IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 25 July 2025, accepted 11 September 2025, date of publication 19 September 2025, date of current version 1 October 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3611914

RESEARCH ARTICLE

On Ensemble Learning Models for Autoencoder-Based Identification of Nonlinear Dynamical Systems

NICOLA ARRIDU[1], (Student Member, IEEE), MARTINA LIPPI[1], (Member, IEEE), MAURO FRANCESCHELLI[2], (Senior Member, IEEE), AND ANDREA GASPARRI[1], (Senior Member, IEEE)

[1]UniversitÃ degli Studi Roma Tre, 00146 Rome, Italy
[2]UniversitÃ degli Studi di Cagliari, 09123 Cagliari, Italy

Corresponding author: Martina Lippi (martina.lippi@uniroma3.it)

ABSTRACT Identifying the dynamic model of a system has been a subject of extensive research for decades. However, dealing with highly complex, possibly nonlinear, and noisy systems remains an open problem. In recent years, the use of data-driven techniques based on machine learning for system identification has represented a promising solution for addressing the challenges of complex system identification. Among these methods, auto-encoders have been successfully applied to compactly represent the system state in a latent space used for estimating the system dynamics. In this work, we propose leveraging the ensemble paradigm for system identification, and in particular to exploit an ensemble of autoencoders, to reduce the uncertainty compared to individual autoencoders. To this end, we propose an algorithm for selecting autoencoders based on a combination of an accuracy metric and a diversity index. We validate this approach using traditional benchmarks in the field of nonlinear system identification.

INDEX TERMS Nonlinear system identification, data-driven analysis, ensemble learning, autoencoders.

I. INTRODUCTION

Knowing or identifying the system model is an essential step in accomplishing most tasks involving the system. For instance, it is necessary to implement effective control strategies, monitor, detect possible failures, and optimize performance. Traditional methods for System Identification (SI) are based on defining mathematical models [1]. Among these methods, there are:

- Volterra series methods [2], which consist of functional expansions used to represent nonlinear systems.
- Block-structured systems [3], which are based on interconnected blocks, where each block represents a subsystem that can be linear or nonlinear.
- NARMAX (Nonlinear AutoRegressive Moving Average with eXogenous inputs) [4], [5], which can capture both linear and nonlinear relationships between inputs, outputs, and noise.
- Stochastic nonlinear models [6], which include stochastic behavior in the model structure to account for uncertainties and noise.

In recent years, advances in machine learning (ML) techniques, particularly deep learning methods, along with the availability of large amounts of data [7], [8], [9] and high computational power, have significantly expanded the use of ML approaches in SI. These exploit data obtained from experimental observations of the system to directly infer models and relations, potentially capturing nonlinearities, uncertainties, and disturbances that can be present in the system, which might be difficult to capture with classical approaches for SI. Furthermore, ML methods generally provide good scalability, facilitating the identification of large-scale or high-dimensional systems [10], [11], [12]. In addition, they potentially offer adaptability features [13], [14], [15], allowing the identified model to adjust to

The associate editor coordinating the review of this manuscript and approving it for publication was Jason Gu.

© 2025 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

environmental changes, the time-evolving nature of the systems, or intentional modifications when provided with sufficient or new data. Hence, ML techniques have paved the way to achieving a comprehensive understanding of real-world complex systems [16], [17] in various fields of science and engineering, such as robotics, economics, and biology.

However, for the successful application of ML methods in system identification, it is crucial that these techniques produce consistently robust and reliable results. Failure to do so may compromise the effectiveness and safety of downstream processes that rely on the identified model. Thus, the *ensemble* paradigm [18] can be leveraged. More specifically, the goal of ensemble learning is to train multiple models and properly combine them to achieve higher accuracy and robustness than individual networks [19], [20]. The core concept is that having an ensemble of networks allows for leveraging the outputs from some networks to compensate for any potential failures in one or more of them. To achieve this, a certain degree of *diversity* inside the ensemble is necessary, as discussed in [21], [22], and [23].

Despite the general benefits provided by the ensemble paradigm, only a few studies exist in the literature that exploit it for system identification purposes, as reported in the next section. In this context, an ensemble of ML-based models can be used to learn diverse representations of the underlying dynamics of the system from its input-output data, and these can be effectively combined to enhance identification accuracy and robustness. Motivated by these considerations, in this study, we propose to combine multiple autoencoder-based models in an ensemble for SI and design an algorithm for performing model selection based on accuracy and diversity metrics. Numerical results on standard benchmarks corroborate the proposed ensemble architecture.

II. RELATED WORKS AND CONTRIBUTION

In this section, we review relevant studies focusing on ML-based system identification. Different ML approaches have been applied to this task, ranging from deep learning models to autoencoders. In this regard, Chiuso and Pillonetto discussed in [24] the intersection of SI and machine learning and reviewed state-of-the-art approaches for SI based on reproducing kernel Hilbert spaces (RKHSs), devising a robust framework for both linear and nonlinear SI.

Deep neural networks were employed in [25] to identify nonlinear systems in combination with restricted Boltzmann machines. Restricted Boltzmann machines were used to pre-train the hidden layers of the neural network by learning an initial weight configuration. Then, the deep neural network was trained using the input-output pairs from the system to be identified. The authors demonstrated that their framework allows a deep neural network to achieve improved system identification results.

Convolutional Neural Networks (CNNs) have also been applied in the context of SI, for example, in [26] and [27]. In more detail, the authors of [26] resorted to a CNN to handle the temporal sequence of system measurements for identification. Remarkably, they analyzed analogies with traditional approaches to SI, such as Volterra series and block-oriented models, and presented numerical simulations based on benchmarks in the field of SI by comparing their CNN model with Multilayer Perceptron (MLP) and Long Short-Term Memory (LSTM) networks. In [27], CNNs were used for SI purposes in the field of structural health monitoring, demonstrating the validity of learning methods to estimate the dynamic response of various systems of interest in structural engineering.

With regard to autoencoders for SI, Masti and Bemporad [8] were among the first to adopt autoencoders to learn nonlinear state-space models, demonstrating their ability to capture complex system dynamics. Their goal is to identify this type of dynamic model from input/output data. The proposed framework simultaneously identifies the nonlinear output and state functions. The study in [5] was based on the integration of variational autoencoders (VAE) and Nonlinear autoregressive exogenous (NARX) methods. In their approach, the decoder was replaced with a NARX module taking the latent space vector from the VAE as input, along with previous inputs and outputs of the system to identify. The study in [28] used an encoder-based model to handle noise and ensure computational efficiency. Their method consists of selecting multiple truncated subsections from the time series used to train the model, computing a loss function for each subsection, and averaging the loss over these sections. The combination of an autoencoder with linear recurrent dynamics was explored in [9]. The proposed method addressed the trade-off between representational capacity and overfitting, providing a balanced approach for model reduction and accurate dynamics learning. Finally, the work in [17] compared various deep learning techniques for nonlinear SI, including autoencoders, Recurrent Neural Networks (RNNs), and a combination of both.

However, all the above methods resorted to individual models for SI. Only a few contributions are available in the literature that resort to the ensemble paradigm for SI. In particular, Ribeiro et al. [29] introduced an SI framework based on four different ML models, partial least squares, support vector machine, boosted generalized linear model, and weighted k-nearest Neighbors, all combined via a boosted linear meta-learner. Negrini et al. [14] proposed a *neural network* ensemble technique for SI. Specifically, multiple feed-forward neural networks were used to obtain different representations of the state, and are then combined through an interpolation network. These approaches have demonstrated the potential of ensemble techniques in SI.

A. CONTRIBUTION

The main contribution of this study is the introduction of an ensemble learning method for nonlinear SI using autoencoder architectures. In addition, by building on the work in [8], which considers individual autoencoder models for SI, we propose to combine multiple models in an

ensemble to improve the identification accuracy. Furthermore, we introduce an algorithm to select the models for inclusion in the ensemble based on an accuracy metric. We validate the proposed approach using two two-tank and the Hammerstein-Wiener SI problems.

As briefly mentioned before, to the best of the authors' knowledge, only the works in [14] and [29] leveraged ensemble learning techniques in the context of nonlinear SI.

The study in [14] used ensembles of feed-forward neural networks, where each network takes as input the state of the system at a given time and outputs the state at the next time instant. Then, the networks are combined via an interpolation network providing an approximation of the dynamics of the system, without resorting to latent representations of the state, as in our case. While well-suited for static scenarios, feed-forward neural networks are potentially less effective when it comes to dealing with dynamic structures.

Ribeiro et al. [29] considered an ensemble of four models: partial least squares, support vector machine, boosted generalized linear model, and weighted k-Nearest Neighbors. These models were then stacked with a meta-learner trained on the prediction of the base learners. Their results showed better accuracy from the ensemble with respect to the base components. A drawback of their approach was the necessary feature engineering, including explicit selection of lagged inputs. Moreover, no selection of the base learners, to discard the poorly performing ones, was realized, and the base learners did not learn a latent representation of the state.

Compared to these papers, we resort to more advanced ML techniques, such as autoencoder-based models. These architectures allow to learn latent representations of complex nonlinear systems and the respective dynamic model. Moreover, with respect to both papers, we introduce an algorithm to select the most convenient components of the ensemble, identifying which members of the ensemble are more suitable for the ensemble by balancing performance and diversity.

III. PRELIMINARIES

Let us consider a discrete-time n-dimensional system with the following nonlinear dynamic model:

$$x_{k+1} = f(x_k, u_k),\ x \in \mathbb{R}^{n_x}, \tag{1}$$
$$y_k = g(x_k),\ y \in \mathbb{R}^{n_o}, \tag{2}$$

where $x_k \in \mathbb{R}^{n_x}$ is the system state, $y_k \in \mathbb{R}^{n_o}$ is the output of the system, and $u_k \in \mathbb{R}^{n_u}$ is the system input. The functions $f : \mathbb{R}^{n_x} \to \mathbb{R}^{n_x}$ and $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_o}$ are assumed to be *unknown*. The goal of system identification is to elaborate on the representation of these two functions.

A. NOTATION FOR SINGLE AUTOENCODER

As previously mentioned, we utilize neural network architectures commonly called autoencoders [30]. They are generally used for unsupervised learning tasks, including SI, and allow the learning of a compact representation of the input data. Briefly, an autoencoder consists of an encoder network that compresses the input into a latent space representation, and a decoder network that reconstructs the input from this representation.

We introduce in the following the notation for a *general* autoencoder i.

- Let I represent the input data, which is a vector containing the measurements of the output and input of the real system that we want to identify. We suppose that the numbers of these two kinds of measurements are equal, with this number being n_I. This assumption is reasonable considering a scenario in which output measurements are collected by providing a system with known inputs.
- Let $Z_i \in \mathbb{R}^{n_z}$ represent the latent space, or compressed, representation obtained by the i-th autoencoder, where $\mathbb{R}^{n_z}$ represents the latent space with dimension n_z, in which Z_i is included.
- Let $\hat{I}_i$ represent the reconstructed output data, produced by the i-th autoencoder.
- Let $f_{\text{enc}} : \mathbb{R}^{2n_I} \to \mathbb{R}^{n_z}$ denote the encoder function of the i-th autoencoder, that maps input data I to the latent space representation Z_i.
- Let $f_{\text{dec}} : \mathbb{R}^{n_z} \to \mathbb{R}^{2n_I}$ denote the decoder function of the i-th autoencoder, which maps the latent space representation Z_i back to space $\hat{I}_i$.

In general, an autoencoder is trained to make the reconstructed output $\hat{I}_i$ as close as possible to the input I.

B. NOTATION FOR AN ENSEMBLE OF AUTOENCODERS

Let us consider an ensemble of N autoencoders, indexed by $i = 1, 2, \ldots, N$. As presented above, for all autoencoders, the encoder networks take the input data (e.g., time series data representing system inputs or outputs) and map them to lower-dimensional latent space representations, whereas the decoder networks attempt to reconstruct the input data from the compressed representations, that is,

$$Z_i = f_{\text{enc},i}(I), \quad \forall i = 1, 2, \ldots, N, \tag{3}$$
$$\hat{I}_i = f_{\text{dec},i}(Z_i), \quad \forall i = 1, 2, \ldots, N. \tag{4}$$

Given the representations of the individual autoencoders, these can be combined as follows to achieve an *ensemble representation*:

$$Z_{\text{en}} = \frac{1}{N}\sum_{i=1}^{N} Z_i = \frac{1}{N}\sum_{i=1}^{N} f_{\text{enc},i}(I), \tag{5}$$
$$\hat{I}_{\text{en}} = \frac{1}{N}\sum_{i=1}^{N} \hat{I}_i = \frac{1}{N}\sum_{i=1}^{N} f_{\text{dec},i}(Z_i), \tag{6}$$

where:

- $Z_{\text{en}} \in \mathbb{R}^{n_z}$ is the average latent representation over the ensemble.
- $\hat{I}_{\text{en}} \in \mathbb{R}^{2n_I}$ is the average reconstruction output data from the ensemble.

As for individual autoencoders, the general objective of an ensemble of autoencoders is to collectively minimize the

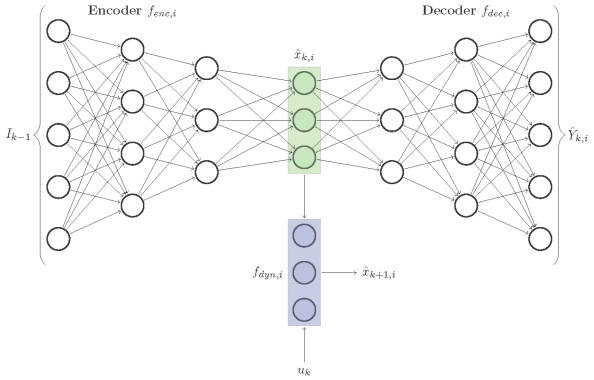


FIGURE 1. Representation of an autoencoder extended with the dynamics model.

reconstruction error, that is, to make $\hat{I}_{en}$ as close as possible to I. Clearly, as highlighted in [18], different combinations can be employed in ensemble functions beyond the average operation. In this study, we use the average as a representative example for system identification.

In the following section, we present how individual autoencoders and ensembles of autoencoders can be exploited for SI.

IV. DATA-DRIVEN SI FRAMEWORK

In this section, we present the structure of the dataset used to identify the system model. Then, by building on [8], we show how ML models based on autoencoders can be employed to solve the problem of nonlinear SI in a data-driven manner, using the measurements in the dataset. Finally, we extend the results to an ensemble of ML models based on autoencoders.

A. INFORMATION EXTRACTION FROM RAW DATA

We assume that measurements of the input and output of the system are available in dataset $\mathcal{D}$, which is used as a training set for autoencoders. Specifically, let us consider that $\mathcal{D}$ contains n_D measurements of the system output, that is, y_k, and n_D measurements of the system input, that is, u_k. We have:

$$\mathcal{D} = \{(y_{k_f}, u_{k_f}), (y_{k_f-1}, u_{k_f-1}), \ldots .(y_{k_0}, u_{k_0})\}, \tag{7}$$

where k_0 is the time instant at which the first measurements are taken and k_f is the time instant of the last measurement. We assume that two subsequent couples of measurements are separated by a time interval T.

Given dataset $\mathcal{D}$, we aim to identify the model of the respective system, that is, to estimate the functions in (3) and (4), in a data-driven manner. To this end, we organize $\mathcal{D}$ in a series of time windows that are provided to the encoder as input I. Each time window datum, called *information* vector I_k, is composed of n_I values related to the output and n_I values related to the input, indexed by time k, that is,

$$I_k = [y_k, y_{k-1}, \ldots, y_{k-n_I+1}, u_k, u_{k-1}, \ldots, u_{k-n_I+1}]^T. \tag{8}$$

From I_k, we can extract the output vector Y_k, which contains n_Y measurements of the output of the real system, with $n_Y \leq n_I$, that is,

$$Y_k = [y_k, y_{k-1}, \ldots, y_{k-n_Y+1}]^T. \tag{9}$$

B. AUTOENCODER-BASED MODEL AND TRAINING FOR SI

To estimate the unknown functions in (1) and (2), we consider that each autoencoder-based model i is composed of the modules in Fig. 1:

1) An encoder takes the information vector I_k at each time step k, and derives a latent space representation of the state.
2) A decoder that, at each time step k, reconstructs the output vector Y_k associated with the state representation at each time step k.
3) A dynamics neural network, denoted by $f_{\text{dyn},i}(\cdot)$, models the dynamics of the system in the latent space.

By following the approach in [8], we first adapt equations (3) and (4) of autoencoder i to the SI purposes as follows

$$\hat{x}_{k,i} = f_{\text{enc},i}(I_{k-1}), \tag{10}$$

$$\hat{Y}_{k,i} = f_{\text{dec},i}(\hat{x}_{k,i}), \tag{11}$$

with $i \in \{1, \cdots, N\}$, where the output of the encoder $\hat{x}_{k,i}$ gives a representation of the state x_k of the system, obtained based on the memory of past observations of inputs and outputs from time step $k-n_I$ to $k-1$ contained in I_{k-1}, while the output of the decoder provides the estimate of the system outputs in the finite-time horizon, i.e., it estimates the vector Y_k. The equations (10) and (11) would define the classical autoencoder as defined in (3) and (4), except for the fact that the decoder in (11) reconstructs only the output of the system to identify, thus neglecting its input.

Note that although the dimension of state n_x is unknown, the dimension of $\hat{x}_{k,i}$, denoted as $n_{\hat{x}}$, is known. The choice of its value should be a compromise between the existence of a compact representation and that of capturing the complexities of the system dynamics. This choice was studied as a technique to *regularize* a neural network [31], [32], [33]. Regularization methods in the field of system identification were considered in [8], [9], and [24].

As far as the dynamics modeling neural network $f_{\text{dyn},i}(\cdot)$ is concerned, this aims to model the dynamics $f(\cdot)$. More specifically, it predicts, for each time step k, the next latent state of the system based on the latent space representation $\hat{x}_{k,i}$, given by the encoder, and the known input u_k, that is, it holds that

$$\hat{x}_{k+1,i} = f_{\text{dyn},i}(\hat{x}_{k,i}, u_k). \tag{12}$$

For each individual model, the autoencoder and the dynamics modeling neural network are trained simultaneously. This joint training allows the model to achieve a compact representation of the system output while capturing system dynamics. For each autoencoder-based model i, we aim to solve the following optimization problem:

$$\min_{f_{\text{enc},i}, f_{\text{dec},i}, f_{\text{dyn},i}} \sum_{k=k_0+n_I}^{n_D+1} \mathcal{L}_{i,k}, \tag{13}$$

where $\mathcal{L}_{k,i}$ represents the overall loss function of model i at time step k. This is chosen as a weighted combination of a reconstruction loss $\mathcal{L}^r_{k,i}$, temporal consistency loss $\mathcal{L}^t_{k,i}$, and decoder-dynamics loss $\mathcal{L}^d_{k,i}$, that is,

$$\mathcal{L}_{k,i} = \alpha \cdot \mathcal{L}^r_{k,i} + \beta \cdot \mathcal{L}^t_{k,i} + \gamma \cdot \mathcal{L}^d_{k,i}, \tag{14}$$

with α, β, and γ are the positive gains.

Regarding the loss terms, the reconstruction term, $\mathcal{L}^r_{k,i}$, quantifies the difference between the actual and reconstructed system outputs and is given by

$$\mathcal{L}^r_{k,i} = L(Y_k, \hat{Y}_{k,i}) + L(Y_{k+1}, \hat{Y}_{k+1,i}), \tag{15}$$

with L being a generic loss function, such as an *L1 loss* or a *L2 loss* [34]. Note that the second contribution using the future outputs Y_{k+1} enhances the capacity of the autoencoder-based model to accurately predict future behaviors. Regarding the temporal consistency loss, $\mathcal{L}^t_{k,i}$, this evaluates the difference between the latent state at time $k+1$, that is, $\hat{x}_{k+1,i} = f_{\text{enc},i}(I_k)$, and the respective state predicted by the dynamics modeling neural network at time step k, that is, it holds that

$$\mathcal{L}^t_{k,i} = L(f_{\text{dyn},i}(\hat{x}_{k,i}, u_k), \hat{x}_{k+1,i}). \tag{16}$$

Finally, the decoder-dynamics loss $\mathcal{L}^d_{k,i}$ measures the difference between the actual future outputs of the system and the reconstructed output from the decoder using the predicted behavior from the dynamics modeling neural network, that is,

$$\mathcal{L}^d_{k,i} = L(Y_{k+1}, f_{\text{dec},i}(f_{\text{dyn},i}(\hat{x}_{k,i}, u_k))). \tag{17}$$

Note that the problem in (13) is a one-step ahead optimization formulation, which might not capture long-term dependencies in the data and result in low prediction capabilities [8]. To improve these capabilities, a multi-step ahead formulation with the Backpropagation Through Time (BPTT) method should be included, in which the initial state estimate is propagated through each step k of the dataset. To reduce the computational complexity of this approach, the Truncated BPTT (TBPTT) [35] method is integrated, which achieves multi-step ahead prediction in an efficient manner by propagating the state estimate in a limited number of steps $F \ll n_D$, as detailed in [8]. Note that the case $F = 1$ leads to the one-step ahead formulation in (13).

C. ENSEMBLE OF AUTOENCODER-BASED MODELS FOR SI

At this point, we present how an ensemble of N autoencoder-based models can be leveraged for system identification. More specifically, by considering the data in $\mathcal{D}$, we adapt the expression of the ensemble equations in (5)-(6) as follows:

$$\hat{x}_{k,en} = \sum_{i=1}^{N} w_i f_{\text{enc},i}(I_{k-1}), \tag{18}$$

$$\hat{Y}_{k,en} = \sum_{i=1}^{N} w_i f_{\text{dec},i}(\hat{x}_{k,i}), \tag{19}$$

where:

- $\hat{x}_{k,en}$ is the estimate of the system state created by the ensemble at time k by resorting to the information vector I_{k-1}.
- $\hat{Y}_{k,en}$ represents the output data at time k reconstructed by the ensemble by resorting to the estimated state.
- w_i is a positive weight associated with the model i.

To train the ensemble, we independently train the N autoencoder-based models and then combine them as reported above. This approach is commonly adopted, as shown in [14] and [29].

However, for the ensemble to successfully solve the SI problem, it is crucial that the representations given by each component of the ensemble are *sufficiently diverse*, as demonstrated in [22]. Intuitively, if we have low diversity, the autoencoders become nearly identical, essentially functioning as copies of one another. Hence, if the ensemble consists solely of almost identical autoencoders, the averaging operation in (18)-(19) yields the same result as that obtained using a single autoencoder. Motivated by this consideration, in this study, we propose a selection algorithm aimed at identifying the most effective components within the ensemble for use in (18)-(19). Indeed, a high number of components N raises the risk of redundancy, and the inclusion of similar or identical autoencoders in the ensemble. To minimize this, N should be kept sufficiently small to ensure diversity and to avoid the inclusion of duplicates. This aspect is investigated in Section VI.

It is worth noting that in future work, the diversity aspect could also be incorporated into the training phase to further enhance the effectiveness of the ensemble. However, this was beyond the scope of this study.

V. SELECTION ALGORITHM FOR ENSEMBLE COMPONENTS

In this section, we present an Ensemble Component Selection Algorithm (*ECSA*) that is designed to choose the most suitable ensemble components for SI. The resulting architecture is depicted in Fig. 2. For the ECSA, we resort to a validation set $\mathcal{V}$ that contains $n_{\mathcal{V}}$ couples in the form (y_k, u_k) where y_k and u_k represent measurements of the system output and input at time k, respectively, i.e.,

$$\mathcal{V} = \{(y_{k_{f,\mathcal{V}}}, u_{k_{f,\mathcal{V}}}), (y_{k_{f,\mathcal{V}}+1}, u_{k_{f,\mathcal{V}}+1}) \ldots, (y_{k_{0,\mathcal{V}}}, u_{k_{0,\mathcal{V}}})\}, \tag{20}$$

where $k_{0,\mathcal{V}}$ and $k_{f,\mathcal{V}}$ are the time instants related to the first and last pairs of validation measurements, respectively. Two subsequent couples are separated by the time interval T, which we assume to be the same as that of the training set in (7). To introduce the selection algorithm, we first present the Normalized Root Mean Square Accuracy (NRMSA) metric to evaluate the performance of single autoencoder-based models and the ensemble, and the correlation matrix to evaluate the correlation between individual models.

Let us consider autoencoder i in an ensemble, with $i \in \{1, \cdots, N\}$. For this autoencoder, the NRMSA is

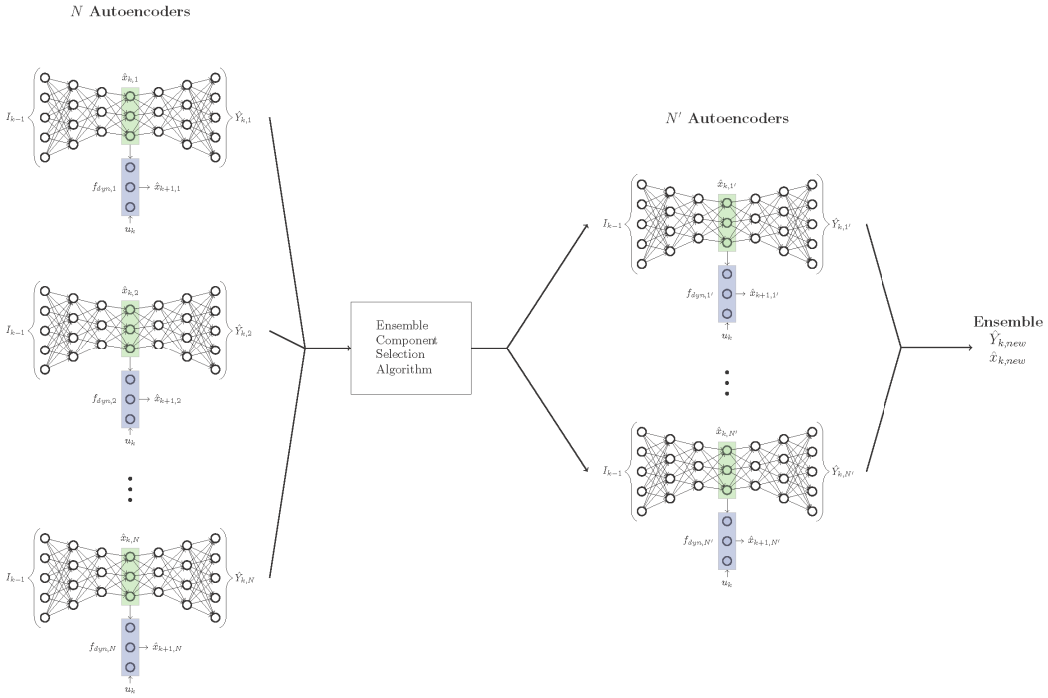


FIGURE 2. Graphic representation of the selection carried out by the Algorithm 1, from an ensemble of N autoencoders to an ensemble of N' autoencoders.

computed as follows:

$$NRMSA_i = 1 - \frac{\sqrt{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}}(\hat{y}_{k,i} - y_k)^2}}{Y_{max}} \tag{21}$$

where $\hat{y}_{k,i}$ is the value of the output representation of the autoencoder i for the time instant k, extracted from the decoder in (11), y_k is obtained from the validation set $\mathcal{V}$, Y_{max} is the maximum of all the outputs y_k in $\mathcal{V}$, and n_T is the number of time instants in $\mathcal{V}$. The formulation in (21) implies that the closer the autoencoder's output aligns with the measurements in the test set, the closer the NRMSA value approaches 1.

To calculate the NRMSA for the ensemble, denoted as $NRMSA_{en}$, the same formulation as in (21) is employed, with $\hat{y}_{k,en}$ used in place of $\hat{y}_{k,i}$. Regarding the correlation matrix, given N models, this represents an $N \times N$ matrix where the coefficient (i, j), denoted as c_{ij}, indicates the correlation between models i and j and is computed as

$$c_{ij} = \frac{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}}\left(\hat{y}_{k,i} - \bar{Y}_i\right)\left(\hat{y}_{k,j} - \bar{Y}_j\right)}{\sqrt{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}}\left(\hat{y}_{k,i} - \bar{Y}_i\right)^2} \cdot \sqrt{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}}\left(\hat{y}_{k,j} - \bar{Y}_j\right)^2}} \tag{22}$$

where $\hat{y}_{k,i}$ and $\hat{y}_{k,j}$ are the outputs of decoders of i and j at time instant k. The values $\bar{Y}_i$ and $\bar{Y}_j$ are the averages of $\hat{Y}_{k,i}$ and $\hat{Y}_{k,j}$ over the validation set $\mathcal{V}$. Therefore, each element of the correlation matrix measures the similarity between two autoencoders belonging to the ensemble. The formula is based on the Pearson correlation coefficient [36], which measures the dependency between two variables. In our case, they consist of the output of two autoencoder-based models. The coefficient ranges between -1 and 1, where the values

Algorithm 1 Ensemble Component Selection Algorithm

Input: N pretrained autoencoder-based models on dataset $\mathcal{D}$, validation set $\mathcal{V}$, N' (number of the selected autoencoders), weights ω, and ρ
Output: ensemble of N' models, weights W

```
/* Phase 1: Initialization */
list_en ← {1, ··· , N};
for i ∈ {1, ··· , N} do
    NRMSA_i ← Calculate NRMSA for model i;
end
corr ← Compute correlation matrix;
/* Phase 2: Score calculation */
list_scores ← {};
for i ∈ {1, ··· , N} do
    s_i ← compute similarity for model i;
    v_i ← ω * NRMSA_i − ρ * s_i;
    list_Score ← Add value score_i;
end
/* Phase 3: Selection */
list_selected ← select the N′ models with the highest scores in ist_Score;
W ← create N′ weights for the models in list_selected;
```

1 and −1 indicate linear dependency between the outputs of the models, while the value 0 denotes no linear correlation. In our framework, we want to privilege autoencoders that are as different as possible with respect to other autoencoders.

At this point, we can introduce the ensemble component selection algorithm, which exploits the NRMSA metrics and correlation values to select the components of the ensemble to improve the overall performance of the ensemble (in terms of NRMSA).

The algorithm takes as inputs the validation set $\mathcal{V}$, N pretrained autoencoder-based models, the number N' of autoencoders that we intend to select, and two positive weights ω and ρ, and it outputs a subset of N' selected models, as well as a weight vector W used to compute the output of the new ensemble, as in (19).

In the initialization phase, we introduce the set $list_{en}$ collecting the indices of the considered autoencoder-based models in the ensemble (line 1) and set it equal to $\{1, \ldots, N\}$ (line 2). Next, we compute the NRMSA values for all individual models (lines 2-4) and the $N \times N$ correlation matrix $corr$ (line 5).

After the initialization is completed, we proceed to calculate, for each autoencoder, the score on which our selection is based, and we collect the resulting values in a set denoted as $list_{score}$ (line 7-11). The score value v_i of an autoencoder-based model i is obtained as the weighted sum of two terms (line 9): a performance term, given by the NRMSA metric, and a similarity term. The similarity term is calculated as the mean of the correlation values between the model i and the other models in the ensemble, i.e., $s_i = \frac{1}{|list_{en}|-1} \sum_{j \in list_{en}, j \neq i} c_{ij}$. Therefore, the similarity s_i quantifies the extent to which model i aligns with the others in terms of output representations. If this similarity is relatively small across the models in the ensemble, this suggests a high degree of diversity within the ensemble. According to [22] and [37], such diversity enhances the overall performance of the ensemble.

The choice of the weighted sum for the score value is motivated by the fact that it balances the selection of models based on performance and promotes diversity by favoring lower similarity within the ensemble. Next, we select the N' models with the highest scores and collect them in the set $list_{selected}$ (line 12). Based on the models in the $list_{selected}$, the output of the new ensemble is obtained as the weighted average of the N' models' decoders, as in (19), where each weight w_i is obtained as

$$w_i = \frac{1}{\sum_{j \in list_{selected}} v_j} v_i. \tag{23}$$

Finally, the weights are stored in the vector W (line 16).

VI. NUMERICAL COMPARATIVE VALIDATION

In this section, we present the numerical simulations related to the identification of a nonlinear two-tank system and to the identification of the Hammerstein–Wiener system.

A. VALIDATION SETTING

As highlighted in [8], it is beneficial not only to search for the nonlinear representation (as in (1)) of a system but also to consider a quasi-Linear Parameter-Varying (quasi-LPV or qLPV) form. This representation is particularly suitable for Model Predictive Control (MPC) because it allows for a more tractable linear approximation for control design and analysis. We used this form to compare our results with those of [8].

The quasi-LPV form for a general system is given by:

$$\begin{cases} x_{k+1} = A(x_k, u_k) \begin{bmatrix} x_k{}^T & 1 \end{bmatrix}^T + B(x_k, u_k) u_k, \\ y_k = C(x_k, u_k) \begin{bmatrix} x_k{}^T & 1 \end{bmatrix}^T, \end{cases} \tag{24}$$

where A, B, and C are the matrices output by a fully connected neural network. We refer to the models in (1) and (2) as the Nonlinear State Space (NSS) model.

We now first introduce the two-tank system S used as a benchmark for testing our framework:

$$S = \begin{cases} x_{k+1,1} = x_{k,1} - k_1 \sqrt{x_{k,1}} + k_2 u_k, \\ x_{k+1,2} = x_{k,2} + k_3 \sqrt{x_{k,1}} - k_4 \sqrt{x_{k,2}}, \\ y_k = x_{k,2}, \end{cases} \tag{25}$$

where $x_{k,1}$ and $x_{k,2}$ represent the fluid levels in the first and second tanks at time step k, respectively, u_k is the input flow rate into the first tank at time step k, y_k is the system output at time step k, equal to the level in the second tank; and k_1, k_2, k_3, and k_4 are system parameters related to the flow rates between the tanks and outflows. The system presents challenges related to its different behaviors, because it exhibits a slow dynamics when the tanks are nearly empty and a fast dynamics when the tanks are nearly full. This forces an identification method to be adaptable to different operating regimes. Identifying S is a typical benchmark in nonlinear system identification [38] and is presented in [8]. We set $k_1 = 0.5$, $k_2 = 0.4$, $k_3 = 0.2$, and $k_4 = 0.3$.

The second benchmark that we considered is the Hammerstein–Wiener system. Its mathematical model is the following:

$$H = \begin{cases} x_{k+1,1} = 0.7555\, x_{k,1} + 0.25\, x_{k,2} - 0.5\, v_k, \\ x_{k+1,2} = -0.1991\, x_{k,1}, \\ v_k = \begin{cases} \sqrt{u_k} & \text{if } u_k > 0 \\ u_k & \text{otherwise} \end{cases} \\ r_k = 0.6993\, x_{k,1} - 0.4427\, x_{k,2} \\ y_k = r_k + 5 \sin(r_k), \end{cases} \tag{26}$$

where u_k is the input, which passes through a nonlinear function, generating the signal v_k. This signal affects the dynamics of the two components of the state $x_{k,1}$ and $x_{k,2}$. The state components $x_{k,1}$ and $x_{k,2}$ are then linearly combined to obtain the signal r_k, which is passed through a nonlinearity to generate the output y_k. The Hammerstein–Wiener system is basically composed of a linear block preceded by a nonlinearity that affects the input, and followed by a nonlinear

transformation at the output. The identification of this system can be useful to prove the effectiveness of our approach for all the real-world problems where a linear system is influenced by nonlinear sensors and actuators.

In the following, we present the test setup for both the S and H models.

For each benchmark, we used three types of input signals u_k:

1) *Noise [*G*]*: Gaussian noise with a mean and standard deviation equal to 1 was considered. Specifically, every five steps, we generated a value with a Gaussian distribution and kept it constant for the subsequent steps. We denote this signal by G in the following.
2) *Multiharmonic [*Mh*]*: A sinusoidal signal with the following form was used, $u_k = 0.5 \sin\left(\frac{k}{20+0.01\,k}\right) + 0.5$. Hereafter, we refer to this signal as *Mh*.
3) *Beta [*B*]*: The Beta distribution, parameterized by the two coefficients δ and ϵ, is often used in Bayesian modeling and Monte Carlo simulations. We selected $\delta = 2$ and $\epsilon = 5$. For a formal definition of the beta distribution, the reader is referred to [39]. This signal is denoted B.

We generated the datasets by establishing the initial conditions for $x_{k,1}$ and $x_{k,2}$ and using (25) and (26) to evolve the systems S and H, respectively, given the above inputs. This formed datasets of pairs (y_k, u_k), which we used to create the training, validation, and test sets. For each benchmark S and H, regarding the training sets, we created the set $\mathcal{D}_G$ starting from a G signal and $\mathcal{D}_h$ from an *Mh* signal. Both sets contained 20000 samples. For the validation sets, we considered 1000 sample sets $\mathcal{V}_G$, from a G signal, and $\mathcal{V}_h$, from an *Mh* signal. For the test sets, we considered 1000 pairs with G and *Mh* inputs, respectively for $\mathcal{T}_G$ and $\mathcal{T}_h$. Moreover, we considered the test set $\mathcal{T}_b$ with 1000 pairs from the B signal.

In both the training and test sets, we normalized the signals composing the datasets by subtracting the respective mean and then dividing by the respective standard deviation. Additionally, we introduced further Gaussian noise to the pairs (y_k, u_k), generated with zero mean and standard deviation equal to 0.02, to simulate the measurement noise.

With regard to the hyperparameters of the neural networks, these were experimentally set as follows. The information vector I_k dimension was set to $n_I = 10$ and the state representation dimension was $n_{\hat{x}} = 6$. The neural networks for the encoder, decoder, and dynamic modeling function had four layers, with the first three layers having 30 neurons each. The encoder's last layer had $n_{\hat{x}}$ units, and the decoder's last layer had n_Y, where n_Y is set to 2. The last layer of the dynamics model had $n_{\hat{x}}$ units. Additionally, in the quasi-LPV case, a further layer was included in this network to estimate A and B. For loss function (14), we used $\alpha = 10$, $\beta = 0.3$, $\gamma = 0$ for the initial five epochs, focusing on input vector reconstruction, and $\alpha = 0$, $\beta = 10$, $\gamma = 1$ for the consecutive five epochs, focusing on system behavior and temporal consistency.

We analyzed both NSS and quasi-LPV systems and trained the autoencoder-based models in two ways: TBPTT with $F = 1$ and $F = 4$, totaling four different configurations. For the ensemble, we considered $N = 10$ autoencoder-based models composing the ensemble and applied Algorithm 1 to select the $N' = 8$ with a better score, to discard only the worse autoencoders while maintaining the vast majority of them. The scores were computed choosing $\omega = 1$ and $\rho = 0.5$, then privileging more the performance and less the diversity, which is anyway not negligible and contributes to selecting the best autoencoders. For each configuration, we trained ten ensembles by initializing the weights of the models with different values.

It is important to note that we used the same values of [8] for the parameters of the two-tank system, the formation of the dataset, the autoencoder's architecture, and the training hyperparameters, in order to have a fair comparison.

Regarding the computational aspects, the training phase of each autoencoder required an average of 7 s per epoch on a PC with an Intel Core i9-11900KF CPU and NVIDIA GeForce RTX 3090 Ti GPU, for both the two-tank and the Hammerstein-Wiener systems. Considering a total of ten epochs, the total training time for each autoencoder was about 70 s. Clearly, the training time for the ensemble scales linearly with the number of autoencoders. However, the training of single autoencoders can be parallelized, considerably reducing the total training time. Similarly, the inference time also scales linearly with the number of ensemble components. On average, an inference time of 0.003 s per autoencoder was recorded, which, also in view of the potential for parallelization, shows the practical feasibility of deploying the proposed models. Regarding the ECSA, its processing time was on average equal to 0.03 s. The majority of the algorithm's running time was dedicated to the evaluation of the performance of the single autoencoders and their diversity with respect to the other autoencoders. These computational results further support the applicability of the proposed approach.

B. NUMERICAL RESULTS

We now present the comparison between the following methods: single autoencoder-based model, as in [8], an ensemble of N models with $w_i = 1/N$ for each model i, and selected ensemble according to ECSA in Algorithm 1.

Table 1 reports the comparative results on the identification of the two-tank system S, whereas Table 2 shows the results on the Hammerstein–Wiener system H. Both tables were obtained using as training test $\mathcal{D}_G$ and $\mathcal{T}_G$ as test set, as in [8]. For the validation set necessary for Algorithm 1, we used the $\mathcal{V}_h$ generated with the *Mh* signal. We chose to use another signal for the validation set to select the autoencoders that can generalize better to a set different than the training one. We ran 10 experiments for each system with this configuration, and the results were averaged over these experiments. In particular, the tables show the results, expressed in terms of NRMSA metric, obtained by the

TABLE 1. Values of the NRMSA metric for different identification methods on different systems (i.e., NSS and qLPV) for the two-tank system *S*. Results with $F = 1$ and $F = 4$ in TBPTT method are also shown. The signal *G* signal is employed to generate both the train and the test sets.

Identification method		NSS/1	NSS/4	qLPV/1	qLPV/4
Single model	Average	0.9930	0.9946	0.9932	0.9943
	Standard deviation	0.0015	0.0006	0.0016	0.0007
	Best	0.9954	0.9962	0.9954	0.9957
Ensemble	Average	0.9945	0.9957	0.9948	0.9955
	Standard deviation	0.0007	0.0003	0.0008	0.0004
	Best	0.9960	0.9965	0.9962	0.9962
Ensemble ECSA	Average	0.9945	0.9958	0.9948	0.9955
	Standard deviation	0.0007	0.0003	0.0008	0.0004
	Best	0.9960	0.9965	0.9962	0.9962

TABLE 2. Values of the NRMSA metric for different identification methods on different systems (i.e., NSS and qLPV) for the Hammerstein–Wiener system *H*. Results with $F = 1$ and $F = 4$ in TBPTT method are also shown. The signal *G* signal is employed to generate both the train and the test sets.

Identification method		NSS/1	NSS/4	qLPV/1	qLPV/4
Single model	Average	0.9310	0.8601	0.8941	0.8349
	Standard deviation	0.0346	0.0263	0.0436	0.0236
	Best	0.9806	0.9268	0.9672	0.8861
Ensemble	Average	0.9752	0.9285	0.9611	0.9164
	Standard deviation	0.0043	0.0076	0.0075	0.0082
	Best	0.9830	0.9400	0.9729	0.9311
Ensemble ECSA	Average	0.9792	0.9296	0.9677	0.9172
	Standard deviation	0.0035	0.0081	0.0079	0.0081
	Best	0.9850	0.9424	0.9786	0.9317

average individual model (top part), by the ensemble of $N = 10$ models (middle part), and the ensemble of $N' = 8$ models resulting from ECSA (lower part). The results are grouped for different systems (i.e., NSS and qLPV) and different TBPTT F values (where the notation $*/F$ is used).

In Table 1, the results show that the ensemble slightly outperforms the best single autoencoder across all configurations. The performance of both the ensemble with N models and the ensemble with ECSA is similar to the single autoencoder due to the low diversity between the autoencoders, which reduces the strength of the ensemble operation. Moreover, the table shows the effectiveness of multi-step ahead optimization (i.e., $F > 1$) compared to one-step ahead prediction (i.e., $F = 1$). Indeed, higher results for the NRMSA metric are achieved with $F = 4$ compared to $F = 1$.

From Table 2, we can see that the ensemble consistently outperforms the single model for all the configurations. Regarding the comparison between the ensemble with all the $N = 10$ autoencoders and the ensemble of $N' = 8$ autoencoders selected by the ECSA, the last one showed better results across all the different configurations, demonstrating the effectiveness of the proposed selection method.

While the results in Table 1 and Table 2 reflect the scenario where the training and test sets are generated from the same signal, we additionally tested a more challenging identification problem on the system S, and considered the case where the training and test sets were obtained with different input distributions. In particular, Table 3 presents the results obtained using $\mathcal{D}_h$ (*Mh* signal) as training set, $\mathcal{V}_G$ (*G* signal) as validation set, and $\mathcal{T}_G$ (*G* signal), $\mathcal{T}_h$ (*Mh* signal), and $\mathcal{T}_B$ (*B* signal) as test sets. Again, we ran 10 experiments, and we averaged over these experiments. In this case, the system analyzed was an NSS with $F = 1$ for the TBPTT method. We chose this configuration because this represents the worst-case scenario for the performance according to the results in Table 1.

We can observe that, especially for the test sets obtained with distributions different from the training one, i.e., for $\mathcal{T}_G$ and $\mathcal{T}_B$, the entire ensemble and the one selected according to Algorithm 1 achieve significantly better performance than the single model, with lower standard deviation, demonstrating the stability of the ensemble approach and of the proposed ECSA. The improvement on unseen distributions shows higher generalization capabilities of the proposed ensemble model with respect to the single autoencoders. As shown in [22] and [37], this is thanks to the diversity of the models, which allows to reduce the generalization error.

We now graphically show the outputs produced by the proposed method compared to ground truth data. The test setup was the same that originated Table 3. The ensemble tested was composed by autoencoders modeled as a Nonlinear State Space (NSS) system, with $F = 1$ for the TBPTT method. The training of the ensemble was conducted using $\mathcal{D}_h$ set (generated with a multi-harmonic signal) as training set, $\mathcal{V}_G$ (Gaussian signal) as validation set, and $\mathcal{T}_G$ (Gaussian signal), $\mathcal{T}_h$ (multi-harmonic signal), and $\mathcal{T}_B$ (Beta signal) as test sets. Fig. 3 shows the direct comparisons (left figures) of the output of the ensemble of autoencoders selected by the ECSA (in blue) and the output of the real two-tank system (in orange). Three different input signals from the three test sets were considered as reported in the right figures (Gaussian, multi-harmonic, and beta inputs from top to down, respectively). The results indicate that the ensemble is capable of generalizing beyond the training set, with varying degrees of accuracy depending on the similarity between the test and training data. Specifically, for the Gaussian input (Fig. 3.a), the predicted output follows the true output quite closely except at the peaks. For the multi-harmonic input (Fig. 3.c), which matches the type of signal used during training, the predictions are highly accurate, whereas performance is lower for the beta input (Fig. 3.e). These observations are further supported by Table 1, which reports the mean and standard deviation of the real and predicted outputs for each input type. The results show that the ensemble approximates the real system in terms of both average values and variability, with performance varying across the different test sets.

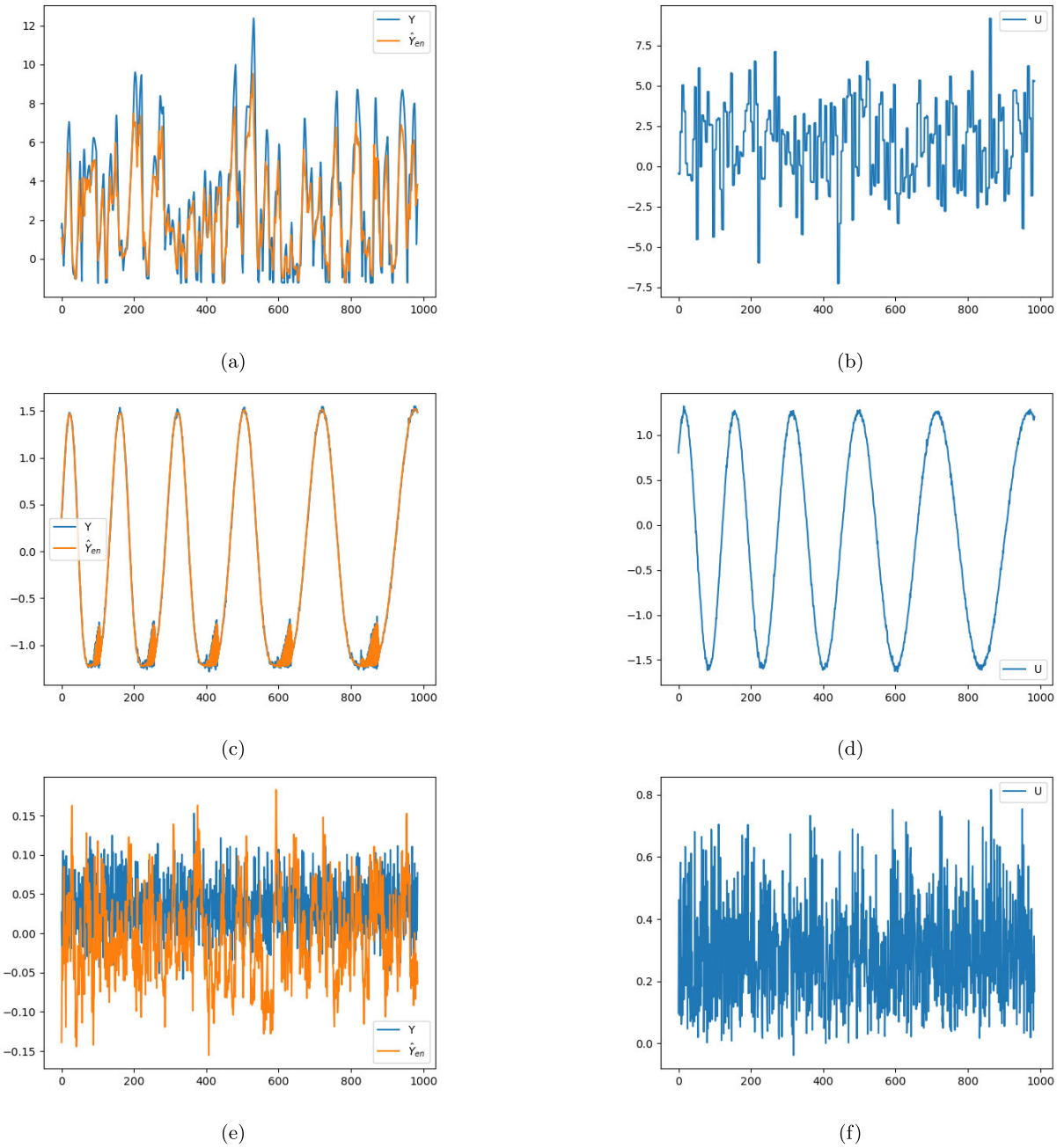


FIGURE 3. Open loop comparison between the output Y of the real two-tank system, the ensemble output $\hat{Y}_{en}$ (figures on the left), and the real system input U (figures on the right) for three different distributions: Gaussian (Figs. (a) and (b)), Multi-harmonic (Figs. (c) and (d)), and Beta (Figs. (e) and (f)).

C. ENSEMBLE RESULTS WITH VARYING NUMBER OF COMPONENTS

We now focus on evaluating the general performance of the ensemble paradigm for SI by employing different numbers of components N, without using the ECSA in Algorithm 1. The comparison was based on their scores on the NRMSA metric. The autoencoder-based models were trained using the parameters presented before.

For this analysis, we considered the system S, we trained 10 distinct ensembles, each composed of $N = 10$ autoencoder-based models. Subsequently, to create ensembles with fewer models ($N < 10$), we selected the N best-performing models from each original ensemble with 10 models. For each ensemble, we evaluated the NRMSA performance on the two test sets.

The mean and standard deviation of the NRMSA values for different sizes are shown in Fig. 4. We can observe that for low values of N, increasing the ensemble size generally improves the performance, as we add relatively good autoencoders and enhance the ensemble's diversity,

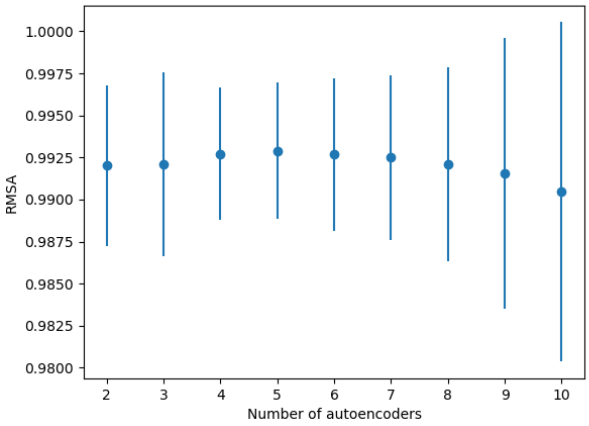


FIGURE 4. Comparison of ensembles with a different number of components based on their NRMSA value.

TABLE 3. Values of the NRMSA metric for different test sets with the NSS two-tank system and $F = 1$ for the TBPTT method, using the training set with *Mh* signal.

Identification method		$\mathcal{T}_G$	$\mathcal{T}_h$	$\mathcal{T}_B$
Single model	Average	0.8721	0.9887	0.3520
	Standard deviation	0.0301	0.0008	0.1033
	Best	0.9243	0.9905	0.5850
Ensemble	Average	0.8911	0.9897	0.6096
	Standard deviation	0.0122	0.0002	0.0286
	Best	0.9104	0.9901	0.6635
Ensemble ECSA	Average	0.8993	0.9896	0.6098
	Standard deviation	0.0116	0.0002	0.0331
	Best	0.9182	0.9900	0.6734

TABLE 4. Statistical comparison between the real system and the Ensemble ECSA for different test sets.

Output model		$\mathcal{T}_G$	$\mathcal{T}_h$	$\mathcal{T}_B$
Real system	Average	2.8388	-0.1283	0.0365
	Standard deviation	3.0093	0.9981	0.0322
Ensemble ECSA	Average	2.4554	-0.1279	-0.0015
	Standard deviation	2.1424	0.9950	0.0526

which, as explained by [22], can be beneficial for the overall effectiveness of the ensemble. However, for higher values of N, we include lower-performing autoencoder-based models (according to our grouping approach), which, despite increasing diversity, may ultimately reduce the ensemble's overall performance and increase its variance. It is important to note that this plot was influenced by the ensemble selection process, where we considered the best models to be grouped first. Clearly, a different approach, such as random selection, would yield a different plot. However, we believe that starting with the best models is a natural choice when determining the ensemble size for our system identification task.

VII. CONCLUSION

In this study, an ensemble learning system identification framework was proposed. To this end, we resorted to autoencoder-based models adapted for system identification and extended them to ensemble settings. Additionally, we introduced ECSA, an algorithm capable of selecting a subset of the autoencoder-based models in order to maximize the system's diversity and NRMSA performance metrics. We demonstrated the effectiveness of the ensemble approach on two well-known benchmarks, which are the identification of a two-tank system and of the Hammerstein-Wiener system. We then showed that the ensemble method generally outperforms the individual models.

In future studies, we will validate the approach on real data collected from a physical system in a laboratory setting or industrial scenario, investigating robustness against measurement noise and hardware nonlinearities. Moreover, future work will focus on leveraging the availability of multiple autoencoder-based models at training time to improve the ensemble performance and to design control algorithms leveraging the learned system model.

REFERENCES

[1] L. Ljung, *System Identification: Theory for the User*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[2] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. Melbourne, FL, USA: Krieger, 2006.

[3] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of Hammerstein–Wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2012.

[4] S. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. Hoboken, NJ, USA: Wiley, 2013. [Online]. Available: https://books.google.it/books?id=SaQ2AAAAQBAJ

[5] J. L. Paniagua and J. A. López, "Nonlinear system identification using modified variational autoencoders," *Intell. Syst. Appl.*, vol. 22, Jun. 2024, Art. no. 200344.

[6] P. Kloeden and E. Platen, *Numerical Solution of Stochastic Differential Equations*. Berlin, Germany: Springer, 1992.

[7] S. L. Brunton and J. N. Kutz, *Data-Driven Science and Engineering: Machine Learning, Dynamical Systems, and Control*. Cambridge, U.K.: Cambridge Univ. Press, 2019.

[8] D. Masti and A. Bemporad, "Learning nonlinear state–space models using autoencoders," *Automatica*, vol. 129, Jul. 2021, Art. no. 109666.

[9] S. E. Otto and C. W. Rowley, "Linearly recurrent autoencoder networks for learning dynamics," *SIAM J. Appl. Dyn. Syst.*, vol. 18, no. 1, pp. 558–593, Jan. 2019.

[10] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," *Nature*, vol. 521, no. 7553, pp. 452–459, 2015.

[11] M. Raissi, P. Perdikaris, and G. E. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," *J. Comput. Phys.*, vol. 378, pp. 686–707, Feb. 2019.

[12] M. Lazar, M.-S. Popescu, and M. Schoukens, "Nonlinear data-driven predictive control using deep subspace prediction networks," in *Proc. 62nd IEEE Conf. Decis. Control (CDC)*, Dec. 2023, pp. 3770–3775.

[13] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Trans. Neural Netw.*, vol. 1, no. 1, pp. 4–27, Mar. 1990.

[14] E. Negrini, G. Citti, and L. Capogna, "A neural network ensemble approach to system identification," 2021, *arXiv:2110.08382*.

[15] M. Forgione, F. Pura, and D. Piga, "From system models to class models: An in-context learning paradigm," *IEEE Control Syst. Lett.*, vol. 7, pp. 3513–3518, 2023.

[16] L. Ljung, C. R. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.
[17] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," in *Proc. IFAC-PapersOnLine*, 2021, vol. 54, no. 7, pp. 481–486.
[18] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers Comput. Sci.*, vol. 14, pp. 241–258, Apr. 2020.
[19] C. Zhang and Y. Ma, *Ensemble Machine Learning*, vol. 144. New York, NY, USA: Springer, 2012.
[20] Z.-H. Zhou, *Ensemble Methods: Foundations and Algorithms*. Boca Raton, FL, USA: CRC Press, 2012.
[21] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Mach. Learn.*, vol. 51, no. 2, pp. 181–207, May 2003.
[22] D. Wood, T. Mu, A. Webb, H. W. J. Reeve, M. Luján, and G. Brown, "A unified theory of diversity in ensemble learning," *J. Mach. Learn. Res.*, pp. 1–49, 2023.
[23] Y. Wu, L. Liu, Z. Xie, K.-H. Chow, and W. Wei, "Boosting ensemble accuracy by revisiting ensemble diversity metrics," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 16464–16472.
[24] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annu. Rev. Control, Robot., Auto. Syst.*, vol. 2, no. 1, pp. 281–304, May 2019.
[25] E. de la Rosa and W. Yu, "Randomized algorithms for nonlinear system identification with deep learning modification," *Inf. Sci.*, vols. 364–365, pp. 197–212, Oct. 2016.
[26] C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön, "Deep convolutional networks in system identification," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 3670–3676.
[27] R.-T. Wu and M. R. Jahanshahi, "Deep convolutional neural network for structural dynamic response estimation and system identification," *J. Eng. Mech.*, vol. 145, no. 1, Jan. 2019, Art. no. 04018125.
[28] G. I. Beintema, M. Schoukens, and R. Tóth, "Deep subspace encoders for nonlinear system identification," *Automatica*, vol. 156, Oct. 2023, Art. no. 111210.
[29] M. H. D. M. Ribeiro, V. C. Mariani, L. F. Manke, and L. dos Santos Coelho, "Nonlinear system identification based on ensemble learning: The cascaded tanks system case," in *Proc. Anais 14th Simpósio Brasileiro De Automação Inteligente*, 2019, pp. 1–6.
[30] G. E. Hinton and R. S. Zemel, "Autoencoders, minimum description length and Helmholtz free energy," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 6, 1993, pp. 3–10.
[31] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE J.*, vol. 37, no. 2, pp. 233–243, Feb. 1991.
[32] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini, "Group sparse regularization for deep neural networks," *Neurocomputing*, vol. 241, pp. 81–89, Jun. 2017.
[33] J. Wang, H. Zhang, J. Wang, Y. Pu, and N. R. Pal, "Feature selection using a neural network with group lasso regularization and controlled redundancy," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 3, pp. 1110–1123, Mar. 2021.
[34] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. New York, NY, USA: Springer, 2009.
[35] G. V. Puskorius and L. A. Feldkamp, "Truncated backpropagation through time and Kalman filter training for neurocontrol," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, Jun. 1994, pp. 2488–2493.
[36] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proc. Roy. Soc. London*, vol. 58, pp. 240–242, Dec. 1895. [Online]. Available: http://www.jstor.org/stable/115794
[37] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *Proc. Int. Conf. Neural Netw. (ICNN96)*, vol. 1, 1996, pp. 90–95.
[38] M. Schoukens and J. P. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 446–451, Jul. 2017.
[39] G. Casella and R. Berger, *Statistical Inference*. Boca Raton, FL, USA: CRC Press, 2024.



NICOLA ARRIDU (Student Member, IEEE) received the B.S. degree in electric and electronic engineering and the M.S. degree (cum laude) in electronic engineering from the University of Cagliari, Cagliari, Italy, in 2018 and 2021, respectively. He is currently pursuing the Ph.D. degree with Roma Tre University, Rome, Italy.

He received a six-month research fellowship at the University of Cagliari. His research interests include system identification, ensemble learning, mutual learning, Lyapunov neural networks, and safety of neural networks.



MARTINA LIPPI (Member, IEEE) received the M.Sc. (cum laude) and Ph.D. degrees in information engineering from the University of Salerno, Fisciano, Italy, in 2017 and 2020, respectively.

In 2019, she was a Visiting Scholar with the KTH Royal Institute of Technology, Stockholm, Sweden. From November 2020 to June 2022, she was a Postdoctoral Researcher with Roma Tre University, Rome, Italy, where she has been an Assistant Professor, since June 2022. Her research interests include multi-agent systems, distributed control, human–robot interaction, and agricultural robotics. Since September 2024, she has been an Associate Editor of IEEE Robotics and Automation Letters.



MAURO FRANCESCHELLI (Senior Member, IEEE) received the Laurea (cum laude) and Ph.D. degrees in electronic engineering from the University of Cagliari, Cagliari, Italy, in 2007 and 2011, respectively.

He is currently an Associate Professor at the University of Cagliari, Italy. His research interests include consensus problems, distributed optimization, learning, multiagent systems, nonsmooth analysis, monotone systems, multirobot systems, and distributed energy systems. He is a member of the IEEE RAS Technical Committee on Multirobot Systems and the IEEE CSS Technical Committee on Networks and Communications. He is the Vice Chair of the IEEE Italy Section Chapter of the IEEE Control Systems Society. He serves on the conference editorial board of the Control Systems Society, since 2019, and of IEEE CASE, since 2015. He has been an Associate Editor of IEEE Transactions on Automatic Control, since 2025, and *Nonlinear Analysis: Hybrid Systems* (IFAC), since 2023, and was an Associate Editor of IEEE Transactions on Automation Science and Engineering, from 2021 to 2023.



ANDREA GASPARRI (Senior Member, IEEE) received the Laurea degree (cum laude) in computer science and the Ph.D. degree in computer science and automation from Roma Tre University, Rome, Italy, in 2004 and 2008, respectively.

He is currently a Professor with the Department of Civil, Computer Science and Aeronautical Technologies Engineering, Roma Tre University. His research interests include robotics, sensor networks, networked multiagent systems, and precision agriculture.

Dr. Gasparri has been a member of the Steering Committee for the IEEE RAS Technical Committee on Multirobot Systems, since 2014, the IEEE CSS Technical Committee on Networks and Communications, since 2015, and the IEEE Technical Committee on Agricultural Robotics, since 2021. He was an Associate Editor of IEEE Transactions on Cybernetics, from 2017 to 2021. Since 2021, he has been an Associate Editor of IEEE Transactions on Control of Network Systems.