Deep Prediction Networks

Alberto Dalla Libera[a,*], Gianluigi Pillonetto[b]

[a] *Women's and Children's health Department, University of Padova, Via Giustiniani 3, Padova, 35128, , Italy*
[b] *Department of Information Engineering, University of Padova, Via Giovanni Gradenigo 6, Padova, 35128, , Italy*

Abstract

The challenge for next generation system identification is to build new flexible models and estimators able to simulate complex systems. This task is especially difficult in the nonlinear setting. In fact, in many real applications the performance of long-term predictors may be severely affected by stability problems arising due to the output feedback. For this purpose, also the use of deep networks, which are having much success to solve classification problems, has not led so far to any significant cross-fertilization with system identification. This paper proposes a novel procedure based on a hierarchical architecture, which we call *deep prediction network*, whose flexibility is used to favor the identification of stable systems. In particular, its structure contains layers whose aim is to improve long-term predictions, with complexity controlled by a kernel-based strategy. The usefulness of the new approach is demonstrated through many examples, including important real benchmark problems taken from the system identification literature.

Keywords: Nonlinear System identification, long-term predictions, system stability, Nonparametric estimation, deep networks

1. Introduction

In many dynamical systems the relationship between the input and the output is described by a nonlinear function. Its estimation thus requires the introduction of a nonlinear model and the problem to infer it from the available measurements

*Corresponding author
Email addresses: dallaliber@dei.unipd.it (Alberto Dalla Libera), giapi@dei.unipd.it (Gianluigi Pillonetto)

is called *nonlinear system identification*. This area is complex and the literature surrounding it turns out to be extremely vast. The main reason underlying such richness is that several input-output parameterizations have been introduced, each with different proposed estimation approaches. As explained e.g. in the survey [1], different parametrizations may describe various degrees of system prior knowledge, hence defining grey box models with different shades of grey.

In this paper, we are concerned with black-box identification of the nonlinear input-output map. As in supervised learning [2], we are given some data in form of couples $\{x_t, y_t\}$ and the aim is to reconstruct the functional relationship underlying them. In our dynamic context, y_t is the system output measured at instant t, x_t contains past input-output samples up to $t-1$. A Nonlinear Autoregressive model with eXogenous inputs (NARX model) is postulated of dimension related to the size of x_t. Its identification provides the one-step ahead predictor which returns, for any possible future input location x, the estimated output $\hat{y}$ [3, 4]. We will just assume that such map is smooth and will reconstruct it in a nonparametric fashion. Hence, we will search for it in a very-high (possibly infinite-dimensional) space, then introducing regularization to control model complexity. Important approaches use kernels to encode in an implicit way smoothness information [5, 6, 7, 8, 9]. Relevant for the system identification scenario is the fact that they can also encode fading memory concepts, i.e. past inputs and outputs are expected to be less influent on y_t as the time lag increases [10, 11], or regularized Volterra models [12, 13, 14, 15]. Complexity is then controlled by continuous tuning of a few hyperparameters whose values establish the sensitivity of the estimated output to inputs variations.

Many recent works document how kernel-based techniques may return highly performing one-step-ahead predictors [16, 17, 18, 19]. However, this does not ensure a good simulation of complex systems. This task is much more difficult than one-step-ahead prediction. Indeed, in simulation, the one-step-ahead predictor is evaluated repetitively, with the input defined using previously estimated outputs $\hat{y}_t$. We will refer to this class of prediction networks as *iterated shallow prediction network* (ISPN). Stability of the one-step ahead predictor does not imply system stability and, hence, reliable long term predictions [20]. Such issues can be often encountered in real applications due to the presence of output feedback.

The problem of stable system identification can be arguably faced through two major routes. The first one is *structural* and consists of designing penalty terms or constraints that guarantee some form of system stability. Notable contributions are [21, 22] where incremental stability [23] is studied in a parametric fashion adopting discrete orders, linearly parametrized polynomials and LMI. However, regularization is not exploited and model complexity selection is an important issue. Considering other important properties, like Sontag's input-to-output stability that includes Lyapunov-like stability notions [24, 25], is instead an hard and largely unexplored task.

The second way aims to *favour estimation of stable systems* by designing computational architectures tailored to improve long-term predictions. For instance, an approach suggested to favour long term prediction performance and stability is to estimate separately different k-step ahead predictors, see [26] (Section 2 - Identification Criteria). The stability of each predictor guarantees the stability of the simulation. In the following, we will refer to this class of predictors as *multiple shallow prediction network* (MSPN). The major drawbacks of MSPN are due to the increasing dimension of the input predictor. Indeed, to compute y_{t+k}, the input of the k-step ahead predictor must include also the history of the system input u up to time $t+k-1$. The increasing dimension of the prediction input augments the risk of overfitting, limiting generalization and accuracy. Moreover, computational cost might become prohibitive, since for each prediction step k we have to train a distinct model, with complexity that augments with k.

In this work, we follow an alternative route, proposing a novel form of deep networks. Deep networks are having much success for classification and pattern recognition [27]. Applications abound even if the reasons of their effectiveness are not well understood. For instance, it is unclear why overparametrization may perform well, see however [28, 29] for some insights on benign overfitting for linear regression and some types of compositional functions. Such approaches have also recently attracted the attention of control community [30, 31] but still without a significant impact on system identification. Likely, the main reason is that no deep networks able to incorporate dynamic systems features have been so far designed.

For this reason, in this work we propose a new computational structure for learning nonlinear dynamics, which we call *deep prediction network* (DPN), whose layers are used to promote system stability. Its architecture does not exploit benign overfitting since it includes regularization. In fact, each layer includes kernel-based technology to improve out-of-sample performance. However, our hierarchical scheme adds flexibility to the long-term predictors and exploits it to guard against possible instabilities of the estimated system. Beyond the use of the nonparametric regularized setting, the novelty of our deep prediction network is its ability to connect all the predictors, which is not exploited in ISP and MSPN. All the layers communicate with each other to improve their performance and share the same hyperparameters to control model complexity. Once such hyperparameters are determined from data, all the desired predictors are obtained by solving a single sequence of convex problems whose solutions are available in closed form. Advantages of this new structure in terms of future data prediction will be demonstrated by using artificial and real data taken from the literature. Our results show that, compared to ISP, the DPN favors stability and accuracy. Instead, compared to MSPN, the DPN is more accurate and much more computationally efficient.

The paper is organized as follows. Section 2 reports the problem statement, while Section 3 describes the proposed strategy, highlighting differences w.r.t. the classical approach. Numerical results are discussed in Section 4. Conclusions then end the paper.

2. Problem statement

We consider a discrete-time time-invariant nonlinear dynamic system fed with an input u_t with $t \in \mathscr{Z}$. Our measurements model is

$$y_t = F(x_t) + e_t, \quad t = 1, \cdots, N, \tag{1}$$

where y_t and e_t indicate, respectively, the system output and the noise at instant t while F is the unknown dynamic system. The input-output map depends on the vectors x_t containing past input-output values, i.e.

$$x_t = [y_{t-1}\ y_{t-2}\ \ldots\ y_{t-m_y}\ u_{t-1}\ u_{t-2}\ \ldots\ u_{t-m_u}]. \tag{2}$$

For the sake of simplicity, let $m = m_y = m_u$ so that m is the system memory. We also assume $m < \infty$ so that (1) now becomes a NARX model. Our aim is then to estimate F from N measured couples $\{x_t, y_t\}_{t=1}^{N}$.

3. From shallow to deep prediction networks

3.1. Reproducing kernel Hilbert spaces

We will assume that the compositional maps which define the predictors belong to a special Hilbert space equipped with a reproducing kernel [32]. Formally, the space which contains all the possible regressors x is $\mathbb{R}^{2m}$. Then, $K : \mathbb{R}^{2m} \times \mathbb{R}^{2m} \to \mathbb{R}$ is a positive definite kernel if $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j K(x_i, x_j) \geq 0$ for any possible n-uple of coefficients c_i and regressors x_i. Such kernel then defines a unique Hilbert space $\mathscr{H}$ generated by basis functions which are implicitly encoded by K. Under mild assumptions, one can write $K(x,z) = \sum_i^d \lambda_i \rho_i(x) \rho_i(z)$ were λ_i are positive scalars. Then, ρ_i are the basis functions and the Reproducing

Kernel Hilbert Space (RKHS) dimension is d.
A relevant example for nonlinear system identification is given by the (inhomogeneous) polynomial kernel [33]. Given a couple of input locations (row vectors) $x, z \in \mathbb{R}^{2m}$, one has

$$K(x,z) = \left(xz^\top + q\right)^p, \quad p \in \mathbb{N}, \quad q \geq 0, \tag{3}$$

where q is the only kernel hyperparameter. When $q > 0$, such a kernel induces a $\binom{2m+p}{p}$-dimensional RKHS spanned by all possible monomials up to the p-th degree. The number d of basis function is thus finite but exponential in p. To make an example, assume $q = 1$, $m = 1$ and $p = 2$ with $x = [x_a \; x_b]$ and $z = [z_a \; z_b]$. Then, one obtains the kernel expansion

$$K(x,z) = 1 + x_a^2 z_a^2 + x_b^2 z_b^2 + 2x_a x_b z_a z_b + 2x_a z_a + 2x_b z_b,$$

and the functions ρ_i which span $\mathscr{H}$ are all the 6 monomials of degree up to 2, i.e.

$$1, \; x_a^2, \; x_b^2, \; x_a x_b, \; x_a, \; x_b.$$

3.2. Shallow network

Let us assume that the NARX model belongs to a RKHS $\mathscr{H}$, i.e. $F \in \mathscr{H}$, and use $\|\cdot\|_{\mathscr{H}}$ to denote the RKHS norm. A kernel-based approach can then be adopted to estimate F. Using a quadratic loss to measure the data fit leads to an estimator known as kernel ridge regression or also regularization network [7]. The regularized NARX is given by

$$\hat{F} = \arg\min_{f \in \mathscr{H}} \sum_{t=1}^{N} (y_t - f(x_t))^2 + \gamma \|f\|_{\mathscr{H}}^2 \tag{4}$$

where the positive scalar γ trades-off adherence to data and the penalty term which typically encodes information on function smoothness. For instance, from the previous discussion around (3), it comes that the polynomial kernel models the unknown function F as a truncated Volterra series. Strictly speaking, such model is called polynomial NARX in the system identification literature, whereas the term Volterra models is typically adopted only for NFIR models, i.e. when x_t contains only past input values, i.e., $u_{t-1} \ldots u_{t-m}$. The induced RKHS norm then penalizes the monomial expansion coefficients leading to a regularized Volterra model, e.g. see [12].

For the moment, we assume that the regularization parameters (which include γ and possibly also other variables present in the kernel) are known. The representer theorem [16, 5] makes the NARX estimate $\hat{F}$ available in closed-form. In particular, let $Y = [y_1, \ldots, y_N]^T$ and $\mathbf{K} \in \mathbb{R}^{N \times N}$ be the kernel matrix with (t,i)-entry given by $\mathbf{K}_{ti} = K(x_t, x_i)$ for t, i ranging over $\{1, \ldots, N\}$. Then, the nonlinear system estimate is sum of the N kernel sections centred on the x_t, i.e.

$$\hat{F}(x) = \sum_{t=1}^{N} \hat{c}_t K(x, x_t), \quad \forall x \in \mathbb{R}^{2m} \tag{5}$$

where the coefficients $\hat{c}_t$ are the components of the vector

$$\hat{c} = (\mathbf{K} + \gamma I_N)^{-1} Y, \tag{6}$$

with I_N the $N \times N$ identity matrix. The expression (5) points out the shallow structure of the network, revealing its one-hidden layer architecture.

The function $\hat{F}$ returned by (4) corresponds to the estimate of the one-step ahead predictor. In principle, it also provides long-term predictions over any desired horizon. In fact, exploiting previously estimated outputs $\hat{y}_t$, they are typically computed as

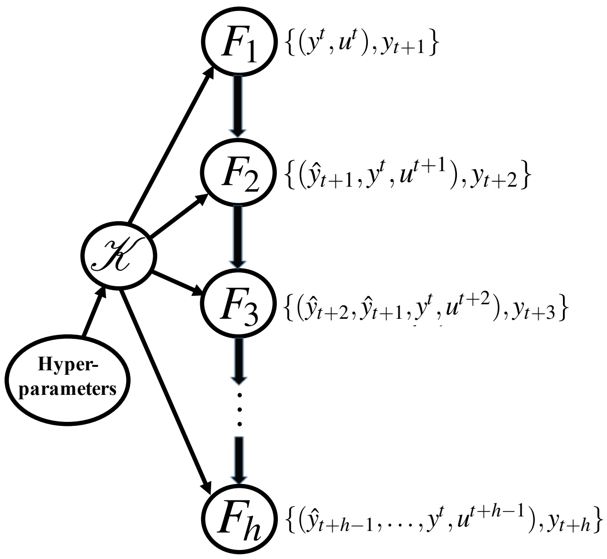
$$\begin{aligned}
\hat{y}_{t+1} &= \hat{F}(y_t, y_{t-1}, y_{t-2}, \ldots, , u_t, u_{t-1}, \ldots) \\
\hat{y}_{t+2} &= \hat{F}(\hat{y}_{t+1}, y_t, y_{t-1}, \ldots, , u_{t+1}, u_t, \ldots) \\
&\vdots \\
\hat{y}_{t+h} &= \hat{F}(\hat{y}_{t+h-1}, \hat{y}_{t+h-2}, \hat{y}_{t+h-3}, \ldots, u_{t+h-1}, u_{t+h-2}, \ldots).
\end{aligned} \tag{7}$$

In the following, we will refer to the predictor in (7) as Iterated Shallow Prediction Network (ISPN).

3.3. Deep prediction network

The proposed predictor, named Deep Prediction Network (DPN), considers a different network w.r.t. ISPN. We instead refine the recursions (7) as follows

$$\begin{aligned}
\hat{y}_{t+1} &= \hat{F}_1(y_t, y_{t-1}, y_{t-2}, \ldots, , u_t, u_{t-1}, \ldots) \\
\hat{y}_{t+2} &= \hat{F}_2(\hat{y}_{t+1}, y_t, y_{t-1}, \ldots, , u_{t+1}, u_t, \ldots) \\
&\vdots \\
\hat{y}_{t+h} &= \hat{F}_h(\hat{y}_{t+h-1}, \hat{y}_{t+h-2}, \hat{y}_{t+h-3}, \ldots, u_{t+h-1}, u_{t+h-2}, \ldots),
\end{aligned} \tag{8}$$



F_1
$\{(y^t, u^t), y_{t+1}\}$
F_2
$\{(\hat{y}_{t+1}, y^t, u^{t+1}), y_{t+2}\}$
$\mathscr{K}$
F_3
$\{(\hat{y}_{t+2}, \hat{y}_{t+1}, y^t, u^{t+2}), y_{t+3}\}$
Hyper-parameters
F_h
$\{(\hat{y}_{t+h-1}, \ldots, y^t, u^{t+h-1}), y_{t+h}\}$

Figure 1: Deep prediction network

hence allowing maps $\hat{F}_i$ to be different each other. Important reasons justify such increased flexibility. While $\hat{F}$ can well perform for one step-ahead prediction, recursions (7) may lead to numerical instabilities. This phenomenon is well documented when using polynomial NARX models, induced e.g. by the kernel (3). To improve the different prediction targets, the rationale behind (8) is to introduce different $\hat{F}_i$ to define the recursions. This appears also well appropriate in the nonlinear setting where (differently from the linear case) minimum variance predictors are not in general the composition of the one-step ahead predictor $\hat{F}$.

How stated in mathematical terms a few lines below, each function that characterizes any layer is assumed to belong to a RKHS. This can make the network especially rich, e.g. able to approximate any function in the space of continuous functions exploiting e.g. universal kernels [34]. Some similarities among the functions F_i, that define the optimal predictors, should however be expected. This can be taken into account by assuming that all of them share the same hyperparameters and, hence, the same kernel model. All the above observations are encoded in the DPN reported in Fig. 1.

Each layer present in the network focuses on a certain prediction horizon and is trained by using the data reported on the right of each node. Here, u^t and y^t are, respectively, the inputs and the outputs collected up to instant t. At the first level the notation F_1 indicates the optimal one-step ahead predictor. It is estimated by exploiting the couples $\{(y^t, u^t), y_{t+1}\}_{t=0}^{N-1}$ through the kernel-based estimator (4). Hence, we let

$$\begin{aligned} x_{t+1} &= [y^t \; u^t], \qquad t = 0, \dots, N-1 \\ Y &= [y_1, \dots, y_N]^T \end{aligned}$$

where here, and in what follows, u^t and y^t are suitably truncated to make the dimension of x_{t+1} equal to $2m$. Then, using the aforementioned representer theorem, one has:

$$\begin{aligned} \hat{F}_1(x) &= \sum_{t=1}^{N} \hat{c}_t K(x, x_t), \quad \hat{c} = (\mathbf{K} + \gamma I_N)^{-1} Y \\ \hat{y}_{t+1}(t) &= \hat{F}_1(x_t), \qquad t = 1, \dots, N, \end{aligned}$$

so that the $\hat{y}_{t+1}$ indicate the one step-ahead predictions. These latter are then propagated to the second layer and the estimate of F_2 is obtained exploiting the couples $\{(\hat{y}_{t+1}, y^t, u^{t+1}), y_{t+2}\}_{t=0}^{N-2}$ and (4). For this purpose, the input locations x_t, the data vector Y and the kernel matrix $\mathbf{K}$ all have to be redefined. In particular, at the second level it holds that

$$\begin{aligned} x_{t+2} &= [\hat{y}_{t+1}, y^t, u^{t+1}], \qquad t = 0, \dots, N-2 \\ Y &= [y_2, \dots, y_N]^T. \end{aligned}$$

Then, the new $(N-1) \times (N-1)$ kernel matrix $\mathbf{K}$ has (t,i)-entry equal to $\mathbf{K}_{ti} = K(x_{t+1}, x_{i+1})$ and one has

$$\begin{aligned} \hat{F}_2(x) &= \sum_{t=2}^{N} \hat{c}_t K(x, x_t), \quad \hat{c} = (\mathbf{K} + \gamma I_{N-1})^{-1} Y \\ \hat{y}_{t+2}(t) &= \hat{F}_2(x_t), \qquad t = 2, \dots, N \end{aligned}$$

so that $\hat{y}_{t+2}$ are the two step-ahead predictions.
The above formulas generalize in an obvious way to obtain all the $\hat{F}_i$. The third layer uses the data sets

$$\begin{aligned} x_{t+3} &= [\hat{y}_{t+2}, \hat{y}_{t+1}, y^t, u^{t+2}], \qquad t = 0, \dots, N-3 \\ Y &= [y_3, \dots, y_N]^T, \end{aligned}$$

the fourth exploits instead

$$\begin{aligned} x_{t+4} &= [\hat{y}_{t+3}, \hat{y}_{t+2}, \hat{y}_{t+1}, y^t, u^{t+3}], \qquad t = 0, \ldots, N-4 \\ Y &= [y_4, \ldots, y_N]^T, \end{aligned}$$

and so on. Then, the data used by the k-th predictor are

$$x_{t+4} = [\hat{y}_{t+k-1}, \ldots, \hat{y}_{t+1}, y^t, u^{t+k-1}], \qquad t = 0, \ldots, N-k \tag{9a}$$

$$Y = [y_k, \ldots, y_N]^T, \tag{9b}$$

The process of sending predictions to form the new data sets (of decreasing size) then continues up to the desired depth h.

3.4. Derivation of ISPN and DPN

Here, we summarize the training procedure needed to derive ISPN and DPN, detailing computational costs and hyperparameters optimization. Hereafter, we will denote the hyperparameters set with θ, which is the vector collecting the regularization parameter γ and the other kernel hyperparameters. For instance, referring to the polynomial kernel in (3), we have $\theta = [\gamma, q]$.

ISPN derivation

The ISPN derivation requires the following steps:

- Hyperparameters estimation: select θ;
- Computation of $\hat{c}$ with (6);

The hyperparameters selection can be done through heuristic approaches, such as cross validation [2], or maximizing the Marginal Likelihood (ML) of the training samples [35, 36]. In this paper, we consider the second strategy. Specifically, we minimized the ML negative log of the training samples $\{(u^t, y^t), y_{t+1}\}_{t=0}^{N-1}$, namely,

$$\hat{\theta} = \text{argmin}_{\theta} \left(\frac{1}{2} Y^T (\mathbf{K} + \gamma I_N)^{-1} Y + \frac{1}{2} \log |\mathbf{K} + \gamma I_N| \right),$$

where matrix $\mathbf{K}$ is the same used in (5). The last optimization problem can be solved relying to any optimization algorithm for non-convex problems. However, experimental evidence shows that also standard gradient descent algorithm with adaptive learning step are effective.

DPN derivation

As discussed before, to consider similarities between the different layers of the DPN we assume that the F_is share the hyperparameters θ, see Figure 1. Then, also with the DPN model, hyperparameters selection is done only one time, adopting the same strategy of ISPN. After optimizing the hyperparameters, the algorithm computes the F_is through the iterative procedure described in the Section 3.3. Each iteration accounts for three steps. In particular, at step k, we have to:

- Build the training dataset following (9);
- Compute the $\hat{c}$ of F_k with (6), defining Y and K in accordance with the dataset build;
- Compute $\hat{y}_t$ with $t = k \ldots N$.

These three steps are iterated h times.

Computational costs

Regarding computational costs, the most expensive step is hyperparameters optimization, which requires the evaluation of the likelihood and its gradient several times. However, this operation must be executed with both ISPN and DPN. From the computational point of view, the difference between ISPN and DPN is that ISPN computes a single vector of coefficients $\hat{c}$, while DPN has h different vectors of coefficients. Assuming that h is significantly smaller than N, the cost to compute each vector is $\mathcal{O}(N^3)$, since it involves a matrix inversion with dimension $N-k$ with $k = 1 \ldots h$. Then, neglecting hyperparameters optimization, the ISPN and DPN derivation cost, respectively, $\mathcal{O}(N^3)$ and $h\mathcal{O}(N^3)$. The last expression show that, compared to ISPN, the DPN derivation is more expensive. However, the feasibility of the derivation depends on the number of samples N, and not on h, so DPN can be applied in the same cases of ISPN. Moreover, it is worth mentioning that these computations are performed offline, an so the gap between the two computational costs is not so significant.

As concerns simulation, both with ISPN and DPN each simulation step requires the evaluation of (5), that costs $\mathcal{O}(N)$. The linear dependence makes such models attractive also for online applications. Finally, it is worth mentioning that, in case that the number of training samples N is too high, the training and evaluation computational cots can be reduced relying on approximate models, see [37] for a complete overview.

4. Numerical experiments

We tested the proposed strategy with artificial data as well as with data coming from real benchmark systems. The predictors compared are:

- **ISPN**: the Iterated Shallow Prediction Network described in Section 3.2. Thus, this is the classical one-step ahead predictor iterated using the predicted in place of the measured output, see (7);
- **DPN**: the Deep Prediction Network, as proposed in Section 3.3 and summarized by (8);
- **MSPNs**: the Multiple Shallow Prediction Networks. MSPNs computes the k-step ahead prediction by means of an ad-hoc predictor $\hat{F}_k$. Each $\hat{F}_k$ is given by a shallow network of the kind described by (5), using the system inputs up to time $t+k-1$ while only the system outputs up to $t-1$ are exploited. This means that the MSPNs prediction is

$$\hat{y}_{t+k} = \hat{F}_k\left([y_{t-1}, y_{t-2}, y_{t-3}, \ldots, u_{t+k-1}, u_{t+k-2}, \ldots]\right).$$

We implemented the three predictors in PyTorch [38], starting from the GPR-PyTorch library[1]. Performances are compared using the FIT, defined as

$$FIT = 100\left(1 - \frac{\left\|Y - \hat{Y}\right\|}{\left\|Y - \bar{Y}\right\|}\right), \tag{10}$$

where Y and $\hat{Y}$ are the system output and the simulated output, while $\bar{Y}$ is the mean of Y. To evaluate the simulation accuracy as a function of the prediction step h we used the norm of the cumulative error, denoted by E_h and defined in the following way:

$$E_h = \left\|Y^h - \hat{Y}^h\right\|, \tag{11}$$

where Y^h and $\hat{Y}^h$ are the vectors collecting the system output and the simulated output up to the prediction step h.

[1] `https://bitbucket.org/AlbertoDallaLibera/gpr-pytorch/admin`

4.1. Simulated pendulum

The first setup considered is a pendulum with dynamics described by the following equation

$$\ddot{y} = \frac{u - b\dot{y} - \frac{1}{2}MLg\sin(y)}{J},$$

where y and u are the pendulum angle and the input torque, while M, L, and J are, respectively, the mass, length, and inertia of the pendulum; g is the gravitational acceleration. We simulated the system using the *odeint* function of the Python package *scipy*. We assumed that noisy measurement of the pendulum position y were collected at 30[Hz]; velocities were not measured. We excited the system with two different realizations of Gaussian noise filtered with a low-pass filter. The cutoff frequency of the low-pass filter was 7.5[Hz]. The standard deviation of the measurement noise was 0.001[rad]. The training and test data sets contain 600 and 1500 samples, respectively. Our aim is to find good long-term predictions by adopting a simple linear kernel with memory $m = 5$, i.e.

$$K(x,z) = \sigma(xz^\top + 1)^p \tag{12}$$

with $p = 1$; σ is a kernel hyperparameter determined, together with γ, through the procedure explained in Section 3.4.

In the first experiment, we compared the simulation performance of ISPN, MSPNs, and DPN through a Monte Carlo experiment composed of 100 experiments. In each experiment, we generated training and test data sets starting from the input trajectories described before, varying the measurement noise's realization. After training the three predictors, we simulated the system ahead for $h = 300$ steps starting from an initial condition randomly picked from the test set. The box-plots in Figure 2 show that DPN outperforms ISPN and MSPNs. MSPNs does not improve the classical approach ISPN, probably due to the high dimensional input and the small number of training samples. DPN exhibits larger variance w.r.t. ISPN (not considering the largely negative fit returned by ISPN), since, in a few experiments, the resulting FIT is quite smaller than in the rest of the runs. However, DPN performs significantly better than ISPN, and the additional variance is acceptable.

In Figure 2, one can see that in one out of the 100 experiments the ISPN FIT is much negative. This result is due to the estimation of an unstable system with a pole outside the unitary circle. To evaluate if DPN can consistently help to find stable predictors when ISPN is unstable, we considered another Monte Carlo experiment with the noise realization fixed to that leading to the unstable

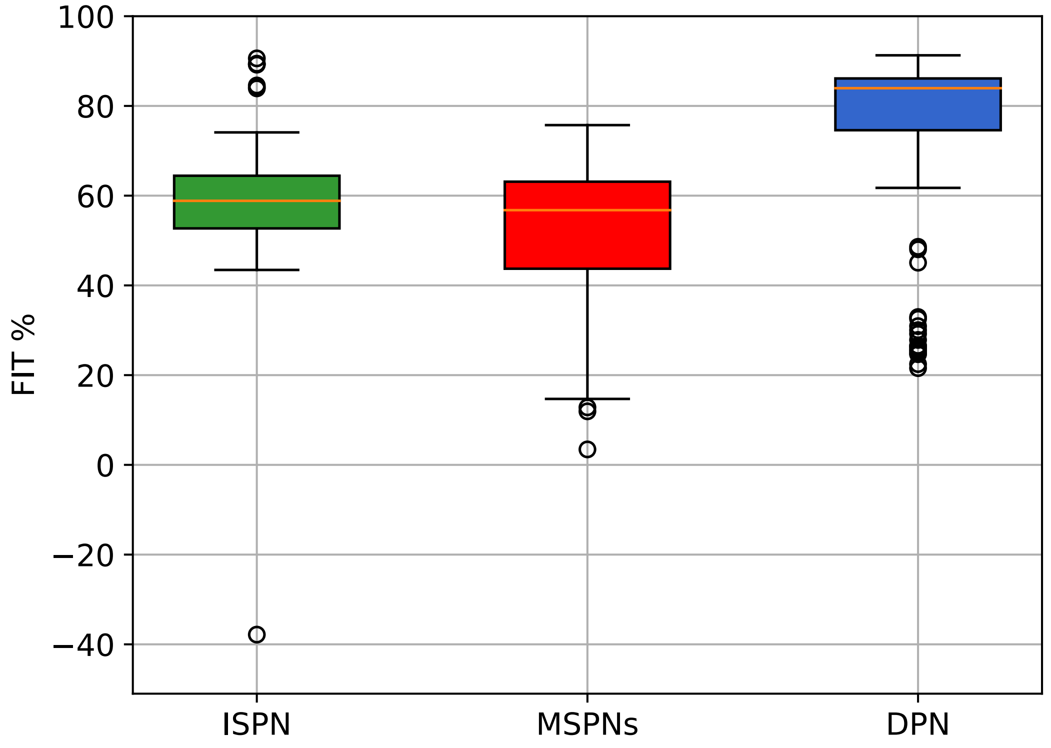


Figure 2: Box-plots of the FITs (10) obtained by ISPN, MSPN and DPN in the pendulum system, varying the noise realization and the test initial condition.

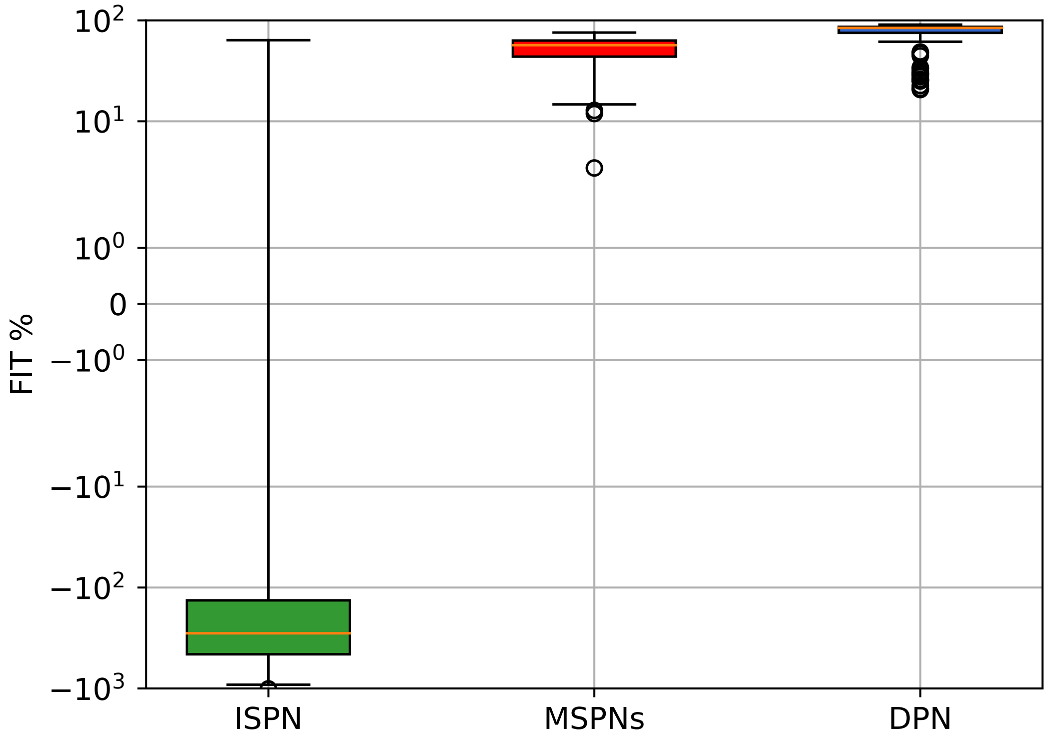


Figure 3: Box-plots of the FITs (10) obtained by ISPN, MSPN and DPN in the pendulum system when ISPN is unstable, varying the test initial condition.

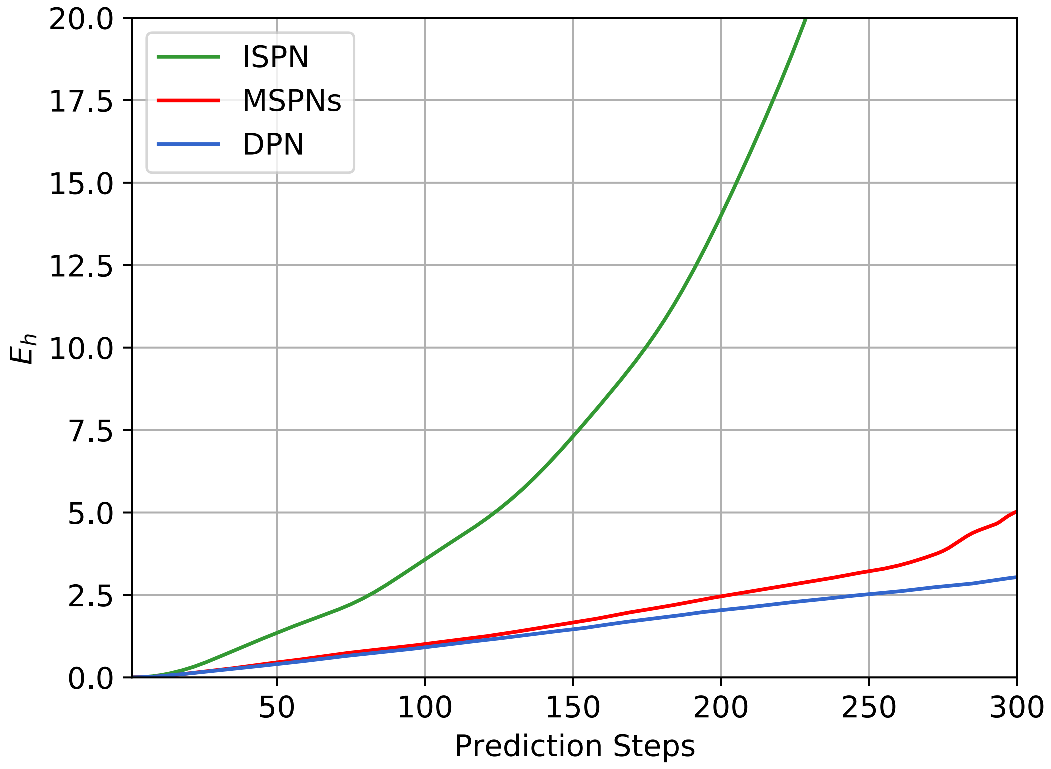


Figure 4: Evolution of the ISPN, MSPN and DPN mean E_s (11) as function of the prediction step obtained during the pendulum experiment when the model returned by ISPN is unstable.

ISPN model, then varying only the system initial conditions. In Figure 3, we reported the box-plots of the obtained FITs. As expected, due to the unstable pole, in almost all the experiment ISPN FIT is negative. In contrast, MSPNs and DPN perform as in the previous experiment, showing their stabilizing features. In Figure 4, we reported the avergae cumulative errors evolution as a function of the prediction step. As expected, with $h = 1$ the three cumulative errors coincide, and they grow with h. Due to the unstable pole, the average E_h of ISPN grows exponentially, while the ones of MSPNs and DPN grow slower. Interestingly, the gap between MSPNs and DPN increases with the prediction step h, confirming the lager accuracy of DPN. This behavior could be due to the fact that MSPNs input dimension augments with h, decreasing generalization.

In Figure 5, we reported the evolution of the noiseless and simulated outputs, computed by ISPN, MSPNs, and DPN, in one of the 100 experiments.

For what the training time is concerned, ISPN training is cheaper than MSPN

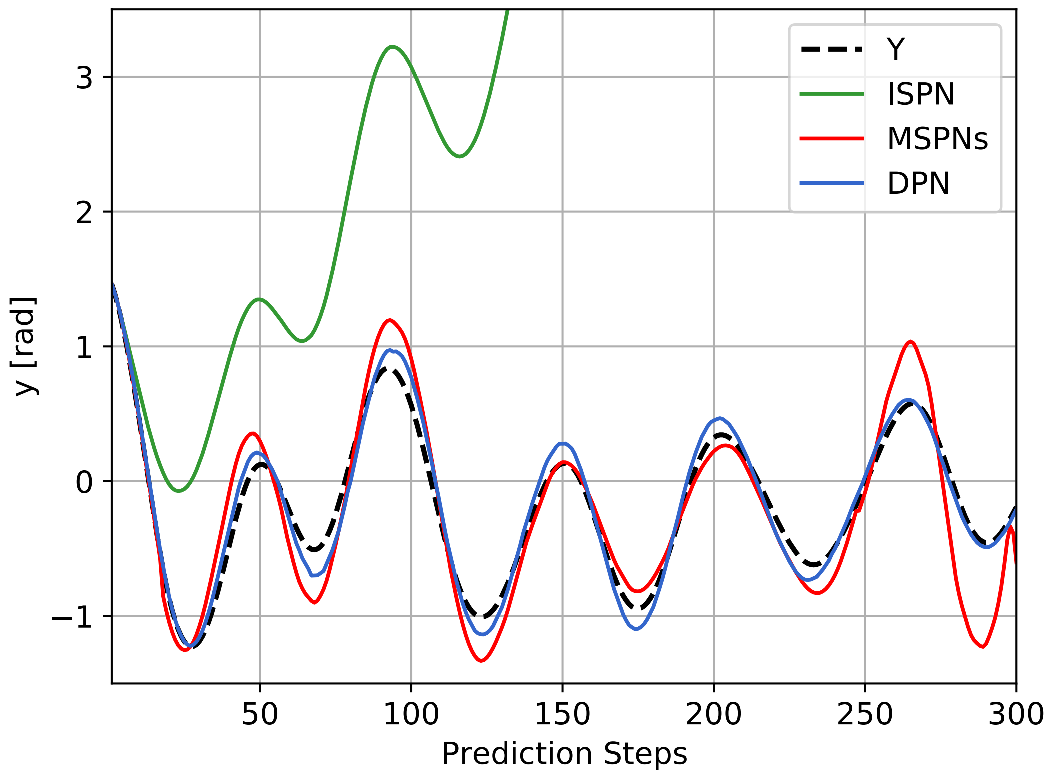


Figure 5: Evolution of the noiseless and simulated output obtained by ISPN, MSPN and DPN in the pendulum system when the model returned by ISPN is unstable.

Algorithm	Training time [min]
ISPN	~ 1
MSPN	~ 150
DPN	~ 2

Table 1: Comparison of the ISPN, MSPN and DPN training time in the pendulum experiment.

and DPN, since it requires a single optimization of the hyperparameters plus the evaluation of (6), which involves a matrix inversion. When compared with ISPN, DPN requires $h-1$ additional matrix inversions, which do not significantly augment the computational burden if the number of training samples is not exceptionally large, as in these examples. On the contrary, MSPN training is much more expensive since it requires the solutions of h hyperparameters optimization problems and h evaluations of (6), one for each prediction layer. In the setup described, the ISPN and DPN training required a few minutes; instead, the MSPNs training needed 2 hours and a half (see Table 1). For this reason, in what follows we limit the comparison to ISPN and DPN.

4.2. Liquid-saturated steam heat exchanger

In this experiment, we considered the heat exchanger dataset [39], publicly available in DaISy database[2]. It contains data from a real process industry system where water is heated by pressurized steam in a copper tube. The system input is the liquid flow rate while the output is the liquid temperature. This data set was recently adopted in [22], where authors highlighted possible instabilities when considering as NARX models a third-degree polynomial. The data set accounts for 4000 samples, collected exiting the system with a random flow rate. The first 100 inputs were sampled from a Gaussian distribution centered on the nominal steady-state speed. The distribution of the next 1800 inputs instead was described by two different beta distributions, one privileges low rates, the other high rates. The remaining inputs were sampled from a uniform distribution. See [39] for details. We used the first 1000 samples for training and the remaining samples for the test. We considered the polynomial kernel in (12), with $m=3$ and $p=3$. As done in [22], the predictors of y_t use as input also u_t, besides the past history of the system input and output; we verified that the use of u_t is important to obtain

[2]`https://homes.esat.kuleuven.be/~smc/daisy/daisydata.html`

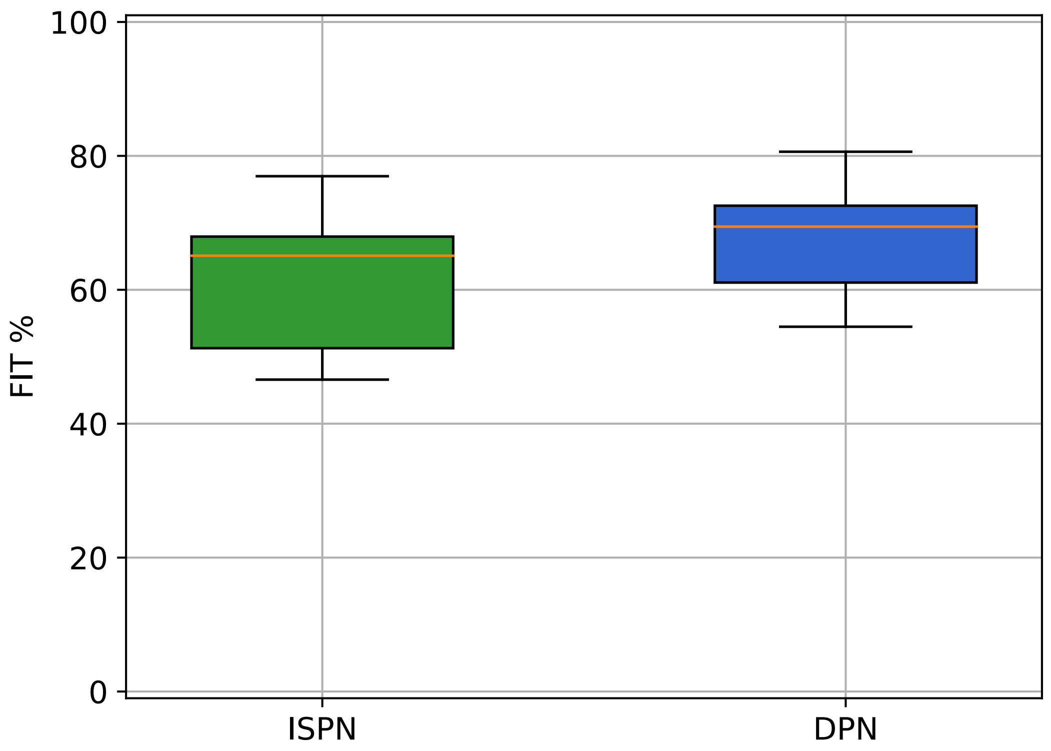


Figure 6: Box-plots of the FITs (10) obtained by ISPN and DPN in the heat exchanger data set for different test initial conditions.

sufficiently accurate one-step ahead predictors since there is no delay between the commanded input and the output variation.

After training ISPN and DPN using the 1000 training samples, we simulated the system for 300 steps, initializing the predictors in 100 different ways by randomly picking the initial conditions from the test set. The FIT box-plots reported in Figure 6 show that ISPN does not diverge in this example: its FITs are positive and larger than 40%. However, DPN does a better job and this reveals its potentialities in improving the prediction capability of ISPN also when this latter returns stable models. Similar considerations hold by considering Figure 7, where we reported the average E_h as a function of the prediction step.

4.3. Silverbox system

We tested the ISPN and DPN in another real system, the Silverbox [40]. This is an electrical system similar to a mass-spring-damper system. The spring exhibits

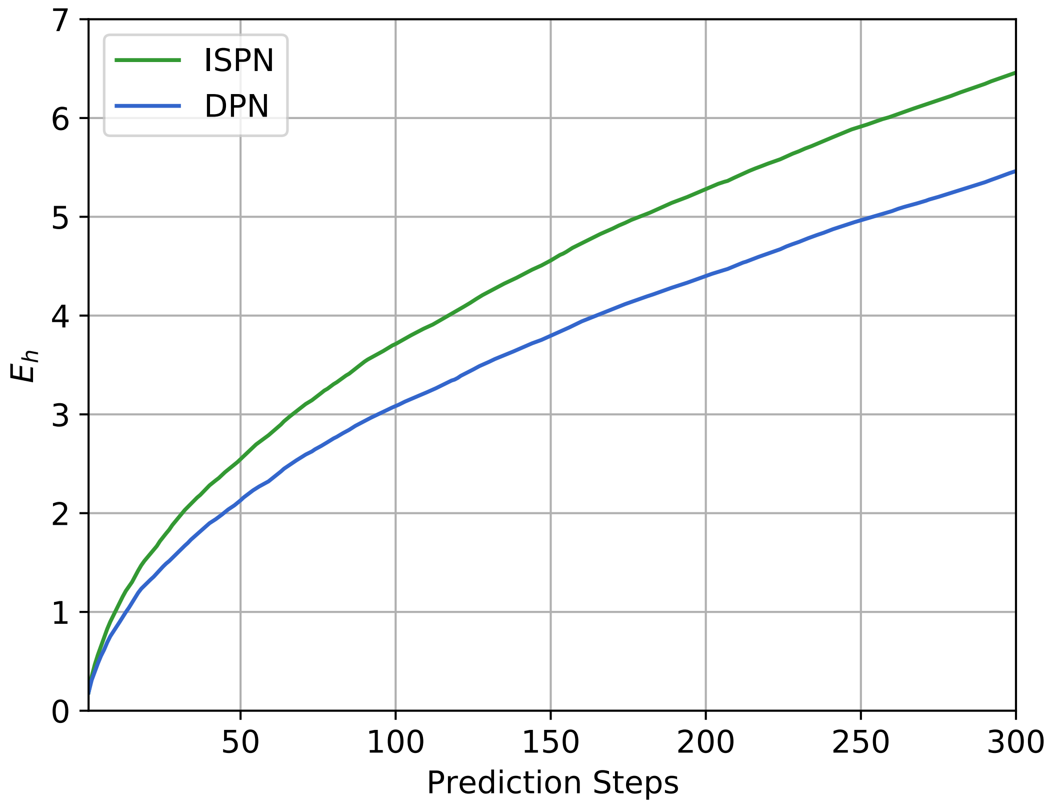


Figure 7: Evolution of the ISPN and DPN mean E_s (11) as function of the prediction step obtained using the heat exchanger data set.

nonlinear behaviors, described by the following differential equation

$$m\ddot{y}(t) + d\dot{y}(t) + k_1 y(t) + k_3 y^3(t) = u(t),$$

where y and u are, respectively, the mass displacement and the input force applied to the mass, while d, k_1, and k_3 are the damper and spring parameters. We considered a polynomial kernel with degree 3 and $m = 4$. Differently from the previous examples, we adopted a richer parametrization of the polynomial kernel, defined by the following equation,

$$K(x,z) = (x\Sigma z^\top + \sigma)^p,$$

where the hyperparameter $\Sigma > 0$ is a diagonal matrix which allows a more fine regularization w.r.t. (12). The data set is publicly available[3]. The training dataset used in this experiment is given by the firsts 4000 samples of the original training data set, which is composed of 87000 samples collected by feeding the system with an odd random phase multisine signal. Instead, the test data set accounts for all the available test samples, which are approximately 40000 samples collected after exciting the system with filtered Gaussian noise. The maximum amplitude of the test signal grows linearly over time, from zero to the higher permissible value. Instead, to stress generalization, the maximum amplitude of the training signal is limited to a smaller value. To compare ISPN and DPN with another standard approach used in nonlinear identification we also tested an adaptive neuro-fuzzy inference system (ANFIS)[41], exploiting the implementation of the MATLAB nonlinear identification toolbox. We trained the ANFIS using the same dataset adopted with ISPN and DPN, selecting hyperparameters by cross validation

As done in the previous experiments, after training ISPN and DPN we initialized the predictors by using 100 different initial conditions randomly picked from the test data set, and we simulated the system for 1000 steps. ISPN returned an unstable model and FIT computation in all the 100 Monte Carlo runs was not possible due to numerical issues. Instead, DPN can simulate the system accurately. Figure 8 reports the DPN and ISPN FITs box-plot. Except for a few cases, DPN FITs are over 97%, and in any case better than ANFIS FITs. In Figure 9 we plotted the mean E_h evolution as a function of the prediction step. The DPN mean E_h does not grow significantly with E_h, while, due to the instability, ISPN mean E_h grows rapidly. Figure 10 compares the evolution of the measured and simulated

[3]`http://www.nonlinearbenchmark.org/`

Algorithm	Training samples	RMSE sim [mV]
DPN (proposed approach)	4000	0.65
ANFIS (in this work)	4000	11.49
PNLSS [42]	87000	0.26
Direct Identification [43]	87000	0.96
State-space Encoder [44]	87000	1.4
Best Linear Approximation [45]	87000	13.5

Table 2: Comparison with the previous simulation results obtained on the Silverbox system.

outputs. The ISPN simulation diverges rapidly, while DPN simulation is adherent to the system output.

Finally, in Table 2 we compared the mean of the DPN and ANFIS Root Mean Squared Errors (RMSEs) with some of the most significant results previously obtained on the silverbox dataset. The DPN mean RMSE is close to the best results obtained in this benchmark, despite we used only 4000 training samples, instead of 87000. This fact further confirms the effectiveness of the proposed solution.

5. Conclusions

Long-term predictions and simulations rely on compositional maps. Hence, they should represent an important topic for system identification, motivating the development of new computational structures for dynamic systems. In this setting, our deep prediction network contains layers which communicate each other and are associated with different k-step ahead predictors. Compared with a shallow network architecture, this introduces a larger number of degrees of freedom in the estimator. They are used to favour system stability, a feature especially critical in nonlinear system identification. Regularization is also introduced in the estimation process: network complexity is controlled by a few hyperparameters contained in the common kernel each layer is equipped with. The use of simulated and real benchmarks problems show that such deep network may outperform a shallow network in terms of prediction performance. DPN application might be particularly convenient in all the applications where ISPN simulation may not be stable.

Developments of this work along many directions are possible. For instance, the use of different hyperparameter estimation strategies, more tailored to the different layers, could lead to improved results in some circumstances. Open issues could regard also the possibility to avoid to update the predictors at any network

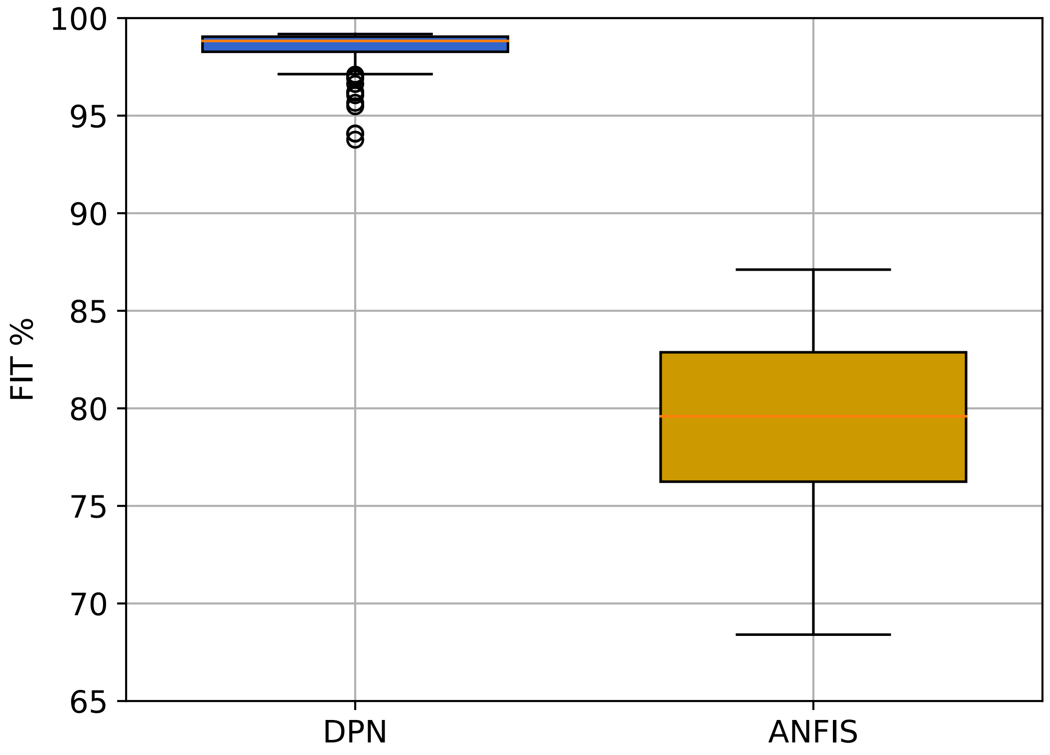


Figure 8: Box-plots of the FITs (10) obtained by DPN in the silverbox data set varying the test initial condition.

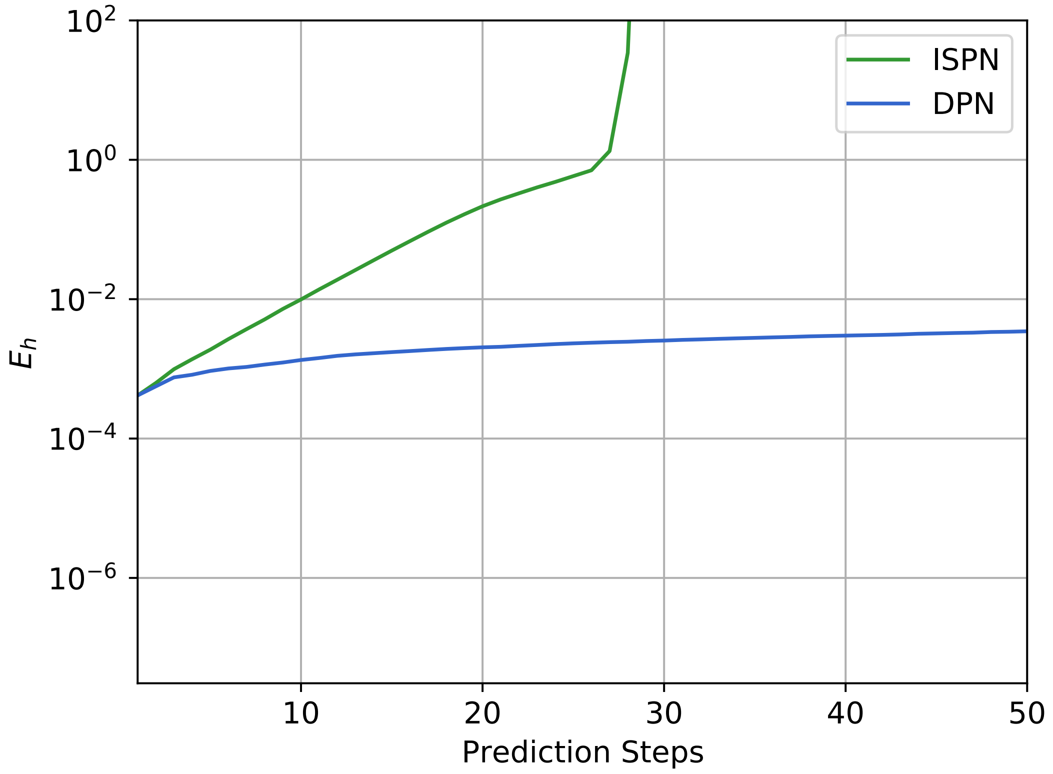


Figure 9: Evolution of the ISPN and DPN mean E_s (11) as a function of the prediction step obtained in the silverbox data set.

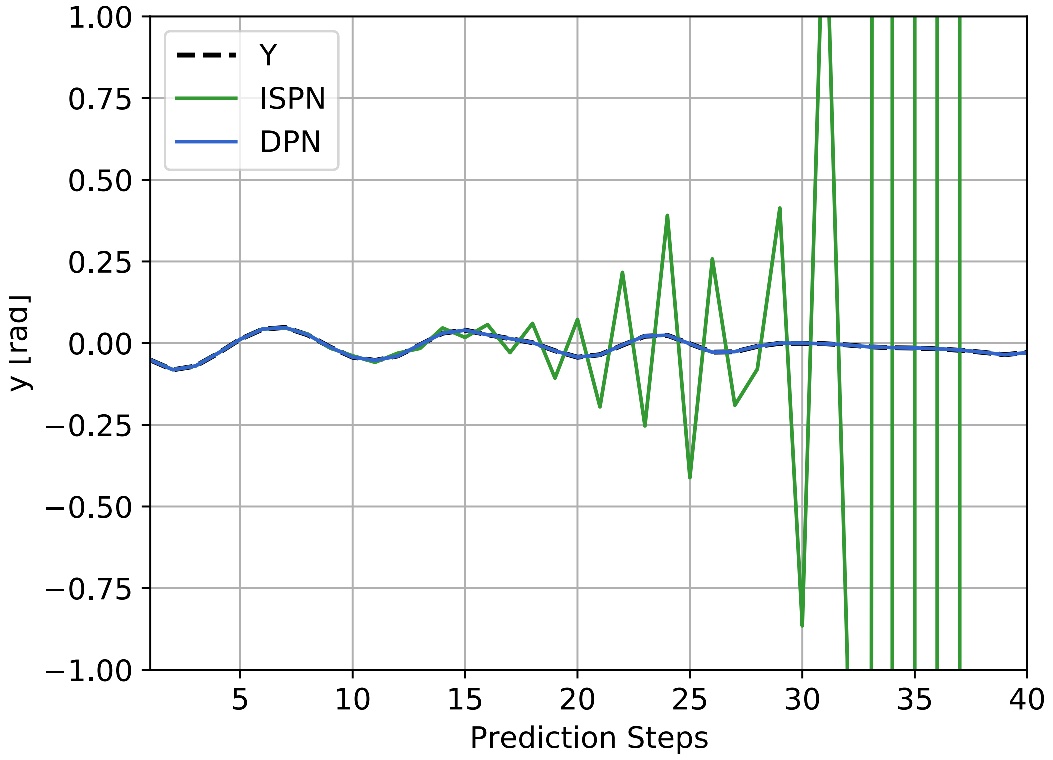


Figure 10: Evolution of the noiseless and simulated outputs obtained by ISPN and DPN in the silverbox data set.

level to better control the variance of the hierarchical structure. Developments could also include the design of even more complex architectures, with layers including other important information on system dynamics, as well as theoretical results concerning statistical consistency and learning rate of this new kind of networks for prediction.

Acknowledgements

This work has been in part supported by the H2020 FETPROACT-01-2019 project A BIONIC INVISIBLE PANCREAS TO FORGET DIABETES - FORGETDIABETES: 951933

References

[1] J. Schoukens, L. Ljung, Nonlinear system identification – a user-oriented roadmap, IEEE Control Systems Magazine 39 (6) (2019) 28–99.

[2] T. J. Hastie, R. J. Tibshirani, J. Friedman, The Elements of Statistical Learning. Data Mining, Inference and Prediction, Springer, Canada, 2001.

[3] L. Ljung, System Identification - Theory for the User, 2nd Edition, Prentice-Hall, Upper Saddle River, N.J., 1999.

[4] T. Söderström, P. Stoica, System Identification, Prentice-Hall Int., London, 1989.

[5] B. Schölkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, (Adaptive Computation and Machine Learning), MIT Press, 2001.

[6] M. Espinoza, J. A. K. Suykens, B. De Moor, Kernel based partially linear models and nonlinear identification, IEEE Trans. on Automatic Control 50 (10) (2005) 1602–1606.

[7] T. Evgeniou, M. Pontil, T. Poggio, Regularization networks and support vector machines, Advances in Computational Mathematics 13 (2000) 1–50.

[8] T. Chen, T. Ardeshiri, F. Carli, A. Chiuso, L. Ljung, G. Pillonetto, Maximum entropy properties of discrete-time first-order stable spline kernel, Automatica 66 (2016) 34 – 38.

[9] B. Bell, G. Pillonetto, Estimating parameters and stochastic functions of one variable using nonlinear measurement models, Inverse Problems 20 (3) (2004) 627.

[10] G. Pillonetto, M. H. Quang, A. Chiuso, A new kernel-based approach for nonlinear system identification, IEEE Transactions on Automatic Control 56 (12) (2011) 2825–2840.

[11] G. Pillonetto, System identification using kernel-based regularization: New insights on stability and consistency issues, Automatica 93 (2018) 321 – 332. doi:https://doi.org/10.1016/j.automatica.2018.03.065.
URL `http://www.sciencedirect.com/science/article/pii/S0005109818301602`

[12] M. Franz, B. Schölkopf, A unifying view of Wiener and Volterra theory and polynomial kernel regression, Neural Computation 18 (2006) 3097–3118.

[13] M. Schetzen, The Volterra and Wiener Theories of Nonlinear Systems, Wiley, New York, 1980.

[14] G. Birpoutsoukis, A. Marconato, J. Lataire, J. Schoukens, Regularized nonparametric volterra kernel estimation, Automatica 82 (2017) 324 – 327.

[15] A. Dalla Libera, R. Carli, G. Pillonetto, A novel multiplicative polynomial kernel for volterra series identification, in: Proceedings of the 21st IFAC World Congress, Berlin, Germany, 2020.

[16] G. Pillonetto, F. Dinuzzo, T. Chen, G. De Nicolao, L. Ljung, Kernel methods in system identification, machine learning and function estimation: A survey, Automatica 50 (March 2014).

[17] G. Pillonetto, A new kernel-based approach to hybrid system identification, Automatica 70 (2016) 21 – 31.

[18] F. Lindsten, T. Schön, M. Jordan, Bayesian semiparametric Wiener system identification, Automatica 49 (2013) 2053–2063.

[19] R. Risuleo, F. Lindsten, H. Hjalmarsson, Bayesian nonparametric identification of Wiener systems, Automatica 108 (2019) 108480.

[20] L. Ljung, Perspectives on system identification, Annual Rev. Control 34 (2010) 1–12.

[21] J. Umenberger, I. Manchester, Specialized interior-point algorithm for stable nonlinear system identification, IEEE Transactions on Automatic Control 64 (6) (2019) 2442–2456.

[22] J. Umenberger, I. Manchester, Convex bounds for equation error in stable nonlinear identification, IEEE Control Systems Letters 3 (1) (2019) 73–78.

[23] W. Lohmiller, J. Slotine, On contraction analysis for nonlinear systems, Automatica 34 (1998) 683–696.

[24] E. Sontag, Smooth stabilization implies coprime factorization, IEEE Transactions on Automatic Control 34 (4) (1989) 435–443.

[25] Z. Jing, Y. Wuang, Input-to-state stability for discrete-time nonlinear systems, Automatica 37 (6) (2001) 857 – 869.

[26] B. Wahlberg, L. Ljung, Design variables for bias distribution in transfer function estimation, IEEE Transactions on Automatic Control 31 (2) (1986) 134–144.

[27] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.

[28] P. Bartlett, P. Long, G. Lugosi, A. Tsigler, Benign overfitting in linear regression, PNAS 117 (2020) 30063–30070.

[29] T. Poggio, A. Banburski, Q. Liao, Theoretical issues in deep networks, PNAS 117 (2020) 30039–30045.

[30] L. Ljung, C. Andersson, K. Tiels, T. Schon, Deep learning and system identification, in: Proceedings of the IFAC World Congress, 2020.

[31] W. Liu, Z. Wang, X. Liu, N. Zeng, Y. Liu, F. E. Alsaadi, A survey of deep neural network architectures and their applications, Neurocomputing 234 (2017) 11–26. doi:https://doi.org/10.1016/j.neucom.2016.12.038.
URL `https://www.sciencedirect.com/science/article/pii/S0925231216315533`

[32] N. Aronszajn, Theory of reproducing kernels, Trans. of the American Mathematical Society 68 (1950) 337–404.

[33] T. Poggio, On optimal nonlinear associative recall, Biological Cybernetics 19 (4) (1975) 201–209.

[34] C. A. Micchelli, Y. Xu, H. Zhang, Universal kernels, J. of Machine Learning Research 7 (2006) 2651–2667.

[35] C. Rasmussen, C. Williams, Gaussian Processes for Machine Learning, The MIT Press, 2006.

[36] G. Pillonetto, A. Chiuso, Tuning complexity in regularized kernel-based regression and linear system identification: The robustness of the marginal likelihood estimator, Automatica 58 (2015) 106–117.

[37] J. Quiñonero-Candela, C. E. Rasmussen, A unifying view of sparse approximate gaussian process regression, Journal of Machine Learning Research 6 (65) (2005) 1939–1959.
URL `http://jmlr.org/papers/v6/quinonero-candela05a.html`

[38] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch (2017).

[39] S. Bittanti, L. Piroddi, Nonlinear identification and control of a heat exchanger: A neural network approach, Journal of the Franklin Institute 334 (1) (1997) 135–153. doi:https://doi.org/10.1016/S0016-0032(96)00059-2.
URL `https://www.sciencedirect.com/science/article/pii/S0016003296000592`

[40] T. Wigren, J. Schoukens, Three free data sets for development and benchmarking in nonlinear system identification, in: 2013 European Control Conference (ECC), 2013, pp. 2933–2938. doi:10.23919/ECC.2013.6669201.

[41] J.-S. R. Jang, Fuzzy modeling using generalized neural networks and kalman filter algorithm, in: Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 2, AAAI'91, AAAI Press, 1991, p. 762–767.

[42] J. Paduart, L. Lauwers, J. Swevers, K. Smolders, J. Schoukens, R. Pintelon, Identification of nonlinear systems using polynomial nonlinear state space models, Automatica 46 (4) (2010) 647–656. doi:https://doi.org/10.1016/j.automatica.2010.01.001.
URL `https://www.sciencedirect.com/science/article/pii/S000510981000021X`

[43] H. Hjalmarsson, J. Schoukens, On direct identification of physical parameters in non-linear models, IFAC Proceedings Volumes 37 (13) (2004) 375–380, 6th IFAC Symposium on Nonlinear Control Systems 2004 (NOLCOS 2004), Stuttgart, Germany, 1-3 September, 2004. doi:https://doi.org/10.1016/S1474-6670(17)31252-1.
URL `https://www.sciencedirect.com/science/article/pii/S1474667017312521`

[44] G. Beintema, R. Toth, M. Schoukens, Nonlinear state-space identification using deep encoder networks, in: Proceedings of the 3rd Conference on Learning for Dynamics and Control, Vol. 144 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 241–250.
URL `http://proceedings.mlr.press/v144/beintema21a.html`

[45] A. Marconato, J. Sjöberg, J. Suykens, J. Schoukens, Identification of the silverbox benchmark using nonlinear state-space models, IFAC Proceedings Volumes 45 (16) (2012) 632–637, 16th IFAC Symposium on System Identification. doi:https://doi.org/10.3182/20120711-3-BE-2027.00135.
URL `https://www.sciencedirect.com/science/article/pii/S147466701538023X`

Credit author statement
Gianluigi Pillonetto: Conceptualization, Methodology, Writing
Alberto Dalla Libera: Methodology, Software, Writing