Roma Tre

Ph.D. in Computer Science and Automation
Department of Engineering

XXXVII Ph.D. Cycle

Data-driven Methodologies for System Identification

Ph.D. Student — Nicola Arridu

.....

Ph.D. Advisor — Prof. Andrea Gasparri

.....

Ph.D. Coordinator — Prof. Fabrizio Frati

.....

Data-driven Methodologies for System Identification

A thesis presented by
Nicola Arridu
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy
in Computer Science and Automation

Roma Tre University
Department of Engineering

Advisors:
Prof. Andrea Gasparri

Reviewers:
Prof. Alessandro Marino (Università degli Studi di Cassino e del Lazio Meridionale)
Prof. Gabriele Oliva (Università Campus Bio-Medico di Roma)

Abstract

System Identification aims to understand the behavior and evolution of physical processes or industrial systems, which are often nonlinear. In the context of modern industrial automation, this task is essential for control, fault diagnosis, monitoring, and performance optimization. These activities require accurate mathematical models capable of describing the nonlinear relationships between input and output. The problem that arises is that determining such models can be complex or not fully possible. In recent years, system identification has addressed this issue through *data-driven* approaches, based on the collection of data from observable quantities of the system—particularly its inputs and outputs. Such data can be used to train neural networks so that they can learn patterns and relationships useful for constructing an analytical model of the system.

This thesis covers two data-driven approaches for the identification of nonlinear systems using neural networks. The first approach is based on an ensemble learning framework, where multiple autoencoder-type neural networks are combined to improve prediction of system's evolution. Through a dedicated procedure, the best autoencoders, according to a score based on performance and diversity respect to other autoencoders, are selected from the ensemble, enhancing the overall performance. The approach was tested on nonlinear benchmarks, showing improved identification performance compared to individual models.

The second approach focuses on the modeling of a thermal process consisting of the cooling dynamics of a thin bread produced in an industrial bakery. The dynamics are identified using a neural network trained with thermal images of the bread, achieving high prediction accuracy and consistency with thermodynamic principles through the integration of physical knowledge into the training process.

Together, these studies contribute to the broader development of data-driven approaches, supporting the development of automation in line with the principles of *Industry 5.0*, with potential applications in small and medium-

sized enterprises located in regions where the technological transformation is currently underway.

Acknowledgments

I would like to thank my advisor, Prof. Andrea Gasparri, for his guidance throughout these years. His support and encouragement, combined with his vision and attention to detail, have been fundamental to the success of this work.

My thanks also go to Prof. Mauro Franceschelli, with whom I have had the pleasure of collaborating for several years. His mentorship has contributed significantly to the development of this thesis.

I am grateful to Dr. Martina Lippi for her indispensable collaboration. Her patience, together with her remarkable scientific expertise, has been crucial to my research.

I dedicate this thesis first and foremost to my family, to my parents, and to my brother. Everything I have accomplished would not have been possible without you.

Thank you to all my colleagues and friends who have been close to me during these years between Rome and Cagliari. I'm sorry, but I cannot express my gratitude in just a few words, each of you deserves a poem. I will make it up to you when we meet again.

Finally, I want to dedicate this thesis to my chess coach, Grandmaster Daniel Naroditsky, and to all those who have fought against something greater than themselves.

Contents

Chapter 1

Introduction

1.1 Motivation and context

Automation consists in the use of machines to perform tasks that were previously carried out by humans, or tasks that would not be possible without machines [1]. The advantages brought by automation are significant, first and foremost an increase in productivity, particularly in environments that previously required a large workforce. In many sectors, the lack of labor has been a factor leading to the rise of automation, especially in rural and sparsely populated areas. The reduction in the use of manual labor for critical activities has led to improved safety conditions for workers in industry and agriculture. Automation is also aligned with the demand for greater environmental sustainability in industrial and agricultural activities, since it is associated with resource optimization and a consequent reduction in waste and pollutant emissions.

In recent years, automation has been a key global factor in the pursuit of greater efficiency and competitiveness. Many reports, including [2], indicate that automation is transforming production processes and employment within them. That said, the adoption of new technologies is not yet advanced in many contexts. As pointed out in [3], the technological innovation of a company often depends on its size. Therefore, making new technologies more accessible can have a major impact, especially in areas that have historically not been affected by technological innovations or by large industries.

The concept of automation cannot refer only to the replacement of human labor by machines but also implies a system of *self-governance*. This deeply

connects automation with control systems theory. In general, a control system for an industrial process requires a set of sensors to monitor the state of the system and its surroundings, actuators to make the system act in a given way, and—most importantly—a controller that automatically decides how the system should behave. For a controller to be effective, it is essential that it "knows" the system in order to direct it, through the formulation of a mathematical model of the system. It is on this premise that the problem of *System Identification* is based, namely finding reliable models in order to establish an appropriate control strategy. This work aims to address this problem, as will be shown in the following chapters.

Traditionally, system identification relied on techniques based on series expansions or on partial knowledge of the system. These methodologies have shown limitations when applied to the identification of complex and highly nonlinear systems. In recent years, *data-driven* approaches [4], [5] have become particularly widespread, thanks to increased computational power and access to an ever-growing amount of data. A major contribution has come from the enormous development of research and investment in the fields of machine learning and neural networks. Neural networks are powerful and flexible tools capable of learning the relationships that characterize a system, proving in many contexts their effectiveness and adaptability for nonlinear systems.

The first part of this research falls within the field of system identification using neural networks. The idea is not to rely on a single network but to improve identification accuracy through the ensemble learning paradigm [6], [7]. According to this paradigm, the outputs of several neural networks are combined to obtain a more reliable representation. This approach makes it possible to reduce overfitting on training data and to increase robustness to noise, achieving greater fidelity in system identification.

The second part focuses on the modeling of a real thermal process, thus considering a practical application related to the cooling of bread in an industrial bakery context. In this scenario, the techniques used to understand the evolution of temperature are based on classical models such as the *Verma* model, which are characterized by parameters that depend on environmental conditions. Given the difficulty that classical optimization methods face in determining these parameters each time such conditions change, the use of neural networks makes it possible to learn how these parameters vary directly from data, incorporating the physical knowledge of the process into the training phase through physics-informed loss functions.

The purpose of this thesis is to contribute to the development of automatic, accurate, and robust methods, particularly suited for small-scale industrial sys-

tems where resources and expertise for advanced automation are often limited. The first contribution demonstrates the stability of ensemble learning methods for system identification, while the second contribution shows how neural networks can model the physical laws governing thermal processes. These two research directions are aligned with the *Industry 5.0* [8] framework, which emphasizes sustainability, human–machine collaboration, and the use of technology for human well-being.

1.2 Thesis Contributions

In this section, we present the main contributions of the thesis, divided according to the two research directions undertaken. Both share a common vision: to develop methods that are accurate, reliable, and robust for the data-driven identification of nonlinear systems, using data obtained from observations of physical systems or processes whose evolution we aim to study, and employing neural networks for their adaptability to complexity and nonlinearity.

1.2.1 Ensemble Learning for System Identification

The work proposes an ensemble learning framework to overcome the limitations of nonlinear system identification with neural networks, namely the variability and instability arising from the training process. The architecture considered for each individual network is based on that of an *autoencoder* [9], [10], and includes an encoder, a decoder, and a network that models the system dynamics. Overall, the autoencoder is trained to capture the nonlinearities and uncertainties of a dynamical system using data from experimental observations of the system. The approach consists in jointly using multiple autoencoders to improve accuracy and generalization with respect to data not included in the training set. The effectiveness of the ensemble is further enhanced by the design of the *Ensemble Component Selection Algorithm* (ECSA), which selects the autoencoders based on an *accuracy–diversity* trade-off. Finally, the proposed method proves to be more effective than individual models in the identification of two nonlinear dynamical systems, namely the Two-Tank and Hammerstein–Wiener systems.

1.2.2 Thermal Modeling of Carasau Bread

This study focuses on the application of system identification techniques to a practical case consisting of the modeling of the cooling of the *Carasau* bread,

that is a typical Sardinian thin bread, in a production line. Among the main outcomes of this research, we have the design of an automatic acquisition system based on thermal cameras to measure the thermal decay of the bread. Following a data-driven approach, the collected thermal data is used to train a neural network with the aim of determining, in real time, the parameters of the *Verma* model, that is, a model used to describe the cooling dynamics. The neural network makes it possible to flexibly obtain the model parameters as environmental conditions change, incorporating physical knowledge into the loss function. The method is then evaluated on real data from an industrial context, confirming both the accuracy and the physical interpretability of the physics-informed network.

1.3 Thesis Outline

The remainder of this thesis is organised as follows.

Chapter 2 presents the theoretical foundations of the thesis, starting from the concept of a nonlinear dynamical system. The problem of system identification and the main approaches found in the literature are then introduced. Given its importance for the rest of the thesis, one section is dedicated to neural networks and their ability to approximate a given function, and another to the ensemble learning method and its properties. Finally, the chapter presents the analytical models used to study the thermal processes related to the baking of Carasau bread.

Chapter 3 discusses the use of ensemble learning for the identification of nonlinear systems. It presents the proposed framework and analyses its performance on representative nonlinear benchmarks, highlighting the benefits of ensemble-based identification compared to single models.

Chapter 4 describes the procedure that led from data acquisition with thermal cameras to the neural network modeling of the cooling process of pane carasau. In particular, it is shown how the neural networks are trained by incorporating the physical intuition provided by the Verma model into the loss function. It concludes with an analysis of the obtained models and their relevance for automation in an industrial context.

Finally, **Chapter 5** constitutes the conclusion of the thesis. The chapter summarizes the main contributions of the studies carried out and discusses their applicability to the automation of small and medium-sized industries. Afterwards, the directions for future research are outlined, including the combination of ensemble and physics-informed methods, as well as the application of system identification method as a preliminary phase to the definition of control laws.

Chapter 2

Preliminaries

2.1 System Identification Fundamentals

The scope of this section is to provide the basic notions related to *System Identification*, to let the reader understand in a more comprehensive way the problems illustrated in Chapter 3 and Chapter 4.

2.1.1 Dynamical Systems

Before address System Identification problems and challenges, it is necessary to give a definition of *System*. According to the *IEEE Standard Dictionary of Electrical and Electronics Terms* [11], a system is a set of interconnected elements that achieve a given objective through the performance of a specified function. We can study the interactions between these elements by the *observable signals* produced by the system [12]. We can call these signals *outputs*. A system could be conditioned by *external signals*. If an external signal is measurable and can be handled by the user, we call it *input*, otherwise if it is measurable or not and, more important, it can not be manipulated by the user, we call it *disturbance*.

We call a system *dynamical*, if the system's outputs evolve responding to the inputs, over time. Let's assume for simplicity that the system has a single output conditioned by only one input without disturbances, we have that the dynamical evolution of the output can be described by the following differential

equation:[1]

$$a_n \frac{d^n y(t)}{dt^n} + a_{n-1} \frac{d^{n-1} y(t)}{dt^{n-1}} + \cdots + a_0 y(t) = b_m \frac{d^m u(t)}{dt^m} + b_{m-1} \frac{d^{m-1} u(t)}{dt^{m-1}} + \cdots + b_0 u(t), \quad (2.1)$$

where t is the time variable, $y(t) \in \mathbb{R}$ and $u(t) \in \mathbb{R}$ denote, respectively, the output and input signals of the system, $\frac{d^i y(t)}{dt^i}$ and $\frac{d^i u(t)}{dt^i}$ indicate the i-th time derivative of, respectively, $y(t)$ and $u(t)$, n is the maximum degree of output's derivation, m is the maximum degree of input's derivation and $a_i, b_j \in \mathbb{R}$ are coefficients describing the system dynamics.

Since in a real system identification problem, the observable signals are measured and sampled every Δ seconds, with Δ called sample time, we can describe the evolution of the output respect the input in *discrete-time* with a difference equation:

$$y_k + a_1 y_{k-1} + a_2 y_{k-2} + \cdots + a_n y_{k-n} = b_1 u_{k-1} + b_2 u_{k-2} + \cdots + b_m u_{k-m}. \quad (2.2)$$

where k is the discrete-time variable. In the following of this thesis, we use a discrete-time representation, as in (2.2), as oppose to the continuous-time representation in (2.1), due to the fact that our data-driven approach considers data collected by sampling.

Between inputs and outputs, we can introduce the *state* variable, gathering in it the information on the past evolution. With this introduction, we can rewrite (2.2) in a more compact form, passing from an higher-order input–output relationship to a first-order vector equation. We have:

$$\begin{cases} x_{k+1} = f(x_k, u_k), \ x \in \mathbb{R}^n, \\ \\ y_k = g(x_k), \ y \in \mathbb{R}^{n_o} \end{cases} \quad (2.3)$$

where x is the state, f is the function denoting the dynamics of the state, n is the dimension of state and it is called the system's order, g is function that describes the relation between the state and the output y_k, and n_o is the output dimension. This *state variable* model is often more convenient for analysis and control design. The functions f e g are in general nonlinear, in this case we call the dynamical system nonlinear. In the following chapters of the thesis, we focus on this kind of systems.

[1] This formulation assumes lumped parameters and stationarity; see [13], [14] for detailed discussions.

2.1.2 Modeling and identification

In system theory, we refer to *modeling* as the process of building a mathematical model for the system [12]. The process of constructing a model like (2.3) can be conducted through the study of the subsystems composing the system, whose key parameters and the knowledge of the underlying physical laws are needed to the overall description of the system. This procedure is application dependent and it is strongly connected to the techniques used in the application area and to the specific user's experience in it. *System identification* is a branch of modeling that uses data-driven techniques to estimate a model for the system, with a partial or an absent knowledge of the internal structure, relations and components of the system. Mathematically, we have that given a dataset from observed signals $\mathcal{D} = \{(u_k, y_k)\}_{k=1}^{N}$, we want to find a model $\mathcal{M}_\theta$ to reproduce the input-output relation. Considering the model parameterized by θ, we aim to find θ such that the error

$$E(\theta) = \sum_{k=1}^{N} \|\hat{y}_k - y_k\|^2, \tag{2.4}$$

is minimum, meaning that the model prediction $\hat{y}_k = \mathcal{M}_\theta(u_k)$ reproduce the output of the system y_k.

In practice, we can have a mixture between modeling from physical insight and identification based on data. Depending on the level of knowledge, we can distinguish different approaches in system identification [15]:

- **White-box approaches**, where models are constructed from the governing physical laws of the system, and only some parameters are derived from data.
- **Black-box approaches**, where models are derived uniquely from data. This means that no prior knowledge of the system is needed and that the model has not a direct relation with physical laws.
- **Gray-box approaches**, that are an intermediate version between the former two types of approaches. Typically, the model structure is deducted from physical insights, whereas key parameters are chosen through data.

In the following we present classical system identification methods known in literature.

Volterra series method [16]. This method considers a series of polynomial operators to model a nonlinear dynamical system. Mathematically, the system output is expressed as a functional expansion of convolutions between the input signal and a set of Volterra kernels:

$$y_k = h_0 + \sum_{p=1}^{P} \sum_{\tau_1=0}^{M} \cdots \sum_{\tau_p=0}^{M} h_p(\tau_1, \ldots, \tau_p) \prod_{i=1}^{p} u_{k-\tau_i}, \tag{2.5}$$

here h_0 is a constant term, $h_p(\tau_1, \ldots, \tau_p)$ denotes the p-th-order Volterra kernel, and M and P represent the system memory and the highest nonlinearity order considered, respectively. Since the Volterra kernels have not a physical meaning, the approach is classified as a black-box method. Although the method has been applied with success in many fields, its limitations resides mainly in the number of coefficients that grows combinatorially with the system order and the memory length, which restricts its practical use to low-order nonlinearities, and to the requirement of using inputs with the ability of stimulate all the nonlinearities.

Block-structured systems method [17]. This approach is based on the interconnection of static nonlinearities and basic linear dynamic blocks. The method belongs to the category of gray-box approaches because the choice of blocks interconnections is in accordance with the prior knowledge of the system, whereas the parameters of the blocks are estimated from data. The use of this method is motivated by the fact that many real system can be described by a nonlinearity due to actuators at input, followed by a linear system representing the actual dynamics, and with a nonlinearity in the output related to nonlinear characteristics of sensors for the measurement. The main drawback of the approach is in the difficulty in structure selection. Moreover, some systems may contain multiple nonlinearities that cannot be captured by a simple cascade structure, and the nonlinear blocks are usually parameterized with simple basis functions (e.g., polynomials, piecewise-linear, or sigmoids) that are not adapt to capture strong nonlinearities.

NARMAX method [18]. The Nonlinear AutoRegressive Moving Average model with eXogenous inputs (NARMAX) method considers a finite number of past values of the system output y_k, that is $y_{k-1}, y_{k-1}, \ldots, y_{k-n_y}$, and a finite number of past values of the system input u_k, that is, $u_{k-1}, u_{k-1}, \ldots, u_{k-n_u}$.

From these values, it reconstructs y_k as follows:

$$\hat{y}_k = h(y_{k-1}, y_{k-1}, \dots, y_{k-n_y}, u_{k-1}, u_{k-1}, \dots, u_{k-n_u}),$$

with $f(\cdot)$ the NARX function and $\hat{y}_k$ the reconstruction of the real output y_k. The module is optimized during training to minimize the difference between $\hat{y}_k$ and y_k. Because the structure is entirely determined by data, NARMAX models belong to the black-box category, with a representation that can approximate a broad class of discrete-time nonlinear system. Despite this flexibility, the limitations of the method are several. In particular we cite the combinatorial growth of nonlinear terms in h with the system order, and that it has the tendency to not generalize beyond the range of input and output values covered in the data collection.

Challenges and limitations The approaches reviewed above have in common some non-negligible limitations when applied for the identification of complex dynamical systems. Thea main difficulties encountered are connected to nonlinearity: the presented methods can approximate weak nonlinear behaviors, but their accuracy decays for strongly nonlinear processes. The performance of these methods is greatly connected to the model order, with a possible combinatorial increase of parameters, and consequently a growing risk of overfitting. For these reasons in the following section, we introduce *neural networks* for system identification. Neural networks can learn nonlinearities through the adjustments to its structure done in the training phase, without a combinatorial explosion of its parameters. Moreover, they can can integrate physical knowledge, like (e.g., constraints, differential equations, conservation laws), as in the case of Physics Informed Neural Network [19].

2.2 Neural Networks for System Identification

In this section, we present the main result that makes neural networks suitable for system identification and a training method that allows a neural network to learn the characteristics of a system. Then, we introduce the autoencoder, a type of neural network useful for system identification, considered in Chapter 3.

2.2.1 Neural networks as universal approximators

A *neural network* is a nonlinear map $f_\theta : \mathbb{R}^{n_{\text{in}}} \to \mathbb{R}^{n_{\text{out}}}$ characterized by a set of parameters θ. We consider the network composed of L layers, each one

performing the transformation:

$$f_\ell(x_{\ell-1}) = x_\ell = \sigma(W_\ell x_{\ell-1} + b_\ell)\,, \quad \ell = 1, \dots, L, \tag{2.6}$$

where:

- x_ℓ is the input to the ℓ-th layer, with x_0 being the overall network input;
- W_ℓ and b_ℓ are the weight matrix and bias vector of the ℓ-th layer, both collected in θ;
- $\sigma(\cdot)$ is a nonlinear activation function applied elementwise.

From the last layer L, the network output is obtained as $y = f_\theta(x_0) = x_L$, where $f_\theta := f_L \circ \cdots \circ f_1$ is defined as the composition of affine transformations and activation functions. By adjusting the parameters in θ, the function f_θ can approximate highly nonlinear mappings between input and output variables.

We now state the **Universal Approximation Theorem** [20], [21].

Theorem 2.1 (Universal Approximation Theorem [20], [21]) *Let $K \subset \mathbb{R}^{n_{in}}$ be a compact set and let $C(K)$ denote the space of continuous functions on K. Let $\sigma : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous activation function. Then, for every $f \in C(K)$ and for every $\varepsilon > 0$, there exists a neural network $f_\theta : \mathbb{R}^{n_{in}} \to \mathbb{R}^{n_{out}}$ of the form*

$$f_\theta(x) = \sum_{i=1}^{N} \alpha_i\, \sigma(w_i^\top x + b_i), \tag{2.7}$$

with finite N, weight vectors $w_i \in \mathbb{R}^{n_{in}}$, biases $b_i \in \mathbb{R}$, and output coefficients $\alpha_i \in \mathbb{R}$, such that

$$\sup_{x \in K} \big| f(x) - f_\theta(x) \big| < \varepsilon. \tag{2.8}$$

The Universal Approximation Theorem implies that the neural network can represent the mappings denoting the nonlinear system in (2.3), without any prior knowledge of their analytical form. The theorem certify the existence of the parameters that let the network represent complex nonlinear relationships, but this set of parameters has to be determined, learning the attributes of the system from data. We now present a training method to learn temporal dependencies and nonlinearities.

Truncated Backpropagation Through Time Neural networks are typically trained using an algorithm based on *backpropagation* [22], which computes parameter gradients through the chain rule and updates the model parameters to minimize a predefined loss function. Let $\mathcal{D} = \{(u_k, y_k)\}_{k=1}^{N}$ denote a dataset containing the observed input–output pairs of the system. A major limitation of applying standard backpropagation to system identification problems is that the samples in $\mathcal{D}$ are usually assumed to be independent, whereas in dynamical systems, each observation depends on past states and inputs. When learning a model that evolves in time, the error at a given instant k therefore depends on previous values of the input and the state variables. To properly account for these temporal dependencies, we employ the *Backpropagation Through Time* (BPTT) algorithm [23], which computes the parameter gradients by considering the complete temporal sequence contained in $\mathcal{D}$. The drawback of this approach resides on the computational burden that scales linearly with the sequence length, becoming prohibitive for long sequences. For this reason the *Truncated Backpropagation Through Time* (TBPTT) [24] approach has been developed. In this way, the sequence is divided in chunks of a predefined lenght. The gradient computation and update of parameters are limited to each chunk and not to the all sequence, reducing considerably the computational effort. TBPTT allows the the identification of nonlinear dynamical systems over extended datasets, and for this reason was considered for the training of *autoencoders* in the ensemble framework presented in Chapter 3.

2.2.2 Autoencoders for State-Space Modeling

Autoencoders [25] are neural architectures widely used for unsupervised representation learning and have been successfully applied to nonlinear system identification [10]. 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 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 in the ensemble, 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 $\hat{I}_i$.

The encoder network takes the input data (e.g., time series data representing system inputs or outputs) and maps it to lower-dimensional latent space representation, whereas the decoder network attempts to reconstruct the input data from the compressed representation, that is,

$$Z_i = f_{\text{enc},i}(I), \quad \forall i = 1, 2, \ldots, N, \tag{2.9}$$

$$\hat{I}_i = f_{\text{dec},i}(Z_i), \quad \forall i = 1, 2, \ldots, N. \tag{2.10}$$

In general, an autoencoder is trained to make the reconstructed output $\hat{I}_i$ as close as possible to the input I. This notation will be used in Chapter 3, where autoencoder models are extended to an ensemble configuration.

2.3 Ensemble Learning

In practice, we assist often to the condition when a single neural network is not capable of understanding the properties of a nonlinear system. The main reasons of this problems have to be sought in the learning process, where the training dataset can lead to bias, especially if the considered input signals are not sufficient to guide the system in all the significant operative conditions. Moreover, it is possible that the architecture chosen for the neural network is too shallow, and a more complex structure should be considered. The question is addressed by *Ensemble learning*, that consists in combine the output of more neural networks through a mathematical operator such as averaging, obtaining an increased accuracy in the identification [6], [26]. If we consider N_E neural networks aggregated with an average operation, we have that the ensemble is

denoted by the following function:

$$F_E(x) = \frac{1}{N_E} \sum_{i=1}^{N_E} f_{\theta_i}(x), \tag{2.11}$$

where f_{θ_i} is the output of the i-neural network and x is the input to all the networks.

In general, an ensemble of neural networks guarantees a better performance because the ensemble's components made errors that are statistically independent, then when a model fails, the other models compensate ensuring accuracy.

2.3.1 Diversity and the Bias–Variance–Diversity Trade–off

The behavior of the single components in the ensemble is crucial for its performance, in particular the difference between their behaviors is a key factor. We refer to this difference with the concept of *diversity* [27], [28]. Intuitively, diversity plays a fundamental role given that the ensemble would exhibit the same predictions of a single model, if all the ensemble's components show the same exact behavior.

Let's consider the scenario of a neural network making a prediction on the output of a system to identify, let's define the *bias* as the measure of the distance between prediction and real value and the *variance* as the measure on how much the prediction fluctuates depending on the training process. Wood et al. [28] demonstrated that a loss function, that computes the difference between ensemble's prediction and real data, can be decomposed in the following way:

$$\mathbb{E}[\ell(Y, q)] = \text{Average Bias} + \text{Average Variance} - \text{Diversity}, \tag{2.12}$$

where the first term is the average bias between ensemble's components, that can be seen as the systematic error of the ensemble, the second term is the average variance, that can be seen as the variability of the ensemble's prediction, and the third term is in fact the diversity, that measures the disagreement among single components. We can notice how an increase of diversity corresponds to a decrease of loss and a corresponding better ensemble's prediction. The problem that arises is that we cannot simply improve performance maximizing diversity. That can lead to instability and to a deterioration of single network performance, and consequent decrease of the ensemble effectiveness. In this way, we need a bias-variance-diversity trade-off, with the ways of design it being an open subject in ensemble learning theory.

Ensemble learning and the concept of diversity have been studied in a practical case in Chapter 3.

2.4 Thermal Process Modeling

In this section, we present the models that govern the cooling of foods with a thin thickness. The following discussion is based on [29]. When the thin-layer assumption is valid, as in the case of pane carasau, the thermal process can be described as a heat transfer between the bread and the surrounding air. Under this assumption, we model the bread as a sheet whose radius is much larger than its thickness. Considering additional assumptions such as homogeneity, isotropy, negligible shrinkage, and thermal convection[2], the heat concentration in the product $\tau(x,t)$ is described by the following equation:

$$\frac{\partial \tau(x,t)}{\partial t} = D\frac{\partial^2 \tau(x,t)}{\partial x^2}, \tag{2.13}$$

where x is the spatial coordinate, t is time, and D is the diffusivity constant. This equation is known in literature as Fourier's second law of heat transfer. If we assume a uniform initial temperature distribution and heat transfer occurring only through one surface, while the other is assumed to be at the thermal equilibrium temperature τ_e (assumed constant $\tau_e(t) \approx \tau_e(0) =: \tau_e$ since the cooling process is much faster than ambient temperature changes), we can define the temperature ratio as:

$$\tau_r(t) = \frac{\bar{\tau}(t) - \tau_e}{\tau_0 - \tau_e}, \tag{2.14}$$

where $\bar{\tau}(t)$ is the average temperature inside the product. The analytical solution of (2.13) for a sheet of thickness h is:

$$\tau_r(t) = \sum_{i=1}^{\infty} \frac{8}{\pi^2(2i-1)^2} \exp\left[-\frac{(2i-1)^2\pi^2 D}{4h^2}t\right]. \tag{2.15}$$

This result is not easily usable due to the difficulty in accurately determining the thickness h, the diffusivity D, and the equilibrium temperature τ_e. For these reasons, the adopted approach consists of neglecting the higher-order terms, obtaining a finite sum of exponentials:

$$\tau_r(t) \approx \sum_{i=1}^{n} A_i \exp(-k_i t), \tag{2.16}$$

[2]We refer the reader to [29] for a more rigorous discussion of the assumptions considered.

where A_i and k_i are parameters that, in practice, are estimated from experimental data. Among the models derived from (2.16), we find the Verma model [30]:

$$\tau_r(t) = ae^{-\lambda_0 t} + (1-a)e^{-\lambda_1 t}, \tag{2.17}$$

where

- $\tau_r(t) \in [0,1]$ denotes the dimensionless temperature, with $\tau_r(0) = 1$ and $\lim_{t\to\infty} \tau_r(t) = 0$;
- $\lambda_0, \lambda_1 \geq 0$ are the two dominant modes of the system, representing the decay rate at the beginning (fast) and at the end (slow) of the process;
- $a \in [0,1]$ is a coefficient that weights the contribution deriving from the two modes.

Since a, λ_0, and λ_1 stem from simplifications such as assuming constant diffusivity and uniform material properties, the model may become unreliable when environmental conditions change or in the presence of external disturbances. For this reason, adaptability of the Verma model parameters is required. Neural networks can address this need, as through a data-driven approach they can capture the nonlinearities and disturbances that make it necessary to adjust the model parameters. This aspect will be further explored in Chapter 4.

Chapter 3

Ensemble Learning for System Identification

This chapter is based on the paper:
N. Arridu, M. Lippi, M. Franceschelli and A. Gasparri,
"On Ensemble Learning Models for Autoencoder-Based Identification of Nonlinear Dynamical Systems",
in *IEEE Access*, vol. 13, pp. 169071-169082, 2025.

3.1 Introduction

Identifying the dynamic model of a system has been a subject of extensive research for decades. As discussed in Chapter 2, traditional identification methods rely on analytical or parametric model structures [12], while modern data-driven approaches exploit measured input–output data to directly infer system dynamics. This chapter focuses on the use of machine learning (ML) techniques for nonlinear system identification.

In recent years, there has been an expansion of machine learning methods in system identification. This is linked to greater computational power and the availability of large datasets [5], [10], [31]. These methods have the ability to capture the dynamics of nonlinear systems and to scale with the increasing size of the system to be identified [32], [33], [34]. In addition, they potentially offer adaptability features [35], [36], [37], allowing the identified model to adjust to environmental changes, the time-evolving nature of the systems, or intentional

modifications when provided with sufficient or new data.

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 [7] can be leveraged. More specifically, the goal of ensemble learning is to train multiple models and properly combine their outputs to achieve higher accuracy and robustness than individual networks [6], [38]. 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 [28], [39], [40].

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. The numerical results of the standard benchmarks corroborate the proposed ensemble architecture.

Related works 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 [41] 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 [42] 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 [43] and [44]. In more detail, the authors of [43]

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 [44], 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 [10] 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 [45] 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 [46] 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 [31]. The proposed method addresses the trade-off between representational capacity and overfitting, providing a balanced approach for model reduction and accurate dynamics learning. Finally, the work in [47] 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. [48] 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. [36] 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.

Position of This Work In light of the limited adoption of ensemble learning techniques in system identification, this chapter investigates a neural ensemble framework based on autoencoder architectures. Building on the approach proposed by Masti and Bemporad [10], which employs individual autoencoders to identify nonlinear state–space models, the present framework combines multiple models to enhance identification accuracy and robustness. This chapter investigates an ensemble learning method for nonlinear SI using autoencoder architectures. By building on the work in [10], which considers individual autoencoder models for SI, multiple models are combined in an ensemble to improve the identification accuracy. Furthermore, the present framework is equipped with an algorithm to select the models for inclusion in the ensemble based on an accuracy metric. The proposed approach is then validated using two two-tank and the Hammerstein-Wiener SI problems.

To the best of the authors' knowledge, only the works in [36] and [48] leveraged ensemble learning techniques in the context of nonlinear SI.

The study in [36] 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. [48] 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, the approach developed in this chapter relies on 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 introduced 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.

3.2 Notation

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

$$x_{k+1} = f(x_k, u_k),\ x \in \mathbb{R}^{n_x}, \tag{3.1}$$

$$y_k = g(x_k),\ y \in \mathbb{R}^{n_o}, \tag{3.2}$$

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 $g : \mathbb{R}^{n_x} \to \mathbb{R}^{n_o}$ were assumed to be *unknown*. The goal of system identification is to learn suitable approximations of these two functions.

Ensemble of autoencoders The ensemble concept introduced in Chapter 2 is here specialized to the case of autoencoder networks, presented in the same chapter. Let us consider an ensemble of N autoencoders, indexed by $i = 1, 2, \ldots, N$. Given the representations of the individual autoencoders, these can be combined as follows to achieve an *ensemble representation*:

$$Z_{\text{en}} = \frac{1}{N}\sum_{i=1}^{N} Z_i = \frac{1}{N}\sum_{i=1}^{N} f_{\text{enc},i}(I), \tag{3.3}$$

$$\hat{I}_{\text{en}} = \frac{1}{N}\sum_{i=1}^{N} \hat{I}_i = \frac{1}{N}\sum_{i=1}^{N} f_{\text{dec},i}(Z_i), \tag{3.4}$$

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 reconstruction error, that is, to make $\hat{I}_{en}$ as close as possible to I. The average operation is employed to aggregate the representations of individual autoencoders but it is worth remarking that, as highlighted in [7], different combinations can be employed in ensemble functions beyond the average operation. In the following section, we present how individual autoencoders and ensembles of autoencoders can be exploited for SI.

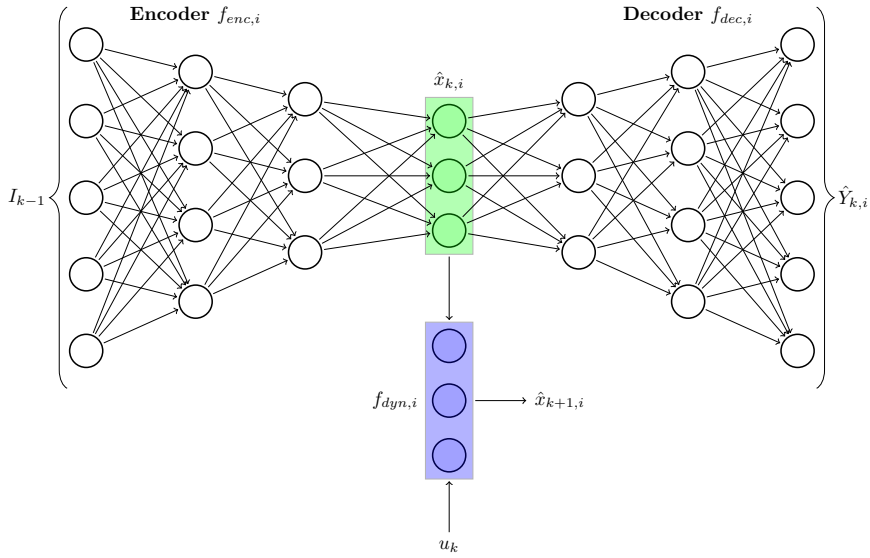


Figure 3.1: Representation of an autoencoder extended with the dynamics model.

3.3 Data-driven SI framework

In this section, we present the structure of the dataset used to identify the system model. Then, by building on [10], 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.

3.3.1 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:

$$\mathcal{D} = \{(y_{k_f}, u_{k_f}), (y_{k_f-1}, u_{k_f-1}), \ldots . (y_{k_0}, u_{k_0})\}, \tag{3.5}$$

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 (3.1) and (3.2), 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,

$$I_k = [y_k, y_{k-1}, \ldots, y_{k-n_I+1}, u_k, u_{k-1}, \ldots, u_{k-n_I+1}]^T. \tag{3.6}$$

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,

$$Y_k = [y_k, y_{k-1}, \ldots, y_{k-n_Y+1}]^T. \tag{3.7}$$

3.3.2 Autoencoder-based model and training for SI

To estimate the unknown functions in (3.1) and (3.2), we consider that each autoencoder-based model is composed of the modules in Figure 3.1:

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 [10], we first modify the equations presented in subsection 2.2.2 to adapt the autoencoder i to the SI purposes as follows

$$\hat{x}_{k,i} = f_{\text{enc},i}(I_{k-1}), \tag{3.8}$$

$$\hat{Y}_{k,i} = f_{\text{dec},i}(\hat{x}_{k,i}), \tag{3.9}$$

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 he 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 is the estimate of the system outputs in the finite-time

horizon, i.e., it estimates the vector Y_k. The equations (3.8) and (3.9) would define the classical autoencoder as defined in in subsection 2.2.2, except for the fact that the decoder in (3.9) 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 [49], [50], [51]. Regularization methods in the field of system identification were considered in [10], [31], [41].

As far as the dynamics modeling neural network $f_{\text{dyn},i}(\cdot)$ is concerned, this aims to model the dynamics f. 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

$$\hat{x}_{k+1,i} = f_{\text{dyn},i}(\hat{x}_{k,i}, u_k). \tag{3.10}$$

For each individual model, the autoencoder and the dynamics modeling neural network were 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:

$$\min_{f_{\text{enc},i}, f_{\text{dec},i}, f_{\text{dyn},i}} \quad \sum_{k=k_0+n_I}^{n_D+1} \mathcal{L}_{i,k}, \tag{3.11}$$

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,

$$\mathcal{L}_{k,i} = \alpha \cdot \mathcal{L}^r_{k,i} + \beta \cdot \mathcal{L}^t_{k,i} + \gamma \cdot \mathcal{L}^d_{k,i}, \tag{3.12}$$

where 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

$$\mathcal{L}^r_{k,i} = L(Y_k, \hat{Y}_{k,i}) + L(Y_{k+1}, \hat{Y}_{k+1,i}), \tag{3.13}$$

where L being a generic loss function, such as an *L1 loss* or a *L2 loss* [52]. 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

$$\mathcal{L}^t_{k,i} = L(f_{\text{dyn},i}(\hat{x}_{k,i}, u_k), \hat{x}_{k+1,i}). \tag{3.14}$$

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,

$$\mathcal{L}^d_{k,i} = L(Y_{k+1}, f_{\text{dec},i}(f_{\text{dyn},i}(\hat{x}_{k,i}, u_k))). \tag{3.15}$$

Note that the problem in (3.11) is a one-step ahead optimization formulation, which might not capture long-term dependencies in the data and result in low prediction capabilities [10]. To improve these capabilities, as discussed in Chapter 2, 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) method is integrated, which achieves multi-step ahead prediction in an efficient manner by propagating the state estimate in a limited number of steps $F << n_D$, as detailed in [10]. Note that the case $F = 1$ leads to the one-step ahead formulation in (3.11).

3.3.3 Ensemble of Autoencoder-based models for SI

At this point, we present how an ensemble of autoencoder-based models can be leveraged for system identification. More specifically, by considering the data in $\mathcal{D}$, we adapted the expression of the ensemble equations in (3.3)-(3.4) as follows:

$$\hat{x}_{k,en} = \sum_{i=1}^{N} w_i \, f_{\text{enc},i}(I_{k-1}), \tag{3.16}$$

$$\hat{Y}_{k,en} = \sum_{i=1}^{N} w_i \, f_{\text{dec},i}(\hat{x}_{k,i}), \tag{3.17}$$

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 trained the N autoencoder-based models and then combined them as reported above. This approach is commonly adopted, as shown in [36] and [48].

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 mentioned in Chapter 2. 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 (3.16)-(3.17) yields the same result as that obtained using a single autoencoder. Motivated by this consideration, this chapter introduces algorithm aimed at identifying the most effective components within the ensemble for use in (3.16)-(3.17). 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 was investigated in Section 3.5.

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.

3.4 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 Figure 3.2. 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.,

$$\mathcal{V} = \{(y_{k_{f,\mathcal{V}}}, u_{k_{f,\mathcal{V}}}), (y_{k_{f,\mathcal{V}}+1}, u_{k_{f,\mathcal{V}}+1}) \ldots, (y_{k_{0,\mathcal{V}}}, u_{k_{0,\mathcal{V}}})\}, \tag{3.18}$$

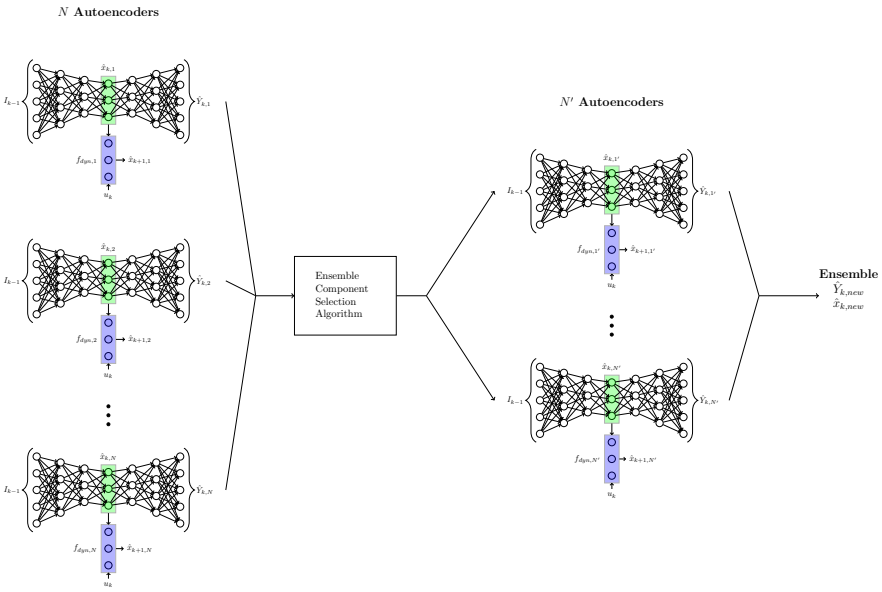


Figure 3.2: Graphic representation of the selection carried out by the Algorithm 1, from an ensemble of N autoencoders to an ensemble of N' autoencoders.

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 (3.5). 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 computed as follows:

$$NRMSA_i = 1 - \frac{\sqrt{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}}(\hat{y}_{k,i} - y_k)^2}}{Y_{max}} \tag{3.19}$$

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 (3.9), 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 (3.19) 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 (3.19) 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

$$c_{ij} = \frac{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}} \left(\hat{y}_{k,i} - \bar{Y}_i\right)\left(\hat{y}_{k,j} - \bar{Y}_j\right)}{\sqrt{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}} \left(\hat{y}_{k,i} - \bar{Y}_i\right)^2} \cdot \sqrt{\frac{1}{n_T}\sum_{k=k_{0,\mathcal{V}}}^{k_{f,\mathcal{V}}} \left(\hat{y}_{k,j} - \bar{Y}_j\right)^2}} \tag{3.20}$$

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 [53], 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 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 (3.17).

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 $\{1, ..., N\}$ (line 2). Next, we computed 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 re-

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 */`

1 $list_{en} \leftarrow \{1, \cdots, N\}$;
2 **for** $i \in \{1, \cdots, N\}$ **do**
3 | $NRMSA_i \leftarrow$ Calculate NRMSA for model i;
4 **end**
5 $corr \leftarrow$ Compute correlation matrix;

`/* Phase 2: Score calculation */`

6 $list_{scores} \leftarrow \{\}$;
7 **for** $i \in \{1, \cdots, N\}$ **do**
8 | $s_i \leftarrow$ compute similarity for model i;
9 | $v_i \leftarrow \omega * NRMSA_i - \rho * s_i$;
10 | $list_{Score} \leftarrow$ Add value $score_i$;
11 **end**

`/* Phase 3: Selection */`

12 $list_{selected} \leftarrow$ select the N' models with the highest scores in ist_{Score};
13 $W \leftarrow$ create N' weights for the models in $list_{selected}$;

sulting 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, 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 [27], [28], 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 (3.17), where each weight w_i is obtained as

$$w_i = \frac{1}{\sum_{j \in list_{selected}} v_j} v_i. \tag{3.21}$$

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

3.5 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.

3.5.1 Validation setting

As highlighted in [10], it is beneficial not only to search for the nonlinear representation (as in (3.1)) 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 [10].

The quasi-LPV form is given by:

$$\begin{cases} x_{k+1} = A(x_k, u_k) \begin{bmatrix} {x_k}^T & 1 \end{bmatrix}^T + B(x_k, u_k) u_k, \\ \\ y_k = C(x_k, u_k) \begin{bmatrix} {x_k}^T & 1 \end{bmatrix}^T, \end{cases} \tag{3.22}$$

where A, B, and C are the matrices output by a fully connected neural network. We refer to the modes in (3.1) and (3.2) as the Nonlinear State Space (NSS) model.

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

$$S = \begin{cases} x_{k+1,1} = x_{k,1} - k_1 \sqrt{x_{k,1}} + k_2 u_k, \\ x_{k+1,2} = x_{k,2} + k_3 \sqrt{x_{k,1}} - k_4 \sqrt{x_{k,2}}, \\ y_k = x_{k,2}, \end{cases} \tag{3.23}$$

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 [54] and is presented in [10]. 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:

$$H = \begin{cases} x_{k+1,1} = 0.7555x_{k,1} + 0.25x_{k,2} - 0.5v_k, \\ x_{k+1,2} = -0.1991x_{k,1}, \\ v_k = \begin{cases} \sqrt{u_k} & \text{if } u_k > 0 \\ u_k & \text{otherwise} \end{cases} \\ w_k = 0.6993x_{k,1} - 0.4427x_{k,2} \\ y_k = w_k + 5\sin(w_k), \end{cases} \tag{3.24}$$

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 w_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 wass 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 as follows.

2. *Multiharmonic [*Mh*]*: where a sinusoidal signal was used. Specifically, we used $u_k = 0.5\sin\left(\frac{k}{20+0.01k}\right) + 0.5$. Hereafter, we refer to this signal as *Mh*.

3. *Beta [*B*]*: Beta distribution was used. 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, refer to [55]. This signal is denoted *B*.

We generated the datasets by establishing the initial conditions for $x_{k,1}$ and $x_{k,2}$ and using (3.23) and (3.24) 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 consider 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 dataset by subtracting the respective mean and then dividing by the respective standard deviation. Additionally, we introduce 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 has $n_{\hat{x}}$ units, and the decoder's last layer has 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 is included in this network to estimate A and B. For loss function (3.12), 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 analyze both NSS and quasi-LPV systems and train autoencoder-based models in two ways: TBPTT with $F = 1$ and $F = 4$, totaling four different configurations. For the ensemble, we consider $N = 10$ autoencoder-based models composing the ensemble and apply 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 [10] 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 is 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.

3.5.2 Numerical results

We now present the comparison between the following methods: single autoencoder-based model, as in [10], an ensemble of N models with $w_i = 1/N$ for each model i, and selected ensemble according to ECSA in Algorithm 1.

Table 3.1 reports the comparative results on the identification of the two-tank system S, whereas Table 3.2 shows the results on the Hammerstein–Wiener system H. Both tables are obtained using as training test $\mathcal{D}_G$ and $\mathcal{T}_G$ as test set, as in [10]. For the validation set necessary for Algorithm 1, we used the $\mathcal{V}_h$ generated with the *Mh* signal. We choose 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 run 10 experiments for each system with this configuration, and the results are averaged over these experiments. In particular, the tables show the results, expressed in terms of NRMSA metric, obtained by the average individual model (top part), by the ensemble of $N = 10$

Table 3.1: 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

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 3.1, 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 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 3.2, 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 $N' = 8$

Table 3.2: 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

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 3.1 and Table 3.2 reflect the scenario where the training and test sets are generated from the same signal, we additionally test a more challenging identification problem on the system S, and consider the case where the training and test sets are obtained with different input distributions. In particular, Table 3.3 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 run 10 experiments, and we average over these experiments. In this case, the system analyzed is an NSS with $F = 1$ for the TBPTT method. We choose this configuration because this represents the worst-case scenario for the performance according to the results in Table 3.1.

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 [27], [28], 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 3.3. 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. Figure 3.3 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 (Figure 3.3.a), the predicted output follows the true output quite closely except at the peaks. For the multi-harmonic input (Figure 3.3.c), which matches the type of signal used during training, the predictions are highly accurate, whereas performance is lower for the beta input (Figure 3.3.e). These observations are further supported by Table 3.4, 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.

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

Table 3.3: Values of the NRMSA metric for different test sets with the system NSS 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

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 evaluate the NRMSA performance on the two test sets.

The mean and standard deviation of the NRMSA values for different sizes are shown in Figure 3.4. 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, which, as explained by [28], 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

Table 3.4: 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

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.

3.6 Conclusions

This chapter proposed an ensemble learning system identification framework, resorting to autoencoder-based models adapted for system identification and extending them to ensemble settings. Additionally, ECSA was introduced as a procedure capable of selecting a subset of the autoencoder-based models in order to maximize the system's NRMSA performance metrics. The ensemble approach demonstrated to be effective on two well-known benchmarks, which are the identification of a two-tank system and of the Hammerstein-Wiener system, generally outperforming individual models.

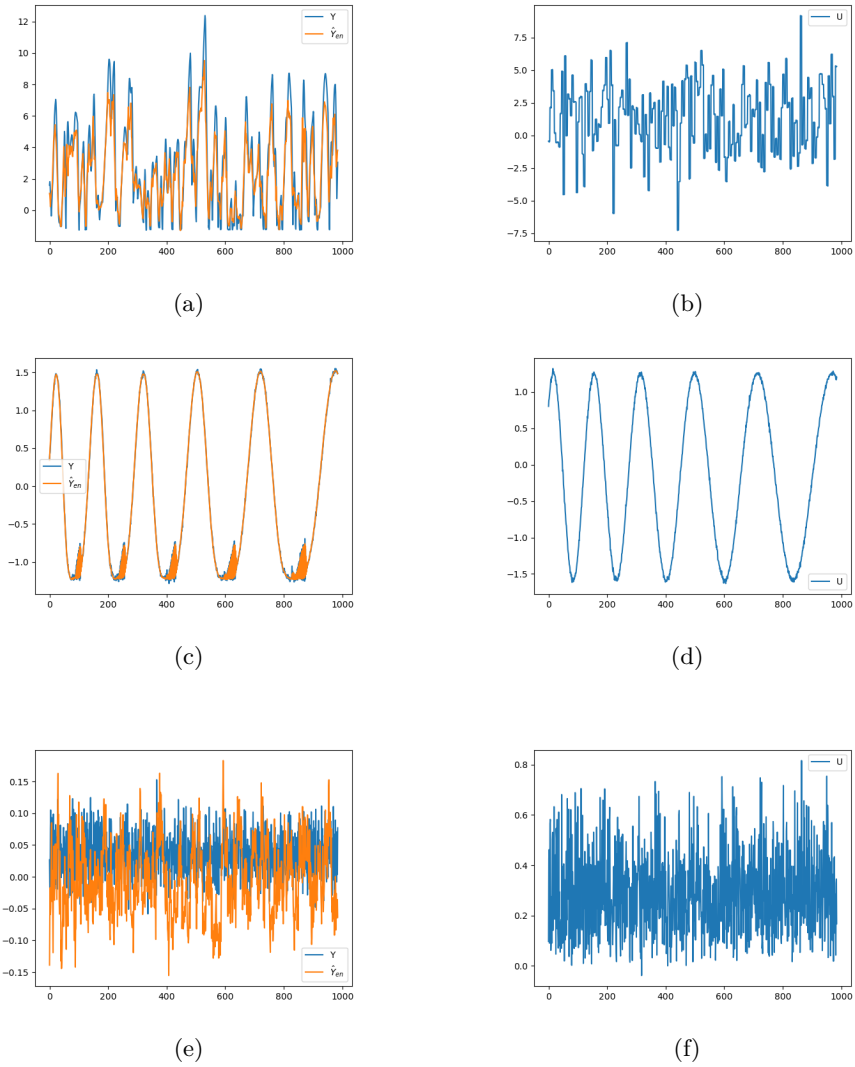


Figure 3.3: Open loop comparison between the real system output Y, the ensemble output $\hat{Y}_{en}$ (figures on the left), and the real system input U (figures on the right) for three different distributions: Gaussian (Figs. (a) and (b)), Multi-harmonic (Figs. (c) and (d)), and Beta (Figs. (e) and (f)).

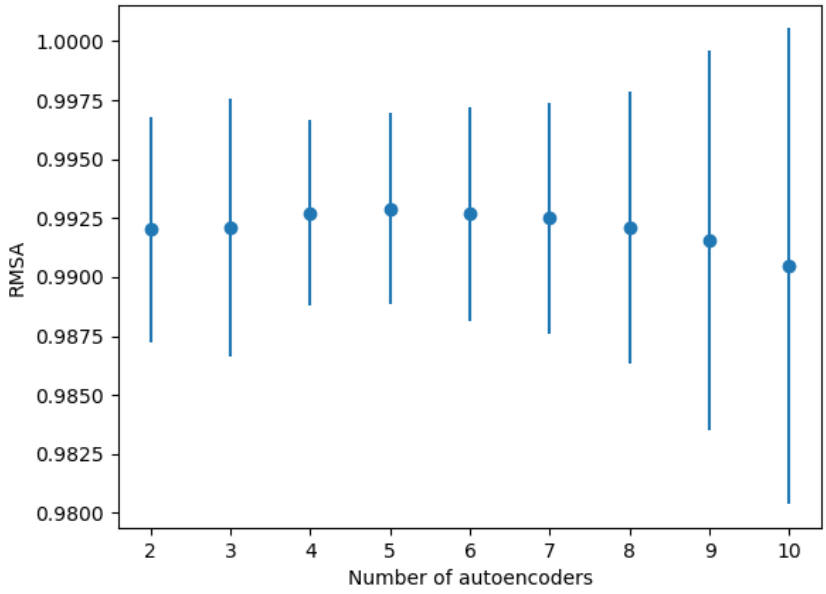


Figure 3.4: Comparison of ensembles with a different number of components based on their NRMSA value.

Chapter 4

Thermal Modeling of Thin-Layer Bread via Neural Networks

This chapter is based on the paper:
D. Deplano, N. Arridu, C. Seatzu and M. Franceschelli,
"Thermal Modeling of Thin-Layer Bread via Neural Networks: Experimental Validation in a Sardinian Bakery",
in *Control Engineering Practice*, 2026.

4.1 Introduction

Following the ensemble-learning framework introduced in the previous chapter, this chapter presents neural network–based methods for modeling the cooling behavior of flatbreads, validated through experimental data collected in a Sardinian bakery. Given the Industry 5.0–oriented operational setting of the bakery, the proposed framework is conceived as a human-centric decision-support tool that assists operators in adapting cooling conditions to varying scenarios. Building upon previous research on the drying–cooling dynamics of Carasau bread [29], a traditional Sardinian flatbread, which identified the Verma model as one of the most suitable thin-layer models for flatbreads, we develop a data-driven framework to tune the Verma model coefficients. The goal is to learn how these coefficients vary with boundary conditions – such as initial temperature and ambient conditions – thereby enhancing model accuracy and generalization in real industrial settings. This, in conjunction with

the real-time recalculation of the parameters of the Verma model at the level of individual sheets, allows the information on the cooling process to be used within the regulation of the industrial process.

Flatbreads serve as a dietary cornerstone for nearly one-third of the global population, with the market valued at $38.8 billion in 2018 and expected to expand to $62.8 billion by 2026 [56]. Typically made from a simple blend of flour, water, and salt, these breads exhibit considerable diversity in shape, size, and production methods across cultures [57]. For example, the Middle Eastern Lavash, Indian Chapati, and Central American Tortillas each reflect unique culinary traditions [58], [59], [60]. Research has historically concentrated on conventional three-dimensional breads [61], leaving a gap in the theoretical understanding and practical optimization of the drying and cooling processes specific to flatbreads. Notably, Salari et al. [58] is one of the few studies addressing the baking-drying kinetics of flatbreads. This shortfall in research is further compounded by the fact that many flatbreads are produced in small-scale operations with limited automation, which poses challenges in meeting the growing global demand. In these settings, even where some level of mechanization exists, much of the work still relies on outdated equipment or manual labor, which negatively impacts production efficiency and increases labor costs [62].

To tackle these issues, this chapter studies data-driven models based on neural networks designed to improve automation, productivity, and reliability in less automated flatbread bakeries, with a focus on Carasau bread [29], [63], [64], [65], [66], [67], [68]. The production process of Carasau begins with mixing flour, water, salt, and yeast to form the dough, which is then rolled into thin sheets and baked. During baking, the rapid temperature change causes the bread to puff up due to the expansion of air and moisture. After baking, the sheets undergo a crucial cooling phase, where precise control of temper-

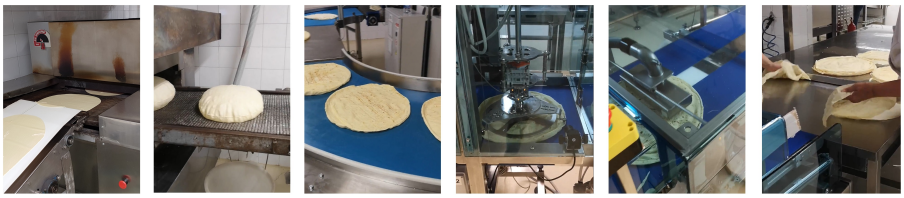


Figure 4.1: Pictures of Carasau production process showing the steps of baking in the oven, cooling/drying over the conveyor belt, and automated separation.

ature is necessary to prepare the product for the subsequent separation step. In the plant examined, these sheets are automatically fed into a machine that splits them into two pieces. The proper operation of this machine depends on achieving exact thermal conditions, as deviations can lead to waste through incorrect separation. Therefore, a deep understanding of heat transfer processes is essential for enhancing both the efficiency of automation and the quality of the final product [69]. An accurate prediction of the temperature during the cooling process makes it possible to adjust the oven temperature and the speed of the conveyor belt, increasing the success of the subsequent cutting phase. In this way, waste is reduced and energy efficiency is improved, which, together with the support provided to human operator supervision, is in line with the guidelines of Industry 5.0 [70], [71].

Related works Recent advances in Artificial Intelligence (AI) and Machine Learning have enabled diverse neural network models to optimize food production, addressing challenges in process control, quality assurance, and supply chain management. Deep learning methods have automated tasks such as ingredient selection and dynamic process simulation (e.g., cooking and dough kneading) [72], [73], [74]. Early approaches using feed-forward and recurrent networks captured temporal and nonlinear dynamics, but they required extensive, high-quality data and were sensitive to operational variations [73], [75], [76]. To overcome these limitations, researchers have developed models that incorporate physical principles—such as physics-informed neural networks—to improve computational efficiency in inverse problems like parameter estimation and boundary condition determination [19], [77]. Although these advanced methods perform well in controlled settings, they still struggle with robustness in noisy, real-world environments [78]. At the same time, AI-driven temperature monitoring systems have made strides in food transport and quality control. By integrating multi-source data, including thermal imaging and ambient conditions, these models accurately predict temperature distributions to ensure food safety and freshness [79], [80], [81], [82], [83]. Hybrid approaches that combine deep learning with traditional methods, such as convolutional neural networks paired with support vector machines, further enhance prediction accuracy while maintaining computational efficiency [84].

Position of This Work In the context of Industry 5.0, the chapter provides an experimental continuation of the work previously presented in [29], which identified the Verma model as one of the most suitable thin-layer models for

the heat decay of flatbreads. Here, we introduce two data-driven methods for identifying the parameters of the Verma model using neural networks. The first, physics-informed learning, integrates the Verma law directly within the training objective, while the second uses fully supervised regression based on pre-fitted model parameters. Once the training is completed, the neural network enables real-time computation of the Verma model parameters, which may vary with changes in the initial and ambient temperatures, and, in turn, allows real-time prediction of the temperature decay of the flatbread sheets. The chapter aims to bridge classical thermodynamic modeling with machine learning, enabling a practical step toward predictive automation of Carasau bread production.

The remainder of this chapter describes the data acquisition setup, the formulation of two learning strategies, and the experimental results that show that both approaches perform effectively and with similar performance, thus demonstrating the advantages of integrating physical modeling with data-driven learning.

4.2 Data-driven models for the cooling dynamics of Carasau bread

The production process of Carasau bread can be divided into several steps, which are pictured in Figure 4.1: mixing the ingredients to obtain a dough; rolling out the dough into round sheets of about 1 mm thickness; baking the sheets in an oven and then leaving them to cool down and dry; separating the cooled and dried sheets into two separate sheets; re-baking the separated sheets.

This work focuses on the automation of the *cooling* process and the subsequent separation process. In our plant, the sheets are left to cool down over a conveyor belt that brings the sheets directly into the separation machine that cuts and splits them apart. Note that, while traditionally the separation process is done manually, in our plant, it has been designed a custom separation machine that automates this process. To ensure it functions correctly, the separation machine relies on specific sheet temperature levels.

Let's consider the Verma model, previously discussed in Chapter 2, that is a physics-based model describing the cooling dynamics. We have:

$$\tau_r(t) = ae^{-\lambda_0 t} + (1-a)e^{-\lambda_1 t}, \qquad t \in \mathbb{R}, \tag{4.1}$$

where $\tau_r(t)$ is the dimensionless temperature, and a, λ_0, and λ_1 are the model parameters. This model consists of two exponential modes representing two

competing cooling rates, which indicate the presence of multiple factors causing different effects.

Given a set of parameters $\{a, \lambda_0, \lambda_1\}$, it is possible to predict the average temperature decay of the bread sheet by knowing the initial temperature $\tau(0) \geq 0$ and the environmental temperature $\tau_e \geq 0$ (assumed constant since the cooling process is much faster than ambient temperature changes) by means of the following:

$$\hat{\tau}(t) = \tau_r(t)(\tau(0) - \tau_e) + \tau_e, \tag{4.2}$$

The main drawback of using the Verma model to predict the cooling behavior of flatbread lies in the difficulty of properly tuning its parameters, which

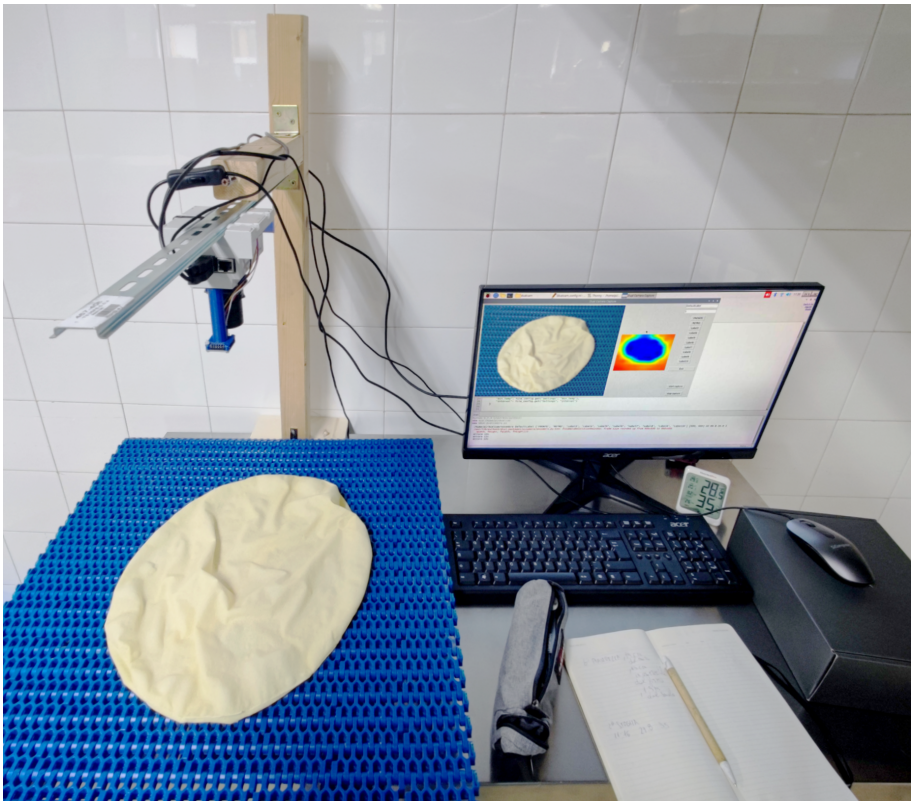


Figure 4.2: A photo of the set up used to take thermal photos of the bread.

vary with the initial temperature $\tau(0)$ and the ambient (environmental) temperature τ_e. In this work, we propose two alternative methods for learning the Verma model parameters using neural networks.

We consider feedforward neural networks with $L \in \mathbb{N}$ layers of $n_\ell \in \mathbb{N}$ neurons for $\ell \in \{1, \cdots, L\}$ such that

$$\begin{bmatrix} \hat{a} \\ \hat{\lambda}_0 \\ \hat{\lambda}_1 \end{bmatrix} = f_\theta \left(\begin{bmatrix} \tau(0) \\ \tau_e \end{bmatrix} \right) \tag{4.3}$$

where $\hat{a}$, $\hat{\lambda}_0$, $\hat{\lambda}_1$ denote the estimated parameters, and θ is the set of the neural network parameters to be learned during training. Recalling the definition of neural network given in Chapter 2, the state update rule $f_\theta := f_L \circ \cdots \circ f_1$ is defined as the composition of affine transformations and ReLU activation functions, which we conveniently express using the $\mathrm{LReLU}_\alpha(x) = \max(\alpha x, x)$ operator with $\alpha_\ell \in 0, 1$ as

$$f_\ell(x_{\ell-1}) = x_\ell = \mathrm{LReLU}_{\alpha_\ell}(W_\ell x_{\ell-1} + b_\ell). \tag{4.4}$$

Here, $W_\ell \in \mathbb{R}^{n_\ell \times n_{\ell-1}}$ denotes the weight matrix, $b_\ell \in \mathbb{R}^{n_\ell}$ the bias vector, and the input layer is given by $x_0 = [\tau(0), \tau_e]^\top$ with $n_0 = 2$ and $n_L = 3$. The activation parameters are set as $\alpha_1 = \cdots = \alpha_{L-1} = 0$ and $\alpha_L = 1$. We have that W_ℓ and b_ℓ are both contained in the set θ.

We now present and compare two different methodologies to learn the parameters $\hat{a}$, $\hat{\lambda}_0$, $\hat{\lambda}_1$ and then explain how the data have been collected and how the datasets for these two methodologies have been constructed. In so doing, we discretize the time by $t = \Delta k$ where $\Delta \in \mathbb{R}_{>0}$ is the sampling time and $k \in \mathbb{N}$ is the discrete-time variable.

4.2.1 Physics-informed learning

This strategy aims at estimating the Verma model parameters by training a feedforward neural network which minimizes the distance between the real curve $\tau_i(k)$ – sampled at discrete instants of time $k \in \mathbb{N}$ with steps $\Delta \in \mathbb{R}_{>0}$ – of the i-th sample and the curve $\hat{\tau}_{i,k}$ obtained by evaluating the Verma model with the learned parameters, namely

$$\hat{\tau}_{i,k} = (\hat{a}e^{-(\hat{\lambda}_0 \Delta)k} + (1 - \hat{a})e^{-(\hat{\lambda}_1 \Delta)k})(\tau_0 - \tau_e) + \tau_{i,e},$$

with $\tau_{i,e}$ denoting the initial ambient temperature for the i-th sample. The loss to be minimized during the training is thus

$$\mathcal{L}(\theta) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{n_i}\sum_{k=0}^{n_i}(\hat{\tau}_{i,k} - \tau_{i,k})^2,$$

where $n_i \in \mathbb{N}$ is the total number of measurements collected for the i-th sample and $N \in \mathbb{N}$ is the total number of samples.

4.2.2 Supervised learning

This strategy aims at estimating the Verma model parameters by training a feedforward neural networks which minimizes the distance between the param-

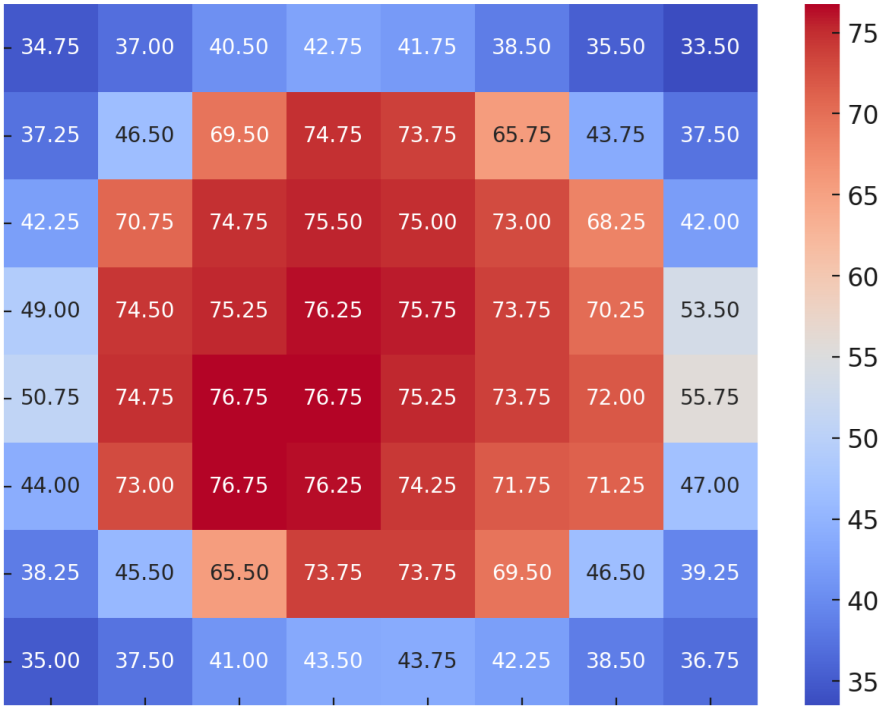


Figure 4.3: Surface temperature heatmap (°C) of a Carasau bread sheet at the start of the cooling phase.

eters a_i, $\lambda_{i,0}$, $\lambda_{i,1}$ obtained by curve fitting of the i-th sample data and the learned parameters $\hat{a}_i$, $\hat{\lambda}_{i,0}$, $\hat{\lambda}_{i,1}$ for that sample. The loss to be minimized during the training is given by

$$\mathcal{L}(\theta) = \frac{1}{3N} \sum_{i=1}^{N} \left\| \begin{bmatrix} \hat{a}_i - a_i \\ \hat{\lambda}_{i,0} - \lambda_{i,0} \\ \hat{\lambda}_{i,1} - \lambda_{i,1} \end{bmatrix} \right\|^2,$$

where $N \in \mathbb{N}$ is the total number of samples.

4.3 Dataset construction

For the dataset construction, we carried out a series of measurements directly on a Carasau production plant in Vecchio Forno bakery in Fonni, Sardinia, Italy, using thermal imaging on $N \in \mathbb{N}$ semi-finished flatbread sheets just after the baking. The configuration of the thermal camera is visible in Figure 4.4. The thermal camera is an *Adafruit AMG8833 8x8 Thermal Camera*, which has an uncertainty in the measure of 2.5°C. The camera is coupled with a *Raspberry Pi* device. An example of a Carasau bread sheet thermal measurement is given in Figure 4.3. For each sheet of bread, several thermal images have been taken to monitor the decay of temperature.

Considering that a thermal image is composed by 8x8 pixels, i.e., a 8×8 matrix. Let $T_{i,k}$ be the generic matrix associated to the i-th sample, where $k \in \mathbb{N}$ refers to the time instant. For each bread sheet we have collected $n_i \in \mathbb{N}$ thermal photos, with a sampling time of about 1 second, thus forming the succession $\{T_{i,0}, T_{i,1}, \ldots, T_{i,n_i}\}$. For each sequence, we construct two different scalar curves:

- *Average temperature*: We identified the 3×3 submatrix of $T_{i,k}$ with the highest average temperature, and considered the average temperature of these pixels for all $k \in \mathbb{N}$ to construct the temperature decay curve $\tau_{i,k}^{\text{avg}}$.
- *Maximum temperature*: We considered the pixel of $T_{i,k}$ with the highest temperature for all $k \in \mathbb{N}$ to construct the temperature decaying curve $\tau_{i,k}^{\max}$.

Both choices allow to neglect the background and the borders that are relatively colder, which may distort the prevision of the temperature decay.

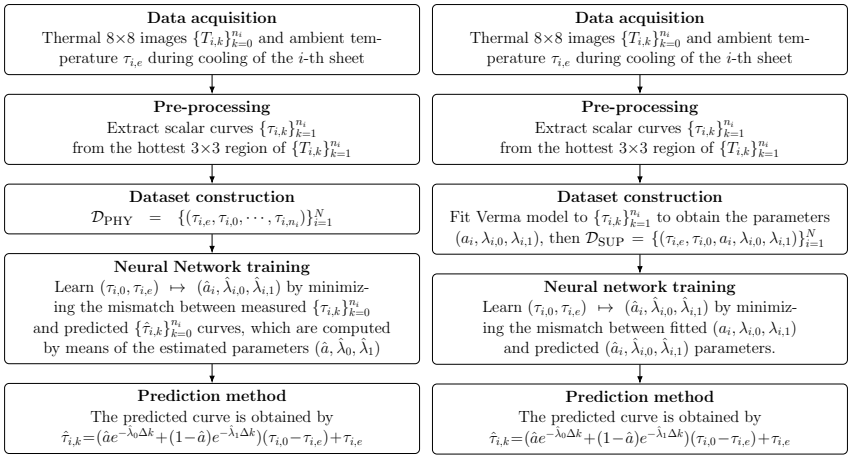
For the physics-informed learning methodology, the datasets are constructed by associating the environmental temperatures to the real curves, namely,

$$\mathcal{D}_{\text{PHY}}^{\text{avg}} = \left\{\left(\tau_{i,e}, \tau_{i,0}^{\text{avg}}, \cdots, \tau_{i,n_i}^{\text{avg}}\right)\right\}_{i=1}^{N},$$
$$\mathcal{D}_{\text{PHY}}^{\text{max}} = \left\{\left(\tau_{i,e}, \tau_{i,0}^{\text{max}}, \cdots, \tau_{i,n_i}^{\text{max}}\right)\right\}_{i=1}^{N}.$$

For the supervised learning methodology, the datasets are constructed by associating the environmental temperatures with the initial temperature and the parameters fitted to the Verma model for the curves in $\mathcal{D}_{\text{PHY}}^{\text{avg}}$ and $\mathcal{D}_{\text{PHY}}^{\text{max}}$, repsectively, namely,

$$\mathcal{D}_{\text{SUP}}^{\text{avg}} = \left\{\left(\tau_{i,e}, \tau_{i,0}^{\text{avg}}, a_i, \lambda_{i,0}, \lambda_{i,1}\right)\right\}_{i=1}^{N},$$
$$\mathcal{D}_{\text{SUP}}^{\text{max}} = \left\{\left(\tau_{i,e}, \tau_{i,0}^{\text{max}}, a_i, \lambda_{i,0}, \lambda_{i,1}\right)\right\}_{i=1}^{N}.$$

In Figure 4.5 we show a graphical representation of the necessary steps to construct the dataset and perform the training process with both methods.



(a) Physics-informed. (b) Supervised.

Figure 4.5: Graphical representation of the Verma-model parameter learning methodologies: (a) physics-informed vs (b) supervised.

4.4 Experimental results and discussion

This section compares the two identification strategies described in the previous section for the Verma parameters – namely, a *physics-informed* approach

and a *supervised* approach – under identical architectures, data splits, and evaluation protocols, interpreting their accuracy, variability, and implications for the Carasau production line.

Each dataset contains N temperature-decay sequences acquired on semi-finished sheets immediately after baking using an 8×8 thermal camera (AMG8833, ±2.5°C), with sampling interval ~ 1.25 s and variable sequence lengths. As mentioned above, we consider two scalar signals $\tau_{i,k}^{\text{avg}}$, $\tau_{i,k}^{\max}$ computed as the time evolution of the average/maximum over the 3×3 hottest submatrix at $k = 0$, respectively, emphasizing the core thermal region while mitigating boundary and background artifacts. We adopt an 80/20 split into training and validation sets; the supervised sets are derived from the same bread samples, thus ensuring a like-for-like comparison.

Both approaches use the same feedforward neural network with four fully connected layers with $50 \times 50 \times 16 \times 3$ neurons, ReLU activations in all internal layers, and the following activation functions in the output layer: a softplus function for the neurons associated with λ_0, λ_1 ensuring nonnegativity of the parameters, and a logistic function for the neuron associated with the convex combination parameter a ensuring it belongs to the interval $[0, 1]$. The neural networks are implemented in PyTorch [85], and trained for 400 epochs (~ 0.03 s per epoch). For the sake of clarity, we reiterate that the physics-informed network minimizes trajectory mismatch between the real curve temperatures $\tau_{i,k}^{\text{avg}}$, $\tau_{i,k}^{\max}$ and the Verma reconstructions generated by predicted parameters $(\hat{a}, \hat{\lambda}_0, \hat{\lambda}_1)$, whereas the supervised network minimizes parameter error relative to per-curve Verma fits $(a_i, \lambda_{i,0}, \lambda_{i,1})$, contrasting end-to-end trajectory supervision with label-supervised regression.

For the sake of clarity, let us denote with $f_{\text{PHY}}^{\text{avg}}$, $f_{\text{PHY}}^{\max}$ the networks trained with the physics-informed approach and $f_{\text{SUP}}^{\text{avg}}$, $f_{\text{SUP}}^{\max}$ the networks trained with the supervised approach, and pair them into a number of configurations equal to the total number of experiments, that is 100, to quantify robustness across

Table 4.1: Comparison over 100 configurations for maximum curves temperature: best, worst, average, and standard deviation of NRMSE on the validation set.

Model	Best	Worst	Avg	Var
$f_{\text{PHY}}^{\max}$	0.0060	0.0329	0.0163	0.0055
$f_{\text{SUP}}^{\max}$	0.0108	0.0633	0.0238	0.0089

random initializations and stochastic optimization.

Performance is measured via the normalized root mean square error (NRMSE) on the shared validation set $\mathcal{V}$:

$$\text{NRMSE} = \frac{1}{NT_{\max}} \sum_{i=1}^{N} \sqrt{\frac{1}{n_i} \sum_{k=0}^{n_i} \left(\hat{\tau}_{i,k} - \tau_{i,k}\right)^2}, \tag{4.5}$$

where $T_{\max}$ is the maximum temperature across all sequences, normalizing for varying trajectory ranges and lengths induced by τ_0 and τ_e.

Figure 4.6 and Figure 4.7 on the next page illustrate representative predictions for both maximum and average temperature curves on randomly selected samples. The visual agreement between both approaches and the measured data confirms the quantitative trends observed in Table 4.1 and Table 4.2, namely that physics-informed modeling provides slightly better predictions of peak temperature decay, while both methods perform similarly for averaged curves.

Table 4.1 summarizes the results for the maximum temperature curves across 100 configurations. In this regime, the physics-informed model clearly outperforms the supervised approach, achieving lower best (0.0060 vs. 0.0108), average (0.0163 vs. 0.0238), and variance (0.0055 vs. 0.0089) NRMSE values. These improvements indicate that embedding physical constraints effectively regularizes the model, guiding it toward more stable and physically consistent solutions, which is particularly relevant in industrial scenarios where measurement noise and operating conditions vary across production cycles. Together these results indicate that incorporating the physical prior yields both higher ceiling performance and greater run-to-run stability when modeling peak temperatures.

For the average temperature curves (Table 4.2), the two methods are largely comparable. The supervised model attains the best single run (0.0107 vs. 0.0121) and a marginally lower mean NRMSE (0.0290 vs. 0.0306), while vari-

Table 4.2: Comparison over 100 configurations for average curves temperature: best, worst, average, and standard deviation of NRMSE on the validation set.

Model	Best	Worst	Avg	Var
$f_{\text{PHY}}^{\text{avg}}$	0.0121	0.0744	0.0290	0.0120
$f_{\text{SUP}}^{\text{avg}}$	0.0107	0.0603	0.0306	0.0115

ances are similar (0.0115 vs. 0.0120). This suggests that for smoother, mean behavior the data alone suffices to reach near-optimal fits, and the physical prior offers little extra advantage.

Overall, these findings highlight a context-dependent trade-off: physics-informed learning provides clear advantages for modeling peak or boundary-dominant behaviors, where physical consistency constrains the solution space effectively, while the supervised approach performs equally well—or slightly better—in smoother, average regimes dominated by data regularity. The small observed differences fall within expected stochastic variability, suggesting that both models achieve broadly comparable accuracy under the same architectural and training conditions.

4.5 Conclusions

This chapter extended previous investigations on thin-layer modeling of the cooling and drying process for Carasau bread by introducing a data-driven identification framework grounded in the Verma model, for the real time prediction of temperature decay. The proposed physics-informed neural technique successfully bridged the gap between theoretical model accuracy and practical variability observed in industrial measurements. Experimental results demonstrated that this hybrid approach produced more stable and precise predictions than classical curve fitting, with a consistent reduction in normalized error, bringing a reliable basis for real-time monitoring of the cooling process.

Compared to the earlier work that focused solely on analytical modeling, the present chapter extends the methodology with machine-learning-based estimation, providing data-driven generalization of the Verma model parameters under changing operating conditions.

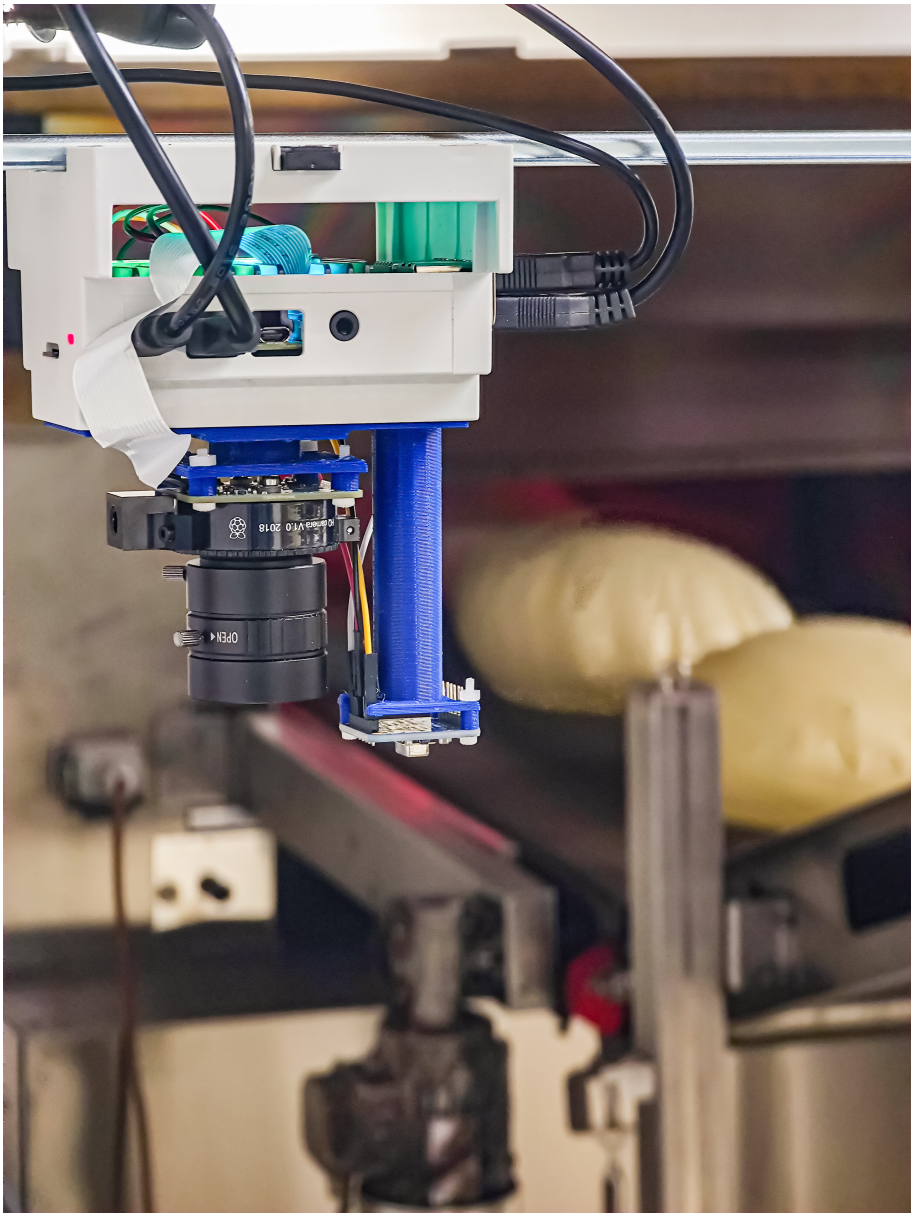


Figure 4.4: Custom device used to collect temperature profiles of Carasau bread. The Adafruit AMG8833 8×8 thermal camera is mounted on a blue support bracket.

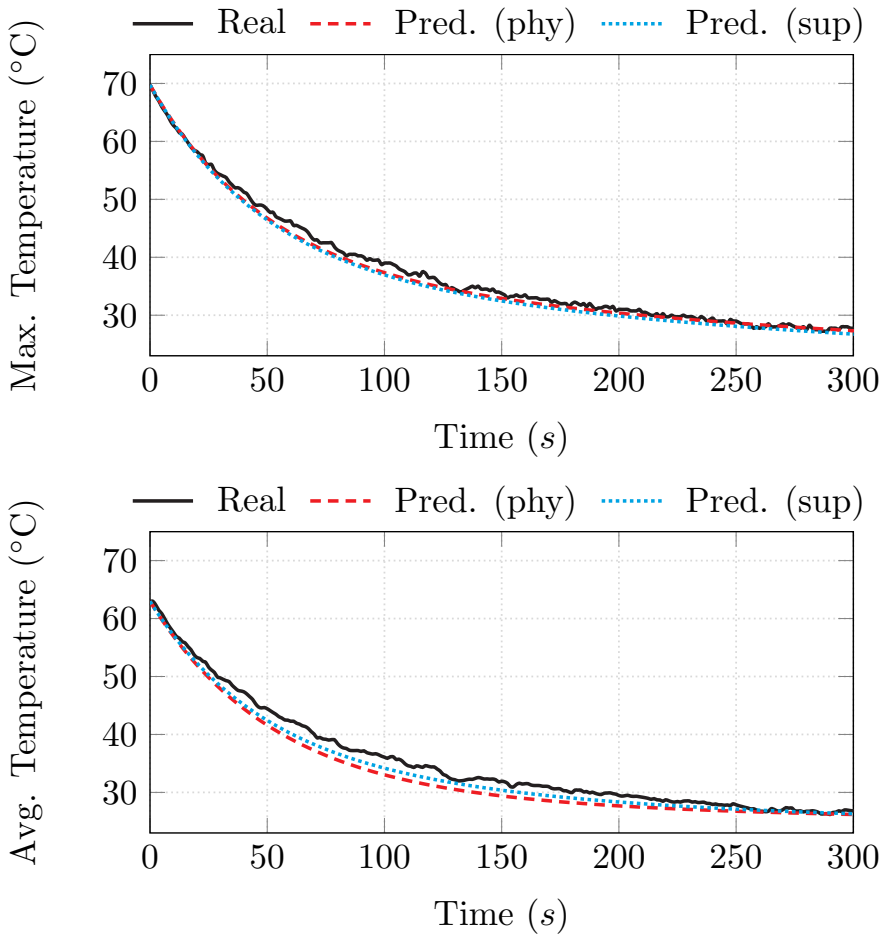


Figure 4.6: Average and maximum curve temperatures temperature for the bread sample with highest NRMSE.

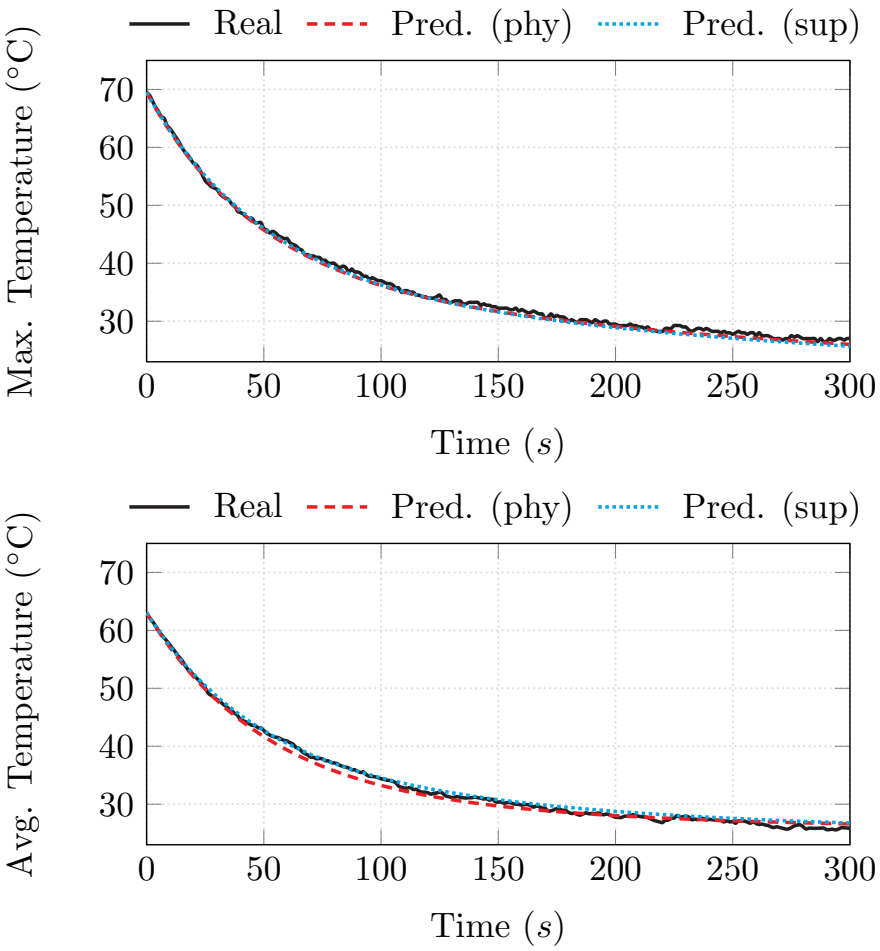


Figure 4.7: Average and maximum curve temperatures temperature for the bread sample with lowest NRMSE.

Chapter 5

Conclusions and Future Work

5.1 Summary of Contributions

This thesis is positioned within the research field of nonlinear dynamical system identification through the use of data-driven methods based on neural networks, ranging from ensemble learning techniques to the integration of physics-informed networks for the identification of thermal processes.

5.1.1 Ensemble Learning for System Identification

The first contribution was aimed at improving the effectiveness of system identification methods using neural networks. To achieve this, a neural network architecture based on autoencoders was developed, capable of jointly learning the state representation, the relationship between the state and the output, and the dynamics governing the system's evolution. To enhance the approach, multiple such models were combined using an ensemble learning method. Furthermore, the *Ensemble Component Selection Algorithm* (ECSA) was designed to distinguish models according to a score based on the performance–diversity criterion. The results obtained on two nonlinear systems, namely the Two-Tank and Hammerstein–Wiener systems, show that the ensemble learning strategy leads to improved performance and reliability.

5.1.2 Neural Modelling of Thermal Dynamics

The second contribution was focused on the identification of a thermal process consisting of the cooling of *Carasau* bread, a typical Sardinian flatbread, in the context of an industrial bakery. The data-driven approach involved the design of a data acquisition system to obtain sequences of thermal images for monitoring heat transfer. A neural network was trained with the collected data to understand the cooling dynamics through the estimation of the parameters of the Verma model, which describes this evolution as environmental conditions change. The experimental validation of this method demonstrated accurate predictions consistent with physical knowledge.

5.1.3 Overall Impact

This thesis has focused on the identification of nonlinear systems with the aim of contributing to automation through data-driven methods. The two research directions establish a hybrid framework that combines data exploitation with a physics-informed approach, ensuring the accuracy and reliability of the proposed methods. All of this aligns with the principles of *Industry 5.0*, promoting sustainable and human-centered automation.

This research was intended to contribute to the technological development of small and medium-sized enterprises, particularly in disadvantaged regions where automation is not yet widely adopted in production sectors. In conclusion, this work provides tools for advancing data-driven automation in industrial contexts.

5.2 Future Research Directions

The methodologies applied in this thesis offer the possibility of being extended in multiple directions. Below, we present the main and most promising ones capable of advancing the contributions made.

The most immediate direction that can be pursued is related to the application of the proposed methods to other industrial scenarios, developing data-driven approaches for other domains such as energy or hydraulic systems, where physical knowledge does not allow for a complete modeling of the systems under consideration.

One of the most important motivations for system identification lies in the perspective of controlling such systems. For this reason, the system representation derived from the autoencoder ensemble can be used in *Model Predictive*

Control (MPC) algorithms [86]. In this context, the ensemble is capable of providing an accurate estimate of future state and output values, enabling the controller to determine the necessary future control actions.

The ensemble learning framework can also be integrated with a physics-informed approach, combining the physical interpretability of individual models with the performance improvements achieved through the combination of multiple models. This could be particularly useful for the thermal process described in Chapter 4, obtaining from each autoencoder a physically interpretable representation of the sequence of thermal images of *Carasau* bread. Further research opportunities arise from the possibility of modifying the architecture of individual autoencoders, introducing *recurrent neural networks* [87] or *transformers* [88] to better capture temporal dependencies in dynamical systems. The structure of the autoencoders can also be extended with an explicit treatment of uncertainty, estimating confidence intervals for the predictions to quantify the risk involved in using the model for control or monitoring purposes.

Regarding neural network training methods for system identification, an interesting proposal is given by *mutual learning* [89]. According to this framework, the networks composing the ensemble are not trained independently but share information during training to achieve collaborative improvement, enhancing ensemble performance while maintaining diversity within it. Another important direction is *adaptive learning* [90], which allows continuous adaptation of the neural network—crucial in all contexts characterized by significant variability of environmental conditions and requiring real-time responsiveness to such changes.

Publications

In the following, the list of Author's publications with relevance to this thesis.

Journals

[J1] N. Arridu, M. Lippi, M. Franceschelli and A. Gasparri, "On Ensemble Learning Models for Autoencoder-Based Identification of Nonlinear Dynamical Systems", in *IEEE Access*, vol. 13, pp. 169071-169082, 2025.

[J2] D. Deplano, N. Arridu, C. Seatzu and M. Franceschelli, "Thermal Modeling of Thin-Layer Bread via Neural Networks: Experimental Validation in a Sardinian Bakery", in *Control Engineering Practice*, 2026.

Bibliography

[1] Encyclopaedia Britannica. "Automation."

[2] Eurofound, "Transforming production and employment in europe: Automation and digitalisation," European Foundation for the Improvement of Living and Working Conditions, Tech. Rep., 2018.

[3] G7 Italy Presidency, "Driving factors and challenges of ai adoption among micro, small and medium-sized enterprises (msmes)," Ministry of Enterprises and Made in Italy, Rome, Italy, Tech. Rep., Oct. 2024.

[4] L. Ljung, "Perspectives on system identification," *Annual Reviews in Control*, vol. 34, no. 1, pp. 1–12, 2010.

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

[6] Z.-H. Zhou, *Ensemble methods: foundations and algorithms*. Chapman and Hall/CRC, 2012.

[7] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, 2020.

[8] European Commission. "Industry 5.0 – towards a sustainable, human-centred and resilient european industry."

[9] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[10] D. Masti and A. Bemporad, "Learning nonlinear state–space models using autoencoders," *Automatica*, vol. 129, p. 109 666, 2021.

[11] S. Edition, "The authoritative dictionary of ieee standards terms," *IEEE Std*, pp. 100–2000, 2000.

[12] L. Ljung, *System Identification: Theory for the User*. Prentice-Hall, Inc., 1999.

[13] C.-T. Chen, *Linear system theory and design*. Saunders college publishing, 1984.

[14] K. Ogata, *Modern control engineering*. Prentice hall, 2010.

[15] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*. Springer, 2001.

[16] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*. Krieger Pub., 2006.

[17] A. Wills, T. B. Schön, L. Ljung, and B. Ninness, "Identification of hammerstein–wiener models," *Automatica*, vol. 49, no. 1, pp. 70–81, 2013.

[18] S. Billings, *Nonlinear System Identification: NARMAX Methods in the Time, Frequency, and Spatio-Temporal Domains*. John Wiley & Sons, 2013.

[19] J. Chen, J. Shi, A. He, and H. Fang, "Data-driven solutions and parameter estimations of a family of higher-order kdv equations based on physics informed neural networks," *Scientific Reports*, vol. 14, no. 1, pp. 1–27, 2024.

[20] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[21] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[23] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1550–1560, 2002.

[24] G. Puskorius and L. Feldkamp, "Truncated backpropagation through time and kalman filter training for neurocontrol," *IEEE International Conference on Neural Networks*, vol. 4, 2488–2493 vol.4, 1994.

[25] G. E. Hinton and R. Zemel, "Autoencoders, minimum description length and helmholtz free energy," *Advances in Neural Information Processing Systems*, vol. 6, 1993.

[26] T. G. Dietterich, "Ensemble methods in machine learning," in *Multiple Classifier Systems*, 2000, pp. 1–15.

[27] N. Ueda and R. Nakano, "Generalization error of ensemble estimators," in *International Conference on Neural Networks*, vol. 1, 1996, pp. 90–95.

[28] D. Wood, T. Mu, A. M. Webb, H. W. Reeve, M. Lujan, and G. Brown, "A unified theory of diversity in ensemble learning," *Journal of Machine Learning Research*, vol. 24, no. 359, pp. 1–49, 2023.

[29] D. Deplano, M. Franceschelli, and C. Seatzu, "Experimental comparison of models of the drying-cooling process of flatbreads for optimized automated production: The case study of carasau bread," in *2023 9th International Conference on Control, Decision and Information Technologies (CoDIT)*, IEEE, 2023, pp. 2014–2019.

[30] L. R. Verma, R. Bucklin, J. Endan, and F. Wratten, "Effects of drying air parameters on rice drying models," *Transactions of the ASAE*, vol. 28, no. 1, pp. 296–0301, 1985.

[31] S. E. Otto and C. W. Rowley, "Linearly recurrent autoencoder networks for learning dynamics," *SIAM Journal on Applied Dynamical Systems*, vol. 18, no. 1, pp. 558–593, 2019.

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

[33] 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," *Journal of Computational Physics*, vol. 378, pp. 686–707, 2019.

[34] M. Lazar, M.-S. Popescu, and M. Schoukens, "Nonlinear data-driven predictive control using deep subspace prediction networks," *IEEE Conference on Decision and Control*, pp. 3770–3775, 2023.

[35] K. S. Narendra and K. Parthasarathy, "Identification and control of dynamical systems using neural networks," *IEEE Transactions on Neural Networks*, vol. 1, no. 1, pp. 4–27, 1990.

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

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

[38] C. Zhang and Y. Ma, *Ensemble machine learning*. Springer New York, NY, 2012, vol. 144.

[39] L. I. Kuncheva and C. J. Whitaker, "Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy," *Machine learning*, vol. 51, pp. 181–207, 2003.

[40] Y. Wu, L. Liu, Z. Xie, K.-H. Chow, and W. Wei, "Boosting ensemble accuracy by revisiting ensemble diversity metrics," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 16 469–16 477.

[41] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 281–304, 2019.

[42] E. De la Rosa and W. Yu, "Randomized algorithms for nonlinear system identification with deep learning modification," *Information Sciences*, vol. 364, pp. 197–212, 2016.

[43] C. Andersson, A. H. Ribeiro, K. Tiels, N. Wahlström, and T. B. Schön, "Deep convolutional networks in system identification," in *IEEE Conference on Decision and Control*, 2019, pp. 3670–3676.

[44] R.-T. Wu and M. R. Jahanshahi, "Deep convolutional neural network for structural dynamic response estimation and system identification," *Journal of Engineering Mechanics*, vol. 145, no. 1, p. 04 018 125, 2019.

[45] J. L. Paniagua and J. A. López, "Nonlinear system identification using modified variational autoencoders," *Intelligent Systems with Applications*, vol. 22, p. 200 344, 2024.

[46] G. I. Beintema, M. Schoukens, and R. Tóth, "Deep subspace encoders for nonlinear system identification," *Automatica*, vol. 156, p. 111 210, 2023.

[47] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 481–486, 2021.

[48] 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," *Anais do 14º Simpósio Brasileiro de Automação Inteligente*, 2019.

[49] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE journal*, vol. 37, no. 2, pp. 233–243, 1991.

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

[51] 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 Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 1110–1123, 2020.

[52] 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.

[53] K. Pearson, "Note on regression and inheritance in the case of two parents," *Proceedings of the Royal Society of London*, vol. 58, pp. 240–242, 1895.

[54] M. Schoukens and J. P. Noël, "Three benchmarks addressing open challenges in nonlinear system identification," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 446–451, 2017.

[55] G. Casella and R. Berger, *Statistical inference*. Chapman and Hall/CRC, 2024.

[56] F. Boukid, "Flatbread-a canvas for innovation: A review," *Applied Food Research*, p. 100 071, 2022.

[57] S. Banooni, S. Hosseinalipour, A. Mujumdar, P. Taherkhani, and M. Bahiraei, "Baking of flat bread in an impingement oven: Modeling and optimization," *Drying Technology*, vol. 27, no. 1, pp. 103–112, 2009.

[58] A. Salari, M. M. Tehrani, and S. M. Razavi, "Baking-drying kinetics of crisp bread: The influence of bran content and baking temperature," *Iranian Food Science and Technology Research Journal*, vol. 11, no. 3, p. 225, 2015.

[59] K. Parimala and M. Sudha, "Wheat-based traditional flat breads of india," *Critical reviews in food science and nutrition*, vol. 55, no. 1, pp. 67–81, 2015.

[60] S. Xu and W. L. Kerr, "Modeling moisture loss during vacuum belt drying of low-fat tortilla chips," *Drying Technology*, vol. 30, no. 13, pp. 1422–1431, 2012.

[61] M. Hasatani, N. Arai, H. Harui, Y. Itaya, N. Fushida, and C. Hori, "Effect of drying on heat transfer of bread during baking in oven," *Drying Technology*, vol. 10, no. 3, pp. 623–639, 1992.

[62] F. Paschino, F. Gabella, F. Giubellino, and F. Clemente, "The level of automation of "carasau" bread production plants," *Journal of Agricultural Engineering*, vol. 38, no. 2, pp. 61–64, 2007.

[63] G. Cadalanu, "Il pane carasau. preparazione e cottura," *La Ricerca Folklorica*, pp. 81–88, 1984.

[64] A. Raffo, F. Paoletti, F. Sinesio, and G. Quaglia, "Organoleptic characters and keeping quality of typical hard wheat breads. altamura and carasau bread," *Rivista di Scienza dell'Alimentazione*, 2001.

[65] F. Paschino and F. Gambella, "Comparison between traditional and industrial plants used for production of carasau bread and evaluation of final products," *Bollettino della Comunitá Scientifica in Australasia*, 2008.

[66] G. Cavone, M. Dotoli, N. Epicoco, M. Franceschelli, and C. Seatzu, "Hybrid Petri nets to re-design low-automated production processes: The case study of a sardinian bakery," *IFAC-PapersOnLine*, vol. 51, no. 7, pp. 265–270, 2018.

[67] M. Baire et al., "A wireless sensors network for monitoring the carasau bread manufacturing process," *Electronics*, vol. 8, no. 12, 2019.

[68] M. Abbondio et al., "Fecal metaproteomic analysis reveals unique changes of the gut microbiome functions after consumption of sourdough carasau bread," *Frontiers in microbiology*, vol. 10, p. 1733, 2019.

[69] I. Ruiz-López and M. García-Alvarado, "Analytical solution for food-drying kinetics considering shrinkage and variable diffusivity," *Journal of food engineering*, vol. 79, no. 1, pp. 208–216, 2007.

[70] A. Hassoun et al., "From food industry 4.0 to food industry 5.0: Identifying technological enablers and potential future applications in the food sector," *Comprehensive reviews in food science and food safety*, vol. 23, no. 6, e370040, 2024.

[71] T. Y. Melesse and P. F. Orrù, "The digital revolution in the bakery sector: Innovations, challenges, and opportunities from industry 4.0," *Foods*, vol. 14, no. 3, p. 526, 2025.

[72] J. Huang, M. Zhang, A. S. Mujumdar, and C. Li, "Ai-based processing of future prepared foods: Progress and prospects," *Food Research International*, p. 115 675, 2025.

[73] B. Lamrini, G. Della Valle, I. C. Trelea, N. Perrot, and G. Trystram, "A new method for dynamic modelling of bread dough kneading based on artificial neural network," *Food Control*, vol. 26, no. 2, pp. 512–524, 2012.

[74] J. Fañanás-Anaya, G. López-Nicolás, C. Sagüés, and S. Llorente, "Food cooking process modeling with neural networks," *IEEE Access*, 2024.

[75] A. Oulmelk, M. Srati, L. Afraites, and A. Hadri, "An artificial neural network approach to identify the parameter in a nonlinear subdiffusion model," *Communications in Nonlinear Science and Numerical Simulation*, vol. 125, p. 107 413, 2023.

[76] Y. Liu, T. Kawaguchi, S. Xu, and S. Hashimoto, "Recurrent neural network-based temperature control system weight pruning based on nonlinear reconstruction error," *Processes*, vol. 10, no. 1, p. 44, 2021.

[77] W. Qiu, H. Chen, and H. Zhou, "Estimating the boundary conditions for 3d transient heat conduction by bidirectional long short-term memory network and attention mechanism," *International Journal of Heat and Mass Transfer*, vol. 233, p. 126 042, 2024.

[78] F. Mostajeran and R. Mokhtari, "Deepbhcp: Deep neural network algorithm for solving backward heat conduction problems," *Computer Physics Communications*, vol. 272, p. 108 236, 2022.

[79] Y. Zou, J. Wu, X. Wang, K. Morales, G. Liu, and A. Manzardo, "An improved artificial neural network using multi-source data to estimate food temperature during multi-temperature delivery," *Journal of Food Engineering*, vol. 351, p. 111 518, 2023.

[80] R. Badia-Melis, J. Qian, B. Fan, P. Hoyos-Echevarria, L. Ruiz-Garcia, and X. Yang, "Artificial neural networks and thermal image for temperature prediction in apples," *Food and Bioprocess Technology*, vol. 9, pp. 1089–1099, 2016.

[81] M. Canatan, R. Muñoz-Carpena, and Z. Boz, "Continuous surface temperature monitoring of refrigerated fresh produce through visible and thermal infrared sensor fusion," *Postharvest Biology and Technology*, vol. 222, p. 113 354, 2025.

[82] N. Oz, N. Sochen, D. Mendlovic, and I. Klapp, "Estimating temperatures with low-cost infrared cameras using physically-constrained deep neural networks," *Optics Express*, vol. 32, no. 17, pp. 30 565–30 582, 2024.

[83] S. Mercier and I. Uysal, "Neural network models for predicting perishable food temperatures along the supply chain," *Biosystems engineering*, vol. 171, pp. 91–100, 2018.

[84] W. da Silva Cotrim, L. B. Felix, V. P. R. Minim, R. C. Campos, and L. A. Minim, "Development of a hybrid system based on convolutional neural networks and support vector machines for recognition and tracking color changes in food during thermal processing," *Chemical Engineering Science*, vol. 240, p. 116 679, 2021.

[85] A. Paszke, "Pytorch: An imperative style, high-performance deep learning library," *arXiv preprint arXiv:1912.01703*, 2019.

[86] S. J. Qin and T. A. Badgwell, "A survey of industrial model predictive control technology," *Control engineering practice*, vol. 11, no. 7, pp. 733–764, 2003.

[87] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[88] A. Vaswani et al., "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[89] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4320–4328.

[90] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural networks*, vol. 113, pp. 54–71, 2019.