Learning Dynamic Generator Model by Alternating Back-Propagation Through Time

Jianwen Xie[1,*], Ruiqi Gao[2,*], Zilong Zheng[2], Song-Chun Zhu[2], Ying Nian Wu[2] (*equal contribution)

[1] Hikvision Research Institute, [2] University of California, Los Angeles

Abstract

This paper studies the dynamic generator model for spatial-temporal processes such as dynamic textures and action sequences in video data. In this model, each time frame of the video sequence is generated by a generator model, which is a non-linear transformation of a latent state vector, where the non-linear transformation is parametrized by a top-down neural network. The sequence of latent state vectors follows a non-linear auto-regressive model, where the state vector of the next frame is a non-linear transformation of the state vector of the current frame as well as an independent noise vector that provides randomness in the transition. The non-linear transformation of this transition model can be parametrized by a feedforward neural network. We show that this model can be learned by an alternating back-propagation through time algorithm that iteratively samples the noise vectors and updates the parameters in the transition model and the generator model. We show that our training method can learn realistic models for dynamic textures and action patterns.

Dynamic generator model

Let $X = (x_t, t = 1, ..., T)$ be the observed video sequence, where x_t is a frame at time t. The dynamic generator model consists of the following two components:

$$s_t = F_\alpha(s_{t-1}, \xi_t), \tag{1}$$
$$x_t = G_\beta(s_t) + \epsilon_t, \tag{2}$$

where $t = 1, ..., T$. (1) is the transition model, and (2) is the emission model. s_t is the hidden state vector. $\xi_t \sim \mathrm{N}(0, I)$ is the noise vector. The Gaussian noise vectors $(\xi_t, t = 1, ..., T)$ are independent of each other. The sequence of $(s_t, t = 1, ..., T)$ follows a non-linear auto-regressive model, where the noise vector ξ_t encodes the randomness in the transition from s_{t-1} to s_t in the state space. F_α is a feedforward neural network or multi-layer perceptron, where α denotes the weight and bias parameters of the network. G_β is a top-down convolutional network (sometimes also called deconvolution network), where β denotes the weight and bias parameters of this top-down network. $\epsilon_t \sim \mathrm{N}(0, \sigma^2 I_D)$ is the residual error. We let $\theta = (\alpha, \beta)$ denote all the model parameters.

Alternative back-propagation Through Time

We estimate the model parameter θ by the maximum likelihood method that maximizes the observed-data log-likelihood $\log p_\theta(X)$. The gradient of the log-likelihood $\log p_\theta(X)$ is:

$$\begin{aligned}\frac{\partial}{\partial \theta} \log p_\theta(X) &= \frac{1}{p_\theta(X)} \frac{\partial}{\partial \theta} p_\theta(X) \\ &= \frac{1}{p_\theta(X)} \int \left[\frac{\partial}{\partial \theta} \log p_\theta(\xi, X)\right] p_\theta(\xi, X) d\xi \\ &= \mathrm{E}_{p_\theta(\xi|X)} \left[\frac{\partial}{\partial \theta} \log p_\theta(\xi, X)\right], \end{aligned} \tag{3}$$

where $p_\theta(\xi|X) = p_\theta(\xi, X)/p_\theta(X)$ is the posterior distribution of the latent ξ given the observed X. The above expectation can be approximated by Monte Carlo average.
The learning algorithm iterates the following two steps:
(1) **Inference step**: given the current θ, sample ξ from $p_\theta(\xi^{(\tau)}|X)$ by the Langevin dynamics

$$\xi^{(\tau+1)} = \xi^{(\tau)} + \frac{\delta^2}{2} \frac{\partial}{\partial \xi} \log p_\theta(\xi^{(\tau)}|X) + \delta z_\tau, \tag{4}$$

(2) **Learning step**: given ξ, update θ by stochastic gradient descent

$$\Delta\theta \propto \frac{\partial}{\partial \theta} \log p_\theta(\xi, X), \tag{5}$$

Since $\frac{\partial}{\partial \xi} \log p_\theta(\xi|X) = \frac{\partial}{\partial \xi} \log p_\theta(\xi, X)$, both steps involves derivatives of

$$\log p_\theta(\xi, X) = -\frac{1}{2}\left[\|\xi\|^2 + \frac{1}{\sigma^2}\|X - H_\theta(\xi)\|^2\right] + \text{const},$$

where the constant term does not depend on ξ or θ. Both can be computed by back-propagation through time.

Learning from multiple sequences

We can learn the model from multiple sequences of different appearances but of similar motion patterns. We can use an appearance (or content) vector $a^{(i)}$ for each sequence to account for the variation in appearance. The model is of the following form

$$s_t^{(i)} = F_\alpha(s_{t-1}^{(i)}, \xi_t^{(i)}), \tag{6}$$
$$x_t^{(i)} = G_\beta(s_t^{(i)}, a^{(i)}) + \epsilon_t^{(i)}, \tag{7}$$

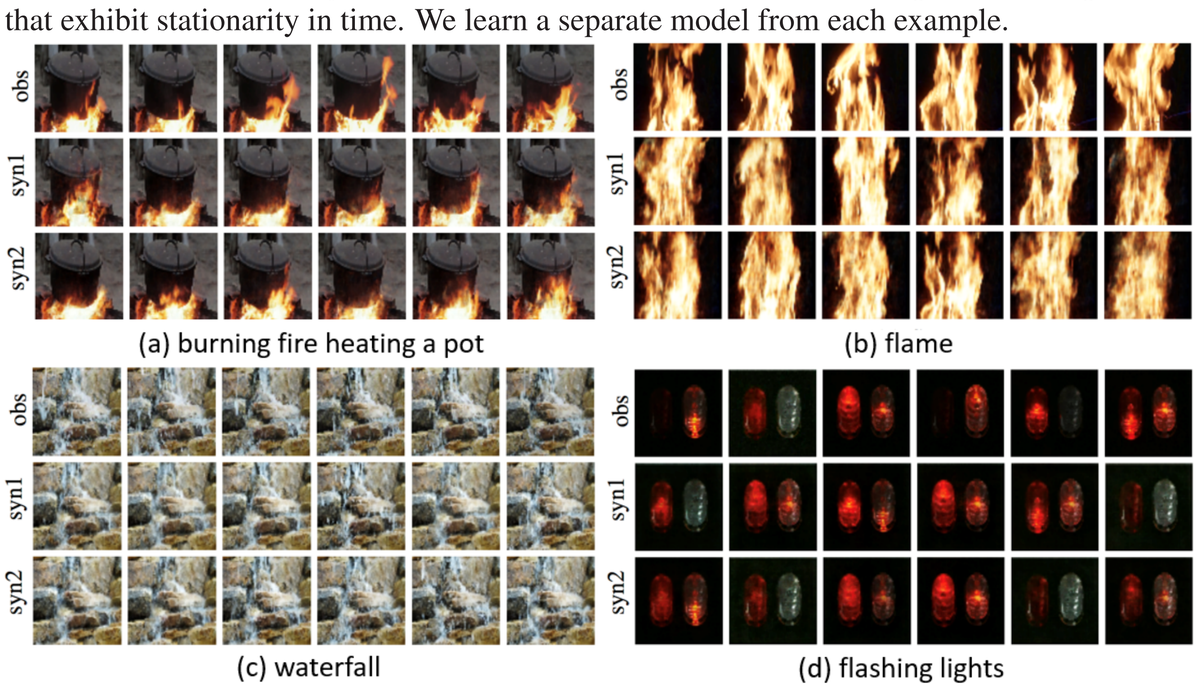
where $X^{(i)} = (x_t^{(i)}, t = 1, ..., T)$ be the i-th training sequence, $i = 1, ..., n$, $a^{(i)} \sim \mathrm{N}(0, I)$, and $a^{(i)}$ is fixed over time for each sequence i.

Conclusion

This paper studies a dynamic generator model for spatial-temporal processes. The model is a non-linear generalization of the linear state space model where the non-linear transformations in the transition and emission models are parameterized by neural networks. The model can be conveniently and efficiently learned by an alternating back-propagation through time (ABPTT) algorithm that alternatively samples from the posterior distribution of the latent noise vectors and then updates the model parameters.

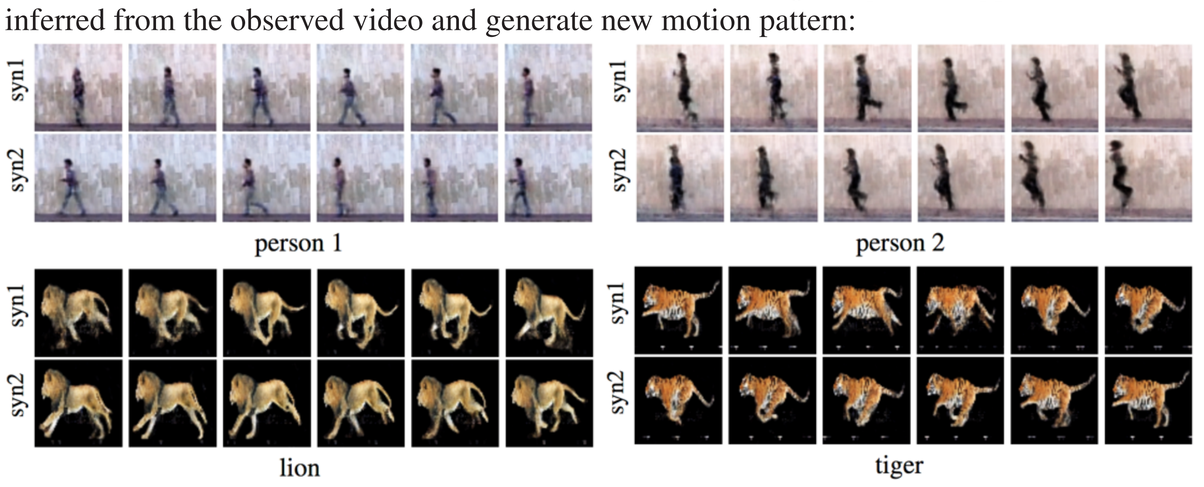
Experiment 1: Learning to generate dynamic texture

We learn the model for dynamic textures, which are sequences of images of moving scenes that exhibit stationarity in time. We learn a separate model from each example.



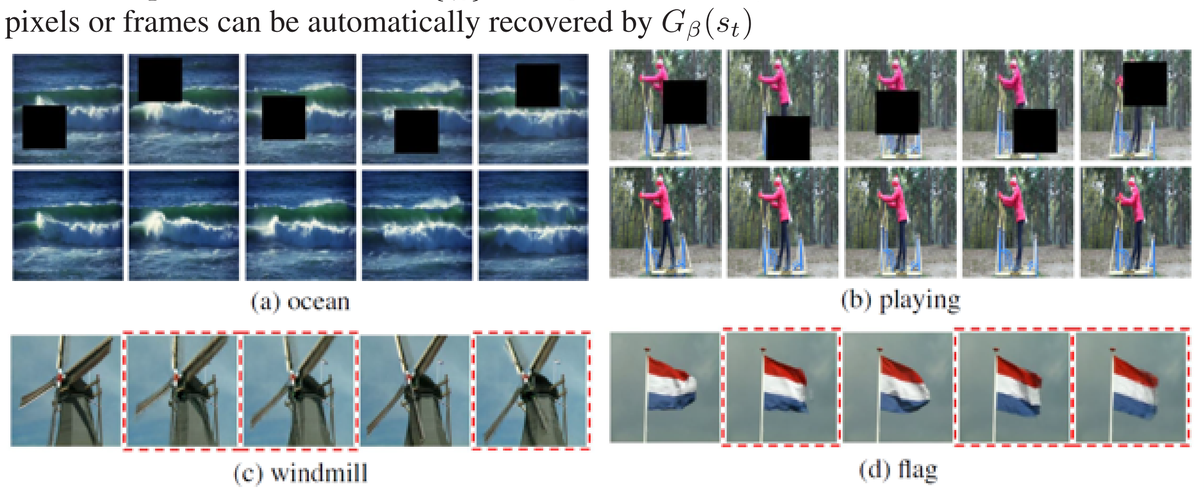
Experiment 2: Learning to generate action sequences

We learn the model from multiple examples with different appearances. A single model is trained on the whole dataset without annotations. After learning, we pick an appearance vector inferred from the observed video and generate new motion pattern:



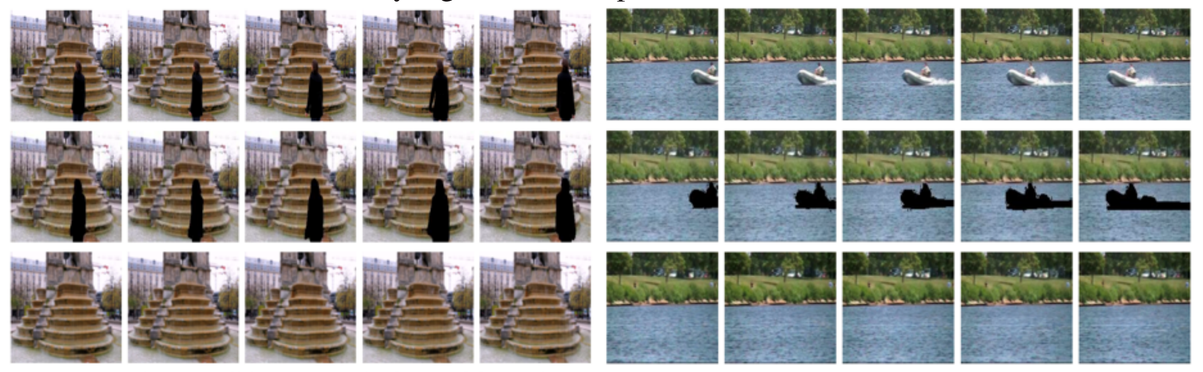
Experiment 3: Learning from incomplete data

We can learn from incomplete data. We compute $\sum_{t=1}^{T} \|x_t - G_\beta(s_t)\|^2$ by summing over only the visible pixels. With inferred $\{\xi_t\}$ and s_0, and learned β and α, the video with occluded pixels or frames can be automatically recovered by $G_\beta(s_t)$



Experiment 4: learning to inpaint

The model can be used to remove undesirable content in the video for background inpainting. We manually mask the undesirable moving object in each frame, and learn the model from the masked video with the recovery algorithm in Exp 3.



Experiment 5: learning to predict from static image

Image to video prediction. For each example, the first image is the static image frame, and the rest are 6 frames of the predicted sequence.