DOI: https://doi.org/10.15276/hait.09.2026.20
UDC 004.942

Long Short-Term Memory based reference model method to nonlinear dynamic system identification

Oleksandr O. Fomin[1)]
ORCID: https://orcid.org/0000-0002-8816-0652; fomin@op.edu.ua. Scopus Author ID: 57103429400
Viktor O. Speranskyy[1)]
ORCID: https://orcid.org/0000-0002-8042-1790; speranskyy@op.edu.ua. Scopus Author ID: 54401618900
Andriy A. Verlan[2)]
ORCID: https://orcid.org/0000-0002-6469-2638; a.verlan@edu.kpi.ua. Scopus Author ID: 57192176307
Oleksiy V. Tataryn[1)]
ORCID: https://orcid.org/0009-0005-3888-6569; otataryn@stud.op.edu.ua. Scopus Author ID: 59390593400
Andriy M. Chmelevskyi[1)]
ORCID: https://orcid.org/0009-0008-6450-6875; stech@stud.op.edu.ua
[1)] Odesa Polytechnic National University, 1 Shevchenko Ave. Odesa, 65044, Ukraine
[2)] Norwegian University of Science and Technology, Torgarden, NO-7491 Trondheim, Norway

ABSTRACT

The relevance of this research stems from the fact that modern engineering problems and intelligent control systems require highly efficient mathematical tools for modeling complex nonlinear dynamic systems, capable of resolving the fundamental trade-off between ensuring approximation accuracy and minimizing computational costs and model training time. Traditional approaches based on neural networks with time delays are limited by a fixed memory window size, which makes it impossible to adequately describe processes with deep delays and non-stationary modes without a significant increase in the number of parameters. **The aim of this work** is to reduce the time required to construct models of nonlinear dynamics while ensuring a specified modeling accuracy by developing the reference model method through the use of neural network architecture with long-term short-term memory. To achieve this goal, **the following tasks are** set: to carry out a theoretical development of the reference model method through a transition to dynamic recurrent structures, to develop an algorithm for parameter alignment of reference models with long short-term memory to overcome permutation invariance, and to perform experimental verification of the proposed approach. **The research methods** are based on system identification theory, transfer learning methodologies, and model fusion. To eliminate the permutation symmetry of hidden layers, a weight tuning optimization algorithm based on solving linear programming problems has been adapted, which allows independent parameter matrices to be transferred to a common loss function pool before their superposition. **The results of the work** include the formalization of a reference model method based on long short-term memory and an algorithm for parameter alignment of these models, which ensures linear connectivity of modes in parameter space. Experimental verification on a test nonlinear system demonstrated that the proposed method accelerates model convergence by nearly a factor of 2 compared to an approach based on neural networks with time delays and reduces the final error by a factor of 3.3. **The obtained results** confirm the effectiveness of using recurrent architectures in the model superposition process. The scientific novelty of the work lies in the development of a reference model method based on long short-term memory, which allows overcoming the limitations inherent in architectures with time delays associated with the finiteness of the memory window. This ensures adequate modeling of systems characterized by deep delay, significant inertia, and complex nonlinear hysteresis effects. Furthermore, to address the problem of permutation invariance of hidden states in independently trained recurrent networks – which traditionally precludes direct averaging of model weight parameters – this work adapts a specialized weight-tuning optimization algorithm to the structure of long short-term memory.

Keywords: Modeling; nonlinear dynamics; neural networks; long short-term memory; model ensembling; transfer learning, model merging, neuron alignment, weight matching, linear mode connectivity

***For citation*: Fomin O. O., Speranskyy V. O., Verlan A. A., Tataryn O. V., Chmelevskyi A. M. "Long Short-Term Memory based reference model method to nonlinear dynamic system identification". *Herald of Advanced Information Technology*. 2026; Vol. 9 No. 3: 307–320. DOI: https://doi.org/10.15276/hait.09.2026.20**

1. INTRODUCTION

The identification of nonlinear dynamic systems remains one of the central problems in control theory, robotics, and systems analysis. This is driven by significantly increased demands on control systems, caused by the complexity of modern systems characterized by significant nonlinearity and complex dynamics [1]. Such systems exhibit complex behavior, including bifurcations and dependence on initial conditions, rendering their modeling an exceptionally challenging task.

© Fomin O., Speranskyy V., Verlan A., Tataryn O., Chmelevskyi A., 2026

This is an open access article under the CC BY license (https://creativecommons.org/licenses/by/4.0/deed.uk)

Conventional system identification methodologies based on parametric models (e.g., Hammerstein–Wiener models, polynomial models) frequently fail to provide adequate descriptions in the absence of deep prior knowledge regarding the system's underlying structure. Conversely, nonparametric models (e.g., Volterra and Volterra–Wiener series) are primarily designed for systems with a low degree of nonlinearity (typically up to the 2nd or 3rd order). They often struggle with complex nonlinearities – such as hysteresis, saturation, dead zones, or discontinuities – requiring an excessively large number of series terms to describe them adequately. Consequently, the number of parameters (Volterra kernels) that must be estimated grows exponentially with the increasing order of nonlinearity and system memory length. This results in prohibitive computational complexity, a phenomenon widely known as the *curse of dimensionality*. Furthermore, the process of estimating high-order kernels becomes computationally unstable and complex in the presence of measurement noise.

Over the past decade, the field of system identification has undergone a radical transformation driven by the shift from classical approaches to machine learning methods, particularly artificial neural networks (ANNs) [1], [2]. These models serve as universal *black-box* and *gray-box* approximators. They are capable of learning directly from observed input-output data and facilitate the modeling of systems with complex nonlinearities and dynamics that are poorly captured by traditional models [3].

However, the application of ANNs in the identification of complex nonlinear dynamic systems involves two specific challenges: accuracy (modeling dynamic systems requires specialized ANN architectures) and computational costs (ensuring convergence during the training phase requires stochastic gradient descent algorithms or their derivatives). In applications demanding rapid model adaptation (e.g., equipment parameter drift due to wear) or under limited computational resources (edge computing), a prolonged training process utilizing random initial conditions presents a critical drawback [4].

2. LITERATURE REVIEW AND PROBLEM STATEMENT

To address the challenge of ensuring the adequacy of neural network models and improving the accuracy of nonlinear dynamics identification, several ANN architectures are currently employed.

Time-Delay Neural Networks (TDNNs) represent one of the earliest successful attempts to adapt feedforward networks for time-series processing [4], [5]. Essentially, a TDNN simulates dynamic memory through an input delay line.

The input vector $\mathbf{x}(t_n)$ at time step t_n is constructed by concatenating the current signal value $x(t_n)$ and the M-1 previous values, where M denotes the memory window size:

$$\mathbf{x}(t_n)=[x(t_n), x(t_{n-1}), \ldots, x(t_{n-M+1})]. \quad (1)$$

This vector is fed into the input of a fully connected feed forward network, thereby projecting the temporal sequence into the spatial domain: the dynamics are represented as a static pattern of fixed length M.

The output of a TDNN with a hidden layer of size K and an input vector of dimension M is described as:

$$y(t_n) = S_0\left[b_0 + \sum_{i=1}^{K} w_i S_i\left(b_i + \sum_{j=1}^{M} w_{i,j} x(t_{n-j}) \right)\right], \quad (2)$$

where b_0, b_i are biases, S_0, S_i are activation functions, and w_i, $w_{i,j}$ are the weights of the neurons in the output and hidden layers, respectively.

Since the TDNN lacks feedback connections, it is trained using the standard backpropagation algorithm, which is computationally more efficient and stable than the backpropagation through time (BPTT) algorithm required for RNNs. The absence of recursion ensures that the network remains stable in the bounded-input bounded-output (BIBO) sense when utilizing bounded activation functions – a property critically important for control tasks [6].

Despite these advantages, TDNNs possess fundamental limitations that have led to their gradual displacement by recurrent architectures [7].

1. The memory of a TDNN is strictly constrained by the window size M. If a temporal dependency in the data extends M+1 steps back, the network is fundamentally incapable of capturing it. Conversely, increasing M leads to a linear increase in the number of parameters in the first layer, exacerbating the risk of over fitting. Many physical systems (e.g., thermal processes, hydraulic systems, or chemical reactions) exhibit significant inertia, where the current state depends on inputs applied thousands of steps previously. TDNN architectures struggle to process such extensive historical information.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

2. TDNNs do not model the internal state of the system; they merely map an input history to an output. This characteristic makes them unsuitable for modeling autonomous systems or systems where the output depends not only on external inputs but also on unmeasurable hidden internal processes (e.g., battery state-of-charge or cumulative errors in a control system).

3. Real-world data may arrive at irregular intervals or contain missing values, presenting a significant challenge for discrete architectures that rely on a fixed sampling step.

Recurrent Neural Networks (RNNs) introduced a paradigm shift: transitioning from modeling "memory as space" in TDNNs to modeling "memory as state" [7, 8]. In these architectures, a neuron's output is fed back to its input with a single-step delay, creating a cyclical feedback loop. The hidden state of the RNN functions as memory, aggregating information across the entire history of inputs $x(t_0)$, $x(t_1)$, …, $x(t_n)$.

An RNN model with a hidden layer of size K and an input vector of dimension M for time step $x(t_n)$ is formulated as:

$$\begin{cases} y(t_n) = S_0\left[b_0 + \mathbf{V}\mathbf{h}(t_n)\right] \\ \mathbf{h}(t_n) = S_1\left[\mathbf{B}_1 + \mathbf{W}x(t_n) + \mathbf{U}\mathbf{h}(t_{n-1})\right] \end{cases}, \quad (3)$$

where b_0, $\mathbf{B}_1$ represent the biases of the output and hidden layers; S_0, S_1 denote the activation functions of the output and hidden layers; $\mathbf{W}\in R^{K\times M}$, $\mathbf{V}\in R^K$ are the weight matrices of the hidden and output layers; $\mathbf{U}\in R^{K\times K}$ denotes the recurrent weight matrix; and $\mathbf{h}(t_n)$ represents hidden state vector.

Although standard RNNs can theoretically model dependencies of arbitrary length, in practice, they suffer from the vanishing gradient problem [9, 10]. When trained on long temporal sequences, the network tends to "forget" long-term dependencies, thereby restricting the application of standard RNNs in complex nonlinear dynamics problems.

The *Long Short-Term Memory* (LSTM) architecture was specifically designed to mitigate the vanishing gradient problem and has recently become the dominant approach for identifying dynamic systems. The core innovation of LSTM is the introduction of a cell state C_t, acting as a "conveyor belt" that transfers information across time steps with minimal modification.

Information flow is regulated via three gating mechanisms [9], [10], [11]:

– the forget gate f_t dictates which information to discard from the cell;

– the input gate i_t determines which new information to write;

– the output gate O_t controls which portion of the cell state is emitted as the hidden state h_t.

An LSTM model with a hidden layer of size K and an input vector of dimension M is described by the following matrices:

– $\mathbf{W}\in R^{4K\times M}$ for input weights (across the 4 gating layers);

– $\mathbf{U}\in R^{4K\times K}$ for recurrent weights;

– $\mathbf{B}_1\in R^{4K}$ for bias vectors.

At time step t_n the gating variables (i_t, f_t, o_t, g_t) are computed as:

$$\begin{pmatrix} i_t \\ f_t \\ o_t \\ g_t \end{pmatrix} = \begin{pmatrix} S_0 \\ S_0 \\ S_0 \\ S_1 \end{pmatrix} [\boldsymbol{B}_1 + \boldsymbol{W}x(t_n) + \boldsymbol{U}\boldsymbol{h}(t_{n-1})] \quad (4)$$

$$\boldsymbol{h}(t_n) = o_t \odot S_1(C_t).$$

The state of the memory cell C_t is described as:

$$C_t = f_t \odot C_{t-1} + i_t \odot g_t, \quad (5)$$

where $\odot$ denotes element-wise multiplication.

The *Gated Recurrent Unit* (GRU) architecture is a streamlined variant of LSTM that merges the memory cell and hidden state, and combines the input and forget gates into a single *update gate* [8], [13]. Studies [14] indicate that GRUs often achieve accuracy comparable to LSTMs while utilizing fewer parameters, resulting in faster convergence, particularly on smaller datasets. However, for tasks requiring the precise modeling of highly complex, long-term dependencies, LSTMs generally maintain an accuracy advantage due to the isolated cell state C_t.

Both LSTM and GRU architectures overcome the aforementioned TDNN drawbacks, facilitating the training of deep models on prolonged sequences where the time scales of underlying physical processes may differ by orders of magnitude [9], [10].

Transformer Architectures for Time Series

Based on the self-attention mechanism, the Transformer architecture has been actively adopted for system identification since 2020. Unlike RNNs, which process data sequentially, Transformers evaluate the entire time sequence in parallel, directly capturing dependencies between distant temporal points.

Research [15], [16] demonstrates that Transformers outperform LSTMs in tasks with extensively delayed responses and also excel over short time horizons; however, they require substantially more data and computational resources. The standard Vanilla Attention mechanism exhibits quadratic computational complexity $O(N^2)$ relative to the sequence length N, restricting its application to extraordinarily long memory windows [16]. To mitigate this, several state-of-the-art architectures have emerged. Notably, the Informer model [17] employs a ProbSparse Self-Attention mechanism, reducing computational complexity and memory footprint to $O(N \ln N)$. Nevertheless, despite the high accuracy of attention-based architectures in capturing long-term trends, their deployment in real-time embedded control systems is often bottlenecked by the limited hardware resources of microcontrollers. Consequently, finding a compromise between the precision of Transformer-type models and the computational efficiency of recurrent models remains a highly relevant research direction [17].

To address the computational and temporal overhead associated with the training phase of neural network modeling, techniques such as network pruning, quantization, and model compression are widely utilized [18], [19]. Another prominent strategy to accelerate training involves advanced optimization algorithms, such as stochastic gradient descent with momentum and adaptive learning rate methods [20].

A significant trend in recent years is the development of *model ensembling* and *model merging* methodologies, which advocate leveraging multiple models trained with varying hyperparameters rather than selecting a single optimal model [21]. Within the domain of nonlinear dynamics modeling, a *reference model method* has been proposed. This method aims to accelerate the synthesis of a target model by superimposing a set of pre-trained reference ANNs [18]. However, because the conventional implementation of this method relies on TDNNs, it inherits a fundamental architectural constraint: the system's memory is rigidly bound by the input delay window. Modeling processes governed by large time constants or slowly decaying transients necessitates an impractically large delay window. This leads to a linear escalation in the parameter count and severely complicates the optimization process due to the curse of dimensionality.

In summary, an analysis of the literature over the past decade reveals a definitive evolutionary trajectory in nonlinear dynamics identification. The transition from TDNNs to RNNs is motivated by the necessity to model systems featuring complex, long-term temporal dependencies. The prevailing dominance of LSTM models stems from an effective architectural design (gating mechanisms and cell states) that successfully resolves the vanishing gradient problem and enables the training of deep networks on extended time series.

Current global trends [21], [22] suggest that the future of identifying complex nonlinear dynamic systems lies not only in designing novel ML architectures but also in establishing more efficient frameworks for reusing and integrating existing knowledge, such as pre-trained models and model merging techniques.

This paper extends the pre-trained reference model method [18] by transitioning from TDNNs to dynamic recurrent networks (specifically the LSTM architecture), which possess the inherent capability to model systems across varying temporal scales.

The formal statement of the pre-training problem is defined as follows.

Let S denote a broad domain encompassing general-purpose tasks, for which a comprehensively labeled dataset D_S of sufficient size N_S is available:

$$D_S = \{(\mathbf{x}_i^S, y_i^S)\}, \quad (6)$$

where $\mathbf{x}_i^S$ is a vector of independent variables, y_i^S is the corresponding target variable (label), and $i=1,\ldots, N_S$.

Let $f_{\theta S}$ (D_S) represent a general (*coarse*) model parameterized by θ_S, which has been pre-trained on the dataset D_S.

Let T_k represent a specific target task from the set of target tasks **T** defined within the domain S ($k=1, \ldots, p$, where p is the cardinality of **T**) . For this task, a labeled dataset D_{Tk} of limited size N_{Tk} is available:

$$D_{Tk} = \{(\mathbf{x}_j^{Tk}, y_j^{Tk})\}, \quad (7)$$

where $\mathbf{x}_j^{Tk}$ is a vector of independent variables, y_j^{Tk} is the corresponding target variable, and $j=1,\ldots, N_{Tk}$.

Let $f_{\theta Tk}(\theta_S, D_{Tk})$ denote the *target* (fine-tuned) model for task T_k with parameters θ_{Tk}, obtained by fine-tuning the coarse model $f_{\theta S}$ on the dataset D_{Tk}.

The objective of constructing a target model based on a pre-trained general model is to identify a set of parameters θ_S for the general model $f_{\theta S}$ (D_S) such that, when utilized as initial weights $\theta_{Tk0}=\theta_S$ for fine-tuning each of the p target models $f_{\theta Tk}$ (θ_S, D_{Tk}), a specified accuracy threshold (acceptable error $E_{\theta T}$) is achieved in the minimum average time:

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

$$\begin{cases} \theta_S = \underset{\theta}{arg\ min}\ t_{\theta T}(f_{\theta Tk}(\theta, \mathbf{x}_j^{Tk})) \\ L_T(f_{\theta Tk}(\theta_S, \mathbf{x}_j^{Tk}), y_j^{Tk}) < E_{\theta T} \end{cases}, \quad (8)$$

where $t_{\theta T}$ denotes the average fine-tuning duration across the p target models $f_{\theta Tk}$ (θ_S, D_{Tk}) (typically measured in training epochs); the average number of model training epochs can be used as the value of $t_{\theta T}$; and L_T is the designated loss function for the set of target models $f_{\theta Tk}$ (θ_S, D_{Tk}).

When conditions (8) are satisfied, the model $f_{\theta S}(D_S)$ is considered *pre-trained*. Standard evaluation metrics, such as Mean Absolute Error (MAE) and Mean Squared Error (MSE), are typically employed as the loss function L_T:

$$mae = \frac{1}{N_{Tk}} \sum_{j=1}^{N_{Tk}} |\Delta f_{\theta Tk}|, \quad (9)$$

$$mse = \frac{1}{N_{Tk}} \sum_{j=1}^{N_{Tk}} (\Delta f_{\theta Tk})^2. \quad (10)$$

where $\Delta f_{\theta Tk} = y_j^{Tk} - f_{\theta Tk}(\theta_S, \mathbf{x}_j^{Tk})$ is the difference between the true value of the target variable and the value predicted by the model.

3. RESEARCH AIM AND OBJECTIVES

This study *aims* to reduce the time required to synthesize nonlinear dynamic models while ensuring a specified modeling accuracy. This is achieved by extending the reference model method through the integration of the LSTM architecture.

To fulfill the aim, the following specific objectives were established.

1. Develop a modification of the reference model method tailored for LSTM recurrent neural networks, explicitly accounting for their internal structural features (i.e., forget, input, and output gating mechanisms).

2. Propose a reference LSTM model alignment algorithm based on optimal neuron permutation to eliminate permutation ambiguity in the parameter space.

3. Conduct an experimental comparative analysis evaluating the proposed method against the standard TDNN-based reference model method and a randomly initialized target model, utilizing a nonlinear system identification benchmark.

4. Evaluate the impact of reference model alignment on the quality of initialization and the convergence rate of the target model.

4. MATERIALS AND METHODS

4.1. Issue of over-parameterization in pre-trained models

A primary factor degrading the efficiency of standard pre-training paradigms is the overly generalized nature and vast volume of the initial training dataset D_S [4]. This issue arises when attempting to encapsulate the system's behavior across an excessively broad spectrum of external conditions, operational modes, and input signal variations. Consequently, when addressing specific target tasks that span only a narrow sub-region of the domain, both the coarse model $f_{\theta S}$ (D_S) and the target models $f_{\theta Tk}$ (θ_S, D_{Tk}) become unnecessarily over-parameterized. This complexity impedes the fine-tuning process and diminishes the target model's operational efficiency.

To circumvent this over-complexity and accelerate target model training, the *reference model method* is utilized [18].

4.2. Overview of the Reference model method

The methodology relies on utilizing g reference datasets, each characterizing a discrete fundamental property of the investigated domain. A conceptual diagram illustrating the preparation process of reference datasets is presented in Fig. 1.

Based on these datasets, reference pre-trained models $f_{\theta Sv}$ (D_{Sv}) with parameters $\theta_{(Sv)}$ (v=1, …, g) are constructed. By combining the corresponding reference models, a coarse model $f_{\theta Tk}$ (θ_S, D_{Tk}) is constructed, which possesses certain characteristics (nonlinear and dynamic) of the object of study. This allows us to obtain an initialization of the weights for the target model that is already close to the global minimum of the loss function L_T. The target model $f_{\theta Tk}(\theta_S, D_{Tk})$ of the system is constructed by fine-tuning the coarse model on the dataset D_{Tk}.

Employing pre-trained reference models – which encapsulate the fundamental physics of the system – to initialize the target model generates a "warm start" effect, thereby eliminating protracted training phases. Unlike random initialization, which commences the optimization process from an arbitrary coordinate on the loss landscape, parameter averaging consolidates the generalized knowledge of multiple base models without corrupting their intrinsic topological structure.

This approach aligns with current industry trajectories favoring transfer learning and model merging methodologies over standard random

initialization paradigms [21, 22]. In this study, the reference model method is mathematically generalized to accommodate reference models implemented as LSTMs.

4.3. Algorithm of the reference LSTM model method

A block diagram of the target model based on reference LSTM models is shown in Fig. 1.

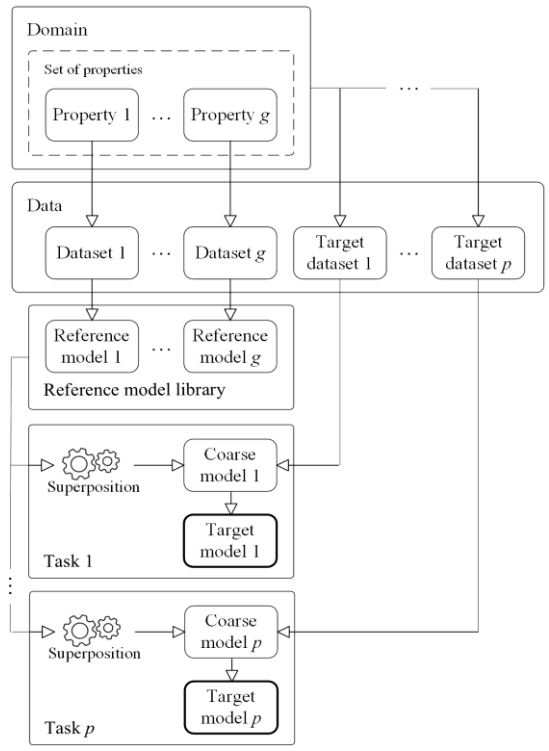


Fig. 1. **Block diagram of the target model based on reference LSTM models**
Source: **compiled by the authors**

The structural pipeline for synthesizing a target model utilizing reference LSTM models involves the following sequential steps:

Step 1. Domain S decomposition and reference dataset D_{Sv} formation.

The class of systems under investigation is systematically analyzed to extract foundational properties. Emphasis is placed on extracting:

– linear dynamic components (1st and 2nd order transfer functions, oscillatory and aperiodic elements);

– nonlinear static characteristics (saturation constraints, dead zones, hysteresis loops, piecewise-linear, and smooth continuous nonlinearities).

Subsequently, a designated set of reference datasets D_{Sv} is generated to isolate and reflect these properties.

Step 2. Training the reference model library. The architectural topology of the reference models $f_{\theta Sv}(D_{Sv})$ is strictly defined as an LSTM network with fixed structural dimensions θ_{Sv}. Absolute structural uniformity across all reference models is mandatory to enable subsequent neuron alignment and valid weight superposition.

Preliminary training is executed on the isolated datasets D_{Sv} employing the standard BPTT algorithm. Training iterates until the predefined convergence criterion (loss function L_T) is satisfied. This yields a comprehensive library of reference models parameterized by $\{\theta_1, \theta_2, \ldots, \theta_U\}$, where each specific model $f_{\theta Sv}(D_{Sv})$ represents a localized property within a narrow class of dynamics or nonlinearity.

Step 3. Reference model alignment. The neurons constituting the hidden states of the reference models undergo rigorous alignment using the algorithm detailed in Section 4.4.

Step 4. Coarse model synthesis. To synthesize the coarse model $f_{\theta S}(D_S)$, the specific required properties are selected from the global domain inventory. The coarse model is derived via the parameter-space superposition of the aligned reference models (generated in *Step 2*) corresponding to these identified traits. The resulting coarse model $f_{\theta S}(D_S)$ embeds a synthesized superposition of the dynamic and nonlinear characteristics localized within the aligned nodes.

Step 5. Fine-tuning on target data. The synthesized coarse model $f_{\theta S}(D_S)$ with weights θ_S acts as the initial state matrix for fine-tuning the target model $f_{\theta Tk}(\theta_S, D_{Tk})$ on the empirical target dataset D_{Tk}.

Given that $f_{\theta S}(D_S)$ inherently contains an advanced approximation of the system, a drastically reduced number of training epochs $t_{\theta T}$ is required to satisfy the accuracy threshold L_T. Furthermore, the coarse model explicitly mitigates the risk of converging to suboptimal local minima (endemic to random initialization) and strongly suppresses overfitting tendencies when operating on restricted datasets.

Step 6. Quality metric evaluation. Upon conclusion of the fine-tuning phase, the cumulative training time $t_{\theta T}$ and statistical accuracy metrics (Eqs. 10, 11) are evaluated. If the resultant target model's quality metrics fall below acceptable thresholds, the procedure reverts to Step 2 to reconfigure the structural hyperparameters $\theta_{(Sv)}$ of the reference models $f\theta_{Sv}$ (D_{Sv}), or, if structurally necessary, to Step 1 to re-evaluate the fundamental domain properties and associated datasets D_{Sv}.

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

4.4. Reference model alignment

Directly porting the TDNN model superposition methodology to the LSTM architecture is mathematically unfeasible. A profound theoretical barrier to integrating weight matrices across different neural networks is *permutation symmetry*. This phenomenon is formally characterized within the context of *linear mode connectivity*: two ANNs reside within the same linearly connected basin of the loss landscape if, and only if, they can be interpolated by a linear vector in weight space without incurring a severe degradation in performance (a high error barrier) [23, 24]. In standard practice, models trained via independent stochastic initializations converge into distinctly disjoint optimization basins [21, 25].

Unlike TDNNs, where neurons within hidden layers operate with relative combinatorial independence, the hidden states in an LSTM are strictly sequentially coupled via recurrent matrices. This induces *permutation invariance* (or permutation symmetry): the specific ordering of internal neurons within the hidden layer is arbitrary and uniquely dictated by the random initialization seed.

Simply averaging the parameters of k independent LSTM models with the same structure $\theta_{avg} = \frac{1}{g}\sum_{k=1}^{g}\theta_k$ will destroy the network's functionality, since neurons responsible for the same properties may be located in different positions (the matrices $\mathbf{U}_k$ depend on the order of elements in the vector $\mathbf{h}(t_{n-1})$ [10].

To resolve this ambiguity, *neuron alignment algorithms* are deployed. Applying such an algorithm to the weight matrices of a secondary model effectively "rotates" it within the parameter space, forcing its trajectory into the identical optimization basin occupied by the primary model [25]. Traditionally, these algorithms are highly computationally intensive due to the necessity of solving complex linear assignment problems to deduce the optimal weight permutations. However, within the proposed architectural framework, this computational overhead is negligible because the generation and alignment of the reference model library are executed entirely offline during the system design phase or scheduled software maintenance windows.

The alignment algorithm applies a permutation matrix $\mathbf{\Pi}$ (of dimension $K \times K$) to the hidden nodes of the model $f\theta_{Sv}(D_{Sv})$. The parameter matrices are algebraically transformed as follows:

– input weights: $\mathbf{W}'_k = \mathbf{\Pi}\mathbf{W}_k$ (row permutation);

– recurrent weights: $\mathbf{U}'_k = \mathbf{\Pi}\mathbf{U}_k\mathbf{\Pi}^T$ (simultaneous row and column permutation);

– bias vectors: $\mathbf{B}'_k = \mathbf{\Pi}\mathbf{B}_k$;

– output weights (if a fully connected readout layer exists): $\mathbf{V}'_k = \mathbf{V}_k\mathbf{\Pi}^T$.

The optimization objective is to compute a set of distinct permutations $\{\mathbf{\Pi}_1, \ldots, \mathbf{\Pi}_K\}$ that rigorously maximizes structural consistency (minimizes the L_2 distance between corresponding neuron weight vectors across the networks) prior to enacting parameter superposition.

This study adapts standard neuron alignment algorithms to specifically address the intricate gating mechanisms of LSTM recurrent layers. The alignment protocol executes an iterative methodology analogous to FedMA [25, 26], structured in four sequential stages:

Stage 1. Initialization. The reference model $f\theta_{S\mu}(D_{S\mu})$ (where $\mu \in [1, g]$) that exhibits the most severe dynamic complexity is designated as the *anchor model*, establishing the baseline topological structure of the recurrent matrices. A secondary model $f\theta_{Sv}(D_{Sv})$ (where $v \in [1, g]$ and $v \neq \mu$) is selected as the *local model* scheduled for alignment. Selecting the dynamically dominant model as the anchor relies on established theoretical patterns governing recurrent network behavior within state spaces and loss landscape geometry [27, 28].

For both networks, composite neuron feature matrices are constructed by concatenating the input $\mathbf{W}$ and recurrent $\mathbf{U}$ weight matrices. A specific weight vector w_j^k assigned to the j-th neuron of the k-th model integrates the specific rows from $\mathbf{W}$ and $\mathbf{U}$ governing the behavior of the internal cell state C_t alongside the forget f_t, input i_t and output O_t gates. The algorithm initializes at hidden layer index $l = 1$.

Stage 2. Permutation computation. For the active hidden layer l, the following procedures are executed:

– *Similarity Matrix Computation*: the Pearson correlation coefficient or L_2 Euclidean distance is computed across all conceivable pairs of neurons residing in layer l between the anchor and local models. It is imperative to note that computations evaluating layer l must analytically incorporate the permutations already applied to the inputs emanating from layer l–1.

– *Linear Assignment Resolution:* an optimal permutation matrix $\mathbf{\Pi}^l$ is computed to maximize inter-neuron similarity (minimizing the aggregate

distance between anchor and local neurons) via linear programming formulation [26]:

$$\min_{\{\mathbf{\Pi}_k\}_{k=1}^{K},\theta_{loc}} \sum_{k=1}^{K} \left\| \theta_{loc} - \mathbf{\Pi}_k \left(\theta_k\right) \right\|^2 , \quad (11)$$

where θ_{loc} denotes the anchor parameters; θ_k represents the parameters of the *k*-th local model; $\mathbf{\Pi}_k$ is the permutation operator applied to the k-th model; $\sum_{k=1}^{K} \|\dots\|^2$ minimizes the sum of squared Euclidean distances bridging the global anchor weights and the permuted local weights.

– *Neuron Alignment Execution:* the weights and biases of the current layer l within the local model are permuted: $\mathbf{W}_v^l \leftarrow \mathbf{\Pi}\mathbf{W}_v^l$. Consequently, the input channels propagating to the subsequent layer *l*+1 must be equivalently permuted to preserve mathematical continuity: $\mathbf{W}_v^{l+1} \leftarrow \mathbf{W}_v^{l+1}\left(\mathbf{\Pi}^l\right)^T$

Stage 3. *Iteration*. The hidden layer index increments ($l = l + 1$), and the algorithm loops back to Stage 2. Iteration proceeds sequentially until the final hidden layer is fully aligned.

Stage 4. Result output. The terminal state yields a local model $f\theta_{Sv}(D_{Sv})$ that operates within the identical optimization basin as the anchor model.

This procedure guarantees that structurally corresponding neurons – governing identical physical features – are rigorously mapped to adjacent topological coordinates across both models.

4.5. Superposition of Reference LSTM models

Once strict linear connectivity is established (all reference models co-reside within an identical loss basin), the models undergo parameter superposition. In the standard scenario, superposition is achieved by calculating the unweighted arithmetic mean of the parameter tensors. This methodology is theoretically valid because the geometric trajectory connecting parameter coordinates within a shared, aligned optimization basin is approximately linear [23, 25].

To enhance the precision of the synthesized model, specific weighting coefficients α_k can be assigned to the parameter tensors θ of each reference model, provided a priori domain knowledge exists concerning the dominance of specific physical phenomena within the target system.

Under these conditions, the superposition is computed as a weighted arithmetic mean:

$$\theta_{coarse} = \frac{1}{g}\sum_{k=1}^{g} \alpha_k \theta_k , \quad (12)$$

This synthesis ensures the rigorous integration of matched network parameters while strictly preserving the functional integrity of the encoded physical properties. The resultant output is the coarse model θ_{coarse}, encapsulating the hybridized physics of the reference library, deployed as the foundational initialization state for the fine-tuning of the target model.

5. RESEARCH RESULTS

5.1. Research system model

A suite of numerical simulations was executed to empirically validate the proposed reference LSTM model methodology. The classical Lur'e system (Fig. 2) was adopted as the standard benchmark system. The Lur'e system presents a complex identification challenge as it integrates a linear dynamic block operating within a feedback loop containing a highly nonlinear element, thereby precluding direct, decoupled measurement of its internal states.

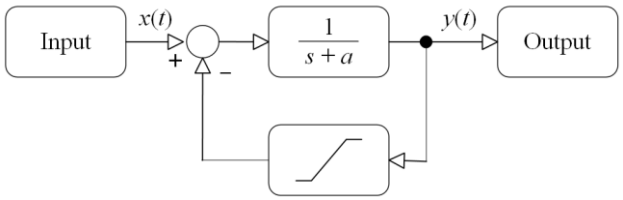


Fig. 2. **Block diagram of the test nonlinear dynamic system**
Source: **compiled by the authors**

The system fundamentally exhibits:

– a first-order linear dynamic element;

– a static nonlinearity (strict saturation threshold).

Isolated datasets were mathematically generated to characterize specific physical traits: D_{S1} strictly captures the isolated linear dynamic response (5,000 samples), rigorously isolates the nonlinear static behavior (5,000 samples). To represent the coupled nonlinear dynamic environment, an empirical target dataset D_{T1} was synthesized modeling the holistic Lur'e system response. This target dataset was purposefully restricted to 2,000 samples to deliberately simulate severe data-scarcity conditions.

5.2. Experimental Formulation

To benchmark the efficacy of the proposed methodology, the following distinct system models were instantiated and trained:

– TDNN-Ref: the standard reference model method utilizing TDNNs;

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

– LSTM-Random: the LSTM network initialized with completely stochastic parameters (Xavier initialization);

– GRU-Random: the GRU network initialized with completely stochastic parameters (Xavier initialization);

– LSTM-Ref-Naive: the proposed reference LSTM method implementing naïve superposition *without* prior neuron alignment;

– LSTM-Ref-Aligned: the proposed reference LSTM method utilizing mathematically rigorous neuron alignment.

To ensure rigorous baseline parity, TDNN, LSTM, and GRU architectures of precisely equivalent parametric complexity were selected. This parity was validated by plotting the parameter count as a function of the memory context window (defining the LSTM parameter count as the 100% baseline), shown in Fig. 3.

The specific architectural topologies utilized were:

– LSTM, GRU: 1 hidden layer, 64 neurons;

– TDNN: fixed input delay window of 20 discrete samples, 1 hidden layer, 64 neurons.

Optimizer: *Adam* (learning rate = 0.001).

Loss function: Mean Squared Error (*MSE*).

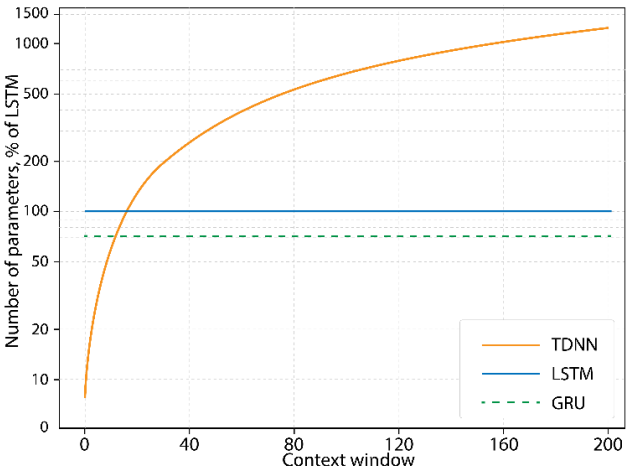


Fig. 3. **Dependence of the number of TDNN and LSTM parameters on the memory window size**
Source: **compiled by the authors**

5.3. Experimental results

Fig. 4 shows the dependence of the *MSE* loss function on the number of training epochs for a model built using the reference model method based on TDNN (TDNN-Ref), for an LSTM model with randomly initialized weights (LSTM-Random), for a GRU model with randomly initialized weights (GRU-Random), using the reference LSTM method without alignment (LSTM-Ref-Naive), and using the reference LSTM method with neuron alignment (LSTM-Ref-Aligned).

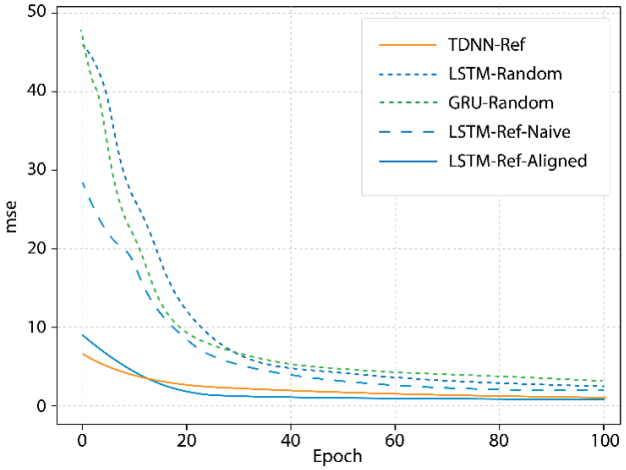


Fig. 4. **Dependence of the MSE loss functions during training on the number of training epochs for the models:**
TDNN-Ref LSTM-Random; GRU-Random; LSTM-Ref-Naive, LSTM-Ref-Aligned
Source: **compiled by the authors**

To guarantee statistical reliability, the evaluation was executed across 10 independent simulation runs. Table 1 summarizes the empirical training outcomes constrained by standard early-stopping criteria.

5.4. Analysis of the results

Fig. 4 and Table 1 demonstrate the advantage of using pre-trained reference LSTM models for the identification of complex nonlinear dynamic systems.

Table 1. **Training metrics across evaluated architectures based on early stopping criteria**

Method	Initial error (*MSE*), Mean±SD	Epochs to convergence, Mean±SD	Final error (*MSE*), Mean±SD	Training time (s), Mean±SD
TDNN-Ref	8.13±0.45	77±5	0.141±0.012	129±8
LSTM-Random	43.71±8.12	176±18	0.130±0.015	439±35
GRU-Random	45.50±6.85	145±14	0.132±0.011	325±24
LSTM-Ref-Naive	26.29±3.40	159±12	0.115±0.009	397±22
LSTM-Ref-Aligned	9.61±0.82	40± 4	0.042±0.004	118±6

Source: **compiled by the authors**

Initialization efficiency

The proposed *LSTM-Ref-Aligned* technique demonstrated a drastically reduced initial error margin (9.61) compared to purely stochastic initialization. This empirically verifies the core hypothesis: the mathematically aligned superposition of reference models generates a highly precise coarse matrix that fundamentally encapsulates the system's governing dynamics prior to fine-tuning. Conversely, the *LSTM-Ref-Naive* protocol yielded an initial error of 26.29, closely mimicking random noise. This confirms that attempting superposition without executing topological alignment induces severe structural corruption, destroying the physical knowledge embedded within the reference matrices.

Convergence rate

The *LSTM-Ref-Aligned* model achieved absolute convergence in merely 40 epochs on average. This represents an acceleration factor of 4.4× compared to standard stochastic LSTM training (*LSTM-Random*) and a 1.9× improvement over the standard TDNN reference methodology (*TDNN-Ref*) under a strict validation threshold of *MSE* = 0.15. This confirms that the proposed technique substantially slashes required processing cycles while preserving or exceeding baseline accuracy.

Modeling accuracy

The terminal MSE of the proposed method (0.042) severely undercuts the TDNN benchmark (0.141). This disparity is directly attributable to the TDNN's fundamental architectural bottleneck: a rigid 20-sample delay window completely fails to capture the heavily delayed, low-frequency dynamics intrinsic to the Lur'e system. In contrast, the LSTM architecture naturally accommodates these phenomena via its recursive memory cells.

Comparison with TDNN

TDNN trains fairly quickly (on average 129 s), but yields lower accuracy. LSTM with random weight initialization takes a long time to train and yields average accuracy. The LSTM-Ref-Aligned method combines the training speed of TDNN (due to a better starting point) and the accuracy of LSTM (due to its architecture).

Comparison with GRU

GRU with random weight initialization trains on average faster (325 s) than a similar LSTM model (439 s) and achieves accuracy comparable to LSTM. These advantages hold when using a memory length of about 20 steps. As the memory length increases, LSTM models provide better accuracy.

6. DISCUSSION OF RESULTS

Migrating the reference model paradigm to the LSTM architecture successfully eliminates the method's most critical fundamental constraint: strict dependence on the spatial delay window. Utilizing a TDNN necessitates a crippling compromise: restricting the window destroys long-term dynamic resolution, whereas expanding it exponentially inflates algorithmic complexity, virtually guaranteeing catastrophic overfitting. The LSTM architecture – when strategically initialized via aligned reference models – facilitates the construction of highly complex nonlinear dynamic models entirely free from spatial window constraints. It simultaneously slashes computational overhead while guaranteeing extreme predictive precision.

Empirical simulations utilizing the nonlinear Lur'e system substantiate the hypothesis that local minima within the highly non-convex LSTM parameter space exhibit strict linear connectivity characteristics, provided permutation symmetries are computationally resolved. Reference models trained on disparate – yet physically homologous – sub-processes will invariably reside within a unified, functionally homologous optimization basin if they are topologically aligned prior to synthesis. This theoretical confirmation paves the way for generating robust, universally applicable libraries of pre-trained *dynamic primitives*. These libraries will enable the rapid, modular synthesis of initialization matrices capable of identifying industrial systems of virtually infinite complexity.

7. CONCLUSIONS

This paper details the theoretical extension and algorithmic implementation of the reference model method for the identification of highly nonlinear dynamic systems, achieved via the integration of LSTM architectures and advanced internal neuron alignment protocols.

The investigation yielded the following definitive conclusions:

Scientific contribution

This study introduces the application of structural neuron alignment techniques to execute the valid superposition of reference LSTM models within the scope of system identification. Empirical evidence conclusively demonstrates that the strict mathematical elimination of permutation invariance is an absolute prerequisite for the valid parameter-space superposition of recurrent neural networks. The standard reference model method has been

ISSN 2663-0176 (Print)
ISSN 2663-7731 (Online)

significantly advanced by migrating from rigid spatial architectures to dynamic LSTM representations. This methodology eliminates fundamental limitations imposed by spatial memory windows and substantially compresses the time required to model a target system without compromising accuracy, primarily by circumventing the exhaustive pre-training phases traditionally required for coarse modeling.

The practical value

The study delivers a functional, algorithmic implementation of a neuron alignment protocol tailored for LSTM architectures, effectively resolving the permutation invariance anomaly inherent to hidden state variables. The executed algorithm reliably accelerates training speeds by a factor of 4.4 compared to conventional stochastic initialization methods while simultaneously amplifying identification accuracy by a factor of 3 relative to existing TDNN benchmarks.

Applicability

The proposed methodology is exceptionally robust for modeling complex dynamic systems characterized by unknown internal topologies, particularly when operating under extreme empirical data scarcity.

USE OF ARTIFICIAL INTELLIGENCE

While preparing this article, the authors used Google Gemini to check the logical consistency of the text to improve its readability and ensure that the bibliographic descriptions comply with international standards. The authors are solely responsible for the content of this publication.

REFERENCES

1. Schoukens, J. & Ljung, L. “Nonlinear System Identification: A User-Oriented Road Map”. In *IEEE Control Systems Magazine*. 2019; 39 (6): 28–99. DOI: https://doi.org/10.1109/MCS.2019.2938121.

2. Pillonetto, G., Aravkin, A., Gedon D., et al. “Deep networks for system identification: A survey”. *Automatica*. 2025; 171: 111907. DOI: https://doi.org/10.1016/j.automatica.2024.111907.

3. Wang, Q.-G. & Zhang, L. “System identification in the network era: A survey of data issues and innovative approaches”. In *IEEE/CAA Journal of Automatica Sinica*. 2025; 12 (7): 1305–1319. DOI: https://doi.org/10.1109/JAS.2024.125109.

4. Rossi, R., Murari, A. & Gaudio, P. “On the potential of time delay neural networks to detect indirect coupling between time series”. *Entropy*. 2020; 22 (5): 584. DOI: https://doi.org/10.3390/e22050584.

5. Torres, J. F., Hadjout, D., Sebaa, A., et al. “Deep learning for time series forecasting: A survey”. *Big Data*. 2021; 9 (1): 3–21. DOI: https://doi.org/10.1089/big.2020.0159.

6. Tsmots I. G., Berezsky O. M., Berezkyy M. O. “Methods and Hardware to Accelerate the Work of a Convolutional Neural Network”. *AAIT.* 2023; 6 (1): 13–27. DOI: https://doi.org/10.15276/aait.06.2023.1.

7. Molina, D., Liang, J., Harley R., et al. “Comparison of TDNN and RNN performances for neuro-identification on small to medium-sized power systems”. *IEEE Symposium on Computational Intelligence Applications in Smart Grid (CIASG)*. Paris, French Guiana. 2011. p. 1–8. DOI: https://doi.org/10.1109/CIASG.2011.5953344.

8. Bai, S., Kolter J. Z. & Koltun, V. “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling”. *arXiv*. 2018. DOI: https://doi.org/10.48550/arXiv.1803.01271.

9. Yu, Y., Si, X., Hu, C. & Zhang, J. “A review of recurrent neural networks: LSTM cells and network architectures”. *Neural Computation*. 2019; 31 (7): 1235–1270. DOI: https://doi.org/10.1162/neco_a_01199.

10. Sherstinsky, A. “Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network”. *Physica D: Nonlinear Phenomena*. 2020; 404: 132306. DOI: https://doi.org/10.1016/j.physd.2019.132306.

11. Greff, K., Srivastava, R. K., Koutník, J., et al. “LSTM: A search space Odyssey”. *IEEE Transactions on Neural Networks and Learning Systems*. 2017: 28 (10): 2222–2232. DOI: https://doi.org/10.1109/TNNLS.2016.2582924.

12. Kashkevich, S., Litvinenko, O., Shyshatskyi, A., Salnyk, S. & Velychko, V. “The method of self-organization of information networks in the conditions of the complex influence of destabilizing factors”. *Advanced Information Systems*. 2024; 8 (3): 59–71, https://scopus.com/pages/publications/85215730245. https://doi.org/DOI: 10.20998/2522-9052.2024.3.07.

13. Mienye, I. D., Swart, T. G. & Obaido, G. “Recurrent Neural Networks: A comprehensive review of architectures, variants, and applications”. *Information*. 2024; 15 (9): 517. DOI: https://doi.org/10.3390/info15090517

14. Gruber, N. & Jockisch, A. “Are GRU cells more specific and LSTM cells more sensitive in time series forecasting?” *Frontiers in Applied Mathematics and Statistics*. 2020; 3. DOI: https://doi.org/10.3389/fams.2020.00067.

15. Pölz, A., Blaschke, A. P., Komma, J., et al. “Transformer versus LSTM: A comparison of deep learning models for karst spring discharge forecasting”. *Water Resources Research*. 2024; 60: e2022WR032602. DOI: https://doi.org/10.1029/2022WR032602.

16. Fares, I. A., et al. “Deep transfer learning based on hybrid swin transformers with LSTM for intrusion detection systems in IoT environment”. *IEEE Open Journal of the Communications Society*. 2025; 6: 4342–4365. DOI: https://doi.org/10.1109/OJCOMS.2025.3569301.

17. Zhou, H., et al. “Informer: Beyond efficient transformer for long sequence time-series forecasting”. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021: 35 (12): 11106–11115. DOI: https://doi.org/10.1609/aaai.v35i12.17325.

18. Fomin, O. O., Speranskyy, V. O., Orlov, A. A., Tataryn, O. V. & Kushchevyi, D. V. “Method of reference models for synthesis of intellectual systems of nonlinear dynamic objects identification”. *Herald of Advanced Information Technology*. 2024; 7 (3): 262–274, https://scopus.com/pages/publications/84896376427. DOI: https://doi.org/10.15276/hait.07.2024.18.

19. Li, Z., Li, H. & Meng, L. “Model compression for deep neural networks: A survey”. *Computers*. 2023; 12 (3): 60. DOI: https://doi.org/10.3390/computers12030060.

20. Wang, H., et al. “A comprehensive survey on training acceleration for large machine learning models in IoT”. *IEEE Internet of Things Journal*. 2022; 9 (2): 939–963. DOI: https://doi.org/10.1109/JIOT.2021.3111624.

21. Wortsman, M., et al. “Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time”. In *Proc. International Conference on Machine Learning*. 2022. DOI: https://doi.org/10.48550/arXiv.2203.05482.

22. Liu, J., Zhang, Yu., Zhang, R., et al. “Domain-Adaptive model merging across disconnected modes”. *arXiv preprint*. 2026. DOI: https://doi.org/10.48550/arXiv.2603.05957.

23. Ito, A., Yamada, M. & Kumagai, A. “Analysis of linear mode connectivity via permutation-based weight matching”. *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2402.04051.

24. Qi, B., et al. “Less is more: efficient model merging with binary task switch”. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Nashville, TN, USA. 2025. p. 15265–15274. DOI: https://doi.org/10.1109/CVPR52734.2025.01422.

25. Ainsworth, S. K., Hayase, J. & Srinivasa, Y. “Git Re-Basin: Merging models modulo permutation symmetries”. *International Conference on Learning Representations (ICLR)*. 2023. DOI: https://doi.org/10.48550/arXiv.2209.04836.

26. Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D. & Khazaeni, Y. “Federated learning with matched averaging”. *International Conference on Learning Representations (ICLR)*. 2020. DOI: https://doi.org/10.48550/arXiv.2002.06440.

27. Feizi-Derakhshi, M.-R. & Mutasher AL-Talebei, W. G. “Dcgan data balancing to improve accuracy of hybrid CNN-LSTM intrusion detection framework in sdn environment”. *Advanced Information Systems*. 2025; 9 (4): 120–131, https://scopus.com/pages/publications/84896376427. DOI: https://doi.org/10.20998/2522-9052.2025.4.14.

28. Guarneros-Sandoval, A., Ballesteros, M., Salgado, I. & Chairez, I. “Stable learning laws design for long short-term memory identifiers for uncertain discrete systems via control Lyapunov functions”. *Neurocomputing*. 2022; 491: 144–159. DOI: https://doi.org/10.1016/j.neucom.2022.03.070.

Conflicts of Interest: The authors declare that they have no conflict of interest regarding this study, including financial, personal, authorship or other, which could influence the research and its results presented in this article

Received 26.03.2026
Received after revision 08.06.2026
Accepted 17.06.2026

DOI: https://doi.org/10.15276/hait.09.2026.20
UDC 004.942

Метод опорних моделей на основі довгої короткочасної пам'яті для ідентифікації нелінійних динамічних систем

Фомін Олександр Олексійович[1)]
ORCID: https://orcid.org/0000-0002-8816-0652; fomin@op.edu.ua. Scopus Author ID: 57103429400
Сперанський Віктор Олександрович[1)]
ORCID: https://orcid.org/0000-0002-8042-1790; speranskyy@op.edu.ua.Scopus Author ID: 54401618900
Верлань Андрій Анатолійович[2)]
ORCID: https://orcid.org/0000-0002-6469-2638; a.verlan@edu.kpi.ua. Scopus Author ID: 57192176307
Татарин Олексій Васильович[1)]
ORCID: https://orcid.org/0009-0005-3888-6569; otataryn@stud.op.edu.ua. Scopus Author ID: 59390593400
Чмелевський Андрій Миколайович[1)]
ORCID: https://orcid.org/0009-0008-6450-6875; stech@stud.op.edu.ua
[1)] Національний університет «Одеська політехніка», проспект Шевченка, 1. Одеса, 65044, Україна
[2)] Норвезький університет природничих та технічних наук, Торгарден, NO-7491. Трондхейм, Норвегія

АНОТАЦІЯ

Актуальність дослідження зумовлена тим, що сучасні інженерні задачі та інтелектуальні системи управління вимагають високоефективних математичного забезпечення для моделювання складних нелінійних динамічних об'єктів, яке здатне розв'язати фундаментальну суперечність між забезпеченням точності апроксимації та мінімізацією обчислювальних витрат і тривалості навчання моделей. Традиційні підходи на основі нейронних мереж із часовими затримками обмежені фіксованим розміром вікна пам'яті, що унеможливлює адекватний опис процесів із глибоким запізненням та нестаціонарними режимами без істотного зростання кількості параметрів. **Метою роботи** є скорочення часу побудови моделей нелінійної динаміки при забезпеченні заданої точності моделювання шляхом розвитку методу опорних моделей за рахунок використання архітектури нейронної мережі з довгою короткочасною пам'яттю. Для досягнення цієї мети поставлено такі **завдання**: здійснити теоретичний розвиток методу опорних моделей через перехід до динамічних рекурентних структур, розробити алгоритм вирівнювання параметрів опорних моделей з довгою короткочасною пам'яттю для подолання пермутаційної інваріантності та виконати експериментальну верифікацію запропонованого підходу. **Методи дослідження** базуються на теорії ідентифікації систем, методологіях трансферного навчання та злиття моделей. Для усунення пермутаційної симетрії прихованих шарів адаптовано оптимізаційний алгоритм узгодження ваг на основі розв'язання задач лінійного програмування, що дозволяє перевести незалежні матриці параметрів у спільний басейн функції втрат перед їх суперпозицією. **Результатами роботи** є формалізація методу опорних моделей на основі довгої короткочасної пам'яті та алгоритму вирівнювання параметрів цих моделей, який забезпечує лінійну зв'язність мод у просторі параметрів. Експериментальна верифікація на тестовій нелінійній системі довела, що запропонований метод прискорює збіжність моделі майже у 2 рази порівняно з підходом на основі нейронних мереж із часовими затримками та зменшує фінальну помилку в 3.3 рази. **Отримані висновки** підтверджують ефективність застосування рекурентних архітектур у процес суперпозиції моделей. Наукова новизна роботи полягає в розвитку методу опорних моделей на основі довгої короткочасної пам'яті, що дозволяє подолати властиві архітектурам із часовими затримками обмеження, пов'язані зі скінченністю вікна пам'яті. Це забезпечує адекватне моделювання систем, які характеризуються глибоким запізненням, значною інерційністю та складними ефектами нелінійного гістерезису. Крім того, для вирішення проблеми пермутаційної інваріантності прихованих станів незалежно навчених рекурентних мереж, яка традиційно унеможливлює пряме усереднення параметрів ваг моделей, у роботі адаптовано спеціалізований оптимізаційний алгоритм узгодження ваг до структури довгої короткочасної пам'яті.

Ключові слова: моделювання; нелінійна динаміка; нейронні мережі; довга короткочасна пам'ять; об'єднання моделей, трансферне навчання, об'єднання моделей, вирівнювання нейронів, узгодження ваг, лінійна зв'язність режимів

ABOUT THE AUTHORS



Oleksandr O. Fomin - Doctor of Engineering Sciences, Professor, Department of Computerized Systems and Software Technologies. Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0002-8816-0652; fomin@op.edu.ua. Scopus Author ID: 57103429400
***Research field*:** math modeling, information systems, intelligent control systems

Фомін Олександр Олексійович - доктор технічних наук, професор кафедри Комп'ютеризованих систем та програмних технологій. Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044,Україна



Viktor O. Speranskyy - Candidate of Engineering Sciences, Associate Professor, Department of Computerized Systems and Software Technologies. Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0002-8042-1790; speranskyy@op.edu.ua. Scopus Author ID: 54401618900
Research field: nonlinear systems; modeling; identification; software development; neural networks

Сперанський Віктор Олександрович - кандидат технічних наук, доцент кафедри Комп'ютеризованих систем та програмних технологій. Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна



Andrii A. Verlan - Doctor of Engineering Sciences, Professor, Department of Software Engineering for Power Industry. Norwegian University of Science and Technology, Torgarden, NO-7491. Trondheim, Norway
ORCID: https://orcid.org/0000-0002-6469-2638; a.verlan@edu.kpi.ua. Scopus Author ID: 57192176307
***Research field*:** mathematical modeling, energy processes, artificial intelligence systems

Верлань Андрій Анатолійович - доктор технічних наук, професор кафедри Інженерії програмного забезпечення в енергетиці. Норвезький університет природничих та технічних наук. Торгарден, NO-7491. Трондхейм, Норвегія



Oleksiy V. Tataryn - graduate student, Department of Computerized Systems and Software Technologies. Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0000-0002-3256-5044; otataryn@stud.op.edu.ua. Scopus Author ID: 59390593400
Research field: information technologies, artificial neural networks

Татарин Олексій Васильович - аспірант кафедри комп'ютеризованих систем та програмних технологій, Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна



Andriy M. Chmelevskyi - graduate student, Department of Computer Systems. Odesa Polytechnic National University, 1, Shevchenko Ave. Odesa, 65044, Ukraine
ORCID: https://orcid.org/0009-0008-6450-6875; stech@stud.op.edu.ua
Research field: computer engineering, control of mobile objects

Чмелевський Андрій Миколайович - аспірант кафедри комп'ютерних систем, Національний університет «Одеська політехніка», пр. Шевченка, 1. Одеса, 65044, Україна