Synthesizing Dynamic Patterns by Spatial-Temporal Generative ConvNet

Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu
University of California, Los Angeles (UCLA), USA
jianwen@ucla.edu, sczhu@stat.ucla.edu, ywu@stat.ucla.edu

Abstract

Video sequences contain rich dynamic patterns, such as dynamic texture patterns that exhibit stationarity in the temporal domain, and action patterns that are non-stationary in either spatial or temporal domain. We show that a spatial-temporal generative ConvNet can be used to model and synthesize dynamic patterns. The model defines a probability distribution on the video sequence, and the log probability is defined by a spatial-temporal ConvNet that consists of multiple layers of spatial-temporal filters to capture spatial-temporal patterns of different scales. The model can be learned from the training video sequences by an "analysis by synthesis" learning algorithm that iterates the following two steps. Step 1 synthesizes video sequences from the currently learned model. Step 2 then updates the model parameters based on the difference between the synthesized video sequences and the observed training sequences. We show that the learning algorithm can synthesize realistic dynamic patterns.

1. Introduction

There are a wide variety of dynamic patterns in video sequences, including dynamic textures [2] or textured motions [24] that exhibit statistical stationarity or stochastic repetitiveness in the temporal dimension, and action patterns that are non-stationary in either spatial or temporal domain. Synthesizing and analyzing such dynamic patterns has been an interesting problem. In this paper, we focus on the task of synthesizing dynamic patterns using a generative version of the convolutional neural network (ConvNet or CNN).

The ConvNet [14, 12] has proven to be an immensely successful discriminative learning machine. The convolution operation in the ConvNet is particularly suited for signals such as images, videos and sounds that exhibit translation invariance either in the spatial domain or the temporal domain or both. Recently, researchers have become increasingly interested in the generative aspects of ConvNet, for the purpose of visualizing the knowledge learned by the ConvNet, or synthesizing realistic signals, or developing generative models that can be used for unsupervised learning.

In terms of synthesis, various approaches based on the ConvNet have been proposed to synthesize realistic static images [3, 7, 1, 13, 16]. However, there has not been much work in the literature on synthesizing dynamic patterns based on the ConvNet, and this is the focus of the present paper.

Specifically, we propose to synthesize dynamic patterns by generalizing the generative ConvNet model recently proposed by [29]. The generative ConvNet can be derived from the discriminative ConvNet. It is a random field model or an energy-based model [15, 20] that is in the form of exponential tilting of a reference distribution such as the Gaussian white noise distribution or the uniform distribution. The exponential tilting is parametrized by a ConvNet that involves multiple layers of linear filters and rectified linear units (ReLU) [12], which seek to capture features or patterns at different scales.

The generative ConvNet can be sampled by the Langevin dynamics. The model can be learned by the stochastic gradient algorithm [31]. It is an "analysis by synthesis" scheme that seeks to match the synthesized signals generated by the Langevin dynamics to the observed training signals. Specifically, the learning algorithm iterates the following two steps after initializing the parameters and the synthesized signals. Step 1 updates the synthesized signals by the Langevin dynamics that samples from the currently learned model. Step 2 then updates the parameters based on the difference between the synthesized data and the observed data in order to shift the density of the model from the synthesized data towards the observed data. It is shown by [29] that the learning algorithm can synthesize realistic spatial image patterns such as textures and objects.

In this article, we generalize the spatial generative ConvNet by adding the temporal dimension, so that the resulting ConvNet consists of multiple layers of spatial-temporal filters that seek to capture spatial-temporal patterns at various scales. We show that the learning algorithm for training the spatial-temporal generative ConvNet can synthesize realistic dynamic patterns. We also show that it is possible to learn the model from incomplete video sequences with either occluded pixels or missing frames, so that model learning and

pattern completion can be accomplished simultaneously.

2. Related work

Our work is a generalization of the generative ConvNet model of [29] by adding the temporal dimension. [29] did not work on dynamic patterns such as those in the video sequences. The spatial-temporal discriminative ConvNet was used by [11] for analyzing video data. The connection between discriminative ConvNet and generative ConvNet was studied by [29].

Dynamic textures or textured motions have been studied by [2, 24, 25, 9]. For instance, [2] proposed a vector auto-regressive model coupled with frame-wise dimension reduction by single value decomposition. It is a linear model with Gaussian innovations. [24] proposed a dynamic model based on sparse linear representation of frames. See [30] for a recent review of dynamic textures. The spatial-temporal generative ConvNet is a non-linear and non-Gaussian model and is expected to be more flexible in capturing complex spatial-temporal patterns in dynamic textures with multiple layers of non-linear spatial-temporal filters.

Recently [23] generalized the generative adversarial networks [6] to model dynamic patterns. Our model is an energy-based model and it also has an adversarial interpretation. See section 3.4 for details.

For temporal data, a popular model is the recurrent neural network [27, 10]. It is a causal model and it requires a starting frame. In contrast, our model is non-causal, and does not require a starting frame. Compared to the recurrent network, our model is more convenient and direct in capturing temporal patterns at multiple time scales.

3. Spatial-temporal generative ConvNet

3.1. Spatial-temporal filters

To fix notation, let $\mathbf{I}(x, t)$ be an image sequence of a video defined on the square (or rectangular) image domain $\mathcal{D}$ and the time domain $\mathcal{T}$, where $x = (x_1, x_2) \in \mathcal{D}$ indexes the coordinates of pixels, and $t \in \mathcal{T}$ indexes the frames in the video sequence. We can treat $\mathbf{I}(x, t)$ as a three dimensional function defined on $\mathcal{D} \times \mathcal{T}$. For a spatial-temporal filter F, we let $F * \mathbf{I}$ denote the filtered image sequence or feature map, and let $[F * \mathbf{I}](x, t)$ denote the filter response or feature at pixel x and time t.

The spatial-temporal ConvNet is a composition of multiple layers of linear filtering and ReLU non-linearity, as expressed by the following recursive formula:

$$[F_k^{(l)} * \mathbf{I}](x, t) = h\Big(\sum_{i=1}^{N_{l-1}} \sum_{(y,s)\in\mathcal{S}_l} w_{i,y,s}^{(l,k)} \times [F_i^{(l-1)} * \mathbf{I}](x + y, t + s) + b_{l,k}\Big), \tag{1}$$

where $l \in \{1, 2, ..., L\}$ indexes the layers. $\{F_k^{(l)}, k = 1, ..., N_l\}$ are the filters at layer l, and $\{F_i^{(l-1)}, i = 1, ..., N_{l-1}\}$ are the filters at layer $l - 1$. k and i are used to index filters at layers l and $l - 1$ respectively, and N_l and N_{l-1} are the numbers of filters at layers l and $l - 1$ respectively. The filters are locally supported, so the range of (y, s) is within a local support $\mathcal{S}_l$ (such as a $7 \times 7 \times 3$ box of image sequence). The weight parameters $(w_{i,y,s}^{(l,k)}, (y, s) \in \mathcal{S}_l, i = 1, ..., N_{l-1})$ define a linear filter that operates on $(F_i^{(l-1)} * \mathbf{I}, i = 1, ..., N_{l-1})$. The linear filtering operation is followed by ReLU $h(r) = \max(0, r)$. At the bottom layer, $[F_k^{(0)} * \mathbf{I}](x, t) = \mathbf{I}_k(x, t)$, where $k \in \{\mathrm{R}, \mathrm{G}, \mathrm{B}\}$ indexes the three color channels. Subsampling may be implemented so that in $[F_k^{(l)} * \mathbf{I}](x, t)$, $x \in \mathcal{D}_l \subset \mathcal{D}$, and $t \in \mathcal{T}_l \subset \mathcal{T}$.

The spatial-temporal filters at multiple layers are expected to capture the spatial-temporal patterns at multiple scales. It is possible that the top-layer filters are fully connected in the spatial domain as well as the temporal domain (e.g., the feature maps are 1×1 in the spatial domain) if the dynamic pattern does not exhibit spatial or temporal stationarity.

3.2. Spatial-temporal generative ConvNet

The spatial-temporal generative ConvNet is an energy-based model or a random field model defined on the image sequence $\mathbf{I} = (\mathbf{I}(x, t), x \in \mathcal{D}, t \in \mathcal{T})$. It is in the form of exponential tilting of a reference distribution $q(\mathbf{I})$:

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp\left[f(\mathbf{I}; w)\right] q(\mathbf{I}), \tag{2}$$

where the scoring function $f(\mathbf{I}; w)$ is

$$f(\mathbf{I}; w) = \sum_{k=1}^{K} \sum_{x\in\mathcal{D}_L} \sum_{t\in\mathcal{T}_L} [F_k^{(L)} * \mathbf{I}](x, t), \tag{3}$$

where w consists of all the weight and bias terms that define the filters $(F_k^{(L)}, k = 1, ..., K = N_L)$ at layer L, and q is the Gaussian white noise model, i.e.,

$$q(\mathbf{I}) = \frac{1}{(2\pi\sigma^2)^{|\mathcal{D}\times\mathcal{T}|/2}} \exp\left[-\frac{1}{2\sigma^2}||\mathbf{I}||^2\right], \tag{4}$$

where $|\mathcal{D} \times \mathcal{T}|$ counts the number of pixels in the domain $\mathcal{D} \times \mathcal{T}$. Without loss of generality, we shall assume $\sigma^2 = 1$.

The scoring function $f(\mathbf{I}; w)$ in (3) tilts the Gaussian reference distribution into a non-Gaussian model. In fact, the purpose of $f(\mathbf{I}; w)$ is to identify the non-Gaussian spatial-temporal features or patterns. In the definition of $f(\mathbf{I}; w)$ in (3), we sum over the filter responses at the top layer L over all the filters, positions and times. The spatial and temporal pooling reflects the fact that we assume the model is stationary in spatial and temporal domains. If the dynamic

texture is non-stationary in the spatial or temporal domain, then the top layer filters $F_k^{(L)}$ are fully connected in the spatial or temporal domain, e.g., $\mathcal{D}_L$ is 1×1.

A simple but consequential property of the ReLU nonlinearity is that $h(r) = \max(0, r) = 1(r > 0)r$, where $1()$ is the indicator function, so that $1(r > 0) = 1$ if $r > 0$ and 0 otherwise. As a result, the scoring function $f(\mathbf{I}; w)$ is piecewise linear [17], and each linear piece is defined by the multiple layers of binary activation variables $\delta_{k,x,t}^{(l)}(\mathbf{I}; w) = 1\left([F_k^{(l)} * \mathbf{I}](x, t) > 0\right)$, which tells us whether a local spatial-temporal pattern represented by the k-th filter at layer l, $F_k^{(l)}$, is detected at position x and time t. Let $\delta(\mathbf{I}; w) = \left(\delta_{k,x,t}^{(l)}(\mathbf{I}; w), \forall l, k, x, t\right)$ be the activation pattern of $\mathbf{I}$. Then $\delta(\mathbf{I}; w)$ divides the image space into a large number of pieces according to the value of $\delta(\mathbf{I}; w)$. On each piece of image space with fixed $\delta(\mathbf{I}; w)$, the scoring function $f(\mathbf{I}; w)$ is linear, i.e.,

$$f(\mathbf{I}; w) = a_{w,\delta(\mathbf{I};w)} + \langle \mathbf{I}, B_{w,\delta(\mathbf{I};w)} \rangle, \tag{5}$$

where both a and B are defined by $\delta(\mathbf{I}; w)$ and w. In fact, $B = \partial f(\mathbf{I}; w)/\partial \mathbf{I}$, and can be computed by back-propagation, with $h'(r) = 1(r > 0)$. The back-propagation process defines a top-down deconvolution process [32], where the filters at multiple layers become the basis functions at those layers, and the activation variables at different layers in $\delta(\mathbf{I}; w)$ become the coefficients of the basis functions in the top-down deconvolution.

$p(\mathbf{I}; w)$ in (2) is an energy-based model [15, 20], whose energy function is a combination of the ℓ_2 norm $\|\mathbf{I}\|^2$ that comes from the reference distribution $q(\mathbf{I})$ and the piecewise linear scoring function $f(\mathbf{I}; w)$, i.e.,

$$\begin{aligned} \mathcal{E}(\mathbf{I}; w) &= -f(\mathbf{I}; w) + \frac{1}{2}\|\mathbf{I}\|^2 \\ &= \frac{1}{2}\|\mathbf{I}\|^2 - \left(a_{w,\delta(\mathbf{I};w)} + \langle \mathbf{I}, B_{w,\delta(\mathbf{I};w)} \rangle\right) \\ &= \frac{1}{2}\|\mathbf{I} - B_{w,\delta(\mathbf{I};w)}\|^2 + \text{const}, \end{aligned} \tag{6}$$

where $\text{const} = -a_{w,\delta(\mathbf{I};w)} - \|B_{w,\delta(\mathbf{I};w)}\|^2/2$, which is constant on the piece of image space with fixed $\delta(\mathbf{I}; w)$.

Since $\mathcal{E}(\mathbf{I}; w)$ is a piecewise quadratic function, $p(\mathbf{I}; w)$ is piecewise Gaussian. On the piece of image space $\{\mathbf{I} : \delta(\mathbf{I}; w) = \delta\}$, where δ is a fixed value of $\delta(\mathbf{I}; w)$, $p(\mathbf{I}; w)$ is $\text{N}(B_{w,\delta}, \mathbf{1})$ truncated to $\{\mathbf{I} : \delta(\mathbf{I}; w) = \delta\}$, where we use $\mathbf{1}$ to denote the identity matrix. If the mean of this Gaussian piece, $B_{w,\delta}$, is within $\{\mathbf{I} : \delta(\mathbf{I}; w) = \delta\}$, then $B_{w,\delta}$ is also a local mode, and this local mode $\mathbf{I}$ satisfies a hierarchical auto-encoder, with a bottom-up encoding process $\delta = \delta(\mathbf{I}; w)$, and a top-down decoding process $\mathbf{I} = B_{w,\delta}$. In general, for an image sequence $\mathbf{I}$, $B_{w,\delta(\mathbf{I};w)}$ can be considered a reconstruction of $\mathbf{I}$, and this reconstruction is exact if $\mathbf{I}$ is a local mode of $\mathcal{E}(\mathbf{I}; w)$.

3.3. Sampling and learning algorithm

One can sample from $p(\mathbf{I}; w)$ of model (2) by the Langevin dynamics:

$$\mathbf{I}_{\tau+1} = \mathbf{I}_\tau - \frac{\epsilon^2}{2}\left[\mathbf{I}_\tau - \mathbf{B}_{w,\delta(\mathbf{I}_\tau;w)}\right] + \epsilon Z_\tau, \tag{7}$$

where τ indexes the time steps, ϵ is the step size, and $Z_\tau \sim \text{N}(0, \mathbf{1})$. The dynamics is driven by the reconstruction error $\mathbf{I} - \mathbf{B}_{w,\delta(\mathbf{I};w)}$. The finiteness of the step size ϵ can be corrected by a Metropolis-Hastings acceptance-rejection step. The Langevin dynamics can be extended to Hamiltonian Monte Carlo [18] or more sophisticated versions [5].

The learning of w from training image sequences $\{\mathbf{I}_m, m = 1, ..., M\}$ can be accomplished by the maximum likelihood. Let $L(w) = \sum_{m=1}^M \log p(\mathbf{I}; w)/M$, with $p(\mathbf{I}; w)$ defined in (2),

$$\frac{\partial L(w)}{\partial w} = \frac{1}{M}\sum_{m=1}^M \frac{\partial}{\partial w} f(\mathbf{I}_m; w) - \text{E}_w\left[\frac{\partial}{\partial w} f(\mathbf{I}; w)\right]. \tag{8}$$

The expectation can be approximated by the Monte Carlo samples [31] produced by the Langevin dynamics. See Algorithm 1 for a description of the learning and sampling algorithm. The algorithm keeps synthesizing image sequences from the current model, and updating the model parameters in order to match the synthesized image sequences to the observed image sequences. The learning algorithm keeps shifting the probability density or low energy regions of the model from the synthesized data towards the observed data.

In the learning algorithm, the Langevin sampling step involves the computation of $\partial f(\mathbf{I}; w)/\partial \mathbf{I}$, and the parameter updating step involves the computation of $\partial f(\mathbf{I}; w)/\partial w$. Because of the ConvNet structure of $f(\mathbf{I}; w)$, both gradients can be computed efficiently by back-propagation, and the two gradients share most of their chain rule computations in back-propagation. In term of MCMC sampling, the Langevin dynamics samples from an evolving distribution because $w^{(t)}$ keeps changing. Thus the learning and sampling algorithm runs non-stationary chains.

3.4. Adversarial interpretation

Our model is an energy-based model

$$p(\mathbf{I}; w) = \frac{1}{Z(w)} \exp[-\mathcal{E}(\mathbf{I}; w)]. \tag{9}$$

The update of w is based on $L'(w)$ which can be approximated by

$$\frac{1}{\tilde{M}}\sum_{m=1}^{\tilde{M}} \frac{\partial}{\partial w}\mathcal{E}(\tilde{\mathbf{I}}_m; w) - \frac{1}{M}\sum_{m=1}^M \frac{\partial}{\partial w}\mathcal{E}(\mathbf{I}_m; w), \tag{10}$$

where $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ are the synthesized image sequences that are generated by the Langevin dynamics. At

Algorithm 1 Learning and sampling algorithm

Input:

(1) training image sequences $\{\mathbf{I}_m, m = 1, ..., M\}$
(2) number of synthesized image sequences $\tilde{M}$
(3) number of Langevin steps l
(4) number of learning iterations T

Output:

(1) estimated parameters w
(2) synthesized image sequences $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$

1: Let $t \leftarrow 0$, initialize $w^{(0)}$.
2: Initialize $\tilde{\mathbf{I}}_m$, for $m = 1, ..., \tilde{M}$.
3: **repeat**
4: For each m, run l steps of Langevin dynamics to update $\tilde{\mathbf{I}}_m$, i.e., starting from the current $\tilde{\mathbf{I}}_m$, each step follows equation (7).
5: Calculate $H^{\text{obs}} = \sum_{m=1}^{M} \frac{\partial}{\partial w} f(\mathbf{I}_m; w^{(t)})/M$, and $H^{\text{syn}} = \sum_{m=1}^{\tilde{M}} \frac{\partial}{\partial w} f(\tilde{\mathbf{I}}_m; w^{(t)})/\tilde{M}$.
6: Update $w^{(t+1)} \leftarrow w^{(t)} + \eta_t (H^{\text{obs}} - H^{\text{syn}})$, with step size η_t.
7: Let $t \leftarrow t + 1$
8: **until** $t = T$

the zero temperature limit, the Langevin dynamics becomes gradient descent:

$$\tilde{\mathbf{I}}_{\tau+1} = \tilde{\mathbf{I}}_\tau - \frac{\epsilon^2}{2} \frac{\partial}{\partial \tilde{\mathbf{I}}} \mathcal{E}(\tilde{\mathbf{I}}_\tau; w). \tag{11}$$

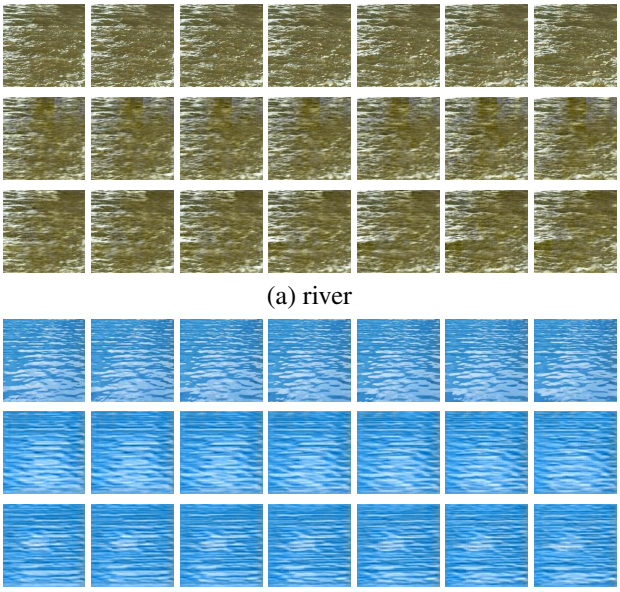
Consider the value function $V(\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}; w)$:

$$\frac{1}{\tilde{M}} \sum_{m=1}^{\tilde{M}} \mathcal{E}(\tilde{\mathbf{I}}_m; w) - \frac{1}{M} \sum_{m=1}^{M} \mathcal{E}(\mathbf{I}_m; w). \tag{12}$$

The updating of w is to increase V by shifting the low energy regions from the synthesized image sequences $\{\tilde{\mathbf{I}}_m\}$ to the observed image sequences $\{\mathbf{I}_m\}$, whereas the updating of $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$ is to decrease V by moving the synthesized image sequences towards the low energy regions. This is an adversarial interpretation of the learning and sampling algorithm. It can also be considered a generalization of the herding method [26] from exponential family models to general energy-based models.

In our work, we let $-\mathcal{E}(\mathbf{I}; w) = f(\mathbf{I}; w) - \|\mathbf{I}\|^2/2\sigma^2$. We can also let $-\mathcal{E}(\mathbf{I}; w) = f(\mathbf{I}; w)$ by assuming a uniform reference distribution $q(\mathbf{I})$. Our experiments show that the model with the uniform q can also synthesize realistic dynamic patterns.

The generative adversarial learning [6, 23] has a generator network. Unlike our model which is based on a bottom-up ConvNet $f(\mathbf{I}; w)$, the generator network generates $\mathbf{I}$ by a top-down ConvNet $\mathbf{I} = g(X; \tilde{w})$ where X is a latent vector that follows a known prior distribution, and $\tilde{w}$ collects the parameters of the top-down ConvNet. Recently [8] developed an alternating back-propagation algorithm to train the generator network, without involving an extra network. More recently, [28] developed a cooperative training method that recruits a generator network $g(X; \tilde{w})$ to reconstruct and regenerate the synthesized image sequences $\{\tilde{\mathbf{I}}_m\}$ to speed up MCMC sampling.



(a) river

(b) ocean

Figure 1. Synthesizing dynamic textures with both spatial and temporal stationarity. For each category, the first row displays the frames of the observed sequence, and the second and third rows display the corresponding frames of two synthesized sequences generated by the learning algorithm. (a) river. (b) ocean.

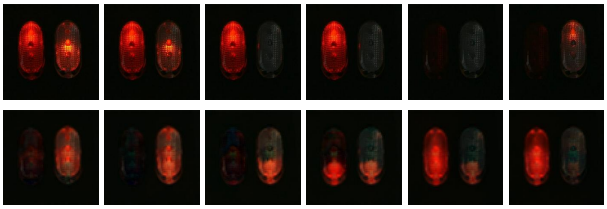
4. Experiments

We learn the spatial-temporal generative ConvNet from video clips collected from DynTex++ dataset of [4] and the Internet. The code in the experiments is based on the MatConvNet of [22] and MexConv3D of [21].

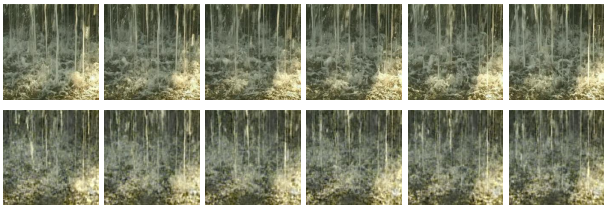
We show the synthesis results by displaying the frames in the video sequences. We have posted the synthesis results on the project page http://www.stat.ucla.edu/~jxie/STGConvNet/STGConvNet.html, so that the reader can watch the videos.

4.1. Experiment 1: Generating dynamic textures with both spatial and temporal stationarity

We first learn the model from dynamic textures that are stationary in both spatial and temporal domains. We use spatial-temporal filters that are convolutional in both spatial and temporal domains. The first layer has 120 $15 \times 15 \times 15$ filters with sub-sampling size of 7 pixels and frames. The

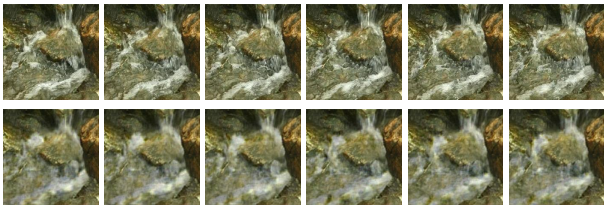


(a) flashing lights



(b) fountain

(c) burning fire heating a pot



(d) spring water

Figure 2. Synthesizing dynamic textures with only temporal stationarity. For each category, the first row displays the frames of the observed sequence, and the second row displays the corresponding frames of a synthesized sequence generated by the learning algorithm. (a) flashing lights. (b) fountain. (c) burning fire heating a pot. (d) spring water.

second layer has 40 $7 \times 7 \times 7$ filters with sub-sampling size of 3. The third layer has 20 $3 \times 3 \times 2$ filters with sub-sampling size of $2 \times 2 \times 1$. Figure 1 displays 2 results. For each category, the first row displays 7 frames of the observed sequence, while the second and third rows show the corresponding frames of two synthesized sequences generated by the learning algorithm.

We use the layer-by-layer learning scheme. Starting from the first layer, we sequentially add the layers one by one. Each time we learn the model and generate the synthesized image sequence using Algorithm 1. While learning the new layer of filters, we refine the lower layers of filters with back-propagation.

We learn a spatial-temporal generative ConvNet for each

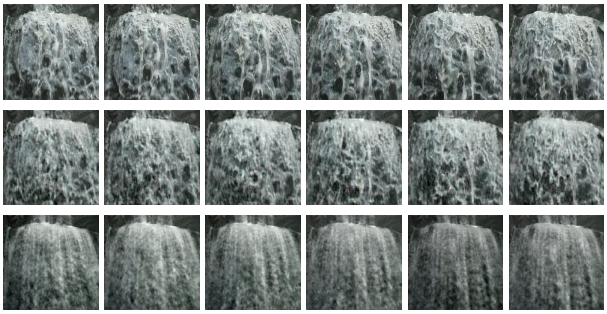


Figure 3. Comparison on synthesizing dynamic texture of waterfall. From top to bottom: segments of the observed sequence, synthesized sequence by our method, and synthesized sequence by the method of [2].

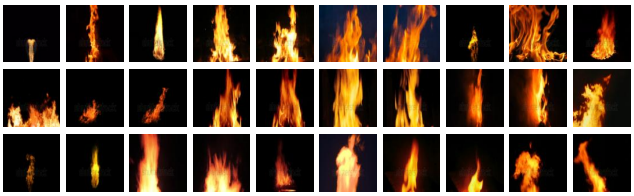
category from one observed video that is prepared to be of the size $224 \times 224 \times 50$ or 70. The range of intensities is $[0, 255]$. Mean subtraction is used as pre-processing. We use $\tilde{M} = 3$ chain for Langevin sampling. The number of Langevin iterations between every two consecutive updates of parameters, $l = 20$. The number of learning iterations $T = 1200$, where we add one more layer every 400 iterations. We use layer-specific learning rates, where the learning rate at the higher layer is less than that at the lower layer, in order to obtain stable convergence.

4.2. Experiment 2: Generating dynamic textures with only temporal stationarity

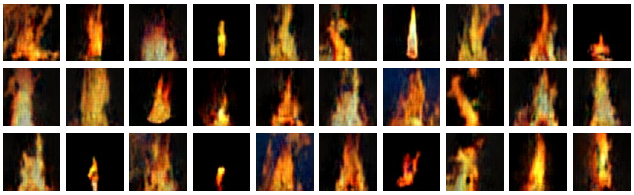
Many dynamic textures have structured background and objects that are not stationary in the spatial domain. In this case, the network used in Experiment 1 may fail. However, we can modify the network in Experiment 1 by using filters that are fully connected in the spatial domain at the second layer. Specifically, the first layer has 120 $7 \times 7 \times 7$ filters with sub-sampling size of 3 pixels and frames. The second layer is a spatially fully connected layer, which contains 30 filters that are fully connected in the spatial domain but convolutional in the temporal domain. The temporal size of the filters is 4 frames with sub-sampling size of 2 frames in the temporal dimension. Due to the spatial full connectivity at the second layer, the spatial domain of the feature maps at the third layer is reduced to 1×1. The third layer has 5 $1 \times 1 \times 2$ filters with sub-sampling size of 1 in the temporal dimension.

We use end-to-end learning scheme to learn the above 3-layer spatial-temporal generative ConvNet for dynamic textures. At each iteration, the 3 layers of filters are updated with 3 different layer-specific learning rates. The learning rate at the higher layer is much less than that at the lower layer to avoid the issue of large gradients.

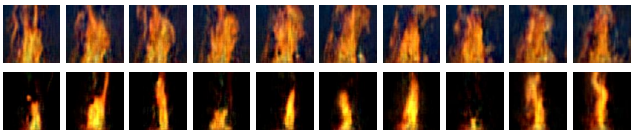
We learn a spatial-temporal generative ConvNet for each category from one training video. We synthesize $\tilde{M} = 3$



(a) 21-st frame of 30 observed sequences



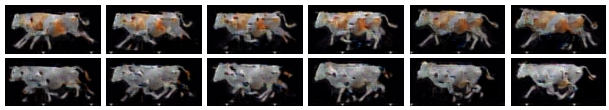
(b) 21-st frame of 30 synthesized sequences



(c) 2 examples of synthesized sequences

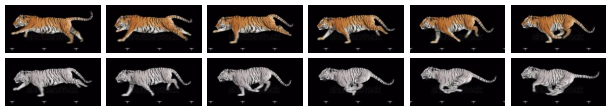
Figure 4. Learning from 30 observed fire videos with mini-batch implementation.

observed sequences

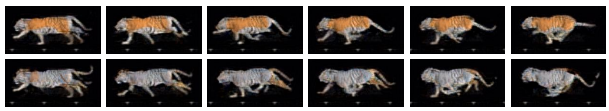


synthesized sequences

(a) running cows



observed sequences



synthesized sequences

(b) running tigers

Figure 5. Synthesizing action patterns. For each action video sequence, 6 continuous frames are shown. (a) running cows. Frames of 2 of 5 training sequences are displayed. The corresponding frames of 2 of 8 synthesized sequences generated by the learning algorithm are displayed. (b) running tigers. Frames of 2 observed training sequences are displayed. The corresponding frames of 2 of 4 synthesized sequences are displayed.

videos using the Langevin dynamics. Figure 2 displays the results. For each category, the first row shows 6 frames of the observed sequence ($224 \times 224 \times 70$), and the second row shows the corresponding frames of a synthesized sequence generated by the learning algorithm. We use the same set of parameters for all the categories without tuning. Figure 3 compares our method to that of [2], which is a linear dynamic system model. The image sequence generated by this model appears more blurred than the sequence generated by our method.

The learning of our model can be scaled up. We learn the fire pattern from 30 training videos, with mini-batch implementation. The size of each mini-batch is 10 videos. Each video contains 30 frames (100×100 pixels). For each mini-batch, $\tilde{M} = 13$ parallel chains for Langevin sampling is used. For this experiment, we slightly modify the network by using 120 $11 \times 11 \times 9$ filters with sub-sampling size of 5 pixels and 4 frames at the first layer, and 30 spatially fully connected filters with temporal size of 5 frames and sub-sampling size of 2 at the second layer, while keeping the setting of the third layer unchanged. The number of learning iterations $T = 1300$. Figure 4 shows one frame for each of 30 observed sequences and the corresponding frame of the synthesized sequences. Two examples of synthesized sequences are also displayed.

4.3. Experiment 3: Generating action patterns without spatial or temporal stationarity

Experiments 1 and 2 show that the generative spatial-temporal ConvNet can learn from sequences without alignment. We can also specialize it to learning roughly aligned video sequences of action patterns, which are non-stationary in either spatial or temporal domain, by using a single top-layer filter that covers the whole video sequence. We learn a 2-layer spatial-temporal generative ConvNet from video sequences of aligned actions. The first layer has 200 $7 \times 7 \times 7$ filters with sub-sampling size of 3 pixels and frames. The second layer is a fully connected layer with a single filter that covers the whole sequence. The observed sequences are of the size $100 \times 200 \times 70$.

Figure 5 displays two results of modeling and synthesizing actions from roughly aligned video sequences. We learn a model for each category, where the number of training sequences is 5 for the running cow example, and 2 for the running tiger example. The videos are collected from the Internet and each has 70 frames. For each example, Figure 5 displays segments of 2 observed sequences, and segments of 2 synthesized action sequences generated by the learning algorithm. We run $\tilde{M} = 8$ paralleled chains for the experiment of running cows, and 4 paralleled chains for the experiment of running tigers. The experiments show that our model can capture non-stationary action patterns.

One limitation of our model is that it does not involve explicit tracking of the objects and their parts.

4.4. Experiment 4: Learning from incomplete data

Our model can learn from video sequences with occluded pixels. The task is inspired by the fact that most of the videos contain occluded objects. Our learning method can be adapted to this task with minimal modification. The modification involves, for each iteration, running k steps of Langevin dynamics to recover the occluded regions of the observed sequences. At each iteration, we use the completed observed sequences and the synthesized sequences to compute the gradient of the log-likelihood and update the model parameters. Our method simultaneously accomplishes the following tasks: (1) recover the occluded pixels of the training video sequences, (2) synthesize new video sequences from the learned model, (3) learn the model by updating the model parameters using the recovered sequences and the synthesized sequences. See Algorithm 2 for the description of the learning, sampling, and recovery algorithm.

Table 1. Recovery errors in occlusion experiments

(a) salt and pepper masks

	ours	MRF-ℓ_1	MRF-ℓ_2
flag	**3.7923**	6.6211	10.9216
fountain	**5.5403**	8.1904	11.3850
ocean	**3.3739**	7.2983	9.6020
playing	**5.9035**	14.3665	15.7735
sea world	**5.3720**	10.6127	11.7803
traffic	**7.2029**	14.7512	17.6790
windmill	**5.9484**	8.9095	12.6487
Avg.	**5.3048**	10.1071	12.8272

(b) single region masks

	ours	MRF-ℓ_1	MRF-ℓ_2
flag	**8.1636**	10.6586	12.5300
fountain	**6.0323**	11.8299	12.1696
ocean	**3.4842**	8.7498	9.8078
playing	**6.1575**	15.6296	15.7085
sea world	**5.8850**	12.0297	12.2868
traffic	**6.8306**	15.3660	16.5787
windmill	**7.8858**	11.7355	13.2036
Avg.	**6.3484**	12.2856	13.1836

(c) 50% missing frames

	ours	MRF-ℓ_1	MRF-ℓ_2
flag	**5.5992**	10.7171	12.6317
fountain	**8.0531**	19.4331	13.2251
ocean	**4.0428**	9.0838	9.8913
playing	**7.6103**	22.2827	17.5692
sea world	**5.4348**	13.5101	12.9305
traffic	**8.8245**	16.6965	17.1830
windmill	**7.5346**	13.3364	12.9911
Avg.	**6.7285**	15.0085	13.7746

We design 3 types of occlusions: (1) Type 1: salt and pepper occlusion, where we randomly place 7×7 masks on the 150×150 image domain to cover 50% of the pixels of the videos. (2) Type 2: single region mask occlusion, where we randomly place a 60×60 mask on the 150×150 image domain. (3) Type 3: missing frames, where we randomly block 50% of the image frames from each video. Figure 6 displays one example of the recovery result for each type of occlusion. Each video has 70 frames.

To quantitatively evaluate the qualities of the recovered videos, we test our method on 7 video sequences, which are collected from DynTex++ dataset of [4], with 3 types of occlusions. We use the same model structure as the one used in Experiment 3. The number of Langevin steps for recovering is set to be equal to the number of Langevin steps for synthesizing, which is 20. For each experiment, we report the recovery errors measured by the average per pixel difference between the original image sequence and the recovered image sequence on the occluded pixels. The range of pixel intensities is $[0, 255]$. We compare our results

Algorithm 2 Learning, sampling, and recovery algorithm

Input:

(1) training image sequences with occluded pixels $\{\mathbf{I}_m, m = 1, ..., M\}$

(2) binary masks $\{O_m, m = 1, ..., M\}$ indicating the locations of the occluded pixels in the training image sequences

(3) number of synthesized image sequences $\tilde{M}$

(4) number of Langevin steps l for synthesizing image sequences

(5) number of Langevin steps k for recovering the occluded pixels

(6) number of learning iterations T

Output:

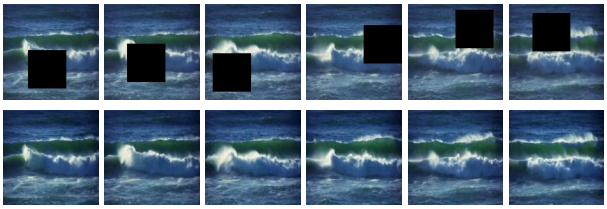
(1) estimated parameters w

(2) synthesized image sequences $\{\tilde{\mathbf{I}}_m, m = 1, ..., \tilde{M}\}$

(3) recovered image sequences $\{\mathbf{I}^{'}_m, m = 1, ..., M\}$

1: Let $t \leftarrow 0$, initialize $w^{(0)}$.
2: Initialize $\tilde{\mathbf{I}}_m$, for $m = 1, ..., \tilde{M}$.
3: Initialize $\mathbf{I}^{'}_m$, for $m = 1, ..., M$.
4: **repeat**
5: For each m, run k steps of Langevin dynamics to recover the occluded region of $\mathbf{I}^{'}_m$, i.e., starting from the current $\mathbf{I}^{'}_m$, each step follows equation (7), but only the occluded pixels in $\mathbf{I}^{'}_m$ are updated in each step.
6: For each m, run l steps of Langevin dynamics to update $\tilde{\mathbf{I}}_m$, i.e., starting from the current $\tilde{\mathbf{I}}_m$, each step follows equation (7).
7: Calculate $H^{\text{obs}} = \sum_{m=1}^{M} \frac{\partial}{\partial w} f(\mathbf{I}^{'}_m; w^{(t)})/M$, and $H^{\text{syn}} = \sum_{m=1}^{\tilde{M}} \frac{\partial}{\partial w} f(\tilde{\mathbf{I}}_m; w^{(t)})/\tilde{M}$.
8: Update $w^{(t+1)} \leftarrow w^{(t)} + \eta(H^{\text{obs}} - H^{\text{syn}})$, with step size η.
9: Let $t \leftarrow t + 1$
10: **until** $t = T$

(a) 50% salt and pepper masks



(b) single region masks

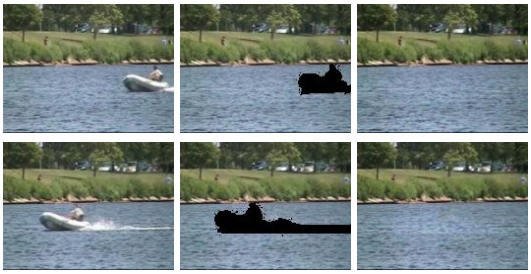
(c) 50% missing frames

Figure 6. Learning from occluded video sequences. For each experiment, the first row shows a segment of the occluded sequence with black masks. The second row shows the corresponding segment of the recovered sequence. (a) type 1: salt and pepper mask. (b) type 2: single region mask. (c) type 3: missing frames.

with the results obtained by a generic Markov random field model defined on the video sequence. The model is a 3D (spatial-temporal) Markov random field, whose potentials are pairwise ℓ_1 or ℓ_2 differences between nearest neighbor pixels, where the nearest neighbors are defined in both the spatial and temporal domains. The image sequences are recovered by sampling the intensities of the occluded pixels conditional on the observed pixels using the Gibbs sampler. Table 1 shows the comparison results for 3 types of occlusions. We can see that our model can recover the incomplete data, while learning from them.

4.5. Experiment 5: Background inpainting

If a moving object in the video is occluded in each frame, it turns out that the recovery algorithm will become an algorithm for background inpainting of videos, where the goal is to remove the undesired moving object from the video. We use the same model as the one in Experiment 2 for Figure 2. Figure 7 shows two examples of removals of (a) a moving boat and (b) a walking person respectively. The videos are collected from [19]. For each example, the first column displays 2 frames of the original video. The second column shows the corresponding frames with masks occluding the target to be removed. The third column presents the inpainting result by our algorithm. The video size is $130 \times 174 \times 150$ in example (a) and $130 \times 230 \times 104$ in example (b). The experiment is different from the video inpainting by interpolation. We synthesize image patches to fill in the empty regions of the video by running Langevin dynamics. For both Experiments 4 and 5, we run a single Langevin chain for synthesis.



(a) removing a moving boat in the lake

(b) removing a walking person in front of fountain

Figure 7. Background inpainting for videos. For each experiment, the first column displays 2 frames of the original video. The second column shows the corresponding frames with black masks occluding the target to be removed. The third column shows the inpainting result by our algorithm. (a) moving boat. (b) walking person.

5. Conclusion

In this paper, we propose a spatial-temporal generative ConvNet model for synthesizing dynamic patterns, such as dynamic textures and action patterns. Our experiments show that the model can synthesize realistic dynamic patterns. Moreover, it is possible to learn the model from video sequences with occluded pixels or missing frames.

Other experiments, not included in this paper, show that our method can also generate sound patterns.

The MCMC sampling of the model can be sped up by learning and sampling the models at multiple scales, or by recruiting the generator network to reconstruct and regenerate the synthesized examples as in cooperative training [28].

Acknowledgments

The work is supported by NSF DMS 1310391, DARPA SIMPLEX N66001-15-C-4035, ONR MURI N00014-16-1-2007, and DARPA ARO W911NF-16-1-0579.

References

[1] E. L. Denton, S. Chintala, R. Fergus, et al. Deep generative image models using a laplacian pyramid of adversarial networks. In *NIPS*, pages 1486–1494, 2015. 1

[2] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003. 1, 2, 5, 6

[3] A. Dosovitskiy, J. Tobias Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *CVPR*, pages 1538–1546, 2015. 1

[4] B. Ghanem and N. Ahuja. Maximum margin distance learning for dynamic texture recognition. In *ECCV*, pages 223–236. Springer, 2010. 4, 7

[5] M. Girolami and B. Calderhead. Riemann manifold langevin and hamiltonian monte carlo methods. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 73(2):123–214, 2011. 3

[6] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, pages 2672–2680, 2014. 2, 4

[7] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. DRAW: A recurrent neural network for image generation. In *ICML*, pages 1462–1471, 2015. 1

[8] T. Han, Y. Lu, S.-C. Zhu, and Y. N. Wu. Alternating back-propagation for generator network. In *AAAI*, 2017. 4

[9] Z. Han, Z. Xu, and S.-C. Zhu. Video primal sketch: A unified middle-level representation for video. *Journal of Mathematical Imaging and Vision*, 53(2):151–170, 2015. 2

[10] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2

[11] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):221–231, 2013. 2

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1

[13] T. D. Kulkarni, W. Whitney, P. Kohli, and J. B. Tenenbaum. Deep Convolutional Inverse Graphics Network. *ArXiv e-prints*, 2015. 1

[14] 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. 1

[15] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, and F. Huang. A tutorial on energy-based learning. *Predicting structured data*, 1:0, 2006. 1, 3

[16] Y. Lu, S.-C. Zhu, and Y. N. Wu. Learning FRAME models using cnn filters. In *AAAI*, 2016. 1

[17] G. F. Montufar, R. Pascanu, K. Cho, and Y. Bengio. On the number of linear regions of deep neural networks. In *NIPS*, pages 2924–2932, 2014. 3

[18] R. M. Neal. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2, 2011. 3

[19] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, and P. Pérez. http://perso.telecom-paristech.fr/~gousseau/video_inpainting. 8

[20] J. Ngiam, Z. Chen, P. W. Koh, and A. Y. Ng. Learning deep energy models. In *ICML*, pages 1105–1112, 2011. 1, 3

[21] P. Sun. https://github.com/pengsun/MexConv3D. 4

[22] A. Vedaldi and K. Lenc. Matconvnet – convolutional neural networks for matlab. *CoRR*, abs/1412.4564, 2014. 4

[23] C. Vondrick, H. Pirsiavash, and A. Torralba. Generating videos with scene dynamics. In *NIPS*, pages 613–621, 2016. 2, 4

[24] Y. Wang and S.-C. Zhu. A generative method for textured motion: Analysis and synthesis. In *ECCV*, pages 583–598. Springer, 2002. 1, 2

[25] Y. Wang and S.-C. Zhu. Analysis and synthesis of textured motion: Particles and waves. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(10):1348–1363, 2004. 2

[26] M. Welling. Herding dynamical weights to learn. In *ICML*, pages 1121–1128. ACM, 2009. 4

[27] R. J. Williams and D. Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989. 2

[28] J. Xie, Y. Lu, R. Gao, S.-C. Zhu, and Y. N. Wu. Cooperative training of descriptor and generator networks. *arXiv preprint arXiv:1609.09408*, 2016. 4, 8

[29] J. Xie, Y. Lu, S.-C. Zhu, and Y. N. Wu. A theory of generative convnet. In *ICML*, 2016. 1, 2

[30] X. You, W. Guo, S. Yu, K. Li, J. C. Príncipe, and D. Tao. Kernel learning for dynamic texture synthesis. *IEEE Transactions on Image Processing*, 25(10):4782–4795, 2016. 2

[31] L. Younes. On the convergence of markovian stochastic algorithms with rapidly decreasing ergodicity rates. *Stochastics: An International Journal of Probability and Stochastic Processes*, 65(3-4):177–228, 1999. 1, 3

[32] M. D. Zeiler, G. W. Taylor, and R. Fergus. Adaptive deconvolutional networks for mid and high level feature learning. In *ICCV*, pages 2018–2025, 2011. 3