IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 31 October 2024, accepted 16 November 2024, date of publication 22 November 2024,
date of current version 3 December 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3504724

RESEARCH ARTICLE

Food Cooking Process Modeling With Neural Networks

JAVIER FAÑANÁS-ANAYA[1], GONZALO LÓPEZ-NICOLÁS[1], (Senior Member, IEEE), CARLOS SAGÜÉS[1], (Senior Member, IEEE), AND SERGIO LLORENTE[2]

[1]Instituto de Investigación en Ingeniería de Aragón (I3A), Universidad de Zaragoza, 50018 Zaragoza, Spain
[2]Research and Development Department, Induction Technology, Product Division Cookers, BSH Home Appliances Group, 50016 Zaragoza, Spain

Corresponding author: Javier Fañanás-Anaya (javierfa@unizar.es)

This work was supported in part by the Government of Aragón, Group T45_23R, in part by MCIN/AEI/10.13039/501100011033 under Project CPP2021-008938, Project PID2021-124137OB-I00, and Project TED2021-130224B-I00, and in part by European Union-NextGenerationEU/PRTR.

ABSTRACT Food cooking process are complex dynamical systems to model. In the state of the art we find that a good solution consists of physics-based finite element models (FEM). FEM models, although being very accurate, have a high computational cost making them unfeasible for real-time applications. To solve this problem, we consider neural networks (NN) trained from FEM simulations. Specifically, we propose a Nonlinear AutoRegressive with eXogenous inputs Neural Network (NARX-NN). The main novelty is that we define a novel training algorithm adapted to the modeling of real-time dynamical systems, allowing a NARX-NN with a simple structure to obtain a negligible error compared to the results of the original FEM model. The NARX-NN trained with the proposed training algorithm obtains an R-squared of more than 0.99 in the test simulations, while the same NARX-NN trained with the standard training algorithm obtains an R-squared of 0.78 in the same tests. The proposed NARX-NN achieves a speedup of 8 orders of magnitude compared to the original FEM model. Moreover, the developed NN is able to predict the complete cooking of the food in a few milliseconds without the need of external sensors. Alternatively, our approach can also be used in real time with information captured with sensors. The presented methodology is highly scalable and could be adapted to different types of food and cooking processes, as well as to other dynamical systems in general.

INDEX TERMS Food modeling, neural network, real-time, sensors, simulation, training algorithm.

I. INTRODUCTION

A. BACKGROUND

The food industry is a sector that faces major challenges, including adaptation to consumer demand, ensuring the quality of food processing and sustainability. To achieve these challenges by improving productivity and decreasing energy expenditure, the industry needs to make a more intelligent use of data across the entire supply chain [1]. This goal can be achieved by providing simulations of the cooking process, connecting it to the real world via IoT sensors and taking advantage of big data [2], [3], [4]. The main interest of accurate real-time simulations of food processing is to optimize processes and improve decision-making in the food industry.

The most accurate and robust models in food science applications with respect to real physical processes are physics-based models, such as the ones solved by the finite element method (FEM) or the finite volume method. However, complex physics-based models are hard to solve in real time [5]. In this context, neural networks (NN) stand out as a highly promising method for real-time applications [6]. The combination of NN with computer vision techniques has been widely used in problems related to food quality and safety detection [7], [8], [9], [10].

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

B. RELATED WORK ON FOOD COOKING PROCESS MODELING WITH NEURAL NETWORKS

Machine learning techniques and the use of NN have great potential in the real-time simulation of food processing and cooking [11], [12]. However, most of the problems are tackled from a steady-state perspective and the problems that simulate the whole transient process do not allow modifying conditions during its execution. For example, we find an adaptive neurofuzzy inference system, a hybrid model that utilizes fuzzy logic and a multilayer perceptron (MLP), to predict the convective heat transfer coefficient during deep-fat frying of pantoa [13]. Their model relies on the constant frying temperature and frying time as input, but it cannot modify them during the deep-fat frying process. In [14], a MLP is trained to obtain the oil uptake of rice flour in a batter-coated fried system. The MLP presented uses parameters obtained after finishing the food processing as inputs, which makes it unsuitable for simulating the evolution of food parameters during the cooking process. Another paper uses AI-based reduced order models trained on data obtained from physics-based FEM in the context of convection oven heating processes [15]. However, no details of the NN architecture used in the reduced order model or the training algorithm employed are shown. The NN proposed by the authors needs to receive real-time oven temperature, preventing this method from being used offline or to simulate the full cooking of the food. In [16] the authors propose to use a MLP trained from data obtained from FEM. They study the cooking of pancakes with frying pans on induction hobs. The solution is computationally efficient, and can be used in real time with pan temperature measurements using external sensors. The main drawback is that the MLP is trained as a feedforward network to make single time-step predictions, which results in the model not capturing the real dynamics of the problem and the increase of the error when simulating the complete cooking of the food. The forecast function presented in that paper needs to know the evolution of the pan temperature for a few seconds in order to predict the complete cooking of the food, making the simulation sensor-dependent.

In summary, the main open challenge in the state of the art of food cooking process modeling with NN is to develop a NN model capable of simulating the complete cooking of food, just to predict final result. The NN model must also be able to be used in real time, adapting to unexpected changes.

C. RELATED WORK ON DYNAMICAL SYSTEMS MODELING WITH NEURAL NETWORKS

In the previous subsection we highlighted relevant work on modeling cooking process using NN. However, this problem can also be generalized as the modeling of a dynamical system. Therefore, in this subsection we describe related work in modeling dynamical systems using NN.

In the modeling of dynamical systems using NN, the use of Recurrent Neural Networks (RNN) stands out [17], [18]. For example, control schemes have been improved by integration of RNN, in cases such as the modeling of a motor [19] or the prediction of the position and velocity of a vehicle [20]. In robotics we also find different approaches, such as the use of RNN and MLP for the modeling and control of a robot that manipulates food [21]. In the case of multistep prediction applied to a quadrotor, the use of RNN also stands out, improving the prediction of physical models [22], [23].

Another NN structure suitable and present in the state of the art for the simulation of dynamic systems is Nonlinear AutoRegressive with eXogenous inputs Neural Network (NARX-NN) [24], [25], [26]. It can be found in many applications, such as joint torque estimation [27], charge estimation of lithium-ion batteries [28], groundwater level prediction [29], health [30] or signal modeling [31].

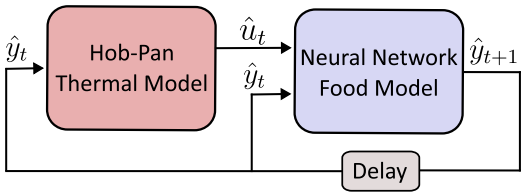
In NARX structures, it is important to use appropriate algorithms for system identification and simulation. A work proposes a cascaded evolutionary algorithm for nonlinear system identification, utilizing radial basis function NN and correlation functions to improve input selection and parameter estimation [32]. Their approach demonstrates the ability to identify complex dynamical systems effectively, showing the potential of evolutionary algorithms in this context. The application of coevolutionary algorithms is also promising in the field of black-box nonlinear system identification using NARX models [33].

NARX-NN is a suitable structure to perform multi-step prediction of nonlinear dynamical systems. However, the development of novel training algorithms adapted to multi-step predictions is still an open challenge [34], and in particular to any prediction horizon.

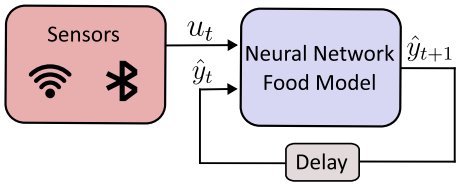
D. CONTRIBUTIONS

In view of the challenges listed in the related work on food cooking process modeling, as well as the related work on dynamical systems modeling in general, our main contributions are:

- A novel NN training algorithm adapted for modeling dynamical systems with any prediction horizon and receiving unexpected inputs in real time.
- We apply the novel training algorithm on a NARX-NN to simulate food cooking. We present how to train a data-driven NARX-NN from physics-based FEM data, capturing the dynamics of the problem and minimizing the size and complexity of the NN.
- The implemented NARX-NN can operate in real time by receiving information from temperature sensors in the kitchen, adapting to unexpected changes or user actions. We have also developed a novel hybrid model using analytical thermal models to simulate the behavior of the cooktop-pan system and, together with the NARX-NN, it can predict the complete cooking of food in milliseconds, being able to make decisions in advance.
- The studied case is the cooking of pancakes on induction hobs. We tested and compared our NARX-NN



(a) Offline Mode



(b) Online Mode

FIGURE 1. (a) Offline Mode, where the external inputs are estimated from a thermal model; (b) Online Mode, where the external inputs are measured in real time.

with the original FEM data, showing negligible error. Additionally, its performance has been validated in real cooking scenarios. It has also been verified that the NARX-NN trained with this methodology not only performs correctly in the ranges of values that appear in the data used during training, but the out-of-distribution generalization test indicates that the NARX-NN learns the real dynamics of the problem and is not over-fitted.

E. PAPER STRUCTURE

The paper is divided as follows: Section II formulates the problem, showing the notation to be used in the paper. Section III motivates the use of NARX-NN in our context. Section IV shows the main contribution of the paper, the two-phase training algorithm. Section V shows experiments and results of the NARX-NN trained with our training algorithm, both with simulated data and in real scenarios. The paper continues with a discussion in Section VI and ends with conclusions in Section VII.

II. PROBLEM FORMULATION

Consider the state of a food, given for example by its temperature or color at different points, at a time instant t by $y_t \in \mathbb{R}^m$. The external inputs at t, for example the temperature of the pan, are represented by $u_t \in \mathbb{R}^n$. The statics inputs, assumed to be constant during the entire cooking process, such as the initial thermal properties or the geometry of the food, are represented by $S \in \mathbb{R}^p$. Consider a function g that estimates the state of the food at the next time instant $\hat{y}_{t+1}$ from S, u_t and $\hat{y}_t$:

$$\hat{y}_{t+1} = g(S, u_t, \hat{y}_t) \tag{1}$$

Simulating food cooking aims to calculate over a time period T its state evolution $Y_{1,T} \in \mathbb{R}^{m\times T}$

$$Y_{1,T} = [y_1, y_2, \ldots, y_T]. \tag{2}$$

If we know y_0, and we know over the time period the external inputs $U_{0,T-1} \in \mathbb{R}^{n\times T}$

$$U_{0,T-1} = [u_0, u_1, \ldots, u_{T-1}], \tag{3}$$

we can iterate T times the function g until we get $\hat{Y}_{1,T} \in \mathbb{R}^{m\times T}$

$$\hat{Y}_{1,T} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T]. \tag{4}$$

The first problem to address in this paper is to obtain a function g using NN that minimizes the error between $Y_{1,T}$ and $\hat{Y}_{1,T}$.

The second problem is to obtain $U_{0,T-1}$. For this, two solutions are proposed in this paper: Online Mode, in which we use the sensors available in the hob to obtain these parameters in real time [35], and Offline Mode, in which we use an analytical thermal model of the hob-pan system to estimate these external variables and simulate the complete cooking of the food [36]. A schematic of these two approaches can be seen in Fig. 1. In Online Mode, the function g that solves the NN is similar to (1), while in Offline Mode the thermal model estimations are used in each time step t as external inputs $\hat{u}_t$:

$$\hat{y}_{t+1} = g(S, \hat{u}_t, \hat{y}_t) \tag{5}$$

III. SIMULATION OF FOOD COOKING WITH NARX-NN

For the problems and challenges we face in this paper, the most suitable NN structure is NARX-NN. This architecture allows to obtain executable models in real time, due to its simple structure. Moreover, in each iteration of the model only one time step is predicted, being this very flexible and allowing us to adapt to unexpected changes. This architecture allows us to iterate the model for any prediction horizon.

In NARX-NN there is a nonlinear function f that uses buffers of size k, called tapped delay lines (TDL) and MLP, to estimate $\hat{y}_{t+1}$ from external inputs and past state estimations. By adding the static parameters S, we can write the function f as g (5) using TDL:

$$\hat{y}_{t+1} = f(S, u_t, u_{t-1}, \ldots, u_{t-k}, \hat{y}_t, \hat{y}_{t-1}, \ldots, \hat{y}_{t-k}) \tag{6}$$

where $\hat{y}_{t+1}$ has a prediction error δ_t with respect to the actual y_{t+1}:

$$y_{t+1} = \hat{y}_{t+1} + \delta_t \tag{7}$$

The objective of the NN training is to minimize δ_t. The NARX-NN architecture used in this paper is shown in Fig. 2. The NARX-NN architecture includes hyperparameters that need to be optimized during training (Table 1).

IV. TWO-PHASE NARX-NN TRAINING ALGORITHM

There are two main approaches for training NARX-NN [37]: Training with teacher forcing (TF) and training without teacher forcing (no-TF). These methods are also known as series-parallel architecture and parallel architecture, respectively [38]. In the TF approach, the model uses the actual

TABLE 1. Hyperparameters of a NARX-NN, including those related to architecture and training. The subscripts *f* and *nf* indicate the first and second training phases, respectively.

Hyperparameter	Type	Description
Num. of hidden layers L	Architecture	Number of hidden layers in the neural network.
Neurons per hidden layer H	Architecture	Number of neurons in each hidden layer of the neural network.
Activation function σ	Architecture	Non-linear activation function applied to the outputs of each neuron.
TDL buffer size k	Architecture	Number of past external inputs and past states used as network input.
Num. of epochs n_f, n_{nf}	Training	Number of times the entire training dataset is processed during training.
Batch size B_f, B_{nf}	Training	Training examples processed before the model is updated during each epoch.
Cost function J_f, J_{nf}	Training	Measurement of the error between network predictions and actual states.
Early stop E	Training	Threshold for terminating the training process if no improvement is observed.
Learning rate η	Training	Controls the scale of model changes when it is updated based on the cost function.
Minimun learning rate $\min_\eta$	Training	Lower bound on the learning rate during training.
Learning rate decay factor α	Training	Factor by which the learning rate is reduced after a certain number of epochs.
Optimizer	Training	Algorithm used to update the network's weights based on the calculated gradients.

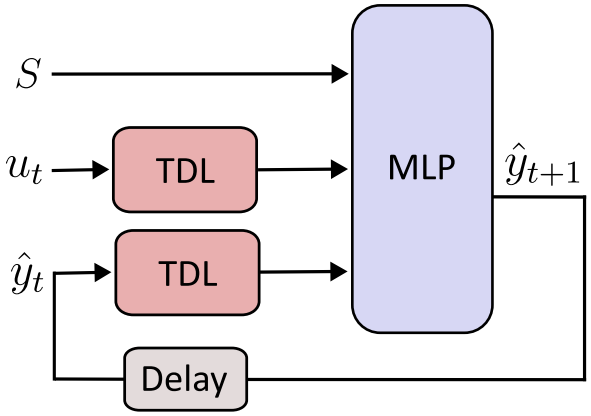
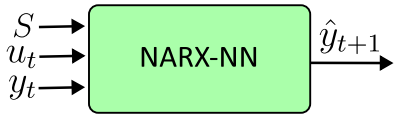
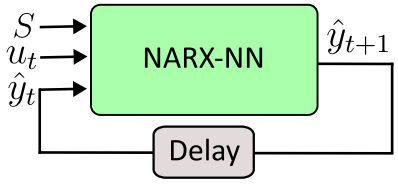


FIGURE 2. Schematic of NARX-NN architecture. The model estimate $\hat{y}_{t+1}$ from static inputs *S*, *k* past external inputs *u* and *k* past state estimations $\hat{y}$. The architecture consists of using TDL buffers, MLP and autoregressive connections.



(a) Training with TF



(b) Training with no-TF

FIGURE 3. (a) Training NARX-NN with TF, where y_t is the actual state of the previous time step; (b) Training NARX-NN with no-TF, where $\hat{y}_t$ is the NARX-NN previous estimate.

states of the k previous time steps as input during training:

$$\hat{y}_{t+1} = f(S, u_t, u_{t-1}, \ldots, u_{t-k}, y_t, y_{t-1}, \ldots, y_{t-k}) \quad (8)$$

The no-TF approach involves using the model state estimations from the k previous time steps as input, as in (6). A schematic of these two approaches is shown in Fig. 3.

Algorithm 1 Proposed Two-Phase Training Algorithm

Input: Model M, Epochs n_f, n_{nf}, Batch Size B_f, B_{nf}, Early Stop E, Learning Rate η, Minimum Learning Rate Min_η, Decay Factor α, Training set D_T, Validation set D_V
Output: Trained model M
// First training phase (with TF):
for n_f epochs **do**
  for each subset i of B_f series of D_T **do**
    Feedforward in M with TF on i
    Compute J_f on i
    Backpropagation in M
    Update weights and bias of M
  end for
  Feedforward in M with no-TF on D_V
  Compute J_{nf} on D_V
  if J_{nf} did not improve in the last E epochs **then**
    break
  end if
end for
// Second training phase (with no-TF):
for n_{nf} epochs **do**
  for each subset i of B_{nf} series of D_T **do**
    Feedforward in M with no-TF on i
    Compute J_{nf} on i
    Backpropagation in M
    Update weights and bias of M
  end for
  Feedforward in M with no-TF on D_V
  Compute J_{nf} on D_V
  if J_{nf} did not improve in the last E epochs **then**
    if $\eta \geq \min_\eta$ **then**
      Reduce η by a factor α
    else
      break
    end if
  end if
end for
return M

NARX-NN uses the previous state estimations to get the next predictions during inference. However, during training

the actual states y_t are available. This allows training the NN with TF as if it were a feedforward NN. Training the model with TF allows faster convergence and training. However, in problems where the behavior of a dynamical system is going to be simulated, training the model with TF actually optimizes it for single-step predictions [22], [39]. In this way, the training of the NN converges quickly but does not really learn the complete dynamics of the problem, obtaining models that do not generalize correctly.

In the state-of-the-art solutions using NARX-NN, the models are trained with TF, due to the numerical instabilities in the calculation of the gradients caused by training directly with no-TF. To solve this, in this paper we propose a two-phase NARX-NN training algorithm: A first phase in which the model is trained with TF, and a second phase in which the model is trained with no-TF. This contribution allows to take advantage of TF to quickly initialize the weights of the NN as accurately as possible and then train the model with no-TF, getting a more robust model during inference. This avoids the slow convergence and the numerical instabilities that would result from training only with the no-TF approach.

In Algorithm 1, the proposed method for NARX-NN training is shown in pseudo-code. The hyperparameters used in both training phases are described in Table 1. In addition, the parameters D_T and D_V refer to the subset of the dataset used during training and validation respectively. The early stop algorithm avoids overfitting and overtraining [40]. The learning rate decay algorithm is also used to improve convergence and generalization.

In the first training phase the model is trained with TF as if it were a non-recurrent NN, where for each instant t, the input of the model is $\{S, u_t, y_t\} \in D_T$. This training can be performed with high-level functions from APIs such as Keras [41] with Tensorflow [42]. As a novelty, after each training epoch, the model M performs the feedforward algorithm with no-TF on D_V, computing J_{nf}. If during E epochs J_{nf} does not improve, or the n_f epochs are performed, the first training phase is completed.

The key in the second training phase is that the model is trained with no-TF, where for each instant t, the input of the model is $\{S, u_t, \hat{y}_t\}$, where $\{S, u_t\} \in D_T$ and $\hat{y}_t$ is obtained from the previous estimation. To achieve this, model M processes each series of D_T sequentially, calculating $\hat{y}_{t+1}$ for each time instant t. Since we cannot parallelize the processing of a series as in the case of a non-recurrent network, training with no-TF is computationally more expensive.

V. EXPERIMENTS AND RESULTS

A. CASE STUDY: PANCAKE COOKING

The dataset of simulations of pancake cooking in frying pans on induction hobs from [16] has been used to train, validate and test the NN. That physics-based model uses transient heat transfer as well as mass transfer to simulate pancake cooking. The model takes into account the changes in thermal properties of the food during the cooking, which increases the non-linearity and complexity of the problem. Moreover, this model not only allows to simulate the thermodynamics of the system, but it is also able to obtain the pancake weight loss as well as to estimate the color evolution on both sides of the pancake. The main reason for choosing this case study is that there are several previous works in the development of the physics-based FEM model in this context, which has generated a complex, computationally expensive model, validated with real experiments, that can accurately simulate the cooking of pancakes [43]. This model was used to simulate pancake cooking under different conditions chosen randomly within pre-defined ranges, in order to obtain a dataset with a variety of different cooking situations. The recipe for the pancake batter is invariant in all simulations, and can be found at [44].

TABLE 2. Pancake cooking parameters saved in the database. External inputs and states are saved every second.

Parameter	Range	Type
Initial batter weight B	[60, 120] g	Static Input
Cooking side	[0, 1]	External Input
Applied power P_{app}	[0, 2200] W	External Input
Pan temperature T_{pan}	[0, 220] °C	External Input
Weight loss W_{loss}	[0, 18] %	State
Average temperature T_a, T_b	[0, 185] °C	State
Average lightness L_a, L_b	[30, 60] L*	State
Absorbed power P_{abs}	[0, 2700] W	State

All simulations have 2 steps: A first step in which the pan is preheated between 80 and 150 s, and a second step in which the pancake cooking is simulated, with initial batter weight $B \in [60, 120]$g. Each side of the pancake is cooked between 30 and 90 s. Therefore, each cooking simulation takes between 140 and 330 s considering both steps.

During the preheating of the frying pan, a target temperature $T_{target} \in [120, 220]$ °C is set. This target temperature is used with the PI control algorithm [45] to calculate in each instant of the simulation the power applied to the frying pan, with $P_{app} \in [0, 2200]$ W. In addition, T_{target} can be randomly modified, within the described temperature range, up to two times during the cooking simulation.

The properties of the pan used in the model are the following: specific heat $c_p \in [300, 450]$ J g^{-1} K^{-1}, conductivity $\kappa \in [50, 220]$ W m^{-1} K^{-1}, thermal contact conductance $h_c \in [50, 100]$ W m^{-2} K^{-1}, emissivity $\epsilon \in [0.5, 1]$, density $\rho = 7800$ kg m^{-3}. The geometry of the pan is constant in all simulations, consisting of 180 mm diameter and 5 mm thickness.

A total of 1400 pancake cookings have been simulated with the model. A database that is suitable for the training of NNs has been generated from these simulations. For this purpose, during the simulations, the values corresponding to the parameters shown in Table 2 were stored for each second of cooking. The NN trained on this dataset should be capable of predicting the parameters that describe the cooking state of the pancake: Percentage of weight loss with respect to the initial batter W_{loss}, average temperature of both sides

of the pancake T_a, T_b, power absorbed from the pan by the pancake P_{abs} and average lightness L^* of the standard color space (CIELAB) of both sides of the pancake. We consider as external input the temperature of the frying pan T_{pan} as well as the power of the induction hob P_{app}. Another external input is the side of the pancake being currently cooked, where Side $\in [0, 1]$. The only static input is the batter weight B.

Although the pan in each of the simulations has random thermal properties within the ranges described above, these parameters are not included in the database since the NN will have T_{pan} and P_{app} as input parameters for each instant. The goal is to train a NN capable of predicting the evolution of pancake cooking regardless of the pan properties.

B. METRICS AND COST FUNCTION

It is necessary to select appropriate metrics, both to choose a loss function ℓ used in the cost function J during NARX-NN training, and to evaluate N predictions made by the proposed models $\hat{Y}_{1,N} = [\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_N]$ with respect to N ground truth samples $Y_{1,N} = [y_1, y_2, \ldots, y_N]$. A metric typically used as ℓ in NN training is the mean squared error (MSE), which for each parameter j (Note that each state y_i has m parameters) can be computed as:

$$\mathrm{MSE}(Y^j_{1,N}, \hat{Y}^j_{1,N}) = \frac{1}{N}\sum_{i=1}^{N}(y^j_i - \hat{y}^j_i)^2 \tag{9}$$

However, MSE is not a normalized measure with respect to the variance of the data, which makes this metric sensitive to the evaluation dataset, as well as sensitive to outliers that can be caused by noise in the database. For these reasons, we have found it more appropriate to use R^2:

$$\mathrm{R}^2(Y^j_{1,N}, \hat{Y}^j_{1,N}) = 1 - \frac{\sum_{i=1}^{N}(y^j_i - \hat{y}^j_i)^2}{\sum_{i=1}^{N}(y^j_i - \bar{Y}^j_{1,N})^2} \tag{10}$$

where $\bar{Y}^j_{1,N}$ is the mean of parameter j of the N ground truth samples, given by:

$$\bar{Y}^j_{1,N} = \frac{1}{N}\sum_{i=1}^{N} y^j_i \tag{11}$$

R^2 can be considered as a normalized version of MSE, where the predictive ability of the model is measured with respect to a model that always predicts $\bar{Y}^j_{1,N}$. This metric varies in the range $(-\infty, 1]$ where 1 is a perfect prediction.

We use R^2 as a performance metric, but for the cost function J we choose $-\mathrm{R}^2$ as loss function. We average the m parameters predicted by the model, in order to have a single metric to minimize during training:

$$J(Y_{1,N}, \hat{Y}_{1,N}) = \frac{1}{m}\sum_{j=1}^{m} -\mathrm{R}^2(Y^j_{1,N}, \hat{Y}^j_{1,N}) \tag{12}$$

In addition to R^2 we use root mean squared error (RMSE) and mean absolute error (MAE) as performance metrics instead of MSE as they are more robust to outliers and respect the original units of the parameters, making the metrics easy to interpret:

$$\mathrm{RMSE}(Y^j_{1,N}, \hat{Y}^j_{1,N}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y^j_i - \hat{y}^j_i)^2} \tag{13}$$

$$\mathrm{MAE}(Y^j_{1,N}, \hat{Y}^j_{1,N}) = \frac{1}{N}\sum_{i=1}^{N}\left|y^j_i - \hat{y}^j_i\right| \tag{14}$$

TABLE 3. Final hyperparameters of the NARX-NN, together with the values explored during the grid search.

Parameter	Value	Search
Num. of epochs n_f, n_{nf}	20, 500	20, 100, 250, 500, 750
Batch size B_f, B_{nf}	128, 32	16, 32, 64, 128, 256
Cost function J_f, J_{nf}	$-\mathrm{R}^2$, $-\mathrm{R}^2$	MSE, $-\mathrm{R}^2$
Early stop E	15	5, 10, 15, 20
Learning rate η	10^{-3}	10^{-2}, 10^{-3}, 10^{-4}
Min. learning rate $\min_\eta$	10^{-5}	10^{-4}, 10^{-5}, 10^{-6}
Decay factor α	2	2, 3, 4, 5
Optimizer	Adam	SGD, Adam, AdaGrad

C. ARCHITECTURE AND TRAINING HYPERPARAMETERS

The training of the NN was performed with 1400 simulations of pancake cooking. A total of 1100 simulations have been used as training set, 200 simulations as validation set and 100 simulations have been reserved as test set. This division ensures that the model is evaluated for overfitting and generalization.

The values of the final hyperparameters chosen are shown in Table 3, with the values at which the hyperparameters have been searched. A range of possible values for each of the hyperparameters has been defined to reduce the complexity of the search. To find the best values, the hyperparameters related to the NN architecture have been fixed ($L = 3$, $H = 256$, $\sigma =$ ReLU and $k = 5$) in order to have a model that can be trained relatively quickly but obtaining good metrics, and allows to try different configurations of hyperparameters related to the training algorithm. The strategy followed was to perform a random search [46] over the hyperparameters, with a total of 30 iterations. After completing the random search, the hyperparameters were fine-tuned sequentially. Specifically, each hyperparameter was optimized iteratively by testing all its possible values while keeping the rest fixed at the values obtained from the random search. The chosen optimizer has been Adam [47].

The next step was to optimize the hyperparameters related to the NN architecture. First, the activation function was chosen, obtaining the best results with ReLU. The Sigmoid and Tanh activation functions were also tested with worse results.

To optimize the number of neurons in the hidden layers H and the number of past external inputs and states k, a grid search was performed, where $L = 3$, $H \in \{64, 128, 256, 512, 1024\}$ and $k \in \{1, 2, 5, 10, 15\}$. In this search, we measured the performance of the model with the training, validation and test sets, calculating the cost function

TABLE 4. Grid search results for optimizing the number of neurons in the hidden layers H and the TDL buffer size k. The metrics shown are R^2, calculated with no-TF, in the training, validation and test sets.

	k	15	10	5	2	1
H	Split					
1024	Training	0.9989	**0.9993**	0.9987	0.9984	0.9973
	Valid.	0.9987	**0.9991**	0.9985	0.9977	0.9963
	Test	0.9939	**0.9953**	0.9899	0.9873	0.9869
512	Training	0.9988	0.9988	0.9991	0.9983	0.9964
	Valid.	0.9985	0.9985	0.9988	0.9977	0.9948
	Test	0.9933	0.9935	0.9948	0.9908	0.9828
256	Training	0.9986	0.9986	0.9985	0.9976	0.9969
	Valid.	0.9984	0.9984	0.9980	0.9971	0.9960
	Test	0.9912	0.9921	0.9919	0.9900	0.9822
128	Training	0.9983	0.9985	0.9978	0.9981	0.9945
	Valid.	0.9982	0.9983	0.9974	0.9975	0.9922
	Test	0.9911	0.9919	0.9881	0.9888	0.9779
64	Training	0.9956	0.9971	0.9954	0.9953	**0.9934**
	Valid.	0.9954	0.9965	0.9945	0.9944	**0.9943**
	Test	0.9659	0.9828	0.9678	0.9688	**0.9658**

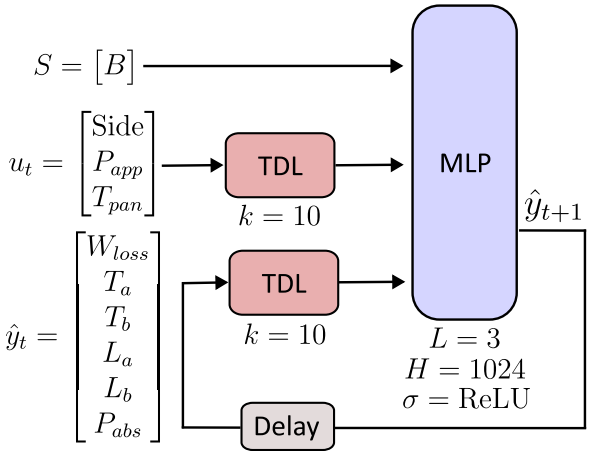
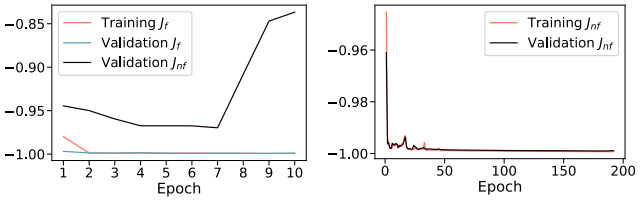


FIGURE 4. Summary of the final NARX-NN architecture for pancake cooking. The parameters used as input (S, u_t and $\hat{y}_t$) and output ($\hat{y}_{t+1}$), as well as the hyperparameters related to the architecture, are depicted.

with no-TF J_{nf}. The results are shown in Table 4, where the model with $H = 1024$ and $k = 10$, obtains the best metrics for all three sets (0.9993, 0.9991, 0.9953). As can be seen, these metrics show that there is no overfitting, obtaining good results also in the test cases. The metrics obtained in all models are very good, being the worst the model with $H = 64$ and $k = 1$ where we still get acceptable R^2 (0.9934, 0.9943, 0.9658). These results allow the number of neurons and of past external inputs and states to be adapted according to the availability of previous data and the computational capacity available, with the certainty that the performance of the model will be good. As a general trend, it is observed that as H increases, accuracy improves, although at 1024 neurons and above the trend begins to change. As k increases up to 10, performance improves too.

A summary of the final NARX-NN architecture for pancake cooking can be found in Fig. 4, where the parameters used as static inputs, external inputs and past outputs are also



(a) J of the first phase of training (b) J of the second phase.

FIGURE 5. (a) J obtained in the first phase of training. Training and validation J_f converge quickly, while J_{nf} diverges from epoch 7; (b) J obtained in the second phase of training where both training and validation J_{nf} converge.

detailed. L has been set to 3, since testing other values resulted in worse metrics.

D. TRAINING METRICS

As detailed in Section IV, a two-phase training algorithm has been developed. In order to show the importance of our training algorithm, we show the cost function J performance in both training phases, calculating it both with teacher forcing (J_f) and without teacher forcing (J_{nf}). Fig. 5a shows the cost function J obtained during the first phase of training. Note that we use $-R^2$ as J, so the minimum value is -1. This training has been performed with the hyperparameters and architecture detailed in the previous section. As can be seen, training and validation cost function with teacher forcing J_f converge rapidly to values close to -0.99. Meanwhile, if we calculate the performance of the model with no-TF, we observe how validation J_{nf} converges to values close to -0.97 in epoch 7, and in the following epochs this value starts to diverge. Note that J_{nf} calculates the model performance using its own predictions, in the same way that the model is then used in inference. While validation J_{nf} diverges, training and validation J_f continue to improve slightly. Similar behavior has been observed for all models trained with TF, confirming that training these models only with TF does not give the best performance when used in inference. This shows that training with TF overfit the models to predict only one time step, making their performance worse when we predict various time steps.

Fig. 5b shows the metrics obtained during the second training phase. J_{nf}, both during training and validation, converges fast in 2 epochs to values below -0.99. The metrics continue to improve slightly until the end of the training algorithm. These results on J_{nf} cannot be obtained by training only with TF.

E. TEST METRICS

In addition to analyzing the NN during training, metrics were obtained from 100 test simulations. These metrics evaluate and compare how the NARX-NN trained with the first phase, and the architecture trained with the two phases, predict each parameter related to the pancake state. Table 5 shows the MAE, RMSE and R^2 for each of the parameters predicted by the NN. These metrics are calculated for each

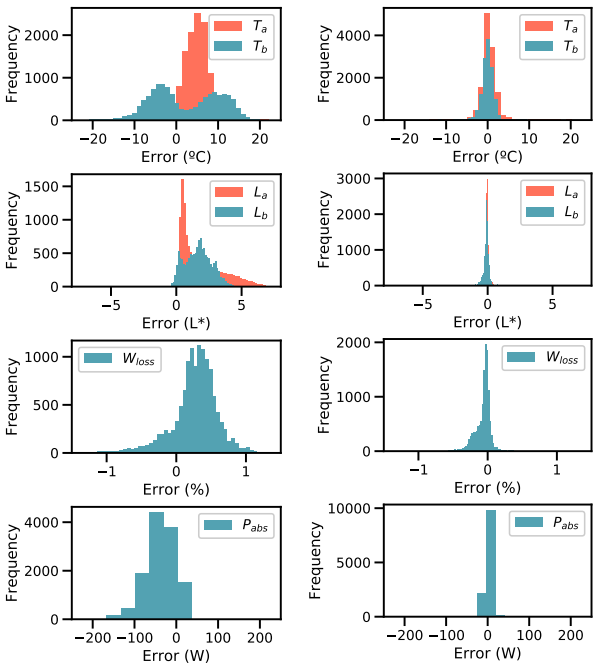
TABLE 5. Mean and standard deviation of MAE, RMSE and R^2 for each of the six parameters predicted by the NARX-NN, calculated in the 100 test simulations. The results of the NARX-NN trained with the two training phases (with TF and with no-TF) outperform those obtained with the NARX-NN trained only with the first phase (with TF).

Parameter	Trained only with first phase			Trained with both phases		
	MAE	RMSE	R^2	MAE	RMSE	R^2
W_{loss} (%)	0.37 ± 0.11	0.41 ± 0.12	0.9859 ± 0.0173	0.08 ± 0.05	0.11 ± 0.06	0.9990 ± 0.0006
T_a (°C)	5.48 ± 1.6	6.53 ± 1.82	0.9753 ± 0.0112	1.25 ± 0.53	2.08 ± 0.86	0.9975 ± 0.0014
T_b (°C)	7.01 ± 1.45	8.12 ± 1.53	0.7720 ± 0.1841	1.04 ± 0.48	1.46 ± 0.59	0.9926 ± 0.0073
L_a (L*)	1.86 ± 0.48	2.37 ± 0.64	0.5069 ± 0.3350	0.12 ± 0.09	0.16 ± 0.11	0.9979 ± 0.0020
L_b (L*)	1.64 ± 0.48	1.84 ± 0.54	0.7128 ± 0.2118	0.17 ± 0.12	0.22 ± 0.16	0.9960 ± 0.0039
P_{abs} (W)	49.04 ± 10.16	85.88 ± 21.86	0.7833 ± 0.0547	5.27 ± 2.53	14.5 ± 11.62	0.9903 ± 0.0094

cooking scenario simulation and the table shows the mean and standard deviation obtained. The NN is used with no-TF, using at each instant the k estimations previously made. We start from the initial conditions of each test and set the static variables. At each iteration of the NN, the state of the pancake is calculated for the next time instant, using the external variables (Side, P_{app} and T_{pan}) that are read for each time instant directly from the test data. In the real environment in which this model is used, P_{app} and T_{pan} are obtained directly from the measurements of the hob sensors or are calculated analytically.

As can be seen in Table 5, the metrics obtained with the NN trained with both training phases, with TF and with no-TF, outperforms the metrics obtained with the NN trained only with TF. The model trained with both phases get mean $R^2 > 0.99$ for all six target variables, while the model trained only with the first phase only obtains values close to 0.99 in the calculation of W_{loss} and T_a, with a mean R^2 of 0.9859 and 0.9753 respectively. For MAE and RMSE metrics, lower mean values and standard deviations are obtained for the NN trained with both phases.

Fig. 6 shows the error histograms obtained for each of the parameters predicted by the NN. The error is calculated for all the time instants of the 100 test simulations. Each time instant is predicted with no-TF, and the actual value is subtracted from the value predicted by the NN. The error histograms for the NN trained only with the first phase show bigger error dispersion than the model trained with both phases, whose error is close to 0 in all parameters.



(a) Error histograms with the NN trained with only first phase

(b) Error histograms with the NN trained with both phases

FIGURE 6. Error histograms with the NN trained only with first phase, with TF (a); and with both training phases, with TF and no-TF (b). The error shown is for the 100 test simulations. The NN trained with both phases outperforms the model trained with only the first phase, with an error close to 0 for all parameters.

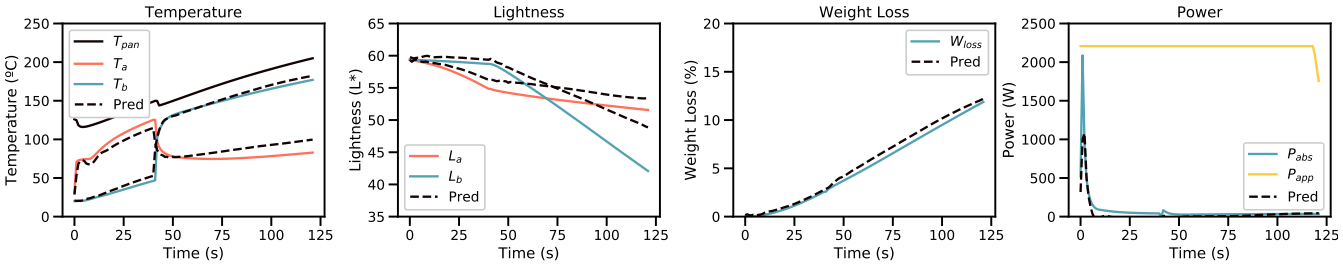
F. TEST PLOTS

In addition to analysing the previous metrics, it is important to visualize the performance of the trained models when predicting at each cooking time instant the six target variables. The following tests are performed in the same way as in the previous section using no-TF, simulating the behavior of the network in a real environment: At each time instant, the static variables of the pancake and the k external inputs are read directly from the test data, and the k previous state estimations of the NN are used as input to the NN.

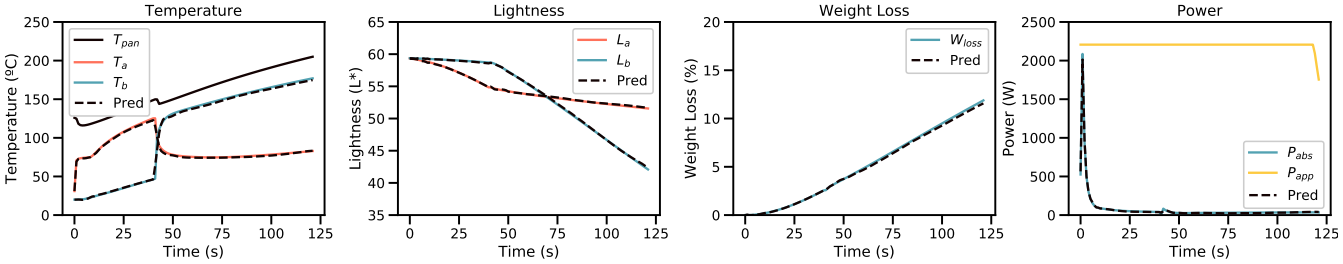
Fig. 7 shows the plots of the first two test simulations, A and B, with the purpose of visualizing the comparison of the prediction performance with the model trained only with the first phase and with the model trained with both phases. The information from these two specific tests can be found in Table 8, where the static variables of the pancake, the thermal parameters of the pan used in that simulations, as well as the target temperature of the pan are detailed. The seconds used to preheat the pan and to cook both sides of the pancake are also detailed. The metrics obtained for both tests can be found in Table 6. As can be seen again in the metrics and plots, the model trained with both phases outperforms the model trained only with TF. The most evident difference of training only with the first phase or with both phases can be observed especially in the estimation of L_a and L_b, where in the calculation of these parameters in Fig. 7a and Fig. 7c, the discrepancy between the real values of the simulation and the predictions of the NN grow as the iterations increase. This

TABLE 6. Metrics obtained for each of the six parameters in test A and B, with the NARX-NN trained only with the first phase and with both phases. The results obtained with the architecture trained with the two phases have negligible error. Plots with the actual data of these tests and the estimations of the NARX-NN are shown in Fig. 7.

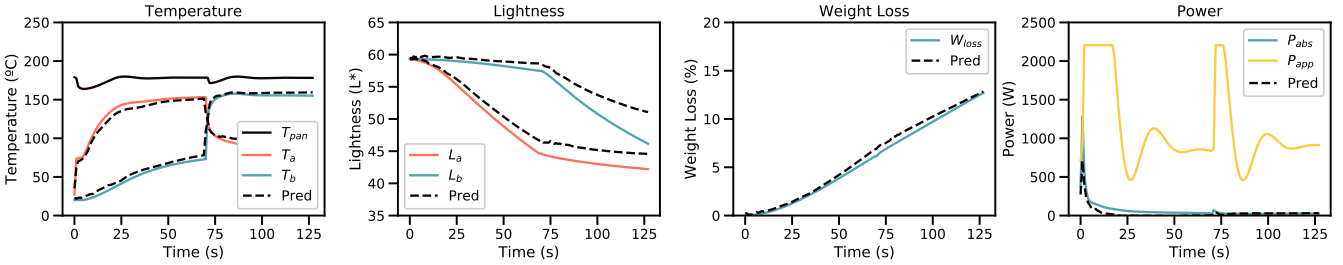
Parameter	Trained only with first phase						Trained with both phases					
	Test A			Test B			Test A			Test B		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
W_{loss} (%)	0.45	0.50	0.9824	0.39	0.44	0.9879	0.11	0.15	0.9985	0.04	0.05	0.9998
T_a (°C)	3.44	5.25	0.9921	4.11	5.22	0.9908	1.58	2.69	0.9979	0.98	2.35	0.9981
T_b (°C)	9.60	10.80	0.5225	6.80	7.68	0.9351	1.03	1.49	0.9909	0.78	1.48	0.9976
L_a (L*)	2.44	3.19	0.6830	1.63	2.13	0.7517	0.07	0.09	0.9997	0.11	0.14	0.9990
L_b (L*)	1.4	1.52	0.5953	1.57	1.73	0.9137	0.06	0.07	0.9991	0.08	0.10	0.9997
P_{abs} (W)	47.96	111.53	0.7215	36.68	66.81	0.7277	5.41	9.99	0.9978	3.96	5.65	0.9981



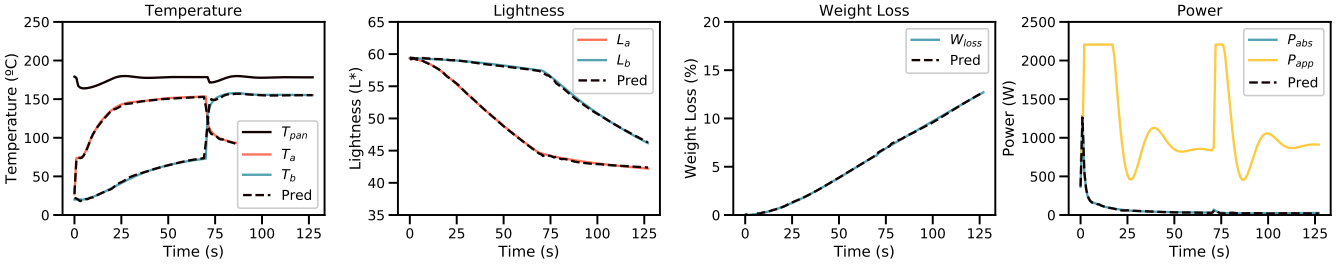
(a) Test A with the model trained only with the first phase



(b) Test A with the model trained with both phases



(c) Test B with the model trained only with the first phase



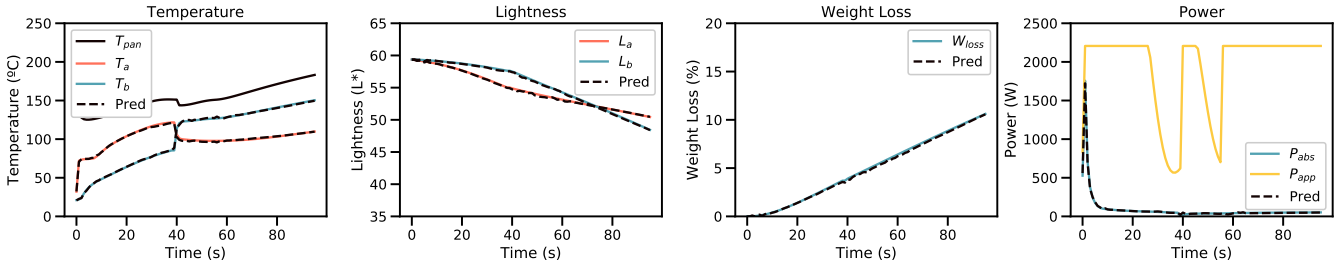
(d) Test B with the model trained with both phases

FIGURE 7. Plots of all inputs and states in test A and B, with the model trained only with the first phase and with both phases. The prediction is performed with no-TF, and the estimations of the model trained with both phases have negligible error.

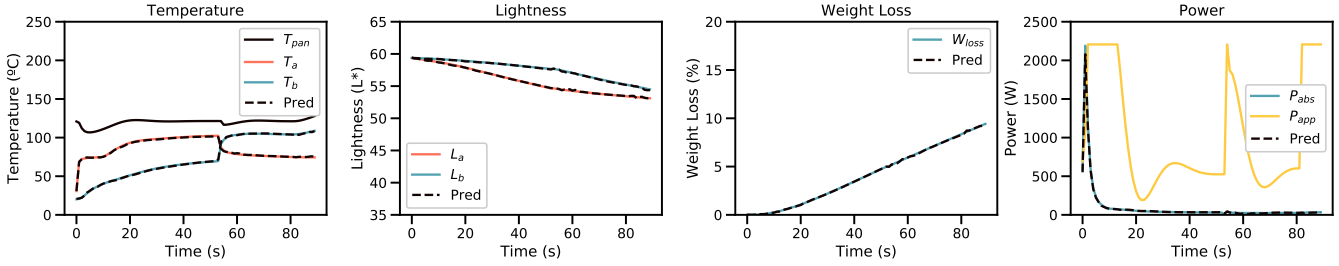
is because by training only with TF, the model does not learn the real dynamics of the problem since in each iteration of the training actual data is used instead of using the estimations of the model. In Fig. 7b and Fig. 7d however, it is observed

TABLE 7. Metrics obtained for each of the six parameters in 4 tests, with the NARX-NN trained with both phases. The results of the architecture with the proposed training algorithm have negligible error. Plots with the actual data of these tests and the prediction are shown in Fig. 8.

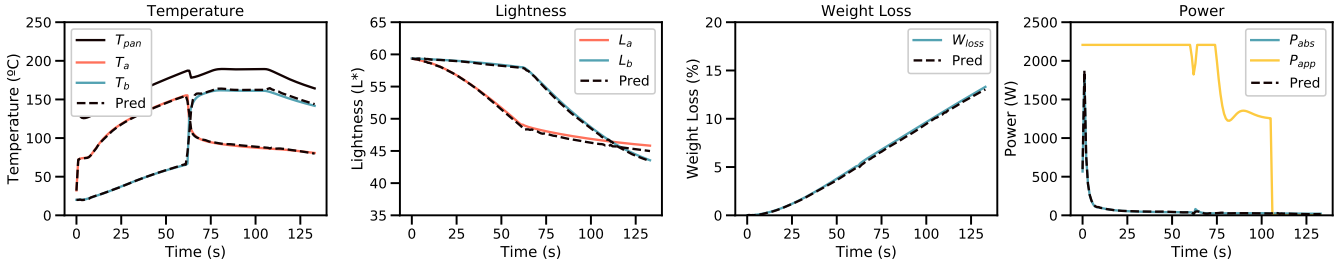
Parameter	Test 1			Test 2			Test 3			Test 4		
	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2	MAE	RMSE	R^2
W_{loss} (%)	0.11	0.13	0.9985	0.03	0.04	0.9998	0.14	0.17	0.9985	0.18	0.25	0.9977
T_a (°C)	0.60	0.78	0.9996	0.45	0.62	0.9995	1.6	3.07	0.9972	1.70	2.5	0.9970
T_b (°C)	0.72	0.97	0.9947	0.51	0.61	0.9977	0.86	1.45	0.9966	2.95	3.28	0.9765
L_a (L*)	0.09	0.11	0.9990	0.05	0.06	0.9981	0.14	0.16	0.9992	0.13	0.19	0.9991
L_b (L*)	0.07	0.09	0.9989	0.04	0.06	0.9992	0.36	0.45	0.9907	0.13	0.16	0.9994
P_{abs} (W)	4.77	7.93	0.9983	6.16	14.45	0.9971	3.71	7.29	0.9983	3.37	7.07	0.9984



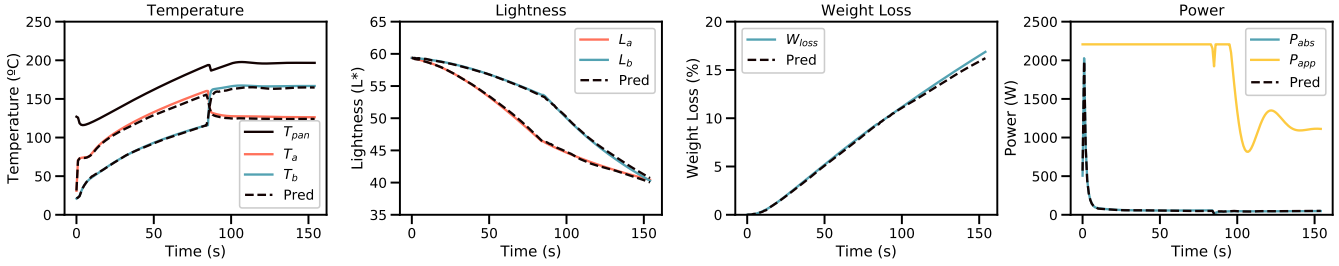
(a) Test 1



(b) Test 2



(c) Test 3



(d) Test 4

FIGURE 8. Plots of all inputs and states in 4 tests, with the NARX-NN trained with both phases. These tests visualize the performance of the proposed architecture in cooking cases where the target temperature of the pan is changed unexpectedly. The model has negligible error for all the six state parameters.

how the model predicts almost perfectly L_a and L_b, preventing the error from increasing over time. A similar effect can be observed in the calculation of T_a and T_b. P_{abs}, in the case where the model is trained only with the first phase, fails to predict the first few seconds. These first seconds in the calculation of P_{abs} are very important since they show

TABLE 8. Static parameters of the pan and pancake, cooking times and target temperature of the pan in simulated test A and B. Plots with the actual data of these tests and the estimations of the NN are shown in Fig. 7.

Test	B (g)	Preheat (s)	Side A (s)	Side B (s)	T_{target} (°C)	c_p (J g^{-1}K^{-1})	κ (W m^{-1} K^{-1})	h_c (W m^{-2} K^{-1})	ϵ (-)
A	115	87	40	80	210	369.40	96.70	74.85	0.60
B	104	114	69	57	180	281.76	118.27	58.05	0.66

TABLE 9. Static parameters of the pan and pancake, cooking times and target temperature of the pan in 4 test simulations. In these tests the target temperature of the pan changes during cooking (Change Target). Plots with the actual data of these tests and the prediction of the NN are shown in Fig. 8.

Test	B (g)	Preheat (s)	Side A (s)	Side B (s)	T_{target} (°C)	Change Target (s)	c_p (J g^{-1}K^{-1})	κ (W m^{-1} K^{-1})	h_c (W m^{-2} K^{-1})	ϵ (-)
1	62	96	38	56	140, 150, 195	1, 54	356.02	162.64	58.61	0.73
2	76	106	52	36	120, 160	79	345.85	206.05	51.23	0.72
3	113	83	61	71	140, 190, 120	1, 105	296.73	246.69	57.76	0.87
4	65	80	84	69	130, 200	1	332.50	104.30	79.12	0.52

how much power is being absorbed by the food at ambient temperature when it is added to the preheated frying pan. The prediction of W_{loss} is good in both models, however, the model trained with both phases minimizes the error.

After evaluating and visualizing the comparison of training the model with one or both phases, the performance of the model trained using both phases is demonstrated in more complex cooking scenarios. To this end, four simulations from the test data have been selected, with details provided in Table 9. Unlike the previous examples, during simulation the target temperature of the pan can change, so the table specifies the seconds in which these changes occur from the moment the pancake batter is placed in the pan. In Fig. 8 the prediction of the NN in these 4 tests is shown, together with the original values of the simulations. Table 7 shows the metrics obtained in these 4 tests. As can be seen, the pancake cooking prediction is robust to changes in target temperature during cooking, to different cooking times and to pans with different thermal properties.

G. GENERALIZATION TEST

All the parameters of the data from which the model has been trained, detailed in Section V-A, have a range of values including the values between which typical pancake cooking is found. For example, in the case of the amount of batter, between 60 and 120 grams, and maintaining the pan temperature during cooking between 120 and 220 °C. When the data are divided into training, validation and test set, all of them have the same range of values. Therefore, although the test cooking simulations are different than those of the training and validation subsets, the range of values of the sets is the same. We have performed different out-of-distribution generalization tests with the goal of ensuring that the architecture does not overfit to the range of values of the sets.

To perform these out-of-distribution generalization tests, the model with the best hyperparameters described in Section V-C has been trained, using the simulations that meet a condition for training and validation, and reserving the rest of the simulations for testing. This condition changes for each of the generalization tests that have been performed, and this can be consulted in Table 10.

TABLE 10. Out-of-distribution generalization tests. NARX-NN shows generalization capability in scenarios where higher batter amounts appear than those seen during training (G1 and G2). It also generalizes well in scenarios with pan temperatures different from those seen in training (G3 and G4).

Test	Split	No. of data	Range	R^2
G1	Training	1036	$B \in [60, 105)$ g	0.9991
	Validation	150	$B \in [60, 105)$ g	0.9988
	Test	214	$B \in [105, 120]$ g	0.9934
G2	Training	763	$B \in [60, 95)$ g	0.9993
	Validation	150	$B \in [60, 95)$ g	0.9990
	Test	487	$B \in [95, 120]$ g	0.9893
G3	Training	986	$T_{pan} \in [0, 190)$ °C	0.9993
	Validation	150	$T_{pan} \in [0, 190)$ °C	0.9990
	Test	264	$T_{pan} \in [190, 220]$ °C	0.9931
G4	Training	785	$T_{pan} \in [0, 180)$ °C	0.9989
	Validation	150	$T_{pan} \in [0, 180)$ °C	0.9985
	Test	465	$T_{pan} \in [180, 220]$ °C	0.9868

In G1, the model is trained and validated with simulations where $B \in [60, 105)$ and tested with simulations where $B \in [105, 120]$. The mean R^2 obtained during training, validation and testing exceeds 0.99. In G2, the same test is repeated but decreasing more the range of values in training and validation, where only the simulations where $B \in [60, 95)$ are going to be used, and the test is performed on the simulations where $B \in [95, 120]$. In this second case, an average of R^2 of more than 0.99 is obtained in the training and validation sets, and a value of 0.9893 is obtained in the case of the test data, being this a very high value considering that the number of simulations used during the training has been considerably reduced and they are mass quantities that the model has not learned during the training.

In G3 and G4, a test similar to the previous ones is performed but studying the impact of cooking with temperatures higher than those used by the model during training and validation. Therefore, the model is trained in both tests with the simulations where $T_{pan} \in [0, 190)$ and $T_{pan} \in [0, 180)$, respectively. Obtaining an R^2 of more

TABLE 11. Execution time of a 240 seconds pancake cooking simulation with the NN. H shows the number of neurons per layer, the parameters are the weights and bias of the NN and the speedup value is with respect to the original FEM model.

H	Parameters	Execution time (ms)	Speedup
1024	2191366	111.71	$\approx 5.37 \times 10^2$
512	571398	14.16	$\approx 4.23 \times 10^3$
256	154630	5.15	$\approx 1.16 \times 10^4$
128	44550	1.98	$\approx 3.02 \times 10^4$
64	14086	1.36	$\approx 4.41 \times 10^4$

than 0.99 during training and validation. G3 is tested with the simulations where $T_{pan} \in [190, 220]$, obtaining an R^2 of 0.9931 and G4 is tested with the data where $T_{pan} \in [180, 220]$, obtaining an R^2 of 0.9868. Again, in this last test, the R^2 drops below 0.99 but it is still good result, especially considering that the model has been trained with less data (785 training simulations and 150 validation simulations, versus the 1300 used if we have all the data) and is tested in cooking situations where the pan and pancake exceed 180 °C, which is the maximum observed by the NN during training.

With the metrics obtained in these generalization tests, it can be verified that the model trained with this method does not overfit to the range of values used during training and is able to learn the real dynamics of the problem and to generalize. This is an especially useful capability, since the NN will be robust to cooking situations that were not taken into account during training.

H. COMPUTATIONAL COST

One of the problems addressed in this paper is that the cooking simulations must be run in real time. Not only to adapt instantly to external changes, but also to predict at once how the system will evolve over time and make decisions based on this prediction. In the case of pancake cooking, the FEM model with a denser mesh presented in [43] takes 25 minutes to perform a simulation of a 240 second pancake cooking. The FEM model with the most optimized mesh presented in that paper takes about one minute to perform the same simulation. These times are obtained on an Intel Core i5 2.70 GHz. Running a 240 s simulation in one minute allows to simulate every second in real time. However, this approach does not allow to predict the complete cooking of the food in real time. It would take one minute to simulate the entire cooking, and for each change or alteration that occurs during cooking, it would take another minute.

In Table 11, the execution times of a 240 seconds pancake cooking simulation, with different numbers of neurons per layer H, on an AMD Ryzen 7 3750H 2.30 GHz can be observed. The speedup value shown is with respect to the minute that the optimized FEM model would take. The presented NNs have $k = 10$ and $L = 3$, which as seen in Section V-C, are the best performing architecture. Different numbers of H have been tested since it is the hyperparameter that has the most impact on the number of operations and execution time of the NN.

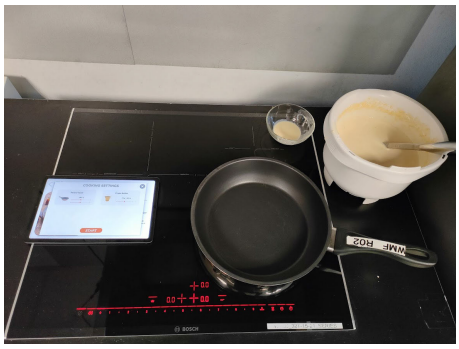


FIGURE 9. Cooking setup used during experimental validation. It consists of a PIF675DC1E induction hob y and a Balay 00570366 frying pan, an 18/10 stainless steel frying pan of 21 cm diameter.

As can be seen, the NN models can simulate 240 seconds of pancake cooking in few milliseconds, obtaining very high speedup values with respect to the original FEM model. In addition to meeting the requirements of being able to run in real time and predict the complete cooking of the food, these NN are light enough to be run on devices with less powerful CPUs such as smartphones or tablets, and still obtain good execution times. Especially the versions where $H \leq 512$, where the state estimations still obtain a $R^2 \geq 0.99$.

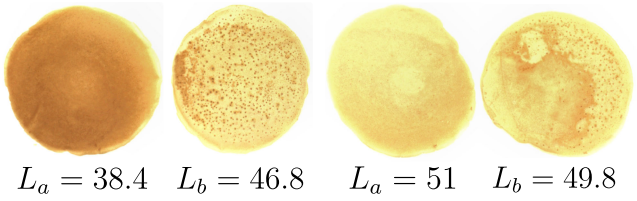
I. EXPERIMENTAL VALIDATION IN REAL SCENARIOS

So far, our proposal has been validated by comparing the results with simulations of the original FEM model of pancake cooking. This FEM model has already been positively validated with real experiments, so the NN trained from these data should also accurately simulate pancake cooking in real scenarios. To validate this accuracy, a series of pancakes have been cooked in real cooking scenarios, and an image of the pancake has been captured at the end of cooking on both sides. The most important parameter for predicting the cooking of a pancake is the color, or in this case the average L* obtained on each side, so this is measured in these tests. These experiments in real scenarios also help us to visualize that the model is not overfitted to the simulated data used during training.

The setup used (Fig. 9) is the Bosh PIF675DC1E induction cooktop, in particular its 21 cm cooking zone has been used with the frying sensor for the temperature control of the frying pan. Balay 00570366 is the frying pan used, an 18/10 stainless steel frying pan of 21 cm diameter. To extract the lightness value of the pancake sides after each cooking, an image is captured with similar light conditions to those used in [44]. The cooked pancakes are captured with a white background, so to extract the pixels that form the pancake, the OpenCV library in Python has been used: The image is converted from RGB to CIELAB color space, a mask is generated that eliminates the white pixels of the background ($L* = 100, a* = 0, b* = 0$), and finally the L* channel is averaged with the remaining pixels, corresponding to the pancake.

TABLE 12. Absolute Error (AE) in the calculation of the average lightness of the two sides of the pancake, in the case of using the Offline mode or the Offline mode. In each test, the target temperature of the induction cooktop is shown (T_{target}), how long each side has been cooked (Side A and Side B), and what was the lightness value obtained for both sides (L_a and L_b). The Online mode outperforms the Offline alternative, with less AE on both sides.

Test	T_{target} (°C)	Side A (s)	Side B (s)	L_a (L*)	L_b (L*)	Offline mode		Online mode	
						AE L_a (L*)	AE L_b (L*)	AE L_a (L*)	AE L_b (L*)
R1	180	46	37	40.2	42	4.53	4.93	3.7	3.2
R2	180	48	39	41.4	42	2.68	4.41	1.6	2.44
R3	180	62	27	37.2	49.2	4.82	0.10	2.39	1.95
R4	180	62	31	37.8	49.2	3.85	0.91	2.91	2.98
R5	180	67	28	38.4	46.8	2.59	2.26	1.25	0.17
R6	180	97	58	33	48	3.39	1.33	0.03	3.45
R7	180	93	90	36	46.8	0.62	3.57	2.53	5.19
R8	160	43	47	52.8	47.4	4.85	0.43	3.73	0.19
R9	160	41	44	51	49.8	2.39	1.44	1.88	1.1



(a) Test R5 (b) Test R9

FIGURE 10. Image obtained with the mean L* value of both sides of the pancake after being cooked with the R5 (a) and R9 (b) test conditions.

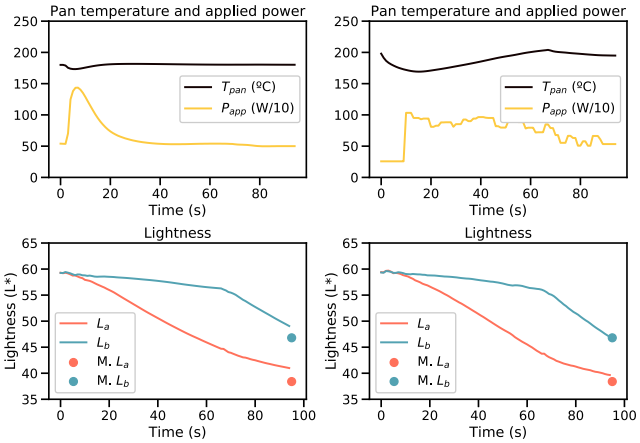
The information and results of the tests performed can be consulted in Table 12, where for each test the target temperature at which the pan is preheated, the time that each side of the pancake is cooked and the value of the average lightness obtained in each of the sides are shown. As a result, the absolute error of the mean lightness for each of the sides is shown, with the Offline and Online modes. An introduction to these two approaches has been made in Section II. In Fig. 10 the images obtained for both sides of the pancake after being cooked with the R5 and R9 test conditions can be seen. The side that is cooked first presents a much more uniform color over its surface, while the side that is cooked later presents a much more rough and irregular surface. This is due to the bubbles that are generated in the mass when the first side is being cooked.

In the 9 tests we have cooked pancakes in different conditions, with the purpose of obtaining more toasted and more raw results, and pancakes cooked for longer and shorter times. In all the tests, the pan was preheated to a target temperature, in this case 180 °C for the first 7 tests and 160 °C for the last two, since these temperatures are the most common in the cooking of pancakes. Once the pan is preheated, the pancake batter is added, where for these tests 95 g of batter has been used and each side is cooked for a specific time. During the cooking of the pancake, through one of the functionalities of the induction hob, the power that the induction hob is applying to the pan as well as the estimated temperature of the pan is obtained every second. This information can be used as input for our NN and test the Online mode, receiving this information in real time.

The Offline mode does not receive real-time information from the induction hob, but uses an analytical thermal model of the pan-hob system, particularly the state-space model developed in [36]. Using this thermal model, the heating of a frying pan on an induction cooktop is calculated if its thermal properties are known. This model was designed to simulate the behavior of the empty frying pan, and for this reason, the NN also predicts how much power the food absorbs. Knowing the power absorbed by the food at each time instant, the power applied to the pan can be estimated by subtracting the power estimated in the thermal model from the power absorbed by the food. In this way, it is possible to calculate P_{app} and T_{pan} and to predict the full cooking of the pancake with the NN in few milliseconds, without the need to use real sensors or communication with the induction cooktop.

As can be seen in Table 12, both modes obtain very good results calculating the mean L* of both sides at the final moment of cooking. Calculating the MAE for the 9 tests, it is obtained 3.30 L* and 2.15 L* for L_a and L_b respectively in the Offline mode and 2.22 L* and 2.30 L* for L_a and L_b respectively in the Online mode. For all tests, the error in L_a decreases in the Online mode with respect to the Offline mode, while the error in L_b in some cases increases and in others gets worse, obtaining a slightly higher average error.

Fig. 11 shows the prediction of L* for both sides of the pancake during cooking in the R5 test, using the Offline mode and the Online mode. In the plot of the lightness evolution, the L* value measured after processing the images obtained from each side of the pancake after cooking is shown with a dot. There is a difference in the temperature of the pan and the power applied between the two modes, despite running the same test. In the case of the Offline version, as P_{app} and T_{pan} are calculated, it can be seen how these values are in an ideal cooking case where there is no noise. In the Online mode, the induction hob estimation of the pan temperature as well as the value of the power being applied is used directly. The P_{app} and T_{pan} values are more noisy, but are closer to reality. The temperature of the pan can vary a lot by the actions of the user, for example, if the pan is not positioned correctly or if it is separated from the hob while cooking. It is always more accurate to measure P_{app} and T_{pan} and use these values as input. In the case of Fig. 11, the Online mode predicts more



(a) Test R5 with Offline mode (b) Test R5 with Online mode

FIGURE 11. Test R5 performed in the Offline mode (a) and in the Online mode (b). The inputs of each time instant (T_{pan} and P_{app}) and the output of the NN being monitored in this test (L_a and L_b) are shown. In the Online case, L_a and L_b end at a value closer to the one measured in the real experiment (M. L_a and M. L_b) than in the Offline case.

accurately the value of L*, since the final values of L_a and L_b are closer to the measured values. However, the error obtained in the Offline mode is not very large considering the noise of the inputs in the real cooking in this particular test.

VI. DISCUSSION

In this paper, we have shown the methodology to develop models for simulating food cooking in real time using a data-driven NARX-NN. Physics-based FEM models accurately simulate food cooking, but using these models in real time is unfeasible due to their high computational cost. This prevents them from simulating the complete cooking of the food in real time and from being able to adapt to external changes. A FEM model that simulates pancake cooking on induction hobs has been used to generate a database to train a NARX-NN. If the NARX-NN is trained with the standard algorithm used for NARX-NN training, i.e. training with TF, we obtain an NN that predicts with a mean R^2 of 0.78 for the six state parameters being estimated, in the 100 simulations used in test. Training the NARX-NN with the proposed two-phase training algorithm, we obtain a mean R^2 of more than 0.99 for the state parameters in the test simulations.

We have trained a data-driven NN model that almost replicate the accuracy of the original physics-based FEM model. The NN model that obtains the best metrics, with 3 hidden layers of 1024 neurons, has a speedup of $\approx 5.37 \times 10^2$ with respect to the original FEM model. NN with fewer neurons have also been presented, where for example an NN with 3 hidden layers of 128 neurons, still has an $R^2 > 0.99$ for all target variables in the test data, and has a speedup of $\approx 3.02 \times 10^4$ with respect to the original FEM model.

In addition, it has been shown that the models trained with our algorithm are not overfitted to the training data. For this, it has been shown that the predictions made by the model in situations where the input data are outside the ranges used during training are still accurate. Therefore, the trained NN has generalization capability and really captures the dynamics of the problem.

In simulating food cooking with NN, external inputs such as pan temperature are used. In previous works, these external variables are either measured directly with sensors, or their calculation is included in the NN. Measuring these variables in real time allows us to simulate the cooking of the food in real time but does not allow us to simulate the complete cooking in advance and make decisions based on it.

In our proposal, the NN also uses external inputs, but it has been decided that the NN does not estimate these parameters, since simulating the heating of the frying pan is a problem that can be solved with real-time analytical thermal models. This strategy allows the function of the NN to predict the cooking evolution of the food, avoiding learning the behavior of the pan. Otherwise, each of the different food models that would be developed with this technique would have to learn the behavior of the frying pans with which they are cooked, limiting the adaptability. Our approach has allowed the development of the Offline mode and the Online mode. The Offline mode uses the NARX-NN developed with the thermal model of the hob-pan system. This allows to simulate complete food cooking in a few milliseconds, with different frying pans, induction hobs, cooking times, etc. In the case of Online mode, the temperature measurements of the frying pan as well as the power applied to the pan are used directly. In this case, it would be executed in real time and would allow to monitor the cooking status, reacting in real time to the changes that may occur in the frying pan. Since they are models with low computational cost, the ideal in a real scenario is to use both: An initial prediction with the Offline mode, and updating the cooking state when the data is received in real time with the Online mode. Both approaches have been validated with real experiments.

VII. CONCLUSION

The main requirements presented in this paper for modeling a food process are to simulate the complete cooking of food, to perform in real time and to be robust to external changes. We may consider as solution to simulate food processing in real time with NN. The main limitation observed is that the NNs presented in the state-of-the-art use standard training algorithms that may not adapt correctly to the problem of simulating a dynamic system such as food cooking. The few details given of the NN architectures available also limit the application of the solution to other cooking processes.

The main contribution of this paper is a methodology for the implementation and training of NARX-NN that meets all the requirements presented. The case study chosen to validate the proposal is the cooking of pancakes on induction hobs. The architecture and training method used have been detailed, both optimized to make our NN computationally efficient and adapted to the dynamics of the problem. Our design yields the NN to have a negligible error compared to

the simulated data. In addition, the performance of the NN has been positively validated both with and without external sensors in experiments in real scenarios.

The methodology shown is highly versatile and can be adapted to different cooking processes. As future work, it would be interesting to test this methodology with different foods and cooking processes. We also note that the use of RNN with this methodology is promising for enabling real-time simulation of dynamical systems from other domains.

REFERENCES

[1] L. Serazetdinova, J. Garratt, A. Baylis, S. Stergiadis, M. Collison, and S. Davis, "How should we turn data into decisions in AgriFood?" *J. Sci. Food Agricult.*, vol. 99, no. 7, pp. 3213–3219, May 2019.

[2] M. J. Kaur, V. P. Mishra, and P. Maheshwari, "The convergence of digital twin, IoT, and machine learning: Transforming data into action," in *Digital Twin Technologies and Smart Cities*. Cham, Switzerland: Springer, 2020, pp. 3–17.

[3] C. Krupitzer, T. Noack, and C. Borsum, "Digital food twins combining data science and food science: System model, applications, and challenges," *Processes*, vol. 10, no. 9, p. 1781, Sep. 2022.

[4] S. Zolfaghari, T. Stoev, and K. Yordanova, "Enhancing kitchen activity recognition: A benchmark study of the rostock KTA dataset," *IEEE Access*, vol. 12, pp. 14364–14384, 2024.

[5] P. Verboven, T. Defraeye, A. K. Datta, and B. Nicolai, "Digital twins of food process operations: The next step for food process models?" *Current Opinion Food Sci.*, vol. 35, pp. 79–87, Oct. 2020.

[6] Y. Lin, J. Ma, Q. Wang, and D.-W. Sun, "Applications of machine learning techniques for enhancing nondestructive food quality and safety detection," *Crit. Rev. Food Sci. Nutrition*, vol. 63, no. 12, pp. 1649–1669, May 2023.

[7] S. Elbassuoni, H. Ghattas, J. E. Ati, Z. Shmayssani, S. Katerji, Y. Zoughbi, A. Semaan, C. Akl, H. B. Gharbia, and S. Sassi, "DeepNOVA: A deep learning Nova classifier for food images," *IEEE Access*, vol. 10, pp. 128523–128535, 2022.

[8] L. Guo, T. Wang, Z. Wu, J. Wang, M. Wang, Z. Cui, S. Ji, J. Cai, C. Xu, and X. Chen, "Portable food-freshness prediction platform based on colorimetric barcode combinatorics and deep convolutional neural networks," *Adv. Mater.*, vol. 32, no. 45, p. 2004805, 2020.

[9] L. Zhu, P. Spachos, E. Pensini, and K. N. Plataniotis, "Deep learning and machine vision for food processing: A survey," *Current Res. Food Sci.*, vol. 4, pp. 233–249, Jan. 2021.

[10] S. S. Alahmari and T. Salem, "Food state recognition using deep learning," *IEEE Access*, vol. 10, pp. 130048–130057, 2022.

[11] M. I. H. Khan, S. S. Sablani, R. Nayak, and Y. Gu, "Machine learning-based modeling in food processing applications: State of the art," *Comprehensive Rev. Food Sci. Food Saf.*, vol. 21, no. 2, pp. 1409–1438, Mar. 2022.

[12] G. V. S. B. Raj and K. K. Dash, "Comprehensive study on applications of artificial neural network in food process modeling," *Crit. Rev. Food Sci. Nutrition*, vol. 62, no. 10, pp. 2756–2783, Apr. 2022.

[13] K. C. Neethu, A. K. Sharma, H. A. Pushpadass, F. M. E. Emerald, and M. Manjunatha, "Prediction of convective heat transfer coefficient during deep-fat frying of pantoa using neurocomputing approaches," *Innov. Food Sci. Emerg. Technol.*, vol. 34, pp. 275–284, Apr. 2016.

[14] S. Jeong, J. Kwak, and S. Lee, "Machine learning workflow for the oil uptake prediction of rice flour in a batter-coated fried system," *Innov. Food Sci. Emerg. Technol.*, vol. 74, Dec. 2021, Art. no. 102796.

[15] M. Kannapinn, M. K. Pham, and M. Schäfer, "Physics-based digital twins for autonomous thermal food processing: Efficient, non-intrusive reduced-order modeling," *Innov. Food Sci. Emerg. Technol.*, vol. 81, Oct. 2022, Art. no. 103143.

[16] I. Cabeza-Gil, I. Ríos-Ruiz, M. Á. Martínez, B. Calvo, and J. Grasa, "Digital twins for monitoring and predicting the cooking of food products: A case study for a French crêpe," *J. Food Eng.*, vol. 359, Dec. 2023, Art. no. 111697.

[17] H. Hewamalage, C. Bergmeir, and K. Bandara, "Recurrent neural networks for time series forecasting: Current status and future directions," *Int. J. Forecasting*, vol. 37, no. 1, pp. 388–427, Jan. 2021.

[18] C. Legaard, T. Schranz, G. Schweiger, J. Drgoňa, B. Falay, C. Gomes, A. Iosifidis, M. Abkar, and P. Larsen, "Constructing neural network based models for simulating dynamical systems," *ACM Comput. Surv.*, vol. 55, no. 11, pp. 1–34, Nov. 2023.

[19] I. Hammoud, S. Hentzelt, T. Oehlschlaegel, and R. Kennel, "Learning-based model predictive current control for synchronous machines: An LSTM approach," *Eur. J. Control*, vol. 68, Nov. 2022, Art. no. 100663.

[20] T. V. Baby, S. M. Sotoudeh, and B. HomChaudhuri, "Data-driven prediction and predictive control methods for eco-driving in production vehicles," *IFAC-PapersOnLine*, vol. 55, no. 37, pp. 633–638, 2022.

[21] I. Lenz, R. Knepper, and A. Saxena, "DeepMPC: Learning deep latent features for model predictive control," in *Proc. Robot., Sci. Syst. (RSS)*, Rome, Italy, 2015.

[22] N. Mohajerin and S. L. Waslander, "Multistep prediction of dynamic systems with recurrent neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 11, pp. 3370–3383, Nov. 2019.

[23] S. Looper and S. L. Waslander, "Temporal convolutions for multi-step quadrotor motion prediction," in *Proc. 19th Conf. Robots Vis. (CRV)*. Toronto, ON, Canada: IEEE, May 2022, pp. 32–39.

[24] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, "Learning long-term dependencies in NARX recurrent neural networks," *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1329–1338, Nov. 1996.

[25] H. T. Siegelmann, B. G. Horne, and C. L. Giles, "Computational capabilities of recurrent NARX neural networks," *IEEE Trans. Syst., Man, Cybern., B*, vol. 27, no. 2, pp. 208–215, Apr. 1997.

[26] Y. Zhao, C. Jiang, M. A. Vega, M. D. Todd, and Z. Hu, "Surrogate modeling of nonlinear dynamic systems: A comparative study," *J. Comput. Inf. Sci. Eng.*, vol. 23, no. 1, Feb. 2023, Art. no. 011001.

[27] Y. Li, W. Chen, H. Yang, J. Li, and N. Zheng, "Joint torque closed-loop estimation using NARX neural network based on sEMG signals," *IEEE Access*, vol. 8, pp. 213636–213646, 2020.

[28] M. Wei, M. Ye, J. B. Li, Q. Wang, and X. Xu, "State of charge estimation of lithium-ion batteries using LSTM and NARX neural networks," *IEEE Access*, vol. 8, pp. 189236–189245, 2020.

[29] F. Di Nunno and F. Granata, "Groundwater level prediction in apulia region (Southern Italy) using NARX neural network," *Environ. Res.*, vol. 190, Nov. 2020, Art. no. 110062.

[30] M. Steinacker, Y. Kheifetz, and M. Scholz, "Individual modelling of haematotoxicity with NARX neural networks: A knowledge transfer approach," *Heliyon*, vol. 9, no. 7, Jul. 2023, Art. no. e17890.

[31] A. Zilio, D. Biadene, T. Caldognetto, and P. Mattavelli, "Black-box large-signal average modeling of DC-DC converters using NARX-ANNs," *IEEE Access*, vol. 11, pp. 43257–43266, 2023.

[32] H. V. H. Ayala and L. D. S. Coelho, "Cascaded evolutionary algorithm for nonlinear system identification based on correlation functions and radial basis functions neural networks," *Mech. Syst. Signal Process.*, vols. 68–69, pp. 378–393, Feb. 2016.

[33] H. V. H. Ayala, D. Habineza, M. Rakotondrabe, and L. D. S. Coelho, "Nonlinear black-box system identification through coevolutionary algorithms and radial basis function artificial neural networks," *Appl. Soft Comput.*, vol. 87, Feb. 2020, Art. no. 105990.

[34] J. Kelley and M. T. Hagan, "Comparison of neural network NARX and NARMAX models for multi-step prediction using simulated and experimental data," *Expert Syst. Appl.*, vol. 237, p. 121437, Mar. 2024.

[35] C. Franco, J. Acero, R. Alonso, C. Sagues, and D. Paesa, "Inductive sensor for temperature measurement in induction heating applications," *IEEE Sensors J.*, vol. 12, no. 5, pp. 996–1003, May 2012.

[36] C. Franco, D. Paesa, C. Sagues, and S. Llorente, "Analytical modeling of a saucepan in an induction hob," in *Proc. 18th Medit. Conf. Control Autom., MED*, Jun. 2010, pp. 298–303.

[37] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Comput.*, vol. 1, no. 2, pp. 270–280, Jun. 1989.

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

[39] M. Sangiorgio and F. Dercole, "Robustness of LSTM neural networks for multi-step forecasting of chaotic time series," *Chaos, Solitons Fractals*, vol. 139, Oct. 2020, Art. no. 110045.

[40] L. Prechelt, "Early stopping—But when?" in *Neural Networks: Tricks of the Trade* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 1998, pp. 55–69.

[41] F. Chollet et al. (2015). *Keras*. [Online]. Available: https://keras.io/getting_started/faq/#how-should-i-cite-keras
[42] M. Abadi et al., "TensorFlow: A system for large-scale machine learning," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement.*, 2016, pp. 265–283.
[43] S. Lorente-Bailo, I. Etayo, M. L. Salvador, A. Ferrer-Mairal, M. A. Martínez, B. Calvo, and J. Grasa, "Modeling domestic pancake cooking incorporating the rheological properties of the batter. Application to seven batter recipes," *J. Food Eng.*, vol. 291, Feb. 2021, Art. no. 110261.
[44] F. Sanz-Serrano, C. Sagues, A. H. Feyissa, J. Adler-Nissen, and S. Llorente, "Modeling of pancake frying with non-uniform heating source applied to domestic cookers," *J. Food Eng.*, vol. 195, pp. 114–127, Feb. 2017.
[45] D. Paesa, C. Franco, S. Llorente, G. Lopez-Nicolas, and C. Sagues, "Adaptive simmering control for domestic induction cookers," *IEEE Trans. Ind. Appl.*, vol. 47, no. 5, pp. 2257–2267, Sep. 2011.
[46] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

JAVIER FAÑANÁS-ANAYA received the degree and M.Sc. degrees in computer science engineering from the University of Zaragoza, Zaragoza, Spain, in 2020 and 2022, respectively. Since 2022, he has been with the Department of Computer Science and Systems Engineering and the Aragon Institute of Engineering Research (I3A), University of Zaragoza. His research interests include dynamical systems modeling, artificial intelligence, computer vision, and robotics.

GONZALO LÓPEZ-NICOLÁS (Senior Member, IEEE) received the Ph.D. degree in systems engineering and computer science from the University of Zaragoza, Zaragoza, Spain, in 2008. He is currently a Full Professor with the Department of Computer Science and Systems Engineering, University of Zaragoza. He is a member with the Robotics, Perception, and Real-Time Group, and the Aragon Institute of Engineering Research (I3A). His current research interests include visual control, autonomous robot navigation, multirobot systems, and the application of computer vision techniques to robotics.

CARLOS SAGÜÉS (Senior Member, IEEE) received the M.Sc. degree in computer science and systems engineering and the Ph.D. degree in industrial engineering from the University of Zaragoza, Zaragoza, Spain, in 1989 and 1992, respectively. In 1994, he joined as an Associate Professor with the Departamento de Informatica e Ingenieria de Sistemas, University of Zaragoza, where he became a Full Professor, in 2009, and also the Head Teacher. He was engaged in research on force and infrared sensors for robots. His current research interests include control systems and industry applications, computer vision, visual control, and multivehicle cooperative control.

SERGIO LLORENTE received the M.Sc. and Ph.D. degrees in electronic engineering from the University of Zaragoza, Zaragoza, Spain, in 2001 and 2016, respectively. In 2001, he joined Bosch-Siemens Home Appliances Group, Zaragoza, where he held different positions with the Research and Development Department of Induction Cooktops. He has been an Assistant Professor with the University of Zaragoza, since 2004. He is currently the in charge of several research lines and projects and is also an inventor with more than 200 patents. His research interests include power electronics, simulation and control algorithms for power electronics, and temperature.