Learning deep dynamical models from image pixels

Niklas Wahlström * **Thomas B. Schön** **
Marc Peter Deisenroth ***

* *Department of Electrical Engineering, Linköping University, Sweden, (e-mail: nikwa@isy.liu.se)*
** *Department of Information Technology, Uppsala University, Sweden, (e-mail: thomas.schon@it.uu.se).*
*** *Department of Computing, Imperial College London, UK, (e-mail: m.deisenroth@imperial.ac.uk)*

Abstract: Modeling dynamical systems is important in many disciplines, such as control, robotics, or neurotechnology. Commonly the state of these systems is not directly observed, but only available through noisy and potentially high-dimensional observations. In these cases, system identification, i.e., finding the measurement mapping and the transition mapping (system dynamics) in latent space can be challenging. For linear system dynamics and measurement mappings efficient solutions for system identification are available. However, in practical applications, the linearity assumptions does not hold, requiring nonlinear system identification techniques. If additionally the observations are high-dimensional (e.g., images), nonlinear system identification is inherently hard. To address the problem of nonlinear system identification from high-dimensional observations, we combine recent advances in deep learning and system identification. In particular, we jointly learn a low-dimensional embedding of the observation by means of deep auto-encoders and a predictive transition model in this low-dimensional space. We demonstrate that our model enables learning good predictive models of dynamical systems from pixel information only.

Keywords: Deep neural networks, system identification, nonlinear systems, low-dimensional embedding, auto-encoder.

High-dimensional time series include video streams, electroencephalography (EEG) and sensor network data. Dynamical models describing such data are desired for forecasting (prediction) and controller design, both of which play an important role, e.g., in autonomous systems, machine translation, robotics and surveillance applications. A key challenge is system identification, i.e., finding a mathematical model of the dynamical system based on the information provided by measurements from the underlying system. In the context of state-space models this includes finding two functional relationships between (a) the states at different time steps (prediction/transition model) and (b) states and corresponding measurements (observation/measurement model). In the linear case, this problem is well studied, and many standard techniques exist, e.g., subspace methods (Van Overschee and De Moor, 1996), expectation maximization (Shumway and Stoffer, 1982; Ghahramani, 1998; Gibson and Ninness, 2005) and prediction-error methods (Ljung, 1999). However, in realistic and practical scenarios we require nonlinear system identification techniques.

⋆ This work was supported by the Swedish Foundation for Strategic Research under the project *Cooperative Localization* and by the Swedish Research Council under the project *Probabilistic modeling of dynamical systems* (Contract number: 621-2013-5524). MPD was supported by an Imperial College Junior Research Fellowship.

Learning nonlinear dynamical models is an inherently difficult problem, and it has been one of the most active areas in system identification for the last decades (Ljung, 2010; Sjöberg et al., 1995). In recent years, sequential Monte Carlo (SMC) methods have received attention for identifying nonlinear state-space models (Schön et al., 2011), see also the recent survey (Kantas et al., 2015). While methods based on SMC are powerful, they are also computationally expensive. Learning nonlinear dynamical models from very high-dimensional sensor data is even more challenging. First, finding (nonlinear) functional relationships in very high dimensions is hard (unidentifiability, local optima, overfitting, etc.); second, the amount of data required to find a good function approximator is enormous. Fortunately, high-dimensional data often possesses an intrinsic lower dimensionality. We will exploit this property for system identification by finding a low-dimensional representation of high-dimensional data and learning predictive models in this low-dimensional space. For this purpose, we need an automatic procedure to find compact low-dimensional representations/features. Doretto et al. (2003) implemented a subspace identification routine to model dynamical textures from high-dimensional pixel data. Whereas that method relies on a linear dimensionality reduction method, we will consider nonlinear mappings from the high-dimensional data to the low-dimensional features.

The state of the art in learning parsimonious representations of high-dimensional data is currently defined by deep learning architectures, such as deep neural networks (Hinton and Salakhutdinov, 2006), stacked/deep auto-encoders (Vincent et al., 2008) and convolutional neural networks (LeCun et al., 1998), all of which have been successfully applied to image, text, speech and audio data in commercial products, e.g., by Google, Amazon and Facebook. Typically, these feature learning methods are applied to static data sets, e.g., for image classification. The auto-encoder gives explicit expressions of two generative mappings: 1) an encoder g^{-1} mapping the high-dimensional data to the features, and 2) a decoder g mapping the features to high-dimensional reconstructions.

In this paper, we combine feature/representation learning and dynamical systems modeling to obtain good predictive models for high-dimensional time series, e.g., videos. In particular, we use deep auto-encoder neural networks for automatically finding a compact low-dimensional representation of an image. In this low-dimensional feature space, we use a neural network for modeling the nonlinear system dynamics. An simplified illustration of our approach is shown in Fig. 1. An encoder g^{-1} maps an image y_{t-1} at time step $t-1$ to a low-dimensional feature z_{t-1}. In this feature space, a prediction model l maps the feature forward in time to z_t. The decoder g can generate a predicted image y_t at the next time step. This framework needs access to both the encoder g^{-1} and the decoder g, which motivates our use of the auto-encoder as dimensionality reduction technique. Crucially, the embedding and the predictive model in feature space are learned *jointly*.

The contributions of this paper are (a) a model for learning a low-dimensional dynamical representation of high-dimensional data, which can be used for long-term predictions; (b) experimental evidence demonstrating that joint learning of the parameters in the latent embedding and in the predictive model in latent space can increase the performance compared to separate training.

1. MODEL

We consider a dynamical system where control inputs are denoted by u and observations are denoted by y. In this paper, the observations are pixel information from images. We assume that a low-dimensional latent variable z exists that compactly represents the relevant properties of y. Since we consider dynamical systems, a low-dimensional representation z of a (static) image y is insufficient to capture important dynamic information, such as velocities. Thus, we introduce an additional latent variable x, the *state*. In our case, the state x_t contains features from multiple time steps (e.g., $t-1$ and t) to capture velocity (or higher-order) information. Therefore, our transition model does not map features at time $t-1$ to time t (as illustrated in Fig. 1), but the transition function f maps states x_{t-1} and control inputs u_{t-1} to states x_t. The full dynamical system is given as the state-space model

$$x_{t+1} = f(x_t, u_t; \theta) + w_t(\theta), \tag{1a}$$
$$z_t = h(x_t; \theta) + v_t(\theta), \tag{1b}$$
$$y_t = g(z_t; \theta) + e_t(\theta), \tag{1c}$$

where each measurement y_t can be described by a low-dimensional feature representation z_t (1c). These features are in turn modeled with a low-dimensional state-space model in (1a) and (1b), where the state x_t contains the full information about the state of the system at time instant t, see also Fig. 2 (left). Here $w_t(\theta)$, $v_t(\theta)$ and $e_t(\theta)$ are sequences of independent random variables and θ are the model parameters. The control inputs u_t will be important in controller design, which is further elaborated upon in Wahlström et al. (2015).

1.1 Approximate prediction model

To identify parameters in dynamical systems, the prediction-error method will be used, which requires a prediction model. In general, it is difficult to derive a prediction model based on the nonlinear state-space model (1), and a closed-form expression for the prediction is only available in a few special cases (Ljung, 1999). However, by approximating the optimal solution, a nonlinear autoregressive exogenous model (NARX) (Ljung, 1999) can be used

$$\widehat{z}_{t|t-1}(\theta_{\mathrm{M}}) = l(z_{t-1}, u_{t-1}, \dots, z_{t-n}, u_{t-n}; \theta_{\mathrm{M}}), \tag{2}$$

where l is a nonlinear function, in our case a neural network and θ_{M} is the corresponding model parameters. The model parameters in the nonlinear function are normally estimated by minimizing the sum of the prediction errors $\|z_t - \widehat{z}_{t|t-1}(\theta_{\mathrm{M}})\|$. However, as we are interested in a good predictive performance for the high-dimensional data y rather than for the features z, we transform the predictions back to the high-dimensional space and obtain a prediction $\widehat{y}_{t|t-1} = g(\widehat{z}_{t|t-1}; \theta_{\mathrm{D}})$, which we use in our error measure.

An additional complication is that we do not have access to the features z_t. Therefore, before training, the past values of the time series have to be replaced with their feature representation $z = g^{-1}(y; \theta_{\mathrm{E}})$, which we compute from the pixel information y. Here, g^{-1} is an approximate inverse of g, which will be described in more detail the next section. This gives the final prediction model

$$\widehat{y}_{t|t-1}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}}) = g(\widehat{z}_{t|t-1}(\theta_{\mathrm{E}}, \theta_{\mathrm{M}}); \theta_{\mathrm{D}}), \tag{3a}$$
$$\widehat{z}_{t|t-1}(\theta_{\mathrm{E}}, \theta_{\mathrm{M}}) = l(z_{t-1}(\theta_{\mathrm{E}}), u_{t-1}, \dots, z_{t-n}(\theta_{\mathrm{E}}), u_{t-n}; \theta_{\mathrm{M}}),$$
$$z_t(\theta_{\mathrm{E}}) = g^{-1}(y_t; \theta_{\mathrm{E}}), \tag{3b}$$

which is also illustrated in Fig. 2 (right). The corresponding prediction error will be

$$\varepsilon^{\mathrm{P}}_t(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}}) = y_t - \widehat{y}_{t|t-1}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}}). \tag{4}$$

1.2 Auto-encoder

We use a deep auto-encoder neural network to parameterize the feature mapping and its inverse. It consists of a deep encoder network g^{-1} and a deep decoder network g. Each layer k of the encoder neural network g^{-1} computes $y_t^{(k+1)} = \sigma(A_k y_t^{(k)} + b_k)$, where σ is an activation function and A_k and b_k are free parameters. The control input to the first layer is the image, i.e., $y_t^{(1)} = y_t$. The last layer is the low-dimensional feature representation of the image $z_t(\theta_{\mathrm{E}}) = g^{-1}(y_t; \theta_{\mathrm{E}})$, where $\theta_{\mathrm{E}} = [\dots, A_k, b_k, \dots]$ are the parameters of all neural network layers. The decoder g consists of the same number of layers in reverse order, see Fig. 3, and can be considered an approximate inverse of the encoder g, such that $\widehat{y}_{t|t}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}) \approx y_t$, where

$$\widehat{y}_{t|t}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}) = g(g^{-1}(y_t; \theta_{\mathrm{E}}); \theta_{\mathrm{D}}) \tag{5}$$

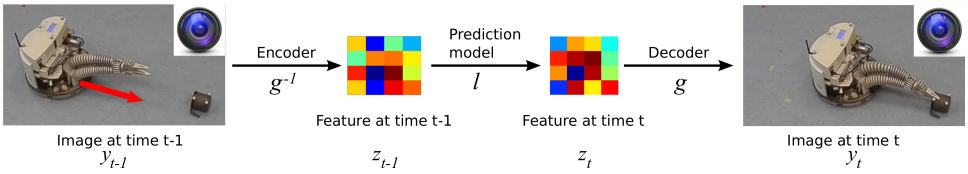

Fig. 1. Combination of deep learning architectures for feature learning and prediction models in feature space. A camera observes a robot approaching an object. A good low-dimensional feature representation of an image is important for learning a predictive model if the camera is the only sensor available.

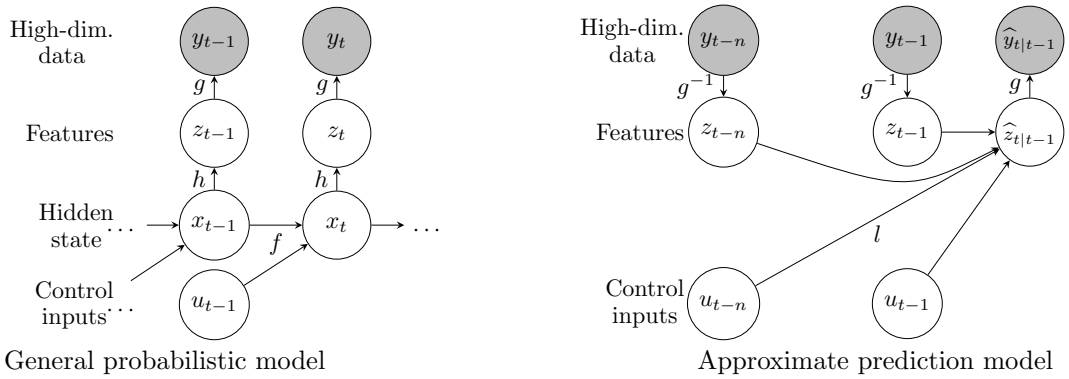

Fig. 2. Left - The general graphical model: Each data point y_t has a low-dimensional representation z_t, which is modeled using a state-space model with hidden state x_t and control input u_t. Right - The approximate prediction model: The predicted feature $\widehat{z}_{t|t-1}$ is a function of the n past features z_{t-n} to z_{t-1} and n past control inputs u_{t-n} to u_{t-1}. Each of the features z_{t-n} to z_{t-1} is computed from high-dimensional data y_{t-n} to y_{t-1} via the encoder g^{-1}. The predicted feature $\widehat{z}_{t|t-1}$ is mapped to predicted high-dimensional data via the decoder g.

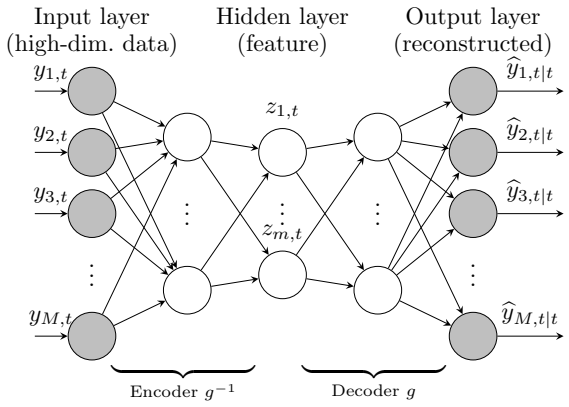

Fig. 3. An auto-encoder consisting of an encoder g^{-1} and a decoder g. The original image $y_t = [y_{1,t}, \cdots, y_{M,t}]^\mathsf{T}$ is mapped into its low-dimensional representation $z_t = [z_{1,t}, \cdots, z_{m,t}]^\mathsf{T} = g^{-1}(y_t)$ with the encoder, and then back to a high-dimensional representation $\widehat{y}_{t|t-1} = g(\widehat{z}_{t|t-1})$ by the decoder g, where $M \gg m$.

is the reconstructed version of y_t. The encoder and decoder are trained jointly to minimize the reconstruction error

$$\varepsilon_t^{\mathrm{R}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}) = y_t - \widehat{y}_{t|t}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}), \tag{6}$$

and the parameters θ_{D}, θ_{E} of g and g^{-1}, respectively, can be coupled to constrain the solution to some degree (Vincent et al., 2008).

The auto-encoder suits our system identification problem well, since it provides an explicit expression of both the mapping g as well as its approximate inverse g^{-1}, which we need for the predictions in (3a).

2. TRAINING

To summarize, our model contains the following free parameters: the parameters for the encoder θ_{E}, the parameters for the decoder θ_{D} and the parameters for the prediction model θ_{M}. To train the model, we employ two cost functions, the sum of the prediction errors (4),

$$V_{\mathrm{P}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}}) = \sum_{t=1}^{N} \|\varepsilon_t^{\mathrm{P}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}})\|^2, \tag{7a}$$

and the sum of the reconstruction errors (6),

$$V_{\mathrm{R}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}) = \sum_{t=1}^{N} \|\varepsilon_t^{\mathrm{R}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}})\|^2. \tag{7b}$$

Generally, there are two ways of finding the model parameters: (i) separate training and (ii) joint training of the auto-encoder and the prediction model, both of which are explained below.

2.1 Separate training

Normally when features are used for learning dynamical models, they are first extracted from the data in a preprocessing step. In a second step the prediction model is estimated based on these features. In our setting, this corresponds to sequentially training the model using two cost functions (7a)–(7b): We first learn a compact feature representation by minimizing the reconstruction error

$$(\widehat{\theta}_{\mathrm{E}}, \widehat{\theta}_{\mathrm{D}}) \in \arg\min_{\theta_{\mathrm{E}}, \theta_{\mathrm{D}}} V_{\mathrm{R}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}), \tag{8a}$$

and, subsequently, train the prediction model by minimizing the prediction error

$$\widehat{\theta}_{\mathrm{M}} = \arg\min_{\theta_{\mathrm{M}}} V_{\mathrm{P}}(\widehat{\theta}_{\mathrm{E}}, \widehat{\theta}_{\mathrm{D}}, \theta_{\mathrm{M}}), \tag{8b}$$

with fixed auto-encoder parameters $\widehat{\theta}_{\mathrm{E}}, \widehat{\theta}_{\mathrm{D}}$. The gradients of these cost functions with respect to the model parameters can be computed efficiently by back-propagation. The cost functions are then minimized by the BFGS algorithm (Nocedal and Wright, 2006).

2.2 Joint training

An alternative to separate training is to minimize the reconstruction error and the prediction error jointly by considering the optimization problem

$$\left(\widehat{\theta}_{\mathrm{E}}, \widehat{\theta}_{\mathrm{D}}, \widehat{\theta}_{\mathrm{M}}\right) = \underset{\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}}}{\arg\min} \left(V_{\mathrm{R}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}) + V_{\mathrm{P}}(\theta_{\mathrm{E}}, \theta_{\mathrm{D}}, \theta_{\mathrm{M}})\right), \quad (9)$$

where we jointly optimize the free parameters in both the auto-encoder θ_{E}, θ_{D} and the prediction model θ_{M}. Again, back-propagation is used for computing the gradients of this cost function. Note that in (9) it is crucial to include not only the prediction error V_{P}, but also the reconstruction error V_{R}. Without this term the multi-step ahead prediction performance will decrease because predicted features are not consistent with features achieved from the encoder. The multi-step ahead predictive performance is crucial to design a controller for this system (Wahlström et al., 2015).

2.3 Initialization

With a linear activation function the auto-encoder and PCA are identical Bourlard and Kamp (1988), which we exploit to initialize the parameters of the auto-encoder: The auto-encoder network is unfolded, each pair of layers in the encoder and the decoder are combined, and the corresponding PCA solution is computed for each of these pairs. We start with high-dimensional image data at the top layer and use the principal components from that pair of layers as input to the next pair of layers. Thereby, we recursively compute a good initialization for all parameters of the auto-encoder. Similar pre-training routines are found in Hinton and Salakhutdinov (2006), in which a restricted Boltzmann machine is used instead of PCA.

3. RESULTS

We report results on identification of the nonlinear dynamics of a planar pendulum (1-link robot arm) and the torque as control input. In this example, we learn the dynamics solely based on pixel information. Each pixel $y_t^{(i)}$ is a component of the measurement $y_t = [y_t^{(1)}, \ldots, y_t^{(M)}]^{\mathsf{T}}$ and assumes a continuous gray-value in $[0, 1]$. In (Wahlström et al., 2015) an model predictive controller is used to compute the control inputs, whereas we in this work use random control inputs.

We simulated 500 frames of a pendulum moving in a plane with $51 \times 51 = 2\,601$ pixels in each frame. To speed up training, the image input has been reduced to $\dim(y_t) = 50$ prior to model learning (system identification) using PCA. With these 50 dimensional inputs, four layers have been used for the encoder g^{-1} as well as the decoder g with dimension 50-25-12-6-2. Hence, the features have dimension $\dim(x_t) = 2$. The order of the dynamics was chosen as $n = 2$ to capture velocity information. For the prediction model l we used a two-layer neural network with a 6-4-2 architecture.

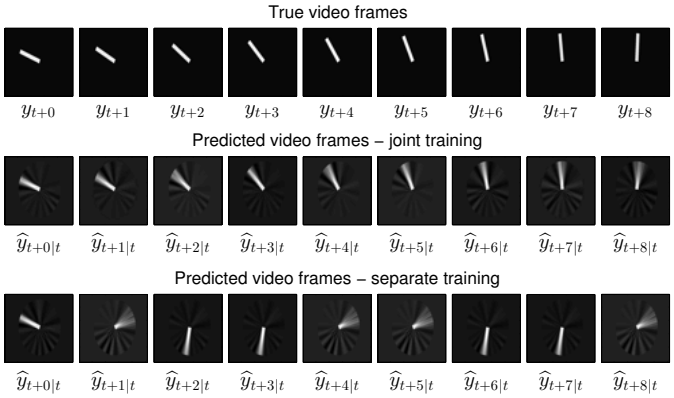

Fig. 4. A typical image sequence and corresponding prediction results (validation data), computed according to (10). The top rows show nine consecutive ground truth image frames from time instant t to $t + 8$. The second and the third rows display the corresponding long-term ahead predictions based on measured images up to time t for both joint (center) and separate training (bottom) of the model parameters.

We evaluate the performance in terms of long-term predictions where we assumed that a sequence of open-loop torques was given. These predictions are constructed by concatenating multiple 1-step ahead predictions. More precisely, the p-step ahead prediction $\widehat{y}_{t+p|t} = g(\widehat{z}_{t+p|t})$ is computed iteratively as

$$\widehat{z}_{t+1|t} = l(\widehat{z}_{t|t}, u_t, \ldots), \quad (10a)$$

$$\ldots$$

$$\widehat{z}_{t+p|t} = l(\widehat{z}_{t+p-1|t}, u_{t+p-1|t}, \ldots), \quad (10b)$$

where $\widehat{z}_{t|t} = g^{-1}(y_t)$ are the image features at time t.

The predictive performance on an exemplary image sequence of the validation data of our system identification models is illustrated in Fig. 4. The top row shows the ground truth images, the center row shows the predictions based on a model using joint training (9), the bottom row shows the corresponding predictions of a model where the auto-encoder and the predictive model were trained sequentially according to (8). The model that jointly learns all parameters yields a good predictive performance for both one-step ahead prediction and multiple-step ahead prediction. Compared to this, the predictive performance of the model that learns features and the dynamics separately is worse. Although the auto-encoder does a perfect job (left-most frame, 0-step ahead prediction), already the (reconstructed) one-step ahead prediction is dissimilar to the ground-truth image. This is also shown in Table 1 where the reconstruction error is equally good for both models, but for the prediction error we manage to get a better value using joint training than using separate training. Let us have a closer look at the model based on separate training: As the auto-encoder performs well, the learned transition model is the cause of bad predictive performance. We believe that the auto-encoder found a good feature representation for reconstruction, but this representation was not ideal for learning a transition model.

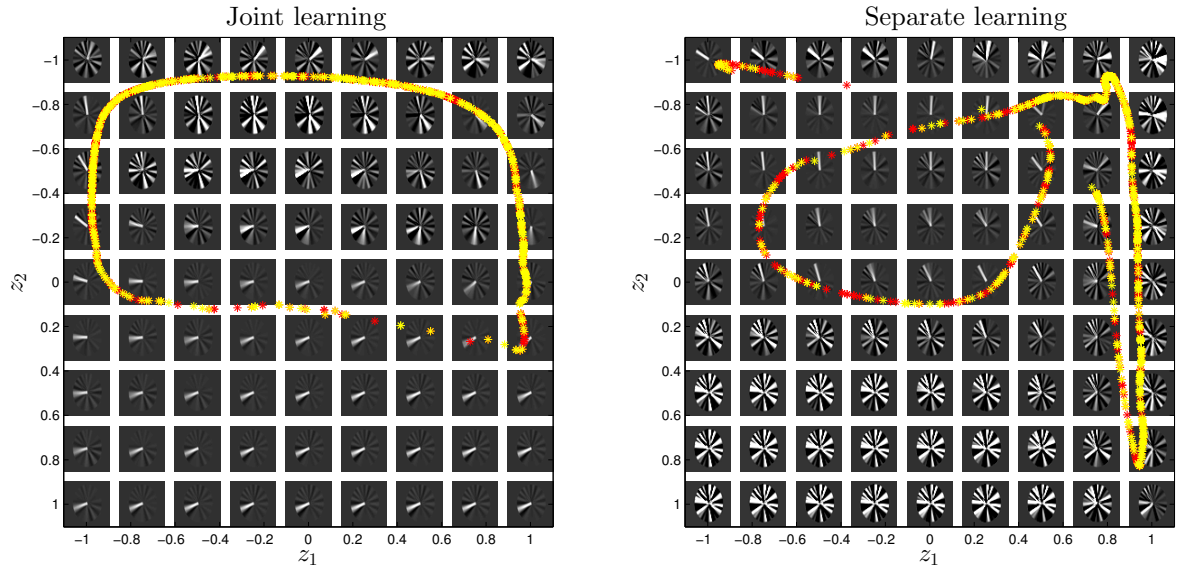

Fig. 5. The feature space $z \in [-1, 1] \times [-1, 1]$ is divided into 9×9 grid points. For each grid point the decoded high-dimensional image is displayed. The features corresponding to the training (red) and validation (yellow) data are displayed. Feature spaces found by joint (left) and separate (right) parameter learning.

Table 1. Prediction error V_{P} and reconstruction error V_{R} for separate and joint training.

Training	V_{P}	V_{R}
Joint training (9)	0.0011	0.0011
Separate training (8)	0.0051	0.0011

Fig. 5 displays the "decoded" images corresponding to the latent representations using joint and separate training, respectively. After joint training the relevant features line up in a circular shape, such that a relatively simple prediction model is sufficient to describe the dynamics. However, for the separate training such an advantageous structure of the feature values are not obtained. Separate training extracts the low-dimensional features without context, i.e., the knowledge that these features constitute a time series.

In this particular data set, the data points clearly reside on one-dimensional manifold, encoded by the pendulum angle. However, a one-dimensional feature space would be insufficient since this one-dimensional manifold is cyclic, see Fig. 5, compare also with the 2π period of an angle. Therefore, we have used a two-dimensional latent space.

To analyze the long-term predictive performance of both training methods, we define the fitting quality as

$$\mathrm{FIT}_p = 1 - \sqrt{\frac{1}{NM} \sum_{t=1}^{N} \|y_t - \widehat{y}_{t|t-p}\|^2}. \qquad (11)$$

As a reference, the predictive performance is compared with a baseline prediction using the previous frame at time step $t - p$ as the prediction at t as $\widehat{y}_{t|t-p} = y_{t-p}$.

The result for a prediction horizon ranging from $p = 0$ to $p = 8$ is displayed in Fig. 6. Clearly, joint learning (blue) outperforms separate learning in terms of predictive performance for prediction horizons greater than 0. Even by using the last available image frame for prediction (const. pred., brown), we obtain a better fit than the model that learns its parameter sequentially (red). This is due to the fact that the dynamical model often predicts frames, which

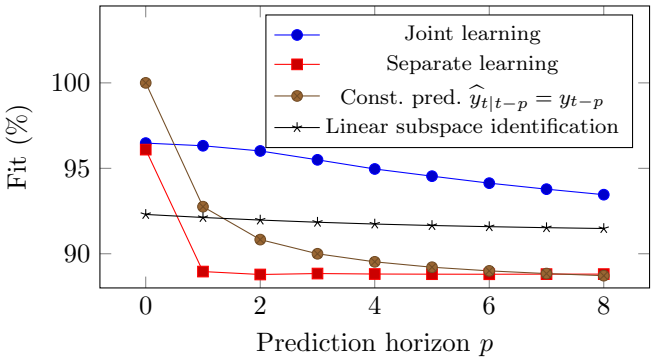

Fig. 6. Fitting quality (11) for joint and separate learning of features and dynamics for different prediction horizons p. The fit is compared with the naive prediction $\widehat{y}_{t|t-p} = y_{t-p}$, where the most recent image is used and a linear subspace-ID method.

do not correspond to any real pendulum, see Fig. 4, leading to a poor fit. Furthermore, joint training gives better predictions than the naive constant prediction. The predictive performance slightly degrades when the prediction horizon p increases, which is to be expected. Finally we also compare with the subspace identification method (Van Overschee and De Moor, 1996) (black, starred), which is restricted to linear models. Such a restriction does not capture the nonlinear, embedded features and, hence, the predictive performance is sub-optimal.

4. DISCUSSION

From a system identification point of view, the prediction-error method, where we minimize the one-step ahead prediction error, is fairly standard. However, in a control or reinforcement learning setting (Wahlström et al., 2015), we are primarily interested in good predictive performance on a longer horizon to do planning. Thus, we have also investigated to include a multi-step ahead prediction error

in the cost (4). These models achieved similar performance, but no significantly better prediction error could be observed either for one-step ahead predictions nor for longer prediction horizons.

Instead of computing the prediction errors in image space, see (4), we can compute errors directly in feature space. However, this will require an extra penalty term to avoid trivial solutions that map everything to zero, resulting in a more complicated and less intuitive cost function.

Although joint learning aims at finding a feature representation that is suitable for modeling the low-dimensional dynamical behavior, the pre-training initialization as described in Section 2.3 does not. If this pre-training yields feature values far from "useful" ones for modeling the dynamics, joint training might not find a good model.

The autoencoder structure has to be chosen before the actual training starts. Especially the dimension of the latent state and the order of the dynamics have to be chosen by the user, which requires some prior knowledge about the system to be identified. In our examples, we chose the latent dimensionality based on insights about the true dynamics of the problem. In general, a model selection procedure will be preferable to find both a good network structure and a good latent dimensionality.

5. CONCLUSIONS AND FUTURE WORK

We have presented an approach to nonlinear system identification from high-dimensional time series data. Our model combines techniques from both the system identification and the machine learning community. In particular, we used a deep auto-encoder for finding low-dimensional features from high-dimensional data, and a nonlinear autoregressive exogenous model was used to describe the low-dimensional dynamics. The framework has been applied to a identifying the dynamics of a planar pendulum from image pixels. The proposed model exhibits good long-term predictive performance, and a major advantage has been identified by training the auto-encoder and the dynamical model jointly compared to training them sequentially.

Possible directions for future work include (a) robustify learning by using denoising autoencoders (Vincent et al., 2008) to deal with noisy real-world data; (b) apply convolutional neural networks, which are often more suitable for images; (c) continue the work in Wahlström et al. (2015) using the model for learning controllers purely based on pixel information; (d) investigate Sequential Monte Carlo methods for systematic treatments of such nonlinear probabilistic models, which are required in a reinforcement learning setting.

In a setting where we make decisions based on predictions, such as optimal control or model-based reinforcement learning, a probabilistic model is often needed for robust decision making as we need to account for model errors (Schneider, 1997; Deisenroth et al., 2015). An extension of our model to a probabilistic setting is non-trivial since random variables have to be transformed through the neural networks, and their exact probability density functions will be intractable to compute. Sampling-based approaches or deterministic approximate inference are two options that we will investigate in future.

REFERENCES

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5), 291–294.

Deisenroth, M.P., Fox, D., and Rasmussen, C.E. (2015). Gaussian processes for data-efficient learning in robotics and control. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 37(2), 408–423.

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

Ghahramani, Z. (1998). Learning dynamic bayesian networks. In C. Giles and M. Gori (eds.), *Adaptive Processing of Sequences and Data Structures*, volume 1387 of *Lecture Notes in Computer Science*, 168–197. Springer.

Gibson, S. and Ninness, B. (2005). Robust maximum-likelihood estimation of multivariable dynamic systems. *Automatica*, 41(10), 1667–1682.

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

Kantas, N., Doucet, A., Singh, S.S., Maciejowski, J., and Chopin, N. (2015). On particle methods for parameter estimation in state-space models. *Statistical Science*. Accepted for publication.

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

Ljung, L. (1999). *System identification, Theory for the user*. System sciences series. Prentice Hall, Upper Saddle River, NJ, USA, second edition.

Ljung, L. (2010). Perspectives on system identification. *Annual Reviews in Control*, 34(1), 1–12.

Nocedal, J. and Wright, S.J. (2006). *Numerical Optimization*. Springer Series in Operations Research. New York, USA, 2 edition.

Schneider, J.G. (1997). Exploiting model uncertainty estimates for safe dynamic control learning. In *NIPS*.

Schön, T.B., Wills, A., and Ninness, B. (2011). System identification of nonlinear state-space models. *Automatica*, 47(1), 39–49.

Shumway, R.H. and Stoffer, D.S. (1982). An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4), 253–264.

Sjöberg, J., Zhang, Q., Ljung, L., Benveniste, A., Delyon, B., Glorennec, P.Y., Hjalmarsson, H., and Juditsky, A. (1995). Nonlinear black-box modeling in system identification: a unified overview. *Automatica*, 31(12), 1691–1724.

Van Overschee, P. and De Moor, B. (1996). *Subspace identification for linear systems - theory, implementation, applications*. Kluwer Academic Publishers.

Vincent, P., Larochelle, H., Bengio, Y., and Manzagol, P.A. (2008). Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*. Helsinki, Finland.

Wahlström, N., Schön, T.B., and Deisenroth, M.P. (2015). From pixels to torques: Policy learning with deep dynamical models. *Preprint, arXiv:1502.02251*.