Learning virtual sensors for estimating the scheduling signal of parameter-varying systems

Daniele Masti[⋆], Daniele Bernardini[†], Alberto Bemporad[⋆]

Abstract— **We propose a novel data-driven virtual sensor architecture to reconstruct an unmeasurable scheduling signal of a parameter-varying system from input/output measurements. The key idea is to train and feed an Artificial Neural Network (ANN) with input/output measurements and with data generated by processing such measurements through a bank of linear observers. Special attention is paid to the design of both the ANN and the feature extraction mechanism to keep the architecture as lightweight as possible, so that the resulting virtual sensor can be easily implemented in embedded hardware platforms. As a special case, the proposed virtual sensor can be used for hidden mode reconstruction of switched linear systems. Applications of the proposed approach are geared towards fault detection and isolation, predictive maintenance, and gain-scheduling control.**

I. Introduction

Most real-world processes exhibit complex nonlinear dynamics that are difficult to model, not only because of nonlinear interactions between input and output variables, but also for the presence of time-varying signals that change the way they interact over time. A typical instance is the case of systems subject to wear of components, in which the dynamics slowly drifts from a nominal behavior to an aged one, or even of systems subject to faults, in which the dynamics suddenly changes. Unmeasured disturbances also fall in this category, for example unknown time-varying transport delays, ambient conditions, etc.

A parameter-varying model is a dynamical model Σ that depends on a vector ρ of parameters, $\rho \in \mathbb{R}^S$,

$$\Sigma := \begin{cases} x_{k+1} = f(x_k, u_k, \rho_k) \\ y_k = g(x_k, \rho_k) \end{cases} \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ is the state vector, $y_k \in \mathbb{R}^{n_y}$ the output vector, $u_k \in \mathbb{R}^{n_u}$ the input vector, $f : \mathbb{R}^{n_x+n_u+S} \to \mathbb{R}^{n_x}$, and $g : \mathbb{R}^{n_x+S} \to \mathbb{R}^{n_y}$.

Special cases widely studied in the literature are linear parameter-varying (LPV) models [1], in which f, g are linear functions of x_k, u_k, and switched affine systems [2], in which ρ_k only assumes a finite set of values.

Estimating the vector ρ_k of parameters from input/output data can be useful for several reasons. In predictive maintenance and anomaly/fault detection and isolation [3], [4], the value of ρ_k can be used to detect the occurrence of a fault, when ρ_k deviates from a nominal range that is typical of correct operation of the process, and to isolate the fault, by classifying the type of fault based on the value of ρ_k. In gain-scheduling control, ρ_k is used to decide the control law to apply at each given time instant. For this reason, vector ρ_k is often referred to as the *scheduling signal* [5]. In case a LPV model scheduled with ρ_k is available, either by white-box modeling or by system identification for LPV systems [6] or for switched affine systems [7], [8], the same dependence on ρ_k can be applied for scheduling the controller.

Depending on the specific application, solutions have been proposed to estimate ρ_k from input/output data. In fault detection, one is mostly interested in knowing which fault model better describes the system during operations. Approaches like unknown input observers [9] have been proven very effective for this task. Alternative approaches include piecewise affine regression [10], [11] and moving horizon estimation based on hybrid dynamical models [12], [13]. When the goal is instead to recover a signal that corresponds to some unmeasured dynamical quantities of the process, procedures to directly synthesize observers without going through the identification of a dynamical model first were proposed, also known as *virtual sensors* [14]–[16]. Robust estimation schemes were also investigated in the context of descriptor-LPV systems in [17]. All the above approaches to synthesize virtual sensors for ρ_k require experimental training data in which ρ_k is available from measurements, which is the typical assumption when training virtual sensors.

In this paper we present an approach for learning virtual sensors from data based on a lightweight artificial neural network (ANN). The proposed approach consists of three main steps:

1. build a finite set of linear time-invariant (LTI) models that roughly cover the behavior of the system for the entire spectrum of values of ρ_k;
2. design a set of linear observers based on such models;
3. train an ANN that maps the output of the observers (such as state and output estimates) and input/output signals into an estimate $\hat{\rho}_k$ of ρ_k.

Although the underlying dynamics may be a nonlinear LPV one, the use of linear observers is useful for creating extra inputs to the ANN in a simple way that depend on the entire history of past measurements.

The paper is organized as follows. In Section II we introduce the proposed virtual sensor architecture and training algorithms. In Section III we study the behavior of the synthesized virtual sensor numerically on a simple LPV problem, analyzing the effect of the hyper-parameters of the strategy. Finally, we draw some conclusions in Section IV.

[⋆]IMT School for Advanced Studies Lucca, Italy. E-mail: {daniele.masti,alberto.bemporad}@imtlucca.it.
[†]ODYS S.r.l., Italy. E-mail: daniele.bernardini@odys.it.

II. VIRTUAL SENSOR ARCHITECTURE

As common to direct virtual sensor synthesis procedures, in our approach we do not assume that the parameter-varying model f, g in (1) is known, nor that x_k is measurable. Our working hypothesis is that u_k, y_k can be measured both during training/testing *and* online operations, while ρ_k can be measured only during training/testing. We also suppose that the dynamics of the process is heavily dependent on ρ_k. For example, the value of ρ_k may be determining whether the process is working properly or is in a physical malfunctioning situation.

Let $D : \{u_k, y_k, \rho_k\},\ \ k = 1, \ldots, K$ be the dataset acquired (off line) from the process. Our approach to synthesize a virtual sensor for estimating ρ_k on line from measurements of u_k, y_k is the following:

1. Identify a local linear model Σ_k^γ at each time k, $k = k_1, \ldots, K$, $k_1 \geq 1$, where Σ_k^γ is parameterized by a vector γ_k of model coefficients, $\gamma_k \in \mathbb{R}^{n_\gamma}$;
2. Use unsupervised learning techniques to partition the set $\mathcal{F} = \{\left[\begin{smallmatrix}\rho_{k_1}\\ \gamma_{k_1}\end{smallmatrix}\right], \ldots, \left[\begin{smallmatrix}\rho_K\\ \gamma_K\end{smallmatrix}\right]\}$ of parameter and model coefficient into N clusters, each one characterized by a corresponding set of indices $I_1, \ldots, I_N$, $\cup_{i=1}^N I_i = \{k_1, \ldots, K\}$, $I_i \cap I_j = \emptyset$, $\forall i, j = 1, \ldots, N$, $i \neq j$;
3. Identify N linear time-invariant (LTI) models Σ_i based on input/output data u_k, y_k, $k \in I_i$, $i = 1, \ldots, N$;
4. Design a Luenberger observer L_i for each model Σ_i;
5. Process the dataset D by running the N observers to generate state estimates $\hat{x}_k^i$, $k \in [k_1, K]$;
6. Train an artificial neural network (ANN) on the resulting augmented dataset to learn a relation between ρ_k and u_k, y_k, and past ℓ values of $\hat{x}_k^1, \ldots, \hat{x}_k^N$.

After the virtual sensor is synthesized, the N observers are run in parallel on line and u_k, y_k, $\hat{x}_{k-i}^j$ ($j = 1, \ldots, N$, $i = 1, \ldots, \ell$) fed to the virtual sensor to estimate ρ_k.

A. LTI model identification

Our goal is to construct a set of local LTI models

$$\Sigma_j := \left\{ \begin{array}{rcl} x_{k+1}^j & = & A_j x_k^j + B_j u_k \\ y_k^j & = & C_j x_k^j,\ j = 1, \ldots, N \end{array} \right. \tag{2}$$

with $x_k^j \in \mathbb{R}^{\bar{n}_x}$ and with matrices A_j, B_j, C_j of appropriate dimensions. The idea is that each of such models roughly approximates the behavior of the real process Σ in (1) for a certain range of the scheduling signal ρ_k, and will be used to design the corresponding observer.

In order to define the subset I_j of datapoints used to identify model Σ_j, we first adopt recursive linear system identification techniques to build a set $\{\Sigma_{k_1}^\gamma, \ldots, \Sigma_K^\gamma\}$ of local linear models describing the behavior of the process at each time step k, each one characterized by a parameter vector γ_k.

Designing a time-varying linear observer based on such models to generate the state estimate $\hat{x}_k$ on line would require estimating the vector γ_k of parameters on line too. In order to avoid on-line recursive identification, we only use the local model coefficient vectors γ_k collected offline, together with the corresponding ρ_k, to partition the dataset D into N clusters $I_1, \ldots, I_N$. The underlying idea is that samples (u_k, y_k), $k \in I_j$, associated with similar values of ρ_k and similar values of γ_k contribute to identify the LTI model Σ_j.

In this work we employ an online ARX estimator based on Kalman filtering [18] to estimate the model coefficients γ_k and the K-means algorithm [19] for clustering the dataset D. Moreover, for simplicity, we do not cluster with respect to $\left[\begin{smallmatrix}\rho_k\\ \gamma_k\end{smallmatrix}\right]$ but only with respect to γ_k, and rather than identifying one ARX model per cluster we consider as model Σ_j a state-space model in minimal realization whose transfer function coefficients are the coordinates of the centroid in the γ-space of each cluster identified by the index set I_j.

B. Observer design

For each model Σ_j we need to design an observer providing an estimate $\hat{x}_k^j$ of the state x_j. To be more general, we denote by $i_k^j \in R^v$ the information vector generated by the observer, where i_k^j could be just $\hat{x}_k^j$ or also contain other information, such as related to the covariance matrix of the state-estimation error if linear time-varying Kalman filters were used as state observers, or the covariance of the output estimation error.

When designing the observers, we need to take into account the trade-off between robustness, speed of convergence, and computational burden. An observer that is highly sensitive to changes of local behavior with respect to model Σ_j could quickly decay in performance even for small changes of the scheduling signal ρ_k. Hence, inferring the value of ρ_k from i_k^j would be rather hard. On the other hand, an observer designed for robustness may suffer from the opposite problem and be weakly sensitive to variations of the scheduling signal ρ_k. In both cases, the information signal i_k^j computed by the observer would not be informative enough to recover the scheduling signal ρ_k.

The computational burden introduced by the observers also needs to be considered. As we need to run the full set of N observers in parallel at runtime, techniques like particle filters might results too expensive from a computation viewpoint. For this reason, in this paper we use simple LTI Luenberger observers [20]

$$\left\{ \begin{array}{rcl} \hat{x}_{k+1}^j & = & (A_j - L_j C_j)\hat{x}_k^j + B_j u_k + L_j y_k \\ \hat{y}_k^j & = & C_j \hat{x}_k^j \end{array} \right. \tag{3}$$

where A_j, B_j, C_j are the matrices of the corresponding model Σ_j and L_j is the observer gain matrix. For the observer in (3) the information vector i_k^j corresponds to the current state estimate $\hat{x}_k^j$.

After the N observers are synthesized, we process the dataset D to generate the information vectors i_k^j, $k \in [k_1, K]$.

C. Learning problem and dimension reduction

Let us now consider the augmented dataset $D_{\text{aug}} := \{i_k^1, \ldots, i_k^N, u_k, y_k, \rho_k\}$, $k = k_1, \ldots, K$. Our aim is to train an artificial neural network (ANN) $f_\theta : \mathbb{R}^{\ell N v + n_u + n_y} \to \mathbb{R}^S$ using the dataset D_{aug} so that

$$\hat{\rho}_k = f_\theta(i_k^1, \ldots, i_{k-\ell}^1, \ldots, i_k^N, \ldots, i_{k-\ell}^N, u_k, y_k) \quad (4)$$

is a good estimate of ρ_k. To this end, we consider the minimization of a loss function $L : \mathbb{R}^{2S} \to \mathbb{R}$ measuring the distance between the measured value ρ_k and its reconstructed value, that is we set

$$\theta^\star = \arg\min_\theta \sum_{k=k_1+\ell}^{K} L(\rho_k, f_\theta(i_k^1, \ldots, i_{k-\ell}^N, u_k, y_k)) \quad (5)$$

In order to embed possible a-priori information in the architecture f about the relation between ρ_k and $\{i_k^1, \ldots, i_{k-\ell}^N, u_k, y_k\}$, we define the following structure for f_θ

$$\hat{\rho}_k = g_\theta(e(i_k^1, i_{k-l}^1, \ldots, i_k^N, \ldots, i_{k-l}^N, u_k, y_k)) \quad (6)$$

where $e : \mathbb{R}^{\ell N v + n_u + n_y} \to \mathbb{R}^{n_I}$ is a pre-assigned function defining the vector $\mathcal{I}_k = e(i_k^1, i_{k-l}^1, \ldots, i_k^N, \ldots, i_{k-l}^N, u_k, y_k)$ of features, and $g : \mathbb{R}^{n_I} \to \mathbb{R}^S$ the function to learn from data. The role of e is to compress the raw data i_k^1, i_{k-l}^1, ..., i_k^N, ..., i_{k-l}^N, u_k, y_k, embedding all the needed pre-processing operations on the data such as normalization and rescaling. Albeit many function approximation approaches have demonstrated the capability of learning feature-extraction maps (see for example [21]), leaving e as a complete degree of freedom (that is to directly learn f_θ instead of g_θ) would require more powerful machinery and typically a larger number n_θ of parameters, that may result in a virtual observer with computation and memory requirements that not compatible with embedded applications. Note that the full parameterization of f_θ is retained by choosing $e(\mathcal{I}_k) = \mathcal{I}_k = \{i_k^1, i_{k-l}^1, \ldots, i_k^N, \ldots, i_{k-l}^N, u_k, y_k\}$.

In this paper we propose a high-compression feature extraction function e that, while reducing the number of inputs to the neural network g_θ, does not degrade fitting performance significantly with respect to a full parameterization f_θ. The key idea is to consider, besides the current input and output samples u_k, y_k, the sum ν_k^i of the squares of the residuals $y_k - \hat{y}_k^j$ over a window of past ℓ data

$$\nu_k^i = \sum_{r=k-\ell}^{k} (y_r^i - \hat{y}_r^i)^2 \quad (7)$$

In this way it is possible to reduce the feature vector $\mathcal{I}_k \in \mathbb{R}^{n_I}$ to

$$I_k = \{\nu_k^1, \ldots, \nu_k^N, u_k, y_k\} \quad (8)$$

which only has $n_I = (N+1)n_y + n_u$ components, so that the input to the ANN g_θ, and therefore the ANN g_θ itself, can get very compact. Finally, before feeding I_k to the ANN, each component of I_k is normalized by subtracting the average of I_k observed on the training dataset D_{aug} and scaled by a factor that provides unit variance on the same dataset.

Note that ℓ is a hyper-parameter of our approach. The value of ℓ must be chosen carefully: if it is too small the time window of past output prediction errors may not be long for a slow observer. On the other hand, if ℓ is too large the virtual sensor may become excessively slow in detecting changes of ρ_k.

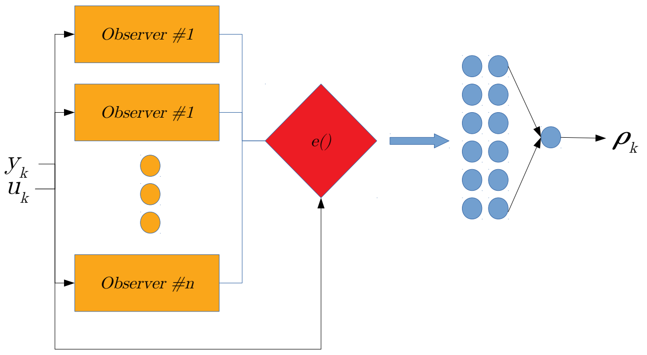

Fig. 1. Virtual sensor architecture: observers (orange), feature extraction map (red), and ANN-based virtual sensor (blue)

D. Choice of the approximator

As highlighted in [22], in order to target an embedded implementation we must envision a learning architecture for g_θ that has a limited memory footprint and requires a small and well predictable throughput. Among various options, as also noted in [23], a good choice is to resort to a very compact feedforward neural network with a small number of layers composed of neurons with Rectified Linear Unit (ReLU) activation functions [24]

$$f_{\text{ReLU}}(x) = \max\{0, x\} \quad (9)$$

The simple structure in (9) requires very limited memory footprint, due to recent advancements in group sparsity regularization techniques [25], and results in a very compact CPU load. Moreover, the number of floating point operations (flops) involved in a evaluating an ANN with ReLU activation functions (9) is independent of the number of samples used in the training phase, in contrast for example to K-nearest neighbor classifiers [19].

We choose a linear activation function for the output layer of the network receiving inputs from the underneath hidden layers. Finally, in order to effectively train the proposed ANN structure, we select the loss function L to be the standard squared prediction error.

The overall architecture proposed for the virtual sensor for the scheduling parameter ρ is shown in Figure 1.

III. Numerical Results

A. Benchmark system

In order to test our approach on a simple example we assume that the underlying system Σ in (1) generating the data is the following linear parameter-varying (LPV) system

$$\Sigma : \begin{cases} x_{k+1}^1 = & A^1 x_k^1 + \rho_k B^1 u_k \\ x_{k+1}^2 = & A^2 x_k^2 + (1-\rho_k) B^2 u_k \\ y_k = & \rho_k C^1 x_k^1 + (1-\rho_k) C^2 x_k^2 \end{cases} \quad (10)$$

with $x^1 \in \mathbb{R}^2$, $x^2 \in \mathbb{R}^2$ ($n_x = 4$), the scheduling signal $\rho_k \in \mathbb{R}$, and

$$\begin{array}{ll} A^1 = \begin{bmatrix} -0.375 & -0.1 \\ 0.125 & 0 \end{bmatrix} & A^2 = \begin{bmatrix} 0.375 & -0.75 \\ 1 & 0 \end{bmatrix} \\ B^1 = \begin{bmatrix} -2 \\ 0 \end{bmatrix} & B^2 = \begin{bmatrix} 2 \\ 0 \end{bmatrix} \\ C^1 = \begin{bmatrix} 0.0625 & 2.5 \end{bmatrix} & C^2 = \begin{bmatrix} -1 & -0.0938 \end{bmatrix} \end{array} \tag{11}$$

A training dataset of 25000 samples and a dataset of 5000 testing samples are generated by exciting the system Σ with a white Gaussian noise input u_k with zero-mean and unit standard deviation starting from the zero initial state. To mimic measurement noise, a zero-mean white Gaussian noise with standard deviation of 0.05 is added on the input, output, and scheduling variable signals.

B. *Virtual observer synthesis*

A recursive ARX estimator based on Kalman filtering [26], parameterized with 3 past outputs, 3 past inputs, and unit input delay has been run on the training set, with $k_1 = \ell + 1$. The resulting vectors $\gamma_k \in \mathbb{R}^6$ of coefficients are then processed using K-means, with the squared Euclidean distance between such vectors used to construct the N clusters. The LTI models Σ_j are obtained by taking a state-space realization of the transfer functions defined by the coefficients of the centroids $\bar{\gamma}_j$ of the clusters, for all $j = 1, \ldots, N$, and a deadbeat observer is then taken for each model Σ_j. We also set the length $\ell = 4$ of the time window of past output estimation errors consumed by the virtual sensor.

The chosen ANN structure consists of 3 layers (2 ReLU layers and an output linear layer), with overall 20 neurons in the nonlinear layers. The network was trained and implemented using the Deep Learning Toolbox of MATLAB R2018b [27], with the training procedure carried out using the Levenberg-Marquardt algorithm [28].

The performance of the resulting virtual sensor is judged on the testing set in terms of the following fit figure

$$\text{FIT} = \max\left\{0, 1 - \frac{\|\rho_k - \hat{\rho}_k\|_2}{\|\bar{\rho}_k - \rho_k\|_2}\right\} \tag{12}$$

where $\bar{\rho}_k$ is the mean value of ρ_k over the test set.

For each examined case we report the mean value and standard deviation of FIT obtained over 10 different runs, each one involving different realizations of the input, scheduling, and noise signals.

Regarding the scheduling signal ρ_k, we generate it in accordance with the following stochastic process

$$\begin{aligned} p_k &\sim \mathcal{U}(0,1) \\ \rho_{k+1} &= \begin{cases} \rho_k + \kappa & \text{if } p_k > 0.75 \wedge |0.5 - \rho_k| < 0.5 \\ \rho_k - 5\kappa & \text{if } p_k < 0.05 \wedge |0.5 - \rho_k| < 0.5 \\ \frac{\rho_k}{2} & \text{if } |0.5 - \rho_k| > 0.5 \\ \rho_k & \text{otherwise} \end{cases} \end{aligned} \tag{13}$$

with $\kappa = 0.015$.

C. *Sensitivity to the number N of models*

In this section we analyze the performance of the virtual sensor with respect to the number N of LTI model/observer pairs used, by comparing the performance obtained using $N = 2, 3, 4, 5$ and 25000 samples for training. The results reported in Table I and in Figure 2 show that performance quickly degrades when too few local models are employed.

TABLE I
AVERAGE FIT (12) OF THE VIRTUAL SENSOR WITH RESPECT TO THE NUMBER N OF LTI MODELS

N	2	3	4	5
mean	0.6258	0.7510	0.7935	0.8054
standard deviation	0.0361	0.0218	0.0157	0.0147

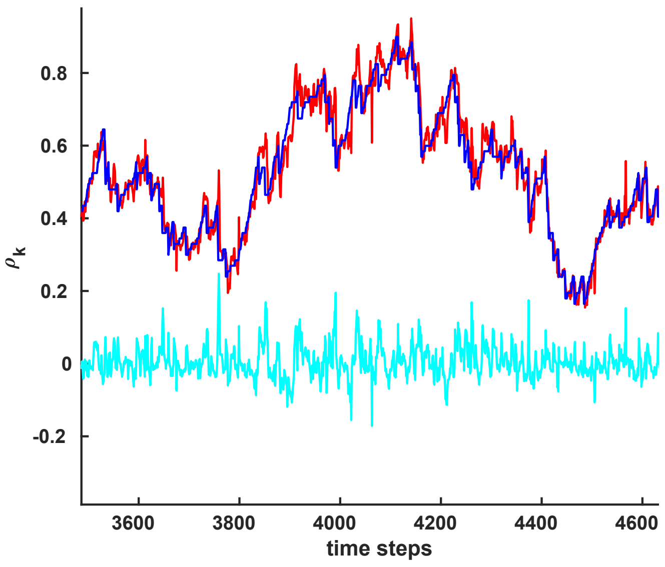

Fig. 2. Example of reconstruction of the scheduling signal by the virtual sensor based on 5 local models and deadbeat observers: ρ_k (blue line), $\hat{\rho}_k$ (red line), reconstruction error (light blue line)

D. *Sensitivity to the number K of samples*

We analyze next the performance obtained by the virtual sensor depending on the number K of samples available for training. Assessing the scalability of the approach with respect to the size of the dataset is extremely important as neural networks often require a large number of samples to be effectively trained.

Table II shows the results obtained by restricting the training procedure to only use a subset of the training dataset, when 5 model/observer pairs are used. It is apparent that good results can be already obtained with only 5000 samples. With smaller datasets fit performance degrades, for example 1000 samples result in a virtual sensor that is not able to reconstruct the scheduling signal satisfactorily.

TABLE II
AVERAGE FIT (12) OF THE VIRTUAL SENSOR USING DATASET OF DIFFERENT SIZES

no. of training samples	1000	2000	5000	25000
mean	0.4115	0.5833	0.7843	0.8054
standard deviation	0.2681	0.2337	0.0147	0.0147

E. *Sensitivity to observer poles*

We now analyze the sensitivity with respect to the location of the observer poles. This analysis could be also relevant to address practical scenarios in which the observers are already in place in a gain-scheduled control system and designed to satisfy different closed-loop control criteria.

Using again $N = 5$ models, we tune the Luenberger observers to have their poles all equal, and vary the pole location in different tests. Results are reported in Table III for the full training dataset of 25000 samples. We highlight that while performance is satisfactory in all cases, fast observers provide better fit performance.

TABLE III

AVERAGE FIT (12) WITH RESPECT TO OBSERVERS POLES

pole location	0.0	0.3	0.6
mean	0.8054	0.7786	0.7157
standard deviation	0.0147	0.0167	0.0304

F. *Sensitivity to ρ_k dynamics*

So far we have analyzed fit performance when the dynamics (13) of the scheduling signal ρ_k is the same in both training and testing data. Here we analyze the performance obtained when the testing dataset is instead generated by the different scheduling signal dynamics

$$\rho_k = \begin{cases} 0.3 + 0.3\cos(\frac{k}{100}) & \text{if } k \le \frac{N}{3} \\ 0.3 & \text{if } k \in [\frac{N}{3}, \frac{2N}{3}] \\ \frac{\text{mod }(k,300)}{300} & \text{otherwise} \end{cases} \tag{14}$$

where N is the number of time steps of the dataset.

The results, reported in Table IV, related to using $N = 5$ models and 25000 or 5000 training samples, show that the performance of the virtual sensor remains quite effective.

TABLE IV

AVERAGE FIT (12) OBTAINED WITH A DIFFERENT LAW FOR ρ_k IN THE TEST DATASET

no. of samples	25000	5000
mean	0.7699	0.7261
standard deviation	0.0149	0.0165

G. *A mode observer for switched linear systems*

An interesting class of systems which can be described as in 1 are switched systems [29], a class of hybrid systems in which the scheduling signal ρ_k can only assume a finite number s of values $\rho^1, \ldots, \rho^s$. In this case, we are approximating model (1) as the discrete-time switched linear system

$$\Sigma := \begin{cases} x_{k+1} = A_{\rho_k} x_k + B_{\rho_k} u_k \\ y_k = C_{\rho_k} x_k \end{cases} \tag{15}$$

The problem of estimating ρ_k from input/output measurements is also known as the *mode-reconstruction* problem. In order to test our virtual sensor approach for mode reconstruction, we let the system generating the data be the switched linear system with $s = 4$ modes obtained from (10) by restricting the scheduling signal ρ_k to only assume the following values

$$\rho_k = \begin{cases} 0.3 & \text{if } k \in [\frac{K}{4}, \frac{K}{2}] \\ 0.75 & \text{if } k \in [\frac{K}{2}, \frac{3K}{4}] \\ 0.94 & \text{if } k > \frac{3K}{4} \\ 0.12 & \text{otherwise} \end{cases} \tag{16}$$

where K is the number of samples collected in the experiment.

We consider two different scenarios. In the first one, the number N of clusters matches the number $s = 4$ of modes of the systems. The corresponding results are reported in Table V and in Figure 3. In the second scenario we only consider $N = 3$ and obtain the results reported in Table VI.

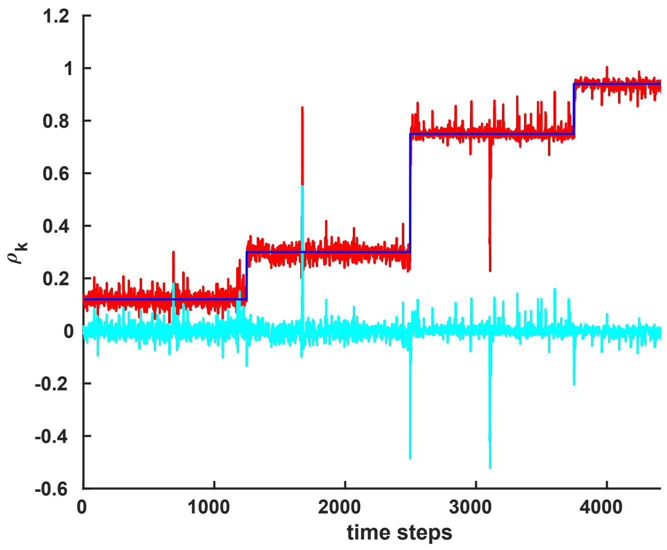

Fig. 3. Example of tracking performance of the scheduling parameter in a switched linear system setting. Blue line is ρ_k, red line is $\hat{\rho}_k$, light blue is the tracking error.

TABLE V

AVERAGE FIT (12) OBTAINED WITH 4 CLUSTERS / 4 MODES

no. of samples	25000	5000
mean	0.9087	0.9018
standard deviation	0.0471	0.0322

TABLE VI

AVERAGE FIT (12) OBTAINED WITH 3 CLUSTERS / 4 MODES

no. of samples	25000	5000
mean	0.8375	0.8241
standard deviation	0.0056	0.0061

Note that in this special case of mode reconstruction, a different ANN architecture could be used to provide a classifier rather than a function regressor, so that it always produces a discrete output.

H. *Computational complexity*

The on-line evaluation of the proposed virtual sensor architecture is extremely light from a computational point of view. Evaluating the virtual sensor on the entire testing set requires 30 ms (roughly 10 μs per sample) on an Intel Core i5 6200U with 16 GB of RAM.

The training procedure also does not require excessive computations: on the same hardware the training process is carried out in less than a minute for a fully sized training set with negligible RAM occupancy. Only about 600 double-precision weights on average are needed to parametrize the ANN in our tests.

I. Effects of information compression

In order to assess how much the information compression (8) proposed in Section II-C affects fit performance, we also test the performance of an ANN modeling directly f_θ in (4) using the same number of layers, which leads to 1101 weights. In both cases we consider $N = 5$ observers and use 25000 training samples.

Results are reported in Table VII. Clearly, with enough storage space for the resulting set of weights characterizing the ANN, it is indeed possible to slightly increase the performance of the virtual sensor, at the cost of almost doubling the number of weights required by the network. On the other hand, the same table also shows that no much is lost due to the compression (8).

TABLE VII

AVERAGE FIT (12) WITH FEATURE EXTRACTION (8) (CASE A) AND USING A UNIT ENCODER e TO DIRECTLY LEARN f_θ IN (4) (CASE B)

case	A	B
mean	0.8054	0.8094
standard deviation	0.0147	0.0148

IV. CONCLUSIONS

In this paper we have proposed a data-driven virtual sensor approach for the reconstruction of the scheduling signal affecting the dynamics of a parameter-varying process. The key idea is to enrich the input/output dataset with the additional data generated by processing the same data through a bank of linear observers. In this way, rather than trying to estimate the scheduling signal directly from a (possibly large) number of past inputs and outputs, we design a very compact ANN which is fed by the current input, current outputs, and by the output estimator errors cumulated over a time-window. The resulting architecture is particularly suitable for embedded applications, due to its limited computational complexity and memory occupancy.

REFERENCES

[1] R. Tóth, *Modeling and Identification of Linear Parameter-Varying Systems*. Springer, Berlin, Heidelberg, 2010.

[2] F. Torrisi and A. Bemporad, "HYSDEL — A tool for generating computational hybrid models," *IEEE Trans. Contr. Systems Technology*, vol. 12, pp. 235–249, Mar. 2004.

[3] D. Rotondo, V. Puig, J. M. A. Valle, and F. Nejjari, "FTC of LPV systems using a bank of virtual sensors: Application to wind turbines," in *Proc. of Conf. on Control and Fault-Tolerant Systems*, pp. 492–497, Oct 2013.

[4] M. Witczak, *Fault Diagnosis and Fault-Tolerant Control Strategies for Non-Linear Systems*. Springer, Cham, 2014.

[5] W. J. Rugh and J. S. Shamma, "Research on gain scheduling," *Automatica*, vol. 36, pp. 1401–1425, 2000.

[6] S. Z. Rizvi, J. M. Velni, F. Abbasi, R. Tóth, and N. Meskin, "State-space LPV model identification using kernelized machine learning," *Automatica*, vol. 88, pp. 38–47, 2018.

[7] V. Breschi, D. Piga, and A. Bemporad, "Piecewise affine regression via recursive multiple least squares and multicategory discrimination," *Automatica*, vol. 73, pp. 155–162, Nov. 2016.

[8] A. Bemporad, V. Breschi, D. Piga, and S. Boyd, "Fitting jump models," *Automatica*, vol. 96, pp. 11–21, Oct. 2018.

[9] M. Witczak, *Unknown Input Observers and Filters*, pp. 19–56. Cham: Springer International Publishing, 2014.

[10] M. Mejari, V. V. Naik, D. Piga, and A. Bemporad, "Regularized moving-horizon pwa regression for lpv system identification," in *Proc. of 18th IFAC Symposium on System Identification*, 2018.

[11] M. Mejari, V. V. Naik, D. Piga, and A. Bemporad, "Energy disaggregation using piecewise affine regression and binary quadratic programming," in *Proc. 57th IEEE Conf. on Decision and Control*, 2018.

[12] A. Bemporad, D. Mignone, and M. Morari, "Moving horizon estimation for hybrid systems and fault detection," in *Proc. of the 1999 American Control Conf.*, vol. 4, pp. 2471–2475 vol.4, 1999.

[13] E. Mazor, A. Averbuch, Y. Bar-Shalom, and J. Dayan, "Interacting multiple model methods in target tracking: a survey," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 34, pp. 103–123, Jan 1998.

[14] L. Ljung, "Convergence analysis of parametric identification methods," *IEEE Transactions on Automatic Control*, vol. 23, no. 5, pp. 770–783, 1978.

[15] M. Milanese, C. Novara, K. Hsu, and K. Poolla, "The filter design from data (fd2) problem: Nonlinear set membership approach," *Automatica*, vol. 45, no. 10, pp. 2350–2357, 2009.

[16] T. Poggi, M. Rubagotti, A. Bemporad, and M. Storace, "High-speed piecewise affine virtual sensors," *IEEE Transactions on Industrial Electronics*, vol. 59, pp. 1228–1237, Feb 2012.

[17] F.-R. L.-E. et al., "Robust sensor fault estimation for descriptor-LPV systems with unmeasurable gain scheduling functions: Application to an anaerobic bioreactor," *Int. Journal of Applied Mathematics and Computer Science*, vol. 25, no. 2, pp. 233–244, 2015.

[18] L. Ljung, *System identification: theory for the user*. Prentice-Hall, 1987.

[19] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[20] S. M. Shinners, *Modern Control System Theory and Design*. New York, NY, USA: John Wiley & Sons, Inc., 2nd ed., 1998.

[21] Y. Gao, L. Zhu, H.-D. Zhu, Y. Gan, and L. Shang, "Extract features using stacked denoised autoencoder," in *Intelligent Computing in Bioinformatics*, pp. 10–14, Springer International Publishing, 2014.

[22] D. Masti and A. Bemporad, "Learning binary warm starts for multiparametric mixed-integer quadratic programming," in *Proc. of European Control Conference*, (Naples, Italy), 2019.

[23] B. Karg and S. Lucia, "Efficient representation and approximation of model predictive control laws via deep learning," *arXiv preprint arXiv:1806.10644*, 2018.

[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Int. Conf. on Machine Learning (ICML)*, pp. 807–814, Omnipress, 2010.

[25] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81 – 89, 2017.

[26] L. Ljung, *System Identification Toolbox for MATLAB – User's Guide*. The Mathworks, Inc., 2001.

[27] M. H. Beale, M. T. Hagan, and H. B. Demuth, *Deep Learning Toolbox – User's Guide*. The Mathworks, Inc., 2018.

[28] N. M. Nawi, A. Khan, and M. Rehman, "A new Levenberg Marquardt based back propagation algorithm trained with Cuckoo search," *Procedia Technology*, vol. 11, pp. 18 – 23, 2013. 4th Int. Conf. on Electrical Engineering and Informatics.

[29] D. Liberzon, *Switching in systems and control*. Springer Science & Business Media, 2003.