Powder Technology 360 (2020) 921–936

Contents lists available at ScienceDirect

Powder Technology

journal homepage: www.elsevier.com/locate/powtec

Deep learning based system identification of industrial integrated grinding circuits

Srinivas Soumitri Miriyala, Kishalay Mitra *

Global Optimization and Knowledge Unearthing Laboratory, Department of Chemical Engineering, Indian Institute of Technology Hyderabad Kandi, Telangana, 502285, India

ARTICLE INFO

Article history:
Received 2 May 2019
Received in revised form 28 September 2019
Accepted 15 October 2019
Available online 25 October 2019

Keywords:
Grinding circuit
System identification
Deep learning
LSTM networks
Wavelets
Multi-objective evolutionary optimization

ABSTRACT

Energy efficiency and maximum productivity in ore beneficiation processes can be ensured when integrated grinding circuits function in an optimal fashion. The complexity of first principles based models prevents online implementation of control and optimization algorithms, thus, creating the need for the development of accurate data-based models. In this work, deep recurrent neural networks (DRNNs) are implemented for nonlinear system identification of 3 input 6 output integrated grinding circuit from an industrial lead-zinc ore beneficiation set-up. Optimal long short term memory networks (LSTMs) with maximum predictability are obtained by solving a novel multi-objective framework for DRNN architecture design. The optimal LSTMs are trained and validated on pseudo random binary sequence (PRBS) signal with an accuracy of 99%, and tested successfully on unseen random Gaussian sequence (RGS) signal. Comprehensive comparison with conventional tools for nonlinear system identification, such as wavelet networks, is performed to show the efficacy of proposed optimal LSTMs.

1. Introduction

Grinding is considered an important unit operation in many mining, metallurgical and mineral processing industries. This is mainly due to its energy and cost-intensive nature in the entire ore beneficiation process. Grinding is equally important for operational reasons since it has to deliver quality products to several significant unit operations, such as floatation, and other downstream unit processes. Modeling, optimization, and control of integrated grinding circuits are, thus, among the key research areas towards minimizing energy consumption while simultaneously improving as well as maintaining product quality in ore beneficiation industries [1].

Modeling of grinding circuits typically needs to consider several milling operations such as ball mill, rod mill, etc. for size reduction, sumps for collection and smooth flow of slurry and hydro-cyclones for separation of particles of different sizes. The collective effort of several researchers over the years has brought modeling of industrial grinding circuits to a reasonable state of robustness [2–7]. The majority of these models were realized from the combination of physical laws of conservation and empirical correlations. Use of empiricism was necessary for many models as the actual physics behind the phenomena is not very clear [8]. Though these models were quite useful as initial attempts, they were valid only in the operational regimes from which the data was collected for estimating the empirical constants [9]. A radical step in the evolution of models for the grinding operation was the introduction of population balance based methods [10–12]. Instead of considering the entire population of particles, they are divided into subpopulations of different sizes and then tracked through mass balance and transition of material across classes of sizes, reflecting breakage phenomenon involved in the mills. Improvement in computational power allowed modeling using various time expensive techniques such as computational fluid dynamics (CFD) [13,14] and discrete element methods (DEM) [15–17]. These models focused on particle-particle interactions in grinding operation. Though significant efforts were also made in studying the effect of hydro-cyclones in grinding circuits, the use of correlations for modeling hydro-cyclones was found very useful [18]. These efforts, however, made way for modeling integrated grinding circuits as a whole using the combination of CFD, population balance based approaches and empirical correlations [2–5 and 18–20].

In addition to functioning as soft-sensors, these models can be integrated with optimization algorithms to facilitate the desired optimal operation of the entire grinding circuit. Optimization and control of integrated grinding circuits have, therefore, seen considerable progress using these hybrid models. Development of strategies for controlling grinding circuits has resulted in the identification of a set of state variables, manipulated variables and objectives in grinding operation, which are implemented globally [1,5]. To name a few, water flowrate into sumps, mills, and flowrate of solids into mills are recognized as manipulated variables, while well-recognized state variables are particle size distribution in the product stream, mill load, levels of slurry in sumps, cyclone overflow, product density, etc. In this regard,

* Corresponding author.
E-mail address: kishalay@iith.ac.in (K. Mitra).

https://doi.org/10.1016/j.powtec.2019.10.065
0032-5910/

maximization of productivity, minimization of recirculation load, minimization of mid-size fraction could be among the popular objective functions considered by the researchers.

While adopting an integrated modeling approach, model prediction for each unit operation needs validation against data from each operation. In most of the industrially operated grinding circuits, hardware sensors are placed only in inlet and outlet flow streams due to several practical constraints, such as cost, etc. Lack of sensors in intermediate locations results in the absence of data for validating such models. Another challenge is the staggering computational cost involved in simulating first principles based models of grinding circuit built using methods such as CFD and DEM. Control strategies require models which are generally fast in execution, to present real-time solutions. This brings the necessity of constructing an end-to-end model mapping the inputs and the outputs of a grinding plant. One efficient solution to these problems is building data based dynamic surrogate models for integrated grinding circuits [21]. These models get trained only on data collected from inlet and outlet streams of the grinding circuit and result in multiple-input-multiple-output (MIMO) system identification models [22,23].

Modeling nonlinear dynamics in data has always been of special interest to researchers working in the field of process systems engineering (PSE). With the growth in development of various machine learning algorithms in the recent past, the interest has shifted towards studying tools such as recurrent neural networks (RNNs) and developing deep learning (DL) based methodologies to compete with prominent system identification tools, such as wavelet networks to model time series data [21–23]. Although RNNs were used for modeling sequential data since a few decades, their success in system identification was limited due to their inability to capture significantly long time delays in process data. The combination of high sampling rates due to efficient sensors and slow dynamics of the process leads to the creation of long term dependencies in sampled data. RNNs fail to capture the trend in such cases due to the problem of vanishing gradients [24]. The current manuscript presents DL based system identification of integrated grinding circuits using long short term memory (LSTMs) networks [24]. LSTMs are deep recurrent neural networks (DRNNs), which are capable of extracting features shared over a long range of time dynamics and then perform the task of machine learning (ML), such as regression in sequential data [25–27]. However, the drawbacks of adopting LSTM as a system identification tool for time series data are:

a) Design of LSTM networks is heuristic thereby making their implementation tedious and ineffective [27].
b) LSTMs also suffer from other problems such as the choice of optimal hyper-parameters for training, overfitting, and instability [28–30].
c) LSTMs are difficult to model and their performances are heavily driven by the experience of users.

Highlighting the above knowledge gap, the authors plan to provide remedies to some of the aforementioned problems associated with LSTMs while focusing on system identification of integrated grinding circuits through this work. The novel contributions are briefly described below:

1. Issues associated with DRNNs, such as heuristic tuning of architecture and activation function and vanishing gradients are listed and an attempt has been made to resolve these by a novel algorithm for the optimal design of LSTMs.
2. The proposed framework is a multi-objective optimization algorithm designed for estimation of parsimonious LSTM networks with maximum generalization ability, thereby making an effort to minimize overfitting along with the minimization of prediction error.
3. A comprehensive comparison of optimal LSTM network is made with the state-of-the-art wavelet networks for system identification of grinding circuits to observe pros and cons.

The rest of the paper is divided into several sections. The formulation section describes the process of the integrated grinding circuit and an industrially validated first principles based model which is utilized to generate the time series data needed for system identification. This is followed by a detailed explanation of LSTM networks and the justification for using them in emulating grinding circuits in the formulation section. The proposed novel multi-objective framework for constructing optimal LSTMs is described next before proceeding to the results section to share the observations. Finally, the findings and the scope of future work are summarized in the conclusion section.

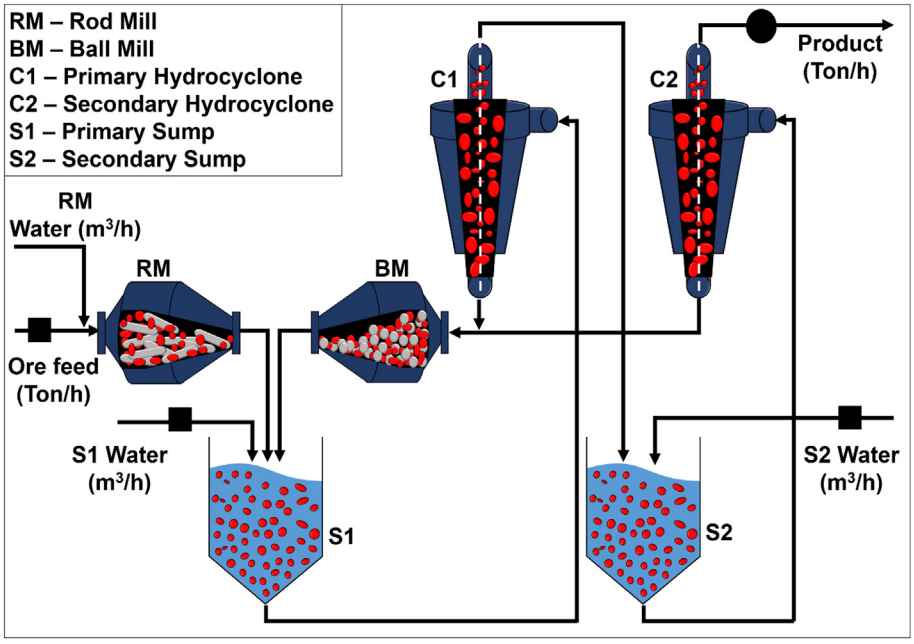

Fig. 1. Process flow diagram of industrial integrated grinding circuit in lead-zinc ore benefciation process consisting of two sumps, rod mill, ball mill and two hydro-cyclones. Data can be collected only through three inlet streams and one outlet stream, which are identified by black squares and black circle, respectively.

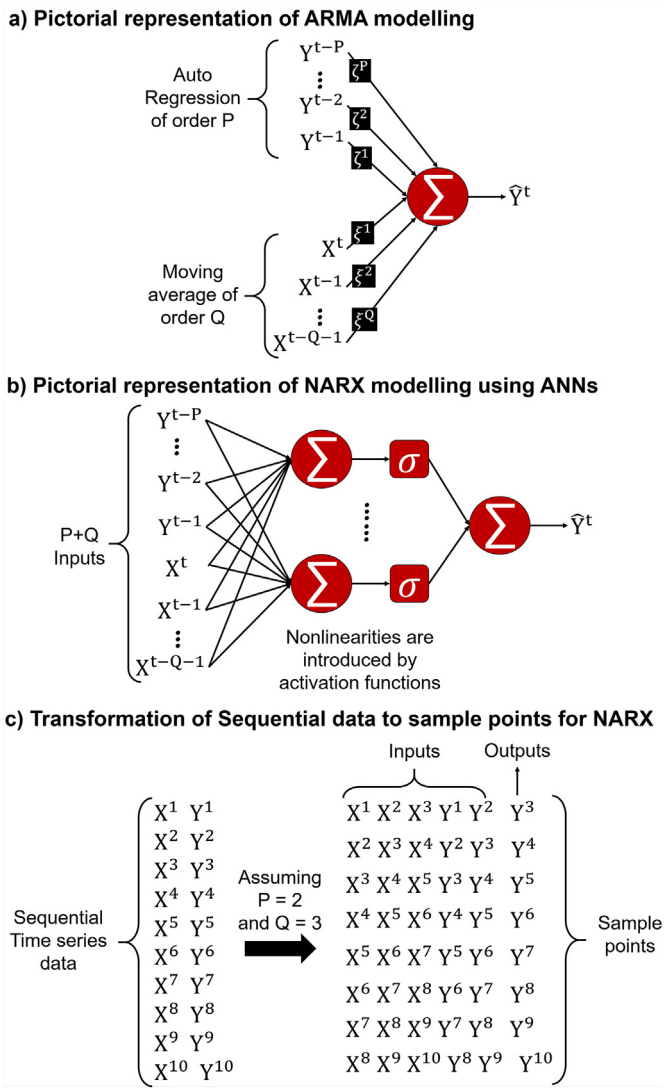

Fig. 2. Conventional modeling of time series using ARMA (Fig. 2a) and NARX using a feedforward ANN (Fig. 2b). The creation of input-output data tuples from sequential data is shown in Fig. 2c.

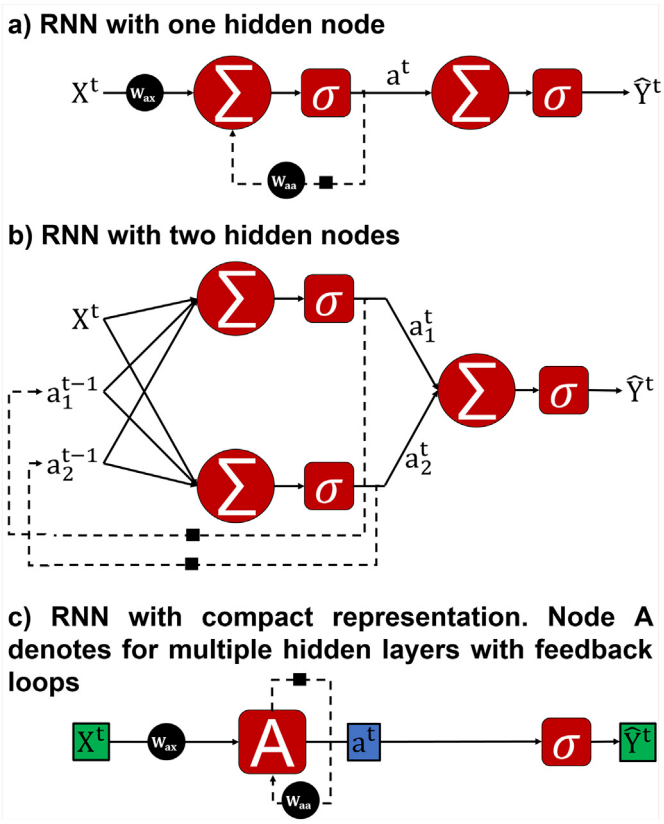

Fig. 3. A recurrent neural network contains feedback loops on each node which allows state regression. This architecture of RNN makes them an efficient tool to model sequential data. The summation unit will be required in Fig. 3c, if there are more than one hidden states.

2. Formulation

2.1. Industrial grinding circuit

The industrial grinding operation considered in this work is a part of lead-zinc ore beneficiation process, which is carried out in two phases – grinding and floatation. Comminution of ore to finely ground particles is performed in the conventional wet grinding circuit to separate the valuables from gangue particles. They are then selectively floated in froth floatation units to extract lead and zinc.

The wet grinding circuit, shown in Fig. 1, consists of one rod mill in open loop, one ball mill in a closed loop and a primary and secondary hydro-cyclone for carrying out two-stage separation. Raw ore from mines is fed to rod mill along with some amount of water. The discharge slurry from the rod mill is accumulated in the primary sump. It is then fed to primary hydro-cyclone whose underflow goes into ball mill while the overflow is collected in a secondary sump. The discharge slurry from the ball mill is mixed with the slurry from rod mill in the primary sump, thereby completing the circuit of ball mill. The slurry from the secondary sump is sent into secondary hydro-cyclone whose underflow is taken back to ball mill and overflow is declared as the final product from grinding circuit. To maintain the desired specific gravity of slurry in both the sumps, water is added to primary and secondary sumps. The product stream from grinding circuit is directly sent into the floatation unit, thus signifying the importance of integrated grinding circuit in ore beneficiation process. The hardware sensors in inlet and outlet streams, which sample the data, are indicated by black squares and circles, respectively, in Fig. 1.

In Fig. 1, there are three inputs, which can be manipulated to control the grinding operation. They are ore feed rate to rod mill and water flow rates to primary and secondary sumps. Since ball mill and primary hydro-cyclone are in closed loops, the only output stream originating from the integrated circuit is the overflow from secondary hydro-cyclone. To measure the quality of product from the grinding circuit, certain properties of overflow slurry from secondary hydro-cyclone are measured. Six such properties, known as key performance indicators (KPIs), are considered in the current study. They are throughput (an indicator of the productivity of the plant), recirculation load (an indicator of energy consumed by the plant), percentage of solids and fraction of three size classes - coarse, mid and fine sizes (an indicator of the quality of the product) present in the outlet stream. Therefore, a set of tuples, each consisting of 3 inputs and 6 outputs as mentioned above, constitutes the dataset used for system identification of integrated grinding circuit in this work. Since this data is sampled during the dynamic operation of the plant, it is safe to assume that the collected data is a MIMO time series.

System identification tools often require large volumes of data sampled at different frequencies from all operational regimes. Collection of such voluminous data from industry is in general unaffordable for many reasons such as safety, stability, and economics of the plant operation. The approach taken here is to use an industrially validated first principles based model (thoroughly across all the regimes) [20] and simulate the same to construct the data for system identification. The

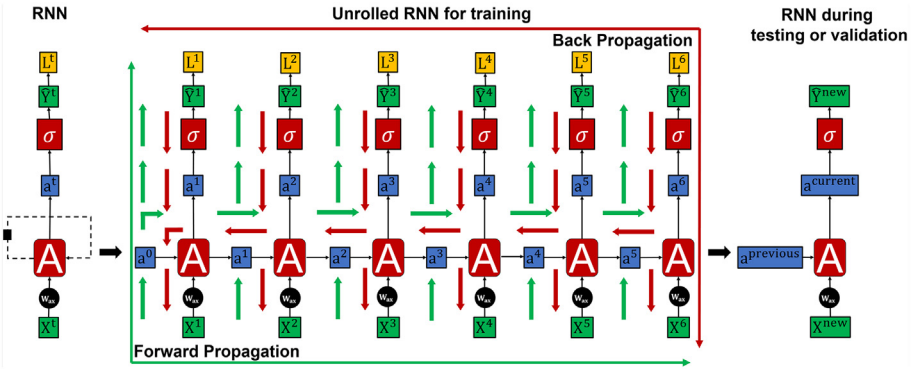

Fig. 4. Pictorial representation of RNN, unrolling of RNN during the training phase and RNN during the prediction phase. If the length of the sequence is T, the RNN is unrolled for T timestamps, however, the weights at every timestamp remain the same. In this figure, T = 6.

prediction from this model matches the plant data quite well [20] and can, therefore, be considered as a close mimic of the original plant in the purview of this work. Each unit operation in the circuit is modelled individually and a connectivity matrix, expressed in terms of binary variables 1 and 0, connects all of them to simulate the whole circuit. Population balance equations are written to model each size fraction in mills and sumps. Selection functions used for modeling the mills are assumed to follow n^{th} order kinetics where n is determined from industrial data. Similarly, the empirical constants present in equations for modeling hydro-cyclones are also determined from industrial data. Population balance equations combined with the empirical laws result in a set of differential algebraic equations (DAE) which are solved using the open source software DASSL [31] in Fortran 90 environment. Readers can refer to Ref. [8, 20] for elaborate details of the simulator used in this work. Given a time series of 3 inputs, the simulator generates a time series of 6 outputs. This data set is used to train LSTMs whose working is explained next.

2.2. Sequential data modeling using LSTM networks

2.2.1. LSTM block

Assuming each tuple (3 input and 6 output data point) to be associated with a timestamp, if the outputs at timestamp t depend not only on current inputs, but also on data tuples at previous timestamps, then such data is called sequential data. Time series is one example of sequential data. For the given timestamp t, let the input be denoted by $X^t = [X_1^t \quad X_2^t \quad \ldots X_n^t \ldots \quad X_N^t]$ and output be denoted by $Y^t = [Y_1^t \quad Y_2^t \quad \ldots Y_k^t \ldots \quad Y_K^t]$. N and K are dimensions of inputs and outputs; for the considered grinding example, $N = 3$ and $K = 6$. To improve the readability, the discussion in this section assumes N and $K = 1$, thereby, making the tuple (X^t, Y^t), a pair of input-output data.

The way of modeling a time series, $(X^t, Y^t)\ \forall\ t = 1$ to T is to build an estimate, $\hat{Y}^t(\zeta, \xi)$, of the output and minimize the error between the estimate and ground truth Y^t, by fine-tuning the set of parameters ζ and ξ over all timestamps. A simple yet, powerful expression which is conventionally used to model linear time series is, auto-regressive moving average (ARMA), as shown in Eq. (1). Here the values P and Q, called the orders of auto-regression and moving average, respectively, are usually fixed a priori and the estimate is made by a linear combination of the output regressing on its own values from previous time steps and moving average of inputs. The idea is presented in Fig. 2a. $\hat{Y}^t$ is a function of the tunable parameters in the model - (ζ, ξ) and set of previous inputs and outputs. However, the notation in left-hand side of Eq. (1) includes only the tunable parameters in the parenthesis to highlight the importance of optimizing the set of parameters ζ and ξ for minimizing the error between $\hat{Y}^t$ and Y^t by the process of training. The set of parameters ζ and ξ becomes nonlinearly involved when the estimate is obtained

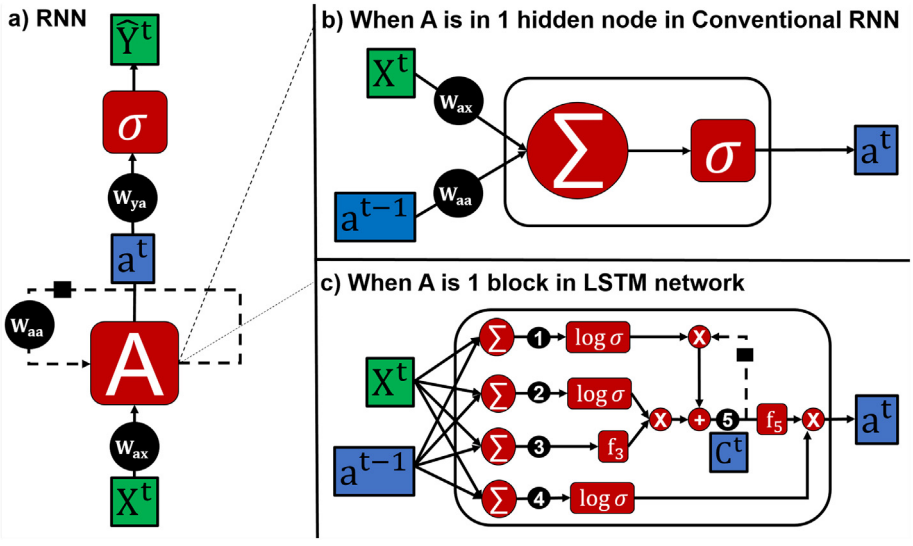

Fig. 5. LSTM networks are a type of recurrent neural networks (Fig. 5a) where the conventional node (Fig. 5b) is replaced by a complex architecture called LSTM block (Fig. 5c). The nodes 1, 2, 3 and 4 indicate the forget gate, input gate, cell gate and output gate, respectively.

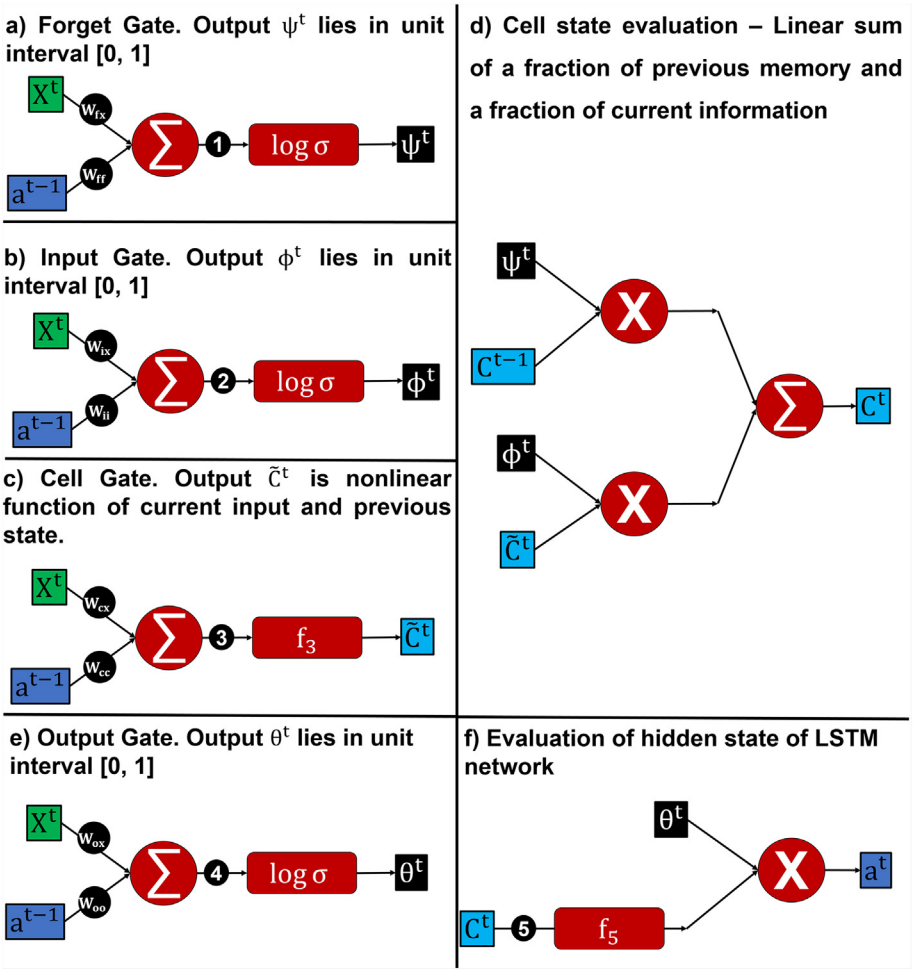

Fig. 6. Graphical representation of forget gate (Fig. 6a), input gate (Fig. 6b) and cell state evaluation (Fig. 6c and 6d), output gate (Fig. 6e) and hidden state evaluation (Fig. 6f) in an LSTM block.

using Nonlinear ARMA [32].

$$\hat{Y}^t(\zeta, \xi) = \sum_{i=1}^{P} \zeta^i Y^{t-i} + \sum_{j=1}^{Q} \xi^j X^{t-j+1} \tag{1}$$

Another popular model is nonlinear auto regressive exogenous inputs (NARX), where the inputs are considered to be exogenous variables. The perspective here is to observe that an entire sequence of T timestamps is now represented by the set of parameters ζ and ξ. Interestingly, this is also achieved by considering a multilayer perceptron network and sending the P number of AR terms and Q number of MA terms as inputs to predict the output at current timestamp t. This representation is depicted in Fig. 2b. A significant concern in such type of modeling is that the notion of sequence in data is represented using sample points obtained by heuristically determined orders P and Q, as shown in Fig. 2c. Due to the nonlinearity in data, the extent of dependency does not remain constant over the entire dataset. Therefore, while modeling the time series data, the length of dependency on past (P and Q) should not be fixed heuristically.

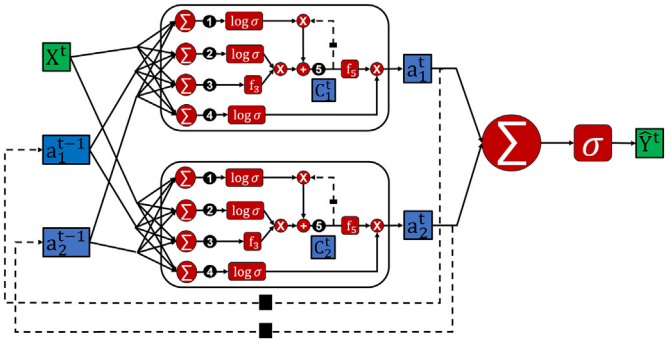

Fig. 7. Diagram of an LSTM network with 1 input, 2 hidden nodes and 1 output.

RNNs on the other hand make no such assumptions about the extent of dependency, but they learn it from data [33]. Instead of regressing on the outputs, the neural networks regress on a hidden state a^t. The hidden state is network activated output from a node, which can be interpreted as a latent feature vector representation of time series data. A feedback loop on the node allows for the state regression as shown in Fig. 3a, where the RNN has one hidden node and one output node whose activations are given by σ_1 and σ_2, respectively. The state regression equations for configuration in Fig. 3a are given in Eqs. (2) and (3), where the notation w_{ij} stands for weight on the connection from i to j. The parameters in the network are the set of weights w_{ij} and biases b_i.

$$a^t = \sigma_1\left(w_{aa}a^{t-1} + w_{ax}X^t + b_1\right) \tag{2}$$

$$\hat{Y}^t = \sigma_2\left(w_{Ya}a^t + b_2\right) \tag{3}$$

An example with 2 hidden nodes is shown in Fig. 3b, while a compact form is presented in Fig. 3c, where node A represents a generic

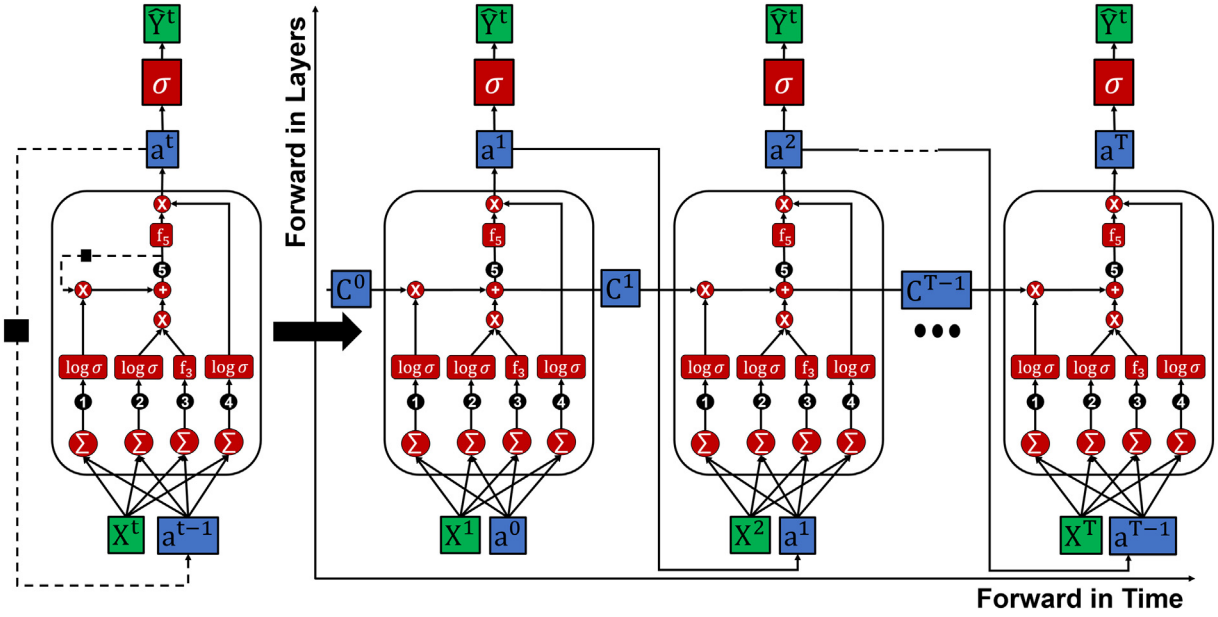

Fig. 8. Unrolling of an LSTM network with 1 input, 1 hidden node and 1 output over the entire time series T.

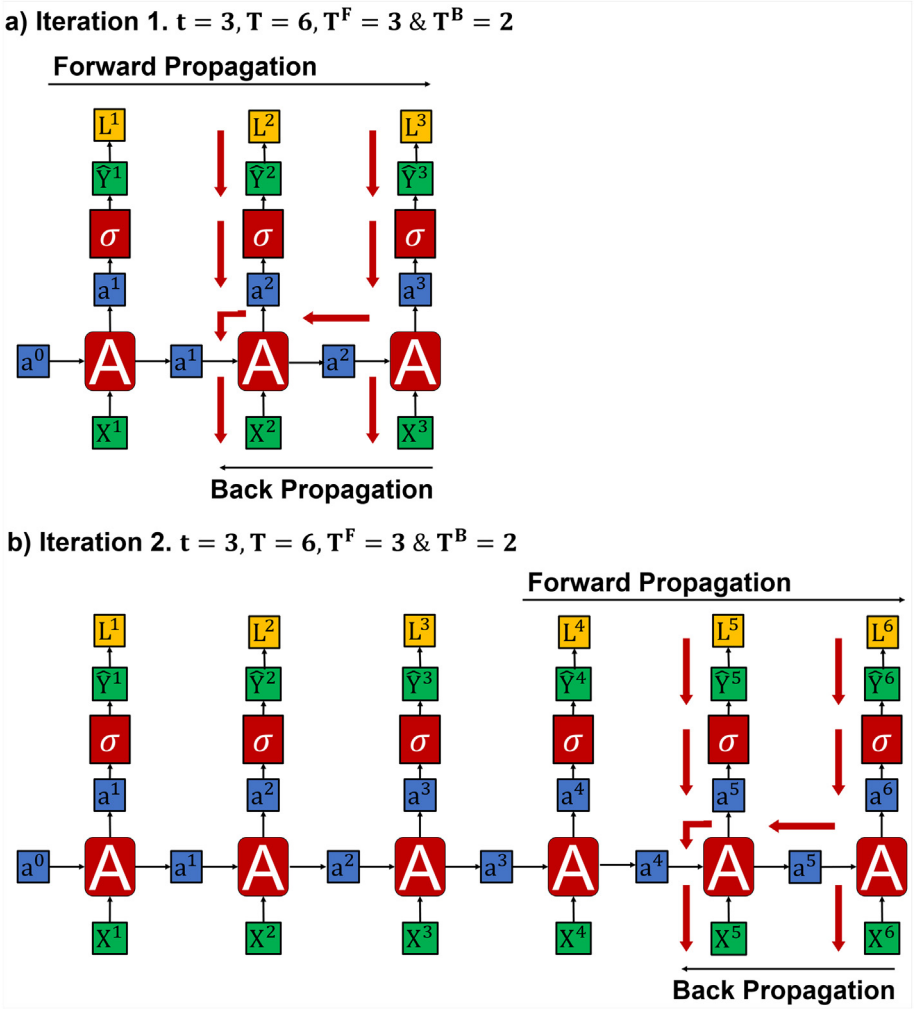

Fig. 9. Training of an RNN using t-BPTT. In the compact representation, the node A can be either the conventional RNN node or an LSTM block.

Table 1
Algorithm for training LSTM networks using Adam and t-BPTT.

Algorithm-1: Training LSTM networks

Fix the hyper-parameters: architecture, learning rate α, training size T, truncated-BPTT parameters T^B and T^F, number of epoch Q, Adam parameters ρ_1 and ρ_2.
Standardize: training data to zero mean and unit variance
Initialize: weights in the network with random values ~ 0, iteration counter: $p = 1$, $\Delta = 1e^{-10}$ for numerical stability, initial Adam updates: $\boldsymbol{\nu}^0 = \mathbf{r}^0 = 0$
for q→1 to Q:
  for t→1 to T:
    Forward propagate LSTM network for given timestamp t.
    if t is divisible by T^F:
      Perform BPTT from t to t− T^B by unrolling the network for T^B timestamps:
        Find $\nabla_{\mathbf{w}}\mathcal{L}$: gradient of loss $\mathcal{L}$ with respects to weights
      Perform weight update using Adam:
        $\boldsymbol{\nu}^{\mathbf{p}} = \rho_1\boldsymbol{\nu}^{\mathbf{p}-1} + (1-\rho_1)\nabla_{\mathbf{w}}\mathcal{L}$
        $\mathbf{r}^{\mathbf{p}} = \rho_2\mathbf{r}^{\mathbf{p}-1} + (1-\rho_2)\nabla_{\mathbf{w}}\mathcal{L}\odot\nabla_{\mathbf{w}}\mathcal{L}$
        $\bar{\boldsymbol{\nu}}^{\mathbf{p}} = \frac{\boldsymbol{\nu}^{\mathbf{p}}}{1-\rho_1^p}$ and $\bar{\mathbf{r}}^{\mathbf{p}} = \frac{\mathbf{r}^{\mathbf{p}}}{1-\rho_2^p}$
        $\Delta\mathbf{w} = \frac{-\alpha}{\sqrt{\Delta + \bar{\mathbf{r}}^{\mathbf{p}}}}\odot\bar{\boldsymbol{\nu}}^{\mathbf{p}}$
        $\mathbf{w}\leftarrow\mathbf{w} + \Delta\mathbf{w}$.
      Increment the iteration counter: $p = p + 1$
    end if loop
  end for loop of t.
  Increment the epoch counter: $q = q + 1$
end for loop of q.

architecture of hidden layers. The number of output nodes are the same as the number of outputs in the network and often, the output activation σ_2 is considered linear. Although RNNs have an architectural advantage over NARX models in terms of feedback loops, the weights in the regression expression, Eq. (2), remain time invariant similar to NARX models.

RNNs are trained by unrolling them as shown in Fig. 4. The unrolled architecture resembles a deep feed-forward neural (FFN) Network. However, there are two differences between an unrolled RNN and a deep FFN network. Unlike the deep FFN network, the unrolled RNN has two dimensions. One is forward in layers (vertical direction in Fig. 4) and the other is forward in time (horizontal direction in Fig. 4). RNN classifies as a deep neural network because of the depth in direction which is forward in time. For example, if the length of the sequence is say 1000, the RNN network will be unrolled for 1000 timestamps. At each timestamp, there exists the same feed forward architecture (vertical direction in Fig. 4). The second difference between the unrolled RNN and deep FFN network is that, *when the RNN is unrolled, the weights in the feedforward network at every timestamp remain same.* The weights in the RNN are trained by an algorithm, where the error signal is back propagated both in time and feedforward layers, thereby the name, back propagation through time (BPTT) [34]. A more rigorous approach towards RNNs and their training is presented in the supplementary material.

LSTM is a type of RNN network. However, unlike the conventional RNNs, the fundamental unit in LSTM network is an LSTM block, which contains four nodes designed to perform specific functions culminating into regression. These nodes regulate the information that passes through them and hence, they are more commonly known as gates - forget gate, input gate, cell gate, and output gate, named according to their functionalities [23,27]. The LSTM block is depicted in Fig. 5. The functioning of an LSTM block is centered on a hypothetical state of the block called cell state C^t. Cell state is the self-recurring unit of LSTM block, as shown in Eq. (4).

$$C^t = \psi^t\, C^{t-1} + \varphi^t\, \tilde{C}^t \tag{4}$$

It is evaluated as a weighted average of its own value from the previous timestamp and an intermediate state $\tilde{C}^t$, which is a representative of information at current timestamp. *However, unlike RNNs, the weights, ψ^t and φ^t, in regression expression* Eq. (4) *are functions of timestamps. When compared to weights in* Eq. (2) *which are estimated and remain constant, the weights in* Eq. (4) *are generated at every timestamp using the forget and input gates.* The output from forget gate determines what fraction of C^{t-1} should be considered for evaluating C^t, that is, how much information from the previous timestamp should be forgotten. Similarly, the output from the input gate decides how much information from the current timestamp should be considered for evaluating C^t. LSTM block, therefore, learns to regulate the quantities of previous memory and current inputs to generate the current outputs. These gates are described in Eqs. (5) and ((6), respectively, and pictorially represented in Fig. 6a and 6b. Log sigmoid activation (see Eq. (7)) ensures that the weights ψ^t and φ^t lie in the unit interval [0, 1].

$$\psi^t = \mathrm{logsig}\left(w_{fx}X^t + w_{ff}a^{t-1} + b_f\right) \tag{5}$$

$$\varphi^t = \mathrm{logsig}\left(w_{ix}X^t + w_{ii}a^{t-1} + b_i\right) \tag{6}$$

$$y = \frac{1}{(1 + \exp(-x))} \tag{7}$$

Here, x is arbitrary scalar and y is logsigmoid activation of x

The nonlinear function, which maps the current inputs with $\tilde{C}^t$ constitutes the cell gate, as shown in Eq. (8) and presented in Fig. 6c, while cell state evaluation (see Eq. (4)) is presented in Fig. 6d.

$$\tilde{C}^t = f_3\left(w_{cx}X^t + w_{cc}a^{t-1} + b_c\right) \tag{8}$$

$$\theta^t = \mathrm{logsig}\left(w_{ox}X^t + w_{oo}a^{t-1} + b_o\right) \tag{9}$$

$$a^t = \theta^t f_5\left(C^t\right) \tag{10}$$

Finally, the network activated output of LSTM block, also called the hidden state a^t, is evaluated as a nonlinear function of cell state C^t. However, the magnitude of this output is further squashed by passing it through an output gate. Eq. (9) and Fig. 6e represents the evaluation of output gate, while the determination of hidden state of LSTM network a^t is presented in Eq. (10) and Fig. 6f. Other than this difference, LSTM networks are similar to conventional recurrent neural networks – there exist an input layer, a set of hidden layers with LSTM blocks in place of nodes and an output layer with conventional nodes having no feedback, as shown in Fig. 7.

The training of LSTM networks is performed using the BPTT algorithm, where the LSTM network is unrolled over the entire length of the sequence and error is back propagated through the unrolled network both in time and feedforward layers to evaluate the gradients of loss function [34]. The unrolled LSTM network is presented in Fig. 8.

Every time, the BPTT algorithm is run, weights are updated, thereby completing one iteration. In practice, due to the large length of sequences, the computational cost associated with one such iteration becomes prohibitively expensive. For instance, in datasets where number of tuples is around 1000, BPTT translates into performing forward and backward pass in time for 1000 layered perceptron network for one iteration of weight update. A more practical approach is to reduce the extent to which error signal is back propagated in the network and perform the weight update. In this version of BPTT, called the truncated-BPTT (t-BPTT) [33,35], forward and backward propagation are not performed on entire sequence of length T. Instead, for every T^F forward passes, BPTT is performed up to T^B timestamps in the past, as shown in Fig. 9.

In Fig. 9, it is assumed that sequence length T is 6, T^F is 3 and T^B is 2. The counter for the tuples is t, implying t→1 to T. As per the t-BPTT

Table 2
Novel algorithm for optimal design of LSTM network.

Algorithm-2: Optimal design of LSTM networks
Initialize: number of binary decision variables as $M^{UB}+2$, real decision variables as 0. Fix number of generations and population, crossover and mutation probability and bounds on decision variables, in NSGA-II and other hyper-parameters (see Table 3).
for q→1 to N_{Gen}:
for p→1 to N_{pop}:
Each population contains $M^{UB}+2$ decision variables $[T^B, n^{TF}, N^m \mid m=1 \text{ to } M^{UB}]$
Deduce the LSTM architecture from given population:
for m→2 to M^{UB}:
if $N^m = 0$:
set M = m−1 and break the for loop of m.
end if loop
end for loop
architecture = [number of inputs:3, N^m : m = 1 to M, number of outputs: 6]
Test for redundancy:
if the set [architecture, T^B and n^{TF}] is present in database:
Return the corresponding objectives [R^2 and P].
else:
With [architecture, T^B and n^{TF}], train LSTM network using Algorithm 1
Validate LSTM network using a test set of size $\bar{T}$ and evaluate R^2
Evaluate the total parameters P in given LSTM network.
Update the database: [architecture, T^B and n^{TF}] corresponds to [R^2 & P]
end if loop.
end for loop of p.
Perform the operations of NSGA-II (crossover mutation selection and sorting) [37]
Update the population.
end for loop of q.

algorithm, at every t, forward propagation will happen but, back propagation will happen only once in T^F steps. Further, in back propagation, gradients will propagate till T^B timestamps from current time stamp t. Fig. 9a shows the picture at first iteration. Since T^F is 3, forward propagation will continue till $t = 3$ before backpropgation is invoked. Gradients will backpropagate from $t = 3$ to $t = 2$ because T^B is fixed at 2. This results in a weight update, completing one iteration. The next weight update will happen when $t = 6$ (multiple of T^F) as shown in Fig. 9b. These two iterations complete one epoch since $T = 6$.

In general, T^F and T^B are heuristically determined hyper-parameters. In practice, T^B is assigned a small value so that the cost associated with weight update is minimal. It is important here to understand that truncation to T^B is to convene the practical implementation of BPTT algorithm during training phase. It does not imply that, during the validation or testing phase, network requires data tuples from previous T^B timestamps to predict the current output.

In this work, LSTM networks with multiple hidden layers are explored. The mathematical equations describing the forward simulation of such networks are multidimensional extensions of Eqs. (4)–(10). The detailed analysis of multilayered LSTM networks is presented in supplementary material. The gradients of loss function are required for training the weights of LSTM network as shown in Algorithm-1, which is presented in Table 1. Among all the hyper-parameters needed for training LSTM networks, learning rate is known to have most significant effect, followed by architecture of LSTM [27].

The optimization routine used for training the weights of LSTM in Algorithm-1 is a variant of steepest descent called Adam, which stands for adaptive momentum estimation. It is observed in Adam

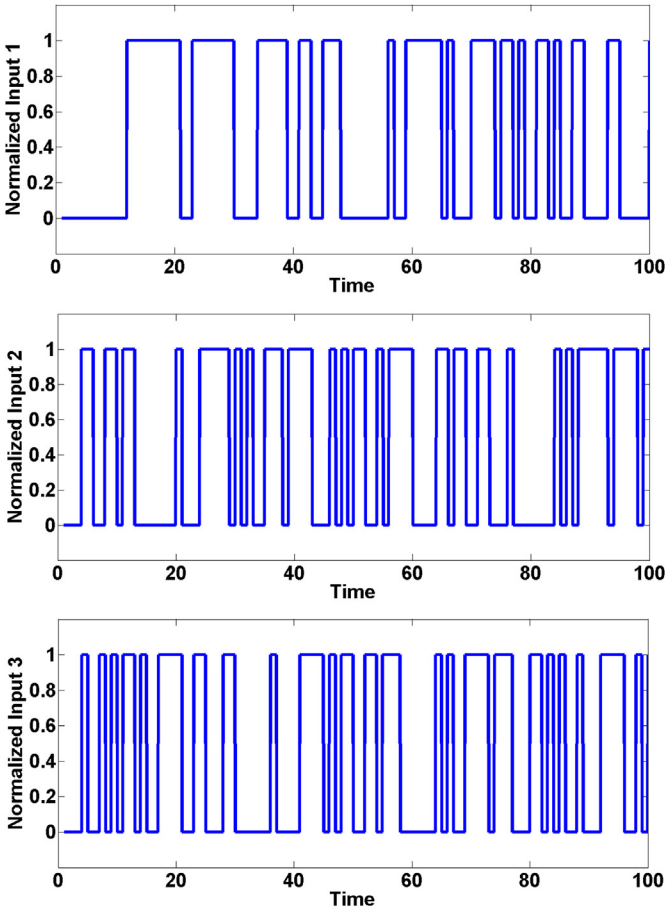

Fig. 10. PRBS input signal for training and validating LSTM and wavelet networks. Only first 100 timestamps are shown for better readability.

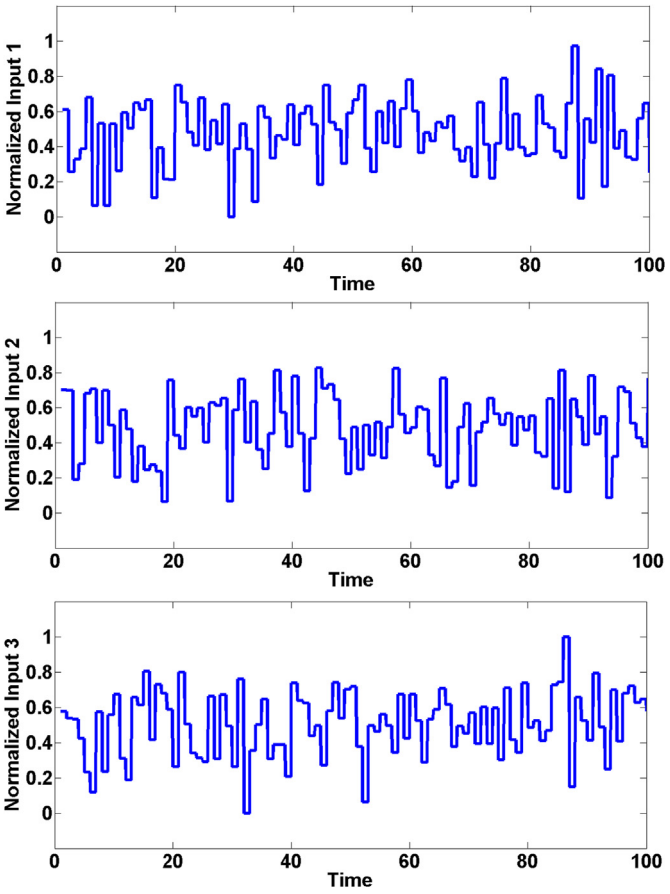

Fig. 11. RGS input signal for testing the optimal LSTM networks. Only first 100 timestamps are shown for better readability.

Table 3
List of all parameters used in Algorithm-2 for optimal design of LSTM network.

S. No	Parameter	Description	Value
1.	N_{Gen}	Number of generations	100
2	N_{pop}	Number of population	200
3	Crossover probability	NSGA-II parameter	0.9
4	Mutation probability	NSGA-II parameter	0.01
5	M^{UB}	Maximum number of hidden layers to be explored in Algorithm 2	3
6	Number of binary variables	NSGA-II parameter	5
7	Number of real variables	NSGA-II parameter	0
9	$\eta^m \mid m = 1$ to M^{UB}	Upper bounds on number of nodes in each hidden layer	{16, 15, 15}
10	T^B_{LB} and T^B_{UB}	Bounds on the length of unrolled network.	12 and 75
11	T	Number of training data points	1400
12	$\bar{T}$	Number of validation data points	600
13	N	Number of inputs in LSTM network	3
14	K	Number of outputs in LSTM network	6
15	T^F	Truncated-BPTT parameter – length of forward propagation before weight update	10
16	α	Initial learning rate	0.001
17	Q	Number of epoch	500
18	ρ_1 and ρ_2.	Adam parameters	0.9 and 0.999

that rate of convergence is comparatively faster and learning rate is adaptively scheduled based on gradients of current iteration [36]. This eliminates the need for scheduling the learning rate, as it is conventionally practiced (see Algorithm S-1 in supplementary material). However, issues such as optimal design of LSTM networks still remain unanswered.

2.2.2. Novel algorithm for optimal design of LSTM networks

ANNs are known to be universal approximators provided sufficient capacity. Predictability of ANNs can be improved by increasing number of hidden layers and number of nodes in each hidden layer. However, the extent of nonlinearity in data is not known a priori, thereby preventing the modeler to rationally decide these hyper-parameters. Tuning these hyper-parameters is a tedious task and lack of an automatic method to estimate them allows the heuristic methods to take-over, eventually rendering ANNs ineffective. LSTM networks have almost four times more number of parameters than those in conventional RNNs. Therefore, an arbitrary increase in number of hidden layers and nodes will result in steep rise in parameters, which may lead to overfitting. Thus, there exists a trade-off between accuracy and the developed model being parsimonious in LSTM network. This trade-off forms the basis for multi-objective complexity optimization in neural networks as discussed in the canonical review paper on different combinations between ANNs and evolutionary algorithms [37].

In this work, a multi-objective optimization problem is formulated to maximize the accuracy and minimize the number of parameters by optimally estimating the configuration of LSTM network, activation function choice for the nonlinearities f_3 and f_5 in Eqs. (8) and (10), respectively, and T^B, which dictates the depth of unrolled network. The prediction accuracy is evaluated by measuring correlation coefficient R^2 on validation dataset. Objectives being nonlinear functions of integral decision variables, the optimization formulation classifies itself as integer non-linear programming problem (INLP) as shown in Eq. (11).

$$\underset{T^B,\ \ N^m:\ m=1 \text{ to } M^{UB} \text{ and } n^{TF} - R^2 \text{ and } P}{\text{minimize}} \tag{11}$$

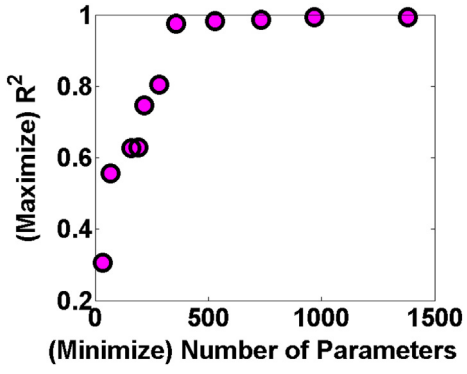

Fig. 12. Solution of multi-objective INLP in Eq. (11) obtained using NSGA-II. Each point in the Pareto front corresponds to an LSTM architecture.

Table 4
List of all estimated Pareto points displayed in Fig. 12. The highlighted entry is the selected model for further analysis. The criterion for selection is Akaike Information Criteria (AIC).

S. No	LSTM network	n^{TF} Choice of activation	T^B Time-stamps for BPTT	R^2 Correlation coefficient for validation set	P Number of parameters	RMSE for validation set	AIC measure for model selection (x10000)
1	[3-1-6]	1	13	0.306	32	0.662	−0.108
2	[3-1-1-4-6]	1	17	0.628	158	0.389	−0.232
3	[3-2-6]	1	12	0.556	66	1.318	0.0905
4	[3-2-4-6]	1	28	0.629	190	0.270	−0.328
5	[3-5-6]	1	13	0.747	216	0.158	−0.472
6	[3-6-6]	1	14	0.804	282	0.161	−0.453
7	[3-7-6]	1	13	0.975	356	0.021	−1.007
8	[3-9-6]	1	13	0.982	528	0.017	−1.031
9	[3-11-6]	1	12	0.987	732	0.017	−0.984
10	[3-13-6]	1	15	0.993	968	0.011	−1.065
11	[3-16-6]	1	12	0.995	1382	0.014	−0.948

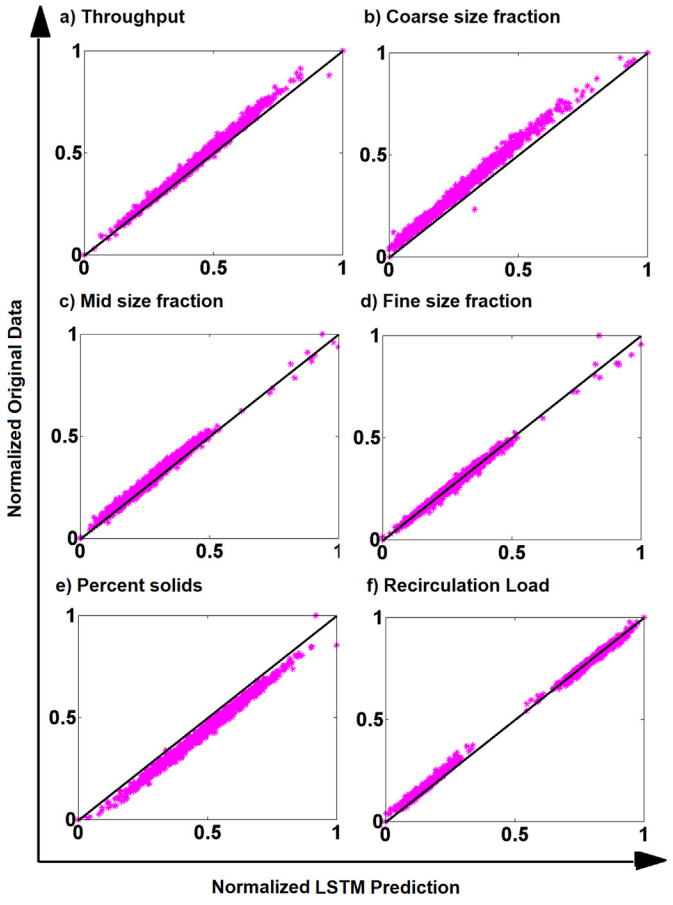

Fig. 13. Training parity plots (LSTM prediction versus original data) for all 6 outputs obtained from the selected LSTM architecture from Table 4.

$$\text{where, } R^2 = \text{mean}\left(R_k^2 : k = 1 \text{ to } K\right) \mid R_k^2 = \left(\frac{\text{cov}\left(\mathbf{Y_k}, \hat{\mathbf{Y}}_k\right)}{\sqrt{\text{var}(\mathbf{Y_k})\text{var}\left(\hat{\mathbf{Y}}_k\right)}}\right)^2$$

$$\text{cov}\left(\mathbf{Y_k}, \hat{\mathbf{Y}}_k\right) = \bar{T}\sum_{t=0}^{\bar{T}} Y_{k,t}\,\hat{Y}_{k,t} - \sum_{t=0}^{\bar{T}} \hat{Y}_{k,t}\sum_{t=0}^{\bar{T}} Y_{k,t}$$

$$\text{var}(\mathbf{Y_k}) = \bar{T}\sum_{t=0}^{\bar{T}} Y_{k,t}^2 - \left(\sum_{t=0}^{\bar{T}} Y_{k,t}\right)^2$$

$$\text{such that, } T_{LB}^B \le T^B \le T_{UB}^B$$

$$L \le N^m \le \eta^m \text{ and } L = \begin{cases} 1, & \text{if } m = 1 \\ 0, & \text{if } m > 1 \end{cases}$$

$$n^{TF} \in 1, 2 \mid \text{if } n^{TF} = \begin{cases} 1, & \text{activation is using tansigmoid} \\ 2, & \text{activation is using logsigmoid} \end{cases}$$

$T_{LB}^B,\ T_{UB}^B,\ M^{UB},\ \eta^m \in \mathbb{Z}_+$, are predefined values

In Eq. (11), $\bar{T}$ is size of validation set. $\mathbf{Y_k}$ and $\hat{\mathbf{Y}}_k$ are each vectors of length $\bar{T}$ such that, $Y_{k,t}$, $\hat{Y}_{k,t}$ is ground truth and LSTM prediction for k^{th} dimension in t^{th} tuple. K is number of outputs in LSTM network having M hidden layers and P number of parameters. Length of unrolled network is T^B.

Population based evolutionary optimization algorithm called non-dominated sorting genetic algorithm – II (NSGA-II) [38] is implemented to handle the multi-objective nature of INLP in absence of gradient information. Since the algorithm estimates the optimal number of hidden layers and nodes, the maximum number of hidden layers (M^{UB}) and nodes (N^m) to be explored by the algorithm should be provided by the user. The proposed algorithm is presented in Table 2.

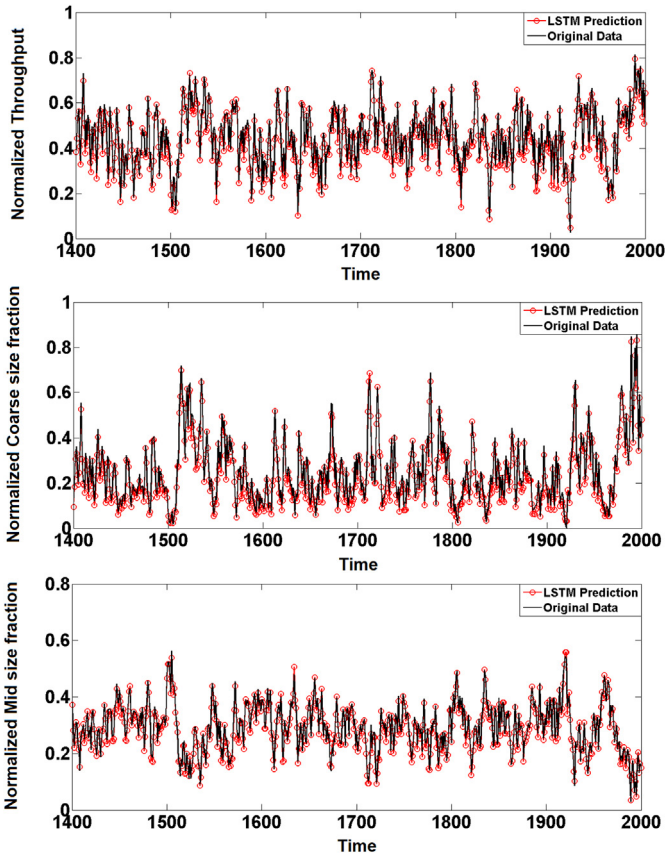

Fig. 14. Comparison of LSTM prediction and original data for 600-point validation dataset for the outputs – throughput, coarse size fraction and mid-size fraction obtained from the selected LSTM architecture from Table 4.

NSGA-II gives set of decision variables $[T^B,\ n^{TF}, N^m : m = 1 \text{ to } M^{UB}]$ from which the architecture of LSTM network can be deciphered as shown in Algorithm-2. Binary coding of NSGA-II ensures only integral decision variables. Consider for example, the case when user of Algorithm-2 decides to explore architectures with hidden layers up to 3 $\left(M^{UB} = 3\right)$, and one of the population turns out to be [97, 2, 30 23 6]. In this case, LSTM architecture will have three hidden layers with 30, 23 and 6 nodes in each hidden layer. It will be unrolled for 97 timestamps and nonlinearities in Eqs. (8) and (10) will be log-sigmoid (as indicated by the choice of n^{TF}as 2). However, in case the population is [97, 2, 30 0 6] (which can be possible due to lower bound L on N^m, see Eq. (11)), then LSTM architecture will be containing only 1 hidden layer with 30 nodes. In this way, all possible architectures with up to 3 hidden layers can be explored by NSGA-II. Decision variable space being integral in nature, there exists a high chance of repetitive populations, leading to redundant calculations. Therefore, a database, which dynamically stores all the calculations at every population, is implemented to save computation as well as improve the rate of convergence by several folds.

3. Results and Discussions

3.1. Data generation for training LSTM networks

Input signals are passed to the simulator in pseudo random binary sequence (PRBS) fashion, which ensures coverage of frequencies at wide spectrum. Corresponding output time series generated by the simulator gives a relationship between three manipulated

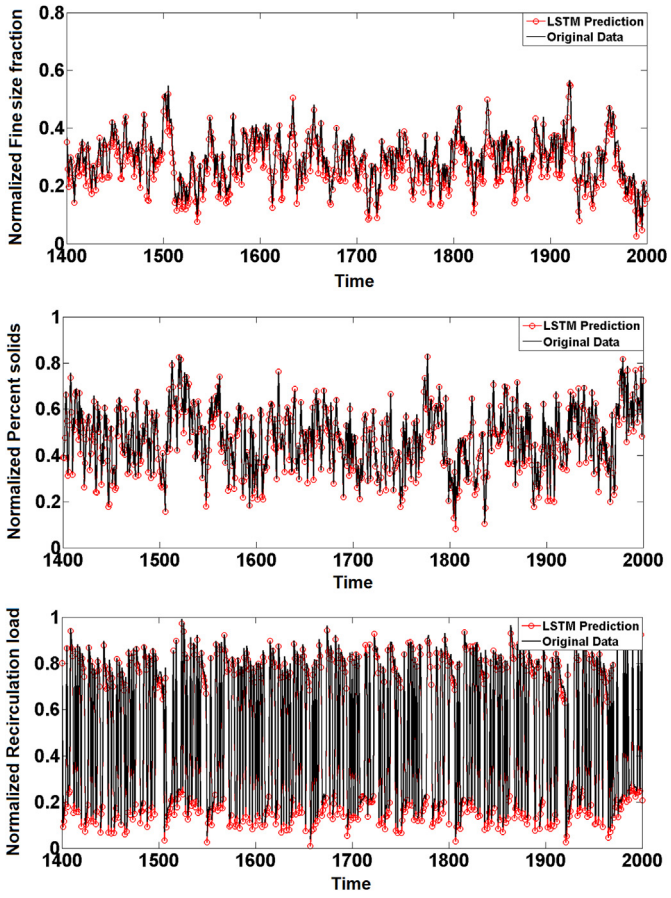

Fig. 15. Comparison of LSTM prediction and original data for 600-point validation dataset for the outputs – fine size fraction, percentage of solids and recirculation load obtained from the selected LSTM architecture from Table 4.

variables (inputs) and six control variables (outputs) [{ore feed rate to rod mill, water flowrate to primary sump, water flowrate to secondary sump} → {throughput, coarse size fraction, mid-size fraction, fine size fraction, percentage of solids and recirculation load}]. In total, a set of 2000 data tuples was collected out of which 70% of data were used for training while remaining were used for validating LSTM networks. The validation set is used for evaluating R^2 in Eq. (11). The same set of data was used for training and validating wavelet networks to provide an unbiased comparison. A separate set of 600 more data tuples was generated using random Gaussian sequence (RGS) to test the trained LSTM model. PRBS and RGS signals of the three inputs used for training and testing LSTM and wavelet networks are presented in Figs. 10 and 11, respectively.

3.2. Optimal LSTM networks

The proposed algorithm for design of optimal LSTM networks requires the training data and values of upper and lower bounds on hyper-parameters of LSTM networks such as the maximum number of hidden layers (M^{UB}) to be explored and maximum number of nodes (N^m) in each hidden layer, apart from the credentials of NSGA-II. Therefore, once the outputs are generated from the simulator using PRBS input signal, the Algorithm-2 is fed with training and validation sets and run with required credentials whose values are listed in Table 3.

Architectures up to 3 hidden layers each containing a maximum of 16 nodes were explored in this work to emulate the nonlinear data from integrated grinding circuit. The results of Algorithm-2, being a two objective INLP problem, led to a set of equally significant objective pairs called the estimated Pareto front as shown in Fig. 12. Trade-off between the objectives is clearly visible from the plot. Fewer number of parameters leads to parsimonious models at the cost of prediction accuracy.

Although NSGA-II was run for 100 generations, the Pareto front reported in Fig. 12 was obtained in generation 39 as demonstrated in the video file in the supporting material. Repeating these runs for several times, the reported Pareto front can be inferred as a close approximation of the global Pareto front. Table 4 lists all such solutions of Algorithm-2, where each of them corresponds to an LSTM architecture, activation function and required T^B to validate 600 timestamps of unseen data with the reported R^2 using the reported number of weights in the given network. Algorithm-2 along with the training of LSTM networks using Algorithm-1 was written in Fortran 90 language and run in Intel(R) Xeon (R) CPU E5-2690 0 @ 2.90 GHz (2 processors) 128 GB RAM machine. Out of a possibility of 524288 ($16 \times 16 \times 16 \times 2 \times 64$) alternatives, Algorithm-2 resulted in the 2-dimensional Pareto front containing 11 equally significant LSTM models by evaluating only 800 candidates in approximately 120 min.

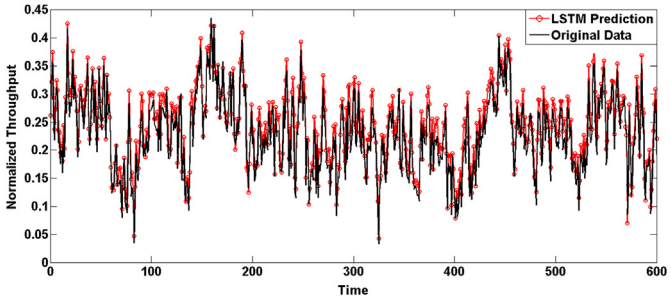

Fig. 16. Comparison of original throughput data from unseen RGS signal and prediction from optimal LSTM network obtained from Algorithm 2.

3.3. Testing and prediction using optimal LSTMs

From the set of Pareto points, one solution should be selected for performing control or optimization of integrated grinding circuit. Several metrics are available in literature, which serve as higher order information for model selection and any one of them can be utilized [38]. Akaike information criterion (AIC) is used as model selection criterion in this work [39]. The model with least value of AIC is said to be least overfitted. Root mean square errors (RMSEs) obtained by each candidate in the reported Pareto front along with corresponding AIC measures are presented in Table 4, where the selected model is highlighted.

The training parity plot for the selected architecture is shown in Fig. 13. It is observed that in Fig. 13a and 13b, the parity plot is slightly above the 45° line, while in Fig. 13e, the same is slightly below the 45° line. Moreover, the offset is almost similar for all the data points constituting the parity plot. The authors believe that the reason for this systematic offset in Fig. 13a, 13b and 13e is suboptimal tuning of bias on corresponding node in the output layer of selected LSTM network. Weights and biases are two types of parameters in ANNs which can be tuned to emulate the trend in the data. The weights in the network are known to be responsible for capturing the nonlinear trend, whereas the biases are similar to thresholds, which are responsible for activation of each node. The biases in nodes of hidden layer of LSTM network are only on the forget, input, cell and output gates. Since they are processed through nonlinear activation function, any minute error in their values will result in large prediction errors. Due to this reason, it can be inferred that the weights in the network and biases in hidden layer are not responsible for the shift observed in Fig. 13a, 13b and 13e. On the other hand, since the output layer of LSTM network contains only the summation units (without feedback and nonlinear activation, as seen in Fig. 7), the biases on these nodes act as intercepts in linear equations. Thus, they have more significant correlation to the systematic offset in outputs of the LSTM. Therefore, it is inferred that the suboptimal values of bias on the nodes in the last layer corresponding to outputs - throughput (Fig. 13a), coarse size fraction (Fig. 13b) and percent solids (Fig. 13e) are responsible for systematic offset seen in their corresponding parity plots. The authors believe that one of the possible solutions to this problem is to train six different multiple-input-single-output (MISO) LSTM models instead of one MIMO model. This will reduce the complexity for local optimizer responsible for training LSTM networks. However, extensive training may also result in overfitting of the network.

In this work, to check overfitting, the optimal MIMO LSTM model was validated on 600-point validation dataset and tested on RGS input signal (see Fig. 11). For the PRBS signal, RMSE and R^2 values obtained on validation dataset were found to be 0.011 and 0.993, respectively. The corresponding plots are presented in Figs. 14 and 15. An RMSE of 11% was obtained when the LSTM network was tested on RGS input signal. The plot comparing a 600-point validation dataset for RGS input signal for output 1 (throughput) is shown in Fig. 16. These results of validation and testing conclude that the selected LSTM network is not over-fitted and the error in training the weights and biases of selected MIMO LSTM model is within the acceptable limit.

3.4. Comparison with wavelets

Dataset obtained by PRBS input signal is also used to train two wavelet based models in MATLAB®. One is Hammerstein-Wiener model [40] and the other is wavelet neural network [41]. Hammerstein-Wiener model decomposes the input-output relationship into a series of static nonlinear block with a dynamic linear block in between. Here, the nonlinearities are captured by activation functions and dynamics are modelled by linear transfer function. For the analysis in this work, wavelet networks are used to model the static nonlinearities in Hammerstein Wiener models. On the other hand, a wavelet neural network is similar to a perceptron neural network consisting of an input, hidden and an output layer. In this case, wavelet functions are used for activation in place of sigmoid functions. To model the dynamics, one has to fix the number of previous timestamps (order of the model) required to predict the current outputs (see Section 2.2.1). Therefore, they belong

Table 5
Analysis of wavelet based models in MATLAB for modeling the time series data obtained from integrated grinding circuit. The RMSE values are averaged over all 6 outputs.

S. No	**Model** (Type of the wavelet network)	**Order** (same for inputs and outputs unless specified)	**Nodes in Hidden layer** (Nonlinearity is using a network)	**RMSE** (Root Mean Square Error over training set for all outputs)
1	Hammerstein Wiener	2	10 for all inputs and outputs	0.12
2	**Hammerstein Wiener**	**2**	**6 nodes for output1** **30 nodes for output 6** **10 for all inputs and other outputs**	**0.11**
3	Hammerstein Wiener	2	5 for all inputs and 6 for output 1 and 10 for other outputs	0.13
4	Hammerstein Wiener	2	2 for all inputs and 1 for output 1 and 6 and 10 for other outputs	0.16
5	Wavelet neural network	2	10 nodes	0.16
6	Wavelet neural network	2	15 nodes	0.15
7	Wavelet neural network	2	25 nodes	0.15
8	Wavelet neural network	2	39 nodes	0.16
9	Wavelet neural network	2 in inputs and 5 in outputs	27 nodes	0.14
10	**Wavelet neural network**	**5 in inputs and 1 in outputs**	**20 nodes**	**0.13**

to a more generic framework called Nonlinear Autoregressive Exogenous (NARX) modeling for system identification of datasets with nonlinear dynamics.

Wavelet based models in MATLAB® being MISO in nature, 6 models corresponding to 6 outputs were built. Several experiments with hyper-parameters were performed and only those with significant fit among all are reported in Table 5, where the best models are highlighted.

These highlighted models in Table 5 are used to plot Fig. 17, which shows the comparison between the optimal LSTM networks (see Table 4), wavelet neural networks and Hammerstein Wiener model for emulating the training data obtained using PRBS input signal for output 1 (throughput). Fig. 18 compares the distribution of training error in each of these cases for all outputs using box plots.

A box plot displays the five-number summary of a set of data. The five-number summary is the minimum, first quartile, median, third quartile, and maximum. A box is drawn from the first quartile to the third quartile. A vertical line or partition goes through the box at the median. The whiskers go from each quartile to the minimum or maximum. In Fig. 18, three box plots corresponding to Hammerstein-Wiener model, wavelet neural networks and LSTM network are drawn for each output of grinding circuit. The set of data for each box plot (plotted on the abscissa and labeled as error) is obtained by calculating the difference between original and predicted output of training data. The interpretations from Fig. 18 are given below.

- The box plots corresponding to LSTM networks for all outputs are much smaller in width and centered uniformly around the median when compared to those for wavelet approaches. This indicates that, for all data points in training set, the error is similar in magnitude for LSTM networks. However, in wavelet approaches, the magnitude of error among the training points is widely distributed.
- The median, which is one of the statistical measures of average for the given set of data, indicates the average of error in training set. The observation that the median for LSTM network is close to 0 implies the accuracy of the model. On the other hand, for wavelet approaches, the median is located far away from 0 mark on both sides (because the dataset contains exact values instead of absolute values of error), signifying the extent of training error.
- The similarity in box plots for all outputs in case of LSTM networks demonstrates their efficiency over wavelet approaches in emulating each output of industrial grinding circuits with similar accuracy.

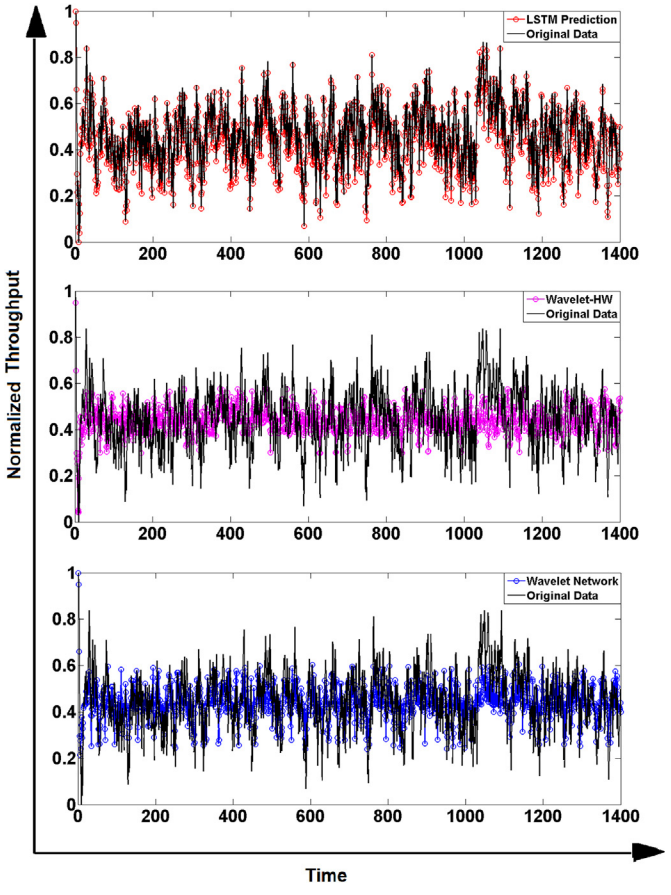

Fig. 17. Comparison of original throughput data from PRBS signal and prediction from LSTM network, Hammerstein Wiener (HW) model and wavelet network.

There are several hyper-parameters, which influence the efficiency of these wavelet based models. For example, in both Hammerstein-Wiener and wavelet neural networks, the main hyper-parameters, which need to be known a priori, are amount of delay, order of the model and number of nodes in the only hidden layer. It should be noted that fixing of these hyper parameters is done quite often based on practical experience or trial and error. Wavelet based models are explored in current work only for comparing them with LSTM networks obtained using Algorithm-2. Thus, without the intent of discrediting wavelet based models, minimal efforts towards hyper-parameter tuning were invested to generate the reported results reflecting a common practice in industry. It could always be possible that an optimal set of hyper-parameters may result in better efficiency than those reported in the current manuscript. However, the analysis of amount of time and resources required for such a study in domain of wavelets is beyond the scope of proposed work as the main aim of this manuscript is LSTMs and the associated optimal designs.

3.5. Discussions

Algorithm-2 resulted in a set of solutions from which the LSTM architecture [3–13] [3-13-6] was selected using AIC criterion. For the architecture [3–13] [3-13-6] to emulate grinding data with maximum accuracy, it was found that tan sigmoid activation function needs to be implemented while unrolling the network for 15 timestamps.

When a conventional RNN is trained with PRBS data by unrolling it for 15 timestamps, it is observed that gradients of error at the 15^{th} timestamp with respect to weights in earlier timestamps in unrolled network are negligible $(\sim 1e^{-10})$. One such realization is demonstrated in Fig. 19, which shows the evidence of vanishing gradients in RNNs while modeling grinding time series data.

Here, an RNN with same configuration as optimal LSTM is considered with tan sigmoid activation. The network is trained by unrolling it for 15 timestamps. This network has 305 weights of which 169 are

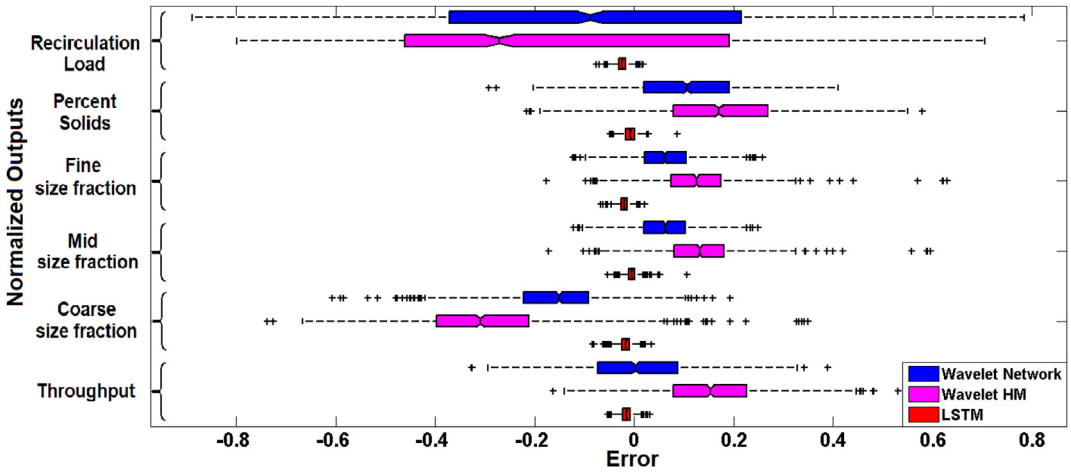

Fig. 18. Distribution of training error for all outputs from the integrated grinding circuit for three types of system identification tools – wavelet networks, Hammerstein Wiener (HW) models and optimal LSTM networks obtained from analysis presented in Table 4 and Algorithm 2, respectively, in a box plot.

context weights (weights on feedback connections) and the remaining 136 are weights on feedforward connections. To visualize the vanishing gradient problem, error at the 15th timestamp is back propagated and mean of its gradient taken over all the 305 weights is plotted at each time stamp in Fig. 19a. The distribution characteristics of this gradient over all the weights is shown in Fig. 19b. It can be observed that the gradients die out by the time they reach 1st time stamp.

The vanishing gradient problem can also be realized by observing the training equations. It is known that RNNs are trained using first order gradients that are obtained by t-BPTT. Consider the simpler case when number of inputs and outputs are equal to 1 and there exists only 1 hidden node in the network. Let it be the first iteration of weight update implying $T^F = T^B$ and let $T^B = 15$ as in the numerical case study demonstrated in Fig. 19. Then, the expression for gradient of error $\mathcal{L}$

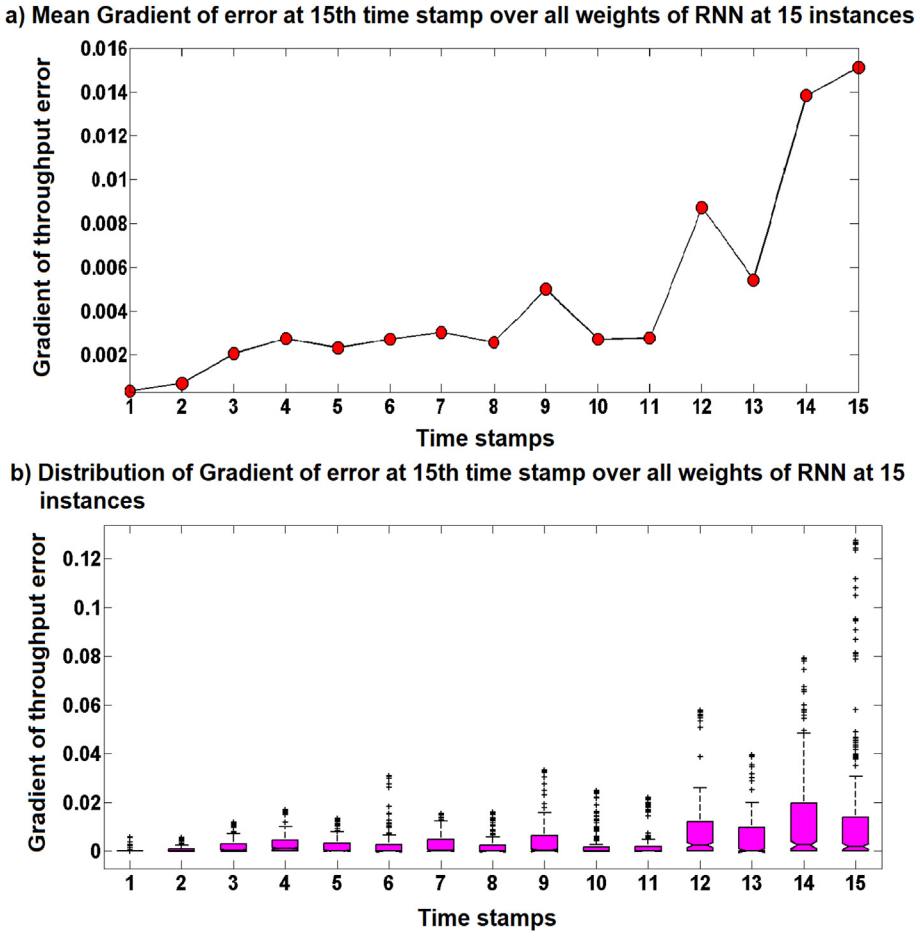

Fig. 19. Demonstration of the issues of vanishing gradients in RNN while learning the time series data of throughput. (a) Average of gradients of error at the 15th timestamp with respect to all 305 weights; (b) Box plot showing the distribution characteristics of the same.

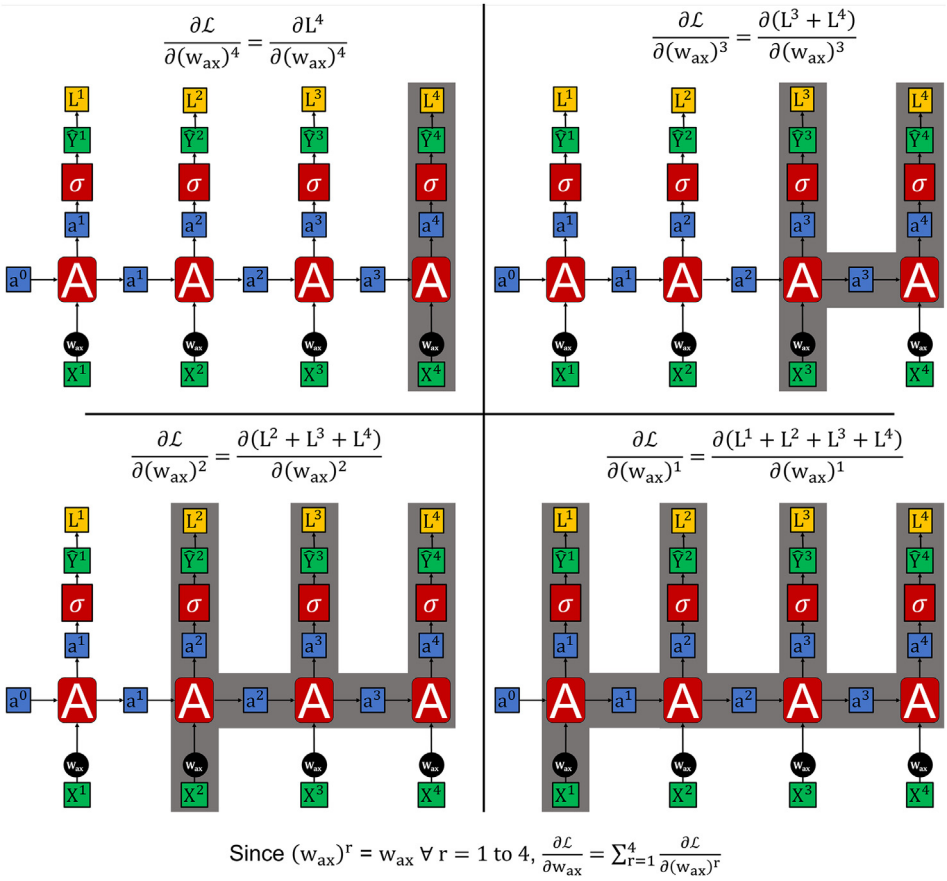

Fig. 20. Pictorial representation of gradient calculation in RNNs (see Eq (13)). This plot demonstrates the effect of weight sharing across timestamps in RNNs. A simple RNN is considered and unrolled for 4 timestamps.

with respect to a weight w_{ax} connecting the input to the hidden node is given by Eq. (12).

$$\frac{\partial \mathcal{L}}{\partial w_{ax}} = \sum_{r=1}^{15} \frac{\partial \mathcal{L}}{\partial (w_{ax})^r} \tag{12}$$

The summation over all timestamps is because, although the unrolled version of the network has T^B timestamps, each timestamp represents a temporal realization of same network. Therefore, the weight w_{ax}, which is invariant of time in RNNs, remains same in all timestamps. Thus, the gradient of $\mathcal{L}$ with respect to weight w_{ax} is sum of gradients of $\mathcal{L}$ with respect to $(w_{ax})^r$ taken over all timestamps. However, it is important to note that $(w_{ax})^r$ in Eq. (12), is only for improving the readability and as explained, $\forall\ r = 1$ to 15, $(w_{ax})^r = w_{ax}$. This is visualized in Fig. 20 and holds true for any type of RNNs including LSTM networks (see supplementary material for more details).

$$\frac{\partial \mathcal{L}}{\partial (w_{ax})^r} = \frac{\partial \left(L^r + L^{r+1} \ldots L^{15}\right)}{\partial (w_{ax})^r} = \frac{\partial L^r}{\partial (w_{ax})^r} + \frac{\partial \mathcal{L}^{r+1}}{\partial (w_{ax})^r} \text{ where } \mathcal{L}^{r+1} = L^{r+1} + \ldots L^{15} \tag{13}$$

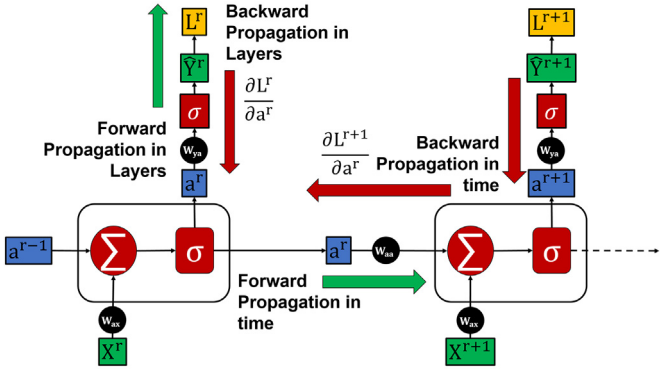

Fig. 21. Pictorial representation of back propagation through time in RNNs.

The gradient at time instance r is evaluated using BPTT as shown in Fig. 21 and Eq. (13). The first term corresponds to back propagation of error signal through the feedforward layers and second term corresponds to back propagation of error signal through timestamps. The second term is of more significance when analyzing the capability of RNNs to model time dependencies.

Since BPTT is the chain rule of differentiation, clearly, the gradient of error traversing through time (second term in Eq. (13)) will contain the recurring product of derivative of activation function evaluated at each timestamp and weight w_{aa}, over n timestamps. For sigmoid activation functions, the derivative is always ≤1. In case the derivative at each timestamp $d^r \to 0$, say d, then the resulting gradient will be function of d^n, which will be negligible. In the example considered in Fig. 19, n = 15. Therefore, conventional RNNs cannot model the long term dependencies in nonlinear time series data.

One of the solutions to minimize the issue of vanishing gradients is to prevent the flow of gradients in time through activation units. This will lead to a linear expression similar to Eq. (1), This is the idea behind LSTMs and is implemented by introducing the concept of cell state in every node in the network as shown in Eq. (4) and Fig. 8. This justifies the use of LSTM networks for system identification of integrated grinding circuit, where they were able to prevent the vanishing gradient problem.

Both of the wavelet based models discussed in this work function similar to NARX models, where the order of the model needs to be fixed a priori. Further, these models unlike the RNNs, do not share the parameters across the timestamps, thereby failing to learn the dynamic characteristics of the data. These models consider the dataset as a set of samples instead of exploring the sequential information. These reasons and the choice of hyper-parameters may have resulted in the failure of wavelet based models for emulating the grinding data. As pointed out earlier, the aim here is not to state the incapability of the wavelet based models as the correct choice of values for hyper-parameters can lead to significantly better match; rather the important aspect is to present an optimization framework, against the popularly used heuristics based approach, which can find optimal values of these hyper-parameters automatically. The framework presented in this work displays that novelty with LSTMs.

The analysis performed using Algorithm-2 revealed that only 4 architectures out of 800 candidates evaluated using the optimization framework involving NSGA-II were capable of modeling the grinding data with accuracy greater than 98% (see Table 4). Without the presence of such a framework (Algorithm-2), the design of LSTM networks would have been an arduous task. Moreover, the proposed algorithm also estimated the optimal unrolling length of the network (T^B), which cannot be assessed heuristically. The multi-objective optimization framework in Algorithm-2, where the predictability of RNNs on validation set is maximized while minimizing total number of parameters in the model, is designed to minimize the chances of overfitting in LSTM networks. The evolutionary optimization strategy eliminated the networks with poor generalization ability thereby resulting in only the set of networks, which are least overfitted. In the current work, the networks trained and validated on PRBS signal from grinding data are also successfully tested on unseen RGS signal. This justifies the robustness of constructed LSTM networks and the possibility of their use in closed loop control strategies.

4. Conclusion

Deep learning based LSTM networks have not witnessed sufficient applicability in the domain of process systems engineering. In this work, the scope of LSTM networks for system identification of industrial grinding circuits is studied. The work is summarized in the following points.

- Process flowsheet of integrated grinding circuit in lead-zinc ore beneficiation process is explained and data for MIMO system identification is generated using an industrially validated simulator.
- Analyzing the time series data from grinding process, presence of long term dependencies is realized through numerical investigations.
- Scope of conventional system identification tools such as NARX modeling using wavelets, Hammerstein Wiener models and RNNs towards modeling the aforementioned system is discussed.
- Issue of vanishing gradients and long term memory are tackled by implementing LSTM networks: however, heuristics associated with deep networks made their modeling difficult.
- Design of LSTM models using a novel multi-objective optimization algorithm enabled the determination of best architectures to emulate the grinding circuit.
- Optimal LSTM model trained and validated on PRBS signal with 99% accuracy is also tested on unseen RGS signal with an accuracy of 98%.

The proposed work concludes that if LSTM networks are trained properly by using optimal hyper-parameters, these deep learning based models can act as accurate surrogates for industrial integrated grinding circuits and enable their real time control and optimization for maximizing productivity and minimizing energy consumption. Implementation of these models in such closed loop control algorithms after performing rigorous theoretical analysis for stability of optimal LSTMs can be considered as future scope for the current work.

Declarations of interest

None.

CRediT authorship contribution statement

Srinivas Soumitri Miriyala: Conceptualization, Methodology, Software, Validation, Investigation, Data curation, Formal analysis, Writing - original draft, Visualization. **Kishalay Mitra:** Conceptualization, Visualization, Investigation, Resources, Writing - review & editing, Supervision, Project administration.

Acknowledgements

This work was supported by the SPARC project, Ministry of Human Resources Development (MHRD), Government of India [grant number: SPARC/2018-2019/P1084/SL].

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.powtec.2019.10.065.

References

[1] A. Pomerleau, D. Hodouin, A. Desbiens, E. Gagnon, A survey of grinding circuit control methods: from decentralized PID controllers to multivariable predictive controllers, Powder Technol. 108 (2000) 103–115.
[2] L.G. Austin, Introduction to the mathematical description of grinding as a rate process, Powder Technol. 5 (1971) 1–17.
[3] H. Berthiaux, C. Chiron, J. Dodds, Modelling fine grinding in a fluidized bed opposed jet mill: part II: continuous grinding, Powder Technol. 106 (1999) 88–97.
[4] J. Yang, S. Li, X. Chen, Q. Li, Disturbance rejection of ball mill grinding circuits using DOB and MPC, Powder Technol. 198 (2010) 219–228.
[5] P. Zhou, S. Lu, M. Yuan, T. Chai, Survey on higher-level advanced control for grinding circuits operation, Powder Technol. 288 (2016) 324–338.
[6] J. Kwon, J. Jeong, H. Cho, Simulation and optimization of a two-stage ball mill grinding circuit of molybdenum ore, Adv. Powder Technol. 27 (2016) 1073–1085.
[7] F.C. Bond, Crushing and grinding calculations, Part II, Br. Chem. Eng. 6 (1961) 543–548.
[8] K. Mitra, M. Ghivari, Modeling of an industrial wet grinding operation using data-driven techniques, Comput. Chem. Eng. 30 (2006) 508–520.
[9] M.S. Powell, R.D. Morrison, The future of comminution modelling, Int. J. Miner. Process. 84 (2007) 228–239.
[10] V.K. Gupta, D. Hodouin, M.D. Everell, An analysis of wet grinding operation using a linearized population balance model for a pilot scale grate-discharge ball mill, Powder Technol. 32 (1982) 233–244.
[11] S. Morrell, U.J. Sterns, K.R. Weller, The application of population balance models to very fine grinding in tower mills, Proceedings of the XIX International Mineral Processing Congress 1993, pp. 61–66 , Sydney.
[12] A. Casali, G. Gonzalez, F. Torres, G. Vallebuona, L. Castelli, P. Gimenez, Particle size distribution soft-sensor for a grinding circuit, Powder Technol. 99 (1998) 15–21.
[13] T.A. Bell, Challenges in the scale-up of particulate processes—an industrial perspective, Powder Technol. 150 (2005) 60–71.
[14] R. Guizani, I. Mokni, H. Mhiri, P. Bournot, CFD modeling and analysis of the fish-hook effect on the rotor separator's efficiency, Powder Technol. 264 (2014) 149–157.
[15] R.K. Rajamani, B.K. Mishra, R. Venugopal, A. Datta, Discrete element analysis of tumbling mills, Powder Technol. 109 (2000) 105–112.
[16] H. Lee, H. Cho, J. Kwon, Using the discrete element method to analyze the breakage rate in a centrifugal/vibration mill, Powder Technol. 198 (2010) 364–372.
[17] E. Bilgili, M. Capece, Quantitative analysis of multi-particle interactions during particle breakage: a discrete non-linear population balance framework, Powder Technol. 213 (2011) 162–173.
[18] K. Nageswararao, Modelling of hydrocyclone classifiers: a critique of 'bypass' and corrected efficiency, Powder Technol. 297 (2016) 106–114.
[19] R. Lestage, A. Pomerleau, D. Hodouin, Constrained real-time optimization of a grinding circuit using steady-state linear programming supervisory control, Powder Technol. 124 (2002) 254–263.
[20] K. Mitra, R. Gopinath, Multiobjective optimization of an industrial grinding operation using elitist nondominated sorting genetic algorithm, Chem. Eng. Sci. 59 (2004) 385–396.
[21] A.K. Pani, H.K. Mohanta, Soft sensing of particle size in a grinding process: application of support vector regression, fuzzy inference and adaptive neuro fuzzy inference techniques for online monitoring of cement fineness, Powder Technol. 264 (2014) 484–497.

[22] E. Hamzeloo, M. Massinaei, N. Mehrshad, Estimation of particle size distribution on an industrial conveyor belt using image analysis and neural networks, Powder Technol. 261 (2014) 185–190.
[23] Y.D. Ko, H. Shang, A neural network-based soft sensor for particle size distribution using image analysis, Powder Technol. 212 (2011) 359–366.
[24] R. Pascanu, T. Mikolov, Y. Bengio, On the difficulty of training recurrent neural networks, Proceedings of International Conference on Machine Learning 2013, pp. 1310–1318.
[25] H. Wang, C. Luo, X. Wang, Synchronization and identification of nonlinear systems by using a novel self-evolving interval type-2 fuzzy LSTM-neural network, Eng. Appl. Artif. Intell. 81 (2019) 79–93.
[26] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (1997) 1735–1780.
[27] K. Greff, R.K. Srivastava, J. Koutník, B.R. Steunebrink, J. Schmidhuber, LSTM, A search space odyssey, IEEE Trans.Neur. Network. Learn. Syst. 28 (2017) 2222–2232.
[28] N. Srivastava, H.E. Geoffrey, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting, J. Mach. Learn. Res. 15 (2014) 1929–1958.
[29] W. Zaremba, I. Sutskever, O. Vinyals, Recurrent Neural Network Regularization, 2014 arXiv preprint arXiv:1409.2329.
[30] N. Reimers, I. Gurevych, Optimal Hyperparameters for Deep Lstm-Networks for Sequence Labeling Tasks, 2017 arXiv preprint arXiv:1707.06799.
[31] L.R. Petzold, A Description of DASSL: a differential/algebraic system solver, in: R.S. Stepleman, et al., (Eds.), Scientific Computing, North-Holland, Amsterdam 1983, pp. 65–68.
[32] L. Ljung, System Identification, Wiley Encyclopedia of Electrical and Electronics Engineering, 1999.
[33] I. Sutskever, O.I. Vinyals, Q.V. Le, Sequence to sequence learning with neural networks, Proceedings of Advances in Neural Information Processing Systems, 2014.
[34] P.J. Werbos, Backpropagation through time: what it does and how to do it, Proceedings of the IEEE, vol. 78, 1990, pp. 1550–1560 , 10.
[35] R. Williams, J. Peng, An efficient gradient-based algorithm for on-line training of recurrent network trajectories, Neural Comput. 2 (1990) 490–501.
[36] D.P. Kingma, J. Ba, Adam, A Method for Stochastic Optimization, 2014 arXiv preprint arXiv:1412.6980.
[37] X. Yao, Evolving artificial neural networks, Proceedings of the IEEE, vol. 87, 1999, pp. 1423–1447 , 9.
[38] K. Deb, A. Pratap, S. Agarwal, T.A.M.T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002) 182–197.
[39] H. Akaike, Factor analysis and AIC, Selected Papers of Hirotugu Akaike, Springer, New York, NY 1987, pp. 371–386.
[40] A. Wills, T.B. Schön, L. Ljung, B. Ninness, Identification of hammerstein–wiener models, Automatica 49 (2013) 70–81.
[41] Q. Zhang, A. Benveniste, Wavelet networks, IEEE Trans. Neural Netw. 3 (1992) 889–898.

Srinivas Soumitri Miriyala received the Bachelor's degree in Chemical Engineering from Jawaharlal Nehru Technological University Kakinada, India in 2013, and his Master's degree in Chemical Engineering from Indian Institute of Technology Hyderabad, India in 2015. He is currently in his PhD program in the Department of Chemical Engineering at Indian Institute of Technology Hyderabad. His research interests include Machine Learning, Deep Learning, Surrogate based Optimization, Optimization under uncertainty and stochastic modeling using Monte Carlo methods.

Dr. Kishalay Mitra received the B.E. degree from the National Institute of Technology Durgapur, India, in 1995, the M. Tech degree from Indian Institute of Technology Kanpur, India, in 1997, and the Ph.D. degree from Indian Institute of Technology Bombay, India, in 2009 (all in Chemical Engineering). He is an Associate Professor in the department of Chemical Engineering at Indian Institute of Technology Hyderabad. His work interests lie in the interface of data analysis and process optimization such as Machine Learning, Surrogate Optimization, Uncertainty Quantification, Planning and Scheduling, Optimal Control, and analysis of Biological and Renewable systems.