair-wise correlations between inputs, thus allowing the hidden units to focus on modeling higher-order structure. Semi-restricted Boltzmann machines also permit deep networks. Each time a new level is added, the previous top layer of units is given lateral connections, so, after the layer-by-layer learning is complete, all layers except the topmost contain lateral connections between units. SRBMs make it possible to learn deep belief nets that model image patches much better, but they still have strong limitations that can be seen by considering the overall generative model. The equilibrium sample generated at each layer influences the layer below by controlling its effective biases. The model would be much more powerful if the equilibrium sample at the higher level could also control the lateral interactions at the layer below using a three-way, multiplicative relationship. Memisevic and Hinton (2007) introduced the gated CRBM, which permitted such multiplicative interactions and showed that it was able to learn rich distributed representations of image transformations (see Section 4.3).

In this section, we explore the idea of multiplicative interactions in the context of a different type of CRBM. Instead of gating lateral interactions with hidden units, we allow a set of real-valued style variables to gate the three types of connections: autoregressive, past to hidden, and visible to hidden within the CRBM. We will use the term "sub-model" to refer to a set of connections of a given type. Our modification of the CRBM architecture does not change the desirable properties related to inference and learning but allows the style variables to modulate the interactions in the model.

Like the CRBM, the multiplicative model is applicable to general time series where conditional data is available (e.g., seasonal variables for modeling rainfall occurrences, economic indicators for modeling financial instruments). However, we are largely motivated by our success thus far in modeling mocap data. In Section 3 we showed that a CRBM could capture many different styles with a single set of parameters. Generation of different styles was purely based on initialization, and the model architecture did not allow control of transitions between styles nor did it permit style blending. By using explicit style variables to gate the connections of a CRBM, we can obtain a much more powerful generative model that permits controlled transitioning and blending. We demonstrate that in a conditional model, the gating approach is superior to simply using labels to bias the hidden units, which is the approach most commonly used in static models (Hinton et al., 2006).

4.2 Style and Content Separation

There has been a significant amount of work on the separation of style and content in motion. The ability to separately specify the style (e.g., sad) and the content (e.g., walk to location A) is highly desirable for animators. One approach to style and content separation is to guide a factor model (e.g., PCA, factor analysis, ICA) by giving it "side-information" related to the structure of the data. Tenenbaum and Freeman (2000) considered the problem of extracting exactly two types of factors, namely style and content, using a bilinear model. In a bilinear model, the effect of each factor on the output is linear when the other is held fixed, but together the effects are multiplicative. This model can be learned efficiently, but supports only a rigid, discrete definition of style and content requiring that the data be organized in a (style $\times$ content) grid.

Previous work has looked at applying user-specified style to an existing motion sequence (Urtasun et al., 2004; Hsu et al., 2005; Torresani et al., 2007). The drawback to these approaches is that the user must provide the content. We propose a generative model for content that adapts to stylistic controls. Recently, models based on the Gaussian process latent variable model (Lawrence, 2004) have been successfully applied to capturing style in human motion (Wang et al., 2007). The advantage of our approach over such methods is that our model does not need to retain the training data set (just a few frames for initialization). Furthermore, training time increases linearly with the number of frames of training data, and so our model can scale up to massive data sets, unlike the kernel-based methods which are cubic in the number of frames. The powerful distributed hidden state of our model means that it does not suffer from the limited representational power of HMM-based methods of modeling style (e.g., Brand and Hertzmann, 2000).

4.3 Gated Conditional Restricted Boltzmann Machines

Memisevic and Hinton (2007) introduced a way of implementing multiplicative interactions in a conditional model. The gated CRBM was developed in the context of learning transformations between image pairs. The idea is to model an observation (the output) given its previous instance (the input). For example, the input and output might be neighbouring frames of video. The gated CRBM has two equivalent views: first, as gated regression (Figure 9a), where hidden units can blend "slices" of a transformation matrix into a linear regression, and second as modulated filters (Figure 9b) where input units gate a set of basis functions used to reconstruct the output. In the latter view, each setting of the input units defines an RBM. This means that conditional on the input, inference and learning in a gated CRBM are tractable.

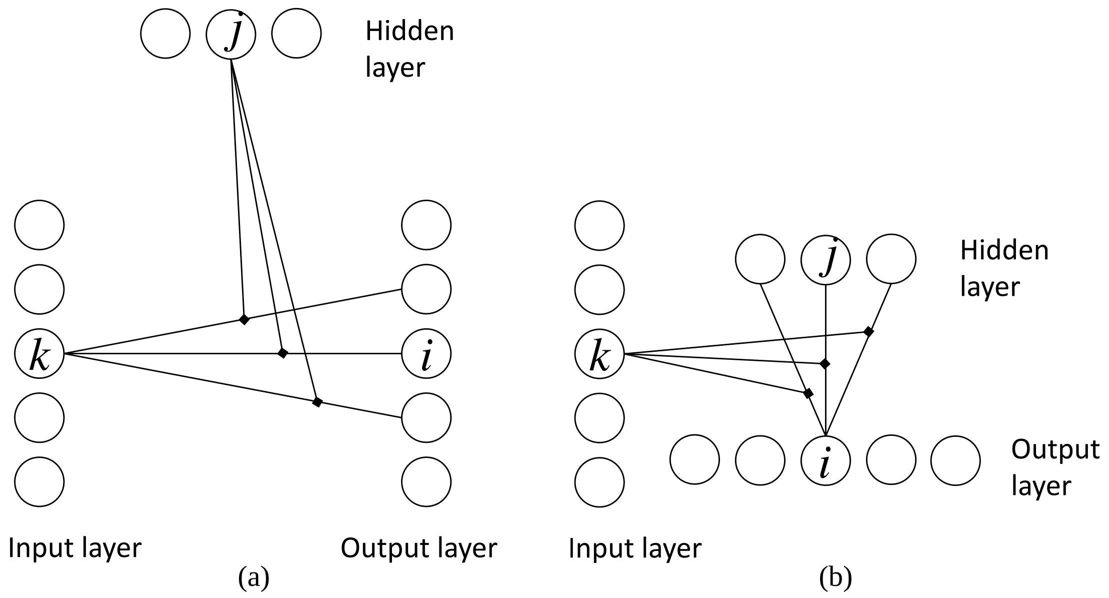

Figure 9: Two views of the gated CRBM, reproduced from the original paper (Memisevic and Hinton, 2007).

For ease of presentation, let us consider the case where all input, output, and hidden variables are binary (the extension to real-valued input and output variables is straightforward). As in Equation 15, the gated CRBM describes a joint probability distribution through exponentiating an energy function and renormalizing. This energy function captures all possible correlations between the components of the input, $\mathbf{x}$, the output, $\mathbf{v}$, and the hidden variables, $\mathbf{h}$:

$$E(\mathbf{v},\mathbf{h}|\mathbf{x}) = -\sum_{ijk} W_{ijk} v_i h_j x_k - \sum_{ij} c_{ij} v_i h_j - \sum_i a_i v_i - \sum_j b_j h_j \tag{20}$$

where a_i, b_j index the standard biases on each unit and c_{ij} index the gated biases, which shift the total input to a unit conditionally. The parameters W_{ijk} are the components of a three-way weight tensor. The CD weight updates for learning a gated CRBM are similar to a standard CRBM (Ackley et al., 1985). For example, the weight update rule for W_{ijk} is:

$$\Delta W_{ijk} \propto \langle v_i h_j x_k \rangle_{\text{data}} - \langle v_i h_j x_k \rangle_{\text{recon}}.$$

Considering the "modulated filters" view of the gated CRBM, we can fold the (given) inputs into the weights to express the input-dependent filters $\hat{W}_{ij} = \sum_k W_{ijk} x_k$. This allows us to rewrite the energy function (Equation 20) as:

$$E(\mathbf{v},\mathbf{h}|\mathbf{x}) = -\sum_{ij}\hat{W}_{ij}v_ih_j - \sum_{ij}c_{ij}v_ih_j - \sum_i a_iv_i - \sum_j b_jh_j.$$

Fixing the input, the first term is bilinear in **v** and **h**. Therefore at first glance, the model appears similar to the bilinear factor model (Tenenbaum and Freeman, 2000). However, the two models differ considerably in both their learning method and structure. Note that the bilinearity only occurs in the energy function: the gated CRBM permits the learned transformations to be highly nonlinear functions of the data.

4.4 Factoring

To model time-series, we can consider the output of a gated CRBM to be the current frame of data, $\mathbf{v} = \mathbf{v}_t$, and the input to be the previous frame (or frames), $\mathbf{x} = \mathbf{v}_{<t} = \mathbf{v}_{t-N:t-1}$. In this sense, the gated CRBM is a kind of autoregressive model where a transformation is composed from a set of basis transformations, with each binary hidden unit specifying whether or not to include one of the basis transformations. The number of possible compositions is exponential in the number of hidden units, but the componential nature of the hidden units prevents the number of parameters in the model from becoming exponential, as it would in a mixture model. Because of the three-way weight tensor, the number of parameters is cubic (assuming that the numbers of input, output and hidden units are comparable).

In many applications, including human motion modeling, strong underlying regularities in the data suggest that structure can be captured using three-way, multiplicative interactions but with less than the cubically many parameters implied by the weight tensor. This motivates us to factor the interaction tensor into a product of pairwise interactions (Figure 10). Factoring changes the energy function (Equation 20) to:

$$E(\mathbf{v},\mathbf{h}|\mathbf{x}) = -\sum_f \sum_{ijk} W^{\mathbf{v}}_{if}W^{\mathbf{h}}_{jf}W^{\mathbf{x}}_{kf}v_ih_jx_k - \sum_{ij}c_{ij}v_ih_j - \sum_i a_iv_i - \sum_j b_jh_j$$

where f indexes a set of deterministic factors. Superscripts differentiate the different types of pairwise interactions: $W^{\mathbf{v}}_{if}$ connect output units to factors (undirected), $W^{\mathbf{h}}_{jf}$ connect hidden units to factors (undirected), and $W^{\mathbf{x}}_{kf}$ connect input units to factors (directed).

The factors correspond to an intermediate layer of "simple cells" which modulate the interactions between units. Each factor is connected to all input units, all hidden units, and all output units. However, there are typically about as many factors as the number of each type of unit, so the introduction of factors corresponds to a kind of low-rank approximation to the interaction tensor, W, that uses about $3N^2$ parameters instead of N^3. Factors are deterministic, and are therefore very different than the visible and hidden units, which have stochastic states. Factors always send the product of the total input from two types of units to the remaining third type of unit. For example, during inference, each factor collects the total input arriving at it from the input and output layers, respectively, multiplies these quantities together, and sends this input to each hidden unit. During reconstruction, each factor collects the total input arriving at it from the input and hidden layers, respectively, multiplies these quantities together, and sends this input to each visible unit. This is in contrast to the visible and hidden units. These must be sampled before sending their stochastic states to the factors, and, unlike factors, they send the same message everywhere. Factors cannot be replaced by a layer of nonlinear stochastic units because this would prevent the hidden states from being conditionally independent.

Although factoring has been motivated by the introduction of multiplicative interactions, models that only involve pairwise interactions can also be factored (e.g., Salakhutdinov et al., 2007). To factor the CRBM, we change the energy function in Equation 16 to:

$$E(\mathbf{v}_t, \mathbf{h}_t | \mathbf{v}_{<t}) = \frac{1}{2}\sum_i (v_{i,t} - \hat{a}_{i,t})^2 - \sum_f \sum_{ij} W^{\mathbf{v}}_{if} W^{\mathbf{h}}_{jf} v_{i,t} h_{j,t} - \sum_i \hat{b}_{j,t} h_{j,t}$$

and additionally, factor the weights of the dynamic biases $\hat{\mathbf{a}}_t$ and $\hat{\mathbf{b}}_t$:

$$\hat{a}_{i,t} = a_i + \sum_m \sum_k A^{\mathbf{v}}_{im} A^{\mathbf{v}_{<t}}_{km} v_{k,<t},$$
$$\hat{b}_{j,t} = b_j + \sum_n \sum_k B^{\mathbf{h}}_{jn} B^{\mathbf{v}_{<t}}_{kn} v_{k,<t}.$$

The indices m and n correspond to the factoring of directed connections, A and B. We may use a different number of factors for each of the three different types of connections in the CRBM. This procedure can be seen as a kind of learned low-rank matrix factorization on each of W,A, and B.

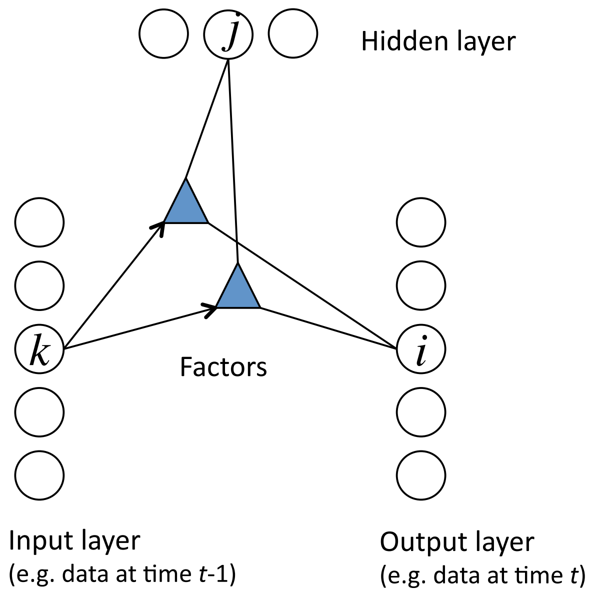

Figure 10: Factoring the gated CRBM.

4.5 A Style-Gated, Factored Model

We now consider modeling multiple styles of human motion using factored, multiplicative, three-way interactions. Hinton et al. (2006) showed that a good generative model of handwritten digits could be built by connecting a softmax label unit to the topmost hidden layer of a DBN (Figure 11a). After learning, clamping a label changes the energy landscape of the autoassociative model formed by the top two layers, so that performing alternating Gibbs sampling produces a joint sample compatible with a particular digit class. It is easy to extend this modification to the CRBM, where discrete style labels bias the hidden units. In a CRBM, however, the hidden units are also conditioned on information from the past that is much stronger than the information coming from the label (Figure 11b). The model has learned to respect consistency of styles between frames and so will resist a transition introduced by changing the label units.

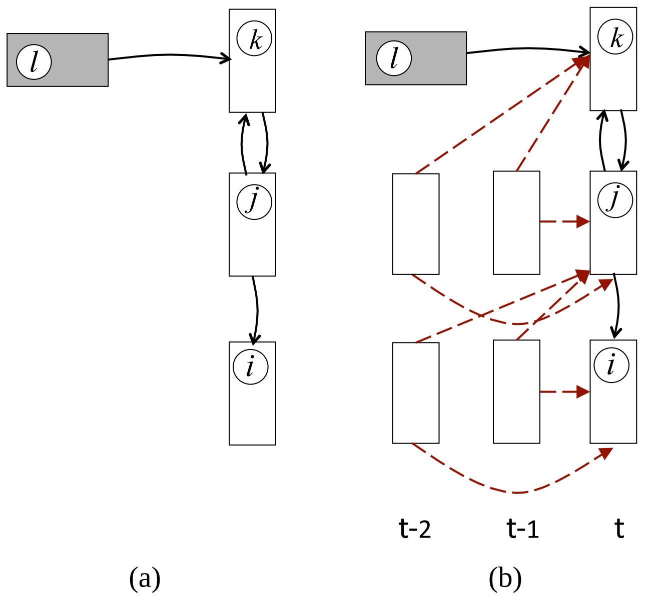

Figure 11: a) In a deep belief network, clamping the label units changes the energy function. b) In a conditional model, label information is swamped by the signal coming from the past.

As in the gated CRBM, we are motivated to let style change the *interactions* of the units as opposed to simply their effective biases. Memisevic and Hinton (2010) used factored three-way interactions to allow the hidden units of a gated CRBM to control the effect of one video frame on the subsequent video frame. Figure 12 shows a different way of using factored three-way interactions to allow real-valued style features, derived from discrete style labels, to control three different sets of pairwise interactions. Like the standard CRBM (Equation 15), the model defines a joint probability distribution over $\mathbf{v}_t$ and $\mathbf{h}_t$, conditional on the past N observations, $\mathbf{v}_{<t}$. However, the distribution is also conditional on the style labels, $\mathbf{y}_t$, through a set of deterministic, real-valued features, $\mathbf{z}_t$. The features are a linear function of the "one-hot" encoded style labels:

$$z_{l,t} = \sum_p R_{pl} y_{p,t}.$$

This resembles the use of componential word-features used in Mnih and Hinton's language model (Mnih and Hinton, 2007).

Similar to our discussion of the CRBM, we assume binary stochastic hidden units and real-valued visible units with additive Gaussian noise and $\sigma_i = 1$. The energy function is:

$$E\left(\mathbf{v}_t, \mathbf{h}_t | \mathbf{v}_{<t}, \mathbf{y}_t\right) = \frac{1}{2}\sum_i (v_{i,t} - \hat{a}_{i,t})^2 - \sum_f \sum_{ijl} W^{\mathbf{v}}_{if} W^{\mathbf{h}}_{jf} W^{\mathbf{z}}_{lf} v_{i,t} h_{j,t} z_{l,t} - \sum_j \hat{b}_{j,t} h_{j,t}. \qquad (21)$$

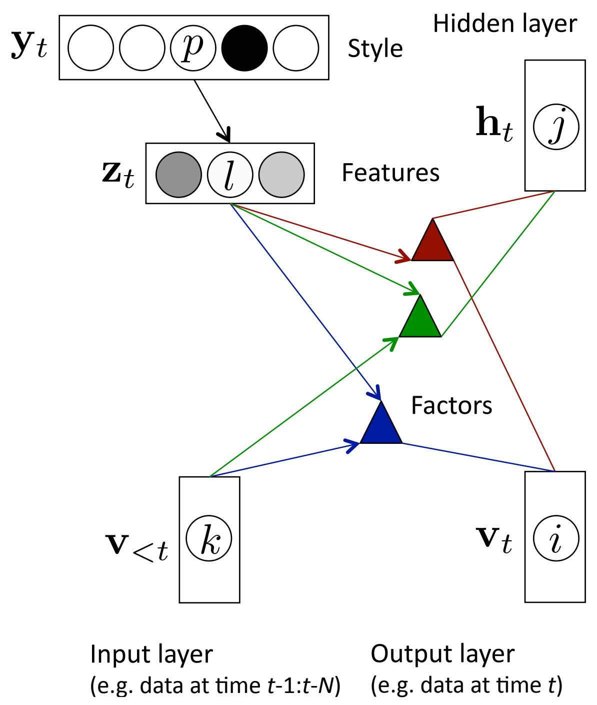

Figure 12: A factored CRBM whose interactions are gated by real-valued stylistic features.

The three terms in Equation 21 correspond to the three sub-models (the groups of links connected to each triangular factor in Figure 12). Note that for each sub-model, what was a matrix of weights is now replaced by three sets of weights connecting units to factors. The three types of weights are differentiated again by superscripts. For example, the matrix of undirected weights in the standard CRBM, W_{ij}, has been replaced by three matrices involved in a factored, multiplicative interaction: $W_{if}^{\mathbf{v}}$, $W_{jf}^{\mathbf{h}}$, and $W_{lf}^{\mathbf{z}}$. The same process is applied to the other two sub-models. Note that the three sub-models may have a different number of factors (which we index by f, m, and n).

The dynamic biases become:

$$\begin{aligned}\hat{a}_{i,t} &= a_i + \sum_m \sum_{kl} A_{im}^{\mathbf{v}} A_{km}^{\mathbf{v}_{<t}} A_{lm}^{\mathbf{z}} v_{k,<t} z_{l,t} \\ &= a_i + \sum_m A_{im}^{\mathbf{v}} \sum_k A_{km}^{\mathbf{v}_{<t}} v_{k,<t} \sum_l A_{lm}^{\mathbf{z}} z_{l,t}, \qquad (22)\\ \hat{b}_{j,t} &= b_j + \sum_n \sum_{kl} B_{jn}^{\mathbf{h}} B_{kn}^{\mathbf{v}_{<t}} B_{ln}^{\mathbf{z}} v_{k,<t} z_{l,t} \\ &= b_j + \sum_n B_{jn}^{\mathbf{h}} \sum_k B_{kn}^{\mathbf{v}_{<t}} v_{k,<t} \sum_l B_{ln}^{\mathbf{z}} z_{l,t} \qquad (23)\end{aligned}$$

where the dynamic component of Equation 22 and Equation 23 is simply the total input to the visible/hidden unit via the factors. The total input is a three-way product between the input to the factors (coming from the past and from the style features) and the weight from the factors to the visible/hidden unit. The dynamic biases include a static component, **a** and **b**. As in the gated

CRBM, we could also add three types of gated biases, corresponding to the pairwise interactions in each of the sub-models. In our experiments, we have not used any gated biases.

4.5.1 INFERENCE AND LEARNING

Adding multiplicative interactions to the model and factoring does not change the property that the posterior distribution is factorial. Inference is performed by considering, in parallel, the total input to each hidden unit via the factors:

$$p(h_{j,t}=1|\mathbf{v}_t,\mathbf{v}_{<t},\mathbf{y}_t)=\frac{1}{1+\exp(-\hat{b}_{j,t}-\sum_f W^{\mathbf{h}}_{jf}\sum_i W^{\mathbf{v}}_{ij}v_{i,t}\sum_l W^{\mathbf{z}}_{lf}z_{l,t})}$$

where $\hat{b}_{j,t}$ is defined in Equation 23. The reconstruction distribution is found by considering the total input to each visible unit via the factors:

$$p(v_{i,t}|\mathbf{h}_t,\mathbf{v}_{<t},\mathbf{y}_t)=\mathcal{N}\left(\hat{a}_{i,t}+\sum_f W^{\mathbf{v}}_{if}\sum_j W^{\mathbf{h}}_{jf}h_{j,t}\sum_l W^{\mathbf{z}}_{lf}z_{l,t},1\right)$$

where $\hat{a}_{i,t}$ is defined in Equation 22.

As in the other models based on RBMs, exact maximum likelihood learning is intractable. However, applying contrastive divergence leads to a set of very simple gradient update rules which are the same for binary or real-valued Gaussian visible units. The gradient with respect to a weight that connects a unit to a factor is the difference of two expectations of products. Each product involves three terms: the activity of the respective unit, and the total input to the factor from each of the two other sets of units involved in the three-way relationship. For example:

$$\Delta W^{\mathbf{v}}_{if}\propto\sum_t\left(\langle v_{i,t}\sum_j W^{\mathbf{h}}_{jf}h_{j,t}\sum_l W^{\mathbf{z}}_{lf}z_{l,t}\rangle_{\text{data}}-\langle v_{i,t}\sum_j W^{\mathbf{h}}_{jf}h_{j,t}\sum_l W^{\mathbf{z}}_{lf}z_{l,t}\rangle_{\text{recon}}\right).$$

The complete set of update rules is given in Appendix C.

The weights connecting labels to features, R, can simply be learned by backpropagating the gradients obtained by CD. Since these weights affect all three sub-models, their updates are more complicated. Applying the chain rule, we obtain:

$$\Delta R_{pl}\propto\sum_t\left(\langle C_{l,t}y_{p,t}\rangle_{\text{data}}-\langle C_{l,t}y_{p,t}\rangle_{\text{recon}}\right),$$
$$C_{l,t}=\sum_f W^{\mathbf{z}}_{lf}\sum_i W^{\mathbf{v}}_{if}v_{i,t}\sum_j W^{\mathbf{h}}_{jf}h_{j,t}+\sum_m A^{\mathbf{z}}_{lm}\sum_i A^{\mathbf{v}}_{im}v_{i,t}\sum_k A^{\mathbf{v}_{<t}}_{km}v_{k,<t}+\sum_n B^{\mathbf{z}}_{ln}\sum_j B^{\mathbf{h}}_{jn}h_{j,t}\sum_k B^{\mathbf{v}_{<t}}_{kn}v_{k,<t}.$$

The updates for the static biases on the hidden and visible biases are the same as in the standard CRBM (Equation 13 and 14).

4.5.2 PARAMETER SHARING

In addition to the large reduction in the number of free parameters obtained by factoring, further savings may be obtained by tying some sets of parameters together. In the fully parameterized model (Figure 13a), there are 9 different sets (matrices) of weights but if we restrict the number of factors to be the same for each of the three sub-models, four sets of parameters are identical in dimension: the weights that originate from the inputs (past visible units), the outputs (visible units), the hidden units and the features. Any combination of the compatible parameters may be tied. Figure 13b shows a fully-shared parameterization. This has slightly less than half the number of parameters of the fully parameterized model, assuming that the number of input, output, hidden, and feature units are comparable.

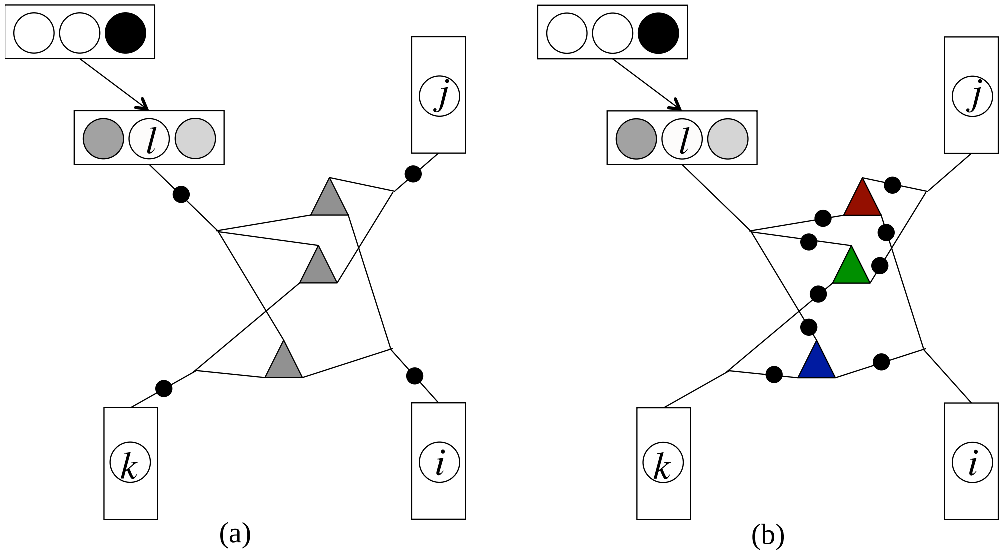

Figure 13: a) Fully parameterized model with each dot representing a different set of parameters and different colors denoting a different number of factors in each sub-model. b) Full parameter sharing where each dot represents a tied group of parameters. The number of factors is restricted to be the same for each sub-model.

In comparing different reduced parameterizations, tying only the feature-factor parameters, $W^{\mathbf{z}}_{lf}$,$A^{\mathbf{z}}_{lm}$, and $B^{\mathbf{z}}_{ln}$ led to models synthesizing the highest quality motion. When sharing the autoregressive weights $A^{\mathbf{v}_{<t}}_{km}$ and $A^{\mathbf{v}}_{im}$ with non-autoregressive weights $B^{\mathbf{v}_{<t}}_{km}$ and $W^{\mathbf{v}}_{if}$, respectively, we found that the component of the gradient related to the autoregressive model tended to dominate the weight update early in learning. This was due to the strength of the correlation between past and present compared to hidden and present or hidden and past. Witholding the autoregressive component of the gradient for the first 100 epochs, until the hidden units were able to extract interesting structure from the data, solved this problem. In our reported experiments we trained models with only the feature-factor parameters tied.

4.6 Experiments

CRBM models share a common deficiency: biasing the hidden units with a style label is not a true integration of context into their architecture. Despite our attempts, we cannot prevent spurious transitions (see Section 3.5.6), nor does a change of label during generation allow us to transition or blend between styles. We carry out a set of experiments that demonstrate that this shortcoming can be addressed by using factored, multiplicative interactions.

4.6.1 Modeling with Discrete Style Labels

Using the 10-styles data set described in Section 3.5.6, we trained a factored CRBM with Gaussian visible units whose parameters were gated by 100 real-valued features driven by discrete style labels (Figure 12). This model had 600 hidden units, 200 factors per sub-model and $N = 12$. Feature-to-factor parameters were also tied between sub-models. All parameters used a learning rate of 10^{-2}, except for the autoregressive parameters, $A^{\mathbf{v}}_{im}$, $A^{\mathbf{v}_{<t}}_{km}$, $A^{\mathbf{z}}_{lm}$ and the label-to-feature parameters, R_{pl}, which used a learning rate of 10^{-3}. After training the model for 500 epochs, we tested its ability to synthesize realistic motion by initializing with 12 frames of training data and holding the label units clamped to the matching style. The single-layer model was able to generate stylized content as well as the 2-layer standard CRBM (see the supplemental videos). In addition, we were able to induce transitions between two or more styles by linearly blending the discrete style label from one setting to another over 200 frames.[4] We were further able to blend together styles (like *sexy* and *strong*) by applying a linear interpolation of the discrete labels. The resulting motion was more natural when a single style was dominant (e.g., an 0.8/0.2 blend). We believe this is simply a case of better performance when the desired motion more closely resembles the cases present in the training data set, so training on a few examples of blends should greatly improve their generation.

4.6.2 Modeling with Real-Valued Style Parameters

The motions considered thus far have been described by a single, discrete label such as *gangly* or *drunk*. Motion style, however, can be characterized by multiple discrete labels or even continuous factors such as the level of flow, weight, time and space formally defined in Laban movement analysis (Torresani et al., 2007). In the case of multiple discrete labels, our real-valued feature units, $\mathbf{z}$, can receive input from multiple categories of labels. For continuous factors of style, we can connect real-valued style units to the real-valued feature units, or we can simply gate the model directly by the continuous description of style.

To test this latter configuration, we trained a model exactly as in Section 4.6.1, but instead of gating connections with 100 real-valued feature units, we gated with 2 real-valued style descriptors that were conditioned upon at every frame. Again we trained with walking data, but the data was captured specifically for this experiment. One style unit represented the speed of walking and the other, the stride length. The training data consisted of nine sequences at 60fps, each approximately 6000 frames corresponding to the cross-product of (slow, normal, fast) speed and (short,normal,long) stride length. The corresponding labels each had values of 1, 2 or 3. These values were chosen to avoid the special case of all gating units being set at zero and nullifying the effective weights of the model. The model was trained for 500 epochs.

After training, the model could, as before, generate realistic motion according to the nine discrete combinations of speed and stride-length with which it was trained based on initialization and setting the label units to match the labels in the training set. Furthermore, the model supported both interpolation and extrapolation along the speed and stride length axes and did not appear overly sensitive to initialization (see the supplemental videos).

4. The number of frames was selected empirically and provided a smooth transition, but the model is not sensitive to this number. A quick (e.g., frame-to-frame) change of labels will simply produce a "jerky" transition.

4.6.3 QUANTITATIVE EVALUATION

In our experiments so far, we have sought a qualitative comparison to the CRBM, based on the realism of synthesized motion. We have also focused on the ability of a factored model with multiplicative interactions to synthesize transitions as well as interpolate and extrapolate between styles present in the training data set. The application does not naturally present a quantitative comparison, but in the past, other time series models have been compared by their performance on the prediction of either full or partial held-out frames (e.g., Wang et al., 2008; Lawrence, 2007). We use the data set first proposed by Hsu et al. (2005) which consists of labeled sequences of seven types of walking: (*crouch*, *jog*, *limp*, *normal*, *side-right*, *sway*, *waddle*) each at three different speeds (*slow*, *medium*, *fast*). We preprocessed the data to remove missing or extremely noisy sections, and smoothed with a low-pass filter before downsampling from 120 to 30fps.

For each architecture: CRBM, factored CRBM, style-gated unfactored CRBM, and style-gated factored CRBM, we trained 21 different models on all style/speed pairs except one, which we held out for testing. Then, for each model, we attempted to predict every subsequence of length M in the test set, given the past $N = 6$ frames. We repeated the experiments for each architecture, each time reporting results averaged over the 21 models. Prediction could be performed by initializing with the previous frame and Gibbs sampling in the same way we generated, but this approach is subject to noise. We found that in all cases, integrating out the hidden units and following the gradient of the negative free energy with respect to the visible units gave less prediction error (see Section 3.3.3). We minimized the free energy using conjugate-gradient descent initialized with the previous frame. The architectures were subject to different learning rates and so the number of epochs for which to train each model was determined by setting aside 10% of the training set for validation.

We have also included a sixth-order autoregressive model as a baseline. This corresponds to the CRBM model without hidden units, except that it is trained using least squares instead of contrastive divergence.

Figure 14 presents the results. With almost half the number of free parameters, the 600-60 factored model performed as well as the fully parameterized CRBM. Gating with style information gives an advantage in longer-term prediction because it prevents the model from gradually changing the style. The unfactored model with style information performed slightly worse than the factored model and was extremely slow to train (it took two days to train whereas the other models were each trained in a few hours). The baseline autoregressive model performed extremely well in the short term, but was quickly eclipsed by the latent variable models for $N > 5$.

4.6.4 COMPUTATIONAL COMPLEXITY

The CRBM (1 or 2 layer) and FCRBM take a few hours to train on a modern single-core workstation. All of the models we have presented can generate motion at least as fast as 60fps (i.e., the visualizations we have produced were generated in real-time). Learning and inference in the CRBM and FCRBM are extremely efficient, with complexity linear in the number of training samples. In practice, this is slightly optimistic since larger and more complex data sets will require more hidden units, and learning and inference are also linear in the number of hidden units. The scale of corpora that we use in our experiments are problematic for GP-LVMs, since learning and inference for those models are $O(N^3)$ and $O(N^2)$, where N is the number of training samples.

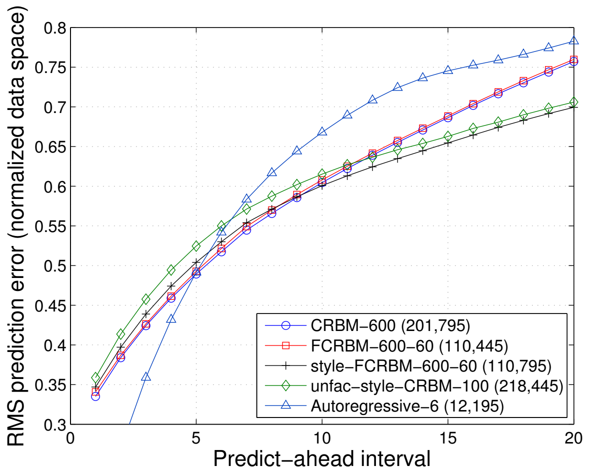

Figure 14: Prediction experiment. The number of free parameters are shown in parentheses. Error is reported in the normalized space in which the models are trained and is per-dimension, per-frame. The first two values for the autoregressive model (0.1506 and 0.2628) have been intentionally cut off.

5. Conclusion

We have introduced the conditional restricted Boltzmann machine (CRBM). The key properties of the CRBM are that it permits rich distributed representations to be learned from time series, and that exact inference is simple and efficient. We derived the contrastive divergence (CD) learning rules for CRBMs and showed how CRBMs can be stacked to form conditional deep belief nets. We demonstrated that a single model can generate many different styles of motion.

Perhaps the two greatest limitations of CRBMs (and RBMs in general) are first, evaluating the quality of trained models, and second, the learning algorithm with which they are trained. Though we have explored different methods of model evaluation, such as N-step forward prediction and the subjective assessment of synthesized data, the most natural way to evaluate a generative model is to compute the log-likelihood it assigns to a held-out test set. For all but the smallest models, this is impossible to do exactly due to the intractability of computing the partition function. Salakhutdinov and Murray (2008) have successfully applied annealed importance sampling (AIS) to RBMs. However, conditioning changes the partition function which implies that we would need to perform AIS for every possible configuration of N-frame histories (where N is the order of the CRBM) if we wish to evaluate the likelihood assigned by the model to an arbitrary sequence. Fortunately, to evaluate models we are often interested in computing likelihoods for a fixed test set rather than arbitrary sequences. This means that we need only to concern ourselves with conditioning on all possible N-frame histories in the test set. If we are evaluating M sequences whose maximum length is T, we would need to make on the order of $M(T-N)$ complete AIS estimates.[5]

A major criticism of contrastive divergence learning is that by "pulling up" on the energy of individual reconstructed data points, the algorithm fails to visit regions far away from the training

5. Note that for each of these "conditional" estimates we would still perform several runs of AIS.

data. Consequently, the bulk of the energy surface is left arbitrarily low. One solution is to abandon CD altogether, and pursue other learning methodologies, such as the sparse “energy-based methods” discussed in Section 3.5 or score matching (Hyvärinen, 2005). The alternative is to improve CD (e.g., Tieleman, 2008).

In Section 4 we extended the CRBM to permit context units to modulate the existing pairwise interactions. The resulting multiplicative model implies cardinality of parameters cubic in the number of units. However, we factorized the weights to make the parameterization quadratic and further reduced this number by tying weights. We demonstrated that the resulting model could capture several different motion styles, as well as transition and blend naturally between them. A sensible and natural extension of this work is to the fully unsupervised setting, where stylistic parameters are learned rather than provided (cf., Brand and Hertzmann, 2000).

Acknowledgments

An earlier version of this work appeared in two conference papers (Taylor et al., 2007; Taylor and Hinton, 2009). The authors thank NSERC and CIFAR for financial support. The authors also thank the anonymous reviewers for their helpful feedback. This work was primarily conducted while the first and third authors were at the University of Toronto.

Appendix A. Data Representation

The most statistically salient patterns of variation in the data may differ considerably from the patterns that humans find perceptually and expressively salient (Brand and Hertzmann, 2000). Therefore our learning algorithms can benefit from a carefully chosen representation that highlights important sources of variation and suppresses irrelevant sources of variation. Specifically, we aim to make our representation of motion invariant to rotation about the gravitational vertical (which we will simply call the vertical) and translation in the ground-plane. In the following discussion, we describe the steps taken to achieve a representation amenable to learning.

A.1 Original Representation

Data from a motion capture system typically consists of the 3D cartesian coordinates of 15-30 virtual markers (usually representing joint centres) for a series of discrete time-steps, which we call frames. The data is processed to remove missing and noisy markers and then converted to a joint angle hierarchy through an optimization that assumes constant limb lengths. For each frame, we obtain a vector of relative joint angle orientations, each 1-3 degrees of freedom (dof) plus a root orientation and translation in global coordinates (6 dof). The definition of the root depends on the data source, but typically it is the coccyx, near the base of the back. In our experiments, we used a variety of mocap sources, each of which provided the data already in a hierarchical “joint-angle” format.

A.2 Conversion to Exponential Maps

The most common representation for orientations in mocap data are Euler angles. Euler angles describe a one, two or three dof orientation by a sequence of rotations about axes in the global or local coordinate system. The order of rotations is user-defined and is a common source of confusion, often differing between data sources. Euler angles do not permit distances between rotations to be directly computed nor do they support interpolation or optimization since the orientation space is highly nonlinear. It is also not trivial to ensure that similar poses are expressed by similar Euler angles. Euler angles also suffer from "gimbal lock", the loss of rotational degrees of freedom due to singularities in the parameter space. Equivalent representations such as the 3×3 rotation matrix or 4D quaternion are not well suited to optimization and synthesis as they require additional constraints to ensure that they remain valid. Therefore we convert joint angles to an exponential map parameterization (Grassia, 1998) before learning.

The exponential map parameterization is also known as "axis-angle" representation since it consists of a three-element vector, whose direction specifies an axis of rotation and whose magnitude specifies the angle by which to rotate about this axis. Exponential maps are well suited to interpolation, optimization and unconstrained synthesis since they are locally linear and every three-element vector maps to a valid rotation. The parameterization still contains singularities and therefore is subject to gimbal lock, but the singularities in the exponential map are often avoidable (Grassia, 1998).[6]

A.3 Conversion to Body-Centred Orientations

We treat the root specially because it encodes a transformation with respect to a fixed global coordinate system. At each frame, t, this transformation can be described by a 3×3 rotation matrix, R_t, and a translation vector,[7] $\begin{bmatrix} x_t & y_t & z_t \end{bmatrix}^T$. We will assume, for our discussion, that z corresponds to the vertical. When R_t is the identity matrix, this defines the "rest position" which is typically defined by skeleton meta-data that accompanies the joint angles. Without loss of generality, let us assume that in the rest position the subject is axis-aligned such that the dorsoventral axis (from spinal column to belly) aligns with the x axis:

$$\mathbf{u}_t^0 = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}^T,$$

the lateral axis (from left to right side of body) aligns with the y axis:

$$\mathbf{v}_t^0 = \begin{bmatrix} 0 & 1 & 0 \end{bmatrix}^T,$$

and the anteroposterior axis (from head to feet) aligns with the negative z axis:

$$\mathbf{w}_t^0 = \begin{bmatrix} 0 & 0 & -1 \end{bmatrix}^T.$$

When the root is rotated (i.e., R_t is not the identity) the body-centred coordinate system is no longer axis aligned. It becomes:

$$\begin{aligned} \mathbf{u}_t &= R_t^T \mathbf{u}_t^0, \\ \mathbf{v}_t &= R_t^T \mathbf{v}_t^0, \\ \mathbf{w}_t &= R_t^T \mathbf{w}_t^0 \end{aligned}$$

6. For joints with a single degree of freedom, the exponential map reduces to an Euler angle and so we do not convert. The orientation of the root does not need to be converted to exponential maps since we build an alternative representation in the following section which requires the orientation to be expressed as a 3×3 rotation matrix.
7. It is also common to represent the transformation by a 4×4 matrix.

where we have assumed a particular convention for the rotation matrix. Note that these axes are simply the rows of the rotation matrix under our chosen convention.

Measuring the angle that the dorsoventral axis makes with the vertical gives us a measure of pitch:

$$\phi_t = \cos^{-1}\left(\frac{\mathbf{u}_t \cdot \mathbf{w}_t^0}{||\mathbf{u}_t||\,||\mathbf{w}_t^0||}\right) = \cos^{-1}\left(\frac{\mathbf{u}_t \cdot \mathbf{w}_t^0}{||\mathbf{u}_t||}\right).$$

Similarly, measuring the angle that the lateral axis makes with the gravitational vertical gives us a measure of roll:

$$\psi_t = \cos^{-1}\left(\frac{\mathbf{v}_t \cdot \mathbf{w}_t^0}{||\mathbf{v}_t||\,||\mathbf{w}_t^0||}\right) = \cos^{-1}\left(\frac{\mathbf{v}_t \cdot \mathbf{w}_t^0}{||\mathbf{v}_t||}\right).$$

Both pitch and roll are invariant to rotation about the vertical and therefore can be thought of as "body-centred" rotations. By projecting $\mathbf{u}_t$ into the ground-plane, this provides a measure of yaw, or rotation about the vertical:

$$\theta_t = \tan^{-1}\left(\frac{u_t^y}{u_t^x}\right)$$

where u_t^x and u_t^y are the first two components of vector $\mathbf{u}_t$. Care should be taken to use the four-quadrant version of $\tan^{-1}$ (often called the `atan2` function). We unwrap θ_t to eliminate discontinuities.

A.4 Conversion to Incremental Changes

We represent the rotation about the vertical, as well as translations in the ground plane by their incremental changes (forward differences) and not their absolute values:

$$\dot{\theta}_t = \theta_{t+1} - \theta_t,$$
$$\dot{x}_t = x_{t+1} - x_t,$$
$$\dot{y}_t = y_{t+1} - y_t.$$

For the last frame, we can use the two preceding frames to make a constant-velocity prediction. To achieve translational invariance, we need to express velocity in the ground-plane with respect to body-centred and not global coordinates. We can represent velocity in the ground-plane by its magnitude:

$$\alpha_t = \sqrt{\dot{x}_t^{\,2} + \dot{y}_t^{\,2}}$$

and its angle with respect to the *x*-axis:

$$\beta_t = \tan^{-1}\left(\frac{\dot{y}_t}{\dot{x}_t}\right).$$

Again we make use of the four-quadrant version of $\tan^{-1}$. The velocity is then expressed with respect to the orientation about the vertical, θ_t, in both a forward and lateral component:

$$\dot{\gamma}_t = \alpha_t \cos(\theta_t - \beta_t),$$
$$\dot{\xi}_t = \alpha_t \sin(\theta_t - \beta_t)$$

where we have used "dot" notation to imply that these quantities are incremental values. Taken collectively, $\left[\begin{array}{cccccc} \dot{\gamma}_t & \dot{\xi}_t & z_t & \phi_t & \psi_t & \dot{\theta}_t \end{array}\right]^T$ form our invariant representation of the root. Note that the height, z_t, is untouched.

A.5 Data Normalization

Any joint angle dimensions that have constant value are not modeled and removed from the training data (they are re-inserted before playback or export). Each component of the data is normalized to have zero mean and unit variance.

Appendix B. Approximations

In practice, we make several small modifications to the algorithms for both learning and generation. These rely on several approximations, most of which are chosen based on collective experience of training similar networks. The approximations typically replace sampled values with expected values, to reduce unnecessary noise.

While training a CRBM, we replace $v_{i,t}$ in Equation 10, 11 and 13 by its expected value and we also use the expected value of $v_{i,t}$ when computing the probability of activation of the hidden units (Equation 8). However, to compute each of the K reconstructions of the data (Equation 9), we use stochastically chosen binary values of the hidden units. This prevents the hidden activities from transmitting an unbounded amount of information from the data to the reconstruction (Teh and Hinton, 2001).

While updating the directed visible-to-hidden connections (Equation 12), the symmetric undirected connections (Equation 10), and the hidden biases (Equation 14), we use the stochastically chosen binary values of the hidden units in the first term (under the data), but replace $h_{j,t}$ by its expected value in the second term (under the reconstruction). We take this approach because the reconstruction of the data depends on the binary choices made when selecting hidden state. Thus, when we infer the hiddens from the reconstructed data, the probabilities are highly correlated with the binary hidden states inferred from the data. On the other hand, we stop after K reconstructions, so the binary choice of hiddens from the Kth reconstruction does not correlate with any other terms, and there is no reason to include this extra noise.

The alternating Gibbs sampling used when generating data is similar to the procedure we use to learn a CRBM. So we make similar approximations during generation: using stochastically chosen binary values of the hidden units but the expected values of the reconstructed visible units. As a further step to reduce noise, on the *final iteration* of Gibbs sampling, we use the real-valued probabilities of the hidden units when updating the visible units.

Appendix C. FCRBM Weight Updates

The CD updates for the parameters of the FCRBM have an intuitive form. The gradient with respect to a weight that connects a unit to a factor is the difference of two expectations of products. Each product involves three terms: the activity of the respective unit, and the total input to the factor from each of the two other sets of units involved in the three-way relationship:

$$\Delta W^{\mathbf{v}}_{if} \propto \sum_t \left(\langle v_{i,t} \sum_j W^{\mathbf{h}}_{jf} h_{j,t} \sum_l W^{\mathbf{z}}_{lf} z_{l,t} \rangle_{\text{data}} - \langle v_{i,t} \sum_j W^{\mathbf{h}}_{jf} h_{j,t} \sum_l W^{\mathbf{z}}_{lf} z_{l,t} \rangle_{\text{recon}} \right),$$

$$\Delta W^{\mathbf{h}}_{jf} \propto \sum_t \left(\langle h_{j,t} \sum_i W^{\mathbf{v}}_{if} v_{i,t} \sum_l W^{\mathbf{z}}_{lf} z_{l,t} \rangle_{\text{data}} - \langle h_{j,t} \sum_i W^{\mathbf{v}}_{if} v_{i,t} \sum_l W^{\mathbf{z}}_{lf} z_{l,t} \rangle_{\text{recon}} \right),$$

$$\Delta W^{\mathbf{z}}_{lf} \propto \sum_t \left(\langle z_{l,t} \sum_i W^{\mathbf{v}}_{if} v_{i,t} \sum_j W^{\mathbf{h}}_{jf} h_{j,t} \rangle_{\text{data}} - \langle z_{l,t} \sum_i W^{\mathbf{v}}_{if} v_{i,t} \sum_j W^{\mathbf{h}}_{jf} h_{j,t} \rangle_{\text{recon}} \right),$$

$$\Delta A^{\mathbf{v}}_{im} \propto \sum_t \left(\langle v_{i,t} \sum_k A^{\mathbf{v}_{<t}}_{km} v_{k,<t} \sum_l A^{\mathbf{z}}_{lm} z_{l,t} \rangle_{\text{data}} - \langle v_{i,t} \sum_k A^{\mathbf{v}_{<t}}_{km} v_{k,<t} \sum_l A^{\mathbf{z}}_{lm} z_{l,t} \rangle_{\text{recon}} \right),$$

$$\Delta A^{\mathbf{v}_{<t}}_{km} \propto \sum_t \left(\langle v_{k,<t} \sum_i A^{\mathbf{v}}_{im} v_{i,t} \sum_l A^{\mathbf{z}}_{lm} z_{l,t} \rangle_{\text{data}} - \langle v_{k,<t} \sum_i A^{\mathbf{v}}_{im} v_{i,t} \sum_l A^{\mathbf{z}}_{lm} z_{l,t} \rangle_{\text{recon}} \right),$$

$$\Delta A^{\mathbf{z}}_{lm} \propto \sum_t \left(\langle z_{l,t} \sum_i A^{\mathbf{v}}_{im} v_{i,t} \sum_k A^{\mathbf{v}_{<t}}_{km} v_{k,<t} \rangle_{\text{data}} - \langle z_{l,t} \sum_i A^{\mathbf{v}}_{im} v_{i,t} \sum_k A^{\mathbf{v}_{<t}}_{km} v_{k,<t} \rangle_{\text{recon}} \right),$$

$$\Delta B^{\mathbf{h}}_{jn} \propto \sum_t \left(\langle h_{j,t} \sum_k B^{\mathbf{v}_{<t}}_{kn} v_{k,<t} \sum_l B^{\mathbf{z}}_{ln} z_{l,t} \rangle_{\text{data}} - \langle h_{j,t} \sum_k B^{\mathbf{v}_{<t}}_{kn} v_{k,<t} \sum_l B^{\mathbf{z}}_{ln} z_{l,t} \rangle_{\text{recon}} \right),$$

$$\Delta B^{\mathbf{v}_{<t}}_{kn} \propto \sum_t \left(\langle v_{k,<t} \sum_j B^{\mathbf{h}}_{jn} h_{j,t} \sum_l B^{\mathbf{z}}_{ln} z_{l,t} \rangle_{\text{data}} - \langle v_{k,<t} \sum_j B^{\mathbf{h}}_{jn} h_{j,t} \sum_l B^{\mathbf{z}}_{ln} z_{l,t} \rangle_{\text{recon}} \right),$$

$$\Delta B^{\mathbf{z}}_{ln} \propto \sum_t \left(\langle z_{l,t} \sum_j B^{\mathbf{h}}_{jn} h_{j,t} \sum_k B^{\mathbf{v}_{<t}}_{kn} v_{k,<t} \rangle_{\text{data}} - \langle z_{l,t} \sum_j B^{\mathbf{h}}_{jn} h_{j,t} \sum_k B^{\mathbf{v}_{<t}}_{kn} v_{k,<t} \rangle_{\text{recon}} \right),$$

$$\Delta a_i \propto \sum_t \left(\langle v_{i,t} \rangle_{\text{data}} - \langle v_{i,t} \rangle_{\text{recon}} \right),$$

$$\Delta b_j \propto \sum_t \left(\langle h_{j,t} \rangle_{\text{data}} - \langle h_{j,t} \rangle_{\text{recon}} \right).$$

References

D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for Boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985.

O. Arikan and D. A. Forsyth. Interactive motion generation from examples. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, pages 483–490. ACM Press, 2002.

O. Arikan, D. A. Forsyth, and J. F. O'Brien. Motion synthesis from annotations. In *Proceedings of the 30th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2003)*, pages 402–408. ACM Press, 2003.

Y. Bengio and O. Delalleau. Justifying and generalizing contrastive divergence. *Neural Computation*, 21(1):1–21, 2008.

Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 153–160. MIT Press, 2007.

A. Bissacco. Modeling and learning contact dynamics in human motion. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, pages 421–428. IEEE, 2005.

M. Brand and A. Hertzmann. Style machines. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2000)*, pages 183–192. ACM Press, 2000.

M. Carreira-Perpinan and G. E. Hinton. On contrastive divergence learning. In *Proceedings of the 10th International Conference on Artificial Intelligence and Statistics (AISTATS 2005)*, pages 59–66, 2005.

Y. Freund and D. Haussler. Unsupervised learning of distributions of binary vectors using 2-layer networks. In J. Moody, S. H. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems (NIPS 4): Proceedings of the 1991 Conference*, pages 912–919. Morgan-Kaufmann, 1992.

P. V. Gehler, A. D. Holub, and M. Welling. The rate adapting poisson model for information retrieval and object recognition. In *Proceedings of the 23rd International Conference on Machine Learning (ICML 2006)*, pages 337–344. ACM Press, 2006.

Z. Ghahramani. Learning dynamic Bayesian networks. In C. Giles and M. Gori, editors, *Adaptive Processing of Sequences and Data Structures*, pages 168–197. Springer-Verlag, Berlin, 1998.

F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, 11(10): 428–434, 2007.

G. E. Hinton. *A practical guide to training restricted Boltzmann machines*. Technical Report UTML TR 2010-000, University of Toronto, 2010.

G. E. Hinton and A. D. Brown. Spiking Boltzmann machines. In S. A. Solla, T. K. Leen, and K.-R. Müller, editors, *Advances in Neural Information Processing Systems (NIPS 12): Proceedings of the 1999 Conference*, pages 122–128. MIT Press, 2000.

G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504 – 507, 2006.

G. E. Hinton, P. Dayan, B. J. Frey, and R. Neal. The wake-sleep algorithm for self-organizing neural networks. *Science*, 268:1158–1161, 1995.

G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

E. Hsu, K. Pulli, and J. Popović. Style translation for human motion. In *Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2005)*, pages 1082–1089. ACM Press, 2005.

A. Hyvärinen. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6:695–709, 2005. ISSN 1532-4435.

L. Kovar and M. Gleicher. Automated extraction and parameterization of motions in large data sets. In *Proceedings of the 31st Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2004)*, pages 559–568. ACM Press, 2004.

L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, pages 473–482. ACM Press, 2002.

H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 473–480. ACM Press, 2007.

N. D. Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In S. Thrun, L. Saul, and B. Schölkopf, editors, *Advances in Neural Information Processing Systems (NIPS 16): Proceedings of the 2003 Conference*, pages 329–326. MIT Press, 2004.

N. D. Lawrence. The Gaussian process latent variable model. Technical Report CS-06-05, University of Sheffield, 2006.

N. D. Lawrence. Learning for larger datasets with the Gaussian process latent variable model. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, pages 243–250, 2007.

N. D. Lawrence and A. J. Moore. Hierarchical Gaussian process latent variable models. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 481–488. ACM Press, 2007.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999.

H. Lee, C. Ekanadham, and A. Y. Ng. Sparse deep belief net model for visual area V2. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*, pages 873–880. MIT Press, 2008.

J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, pages 491–500. ACM Press, 2002.

Y. Li, T. Wang, and H.-Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, pages 465–472. ACM Press, 2002.

C. K. Liu, A. Hertzmann, and Z. Popovic. Learning physics-based motion style with nonlinear inverse optimization. In *Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2005)*, pages 1071–1081. ACM Press, 2005.

K. Matsuoka. Noise injection into inputs in back-propagation learning. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(3):436–440, 1992.

R. Memisevic and G. E. Hinton. Unsupervised learning of image transformations. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)*, 2007.

R. Memisevic and G. E. Hinton. Learning to represent spatial transformations with factored higher-order boltzmann machines. *Neural Computation*, 22:1473–1492, 2010.

A. Mnih and G. E. Hinton. Three new graphical models for statistical language modelling. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 641–648. ACM Press, 2007.

T. Mukai and S. Kuriyama. Geostatistical motion interpolation. In *Proceedings of the 32nd Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2005)*, pages 1062–1070. ACM Press, 2005.

K. P. Murphy. *Dynamic bayesian networks : representation, inference and learning*. PhD thesis, University of California, Berkeley, 2002.

V. Nair and G. E. Hinton. 3-d object recognition with deep belief nets. In *Advances in Neural Information Processing Systems* **22**, pages 1339–1347. MIT Press, Cambridge, MA, 2009.

R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Publishers, 1998.

B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37(23):3311–3325, 1997.

S. Osindero and G. E. Hinton. Modeling image patches with a directed hierarchy of Markov random fields. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*, pages 1121–1128. MIT Press, 2008.

S. I. Park, H. J. Shin, and S. Y. Shin. On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA 02)*, pages 105–111. ACM Press, 2002.

V. Pavlovic, J. M. Rehg, and J. MacCormick. Learning switching linear models of human motion. In *Advances in Neural Information Processing Systems (NIPS 13): Proceedings of the 2000 Conference*, pages 981–987. MIT Press, 2001.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, Santa Mateo, CA, USA, September 1988.

K. Pullen and C. Bregler. Motion capture assisted animation: Texturing and synthesis. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH 2002)*, pages 501–508. ACM Press, 2002.

M. Ranzato, C. S. Poultney, S. Chopra, and Y. LeCun. Efficient learning of sparse representations with an energy-based model. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 1137–1144. MIT Press, 2006.

M. Ranzato, Y. Boureau, S. Chopra, and Y. LeCun. A unified energy-based framework for unsupervised learning. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.

M. Ranzato, Y. Boureau, and Y. LeCun. Sparse feature learning for deep belief networks. In J. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems (NIPS 20): Proceedings of the 2007 Conference*. MIT Press, 2008.

C. Rose, M. F. Cohen, and B. Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications*, 18(5):32–40, 1998.

R. Salakhutdinov and I. Murray. On the quantitative analysis of deep belief networks. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 872–879. ACM Press, 2008.

R. Salakhutdinov, A. Mnih, and G. E. Hinton. Restricted Boltzmann machines for collaborative filtering. In Z. Ghahramani, editor, *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 791–798. ACM Press, 2007.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, MA, 1986.

I. Sutskever and G. E. Hinton. Learning multilevel distributed representations for high-dimensional sequences. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS 2007)*, 2007.

I. Sutskever and G. E. Hinton. Deep narrow sigmoid belief networks are universal approximators. *Neural Computation*, 20(11):2629–2636, 2008.

I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted Boltzmann machine. In *Advances in Neural Information Processing Systems (NIPS 21): Proceedings of the 2008 Conference*, volume 21. MIT Press, 2009.

L. Tanco and A. Hilton. Realistic synthesis of novel human movements from a database of motion capture examples. In *Proceedings of the Workshop on Human Motion (HUMO ’00)*, pages 137–142. IEEE Computer Society, 2000.

G. Taylor and G. Hinton. Factored conditional restricted Boltzmann machines for modeling motion style. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*, pages 1025–1032, 2009.

G. W. Taylor, G. E. Hinton, and S. T. Roweis. Modeling human motion using binary latent variables. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 1345–1352. MIT Press, 2007.

Y. W. Teh and G. E. Hinton. Rate-coded restricted Boltzmann machines for face recognition. In *Advances in Neural Information Processing Systems (NIPS 13): Proceedings of the 2000 Conference*. MIT Press, 2001.

J. B. Tenenbaum and W. T. Freeman. Separating style and content with bilinear models. *Neural Computation*, 12(6):1247–1283, 2000.

T. Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 1064–1071. ACM Press, 2008.

T. Tieleman and G. E. Hinton. Using fast weights to improve persistent contrastive divergence. In *Proceedings of the 26th International Conference on Machine Learning (ICML 2009)*. ACM Press, 2009.

L. Torresani, P. Hackney, and C. Bregler. Learning motion style synthesis from perceptual observations. In B. Schölkopf, J. C. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems (NIPS 19): Proceedings of the 2006 Conference*, pages 1393–1400. MIT Press, 2007.

R. Urtasun, P. Glardon, R. Boulic, D. Thalmann, and P. Fua. Style-based motion synthesis. *Computer Graphics Forum*, 23(4):1–14, 2004.

R. Urtasun, D. Fleet, and P. F. P. 3D people tracking with gaussian process dynamical models. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2006)*, pages 238–245. IEEE, 2006.

R. Urtasun, D. J. Fleet, A. Geiger, J. Popović, T. Darrell, and N. D. Lawrence. Topologically-constrained latent variable models. In *Proceedings of the 25th International Conference on Machine Learning (ICML 2008)*, pages 1080–1087. ACM Press, 2008.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Multifactor Gaussian process models for style-content separation. In *Proceedings of the 24th International Conference on Machine Learning (ICML 2007)*, pages 975–982. ACM Press, 2007.

J. M. Wang, D. J. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(2):283–298, 2008.

M. Welling, M. Rosen-Zvi, and G. E. Hinton. Exponential family harmoniums with an application to information retrieval. In *Advances in Neural Information Processing Systems (NIPS 17): Proceedings of the 2004 Conference*, pages 1481–1488. MIT Press, 2005.