LINEARIZING VISUAL PROCESSES WITH DEEP GENERATIVE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

This work studies the problem of modeling non-linear visual processes by leveraging deep generative architectures for learning linear, Gaussian models of observed sequences. We propose a joint learning framework, combining a *multivariate autoregressive model* and *deep convolutional generative networks*. After justification of theoretical assumptions of linearization, we propose an architecture that allows Variational Autoencoders and Generative Adversarial Networks to simultaneously learn the non-linear observation as well as the linear state-transition model from a sequence of observed frames. Finally, we demonstrate our approach on conceptual toy examples and dynamic textures.

1 INTRODUCTION

While classification of image and video with Convolutional Neural Networks (CNN) is becoming an established practice, unsupervised learning and generative modeling remain to be challenging problems in deep learning. A generative model of a visual process enables the possibility of generating sequences of video frames such that the appearance as well as the dynamics approximately resemble the original training process without copying it. This procedure is typically referred to as video generation (Vondrick & Torralba (2017); De Souza et al. (2017)) or video synthesis (Liu et al. (2017b)). More technically, this means that in addition to a suitable probability model for the individual frames, a probabilistic description for the frame-to-frame transition is also necessary. Analysis and reproduction of visual processes simplifies considerably, if this transition can be described as a *multivariate autoregressive* (MAR) model, i.e., as a combination of linear transformations and Gaussian noise. For instance, linear transformations are easily invertible and by means of spectral analysis, it can be studied how such a process behaves in the long term.

Realistically, most frame transitions in real-world visual processes unlikely are linear functions. Nevertheless, unsupervised learning has come up with many approaches to fit MAR models to real-world processes, for instance by using linear low-rank approximations, as proposed by Doretto et al. (2003), or sparse approximations of the frames, as proposed by Wei et al. (2017), or applying the kernel trick to them (Chan & Vasconcelos (2007)).

The success of Generative Adversarial Networks (GAN) introduced by Goodfellow et al. (2014) and Variational Autoencoders (VAE) introduced by Kingma & Welling (2013) has led to an increased interest in deep generative learning and it seems natural to apply such techniques to sequential processes. We approach this idea from the perspective of *linearization*, in order to keep the model as simple as possible. In an analogous way as physicists transforming non-linear differential equations into linear ones by means of an appropriate change of variables, our approach is to *learn* latent representations of visual processes, such that the latent state-to-state transition can be described by an MAR model. To this end, we *jointly* learn a non-linear observation and a linear state transition function by introducing a *dynamic layer* that can be used in conjunction with deep generative architectures such as GANs and VAEs.

NOTATION

In this paper, letters that appear both in roman and italicized type refer to random variables and realizations thereof, respectively. If not stated otherwise, expected values of more than one random variables assume statistical independence.

2 RELATED WORK

The deep predictive coding network (Chalasani & Principe (2013)), can be considered one of the earliest approaches of combining MAR models with deep learning. More recently, such a combination was studied by Liu et al. (2018). The work by Goroshin et al. (2015) deals with linearizing transformations under uncertainty via neural networks. It resembles our work in that we also focus on *representation learning* rather than on a particular type of process. However, these works do not employ VAEs or GANs.

By contrast, Watter et al. (2015) combine Linear Dynamic Systems (LDSs) with VAEs. However, the focus of their work is on control rather than on synthesis. Furthermore, their model is *locally* linear and the transition distribution is modeled in the variational bound, whereas we model it as a separate layer. This also is the main difference to the work of Krishnan et al. (2015), in which VAEs are used as Kalman Filters. The work of Johnson et al. (2016) combines VAEs with linear dynamic models for forecasting images in video sequences. Mathematically, it is a well-thought out approach, but proposes a training objective that is considerably more complex than the one proposed in this work. GANs in combination with MAR models have been studied in experiments of Han et al. (2017), but the MAR model was learned separately from the GAN.

Theoretical groundwork regarding learned visual transformations has been done by Cohen & Welling (2014; 2016); Hinton et al. (2011) and Memisevic (2013). In particular, the *dynamic layer* proposed in this work bears resemblance to techniques employed in image-to-image translation (Liu et al. (2017a); Zhu et al. (2017)).

The synthesis of video dynamics by means of neural networks has been discussed, among others by Vondrick et al. (2016); Xue et al. (2016); Srivastava et al. (2015) and Xie et al. (2017). We would like to emphasize the difference between video synthesis which is discussed in our work and the *prediction* of video frames as studied, for instance, by Mathieu et al. (2015). While the former refers to the problem of finding a *probabilistic* model for the spatial and temporal behavior of a visual process, the latter refers to finding a *deterministic* mapping from a set of previous frames to one or several future frames to come. As a consequence, a video synthesis model needs to take care of long-term behavior. Additionally, the probabilistic nature of video synthesis makes it considerably harder to evaluate the generated frames, since a unique ground truth cannot be provided, ruling out classical quantitative quality measures such as mean squared error or structured similarity.

Finally, the core contribution of this work is a combination of neural networks with Markov processes. This has been the subject of many works in the recent past. For a broad overview of results in this field, the reader is referred to Chapter 20 of the book *Deep Learning* by Goodfellow et al. (2016).

3 VISUAL PROCESSES

3.1 DYNAMIC SYSTEMS

The *Dynamic Texture* (DT) model by Doretto et al. (2003) has popularized LDSs in the modeling of visual processes. Typically, an LDS is of the following form

$$\begin{aligned} \boldsymbol{h}_{t+1} &= \boldsymbol{A}\boldsymbol{h}_t + \boldsymbol{v}_t, \\ \boldsymbol{y}_t &= \bar{\boldsymbol{y}} + \boldsymbol{C}\boldsymbol{h}_t, \end{aligned} \tag{1}$$

where $\boldsymbol{h}_t \in \mathbb{R}^n$ is the low dimensional *state space variable* at time t, $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ the *state transition matrix*, $\boldsymbol{y}_t \in \mathbb{R}^d$ the observation at time t and $\boldsymbol{C} \in \mathbb{R}^{d \times n}$ the *observation matrix*. The vector $\bar{\boldsymbol{y}} \in \mathbb{R}^d$ represents a constant offset in the observation space. The input term $\boldsymbol{v}_t$ is modeled as i.i.d. zero-mean Gaussian noise, and is independent of $\boldsymbol{h}_t$.

Real-world visual processes are often highly non-linear and non-Gaussian, hence are considerably harder to work with. In the following, we define a more general dynamic model that is applicable to a broad class of visual processes. To keep the problem tractable, we make the assumption of two properties that were already implicitly assumed in the classical DT model.

Assumption 1. *The visual processes of interest are stationary first-order Markov processes.*

The stationarity property guarantees that synthesis of new sequences will not diverge and the Markov property facilitates the statistical inference. However, we believe that the method presented in this work can be generalized to Markov processes of higher orders. We refer to Appendix A.2 for details.

Consider the following generic dynamic equation of observations $\boldsymbol{y}$ in an n-dimensional manifold $\mathbb{M} \subset \mathbb{R}^d$.

$$\boldsymbol{y}_{t+1} = \psi_{\boldsymbol{v}_t}(\varphi(\boldsymbol{y}_t)). \tag{2}$$

Again, $\boldsymbol{y}_t \in \mathbb{R}^d$ denotes the observation at time t. The self map $\varphi : \mathbb{M} \to \mathbb{M}$ models the predicted frame transition and describes the deterministic part of the dynamics. In this work, it is assumed to be differentiable. Furthermore, the function $\psi_{\boldsymbol{v}_t} : \mathbb{M} \to \mathbb{M}$ describes a displacement by $\boldsymbol{v}_t$ in the in the tangent space $\mathcal{T}_{\boldsymbol{y}_t}\mathbb{M}$ of $\mathbb{M}$ at $\boldsymbol{y}_t$, followed by a retraction onto $\mathbb{M}$. The displacement is sampled from i.i.d. zero-mean Gaussian noise. Eq. (1) is a special case of Eq. (2), where $\varphi(\boldsymbol{y}) = \boldsymbol{C}\boldsymbol{A}\boldsymbol{C}^{+}(\boldsymbol{y} - \bar{\boldsymbol{y}}) + \bar{\boldsymbol{y}}$ and $\psi_{\boldsymbol{v}}(\boldsymbol{y}) = \boldsymbol{y} + \boldsymbol{C}\boldsymbol{v}$.

The model Eq. (2) describes a much broader class of visual processes than Eq. (1). However, unlike model Eq. (2), the state transition in the linear model Eq. (1) enables straightforward prediction, generation, and analysis of observations. It is thus of great interest to find a model that linearizes real-world visual processes, so that in some latent state space representation, the state transition admits the MAR model of the first line of Eq. (1). Specifically, this work focuses on the following non-linear dynamic system model, i.e., a linear state transition and a non-linear observation mapping

$$\begin{aligned} \boldsymbol{h}_{t+1} &= \boldsymbol{A}\boldsymbol{h}_t + \boldsymbol{v}_t, \\ \boldsymbol{y}_t &= C(\boldsymbol{h}_t), \end{aligned} \tag{3}$$

where $n < d$ and $C\colon \mathbb{R}^n \to \mathbb{M} \subset \mathbb{R}^d$ is differentiable. Transforming a non-linear model to this form makes video synthesis as easy as sampling from autoregressive noise.

It is worth noticing that the model in Eq. (3) is not unique with respect to changes of basis in the state space (Doretto et al. (2003); Afsari & Vidal (2013)). However, if C is implemented via a neural network, we can ensure that it accounts for a possible change of basis, e.g. by adding a linear layer to the input of the network. We can thus make the following assumption on the latent samples $\boldsymbol{h}_t$ without loss of generality.

Assumption 2. *The latent samples $\boldsymbol{h}_t$ abide a standard normal distribution, i.e., $\mathbf{h}_t \sim \mathcal{N}(\,\cdot\,;\, 0, \boldsymbol{I})$.*

Remark 1. *If the state transition matrix $\boldsymbol{A}$ is given, and the process is stationary, i.e., $p(\mathbf{h}_t) = p(\mathbf{h}_{t+1})$ for all t, then Assumption 2 essentially identifies the process noise model. Namely, we have*

$$\begin{aligned} \mathbb{E}_{\mathbf{h}_{t+1}}\left[\mathbf{h}_{t+1}\mathbf{h}_{t+1}^\top\right] &= \mathbb{E}_{\mathbf{h}_t,\mathbf{v}_t}\left[\left(\boldsymbol{A}\mathbf{h}_t + \mathbf{v}_t\right)\left(\boldsymbol{A}\mathbf{h}_t + \mathbf{v}_t\right)^\top\right] \\ &= \mathbb{E}_{\mathbf{h}_t,\mathbf{v}_t}\left[\boldsymbol{A}\mathbf{h}_t\mathbf{h}_t^\top\boldsymbol{A}^\top + \mathbf{v}_t\mathbf{h}_t^\top\boldsymbol{A}^\top + \boldsymbol{A}\mathbf{h}_t\mathbf{v}_t^\top + \mathbf{v}_t\mathbf{v}_t^\top\right] \\ &= \mathbb{E}_{\mathbf{h}_t,\mathbf{v}_t}\left[\boldsymbol{A}\mathbf{h}_t\mathbf{h}_t^\top\boldsymbol{A}^\top\right] + \mathbb{E}_{\mathbf{v}_t}\left[\mathbf{v}_t\mathbf{v}_t^\top\right] \\ &= \boldsymbol{A}\boldsymbol{A}^\top + \mathbb{E}_{\mathbf{v}_t}\left[\mathbf{v}_t\mathbf{v}_t^\top\right]. \end{aligned} \tag{4}$$

In order to make sure that the latent states $\boldsymbol{h}_t$ remain standard Gaussian in sequential synthesis scenarios, i.e., $\mathbb{E}_{\mathbf{h}_{t+1}}\left[\mathbf{h}_{t+1}\mathbf{h}_{t+1}^\top\right] = \boldsymbol{I}$, we just need to ensure that the process noise is zero-mean and has the covariance matrix $\boldsymbol{I} - \boldsymbol{A}\boldsymbol{A}^\top$. ◊

Before we propose an algorithm to jointly learn C and $\boldsymbol{A}$ by means of deep generative architecture, we further characterize the problem in order to investigate its feasibility and motivate our approach. For a model Eq. (3) to substitute Eq. (2), it needs to transform the transition φ to a multiplication by a matrix and the perturbation $\psi_{\boldsymbol{v}_t}$ to a superposition by zero-mean Gaussian noise. The latter is generally easy to achieve, if the perturbation is sufficiently small and C is a diffeomorphism. In the remainder of the section, we thus focus on linearization of φ.

3.2 Local Linearization

A common linearization method in control theory is to approximate the dynamic system equation by a first-order Taylor polynomial around an equilibrium point (Perko (2013)). Clearly, for this to be possible, such a point needs to exist. We thus propose the following assumption on the transition function φ of Eq. (2).

Assumption 3. *The differentiable self map $\varphi\colon \mathbb{M} \to \mathbb{M}$ has at least one fixed point $\boldsymbol{y}^*$.*

However, such an approach might fail for real-world processes due to the curse of dimensionality. A remedy is to employ an representation function Γ that maps the system observations to a lower-dimensional latent space prior to performing the linearization. We call such a function a *local linearizer*.

Definition 1. *Let $\mathbb{M} \subset \mathbb{R}^d$ be an n-dimensional manifold, $\varphi\colon \mathbb{R}^d \to \mathbb{R}^d$ be differentiable on $\varphi(\mathbb{M}) = \mathbb{M}$, and $\Gamma\colon \mathbb{R}^d \to \mathbb{R}^n$ be a diffeomorphism on $\mathbb{M}$. The map Γ is said to be a local linearizer at $\boldsymbol{y}^* \in \mathbb{M}$ of φ , if there exists a matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n\times n}$ such that the following equality holds true with $\boldsymbol{y} \in \mathbb{R}^d$.*

$$\lim_{\|\boldsymbol{y}-\boldsymbol{y}^*\|\to 0} \frac{\|\varphi(\boldsymbol{y}) - \Gamma^{-1}(\boldsymbol{\Phi}\Gamma(\boldsymbol{y}))\|}{\|\boldsymbol{y} - \boldsymbol{y}^*\|} = 0. \tag{5}$$

Generally speaking, local linearization is made possible by moving the fixed point to the origin of a new coordinate system. More precisely, the following proposition holds.

Proposition 1. *Let $\mathbb{M} \subset \mathbb{R}^d$ be an n-dimensional manifold. Furthermore, let $\varphi\colon \mathbb{R}^d \to \mathbb{R}^d$ be differentiable on $\varphi(\mathbb{M}) = \mathbb{M}$, and $\Gamma\colon \mathbb{R}^d \to \mathbb{R}^n$ be a diffeomorphism on $\mathbb{M}$. If $\boldsymbol{y}^* \in \mathbb{M}$ is a fixed point of φ, then the following map,*

$$\begin{aligned} \Gamma' : \mathbb{R}^n &\to \mathbb{M} \\ \boldsymbol{y} &\mapsto \Gamma(\boldsymbol{y}) - \Gamma(\boldsymbol{y}^*), \end{aligned} \tag{6}$$

locally linearizes φ, if the rows and columns of the Jacobi matrix $\boldsymbol{J}_\varphi$ of φ at $\boldsymbol{y}^*$ lie in the row space of the Jacobi matrix $\boldsymbol{J}_\Gamma$ of Γ at $\boldsymbol{y}^*$.

Proof. See Appendix A.1 □

The Jacobi matrix property is only needed for consistency with Definition 1, where the limit $\boldsymbol{y} \to \boldsymbol{y}^*$ is approached from any possible direction in $\mathbb{R}^d$. It can be ignored, if we restrict the analysis entirely to the manifold $\mathbb{M}$. Proposition 1 states that if φ is a differentiable self map on $\mathbb{M}$ with a fixed point $\boldsymbol{y}^* \in \mathbb{M}$, and a diffeomorphism from $\mathbb{M}$ to $\mathbb{R}^n$ exists, then there is a representation in which φ can be approximated by a linear function for points on $\mathbb{M}$ that are not too far away from $\boldsymbol{y}^*$.

Note that even if the Jacobi matrix $\boldsymbol{J}_\varphi$ can be sufficiently well estimated from the data, the linear approximation directly in the observation space $\mathbb{R}^d$ will likely be worse than via reparametrization of a local linearizer Γ' that functions as a chart of $\mathbb{M}$. The reason is that predictions directly via $\boldsymbol{J}_\varphi$ can not be assumed to remain on $\mathbb{M}$. Figuratively speaking, the aim of the local linearizer is to bend the space spanned by the rows and columns of $\boldsymbol{J}_\varphi$ to match the shape of $\mathbb{M}$.

3.3 Separate Learning

Local linearizability is a useful concept to analyze the feasibility of the problem at hand. However, it does not take into account that two local linearizers can have significantly different linearization properties on a global scale. Moreover, it does not provide any instructions on how to find an appropriate representation Γ and matrix $\boldsymbol{\Phi}$. Traditionally, this task is approached by learning Γ and $\boldsymbol{\Phi}$ separately while considering the sampled observations of the process globally, as will be described in the following.

To characterize the problem of global linearization, we need to introduce a measure. It is sensible to consider the expected squared distance between the result of a transformation φ and its linear approximation. Since it is not possible to have an analytic expression for the distribution on the data manifold $\mathbb{M}$, the latent space is often considered as a heuristic. Because it is a common model assumption for deep generative models and in accordance with Assumption 2, we can assume standard Gaussian distribution for the latent space.

Let $\Gamma : \mathbb{M} \to \mathbb{R}^n$ denote a data representation mapping and $\varphi : \mathbb{M} \to \mathbb{M}$ the transition function to be linearized. Consider the following expression

$$q(\Gamma, \varphi, \boldsymbol{\Phi}) = \frac{\mathbb{E}_{\mathbf{y}}\left[\|\boldsymbol{\Phi}\Gamma(\mathbf{y}) - \Gamma(\varphi(\mathbf{y}))\|^2\right]}{2n}. \tag{7}$$

The denominator $2n$ is a normalization factor. If $\Gamma(\mathbf{y})$ has standard Gaussian distribution, then $q(\Gamma, \varphi, \mathbf{\Phi}) < 1$ denotes that the linear prediction with $\mathbf{\Phi}$ is smaller than the expected distance of two independently drawn samples. A common method of linearizing φ, is to first choose a representation Γ that is assumed to have good linearization properties, and then inferring $\mathbf{\Phi}$ by minimizing $Eq.$ (7) (Doretto et al. (2003); Chan & Vasconcelos (2007); Han et al. (2017)). In that case, the solution is given by

$$\hat{\mathbf{\Phi}} = \mathbb{E}_{\mathbf{y}}[\Gamma(\varphi(\mathbf{y}))\Gamma(\mathbf{y})^\top], \tag{8}$$

if $\Gamma(\boldsymbol{y})$ has standard Gaussian distribution.

However, the drawback of this approach is the difficulty to find an appropriate model for Γ that can be assumed to linearize the transformation φ. Moreover, separate learning implicitly assumes that a small linearization error in the embedded space will carry over to a small error in the observation space, but such an assumption is hard to justify given the high dimension of the problem. We therefore propose to approach the problem by learning the representation and the linear transition jointly, by approximating the data distribution directly via deep generative models.

4 Joint Learning via Deep Generative Models

4.1 Markov Assumptions

We now turn to the problem of finding a model Eq. (3) that approximates Eq. (2). In the following, we will provide the preliminaries to lay out a training procedure to infer C and $\boldsymbol{A}$ from observed sequences of visual processes. A sequence of length N of a visual process $\mathcal{Y}$ is viewed as a realization of the random variable $\mathbf{y}^N = [\mathbf{y}_1, \ldots, \mathbf{y}_N]$. The according sequence in the latent space is a realization of the random variable $\mathbf{h}^N = [\mathbf{h}_1, \ldots, \mathbf{h}_N]$. Consider the random variable

$$\tilde{\mathbf{y}}^N = [\tilde{\mathbf{y}}_1, \ldots, \tilde{\mathbf{y}}_N] := [C(\mathbf{h}_1), \ldots, C(\mathbf{h}_N)]. \tag{9}$$

In order to model $\mathcal{Y}$ by Eq. (3), we need to make sure that the probability distributions of $\mathbf{y}^N$ and $\tilde{\mathbf{y}}^N$ coincide for any $N \in \mathbb{N}$. By taking into account Assumption 1, this is equivalent to demanding that the joint probability of two succeeding frames coincide.

The joint probability distribution of $\mathbf{h}_t$ and $\mathbf{h}_{t+1}$ is zero-mean Gaussian and, more specifically,

$$\begin{bmatrix} \mathbf{h}_t \\ \mathbf{h}_{t+1} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{bmatrix}; 0, \begin{bmatrix} \boldsymbol{I} & \boldsymbol{A}^\top \\ \boldsymbol{A} & \boldsymbol{I} \end{bmatrix} \right), \tag{10}$$

holds due to Assumption 2. To summarize, we are looking for the function C and a matrix $\boldsymbol{A}$, such that the random variable

$$\tilde{\mathbf{y}}^2 = [C(\mathbf{h}_1) \quad C(\mathbf{h}_2)], \tag{11}$$

has the same probability distribution as $\mathbf{y}^2$.

4.2 Deep Generative Models

Estimation of probability distributions for high-dimensional data is still an ongoing research topic in deep learning. The problem is typically framed as an approximation of a function $f_{\boldsymbol{\theta}}$, parameterized by a vector $\boldsymbol{\theta}$ in a finite-dimensional euclidean space. The purpose of $f_{\boldsymbol{\theta}}$ is to map realizations of low-dimensional, standard Gaussian noise to samples that abide the data distribution. Typically, the function $f_{\boldsymbol{\theta}}$ is realized by a neural network with trainable weights $\boldsymbol{\theta}$.

Most of the widely employed deep generative models, including the GAN, and the VAE, adopt this approach. What these models differ in, is merely the way the network parameters $\boldsymbol{\theta}$ are trained. For this work, we employ the Wasserstein GAN and the VAE to evaluate the proposed method. Due to space constraints, we will not review these techniques here and refer the reader to Arjovsky et al. (2017) for the Wasserstein GAN and to Doersch (2016) for the VAE. For the following sections, it is sufficient to assume that these techniques are capable of approximating a function $f_{\boldsymbol{\theta}}$ that maps from a low-dimensional standard Gaussian to a high-dimensional data distribution.

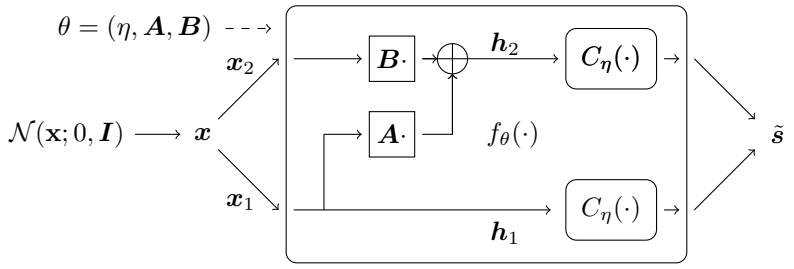


Figure 1: Generative network with dynamic layer. The dashed arrow indicates the trainable parameters

4.3 The Dynamic Layer

We now focus on how to learn a matrix $\boldsymbol{A}$ and a function C from a finite sequence $\boldsymbol{y}^N \in \mathbb{R}^{d \times N}$ of a visual process such that it can be described by Eq. (3). To this end, recall that, as discussed in Section 4.1, we only need to consider the joint probability of two succeeding observed frames. The first step is thus to generate a training set of frame pairs from $\boldsymbol{y}^N$ as $\{\boldsymbol{s}_1, \ldots, \boldsymbol{s}_{N-1}\}$, where

$$\boldsymbol{s}_t = \begin{bmatrix} \boldsymbol{y}_t^\top & \boldsymbol{y}_{t+1}^\top \end{bmatrix}^\top \tag{12}$$

denotes a vectorized pair of frames. We treat the samples as realizations of the random variable $\mathbf{s}$ and are looking for an architecture to learn the function C and a matrix $\boldsymbol{A}$ such that the probability distribution of the random variable

$$\begin{bmatrix} C(\mathbf{h}_1)^\top & C(\mathbf{h}_2)^\top \end{bmatrix}^\top, \tag{13}$$

where $\mathbf{h}_1, \mathbf{h}_2$ have the joint distribution described by Eq. (10), coincides with the probability distribution of $\mathbf{s}$, i.e.

$$p(\mathbf{s}) = p\left(\begin{bmatrix} C(\mathbf{h}_1)^\top & C(\mathbf{h}_2)^\top \end{bmatrix}^\top \right). \tag{14}$$

It turns out that we can use a deep generative architecture to accomplish this task. Remember, that a function f_θ learned by a deep generative model is capable of transforming samples $\boldsymbol{x}$ of standard Gaussian noise to samples from a high dimensional data distribution. Let $\theta = (\boldsymbol{A}, \boldsymbol{B}, \eta)$ contain the matrices $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{n \times n}$ such that the constraint

$$\boldsymbol{A}\boldsymbol{A}^\top + \boldsymbol{B}\boldsymbol{B}^\top = \boldsymbol{I}_n \tag{15}$$

is fulfilled, and the parameter η that defines the model for $C_\eta = C$. Consider the following definition for f_θ.

$$f_\theta : \mathbb{R}^{2n} \to \mathbb{R}^{2d}, \qquad \begin{bmatrix} \boldsymbol{x}_1 \\ \boldsymbol{x}_2 \end{bmatrix} \mapsto \begin{bmatrix} C_\eta\left(\boldsymbol{x}_1\right) \\ C_\eta\left(\boldsymbol{A}\boldsymbol{x}_1 + \boldsymbol{B}\boldsymbol{x}_2\right) \end{bmatrix}. \tag{16}$$

Assume that we can train f_θ, as defined in Eq. (16), to map $\mathbf{x} \sim \mathcal{N}(\mathbf{x}; 0, \boldsymbol{I}_{2n})$ to $\tilde{\mathbf{s}} = f_\theta(\mathbf{x})$ with the probability distribution

$$\tilde{\mathbf{s}} \sim p(\mathbf{s}). \tag{17}$$

Then, $\boldsymbol{A}$ and $C = C_\eta$ indeed fulfill the condition in Eq. (14) for $\mathbf{h}_1, \mathbf{h}_2$ with joint probability distribution Eq. (10). In order to implement f_θ by a neural network, we propose the architecture depicted in Fig. 1. The first layer is linear and will be referred to as the *dynamic layer* in the following. The dynamic layer implements the multiplication with the matrix

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_n & \\ \boldsymbol{A} & \boldsymbol{B} \end{bmatrix}. \tag{18}$$

The output of the dynamic layer is split into an upper half $\boldsymbol{h}_1$ and a lower half $\boldsymbol{h}_2$ and both halves are fed to the subnetwork that implements the observer function C_η. The weights of the dynamic layer contain the matrices $\boldsymbol{A}, \boldsymbol{B}$. Thus, they can be trained along with η, by back-propagation of the stochastic gradient. However, we need to make sure that the stationarity constraint in Eq. (15) is not violated. This can be done by adding a regularizer to the loss function L of the Deep Generative Network. We thus substitute $L(\theta)$ by

$$\tilde{L}(\theta) = L(\theta) + \lambda \| \boldsymbol{A}\boldsymbol{A}^\top + \boldsymbol{B}\boldsymbol{B}^\top - \boldsymbol{I}_n \|_F^2, \tag{19}$$

where $\lambda > 0$ is a regularizer constant. Here, L refers to the variational bound for the VAE or to the Generator loss for the (Wasserstein) GAN.

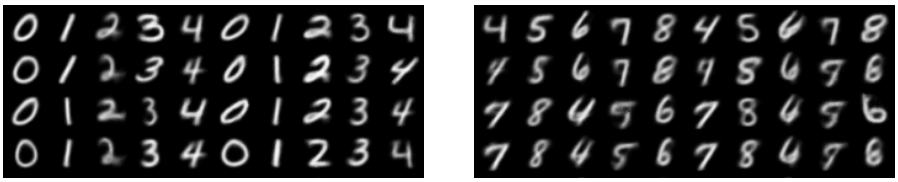


Figure 2: Synthesis result for sequences of MNIST digits with VAE

5 EXPERIMENTS

5.1 OVERVIEW

Due to its impressive results on natural images, we employ the generator of the Deep Convolutional GAN (DCGAN, Radford et al. (2015)) as the foundation for implementing the observer function C_η. We use it in combination with an affine layer at the input. This layer serves three purposes. Firstly, it accounts for changes of bases in the latent space in order to conform with Assumption 2. Secondly, it includes a bias so that the fixed point of the transition function can be transformed to the origin according to Proposition 1. Finally, it further reduces the latent dimension to make the search for the transition matrix $\boldsymbol{A}$ feasible.

We train C_η and $\boldsymbol{A}$ simultaneously by means of the dynamic layer introduced in Section 4.3. To this end, we integrate the construction in Fig. 1 as the decoder of a VAE (without batch normalization) and the generator of a Wasserstein GAN (with batch normalization), respectively. As the critic (discriminator) network of the Wasserstein GAN, we use the original discriminator of the DCGAN but double the number of output channels in order to match the dimensions of the training frame pairs. As the encoder of the VAE, we use the discriminator of the DCGAN, but adapt the number of output channels to be $2n$, where n is the latent dimension of the model. The latent dimension is set to $n = 10$ for all experiments.

Synthesis is performed by sampling from the MAR model described by

$$\boldsymbol{h}_{t+1} = \boldsymbol{A}\boldsymbol{h}_t + \boldsymbol{B}\boldsymbol{v}_t, \qquad \mathbf{v}_t \sim \text{ i.i.d. } \mathcal{N}(\mathbf{v}_t, 0, \boldsymbol{I}_n), \tag{20}$$

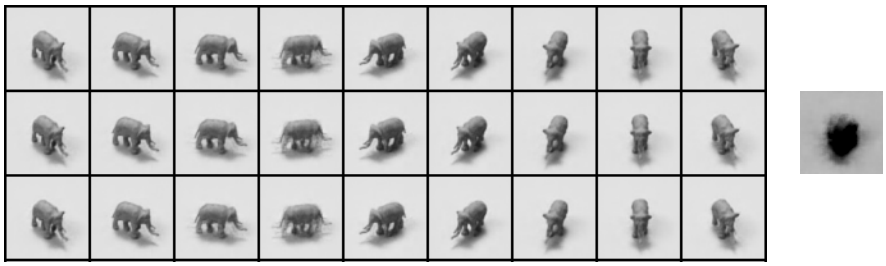
and mapping the latent states to the observation space by means of C_η. The initial latent state $\boldsymbol{h}_0$ is sampled from Gaussian white noise for the Wasserstein GAN experiments. For the VAE experiments, it is estimated by applying the encoder to an observation frame pair from the training sequence. PyTorch code for reproducing the experiments will be made available online upon publication, along with the entire set of results.

5.2 VARIATIONAL AUTOENCODER

An isotropic standard Gaussian model $\mathcal{N}(\mathbf{s}; f_\theta(\mathbf{x}), \sigma^2)$ is assumed for the conditional data distribution. The Adam optimizer with a step size of $2.5 * 10^{-4}$ is used to train the architecture. The regularizer constant is set to $\lambda = 100$. We carry out two series of experiments.

In the first series of experiments, we use the *MNIST* dataset to generate repeating sequences of hand written digits. We choose $\sigma = 4$ and use sequences of length 10000 for training the network for 25 epochs. Most of the number sequences we test can be well reproduced. Fig. 2 depicts the synthesis results for the sequences `0123401234...` and `4567845678...`. Occasional jumps occur, due to the stochastic nature of the model. An observation we make is that higher values of σ improve the probability of synthesizing the correct sequence, but decrease the variability of digit shapes.

To investigate the linearization of geometrical transformations, we employ Category 0 of the *Small NORB* dataset that contains pictures of miniature animals under 6 different lighting conditions, 9 elevational and 18 azimuthal poses. We train our architecture for 100 epochs with $\sigma = \sqrt{5}$ to synthesize sequences depicting azimuthal rotations of 20° per frame. Generally, the synthesis provides good results. For each one of the five animals, the synthesis yields clearly recognizable rotation movements. However, the algorithm tends to confuse pairs of opposite poses, as can be observed in Fig. 3a, where the poses are flipped after the fourth frame in each row. With regard to Section 3.2,



(a) (b)

Figure 3: Synthesis result for NORB sequences (a) and learned fixed point (b)

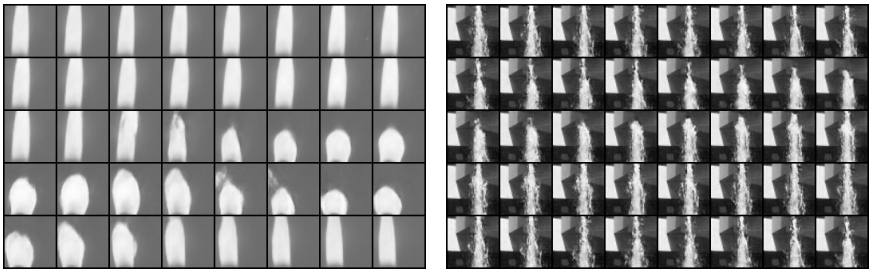


Figure 4: Synthesis result for sequences of the UCLA-50 dataset with Wasserstein GAN

two explanations for this can be identified. On the one hand, it is possible that the assumptions of Proposition 1 are not fulfilled, and no diffeomorphism, i.e. a bijective mapping exists from the image manifold to the embedded space. In that case, the best we can do is finding a non-injective local diffeomorphism that assigns more than one point on the image manifold to a point in the embedded space. On the other hand, it is thinkable, that an actual diffeomorphism exists, but that the euclidean distance implicitly minimized by the VAE is a bad estimation of the geodesic distance on the manifold. Fig. 3b depicts the learned fixed point of the transition function. It can be thought of as a rotationally invariant structure under limited exposure of light from above.

5.3 Wasserstein Generative Adversarial Network

We employ the Wasserstein GAN for dynamic texture synthesis experiments and set the regularizer to $\lambda = 1$. We firstly test our method on the cropped *UCLA-50* dataset of grayscale DTs. The dataset contains 50 classes of 4 sequences of length 75 each. We use one class at a time for the experiments. The architecture is trained for 2000 epochs via RMSPROP with step size $2.5 * 10^{-4}$. Fig. 4 depicts the results for the classes `candle` and `fountain-c-far`.

Finally, we evaluate the method on three RGB sequences. We run the optimization via RMSPROP with step size $5 * 10^{-5}$ for 900 epochs. Fig. 5 depicts the result for the `firepot`, `springwater` and `waterfountain` sequences taken from Xie et al. (2017). Despite the low complexity of our model, we believe that the difference in quality compared to the state of the art results reported in Xie et al. (2017) is negligible.

6 Conclusion

This work presents an approach to learn embedded MAR models from image sequences. We motivate the feasibility of this approach by introducing the concept of local linearizability and propose a joint learning procedure that employs deep generative models in combination with an additional linear component, the dynamic layer. We report first positive results on low-resolution visual processes, where a first-order Markov property can be assumed, and hope to shed some light on the nature of linearization. A possible future research direction is improving the theoretical understanding of linearizing representations and their applicability outside of stationary visual processes.

Figure 5: Synthesis result for RGB sequences. Frames are rescaled to match the original aspect ratio

REFERENCES

Bijan Afsari and René Vidal. The alignment distance on spaces of linear dynamical systems. In *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*, pp. 1162–1167. IEEE, 2013.

Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein gan. *arXiv preprint arXiv:1701.07875*, 2017.

Rakesh Chalasani and Jose C Principe. Deep predictive coding networks. *arXiv preprint arXiv:1301.3541*, 2013.

Antoni B Chan and Nuno Vasconcelos. Classifying video with kernel dynamic textures. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–6. IEEE, 2007.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International Conference on Machine Learning*, pp. 2990–2999, 2016.

Taco S Cohen and Max Welling. Transformation properties of learned visual representations. *arXiv preprint arXiv:1412.7659*, 2014.

CR De Souza, A Gaidon, Y Cabon, and AM Lopez Pena. Procedural generation of videos to train deep action recognition networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

Gianfranco Doretto, Alessandro Chiuso, Ying Nian Wu, and Stefano Soatto. Dynamic textures. *International Journal of Computer Vision*, 51(2):91–109, 2003.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

Ross Goroshin, Michael F Mathieu, and Yann LeCun. Learning to linearize under uncertainty. In *Advances in Neural Information Processing Systems*, pp. 1234–1242, 2015.

Tian Han, Yang Lu, Song-Chun Zhu, and Ying Nian Wu. Alternating back-propagation for generator network. In *AAAI*, volume 3, pp. 13, 2017.

Geoffrey E Hinton, Alex Krizhevsky, and Sida D Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pp. 44–51. Springer, 2011.

Matthew Johnson, David K Duvenaud, Alex Wiltschko, Ryan P Adams, and Sandeep R Datta. Composing graphical models with neural networks for structured representations and fast inference. In *Advances in neural information processing systems*, pp. 2946–2954, 2016.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pp. 700–708, 2017a.

Wenqian Liu, Abhishek Sharma, Octavia Camps, and Mario Sznaier. Dyan: A dynamical atoms-based network for video prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 170–185, 2018.

Ziwei Liu, Raymond Yeh, Xiaoou Tang, Yiming Liu, and Aseem Agarwala. Video frame synthesis using deep voxel flow. In *International Conference on Computer Vision (ICCV)*, volume 2, 2017b.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

Roland Memisevic. Learning to relate images. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1829–1846, 2013.

Lawrence Perko. *Differential equations and dynamical systems*, volume 7. Springer Science & Business Media, 2013.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. 2015.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pp. 843–852, 2015.

Carl Vondrick and Antonio Torralba. Generating the future with adversarial transformers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pp. 2746–2754, 2015.

Xian Wei, Yuanxiang Li, Hao Shen, Fang Chen, Martin Kleinsteuber, and Zhongfeng Wang. Dynamical textures modeling via joint video dictionary learning. *IEEE Transactions on Image Processing*, 26(6):2929–2943, 2017.

Jianwen Xie, Song-Chun Zhu, and Ying Nian Wu. Synthesizing dynamic patterns by spatial-temporal generative convnet. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7093–7101, 2017.

Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *Advances in Neural Information Processing Systems*, pp. 91–99, 2016.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *arXiv preprint*, 2017.

A Appendix

A.1 Proof of Proposition 1

Let $\mathbf{\Phi} \in \mathbb{R}^{n\times n}$ be a matrix. Since Γ' is a diffeomorphism, we define $\phi = \Gamma'^{-1} \circ \mathbf{\Phi} \circ \Gamma'$. We need to show that

$$\lim_{\|\boldsymbol{y}-\boldsymbol{y}^*\|\to 0} \frac{\|\varphi(\boldsymbol{y}) - \phi(\boldsymbol{y})\|}{\|\boldsymbol{y}-\boldsymbol{y}^*\|} = 0 \tag{21}$$

holds. Let us denote the Jacobian of ϕ at $\boldsymbol{y}^*$ by $\boldsymbol{J}_\phi$. Because Γ' maps $\boldsymbol{y}^*$ to the origin, we can reformulate the requirement as

$$\begin{aligned} &\lim_{\|\boldsymbol{y}-\boldsymbol{y}^*\|\to 0} \frac{\|\varphi(\boldsymbol{y}) - \boldsymbol{y}^* - \phi(\boldsymbol{y}) + \boldsymbol{y}^*\|}{\|\boldsymbol{y}-\boldsymbol{y}^*\|} \\ =&\lim_{\|\boldsymbol{y}-\boldsymbol{y}^*\|\to 0} \frac{\|\varphi(\boldsymbol{y}) - \varphi(\boldsymbol{y}^*) - \phi(\boldsymbol{y}) + \phi(\boldsymbol{y}^*)\|}{\|\boldsymbol{y}-\boldsymbol{y}^*\|} \\ =&\lim_{\|\boldsymbol{y}-\boldsymbol{y}^*\|\to 0} \frac{\|\varphi(\boldsymbol{y}) - \varphi(\boldsymbol{y}^*) - \boldsymbol{J}_\phi(\boldsymbol{y}-\boldsymbol{y}^*)\|}{\|\boldsymbol{y}-\boldsymbol{y}^*\|} = 0. \end{aligned} \tag{22}$$

This requirement is fulfilled if the Jacobi matrices of φ and ϕ coincide, i.e.

$$\boldsymbol{J}_\phi = \boldsymbol{J}_\varphi. \tag{23}$$

The Jacobian of ϕ at $\boldsymbol{y}^*$ is given, according to he chain rule, by

$$\boldsymbol{J}_\phi = \boldsymbol{J}_{\Gamma^{-1}} \boldsymbol{\Phi} \boldsymbol{J}_\Gamma, \tag{24}$$

where $\boldsymbol{J}_{\Gamma^{-1}} \in \mathbb{R}^{d \times n}$ is the Jacobian matrix of Γ^{-1} at $\Gamma(\boldsymbol{y}^*)$. A matrix $\boldsymbol{\Phi} \in \mathbb{R}^{n \times n}$ can be always found, such that Eq. (23) is fulfilled, if the columns and rows of $\boldsymbol{J}_\varphi$ lie in the column space of $\boldsymbol{J}_{\Gamma^{-1}}$ and the row space of $\boldsymbol{J}_\Gamma$, respectively. The column space of $\boldsymbol{J}_{\Gamma^{-1}}$ coincides with the row space of $\boldsymbol{J}_\Gamma$, due to the identity $\boldsymbol{J}_\Gamma \boldsymbol{J}_{\Gamma^{-1}} = \boldsymbol{I}_n$. The statement of the proposition follows.

A.2 Generalization to Higher-order Markov Processes

The proposed method is designed to linearize first-order Markov processes only. This is in line with our empirical evaluation, in which the method was not capable of synthesizing sequences with persisting motions of non-constant velocities, e.g. the videos of running animals in Xie et al. (2017). However, the presented approach can be theoretically generalized to Markov processes of order $m \geq 1$. We present a procedure that could be employed for this purpose. Further investigation is necessary to evaluate the feasibility and practical applicability of such an approach.

The general procedure is as follows.

1. Set up an appropriate block matrix model for the joint probability of $m+1$ succeeding observations.
2. Build a dynamic layer that performs a multiplication with a lower-triangular $(m+1) \times (m+1)$ block matrix $\boldsymbol{F}$.
3. Introduce regularizers to preserve the block Toeplitz structure of the covariance matrix,
4. Compute the MAR parameters from the resulting covariance matrix.

We illustrate the procedure for the case $m = 2$.

1. Due to the stationarity assumption, the covariance matrix for three succeeding observations $\mathbf{h}_t, \mathbf{h}_{t+1}, \mathbf{h}_{t+2}$ has the form

$$\boldsymbol{\Sigma}_2 = \begin{bmatrix} \boldsymbol{I}_n & \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1}) & \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+2}) \\ \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1})^\top & \boldsymbol{I}_n & \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1}) \\ \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+2})^\top & \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1})^\top & \boldsymbol{I}_n \end{bmatrix}. \tag{25}$$

2. The matrix describing the dynamic layer has the form

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{I}_n & & \\ \boldsymbol{F}_1 & \boldsymbol{F}_2 & \\ \boldsymbol{F}_3 & \boldsymbol{F}_4 & \boldsymbol{F}_5. \end{bmatrix} \tag{26}$$

   The outputs of the dynamic layer will thus have the distribution

$$p\left(\begin{bmatrix}\mathbf{h}_1^\top & \mathbf{h}_2^\top & \mathbf{h}_3^\top\end{bmatrix}^\top\right) = \mathcal{N}(\begin{bmatrix}\mathbf{h}_1^\top & \mathbf{h}_2^\top & \mathbf{h}_3^\top\end{bmatrix}^\top;\, 0, \boldsymbol{F}\boldsymbol{F}^\top). \tag{27}$$

   We thus need to achieve

$$\boldsymbol{\Sigma}_2 \approx \boldsymbol{F}\boldsymbol{F}^\top = \begin{bmatrix} \boldsymbol{I}_n & \boldsymbol{F}_1^\top & \boldsymbol{F}_3^\top \\ \boldsymbol{F}_1 & \boldsymbol{F}_1\boldsymbol{F}_1^\top + \boldsymbol{F}_2\boldsymbol{F}_2^\top & \boldsymbol{F}_1\boldsymbol{F}_3^\top + \boldsymbol{F}_2\boldsymbol{F}_4^\top \\ \boldsymbol{F}_3^\top & \boldsymbol{F}_3\boldsymbol{F}_1^\top + \boldsymbol{F}_4\boldsymbol{F}_2^\top & \boldsymbol{F}_3\boldsymbol{F}_3^\top + \boldsymbol{F}_4\boldsymbol{F}_4^\top + \boldsymbol{F}_5\boldsymbol{F}_5^\top \end{bmatrix}. \tag{28}$$

3. This can be ensured by using the regularizer

$$\begin{aligned} R(\boldsymbol{F}) = \lambda_1 \|\boldsymbol{F}_1\boldsymbol{F}_1^\top + \boldsymbol{F}_2\boldsymbol{F}_2^\top - \boldsymbol{I}_n\|_F^2 + \lambda_2 \|\boldsymbol{F}_3\boldsymbol{F}_3^\top + \boldsymbol{F}_4\boldsymbol{F}_4^\top + \boldsymbol{F}_5\boldsymbol{F}_5^\top - \boldsymbol{I}_n\|_F^2 \\ + \lambda_3 \|\boldsymbol{F}_3\boldsymbol{F}_1^\top + \boldsymbol{F}_4\boldsymbol{F}_2^\top - \boldsymbol{F}_1\|_F^2, \end{aligned} \tag{29}$$

   with $\lambda_1, \lambda_2, \lambda_3 > 0$.

4. A second-order MAR model has the form

$$\boldsymbol{h}_{t+2} = \boldsymbol{A}_0\boldsymbol{h}_t + \boldsymbol{A}_1\boldsymbol{h}_{t+1} + \boldsymbol{B}\boldsymbol{v}_t,\ \mathbf{v}_t \sim \mathcal{N}(\mathbf{v}_t, 0, \boldsymbol{I}_n). \tag{30}$$

By our assumptions on the process, this yields the system of equations,

$$\begin{aligned}\mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_t) &= \boldsymbol{A}_0\boldsymbol{A}_0^\top + \boldsymbol{A}_1\boldsymbol{A}_1^\top + \boldsymbol{A}_0\mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1})\boldsymbol{A}_1^\top \\ &\quad + \boldsymbol{A}_1\mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1})^\top\boldsymbol{A}_0^\top + \boldsymbol{B}\boldsymbol{B}^\top \\ \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1}) &= \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1})^\top\boldsymbol{A}_0^\top + \boldsymbol{A}_1^\top, \\ \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+2}) &= \boldsymbol{A}_0^\top + \mathrm{Cov}(\mathbf{h}_t, \mathbf{h}_{t+1})\boldsymbol{A}_1^\top.\end{aligned} \tag{31}$$

Thus, we can infer $\boldsymbol{A}_0, \boldsymbol{A}_1$ and $\boldsymbol{B}$ via solving

$$\begin{aligned}\boldsymbol{F}_1^\top &= \boldsymbol{F}_1\boldsymbol{A}_0^\top + \boldsymbol{A}_1^\top, \\ \boldsymbol{F}_3 &= \boldsymbol{A}_0 + \boldsymbol{A}_1\boldsymbol{F}_1, \\ \boldsymbol{I}_n &= \boldsymbol{A}_0\boldsymbol{A}_0^\top + \boldsymbol{A}_1\boldsymbol{A}_1^\top + \boldsymbol{A}_0\boldsymbol{F}_1^\top\boldsymbol{A}_1^\top + \boldsymbol{A}_1\boldsymbol{F}_1\boldsymbol{A}_0^\top + \boldsymbol{B}\boldsymbol{B}^\top.\end{aligned} \tag{32}$$