Neural Computing and Applications (2021) 33:16745–16757
https://doi.org/10.1007/s00521-021-06271-5

ORIGINAL ARTICLE

Dynamical system parameter identification using deep recurrent cell networks

Which gated recurrent unit and when?

Erdem Akagündüz[1] · Oguzhan Cifdaloz[2]

Received: 3 November 2020 / Accepted: 26 June 2021 / Published online: 10 July 2021

Abstract
In this paper, we investigate the parameter identification problem in dynamical systems through a deep learning approach. Focusing mainly on second-order, linear time-invariant dynamical systems, the topic of damping factor identification is studied. By utilizing a six-layer deep neural network with different recurrent cells, namely GRUs, LSTMs or BiLSTMs; and by feeding input/output sequence pairs captured from a dynamical system simulator, we search for an effective deep recurrent architecture in order to resolve the damping factor identification problem. Our study's results show that, although previously not utilized for this task in the literature, bidirectional gated recurrent cells (BiLSTMs) provide better parameter identification results when compared to unidirectional gated recurrent memory cells such as GRUs and LSTM. Thus, indicating that an input/output sequence pair of finite length, collected from a dynamical system and when observed anachronistically, may carry information in both time directions to predict a dynamical systems parameter.

Keywords Dynamical systems parameter identification · Recurrent cells · LSTM · GRU · BiLSTM

1 Introduction

System identification describes a set of methods, which uses experimental input/output data from a system, in order to identify its dynamical properties. Depending on the class of systems under inspection, there are a number of approaches that may be applied to system identification [3, 26, 42]. While nonparametric methods [26, 27, 40, 52] try to estimate a generic model from step responses, impulse responses, frequency responses, etc., parametric methods [4, 16, 18, 26, 33, 41] aim at estimating parameters within a user-specified model.

The steps taken to identify a system may be generalized as follows: obtaining experimental data; determining a structure for the model; devising a criterion for model fitting; estimating the parameters; and finally, model validation. System identification methods utilize different mathematical models in order to achieve their objectives. Models can be continuous-time (differential) equations, discrete-time (difference) equations, or a hybrid combination. Models can also be described in a variety of ways. The use of state-space models based on transfer functions is common. Generating the experimental data involves several approaches. However, input signals that excite all the relevant frequencies of a system is an important factor. Identification can be achieved either on-the-fly or offline. The identification of parameters and the use of identified parameters on-the-fly in order to update controller parameters are highly related to adaptive control schemes.

In this paper, we tackle the problem using a machine learning approach. By utilizing different deep recurrent neural network (DRNN) architectures and defining the issue as a sequence regression problem, we aim at finding

✉ Erdem Akagündüz
akaerdem@metu.edu.tr

Oguzhan Cifdaloz
oguzhanc@cankaya.edu.tr

1 Graduate School of Informatics, Middle East Technical University (METU), Ankara, Turkey

2 Department of Electrical and Electronics Engineering, Çankaya University, Ankara, Turkey

the most practically effective architecture. We compare different gated recurrent cells and then analyze at what exact instant and with what kind of input the system should be excited in order to obtain the best parameter identification results.

The remainder of this paper is organized as follows: the following subsections introduce the problem statement and the related literature. Section 2 describes the dynamical systems model for which the parameters are to be identified. Section 3 presents details of the deep recurrent neural network model used to solve the problem. Section 4 details the experimental setup, including the simulation environment, the extent of the dataset created, and the preprocessing techniques used for training the model's input. Section 5 presents a discussion of the experimental results. Finally, Sect. 6 concludes the paper and outlines potential future research.

1.1 Problem statement

This paper is focused on parameter identification of second-order, linear time-invariant dynamical systems. These systems can be described as second-order transfer functions. A general second-order system has two poles, which can be both real and complex conjugate. Complex conjugate poles are associated with two parameters: natural frequency and damping factor. In this paper, it is assumed that the poles are complex conjugates, and the study aims at identifying the damping factor.

Damping factor is largely an uncertain parameter, especially for electromechanical systems with small inertia/spring and inertia/friction ratios [20, 29]. In such systems, although inertia is a relatively known factor, considerable uncertainty is associated with the spring and friction constants. In this paper, the objective is to identify the damping factor through the use of machine learning algorithms. One important aspect of the method in question is that the algorithms are unaware of the model structure. The only information fed to the algorithms is that a certain input/output pair is associated with a fixed damping factor.

1.2 Related literature

There has been growing interest in the combination of machine learning techniques, specifically recurrent neural networks and dynamic system modeling/identification, since the first introduction of recurrent neural networks [9, 38, 43]. For a detailed overview on the subject, see [7, 44]. The earlier approaches used standard RNNs to attack the problem, which yielded some promising but limited results [10, 23, 30, 32, 39, 45]. The limiting factor with the RNNs, not only in dynamical systems modeling but also in general, was their difficulty in training to learn short- or long-term dependencies within the input sequence. Some other hybrid RNN architectures such as artificial deep belief networks were also utilized [6] for the problem, but they also suffered similar limitations. The main reason for this limitation was that backpropagation deep through time (i.e., sequence dimension) gave rise to the so-called *vanishing gradients* problem, which was first mentioned in 1991 by [21]. In order to overcome this, the same author proposed a specific RNN architecture with gated units in 1997, named long-short-term memory (LSTM) [22].

The idea of the LSTM was to create a memory cell within the RNN architecture by ensuring a constant error overflow, so that long- (or short-) term dependencies were not lost during backpropagation through time. LSTM attracted significant praise and was applied to numerous sequence modeling problems, including dynamical system modeling and identification [51]. Different versions of recurrent cell networks were also proposed such as convex-based LSTM [51], the gated recurrent unit (GRU) [8] and also bidirectional (bi-)LSTM [15].

Alternatively, deep convolutional neural networks (CNN) have also been utilized in the literature [1, 11] for the same problem. Since CNN architectures lack the ability to create time dependencies (i.e., memory), these approaches are uncommon when compared to RNN-based techniques. However, we believe that unexplored feature extraction capabilities of CNNs from dynamical system sequence data are still likely to attract more attention.

Training deep recurrent neural networks that can be successfully utilized for any type of intelligent decision, including system identification, is an active field of research [19, 35]. The last few years have shown an increasing trend in studies that focus on applying deep learning techniques to the systems identification problem [2, 5, 24, 25, 28, 37, 46, 50]. Some of the recent methods even utilize LSTMs for the problem [14]. However, the literature contains no approved or generally accepted off-the-shelf deep learning architecture for dynamical systems parameter identification. Although there have been comparative studies published, such as [34], which utilized a multilayered artificial neural network (ML-ANN), an RNN, a single LSTM cell, and a single GRU cell to assess performance in nonlinear systems identification, there remain unanswered questions such as "How deep a network is needed?," "Which types of layers are needed?" and "What kind of a recurrent cell performs better?" for the dynamical systems identification problem.

2 Dynamic model description

The dynamic model considered in this paper is described by a second-order, linear differential equation. Second-order linear differential equations arise in a variety of systems, such as in rotational dynamics, which are described in their general form as

$$J\ddot{\theta} + b\dot{\theta} + k\theta = u \tag{1}$$

where $J > 0$ denotes the moment of inertia, $b > 0$ denotes the viscous damping (friction) coefficient, $k > 0$ denotes the spring constant, u denotes the externally applied torque, and θ denotes the angle of rotation. In this paper, linearized rotational dynamics of a gimbal are considered. A simple depiction and the model block diagram representing the system is illustrated in Fig. 1.

The differential equation that describes this system from the torque input u, to the angle output θ is given in state-space form by

$$\begin{aligned} \begin{bmatrix} \dot{q} \\ \dot{\theta} \end{bmatrix} &= \begin{bmatrix} -b/J & -k/J \\ 1 & 0 \end{bmatrix} \begin{bmatrix} q \\ \theta \end{bmatrix} + \begin{bmatrix} 1/J \\ 0 \end{bmatrix} u, \\ \theta &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} q \\ \theta \end{bmatrix} \end{aligned} \tag{2}$$

where $q \triangleq \dot{\theta}$ represents the angular rate. The transfer function from the input u, to the output θ is given by

$$T_{u\theta} = \frac{1/J}{s^2 + b/Js + k/J} \tag{3}$$

This transfer function can be represented as a standard second order system multiplied by a constant gain, α, and given by:

$$\frac{\alpha\omega_n^2}{s^2 + 2\zeta\omega_n s + \omega_n^2} \tag{4}$$

where $\omega_n^2 \triangleq k/J$, $2\zeta\omega_n \triangleq b/J$, and $\alpha \triangleq 1/k$. In a standard second-order system, ω_n denotes the natural frequency and ζ is called the damping factor.

Second-order systems may have complex conjugate poles, and hence their impulse responses may be oscillatory. The poles of a standard second-order system given in Eq. 4 are at

$$s_{1,2} = -\zeta\omega_n \pm j\omega_n\sqrt{1 - \zeta^2} \tag{5}$$

In terms of the system parameters, the natural frequency and the damping factor are given by

$$\omega_n = \sqrt{\frac{k}{J}} \quad \text{and} \quad \zeta = \frac{b}{\sqrt{4kJ}}. \tag{6}$$

Since all the parameters (J, b, k) of the system are positive, from Eq. 6, it can be seen that ζ cannot take negative values or zero. $\zeta = 0$ implies $b = 0$, i.e., a system with zero friction. However, small friction always exists in a mechanical system such as described in this paper. $\zeta = 0$ may also imply $k \to \infty$ or $J \to \infty$. Both of these two conditions physically correspond to a rigid system, with no freedom of rotation.

Hence, for the system in consideration, we may safely assume that $\zeta > 0$. With this assumption, two cases are of importance:

1. For $\zeta \geq 1$ (i.e., $b^2 \geq 4$ kJ), the transfer function given in Eq. 4 can be described as two first-order systems connected in series. Both poles are stable real poles. This case is excluded from the study because, in such a series combination, the damping factor loses its oscillatory effect on the system, and there will be no overshoot.
2. For $0 < \zeta < 1$ (i.e., $b^2 < 4$ kJ), the poles of the transfer function given in Eq. 4 are stable complex conjugate poles. This study aims to identify the damping factor that results in (undesired) oscillations. This case is addressed in this paper. Naturally, when $0.5 < \zeta < 1$, oscillations die out relatively quickly. *However, one of the reasons why $0 < \zeta < 1$ is targeted in this study is that there may be a large and varying uncertainty associated with ζ that the control system designers*

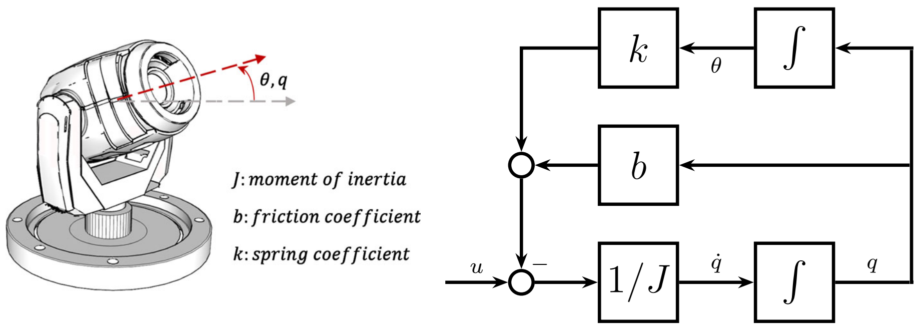

Fig. 1 An illustration of a gimbal (left) and its linear model block diagram (right)

would like to know in order to increase the performance of their designs and not have to design a conservative controller.

The unit step response of the second-order system given in Eq. 4 is given by

$$y(t) = \alpha\left[1 - \frac{e^{-\zeta\omega_n t}}{\sqrt{1-\zeta^2}}\sin\left(\omega_d t + \tan^{-1}\frac{\sqrt{1-\zeta^2}}{\zeta}\right)\right] \tag{7}$$

where $\omega_d \triangleq \omega_n\sqrt{1-\zeta^2}$.

It is noted that the damping factor, ζ, impacts all aspects of the step response, namely the time constant, frequency, phase, and overshoot. Specifically, because of its impact on oscillations in the system response, estimating ζ as accurately as possible is critical. However, in a real physical system, estimating ζ can prove to be difficult, due to uncertainties in the parameters and variations of parameter values in time. Temperature variations are notorious for their effect on parameter values. One other important reason for parameter uncertainty is the mass manufacturing process. In practice, parameters are often identified for a handful of prototypes, and a control design is based on those identified parameters and implemented on all manufactured devices. This process results in undesired variations in closed-loop performance characteristics among products. The procedure described in this paper proposes a method to identify system parameters for an automated custom control design for each device.

3 Deep learning model

The model proposed in this study that predicts a parameter of a given dynamical system is a deep recurrent neural network (DRNN). DRNNs are extensions of RNNS with additional layers such as nonlinearity layers (e.g., ReLU layers), decision layers (e.g., dense, fully connected layers), and regularization layers (e.g., drop-out layers) The model we propose in this paper is a 6-layered DRNN (see Table 1) that predicts a dynamical system parameter (damping coefficient ζ in our case) of the model, when a Δ second (Δ being 3 s in our experiments) input/output (I/O) sequence pair is fed into it.

As shown in Table 1, the first layer of the proposed DRNN architecture is an input layer that accepts an input/output sequence pair of Δ-seconds. The size of the input/output sequence pair that is fed into the model is $2\times(\Delta \cdot f_s)$, with f_s being the sampling frequency of the system and $\Delta \cdot f_s$ being the sequence length. However, as shown in Table 1, the input fed into the recurrent cell of the DRNN model is sizes 168×11, with 11 being the sequence length and 168 (42×4) being the feature vector size. As explained in detail in the next subsection, at the initial layer of the proposed DRNN architecture, a frequency domain transform is applied to the Δ-seconds input/output sequence pairs, in order to obtain a 168×11 sequence. This frequency-domain sequence is the actual input fed into the recurrent cells of the succeeding layer.

The ability to automatically construct a set of feature extractors, in a data-driven and task-dependent manner, is the trademark of deep neural networks (DNNs). Accordingly, one may argue that instead of utilizing an initial handcrafted transform layer, as in the proposed architecture, a set of convolutional layers can provide the necessary transforms (i.e., features) needed for the system. However, we chose to use a frequency transform as the first layer, mainly due to two reasons. First, the parameter that we aim at predicting is a coefficient of the Laplacian transform of a dynamic system, which we assume to be linear. Thus, transforming the input into a frequency domain, heuristically leads to creating a supposedly linearly separable feature space. Second, by using a frequency transform, we shorten the sequence length, which in turn reduces the effective depth of the architecture. The depth of a deep neural model is one of the main factors that helps in its training for the desired task [13, 17, 49]. Although it is the depth of a DNN that helps it create abstract features from the data, depth is also responsible for certain problematic training issues such as vanishing gradients [21]. While we refer to the proposed architecture as "6-layered," the depth of an RNN can be considered ambiguous. During training, the RNN layer is propagated through "time," hence the depth is effectively related to the sequence length (or the

Table 1 The proposed deep recurrent neural network model

Layer No.	Type	Layer properties
1	Input transform	Frequency domain transform (42×4) × *11 tensors*
2	RNN cell	RNN cell (GRU, LSTM, or BiLSTM) *with 256 hidden units*
3	Fully connected	Fully connected layer *256 nodes*
4	ReLU	ReLU nonlinearity layer
5	Dropout	Dropout layer *50% drop*
6	Fully connected	Fully connected layer *single output node*
–	Regression	Mean-squared-error loss

"truncation length" if the backpropagation in time is truncated [36, 48, 53]). If we use a shorter frequency domain sequence (such as size 11), instead of a $\Delta \cdot f_s$ time sequence (which is practically 3000 in our case), the actual depth of the backpropagation is significantly reduced and training the DRNN becomes a much easier task. Regarding the details of the input layer frequency transform, please refer to the next subsection.

The input layer is followed by the recurrent layer of the network. The recurrent layer can be any type of a recurrent cell that accepts sequence inputs, such as a GRU [8], an LSTM [22], or a BiLSTM [15] cell. In our experiments, these three types of recurrent cells were bench-marked in order to discover which recurrent cell was more successful in predicting dynamical system parameters. Details on these recurrent cells are provided in Sect. 3.2.

Any recurrent cell can be designed to output a single value. Hence, the first two layers are technically sufficient to create a parameter prediction (i.e., regression) network. However, in our DRNN model, the recurrent layer is succeeded by a fully connected layer. The hidden units (totaling 256 nodes) of the recurrent cell are fully connected to this dense layer so that more complex features can be obtained using the hidden units (also referred to as the "states") of the recurrent cell. Additionally, this fully connected layer is succeeded by a rectified linear unit (ReLU) [31], with the purpose of creating non linearity within the decision space. Moreover, the ReLU layer is followed by a drop-out layer [47] for the purposes of regularization, and for avoiding the issue of feature overfitting. Finally, another fully connected layer, this time having a single output value that provides the parameter value to be predicted, was appended to the network. During training, an L2 norm regression layer was used to feed the backpropagated derivatives to the stochastic gradient descent optimizer.

In summary, the proposed DRNN architecture is a dynamical system parameter prediction network, with the fundamental properties being transforming a time signal to the frequency domain, utilizing recurrent operation within a sequence, convolving advanced features, providing nonlinearity in the feature space and avoiding overfitting. The training details of the proposed DRNN are provided in Sect. 4.

3.1 Input sequence

The recorded input/output pair sequences are time signals. However, the sequence that we fed into the proposed DRNN model was a frequency domain representation of this time signal, which was obtained by utilizing the short-time Fourier transform (STFT). STFT is a sequence of Fourier transforms of a windowed signal. Instead of providing the frequency information averaged over the entire signal time interval (like the standard Fourier transform), STFT provides the time-localized frequency information for situations, in which frequency components of a signal vary over time. STFT is widely used as features for time signals in many intelligent signal processing applications such as audio signal processing [12].

$$X_{\mathrm{STFT}}[m,n] = \sum_{k=0}^{L-1} x[k] \cdot g[k-m] \cdot e^{-j2\pi nk} \tag{8}$$

In Eq. (8), $x[k]$ denotes a signal and $g[k]$ denotes an L-point window function, hence STFT of $x[k]$ can be interpreted as the Fourier transform of the product, $x[k] \cdot g[k--m]$. Consequently, the calculated $_{\mathrm{STFT}}[m,n]$ is a 2-dimensional complex matrix, where the first dimension represents time, and the second dimension represents sample frequencies. The finite size of the window function $g[k]$ and how much overlap each neighboring frame has designate the sequence size of the calculated $X_{\mathrm{STFT}}[m,n]$.

In our model, $x[k]$ is a $\Delta \cdot f_s$ long sequence with 2 dimensions, that belong to the input/output sequence pair, for both of which STFT are calculated separately. The size of the window function is selected as $L = k \cdot \Delta \cdot f_s$, with p inter-frame overlap, therefore creating a $\frac{1-k}{k(1-p)} + 1$ long sequence in the STFT time dimension. In the frequency dimension, the range is limited to a certain interval with an exponential sampling of N distinct frequency values; ergo, creating two complex sequences of size $N \times (\frac{1-k}{k(1-p)} + 1)$ (also called spectrograms) to be fed to the recurrent cell of the network. In Sect. 4, the actual STFT parameters utilized in our experiments, plus visualizations of the implemented frequency domain transform (see Fig. 2) are provided.

3.2 Recurrent cells

A standard (or so-called vanilla) RNN is basically a feed-forward neural network unrolled in time. It is fundamentally a set of weighted connections between a number of hidden states of the network and the same hidden states from the last time point, providing some sort of "memory." The challenge is that this memory is fundamentally limited in the same way that training very deep networks is difficult, due to factors such as vanishing gradients, hence limiting the memory of vanilla RNNs.

One solution to this elemental problem of RNNs, is creating so-called *cell states* within the recurrent architecture that consist of a common thread through time, affected solely by linear operations at each time step. These recurrent cells can remember short-term memories for

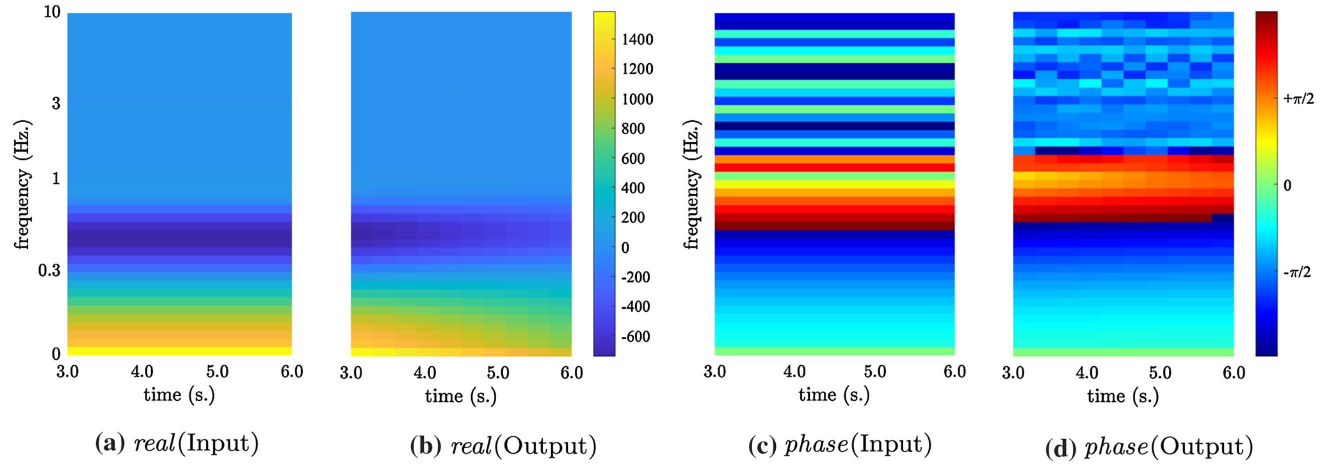

Fig. 2 The figures each depict a visual representation of the feature vector fed into the recurrent cells

relatively longer sequences, mainly because the cell state connection to previous cell states is interrupted only by linear elementary operations such as multiplication and addition.

Two well-known examples of these recurrent architectures are gated recurrent units (GRU) [8] and long-short term memory (LSTM) [22] cells. Although GRU is slightly simpler in its architecture, both cells are characterized by a persistent linear cell state surrounded by nonlinear layers feeding input and parsing output from it. Technically, the cell state functions jointly with so-called gating layers that have the ability to "forget," "update" or "reset" the state of the cell, hence providing long or short-term memory capabilities. We benchmarked these two fundamental recurrent cells in our experiments.

GRU and LSTM, as a result of their architectural design, can only preserve information of the past (i.e., data from earlier in the input sequence), simply because they can only see input from the past. However, the context in a sequence usually preserves information from both past and future. Although the linear dynamical system we use in this study is a causal system, an input/output pair of Δ seconds, when observed anachronistically, may carry information in both directions. For example, we may observe and rationalize an output behavior only after we observe the continuity of the event in the output, which is an inconsequential correlation. For this reason, we also utilize the BiLSTM cell [15], which provides, as the name implies, bidirectional input processing. Simply put, the input in a BiLSTM is fed twice, once from beginning to end and then in reverse from the end to the beginning, so the short- or long-term memory can be related both to and from the past.

All three of the aforementioned recurrent cells are benchmarked in our experiments in order to achieve a solid comparison of their capabilities in predicting dynamical system parameters.

4 Experimental setup

In this section, we provide details of the experimental setup, starting with the dynamical system simulation module, which we used to create a dynamical input/output pairs dataset for different dynamical model parameters. Next, we provide the implementation details of how we constructed the frequency transform to be fed into the recurrent cells of the DRNN model.

4.1 Dynamical system simulation module

In an actual physical scenario, torque inputs cannot be applied directly to a mechanical system such as the one described in the paper. Torque is applied via an actuator such as a motor. Accompanying the motor is also a motor driver card. There are transfer functions associated with both the motor and the driver. However, the motors are selected, and the drivers are designed such that the bandwidths of the actuator and its driver are much higher than the mechanical system to be controlled. In addition, both of the transfer functions are usually a combination of first-order filters with known (designed) parameters.

Also, in an actual scenario, the angle θ is measured via sensors, such as encoders. Encoders can be digital or analog, but in either case, they are much faster (or selected to be faster) than the dynamical system on which they are mounted. Their transfer functions are usually provided by the manufacturer, and for the encoder case, their bandwidths are much higher than the mechanical system.

Parameter identification of an actuator-mechanics-sensor system is usually performed by obtaining the frequency response. Sinusoidal inputs of various frequencies with known magnitudes are applied to the system, and steady-state responses are logged. Based on the magnitude amplification and phase lag, transfer functions are identified. If designed carefully, this process provides the frequency response of the mechanical system and contains the data associated with the mechanical system. Actuator and sensor transfer functions are not captured in this process, largely because applying inputs at frequencies that excite this high-frequency actuator, and sensor dynamics is not possible or necessary. Hence, the experiments in the paper are constructed such that the low frequency, mechanical system dynamics are captured. Naturally, sensor noise impacts the data collected, and we have also included noise in the construction of the experiments.

A discrete (Tustin) standard second-order system was used to generate the input-output data. The state-space description of the system is given as

$$x_{k+1} = Ax_k + Bu_k \tag{9}$$

$$\theta_k = Cx_k + Du_k \tag{10}$$

where

$$A = \begin{bmatrix} 0 & 1 \\ 1 - 2\dfrac{\alpha^2 + \omega_n^2}{M} & 2\dfrac{\alpha^2 - \omega_n^2}{M} \end{bmatrix}, B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad C = \frac{4\alpha\omega_n^2}{M^2}\left[\zeta\omega_n \quad (\alpha + \zeta\omega_n)\right], D = \frac{\omega_n^2}{M}, \tag{11}$$

$\alpha = 2/T_s$, and $M = \alpha^2 + 2\alpha\zeta\omega_n + \omega_n^2$.

Sampling time, $T_s = 0.001\ s$, and natural frequency, $\omega_n = 1$ rad/sec were kept fixed.

In order to construct the I/O sequence pair dataset to be used in our experiments, the system was initially excited with five different types of inputs, with eight different damping factor (ζ) values applied for each input. Measurements were corrupted with normally distributed (Gaussian) noise.

The input signals (sampled at 1 kHz) were: a unit step input, a ramp input with a unit slope, and three sinusoids, which each had a magnitude of 10, and frequencies of 0.5, 1, and 2 Hz. Values for the damping factor, ζ, ranged from $\{0.1, 0.2, \ldots, 0.8\}$. Ten seconds of data samples (i.e., 10s $\times$ 1Khz = 10,001) were generated for each input/damping pair.

In total, using five different input types, eight distinct ζ values, and 7001 overlapping 3-s sequences, we created a total of 280,040 ($5 \times 8 \times 7001$) input/output sequence pairs and corresponding ζ values in our dataset.

4.2 Frequency domain transform of I/O pairs

As mentioned in the previous subsections, we fed complex spectrograms of the collected I/O time sequence pairs into our DRNN. In this subsection, we provide details of the implementation of these spectrogram-based features. The I/O pairs we utilized were of $\Delta \cdot f_s$ = 3000 size, as Δ was selected as 3 seconds and f_s as 1 kHz in our experiments. The size of the window function of the STFT was selected as $L = k \cdot \Delta \cdot f_s = 2$ s length (i.e., $k = 2/3$). The overlap ratio p was selected as 0.95, therefore creating 11 long spectrograms in the time dimension.

The frequency range of the spectrograms is limited to between 0 and 10 Hz, and the selection of the frequency range is down to the frequency content of the system and the input signals. The natural frequency of the system considered is approximately 0.159 Hz, and the highest frequency used in the input is 2 Hz. The range of [0, 10] Hz is selected in order to include the frequency content of the system in consideration. It should be noted here that the frequency range of the spectrograms would be different if another system or another input sequence was used. The intermediate frequencies in this range were sampled exponentially, so that the frequency dimension of the spectrograms become a log-frequency axis. The sampling is carried out using the formula:

$$f[h] = \sum_{h=0}^{N-2} 10^{\frac{h}{20}-1} \tag{12}$$

Using the 41 frequencies calculated using Eq. 12 and appending 0 Hz, we obtained a N = 42 long frequency dimension in the spectrograms[1]. Using the two complex 42×11 spectrograms, we then created a feature vector. Since the RNN model we propose can only have real weights and activations, we extracted a real valued feature vector from the spectrograms using the formula:

$$\begin{aligned} v = [&\text{real}(\text{input}_{\text{STFT}}), \text{real}(\text{output}_{\text{STFT}})\ldots \\ &\text{phase}(\text{input}_{\text{STFT}}), \text{phase}(\text{output}_{\text{STFT}})] \end{aligned} \tag{13}$$

In Eq. 13, real$(\cdot)$ denotes an operator that extracts the real part of a complex number and phase$(\cdot)$ denotes an operator that extracts the phase angle of a complex number in radians. Consequently, the feature vector v becomes $(42 \times 4)\times 11$ tensor, as shown in Table 1.

Figure 2 depicts a visual representation of the proposed feature vector. The complex spectrograms obtained from 3-s-long input/output sequences are of 42×11 size. The real parts of the input and output spectrograms are shown in (a) and (b); whereas (c) and (d) depict the phase angles of the

[1] Such that the selected frequencies are [0 Hz, 0.1 Hz, 0.1122 Hz, 0.1259 Hz, ... 7.9433 Hz, 8.9125 Hz, 10 Hz]

input and output spectrograms, respectively, using false color. The specific example shown in Fig. 2 is 3–6 s of a ramp input fed into our system with a damping coefficient (ζ) of 0.1.

4.3 Experiments

Two sets of cross-validation experiments were conducted. The first set of experiments was twofold cross-validation experiments, in which the entire 280,040 I/O sequence pairs were divided into two sets according to their ζ values. For each fold, one half was used for testing, while the other half as used for training and validation. For the first fold, the ζ values in the training + validation sets were $\zeta = \{0.1, 0.3, 0.5, 0.7\}$, whereas the test set included sequences with ζ values $\{0.2, 0.4, 0.6, 0.8\}$. For the second fold, as expected, the ζ values were interchanged between the test and training sets. In practice, the significance of this experiment was assessing the ability of the system to predict the ζ value, which was not used during training. In other words, we aimed at testing the ability of the model to predict an unseen ζ value. This set of experiments are referred to as "$\frac{1}{2}$- sep. ζ," denoting the twofold experiment with "*separate*" ζ values in the test and training sets.

The second set of experiments was also twofold cross-validation experiments, in which the entire 280,040 I/O sequence pairs were divided into two random non-intersecting sets. Again, for each fold, one half was used for testing and the other half for training and validation. In this set of experiments, the dataset was divided randomly; hence each set consisted of an equal distribution of ζ values, and the models were trained and could be tested with any input type and ζ values in the dataset. However, as expected, the testing and training sets did not include identical sequences (although they included partially overlapping sequences). Our intention in these experiments was to assess the ability of the system to predict a previously seen ζ value from a previously (partially) unseen I/O sequence pair. This set of experiments is referred to as "$\frac{1}{2}$ - mix. ζ," denoting the twofold experiment with "*mixed*" ζ values in the testing and training sets.

Both sets of experiments were then rerun using a different recurrent cell each time. Having three fundamental recurrent cells (i.e., GRU, LSTM, and BiLSTM) as our benchmark, we initially conducted a total of six experiments, namely Exp. 1, Exp. 2 and Exp. 3 belonging to the first set of experiments; whereas Exp. 4, Exp. 5 and Exp. 6 belonging to the second set of experiments. An additional variant of Exp. 6, namely Exp. 6b (which also utilizes BiLSTMs) was also conducted for further analysis. The same hyperparameters were used in all experiments. Stochastic gradient descent with momentum (0.9) was used as the optimizer for all. The initial learning rate was set to 5×10^{-4}. All the aforementioned experiments were trained for 45 epochs. At each 15 epoch interval, the learning rate was dropped by a factor of 0.1.

5 Results

We commence with the results of the "$\frac{1}{2}$ - sep. ζ" experiments, which are presented as Exp. 1, Exp. 2, and Exp. 3 in Table 2. In all experiments, the mean absolute deviation (MAD) of the ζ errors obtained from the separate folds are averaged. For the "$\frac{1}{2}$ - sep. ζ" experiments, the model with BiLSTM recurrent cell performed the best, with a MAD-ζ error of 0.0645 for the test set, whereas the results for the models with GRU and LSTM cells exhibited a relatively poorer level of performance compared to models with BiLSTM. A MAD-ζ error of 0.0645 is, in practical terms, acceptable considering the tested ζ values do not exist in the training set, and their range is between 0.1 and 0.8. Table 2 also presents the MAD ζ error for the training set to aid evaluation of the overfitting extent that occurred during training.

The $\frac{1}{2}$ - mix. ζ" experiments, which are presented as Exp. 4, Exp. 5, and Exp. 6a in Table 2 exhibited better performance, when compared to Exp. 1, Exp. 2, and Exp. 3. Among these experiments, the model with BiLSTM recurrent cell, trained in Exp. 6a performed the best, with a MAD-ζ error of 0.021 for the test set, whereas the results for the models with GRU and LSTM cells performed relatively poorer. From this result, we can draw two main conclusions. First, if we train the system with all types of ζ, the model is capable of predicting the dynamical system parameter with a very high degree of accuracy. Second, BiLSTM always performs better when compared to both GRU and LSTM, denoting that context within a dynamical system sequence model should be classified in a bidirectional manner in time. We consider this to be a significant outcome of the experiments, considering that, to the best of our knowledge, there is no empirically stated emphasis on the benefits of utilizing BiLSTMs for dynamical systems identification problem in the literature.

The experiments in Table 2 were focused on the effect of using different ζ values during the training. In order to answer questions such as "What type of input should be fed into the system to better predict a dynamic parameter?" or "Which time interval of the I/O sequence is more effective in predicting the dynamic system parameter?," we conducted some additional experiments. In order to understand the performance provided by the experiments, we first analyzed sample cases presented in Fig. 3. For

Table 2 Mean absolute deviation (mad) of ζ prediction errors for all experiments

Experiment	Set	RNN Cell	Train. mad.	Test. mad.
Exp. 1	$\frac{1}{2}$ - sep. ζ	GRU	0.0200	0.0755
Exp. 2	$\frac{1}{2}$ - sep. ζ	LSTM	0.0240	0.0790
Exp. 3	$\frac{1}{2}$ - sep. ζ	BiLSTM	0.0145	0.0645
Exp. 4	$\frac{1}{2}$ - mix. ζ	GRU	0.0310	0.0325
Exp. 5	$\frac{1}{2}$ - mix. ζ	LSTM	0.0310	0.0325
Exp. 6a	$\frac{1}{2}$ - mix. ζ	BiLSTM	0.0200	0.0210
Exp. 6b	$\frac{1}{2}$ - mix. ζ (Step Input @ 3–6 s)	BiLSTM	–	0.0097
Exp. 7	$\frac{1}{2}$ - mix. ζ (150 ep. - Extended Dataset)	BiLSTM	0.014	0.015

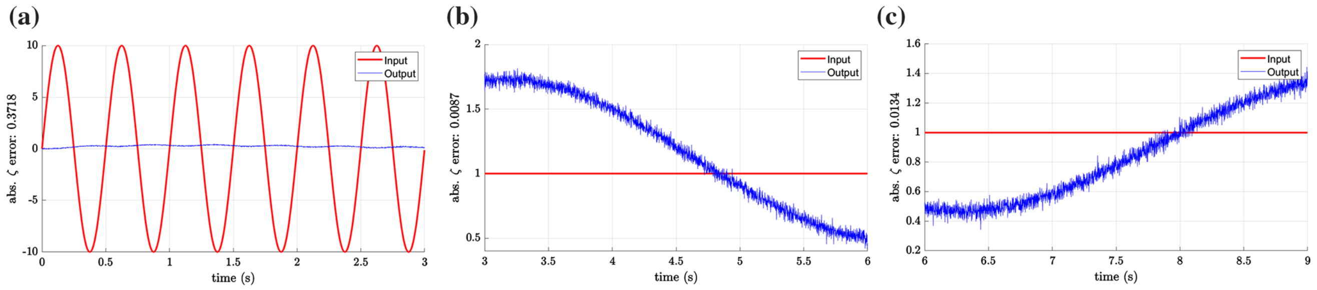

Fig. 3 Figures show sample I/O sequences, with ζ prediction errors provided in captions for each figure

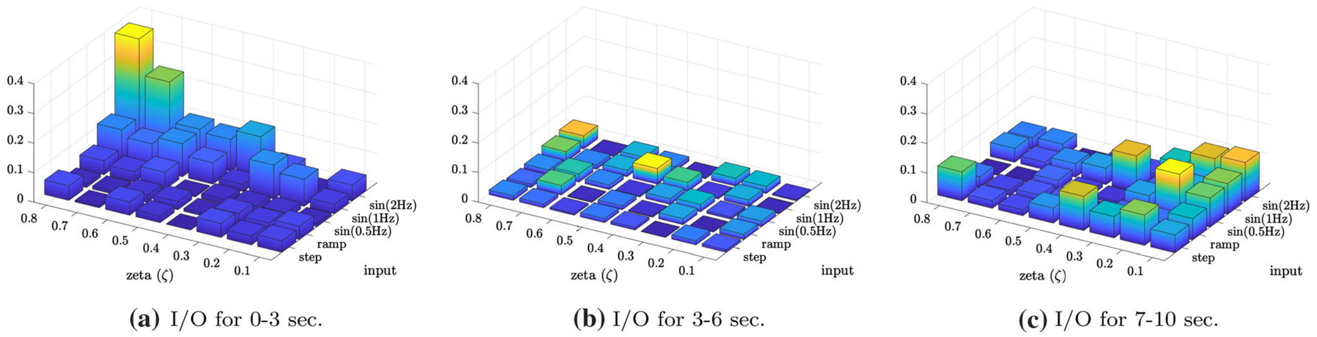

Fig. 4 The figures show 2D histograms of how much "**absolute ζ prediction error**" was obtained for different input types and different damping levels, using the Exp. 6 model

example, in Fig. 3a, a 2 Hz sinusoidal input to a system with ζ = 0.8 and an output response for 0–3 s can be seen. After feeding the depicted I/O sequence pair into the model obtained from Exp. 6 (which performed the best among Exp. 1–6), a ζ value of 0.4282 was predicted. When compared to the actual value of ζ = 0.8, this represented a very poor level of performance.

However, when we observe Fig. 3b (or similarly Fig. 3c), the performance seen was significantly better. For a step input to a system with ζ = 0.1 and using the output response of 3–6 s, the predicted ζ value (again using the model trained in Exp. 6) was 0.1087 (i.e., with an absolute error of 0.087). This achieved a considerably better result.

The varying performance of the sample cases demonstrates that the prediction performance relies upon the type of input used, or the time interval of the response of the system that is fed into the deep learning model. In order to analyze this variance, Fig. 4 shows 2D histograms of how much "**absolute ζ prediction error**" was obtained for different input types and different damping levels, using the model obtained in Exp. 6. The X-axis in these histograms presents the different input types, while the y-axis presents the different ζ values used in the testing. For example, Fig. 4a depicts the 2D histograms of absolute ζ error when 0–3 s of I/O sequence pairs were fed into the DRNN model; whereas, Fig. 4b, c depicts the corresponding 2D histograms when 3–6 s and 7–10 s of the I/O sequence pairs were fed in, respectively. For example, it can be seen from Fig. 4a that using a 2 Hz input, and trying to predict the parameter of a dynamic system with $\zeta = 0.8$ (i.e., highly dampened) in the 0–3 s interval was prone to a high degree of errors. We actually observed this in Fig. 3a, hence Fig. 4b shows that the best results (i.e., the lowest MAD-ζ errors) were obtained when 3–6 s of I/O sequence pairs were fed into the system. For this interval only, and for the step-input only, we calculated the MAD-ζ error of the entire test set for all ζ values using the Exp. 6 model and presented this result in another row of Table 2 as Exp. 6b. The MAD-ζ error for this interval was 0.0097, which was the best-case scenario in our "45-epoch" experiments.

It should also be noted here that the effect of a small ζ (lightly damped) was expected to be more easily distinguishable, since it would exhibit a larger magnitude in the oscillation. As the ζ was increased, the time responses should tend to be more similar to each other, and, hence, would make it harder for the deep learning algorithm to converge to the true value of ζ. This expectation was most notable, as shown in Fig. 3a, specifically when the input signal was a sinusoid at 2 Hz. Since the system is a second-order, low-pass filter, a sinusoid at 2 Hz will be attenuated more than a sinusoid at a lower frequency, i.e., the magnitude of the output signal would be smaller, and hence the signal-to-noise ratio would be smaller. A smaller signal-to-noise ratio was expected to cause problems for the deep learning algorithm.

5.1 Additional experiments using step inputs with varying magnitudes

After analyzing the results of the first two sets of experiments, and witnessing the relatively superior performance of BiLSTM-based models, we have carried out another experiment, with an extended dataset. The new dataset included additional step input signals of varying magnitudes, again sampled at 1 kHz. In addition to our original input set (i.e., a unit step input, a ramp input with a unit slope, and three sinusoids, which each had a magnitude of 10, and frequencies of 0.5, 1, and 2 Hz), we have added three step input sequences with magnitudes -1, $+10$ and -10. For each input, we again have damping factor, ζ, values ranging from $\{0.1, 0.2, \ldots, 0.8\}$. Similarly, ten seconds of data samples (i.e., $10\text{s} \times 1\text{Khz} = 10{,}001$) were generated for each input/damping pair. Hence, in total, using eight different input types, eight distinct ζ values, and 7001 overlapping 3-s sequences, the extended dataset included a total of 448,064 ($8 \times 8 \times 7001$) input/output sequence pairs and corresponding ζ values.

Similarly, to Exp. 4–6, the extended dataset was divided into two random non-intersecting sets for twofold cross-validation. The same hyperparameters were applied except for the fact that this new network was trained for 150 epochs, unlike the previous experiments that were all trained for 45 epochs. After the 45th epoch, the learning rate drop factor was manipulated manually for best performance. This experiment is referred to as Exp. 7 and can be seen in the final row of Table 2.

In Fig. 5, we present absolute ζ prediction error histograms for Exp. 7 model, similarly to Fig. 4, which was prepared using Exp. 6 model. When compared to Fig. 4, the error rates in Fig. 5 are dramatically better for all input types, including the additional input signals introduced in the extended dataset. The prediction errors are almost always under 0.1, except for the case, where sinusoidal inputs and their initial response (i.e., between 0 and 3 s.) are applied to a high friction model. As explained above, this is an acceptable result considering that high friction systems will inevitably lag the response of a high-frequency input.

Furthermore, we analyzed the prediction capabilities of the model trained in Exp. 7, against step inputs of varying magnitudes, including values that were not introduced during training. In Fig. 6, absolute ζ prediction error histograms for step inputs of varying magnitudes can be seen. In this figure, in addition to step inputs with unit, negative unit, -10 and $+10$ magnitudes that were used in training Exp. 7 model; -2,$+2$,-5,$+5$ magnitude step inputs are also tested. It is clearly seen in Fig. 6 that, although the second group of step inputs were not introduced during training, the model trained in Exp. 7 can successfully generalize different magnitude step input sequences and predict the related dynamical systems parameter with very high accuracy.

6 Conclusions and future work

The main goal of this paper was to find an effective deep recurrent neural architecture for the dynamical systems parameter identification task. For this purpose, three gated recurrent cells, namely GRU, LSTM, and BiLSTM, were comparatively experimented with. The results showed that at late transient, by feeding a step input into the system and by utilizing BiLSTM recurrent cells in the proposed 6-layered neural architecture, we achieved the lowest error rates in predicting the damping coefficient of a second-order linear dynamic system. The average test MAD for our best experiment (Exp. 7), which was trained longer with an extended dataset, was significantly better when

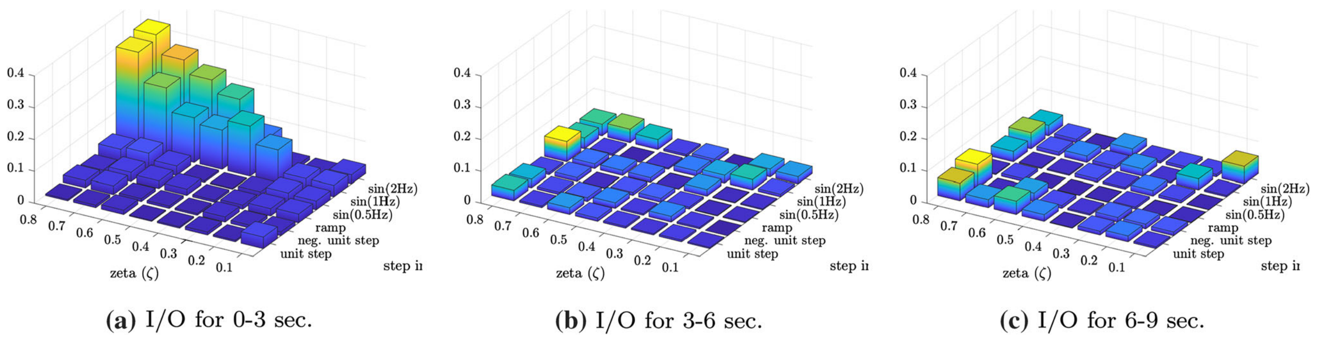

Fig. 5 The figures show 2D histograms of how much "**absolute** ζ **prediction error**" was obtained for different input types and different damping levels, using the **Exp. 7** model

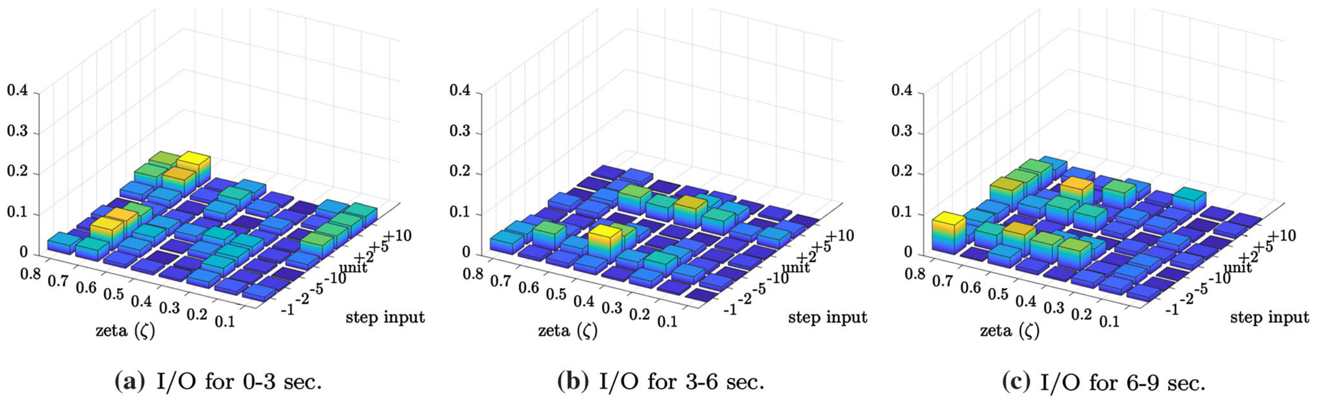

Fig. 6 The figures show 2D histograms of how much "**absolute** ζ **prediction error**" was obtained for different **step input** types and different damping levels, using the Exp. 7 model

compared to previous experiments; showing us that the models are open to further improvement in performance and, more importantly as we add new types of inputs, the machine learning model has the capacity to generalize them without any loss of accuracy.

We principally observe two significant outcomes in our experiments. First, BiLSTM cells with bidirectional gradient flow performed better than single-direction gated recurrent cells (GRU and LSTM in our case); thus showing that context within a dynamical system sequence model correlated in a bidirectional manner, when it comes to the systems identification problem. Second, through experimentation, we investigated at which exact instant and with what kind of input the system should be excited in order to obtain the best parameter identification results.

Future research directions point to a strong collaboration between the fields of automatic control systems engineering and deep learning. We believe that the more these fields work in partnership, the greater the potential impact that they will have in transforming each other's research directions. A good next step, therefore, would be to embed a deep learning-based parameter identification system, such as the one proposed in the current study, into the actual closed-loop of the dynamic system, thus providing parameter-aware automated control.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Andersson C, Ribeiro AH, Tiels K, Wahlström N, Schön TB (2019) Deep convolutional networks in system identification. In: 2019 IEEE 58th conference on decision and control (CDC), pp. 3670–3676. https://doi.org/10.1109/CDC40024.2019.9030219
2. Ayyad A, Chehadeh M, Awad MI, Zweiri Y (2020) Real-time system identification using deep learning for linear processes with application to unmanned aerial vehicles. IEEE Access 8:122539–122553. https://doi.org/10.1109/ACCESS.2020.3006277
3. Bekey GA (1970) System identification: an introduction and a survey. SIMULATION 15(4):151–166. https://doi.org/10.1177/003754977001500403
4. Box GEP, Jenkins GM (1970) Time series analysis: forecasting and control. Holden-Day

5. Brusaferri A, Matteucci M, Portolani P, Spinelli, S (2019) Non-linear system identification using a recurrent network in a bayesian framework. In: 2019 IEEE 17th international conference on industrial informatics (INDIN), 1: 319–324. https://doi.org/10.1109/INDIN41052.2019.8972113
6. Cheon K, Kim J, Hamadache M, Lee D (2015) On replacing PID controller with deep learning controller for DC motor system. J Autom Control Eng 3(6):452–456
7. Chiuso A, Pillonetto G (2019) System identification: a machine learning perspective. Ann Rev Control Robot Auton Syst 2(1):281–304. https://doi.org/10.1146/annurev-control-053018-023744
8. Cho K, van Merrienboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y (2014) Learning phrase representations using RNN encoder-decoder for statistical machine translation. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP). https://doi.org/10.3115/v1/d14-1179
9. Cleeremans A, Servan-Schreiber D, McClelland JL (1989) Finite state automata and simple recurrent networks. Neural Comput 1(3):372–381. https://doi.org/10.1162/neco.1989.1.3.372
10. Dinh H, Bhasin S, Dixon WE (2010) Dynamic neural network-based robust identification and control of a class of nonlinear systems. In: IEEE Conference on Decision and Control (CDC), pp. 5536–5541
11. Genc S (2017) Parametric system identification using deep convolutional neural networks. In: International joint conference on neural networks (IJCNN), pp. 2112–2119
12. Ghoraani B, Krishnan S (2011) Time-frequency matrix feature extraction and classification of environmental audio signals. IEEE Trans Audio Speech Lang Process 19(7):2197–2209
13. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: AISTATS
14. Gonzalez J, Yu, W (2019) Non-linear system modeling using lstm neural networks. IFAC-PapersOnLine. In: 2nd IFAC Conference on modelling, identification and control of nonlinear systems MICNON 2018. 51(13): 485 – 489 . https://doi.org/10.1016/j.ifacol.2018.07.326. http://www.sciencedirect.com/science/article/pii/S2405896318310814
15. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw 18(5):602–610. https://doi.org/10.1016/j.neunet.2005.06.042
16. Gu G, Khargonekar P (1992) A class of algorithms for identification in $\mathcal{H}_\infty$. Automatica 28:299–312
17. He K, Zhang X, Ren S, Sun, J (2015) Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. In: 2015 IEEE international conference on computer vision (ICCV). https://doi.org/10.1109/iccv.2015.123
18. Helmicki AJ, Jacobson CA, Nett CN (1991) Control oriented system identification: a worst-case/deterministic approach in $\mathcal{H}_\infty$. IEEE Trans Autom Control 36(10):1163–1176
19. Hermans M, Schrauwen B (2013) Training and analysing deep recurrent neural networks. In: Burges CJC, Bottou L, Welling M, Ghahramani Z, Weinberger KQ (eds) Advances in neural information processing systems, vol 26. Curran Associates Inc, New York, pp 190–198
20. Hilkert JM, Pautler B (2011) A reduced-order disturbance observer applied to inertially stabilized line-of-sight control. In: W.E. Thompson, P.F. McManamon (eds.) Acquisition, tracking, pointing, and laser systems technologies XXV, vol. 8052, pp. 114 – 125. International Society for Optics and Photonics, SPIE. https://doi.org/10.1117/12.884123
21. Hochreiter S (1991) Iuntersuchungen zu dynamischen neuronalen netzen. Master's thesis, Technische Universität München, Germany
22. Hochreiter S, Schmidhuber J, Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780
23. Wang Jeen-Shing, Chen Yen-Ping (2006) A fully automated recurrent neural network for unknown dynamic system identification and control. IEEE Trans Circuits Syst I Regul Pap 53(6):1363–1372
24. Kumar R, Srivastava S (2020) A novel dynamic recurrent functional link neural network-based identification of nonlinear systems using lyapunov stability analysis. Neural Comput Appl. https://doi.org/10.1007/s00521-020-05526-x
25. Lin CM, Tai CF, Chung CC (2014) Intelligent control system design for uav using a recurrent wavelet neural network. Neural Comput Appl 24(2):487–496. https://doi.org/10.1007/s00521-012-1242-5
26. Ljung L (1999) System identification: theory for the user. Prentice Hall, Hoboken
27. Marple SL (1987) Digital spectral analysis with applications. Prentice-Hall, Upper Saddle River
28. Mastorocostas PA, Theocharis JB (2002) A recurrent fuzzy-neural model for dynamic system identification. IEEE Trans Syst Man Cybern Part B 32(2):176–190. https://doi.org/10.1109/3477.990874
29. Miller R, Mooty G, Hilkert JM (2013) Gimbal system configurations and line-of-sight control techniques for small UAV applications. In: D.J. Henry, D.A. Lange, D.L. von Berg, S.D. Rajan, T.J. Walls, D.L. Young (eds.) Airborne intelligence, surveillance, reconnaissance (ISR) systems and applications X, vol. 8713, pp. 39–53. International Society for Optics and Photonics, SPIE. https://doi.org/10.1117/12.2015777
30. Mohajerin N (2012) Identification and predictive control using recurrent neural networks. Master's thesis, Öreburo University, Department of Technology, Sweden
31. Nair V, Hinton GE (2010) Rectified linear units improve restricted Boltzmann machines. ICML'10, pp. 807–814. Omnipress, Madison, WI, USA
32. Narendra KS, Parthasarathy K (1990) Identification and control of dynamical systems using neural networks. IEEE Trans Neural Netw 1(1):4–27
33. Natke HG (1982) Computational methods and experimental measurements, chap: survey on parameter estimation within system identification using a priori knowledge of system analysis. Springer, Berlin, pp 17–27
34. Ogunmolu O, Gu X, Jiang, S, Gans N (2016) Nonlinear systems identification using deep dynamic neural networks
35. Pascanu R, Gulcehre C, Cho K, Bengio Y (2014) How to construct deep recurrent neural networks. In: Proceedings of the second international conference on learning representations (ICLR 2014)
36. Pascanu R, Mikolov T, Bengio Y (2013) On the difficulty of training recurrent neural networks. ICML'13, pp. III–1310–II–1318
37. Passalis N, Tefas A (2020) Continuous drone control using deep reinforcement learning for frontal view person shooting. Neural Comput Appl 32(9):4227–4238. https://doi.org/10.1007/s00521-019-04330-6
38. Pearlmutter (1989) Learning state space trajectories in recurrent neural networks. In: International 1989 joint conference on neural networks. 2: 365–372
39. Pham DT, Liu X (1995) Neural networks for identification, prediction and control, chap. Dynamic system identification using recurrent neural networks, pp. 47–61. Springer, London
40. Rake H (1980) Step response and frequency response methods. Automatica 16:519–526
41. Åström KJ, Bohlin T (1965) Numerical identification of linear dynamic systems from normal operating records. In: IFAC symposium on self-adaptive systems

42. Åström KJ, Eykhoff P (1971) System identification-a survey. Automatica 7(2):123–162. https://doi.org/10.1016/0005-1098(71)90059-8
43. Richard A, Mahé A, Pradalier C, Rozenstein O, Geist M (2019) A Comprehensive benchmark of neural networks for system identification. https://hal.archives-ouvertes.fr/hal-02278102. Working paper or preprint
44. Richard A, Mahé A, Pradalier C, Rozenstein O, Geist M (2019) A comprehensive benchmark of neural networks for system identification. Tech. Rep. hal-02278102f, HAL archives-ouvertes
45. Rubio JJ, Yu W (2007) Nonlinear system identification with recurrent neural networks and dead-zone Kalman filter algorithm. Neurocomputing. 70(13): 2460–2466. https://doi.org/10.1016/j.neucom.2006.09.004. http://www.sciencedirect.com/science/article/pii/S0925231206003134. Selected papers from the 3rd International Conference on Development and Learning (ICDL 2004) Time series prediction competition: the CATS benchmark
46. Schüssler M, Münker T, Nelles O (2019) Deep recurrent neural networks for nonlinear system identification. In: 2019 IEEE symposium series on computational intelligence (SSCI), pp. 448–454 . https://doi.org/10.1109/SSCI44817.2019.9003133
47. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(56):1929–1958
48. Sutskever I (2013) Training recurrent neural networks. Ph.D. thesis, CAN
49. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucke V, Rabinovich A (2015) Going deeper with convolutions. 2015 IEEE conference on computer vision and pattern recognition (CVPR) pp. 1–9
50. Tavoosi J, Badamchizadeh MA (2013) A class of type-2 fuzzy neural networks for nonlinear dynamical system identification. Neural Comput Appl 23(3):707–717. https://doi.org/10.1007/s00521-012-0981-7
51. Wang Y (2017) A new concept using LSTM neural networks for dynamic system identification. In: American control conference (ACC), pp. 5324–5329 (2017)
52. Wellstead WE (1981) Non-parametric methods of system identification. Automatica 17:55–69
53. Werbos PJ (1990) Backpropagation through time: what it does and how to do it. Proc IEEE 78(10):1550–1560