A Deep Learning Model for Smart Manufacturing Using Convolutional LSTM Neural Network Autoencoders

Aniekan Essien, *Member, IEEE*, and Cinzia Giannetti

***Abstract*—Time-series forecasting is applied to many areas of smart factories, including machine health monitoring, predictive maintenance, and production scheduling. In smart factories, machine speed prediction can be used to dynamically adjust production processes based on different system conditions, optimize production throughput, and minimize energy consumption. However, making accurate data-driven machine speed forecasts is challenging. Given the complex nature of industrial manufacturing process data, predictive models that are robust to noise and can capture the temporal and spatial distributions of input time-series signals are prerequisites for accurate forecasting. Motivated by recent deep learning studies in smart manufacturing, in this article, we propose an end-to-end model for multistep machine speed prediction. The model comprises a deep convolutional LSTM encoder–decoder architecture. Extensive empirical analyses using real-world data obtained from a metal packaging plant in the United Kingdom demonstrate the value of the proposed method when compared with the state-of-the-art predictive models.**

***Index Terms*—Convolutional long short-term memory (ConvLSTM), deep learning (DL), industry 4.0, stacked autoencoders, time-series forecasting.**

I. Introduction

SMART manufacturing integrates big data, advanced analytics, high-performance computing, and the industrial Internet of things to manufacturing systems and industries to improve manufacturing processes, resulting in better quality products that are available at lower costs [1]. In smart factories, devices, machines, and processes are interconnected, monitored, and optimized to enhance productivity and efficiency. The application of these technologies for manufacturing now represents a key enabler for addressing challenges in manufacturing processes, such as demand management, production/inventory supply, and scheduling. A typical example is a situation where sensors can observe production lines in real-time, store this production data and use the analysis (from the observations) to produce forecasts of the projected/anticipated production plan.

Machine learning (ML) is considered as an enabling technology for smart manufacturing, which has contributed to the growth of the 'industry 4.0' era, resulting in increased research interest in the application of data/predictive analytics, ML, and advanced information and communication technologies for improving manufacturing processes. Consequently, both supervised and unsupervised learning approaches, such as principal components analysis [2], [3], artificial neural networks (ANN) [4], Bayes network [5], and regression trees [6] have been applied to manufacturing process optimization.

Data-driven techniques for predictive analytics in smart manufacturing can be classified as (traditional) ML techniques and deep learning (DL) techniques [7]. In the literature, many data-driven approaches have been applied toward time-series forecasting or classification, including autoregressive integrated moving average (ARIMA) [8], support vector machines [9], statistical analysis [10], and instance-based learning techniques. The application of traditional ML techniques on large datasets consistently exposes the inherent vulnerabilities of these models, such as the inability to deal with the high dimensionality of the data feature space, multicollinearity, and varying data aggregation [11].

On the other hand, DL refers to techniques for learning high-level features from data in a hierarchical manner using stacked layer-wise architectures [12]. Four main groups of DL architectures have been identified in the literature:

1) autoencoders;
2) deep belief networks;
3) convolutional neural networks (CNN);
4) recurrent neural networks (RNN).

In recent history, the results obtained from DL have increased research interest in the application of DL to the research area of smart manufacturing [11]. DL models demonstrate excellent predictive capabilities in image and speech recognition, natural language processing, and intelligent gamification. In smart manufacturing, many studies have developed and applied DL algorithms for anomaly detection [13], faults diagnosis [14], and machine health monitoring [15] and [16]. Although they

Manuscript received August 1, 2019; revised October 18, 2019 and December 23, 2019; accepted December 29, 2019. Date of publication January 23, 2020; date of current version May 26, 2020. This work was supported by the UK Engineering and Physical Sciences Research Council (EPSRC) project EP/S001387/1 and the European Regional Development Funds project IMPACT and Supercomputing Wales. Paper no. TII-19-3559. *(Corresponding author: Aniekan Essien.)*

The authors are with Future Manufacturing Research Institute, College of Engineering, Swansea University, Swansea SA1 8EN, UK (e-mail: a.e.essien@swansea.ac.uk; c.giannetti@swansea.ac.uk).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2020.2967556

are adept at image recognition, CNNs mainly operate in a vector space, thereby increasing the difficulty in learning the high-dimensional features of input time series. For this reason, applying CNN architectures alone to the time-series forecasting problem is sub-optimal. However, long short-term memory (LSTM) networks are skillful in sequential learning by passing signal information across time steps. Leveraging the complementary strengths of CNN and LSTM neural networks, the convolutional LSTM (convLSTM) model both preserves spatial information and performs well in sequential learning [17].

Motivated by this modeling approach, this article proposes 2-DConvLSTMAE, a deep ConvLSTM stacked autoencoder for univariate, multistep machine speed forecasting in a manufacturing process. The end-to-end model has three distinct components as follows:

1) convLSTM encoding layers;
2) bidirectional stacked LSTM decoding layers;
3) time-distributed supervised learning [fully connected (FC)] layer.

The input time-series signal is reconstructed into a sequential supervised learning format (i.e., sequences of fixed window size and output sequences) using a sliding-window strategy. Each input sequence is fed into the encoding layer. The output from the first encoding layer serves as the input to the second encoding layer. The resulting sequences are passed onto the flatten and repeat vector layers, respectively, where they are reconstructed into a one-dimensional (1-D) tensor. This is then passed to the decoding layer, where a stack of bidirectional LSTM layers reconstructs the original input time series from the resultant tensor. The last layer is a time-distributed FC regression layer for multistep machine speed forecasting.

In this article, the problem is formulated as a sequential (i.e., sequence-to-sequence) forecasting problem. The input data comprises the internal speed (measured as the number of strokes per minute) of a metal can bodymaker machine, which operates at high speed and produces up to 300 aluminum cans per minute. In can manufacturing production planning processes, predicting the speed of the bodymaker machine, which is related to the number of cans produced by the machine, can be used to optimize production schedules by allowing real-time adjustment of the individual operating speeds for other upstream or downstream machines. For details about metal can manufacturing, the reader can refer to [18].

We test the performance of 2-DConvLSTMAE using historical, real-world machine speed data obtained via machine-embedded sensors in an aluminum can manufacturing machine from a metal packaging plant in the United Kingdom. The results of rigorous empirical analyses in this article substantiate the value of the proposed approach when compared to state-of-the-art DL models.

The contributions of this article are summarised as follows.

1) An end-to-end multistep (i.e., sequence-to-sequence) time-series forecasting model comprising a convLSTM encoder–decoder architecture for multistep machine speed prediction.
2) A time-distributed encoder–decoder model, which is capable of short- and long-term representation learning for machine speed prediction in smart manufacturing.
3) A robust, scalable, DL predictive model that has been evaluated on real-world, real-time machine speed signals in a manufacturing plant.

The remainder of this article is organized as follows. Section II presents the technical preliminaries of key concepts used in this article. Section III presents the proposed 2-DConvLSTMAE model and methodological approach. In Section IV, the experiments and results are discussed, while Section V concludes this article.

II. Technical Preliminaries

This section formulates the sequence-to-sequence time-series forecasting problem and provides a technical background about key concepts in the domain of DL, which are used within this article.

A. Problem Formulation

The goal of multistep (i.e., sequence-to-sequence) time-series forecasting is the use of previously observed (i.e., lagged) input sequences to forecast a fixed-length sequence of the future time-series values. In ML, this is typically regarded as a sequential time-series forecasting problem or sequence-to-sequence forecasting [19]. To achieve this, the sliding-window method [20] is adopted, which converts the sequential input data to a supervised learning problem (i.e., inputs and outputs). In this method, a portion of the input time-series sequence (a window of lagged values) is reconstructed to serve as input features. The number of previous time steps is referred to as the window width/size.

Given a univariate time series $x(t) = \{x_1, x_2, x_3, \dots, x_t\}$, the sequence-to-sequence forecasting problem is to predict the future k values of the sequence, $\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_k) \cong (x_{t+1}, x_{t+2}, \dots, x_{t+k})$, using the values of previous observations in a sliding window of fixed size w, such that

$$\hat{y} = (\hat{y}_1, \hat{y}_2, \dots, \hat{y}_k) = f(x_{t-w}, x_{t-w+1}, x_{t-w+2}, \dots, x_t). \tag{1}$$

In this article, the input observations refer to the machine speed obtained over a regular period (i.e., one minute). It is important to note that sequence-to-sequence forecasting differs from single-step time-series forecasting because the predicted output is a sequence of predicted machine speed values rather than a single value of the predicted variable. The above data transformations for a time series of length N result in a sequence-to-sequence forecasting problem having an input matrix $X \in \mathbb{R}^{n \times w}$ and output matrix $Y \in \mathbb{R}^{n \times k}$, where $n = (N - w - k + 1)$ is the number of training samples (Fig. 5 shows details about the adopted sliding-window approach).

B. Structure of the CNN

The CNN is a feedforward neural network that is mostly adopted in image and video recognition [14]. Fig. 1 shows the structure of a typical CNN model. The input layer takes the input vector and develops a feature graph corresponding to the convolution kernel, which uses a set of weights to produce a feature graph, which is passed onto the next layer. The link between the input and convolution layer is established by a

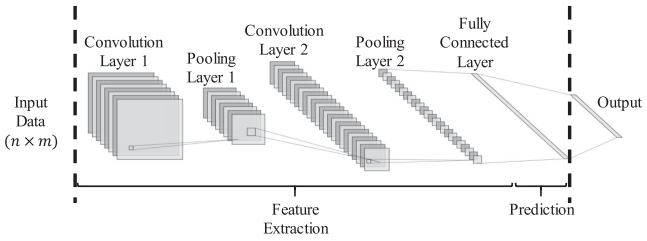

Fig. 1. Structure of a typical CNN.

receptive field, which is a square matrix of weights with sizes that are smaller than the input. As the receptive field strides or "convolves" along the input area, it executes the convolution operation, which is described as

$$y_{ij} = \sigma\left(\sum_{r=1}^{F}\sum_{c=1}^{F} w_{rc} x_{(r+1\times S)(c+j\times S)} + b\right) \tag{2}$$

$$0 \le 1 \le \frac{H-F}{S}, \quad 0 \le j \le \frac{W-F}{S} \tag{3}$$

where y_{ij} denotes the output of a node on the feature map, H and W represent the height (vertical) and width (horizontal) dimensions of the input, respectively. F denotes the height and width size of the receptive field; and S represents the stride length. The term $x_{(r+1\times S)(c+j\times S)}$ refers to the input data element with coordinates $(r + i \times S,\ c + j \times S)$, and w_{rc} and b represent the weight positioned on the receptive field and the bias, respectively. σ denotes the nonlinear activation function used to extract the features from the input.

Within the convolutional layer, the input size $(H \times W \times D)$ is reduced to $[(\frac{H-F+2P}{S+1}) \times (\frac{W-F+2P}{S+1}) \times K]$, where K denotes the number of filters. This process gradually decreases the dimension as the convolution layer stack becomes deeper. The pooling layer has two main functions: reduce the spatial dimension of the input layer by (typically) up to 75% and control overfitting.

C. LSTM Neural Networks

The LSTM is a variant of the traditional RNN that preserves the temporal dimension of sequential data by connecting neurons to and form a network that is a direct cycle of the input data. Fig. 2 shows the basic structure of an LSTM memory cell having two distinct components—the long-term state component $c_{(t)}$ and the short-term state component $h_{(t)}$.

The memory cell, as shown in Fig. 2, contains three control gates—input, output, and forget gates—which, respectively, perform write, read, and reset functions in each cell. The multiplicative gates allow the model to store information over long periods, thereby eliminating the vanishing gradient problem observed in RNNs [21]. The following set of equations represents the description of the input, forget, and cell activation, and output, respectively, which enable the LSTM to predict the output vector:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \tag{4}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \tag{5}$$

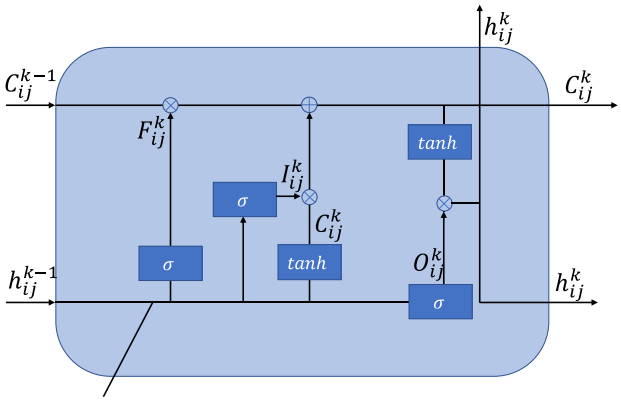

Fig. 2. Basic architecture of the LSTM memory cell.

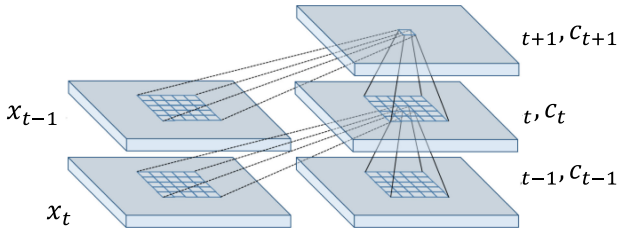

Fig. 3. Inner structure of the ConvLSTM.

$$c_t = f_t c_{t-1} + i_t g\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{6}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \tag{7}$$

$$h_t = o_t h\left(c_t\right) \tag{8}$$

where W and b represent the weight matrix and bias vector, respectively, and $\sigma(.)$ denotes a standard logistic sigmoid function. The variables i, f, o, and c are the input gate, forget gate, output gate, and cell activation vector, respectively.

D. Convolutional LSTM

A convolutional LSTM or convLSTM model is a variant of the LSTM that replaces the FC layer operators with convolutional operators [17]. It uses convolution operators for the input-to-hidden and hidden-to-hidden connections. Within the LSTM memory cell, the $\oplus$ and $\otimes$ operators refer to the matrix addition and dot product operators, respectively. Therefore, by replacing the convolution operators with an LSTM memory cell, the ConvLSTM is able to know what information is to be 'remembered' or 'forgotten' from the previous cell state, using its forget gate. Similarly, the ConvLSTM also decides what information is to be stored in the present cell state. The process of the ConvLSTM is described in a similar manner to (4)–(8) used for the LSTM memory cell computation. However, the difference with the ConvLSTM is that the input vector x_t is fed as images (i.e., 2-D or 3-D matrices) with every weight in the connection replaced by convolution filters. In the ConvLSTM, the transition between the states (from time steps) is analogous to the movement between the frames. Fig. 3 shows the visualization of the inner structure of the ConvLSTM. The input matrix at time

x_t is used to compute the hidden present state h_t at time t and takes into account the hidden state at the previous and next time step h_{t-1} and h_{t+1}, respectively.

E. Autoencoder

The autoencoder is a feedforward neural network in which the input is the same as the output. In other words, autoencoders are (unsupervised) learning algorithms that extract features from input data without the need for labeled target datasets. The autoencoder consists of three basic components: the encoder, the code, and the decoder. These function according to their literal meanings. The encoder compresses the input to a 'code,' which is subsequently decoded by the decoder. For this reason, the autoencoder can be used as a dimensionality reduction strategy in time-series forecasting as it can compress the input to a mapped hidden layer [12]. The stacked autoencoder is a hierarchically layered stack of autoencoders and, just like autoencoders, they learn in an unsupervised manner. The model training process involves greedy layer-wise training to minimize the error between the input and output vectors. The subsequent layer of the autoencoder is the hidden layer of the previous one, with each of the layers trained by gradient descent algorithm using an optimization function.

III. 2-DConvLSTMAE Model

In this section, the proposed deep ConvLSTM autoencoder model for univariate time-series forecasting of machine speed in a smart factory is presented.

A. ConvLSTM Encoder

Although the ConvLSTM has been applied in time-series classification for anomaly detection using video sequences [22], its performance is known to deteriorate with an increase in sequence length. To overcome this limitation, the ConvLSTM applies an attention-based mechanism, which adaptively determines and retains the relevant hidden states across the time steps. Therefore, (4)–(8) are rewritten as

$$i^{t,l} = \sigma\left(W^l_{xz}x^{t,l} + W^l_{xz}h^{t-1.l} + W^l_{xz} \circ c^{t-1.l} + b^l_z\right) \quad (9)$$

$$f^{t,l} = \sigma\left(W^l_{xr}x^{t,l} + W^l_{hz}h^{t-1.l} + W^l_{cr} \circ c^{t-1.l} + b^l_r\right) \quad (10)$$

$$c^{t,l} = i^{t,l} \circ \tanh\left(W^l_{xc}x^{t,l} + W^l_{hc}h^{t-1.l} + b^l_c\right) + r^{t,l} \circ c^{t-1.l} \quad (11)$$

$$o^{t,l} = \sigma\left(W^l_{xo}x^{t,l} + W_{ho}h^{t-1.l} + W^l_{co} \circ c^{t.l} + b^l_o\right) \quad (12)$$

$$h_t = o^{t,l} \circ \tanh\left(c^{t,l}\right) \quad (13)$$

where $\circ$ represents the Hadamard product, σ represents the sigmoid function, $W^l_{xz}, W^l_{xz}, W^l_{xr}, W^l_{cr}, W^l_{xc}, W^l_{hc}, W^l_{xo}, W_{ho}$, and $W^l_{co} \in \mathbb{R}^{n\times T}$ represent the convolutional kernels within the model, while b^l_z, b^l_r, b^l_c, and b^l_o are the bias parameters in the lth layer of the ConvLSTM. Fig. 4 represents the summary architecture of the proposed 2DConvLSTMAE model. In our model, the ConvLSTM layers have 128 and 64 filters, respectively, with the kernel sizes of (1×3). The ConvLSTM layers are arranged in a layered structure to extract the temporal features hierarchically. In the ConvLSTM, the length of sequences is a hyperparameter that affects model performance, and hence must be optimized. The optimal length of sequences (i.e., the number of the previous sequence segments) was determined using a grid search framework (see Section III-C) as 20, with three of these lengths (i.e., three subsequences each of length 20) used in the training regime. The output consists of an FC network of ten units (corresponding to the ten multistep outputs or predictions).

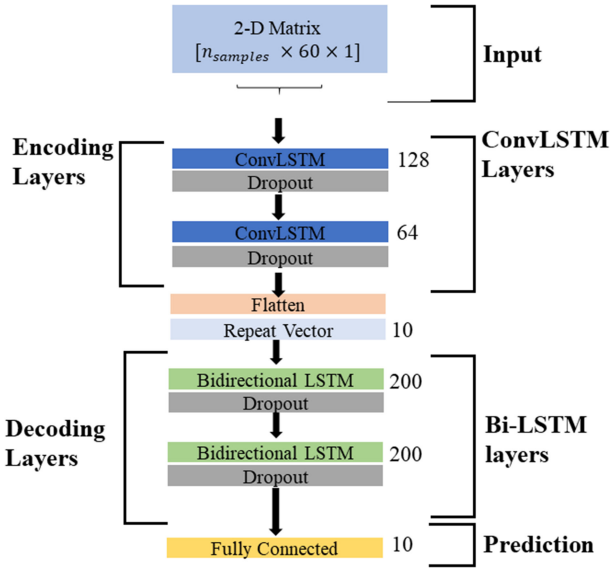

Fig. 4. Model architecture of 2-DConvLSTMAE.

B. Bidirectional LSTM Decoder

The output of the encoder phase of our model is a series of feature map vectors of dimension $(n \times 1 \times 8 \times 64)$, where n represents the number of training samples used. To decode the feature maps obtained from the encoder layers, a repeat vector layer is applied. The main function of the layer is to 'repeat' the final output vector from the encoding layer in a shape that is a constant input to each time step of the decoder. In this way, the decoding layer is able to reconstruct the original input sequence. The output of this repeat vector layer is passed onto a layered bidirectional LSTM stacked network (see Fig. 4). Each LSTM layer is made of 200 LSTM units, with rectified linear unit activation applied. The output of the previous LSTM layer is fed into the next layer as input in a hierarchical manner. In this way, the decoder layer can incorporate the encoded output vector from the ConvLSTM encoder, which improves the performance of the predictive model by fostering representation learning at the individual layers [12].

C. Hyperparameter Optimization

The performance of DL models depends on predetermined hyperparameters, which are obtained using an optimization process. Unlike model parameters, which are learned using an optimization function to minimize an objective (or loss) function, hyperparameters are not learned during the model training

TABLE I
HYPERPARAMETERS OF THE 2-DCONVLSTMAE MODEL

Layer	Hyper-parameter(s)	Value
ConvLSTM1	Filter	16, 32, 64, 128, 256, 512, 1024
	Kernel Size	(1 x 3), (1 x 4), (1 x 8), (1 x 12)
ConvLSTM2	Filter	16, 32, 64, 128, 256, 512
	Kernel Size	(1 x 3), (1 x 4), (1 x 8), (1 x 12)
LSTM 1	Units	100, 200, 300, 400, 500, 600
LSTM 2	Units	100, 200, 300, 400, 500
N/A	Dropout	0.1, 0.2, 0.3, 0.4, 0.5
N/A	Learning Rate	1e-4, 1e-5, 1e-6, 1e-7
N/A	Batch Size	8, 16, 32, 64, 128, 256, 512
N/A	Optimiser	Adam, SGD, Adadelta, RMSprop, Nadelta, Nadam, RAdam

process. Several hyperparameters exist for DL models, and for the model presented in this article, eight hyperparameters were optimized, as presented in Table I.

Many hyperparameter optimization methods exist, such as random search, grid search, and Bayesian optimization. However, for this article, we applied a grid search framework for the hyperparameter optimization [23] of both the proposed 2-DConvLSTMAE and the individual baseline/competitor model hyperparameters. The grid search approach is chosen due to its reliability in low-dimensional spaces such as this present study [24] in comparison to manual search. Second, the grid search method is simple to implement and parallelization can easily be configured. The hyperparameter optimization method adopted in this article is described as follows: Given a set $\forall$, which has an index of n possible configuration hyperparameters h. The grid search requires the selection of a set of values for each hyperparameter $(h^1 \cdots h^k)$ that minimizes the validation loss. The grid search algorithm assembles all combinations of values in a 'grid' format, such that the number of trials in a grid search is $S = \prod_{n=1}^{n} |h^{(k)}|$. Table I tabulates the hyperparameter search space applied within the grid search framework to optimize the model hyperparameters. The other benchmark models included in this article have been optimized using a similar approach.

D. Optimizer

Within DL models, the process of optimizing model parameters is typically performed using stochastic gradient-based optimization algorithms. A number of optimization algorithms exist for use in DL models, such as RMSProp [25], AdaGrad [26], and vSGD [27]. In our network, a stochastic gradient-based optimization algorithm—Adam [28]—is used. The learning rate value determined by the grid search framework was 1×10^{-6}.

E. Loss Function

For this article, the loss function is the root-mean-square error (RMSE). Therefore, the RMSE over the training data is calculated and backpropagated to update the model parameters with each iteration (epoch). RMSE is described as

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y_i} - y_i)^2} \tag{14}$$

where $\widehat{y_i}$ and y_i, respectively, represent the predicted values and target variable, and n represents the sample size. We applied minibatch stochastic gradient descent using the optimizer described in Section III-D to minimize this loss. After training through a number of iterations (epochs), the network parameters are then used to predict the next ten time steps, given a sequence of input prior observations.

IV. EXPERIMENTS AND RESULTS

In this section, we report the experimental setup and evaluation process of the proposed DL sequence-to-sequence time-series predictive model for machine speed prediction. To empirically evaluate the performance of the 2-DConvLSTMAE model, historical data are used from a bodymaker machine.

A. Data Preparation

The dataset used for this article contains historical machine-collected speed data obtained at a frequency of $\frac{1}{60}$ Hz. The data represent the operational speed of a high-speed aluminum can-making machine, measured as the number of strokes per minute and logged internally by the machine. This bodymaker machine is employed in metal can manufacturing to produce the full length can body from a small (metal) cup that is forced through a series of iron rings [18]. The operational speed of the bodymaker machine, which influences production throughput and yield, typically exhibits a mixture of periodic patterns linked to "normal" production schedules and episodic, sporadic patterns due to abnormal operations. In addition, this machine impacts (and is impacted by) the upstream and downstream processes, such as the cupper (upstream) and the can cleaning/sterilizing (downstream) machines.

In this article, the dataset contained 525 600 observations of minute-wise machine speed (strokes/min) within a date range from $31/08/2017\ 00:00$ to $30/08/2018\ 23:59$. From this dataset, 463 978 were used for model training, 51 553 for testing, while 10 000 were validation observations. As described in Section II-A, the data must be transformed into a format that is suitable for (sequential) supervised learning. In this article, a sliding window of size $w = 60$ and a recurrent step size of one are used (see Fig. 5). The prediction interval is chosen as $k = 10$ min. Therefore, the training data were transformed from a univariate input sequence $x(t) = \{x_1, x_2, x_3, \ldots, x_N\}$ of shape $(N \times 1)$ to a matrix of shape $(n \times 60 \times 1)$ from the trajectory of the univariate input time-series data, where N is the total number of observations in the dataset (i.e., 525 600) and n refers to the training dataset size $n = (N - k - w + 1) = 525\,531$.

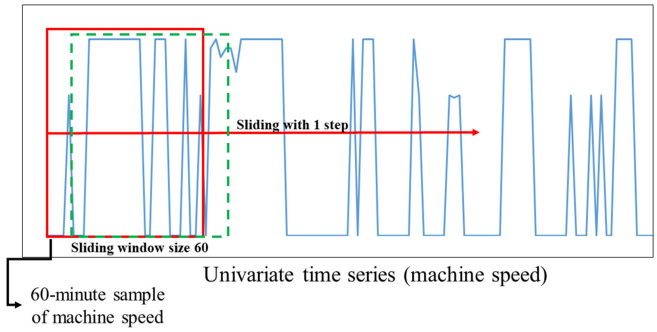

Fig. 5. Transforming input time series to samples.

B. Baseline Models

In order to evaluate the performance of the 2-DConvLSTMAE model for multistep machine speed prediction, we compare the performance against the naïve, statistical, and three state-of-the-art DL baseline models.

1) Persistence Model: The persistence model, a widely used benchmark model for time-series forecasting, operates on the assumption that the predicted value of the target variable remains unchanged from the previous time lag. In other words, the predicted value at time t, $\widehat{y}_t = y_{t-1}$ for all times. This naïve model proves to be highly accurate especially in short-term forecasting but exhibits vulnerabilities in multistep prediction [29].

2) Autoregressive Integrated Moving Average: ARIMA is a well-known time-series forecasting model. The main assumption of ARIMA is the stationarity of the mean and variance, and that there exists a linear relationship between the lags (i.e., past observations) and the future state, which constitutes a limitation.

3) Residual-Squeeze Net (RSNet): The RSNet proposed in [15] comprises of 1-D CNNs using the RSNet architecture via the squeeze operation that fuses the information using an optimal combination of channels learned during the model training. We apply a single-channel input data as described above to train the model.

4) Deep LSTM Encoder–Decoder: We use the model presented in [19], which is a stacked architecture of LSTM layers connected to a time-distributed dense layer.

5) CNN-LSTM Encoder–Decoder: A CNN-LSTM autoencoder model architecture presented in [30] is the third baseline model. The work presented a classifier, but the model was modified to include a regression layer.

C. Model Performance Evaluation

In terms of model evaluation, we adopted a technique referred to as walk-forward validation or backtesting. The traditional prediction evaluation methods, such as k-fold cross validation or train-test splitting do not work well when applied to time-series data because these evaluation methods assume that there is no relationship between the observations, which is not the case with time-series data, where the sequential dimension needs to be preserved.

TABLE II
PERFORMANCE EVALUATION OF PREDICTIVE MODELS USING WINDOW SIZE OF 30

Model	RMSE	MAE	sMAPE	Train Time (s)
2D-ConvLSTMAE	**64.93**	**26.53**	**0.759**	6389
CNN-LSTM Encoder-Decoder [30]	76.93	35.65	1.02	8301
Persistence Model	128.94	57.92	1.657	**≈ 0**
ARIMA (2,1,2)	137.40	69.965	2.001	636
LSTM Encoder-Decoder [19]	116.94	94.49	2.704	16776
RsNet [15]	140.97	136.10	3.894	8936

For model evaluation, we applied three error evaluation metrics—RMSE, mean absolute error (MAE), and symmetrical mean absolute percentage error (sMAPE), which are defined by (14)–(16), respectively

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y}_i - y_i| \tag{15}$$

$$\text{sMAPE} = \frac{200}{n}\sum_{i=1}^{n}\frac{|\widehat{y}_i - y_i|}{|\widehat{y}_i| + |y_i|}. \tag{16}$$

D. Implementation Environment

The experimental environment used for this article was on a single machine with Intel Xeon E-2146G CPU @ 3.50 GHz, 128-GB memory and NVIDIA Tesla V100-PCIE 16GB GPU. The GPU is used for accelerated model training due to a large computation demand in DL models. The development was performed using Python 3.6.8 and Tensorflow 1.12.0.

E. Comparison With Baselines

To empirically evaluate the performance of the individual models, we test the models using window sizes of 60 and 30, respectively, and the evaluation metrics described in Section IV-C. Table II and Fig. 6 represent the results for the empirical analysis of the predictive models trained using a window of size 30. As the results show, 2-DConvLSTMAE outperformed the baselines in all evaluation metrics. From Table II, it can be seen that our model, in addition to displaying superior predictive accuracy, took the least training time in comparison to the DL models. It must be mentioned that although the ARIMA and persistence models took significantly lower training times, they, however, performed worse than the 2-DConvLSTMAE (see Table II). Generally, naïve models (such as the persistence model) are used to benchmark the performance of predictive models. Consequently, a model that outperforms a naïve model is considered as 'skillful' in time-series forecasting [12].

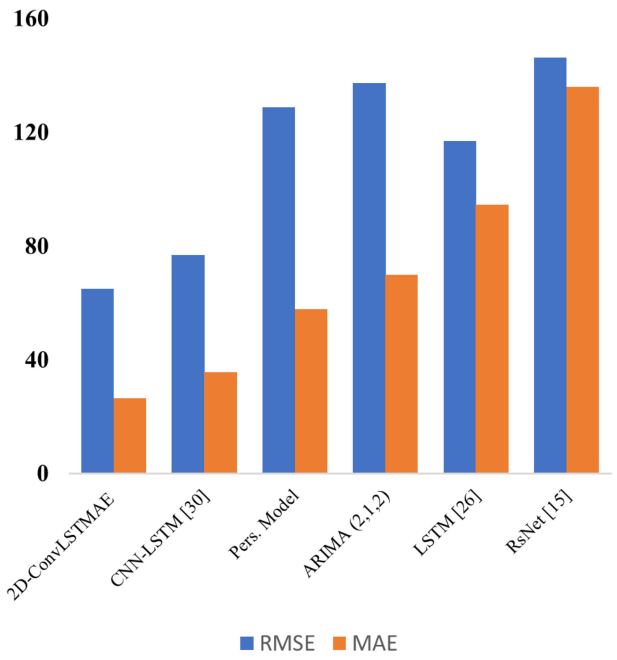

Fig. 6. Performance comparison of baseline models using window size 30.

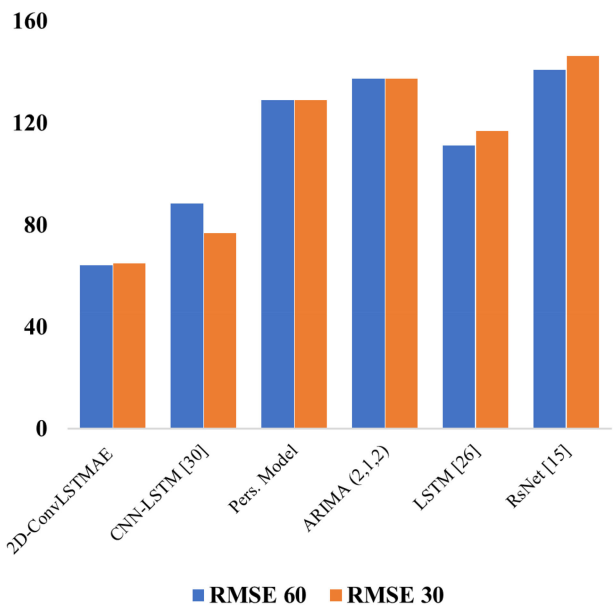

Fig. 7. Performance of models using both window sizes, respectively.

TABLE III
PERFORMANCE EVALUATION OF PREDICTIVE MODELS USING WINDOW SIZE OF 60

Model	RMSE	MAE	sMAPE	Train Time (s)
2D-ConvLSTMAE	**64.23**	**31.70**	**0.903**	8017
CNN-LSTM Encoder-Decoder [30]	88.54	41.03	1.170	8301
Persistence Model	128.94	57.92	1.652	**≈ 0**
ARIMA (2,1,2)	137.40	69.965	1.995	636
LSTM Encoder-Decoder [19]	111.27	89.91	2.561	30318
RsNet [15]	146.24	136.10	3.881	8936

To test the proposed model for robustness in larger window sizes, we used observations from the prior 60 min as input features (i.e., window size $w = 60$). The results of this analysis are presented in Table III. Note that the ARIMA and persistence models are not affected by the window size, since they do not require prior data manipulation or transformation. As can be seen from Table III, the proposed 2-DConvLSTMAE model also outperformed the state-of-the-art DL models in all the evaluation metrics. Furthermore, the proposed model required the lowest training time (in comparison to the DL baselines), which makes it applicable to industrial process modeling. Fig. 7 shows the bar plots of the individual errors (RMSE) for each of the predictive models. As can be seen, there is robustness in our proposed model, as the performance does not significantly deteriorate with a change in the window size. However, as can be seen from Tables II and III, the optimal window size for 2-DConvLSTMAE is 60.

Figs. 8 and 9 both show the respective DL model predictive performances for the first 200 time steps for window size $w = 30$ and $w = 60$, respectively. For each of the subplots, the x-axis represents the time steps, while the y-axis represents the value of the machine speed (for the upper half of the subplot) and the absolute error between the predicted and actual values (for the lower part of the subplot), respectively. From the figures, it can be seen that the 2-DConvLSTMAE significantly outperformed the state-of-the-art benchmark DL models both in the shorter- (i.e., 30 prior inputs) and longer-term window size training regimes. Also, it can be observed from Fig. 8 that the LSTM encoder–decoder [19] model performs better than the other baseline models, given that the LSTM memory cells better capture the long-term dependencies in the sequential dataset than the other baseline models. This ability to capture the long-term dependencies constitutes the main prospect and potential for the application of ConvLSTM deep networks on large time-series dataset from manufacturing operations. It can be seen from Figs. 8 and 9 that the predictive performance of the 2-DConvLSTMAE is robust to the machine speed variance and temporal distribution, evidenced in the superior predictive performance using 30 and 60 prior observations, respectively [see Figs. 8(d) and 9(d)].

The effectiveness of the 2-DConvLSTMAE can be rationalized from the following standpoints. First, the deep ConvLSTM layers adequately capture the spatial and temporal distribution of the sequential data, leveraging the advantages of both the LSTM and CNN models at sequential and automatic feature extraction, respectively. As the results have shown (see Tables II and III; Figs. 7–9), the 2-DConvLSTMAE captures the temporal patterns hidden in the machine speed signals. Second, the encoder–decoder architecture doubles as a dimensionality

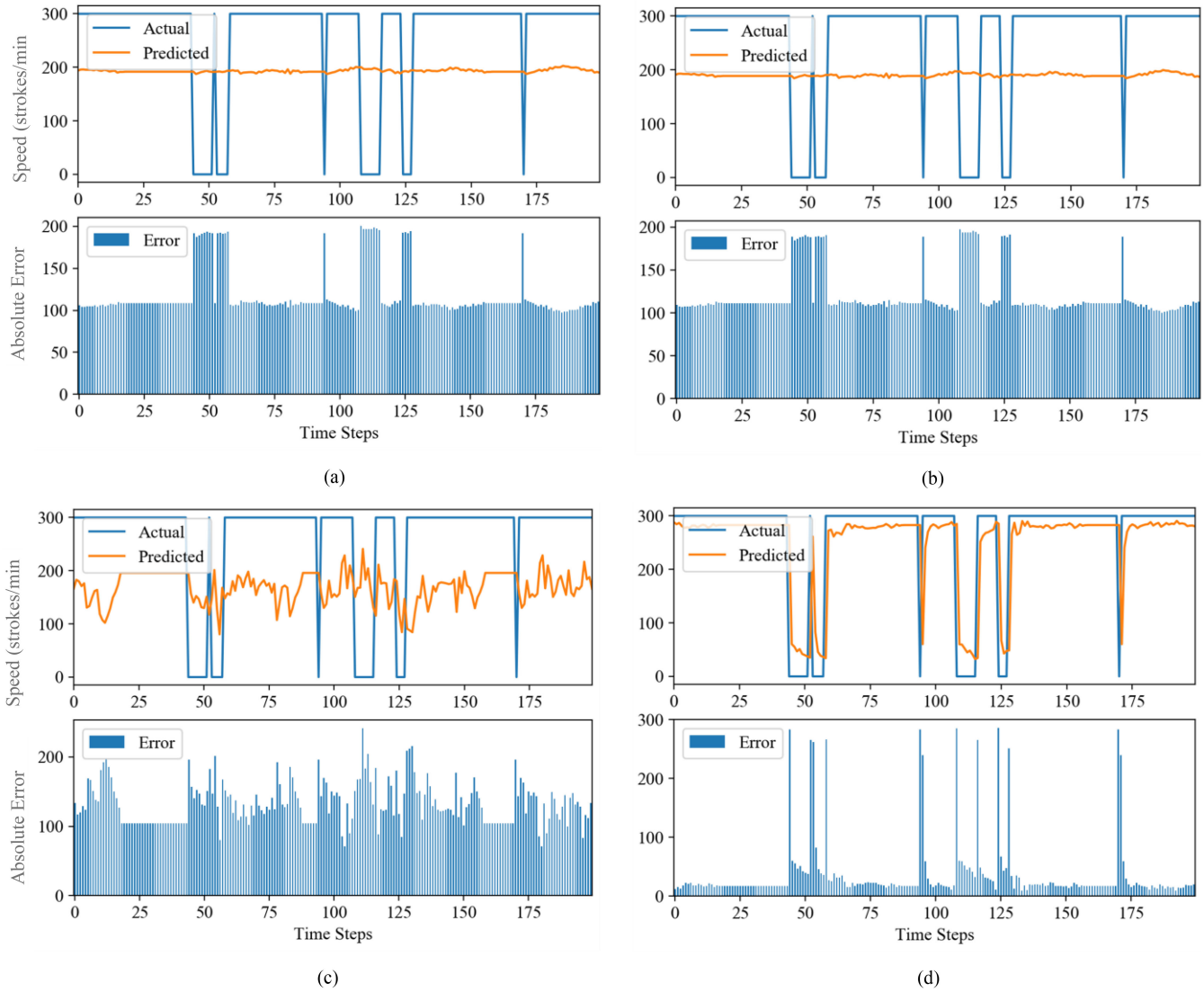

Fig. 8. Predictive performance of models on validation partition trained using a window size of 30. (a) RsNet [15]. (b) CNN-LSTM-SAE [30]. (c) LSTM encoder–decoder [19]. (d) 2-DConvLSTMAE.

reduction technique and an unsupervised learning regime, fostering representation learning, and, thereby, reducing model training time, simultaneously. The combination of the approaches—stacking ConvLSTM and LSTM layers in an encoder–decoder architecture—toward the machine speed prediction resulted in better performance (reduction in training time and prediction error) for the univariate time-series prediction of machine speed.

F. Analysis of Sensitivity of 2-DConvLSTMAE With Respect to Sequence Length

As previously stated in Section II-D, the sequence length in the ConvLSTM is a hyperparameter that needs to be optimized externally. Consider the input to the 2-DConvLSTMAE model, $X \in \mathbb{R}^{n \times w}$, which is further segmented into p subsequences. These subsequences have a uniform length such that $l = \frac{w}{p}$, where l is the subsequence length and w denotes the sliding-window size (i.e., 60 for this current study—see Section IV-A). In order to obtain an optimal l, we performed a sensitivity analysis of this hyperparameter. For this experiment, we set the subsequence length l to 60, 30, 20, 12, and 10, respectively, to test the impact of these on the predictive error and the model training time. The other model hyperparameters remained the same as before (see Table I). Fig. 10 shows the results of the performance of the 2-DConvLSTMAE with the different subsequence lengths. It can be seen that the optimal l is 20 (red dotted line), as the configuration resulted in the lowest RMSE and training time, respectively. It must be mentioned here that the optimal sequence length will depend on the particular application, which may be linked to individual production cycles and different data distributions.

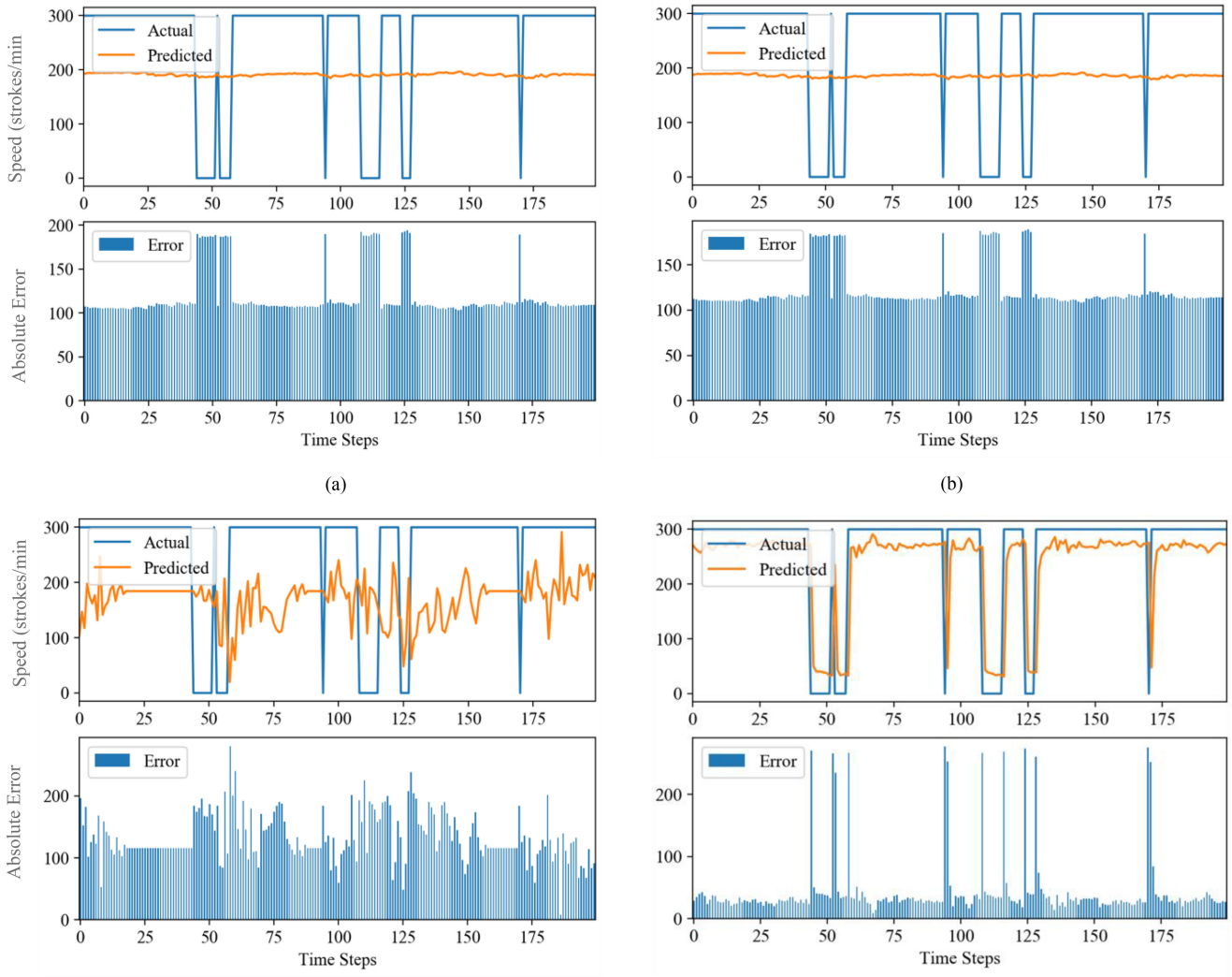

Fig. 9. Predictive performance of models on validation partition trained using window size of 60. (a) RsNet [15]. (b) CNN-LSTM-SAE [30]. (c) LSTM encoder–decoder [19]. (d) 2-DConvLSTMAE.

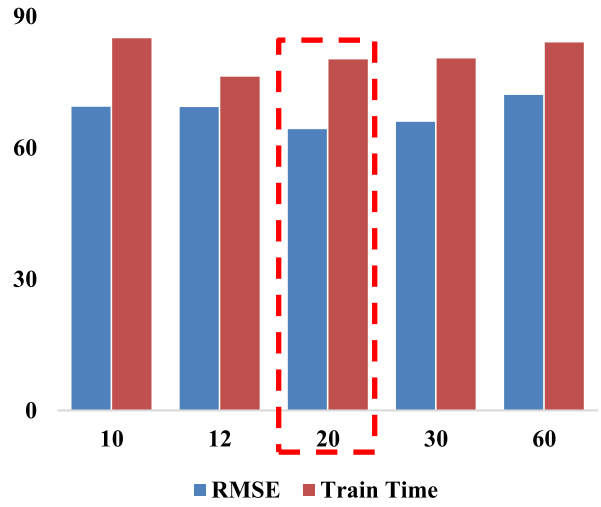

Fig. 10. Sensitivity analysis of impact of subsequence length on accuracy and training time.

V. Conclusion

In this article, a novel deep ConvLSTM autoencoder architecture has been proposed for machine speed prediction in a smart manufacturing process. By restructuring the input sequence to a supervised learning manner using a sliding-window approach, the predictive model—2-DConvLSTMAE—was applied to the multistep time-series forecasting problem. 2-DConvLSTMAE leveraged the advantage power of the CNN in automatic feature extraction and the LSTM for sequential and representation learning. In the proposed model, the encoder–decoder architecture, which doubles as a dimensionality reduction technique, promoted representation learning in the model training regime, resulting in the reduced computational demand and training time. The results from empirical analyses showed that

1) 2-DConvLSTMAE outperformed the naïve and statistical benchmark models as well as three (3) state-of-the-art DL time-series models, achieving improved predictive performance;

2) in addition to improved predictive performance, our model required a significantly lower model training time. This makes 2-DConvLSTMAE not only better in machine speed prediction but also a more practical approach for adoption in real manufacturing processes.

The results obtained from this article can be directly applied to multistep time-series forecasting for smart manufacturing processes operations, enabling the improvement of production scheduling and planning. For instance, predicting the machine speed in advance can be used to foster just in time production by providing an indication of future production output so that the operational requirements can be adjusted accordingly. Future extension of this article includes extending the proposed model to multivariate time series including machine states and external sensors data.

References

[1] H. S. Kang *et al.*, "Smart manufacturing: Past research, present findings, and future directions," *Int. J. Precis. Eng. Manuf.-Green Technol.*, vol. 3, no. 1, pp. 111–128, 2016.

[2] C. Giannetti, R. S. Ransing, M. R. Ransing, D. C. Bould, D. T. Gethin, and J. Sienz, "A novel variable selection approach based on co-linearity index to discover optimal process settings by analysing mixed data," *Comput. Ind. Eng.*, vol. 72, no. 1, pp. 217–229, Jun. 2014.

[3] R. S. Ransing, C. Giannetti, M. R. Ransing, and M. W. James, "A coupled penalty matrix approach and principal component based co-linearity index technique to discover product specific foundry process knowledge from in-process data in order to reduce defects," *Comput. Ind.*, vol. 64, no. 5, pp. 514–523, Jun. 2013.

[4] L. Wang, Z. Zhang, H. Long, J. Xu, and R. Liu, "Wind turbine gearbox failure identification with deep neural networks," *IEEE Trans. Ind. Informat.*, vol. 13, no. 3, pp. 1360–1368, Jun. 2017.

[5] B. Cai, L. Huang, and M. Xie, "Bayesian networks in fault diagnosis," *IEEE Trans. Ind. Inform.*, vol. 13, no. 5, pp. 2227–2240, Oct. 2017.

[6] C. Giannetti and R. S. Ransing, "Risk based uncertainty quantification to improve robustness of manufacturing operations," *Comput. Ind. Eng.*, vol. 101, pp. 70–80, 2016.

[7] T. Wuest, D. Weimer, C. Irgens, and K.-D. Thoben, "Machine learning in manufacturing: Advantages, challenges, and applications," *Prod. Manuf. Res.*, vol. 4, no. 1, pp. 23–45, 2016.

[8] M. Alipour, B. Mohammadi-Ivatloo, and K. Zare, "Stochastic scheduling of renewable and CHP-based microgrids," *IEEE Trans. Ind. Informat.*, vol. 11, no. 5, pp. 1049–1058, Oct. 2015.

[9] C.-H. Wu, C.-C. Wei, D.-C. Su, M.-H. Chang, and J.-M. Ho, "Travel time prediction with support vector regression," in *Proc. IEEE Int. Conf. Intell. Transp. Syst.*, 2003, pp. 1438–1442.

[10] A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio, "The impact of rainfall and temperature on peak and off-peak urban traffic," in *Proc. Int. Conf. Database Expert Syst. Appl.*, 2018, pp. 399–407.

[11] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *J. Manuf. Syst.*, vol. 48, pp. 144–156, 2018.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, vol. 13. Cambridge, MA, USA: MIT Press, 2016.

[13] C. Kim, J. Lee, R. Kim, Y. Park, and J. Kang, "DeepNAP: Deep neural anomaly pre-detection in a semiconductor fab," *Inf. Sci.*, vol. 457–458, pp. 1–11, 2018.

[14] C. Wu, P. Jiang, C. Ding, F. Feng, and T. Chen, "Intelligent fault diagnosis of rotating machinery based on one-dimensional convolutional neural network," *Comput. Ind.*, vol. 108, pp. 53–61, 2019.

[15] L. Su, L. Ma, N. Qin, D. Huang, and A. H. Kemp, "Fault diagnosis of high-speed train Bogie by residual-squeeze net," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 3856–3863, Jul. 2019.

[16] A. Essien and C. Giannetti, "A deep learning framework for univariate time series prediction using convolutional LSTM stacked autoencoders," in *Proc. IEEE Int. Symp. Innov. Intell. Syst. Appl.*, 2019, pp. 1–6.

[17] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-C. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, 2015, pp. 802–810.

[18] E. Wootton, "TALAT lecture 3710 case study on can making," Eur. Aluminium Assoc., Alcan Deutschland GmbH, Göttingen, Germany, 1994.

[19] M. A. Zaytar and C. El Amrani, "Sequence to sequence weather forecasting with long short-term memory recurrent neural networks," *Int. J. Comput. Appl.*, vol. 143, no. 11, pp. 975–8887, 2016.

[20] Y. Yu, Y. Zhu, S. Li, and D. Wan, "Time series outlier detection based on sliding window prediction," *Math. Problem Eng.*, vol. 2014, 2014, Art. no. 879736.

[21] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber, "Gradient flow in recurrent nets: The difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*. Hoboken, NJ, USA: Wiley, 2001.

[22] Z. Yuan, X. Zhou, and T. Yang, "Hetero-ConvLSTM: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2018, pp. 984–992.

[23] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural Networks: Tricks of the Trade*. New York, NY, USA: Springer, 2012, pp. 9–48s.

[24] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, 2012.

[25] T. Tieleman and G. E. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA, Neural Netw. Mach. Learn.*, vol. 4, no. 2, pp. 26–31, 2012.

[26] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *J. Mach. Learn. Res.*, vol. 12, pp. 2121–2159, 2011.

[27] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," in *Proc. 30th Int. Conf. Mach. Learn.*, 2013, pp. 343–351.

[28] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[29] M. Khodayar, O. Kaynak, and M. E. Khodayar, "Rough deep neural architecture for short-term wind speed forecasting," *IEEE Trans. Ind. Informat.*, vol. 13, no. 6, pp. 2770–2779, Dec. 2017.

[30] C. Giannetti, A. Essien, and Y. O. Pang, "A novel deep learning approach for event detection in smart manufacturing," in *Proc. 49th Int. Conf. Comput. Ind. Eng.*, Beijing, China, 2019.

Aniekan Essien (Member, IEEE) received the B.Eng. degree in computer engineering in 2011, and the M.Sc. degree in information systems from the University of Manchester, Manchester, U.K., where he is currently working toward the Ph.D. degree.

He is currently a Research Assistant with the Future Manufacturing Research Institute, College of Engineering, Swansea University Bay Campus, Swansea, U.K. His research interests include deep learning for time-series forecasting, artificial intelligence for smart manufacturing, and traffic predictive analytics.

Cinzia Giannetti received the Engineering Doctorate degree in manufacturing engineering from Swansea University, Swansea, U.K., in 2015.

She is an Associate Professor with the College of Engineering, Swansea University. She is the holder of EPSRC Innovation Fellowship in digital manufacturing (2018–2021) and coinvestigator with the EPSRC Centre for Doctoral Training in Enhancing Human Interactions and Collaborations with Data- and Intelligence-Driven Systems. Her expertise includes smart manufacturing and sensors technologies, with emphasis on knowledge and information management, data analytics, and machine learning techniques to support decision making through the use of sensors and large-scale databases.

Dr. Giannetti has significant experience in delivering and supporting applied industrial research and development projects gained both in industry and academia. Before joining the academia, she has worked for a decade in industry as senior technical roles in the delivering, planning, and coordinating software development projects for existing systems and new products in industry and academia.