Multivariate Long-Term State Forecasting in Cyber-Physical Systems: A Sequence to Sequence Approach

Nikhil Muralidhar
Computer Science
Virginia Tech
Arlington, USA
nik90@vt.edu

Sathappan Muthiah
Computer Science
Virginia Tech
Arlington, USA
sathap1@vt.edu

Kiyoshi Nakayama
Energy Management
NEC Labs America
San Jose, USA
knakayama@nec-labs.com

Ratnesh Sharma
Energy Management
NEC Labs America
San Jose, USA
ratnesh@nec-labs.com

Naren Ramakrishnan
Computer Science
Virginia tech
Arlington, USA
naren@cs.vt.edu

***Abstract*—Cyber-physical systems (CPS) are ubiquitous in several critical infrastructure applications. Forecasting the state of CPS, is essential for better planning, resource allocation and minimizing operational costs. It is imperative to forecast the state of a CPS multiple steps into the future to afford enough time for planning of CPS operation to minimize costs and component wear. Forecasting system state also serves as a precursor to detecting process anomalies and faults. Concomitantly, sensors used for data collection are commodity hardware and experience frequent failures resulting in periods with sparse or no data. In such cases, re-construction through imputation of the missing data sequences is imperative to alleviate data sparsity and enable better performance of down-stream analytic models.**

In this paper, we tackle the problem of CPS state forecasting and data imputation and characterize the performance of a wide array of deep learning architectures – unidirectional gated and non-gated recurrent architectures, sequence to sequence (Seq2Seq) architectures as well as bidirectional architectures – with a specific focus towards applications in CPS. We also study the impact of procedures like scheduled sampling and attention, on model training. Our results indicate that Seq2Seq models are superior to traditional step ahead forecasting models and yield an improvement of at least 28.5% for gated recurrent architectures and about 87.6% for non-gated architectures in terms of forecasting performance. We also notice that bidirectional models learn good representations for forecasting as well as for data imputation. Bidirectional Seq2Seq models show an average improvement of 17.6% in forecasting performance over their unidirectional counterparts. We also demonstrate the effect of employing an attention mechanism in the context of Seq2Seq architectures and find that it provides an average improvement of 57.12% in the case of unidirectional Seq2Seq architectures while causing a performance decline in the case of bidirectional Seq2Seq architectures. Finally, we also find that scheduled sampling helps in training better models that yield significantly lower forecasting error.

***Index Terms*—Cyber-Physical Systems, Deep Learning, Sequence to Sequence Models, Long-term time series forecasting, Data Imputation**

I. Introduction

Cyber-physical systems (CPS) represent a closely coupled relationship between software systems and physical processes controlled through actuators. The data relating to process operation characteristics is logged through the Supervisory Control and Data Acquisition (SCADA) system which is a protocol that provides process monitoring capabilities. In addition to process monitoring ('nowcasting'), it is also important to predict future system states ('forecasting') of complex CPS to enable better planning, system process optimization and resource allocation. For example, it is important for power generation stations to accurately forecast far into the future to decide the level of power generation to account for peak demand . The US National Academy of Engineering has highlighted the importance of sustainable and blackout-free electricity generation and distribution, [2]. An important facet for sustainable blackout-free electricity generation is the effective long-term forecasting of peak energy demand.

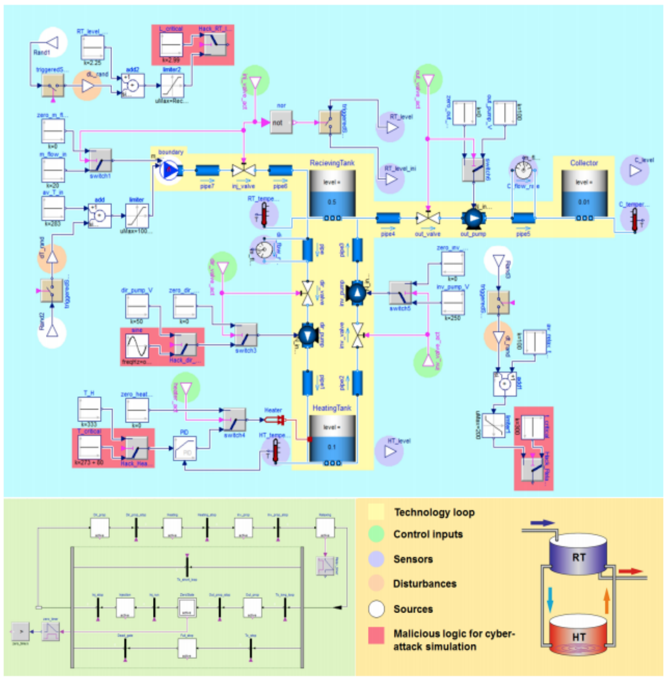


Fig. 1: **Model representing the Gasoil Heating Loop CPS presented by Filonov et al. [1][1]**

CPS forecasting models also serve as a pre-requisite for anomaly detection systems [3] in which the anomaly detector attempts to detect significant deviations of the system operational state from previously forecast system states. Another important application of CPS forecasting systems is for predicting time to failure or predicting the wear of components in a system (a.k.a Remaining Usable Life or RUL prediction). Forecasting systems can use data from previous instances of failure of components in the system to predict when the component will fail next [4].

The proliferation of low-cost sensors and availability of low cost, low power, computing devices coupled with the communication network revolution has lead to CPS systems becoming ubiquitous in contemporary times [5]. However, these low cost hardware devices quite often fail, leading to some periods of system operation being free of process monitoring and consequently without any data logging. These disruptions in data collection could be due to sensor faults, communication malfunctions, environmental interference or human error [6]. Data that is representative of full system operation is imperative for effectively training forecasting models to learn the complete spectrum of the characteristics of the CPS. It is with this goal of reinstating data integrity and completeness of CPS datasets that we propose a bi-directional Seq2Seq (**Bi-Seq2Seq**) deep learning model for forecasting immediate and long-term state of a CPS as well as for imputing periods of missing data from the CPS data logs through sequence reconstruction. Our contributions are as follows:

- We develop a bidirectional Seq2Seq model (**Bi-Seq2Seq**) capable of producing short and long-term forecasts of system operational states of a CPS.
- We characterize the performance of the **Bi-Seq2Seq** model as well as other deep learning models in short and long-term forecasting of the state of a CPS.
- We augment **Bi-Seq2Seq** to perform data imputation to reconstruct missing data sequences in the CPS data logs and evaluate reconstruction performance.

We demonstrate the effectiveness of Seq2Seq models on multivariate time series state forecasting tasks using the Gasoil heating loop (GHL) CPS dataset.

II. Dataset Description

The GHL model as detailed in [1] and depicted in Fig. 1, is used to generate the dataset and consists of three tanks, receiving tank (RT), heating tank (HT) and collection tank (CT). The goal of the GHL model is to heat gasoil from RT to a temperature of 60° Celsius so that it reaches a viscosity wherein it can be transferred to CT. At each state, a portion of the gasoil from RT is transferred into HT, heated to the requisite temperature and transferred back into RT. This process of heating a small portion of the gasoil from RT is carried on until the entire gasoil in RT reaches the requisite temperature. The payload in RT is then transferred over into CT. The empty RT is re-filled with a fresh batch of gasoil from an external resource and the process is repeated. This process can be carried out for any fluid but in the experiment in [1], the fluid chosen was water.

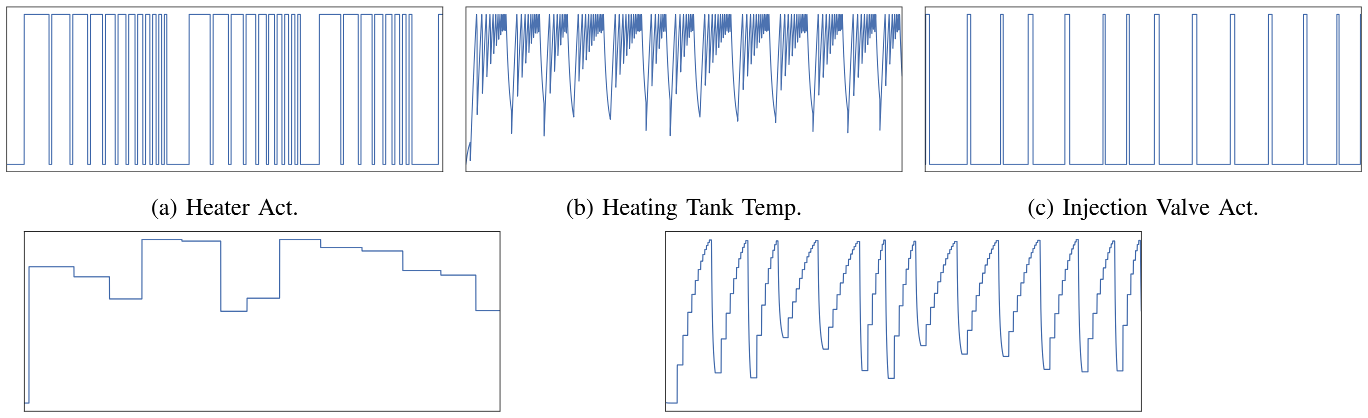
The dataset generated is a multi-variate time-series consisting of 19 variables corresponding to the different components in the GHL cyber-physical system. Fig. 2 represents a visualization of time-series generated by a subset of components that we use in our experiments and it helps to comprehend the complexity of the modeling task at hand. Fig. 2d represents the fluid level of RT at different time steps. We can observe that the time series have weak auto-correlation and show more of a slow changing trendy behavior interspersed with sudden increasing or decreasing steps. Fig. 2b represents the HT temperature over time and the variation in temperature can be observed as being a highly non-linear process with a cyclic pattern. The time series in Fig. 2a and Fig. 2c show more of a discrete nature in that each time series assumes only one of two values indicating a switching mechanism. Hence, the generated dataset has complex nonlinear, non-stationary characteristics. It is non-trivial for traditional auto-regressive models to learn effective representations in such non-linear, non-stationary scenarios as they require certain stationarity assumptions to be satisfied to effectively model sequential data. Hence, we adopt recurrent deep learning architectures to address this problem of sequence forecasting for CPS.

III. Related Work

Time Series Forecasting Methods: Time series forecasting is a well researched topic in many fields and there have been a variety of models proposed like the popular auto-regressive models (AR, MA, ARIMA) as well as state-space models like Kalman and particle filters [7]. Such traditional models however, require certain domain rules governing process state transition (as in the case of Kalman filters) to be known or others like the auto-regressive set of models require certain assumptions made regarding the properties of the time series themselves to hold (e.g. stationarity). In our application, we employ a highly non-linear, non-stationary real world multivariate time series dataset of a cyber physical system. The data is described in Section II.

Deep Learning for Sequence Modeling: Sequence to sequence models have been extremely popular in the natural language processing domain. They were initially proposed for the sequential data modeling task of neural machine translation (MT) [8]–[11]. Since then, multiple variants of encoder decoder models have been employed for machine translation and natural language tasks like abstractive text summarization [12]–[19]. Sequence to sequence models have also been successfully used for speech recognition tasks [20]–[23].

In [1], [24], [25] the authors employ sequence to sequence deep learning models for performing forecasting and use the forecasts to detect anomalies in the operation of CPS. In [4], the authors use convolution and pooling in convolutional neural networks along the temporal dimension over multi-channel CPS sensor data to incorporate automated feature



(a) Heater Act. (b) Heating Tank Temp. (c) Injection Valve Act.

(d) Receiver Tank Level. (e) Receiver Tank Temp.

Fig. 2: **Temporal characteristics of the different time-series used for evaluating models. We can see that the time-series in Fig. 2a and Fig. 2c correspond to turning the heater and injection valve on respectively. These two time-series have a switching nature. The heating tank temperature and receiver tank temperature sensors exhibit a highly non-linear cyclic behavior. Finally, the receiver tank level shows a slowly changing trending behavior.**

learning from raw sensor signals in a systematic way to perform Remaining Useful Life (RUL) estimation.

Data Imputation Methods: Most of the data imputation research adopts a clustering or nearest neighbor approach. We outline the most recent work on data imputation [6], in which the authors employ stacked autoencoders for performing nearest neighbor based data imputation for missing CPS data. However, their system is capable solely of imputation.

In this paper, we propose and characterize the performance of a bi-directional sequence to sequence (**Bi-Seq2Seq**) architecture capable of performing long-term forecasts of the operational states of a CPS as well as re-constructing missing data sequences in the CPS data logs through deep data imputation techniques.

The rest of the paper is organized as follows: section IV discusses some requisite background information while the forecasting and data imputation problems are formally stated in section V. The experimental setup is described in section VI, followed by discussion of experimental results in section VII and conclusion in section VIII.

IV. Background

Recurrent neural networks (RNN) are components that are variants of traditional feed forward neural networks, able to handle variable sequence length inputs. Traditional RNNs are different from feed forward networks as they are able to take advantage of sequential information in the inputs as opposed to treating inputs as independent from each other. An RNN consists of a vector $\mathbf{h_t} \in \mathbb{R}^{h\times 1}$ where $\mathbf{h_t}$ denotes the hidden state at time t. Eq. 1 shows how each hidden state is computed as a function of the previous hidden state and the input x_t at time t. $g(\cdot)$ is usually a non-linear function like a sigmoid or a hyperbolic tangent. Let $\mathbf{x}_t \in \mathbb{R}^{n\times 1}$ denote a multi-variate input vector to the RNN at time t. Then, the output of the RNN at time t is $\hat{\mathbf{x}}_t \in \mathbb{R}^{n\times 1}$.

$$\begin{aligned} \mathbf{h_t} &= g(\mathbf{U}\mathbf{x}_t + \mathbf{W}\mathbf{h}_{t-1}) \\ \hat{\mathbf{x}}_\mathbf{t} &= f(\mathbf{V}\mathbf{h_t}) \end{aligned} \tag{1}$$

Gated Network Units: RNNs are unable to effectively propagate long-term dependencies due to vanishing or exploding gradients [26]. Hence, variants of RNNs were proposed to improve upon long-term dependency retention. A detailed explanation of two popular variants namely the Long Short-Term Memory Unit (LSTM) and the Gated Recurrent Unit (GRU) have been undertaken in [27]. Both these gated variants propose additive state updates as opposed to the traditional RNN in which the content of the current recurrent unit is completely replaced as a function of the previous hidden state and current input [27]. The **LSTM**, initially proposed in [28], uses purpose-built memory cells (with *hidden states* as well as *cell states*) and gating mechanisms to forget irrelevant information while selecting and storing relevant information from the input sequence and previous hidden and cell states enabling LSTM networks to exploit long-range dependencies. We base our LSTM implementation on the variant proposed by Graves et al. in [29]

$$\begin{aligned} i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\ f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\ c_t &= f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\ h_t &= o_t tanh(c_t) \end{aligned} \tag{2}$$

In Eq. 2, i_t represents the input gate, f_t the forget gate, c_t the cell state and o_t the output gate at time step t. The vector h_t represents the hidden state vector at time step t. Each matrix W_* is a weight matrix learned within each gate, used to perform functions like forgetting irrelevant parts of the representations while adding new relevant parts to the representation thereby improving model performance. The $\sigma, \tanh$ represent logistic sigmoid and hyperbolic tangent transfer functions respectively. **Gated Recurrent Units** (GRU) are implemented as described by Cho et al. in [12]. The GRU also employs a gating mechanism for inclusion and exclusion of information from memory similar to an LSTM. However, unlike the LSTM, a GRU cell has fewer parameters as it consists of only two gates and also because it does not maintain any cell state.

$$\begin{aligned} z_t &= \sigma(W_{xz}x_t + W_{xh}h_{t-1} + b_z) \\ r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\ h'_t &= \sigma(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h) \\ h_t &= z_t \odot h_{t-1} + (1 - z_t) \odot h'_t \end{aligned} \tag{3}$$

Each GRU unit (Eq. 3) accepts a vector x_t as its input at time step t and the hidden state from the previous time step (i.e time step $t-1$), h_{t-1}. The input and previous hidden state are passed through the *update gate* z_t which decides how much information to retain for the future. The *reset gate* r_t informs the model about information from the past that is irrelevant and can be forgotten and also operates on the input x_t and previous hidden state h_{t-1}. Finally the input x_t, the *reset gate* r_t and the previous hidden state h_{t-1} are used to produce h'_t which is the hidden state for time step t with information from the past that the reset gate has deemed important. Finally, the hidden state at time t (i.e h_t) is calculated as a convex combination of the previous hidden state and h'_t. It must be noted that the GRU unit uses only one memory cell for storing the hidden representation and does not use any extra memory like the LSTM does. The differences between the GRU and LSTM cells are in how the actual gating mechanisms are implemented and the fact that the GRU maintains only a hidden state unlike in the case of an LSTM cell which has both a hidden state and a cell state thus increasing the number of model parameters and consequently the training complexity of LSTMs.
Both GRUs and LSTMs have mostly been trained on tasks in natural language processing and primarily in sequential modeling tasks like machine translation. Such attempts have generally found GRUs and LSTMs to have comparable performance across various tasks in the Natural Language Processing (NLP) domain [27]. We will now define equivalent sequence modeling tasks for Seq2Seq models in the context of time series.

V. Problem Formulation

Definition 1. ***(Multivariate Time-series)****: A multivariate time series* $\mathbf{X} = \{\mathbf{x_1}, ..., \mathbf{x_m}\}$ *is an ordered set of* m *vector valued quantities where each* $\mathbf{x}_i \in \mathbb{R}^{1\times k}$ *represents the scalar values of* k *time-series variables at the* i^{th} *time step.*

At some time-step t, we wish to forecast future states of the CPS for monitoring and planning purposes. We consider forecast lengths exceeding $t + 10$ time steps ahead to be long-term forecasts and all other forecasts to be short-term forecasts. Hence, in this paper, we address the problem of multi-step ahead (a.k.a sequence) forecasting of the system state of a CPS.

Problem 1. *Given a multivariate time-series* $\mathbf{X}$ *of system operational characteristics of a CPS, develop a model for short and long-term forecasting of system state.*

We treat the multi-step ahead forecasting task described in problem 1 as one of sequence generation. The problem of sequence generation has been popular in natural language processing applications like machine translation. There, the problem is, given a sentence (i.e. sequence of words) in a source language, to generate a sentence in the target language with the same meaning. Sequence to Sequence (Seq2Seq) models are a popular model architecture used for this purpose.
Time series sequence generation: A sequence
$\mathbf{X}^{(i)}_{t:t+w} = \{\mathbf{x_t}, \mathbf{x_{t+1}}, .., \mathbf{x_{t+w}}\}$ represents a multivariate time series of sequence length w, henceforth referred to as $\mathbf{X}^{(i)}$ for brevity. Given a set of j sequences $\{\mathbf{X}^{(1)}, \mathbf{X}^{(2)}, .., \mathbf{X}^{(j)}\}$ each of length w, of CPS operational characteristics, the forecasting model predicts the system state for all time-steps in sequence $j+1$ represented as $\mathbf{Y}^{(j+1)} \in \mathbb{R}^{w\times k}$, where $\mathbf{Y}^{(j+1)} = \{\mathbf{y}_{jw+1}, .., \mathbf{y}_{(j+1)w}\}$.

A. Short-Term Forecasting

Traditionally in time series forecasting applications, models are generally trained with a short-term prediction loss (mostly one step-ahead). That is, given data $\mathbf{X}^{(i)}_{t:t+w} = \{\mathbf{x_t}, \mathbf{x_{t+1}}, .., \mathbf{x_{t+w}}\}$, the model tries to predict $\mathbf{x_{t+w+1}}$. Recently it has been shown that deep recurrent architectures like LSTM/GRU show superior performance to traditional auto-regressive and moving average models. Deep Recurrent architectures like LSTM, GRUs are adept at modeling highly non-linear long sequences due to their ability to remember and propagate forward important parts of the representation. Thus we employ deep recurrent models for the purpose of long-term time series state forecasting and imputation in CPS.

The models trained with single step-ahead losses can be extended to do multiple step-ahead prediction by feeding the forecast at the current step as input to the next step. However, this approach also passes on the error at the current step to the next, thereby leading to the error increasing with the number of step-ahead forecasts the model makes. To overcome this problem and to train sequence forecasting models on multivariate time series data with highly non-linear structure (Fig. 2) it is important to train a model with multi-step ahead loss. Deep learning models are adept at modeling non-linear function spaces and one such deep architecture that allows us to train deep learning models on sequential data is the Sequence to Sequence architecture.

B. Long-Term Forecasting

1) Sequence to Sequence (Seq2Seq) Models: Seq2Seq models also termed *encoder-decoder* models are a neural network architecture introduced in [8] that learn to encode a variable

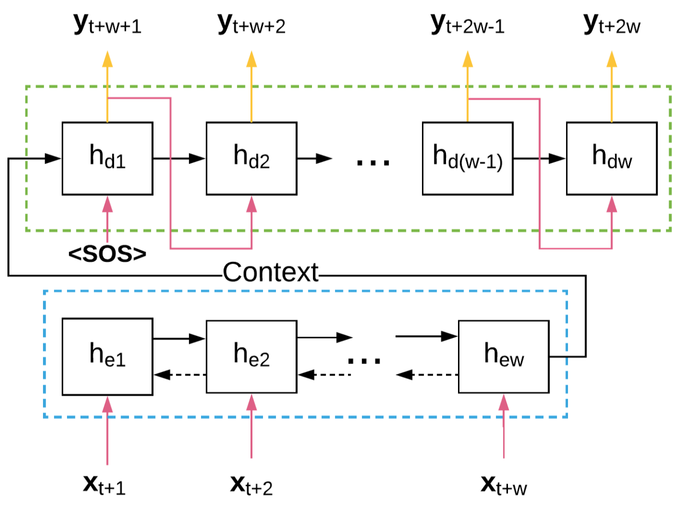


Fig. 3: **Sequence to Sequence (a.k.a Encoder-Decoder) model architecture used for forecasting and sequence reconstruction.**

length input sequence $X^{(i)}$ into a fixed length vector and use the learned sequence representation to predict another sequence. It is typically used in machine translation applications, wherein the encoder learns a fixed size representation of an input sentence in one language and uses it in the decoder to predict the sentence in a different language. We use this architecture to learn a representation of a window of values of the input series to predict the next window of values, i.e., do window size step-ahead forecasts. Given that sequence to sequence models are trained with multi-step loss, the problem of error accumulation with number of forecast steps is reduced. Fig. 3 illustrates the architecture of a seq2seq recurrent model. The encoder (dotted blue box) consists of w recurrent units where each unit accepts a vector $x_i \in \mathbb{R}^{1\times k}$ representing the values of the k time series at time step i. Each encoder unit h_{ei} then updates its state vectors based on the input and the previous hidden states passed into h_{ei}. The hidden state of the last encoder cell h_{ew} can be considered to represent the summary of the input sequence and is termed the *context vector* represented by $\mathbf{c} \in \mathbb{R}^{h\times 1}$ where h denotes the hidden size specified as a hyper-parameter. The context vector is then passed as input into the decoder (dotted green box). The first decoder cell h_{d1} receives as input, the context vector $\mathbf{c}$ along with a special *start of sequence* ($<SOS>$) character to indicate the start of a new decoder sequence. In our case, the start of sequence character is -1. Each subsequent decoder cell takes the output of the previous decoder cell as input along with the hidden state of the previous decoder cell. If sequence $\mathbf{X}^{(i)}$ is passed into the encoder, then the decoder outputs predictions for sequence $\mathbf{X}^{(i+1)}$. Each decoder cell outputs $\mathbf{y_j} \in \mathbb{R}^{1\times k}$ (refer Fig. 3) representing predictions for the j^{th} vector $\mathbf{x}_j$ in the sequence $\mathbf{X}^{(i+1)}$.

$$\arg\min_{\theta} \frac{1}{m}\sum_{i=1}^{m}(g(\mathbf{Y}^{(i+1)}|\mathbf{X}^{(i)}) - \mathbf{X}^{(i+1)})^2 \tag{4}$$

Eq. 4 represents a mean squared error loss function which is used to train the Seq2Seq models. It must be noted that only the output predictions of the decoder are compared with those of the true sequence. Since the decoder does not see any ground-truth sequence data and relies only on the context vector $\mathbf{c}$ it receives from the encoder, the decoder essentially acts as a sequence generator capable of generating sequences representative of future operational characteristics of the CPS. Each decoder unit accepts as input the predictions of the previous unit. This causes slow convergence and instability during training because, at each step of the decoder we make use of the predictions from prior units. However, during training, we have the ground-truth values and thus it is possible to make use of these ground-truth values to speedup convergence and introduce more stability to the training procedure. This procedure of using ground-truth values as input to each decoder unit rather than having connections from the output of preivous units to the input of the next unit is called *teacher-forcing*. Though using teacher-forcing speeds up convergence, it can lead to over-fitting as the model may learn to become overly reliant on ground truth inputs (which is not available at testing). This phenomenon called *exposure bias* has been elucidated in [30], [31]. Thus it is essential to strike a middle ground between teacher forcing and the normal training procedure. One such method is Scheduled Sampling(SS). It is a procedure used during training of encoder-decoder (i.e. Seq2Seq) models in which the decoder is first fed inputs from the ground truth data distribution at the beginning of model training and through the course of training, the percentage of sampling from the ground truth data distribution is gradually reduced in favor of the decoder using its own previous predictions as input to subsequent decoding steps. Finally, the fully trained encoder-decoder model uses only its previous predictions and does not rely on the ground truth data as input to the decoder.

C. Sequence to Sequence models with Attention

In Seq2Seq architectures, the encoder has to learn to represent the whole input sequence using a single fixed size vector (the context vector $\mathbf{c} \in \mathbb{R}^{h\times 1}$). In the case of very long sequences (common in time-series applications), this becomes a limitation as the representative capacity of the model is curtailed. Attention [32] is a mechanism used to overcome this shortcoming of encoder-decoder models. Attention, at each decoder step, learns to pay different levels of focus to each encoder unit, thereby allowing each decoder unit to learn suitable combinations of the encoder hidden states . This keeps the encoder decoder model from having to limit itself to one single representation (or view) of the encoder sequence for all the decoder steps thereby increasing the representative capacity of the model. The model architecture for Seq2Seq models with attention is shown in Fig 4.

The formulation of the Seq2Seq decoder with an attention mechanism is given in Eq. 5 (encoder formulation is the same

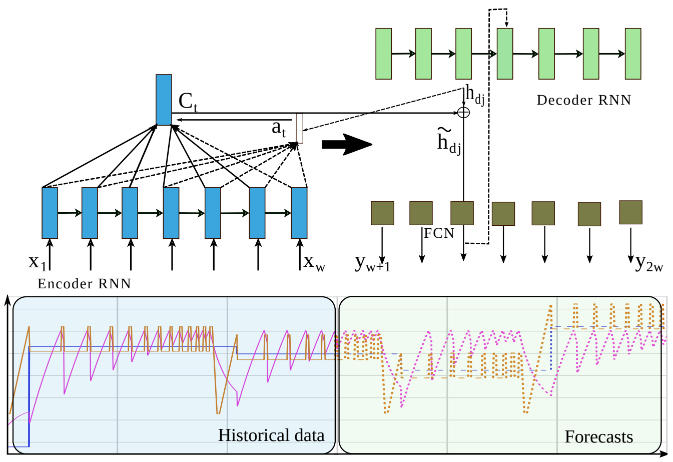


Fig. 4: **Sequence to Sequence models with attention architecture. At each step of the decoder (green boxes), the hidden state produced from the decoder RNN unit (light green units) is combined with a weighted average of the encoder hidden states. The weights which govern how the encoder hidden states are combined to yield the new attentional hidden state are calculated through an attention mechanism.**

as before).

$$\begin{aligned} h_{dj} &= f(y_j, h_{d(j-1)}) \\ \mathbf{a}_j &= align(h_{dj}, H_e) \\ c_j &= \mathbf{a}_j H_e = \sum_k \alpha_k h_{ek} \\ \tilde{h}_{dj} &= tanh(W_c[c_j; h_{dj}]) \end{aligned} \tag{5}$$

In Eq. 5, $f(\cdot)$ represents a recurrent unit like RNN, LSTM or GRU. $h_{dj} \in \mathbb{R}^{h\times 1}$ refers to the decoder hidden state at step j, $y_j \in \mathbb{R}^{1\times k}$ is the prediction output from the previous decoder state. $\mathbf{a}_j \in \mathcal{R}^{w\times 1}$ are the weights (alignment scores) learned for the encoder states by the attention mechanism. For calculating the alignment scores we use a softmax over a fully connected unit that takes as input the vector h_{dj} and $H_e \in \mathbb{R}^{w\times h}$, the set of all encoder hidden states as in Eq. 6.

$$align(h_{dj}, H_e) = \frac{exp(H_e, h_{dj})}{\sum exp(H_e, h_{dj})} \tag{6}$$

In Eq. 6, the denominator is a sum of the exponentiation of h_{dj} multiplied with each encoder hidden state in H_e and is used to perform a Hadamard division of the numerator yielding $\mathbf{a_j} \in \mathbb{R}^{w\times 1}$, the attention vector for the j^{th} decoder unit.

So far, we have discussed architectures that can be used for state forecasting. All such architectures assume continuous data is available for training. However, in real-life applications, the training data obtained from sensors can have missing periods. Thus it is essential to come up with techniques to re-construct these missing sequences (or periods) so that the aforementioned forecasting models can be trained.

Problem 2. *Augment a CPS state forecasting model to address the problem of missing data reconstruction, ubiquitous in sensor data based CPS systems.*

SCADA systems use commodity sensor hardware to collect data and quite often face problems of data sparsity or missing data due to sensor failure. We augment our forecasting model to address the problem of re-construction of such missing data sequences. Each encoder cell h_{ei} in Fig 3, has a backward dotted arrow feeding into it from cell h_{ei+1} in addition to a forward solid arrow from h_{ei} to h_{ei+1}. This architecture termed a *Bi-directional Seq2Seq* (**Bi-Seq2Seq**) model allows the inputs to be passed into the encoder in the forward and the reverse direction thus enabling stronger temporal dependence discovery leading to better representation learning [33], [34]. The number of hidden states is doubled in the case of bi-directional models as the backward path has a separate set of hidden states than the forward path in the encoder. The context vector in this case is constructed as the concatenation of the forward path hidden state $h_{ew} \in \mathbb{R}^{k\times 1}$ and the backward path hidden state $h_{e1}\mathbb{R}^{k\times 1}$. Let us consider the case of a missing sequence $X^{(i)}$ to be re-constructed by the bidirectional seq2seq model. Then, the sequences $X^{(i-1)}, X^{(i+1)}$ are concatenated and passed into the encoder and the decoder returns its predicted reconstruction of the missing sequence $X^{(i)}$.

VI. Experimental Setup

Network Architecture: All experiments are conducted using recurrent network models with a single hidden layer. Each model has been trained for 100 epochs with a learning rate of 0.001. We test gated recurrent units - LSTM, GRU - as well as the non-gated RNN recurrent unit in our experiments. Each of the aforementioned recurrent units are employed with step-ahead forecasting, unidirectional sequence to sequence (**Seq2Seq**) and bidirectional Seq2Seq (**Bi-Seq2Seq**) architectures. We employ *tanh* activation functions followed by a linear layer to produce the predicted output at each time-step. Finally, we also incorporate linear scheduled sampling [31] into the training procedure of Seq2Seq models to yield better trained networks. Unless otherwise stated, for fair performance comparison, we have used dropout $p = 0.0$, hidden size $h = 96$ for all our experiments.

Data Preprocessing: The scale of each time series varies widely for the GHL system and hence we adopt the popular mean normalization scheme to scale the data. For each time-series $\mathbf{x} \in \mathbb{R}^{m\times 1}$ in the GHL dataset, Eq. 7 describes the normalization scheme we employ to scale $\mathbf{x}$ in the range (1,-1).

$$\mathbf{x}^* = \frac{\mathbf{x} - \bar{x}}{max(\mathbf{x}) - min(\mathbf{x})} \tag{7}$$

Data Imputation: For data imputation experiments, we generate data by randomly sampling a set of time points $T_w = \{t_i, t_{i+1}, .., t_{i+w}\}$ and nullifying data from t_i to t_{i+w}. Here, w is the sequence length of the missing sequence T_w and it can be 1500 or 2000 for this experiment. We compare two types of bidirectional Seq2Seq models – 1) a forecasting model which takes as input t_{i-w} to t_i and predicts the missing sequence and 2) a model which looks at sequences appearing both before and after the missing sequence, i.e, t_{i-w} to t_i and t_{i+w} to t_{i+2*w} and predicts the missing sequence.

VII. Experimental Results

We try to answer the following questions through our experiments

1) How do Seq2Seq architectures compare to step-ahead forecasting architectures in short and long-term time series forecasting applications?
2) Do bi-directional models yield superior forecasting performance to their unidirectional counterparts?
3) Do procedures like scheduled sampling help in training more effective models?
4) How does the representation learning capability of recurrent units vary across different models?
5) Can the aforementioned forecasting architectures be effectively utilized in contexts with missing data for data sequence reconstruction?
6) Do attention mechanisms help improve Seq2Seq performance?

*1) **Short-Term Forecasting Performance**:* Table I depicts the comparative performance of Seq2Seq models and step-ahead forecasting recurrent models on a short-term 10 step ahead forecasting task. We observe that the Seq2Seq models perform significantly better than their step-ahead forecasting counterparts. Models with the RNN recurrent unit yield forecasts that are significantly inferior than models that use the LSTM, GRU units in all cases. The Seq2Seq-GRU model (S2S GRU) exhibits a superior performance to all other models employed in the short-term forecasting task.

Model	w	MSE
RNN	10	0.031
LSTM	10	0.0054
GRU	10	0.0042
S2S RNN	10	0.0037
S2S LSTM	10	0.0032
S2S GRU	10	**0.003**

TABLE I: **Performance comparison between Seq2Seq and traditional step-ahead forecasting models for short-term forecasting (sequence Length (w)=10 and hidden size (h) = 32). We can see that even for short sequence lengths, Seq2Seq models outperform basic recurrent models.**

*2) **Long-Term Forecasting Performance**:* Fig. 5 depicts the comparative forecasting performance of two types of model architectures the Seq2Seq architecture (S2S) and the Bidirectional Seq2Seq architecture (Bi-S2S) each with RNN, LSTM, GRU units. In each case, we observe that the RNN model is the worst performing unit. Both the unidirectional and bidirectional GRU models outperform their LSTM counterparts. In the case of LSTMs and GRUs, the bidirectional LSTM and GRU outperform their unidirectional counterparts respectively. We believe this is because of the stronger temporal relationships learned by passing the same sequence in the positive and negative temporal direction [33] thereby tapping into the full representative power of the LSTM and GRU cells. The same effect is not observed in the case of RNNs as the basic RNN cell is unable to effectively propagate temporal dependencies over long sequences. Finally, we see that the Bi-directional Seq2Seq GRU model yields the best performance across all window lengths, significantly outperforming all other models in terms of forecasting performance. Also, it can be observed that the quality of the forecast degrades with increasing sequence length.

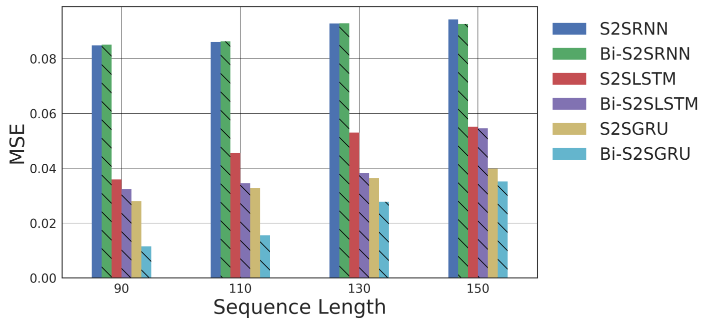


Fig. 5: **Sequence Length vs. MSE (lower is better). Bidirectional models in general seem to show better performance (lower MSE) as compared to their uni-directional counterparts, with the GRU based variant performing the best among all models. The usage of bi-directional layers seem to help the gated variants much more than the non-gated simple RNN based models. The Bi-S2SGRU model provides approx. 60% reduction in MSE over S2SRNN.**

*3) **Effect of Scheduled Sampling on model training**:* Fig. 6 shows the percentage improvement in performance of bi-directional Seq2Seq model (*Bi-S2SGRU(SS)*) and Seq2Seq model (*S2SGRU(SS)*), both using GRU as its recurrent unit and trained with scheduled sampling, over the same models trained without scheduled sampling i.e *Bi-S2SGRU*, *S2SGRU*. We can see that there is a significant performance improvement obtained with scheduled sampling in the case of bidirectional Seq2Seq models. The improvement is negligible for the unidirectional Seq2Seq models (S2SGRU) in the case of sequence length 90. However, the performance improvement increases with increasing sequence length (w) yielding a maximum performance improvement of **8.71**% at w=150 and a mean performance improvement across all sequence lengths of **4.87**%. Scheduled sampling offers significantly higher performance improvement in the case of bidirectional Seq2Seq models with a maximum percentage improvement of **43.73**% for w=150 and a mean performance improvement of **25.89**% across all sequence lengths.

*4) **Seq2Seq Model Representation Learning Characterization**:* We delve deeper into what makes Seq2Seq models (with and without scheduled sampling) perform better than simple recurrent models. For this, we show a heatmap plot

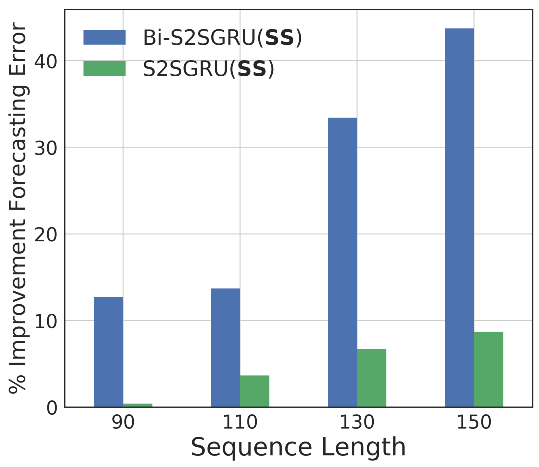


Fig. 6: **Impact of Scheduled sampling based training on forecasting error (MSE) - (higher is better). The bidirectional models trained with scheduled sampling show a max performance improvement of about 44% (for sequence length 150) whereas for uni-directional models, training with scheduled sampling only provides about a 9% increase at the most. Also we can see that effect of using scheduled sampling becomes more pronounced as the sequence length increases.**

of the learned hidden to hidden transformation weights of a GRU cell trained in a Seq2Seq architecture (with and without scheduled sampling) as well as a simple recurrent architecture. The heatmap is shown in Fig. 8. We can see that bidirectional Seq2Seq models learn sparser weights than the simple recurrent models. Also the hidden to hidden transformation weights learned in the update gates of a GRU cell in the Seq2Seq models show more definitive striations or patterns with the scheduled sampling variant being more sparse. There are no discernible patterns in the learned weights of a simple recurrent GRU model. Thus we can say that Seq2Seq models learn higher quality data representations relative to their non Seq2Seq counterparts.

*5) **Data Imputation:**:* For data imputation, we randomly remove 100 sequences of length w and then try to reconstruct them using a model that uses – 1) only the past information and 2) both past and future information. We refer to the second variant as Bi-S2S(**IMP**). To perform the data imputation experiments, we naturally choose the best models from previous experiments i.e Bidirectional Seq2Seq models trained with Scheduled Sampling (Bi-S2S(**SS**)) and employ the RNN, LSTM, GRU recurrent units with this architecture. The results for data imputation are presented in Table II. Using future information leads to only slightly improved performance in RNN and GRU architectures. Thus, we conclude that Bi-Seq2Seq models used for forecasting are effective in sequence reconstruction also and we do not need any separate model for imputation.

*6) **Importance of Attention in Seq2Seq Models:**:* Table III shows the percentage improvement in terms of MSE for the

Cell	w	Bi-S2S(IMP)	Bi-S2S(SS)
RNN	1500	0.0902	0.0922
	2000	0.091361	0.091812
LSTM	1500	0.0933	0.0915
	2000	0.0914	0.0901
GRU	1500	**0.0898**	0.0904
	2000	**0.091**	0.0915

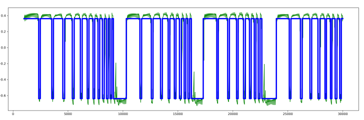
TABLE II: **Imputation performance (in terms of MSE) comparison of bidirectional models that use only past information (Bi-Seq2Seq(SS)) with models that use both past and future information (Bi-Seq2Seq(IMP)). We see that GRU based models perform better than LSTM and RNN in the imputation task. Also, the Bi-Seq2Seq(SS) model that uses only past information yields inferior but comparable performance to Bi-Seq2Seq(IMP).**

Seq2Seq model with attention over the normal Seq2Seq model (i.e without attention). The table also provides the effect of attention for Bi-Seq2Seq models. Specifically,

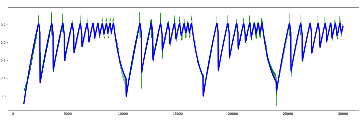
- We can see that using attention significantly helps improve performance of unidirectional Seq2Seq models for cases of longer sequence lengths as expected. This is because, for short sequences, it is possible for the encoder to store all necessary information in a single fixed size context vector and thus attention is not very helpful.
- We also notice that attention does not help in case of Bi-Seq2Seq models. We believe that using both backward and forward recurrent layers yields a final context vector that has a good representation of both the beginning and the end of the input sequence. Thus, an attention mechanism to encode long-term dependencies into the context vector does not have much effect. On the other hand, in unidirectional Seq2Seq models, the final hidden state of the encoder has a better representation of the latter part of the input sequence than its beginning (using gated architectures like LSTM and GRU only reduces this problem but it does not eliminate it). Using attention models in this context, allows each decoder unit to use all hidden states of the encoder, allowing for better representation learning of the entire input sequence encoded into the context vector.

Our claim of higher quality model representation learning using the attention mechanism is further corroborated by Fig. 9 which depicts a comparison between the mean squared error per decoder unit (or time step) of a Seq2Seq model without attention and a Seq2Seq model with attention. Both the models use GRU as the recurrent unit and the sequence length of the predicted sequence (i.e decoder sequence) is 90 time steps.

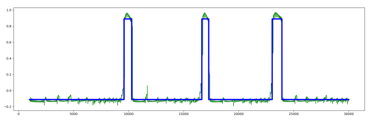
- We observe from the figure that the additive error in the Seq2Seq model without attention, causes the MSE to increase at a significantly higher rate compared to the rate of MSE increase of the Seq2Seq model with attention. This is primarily because of the flexibility offered by the attention mechanism allowing the decoder to focus



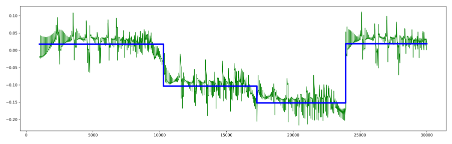
(a) Heater Act.



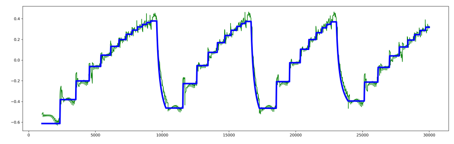
(b) Heating Tank Temp.



(c) Injection Valve Act.



(d) Receiver Tank Level.



(e) Receiver Tank Temp.

Fig. 7: **Seq2Seq model predictions for each time-series in the Gasoil heating loop dataset with a prediction sequence length (w) of 90. We depict the predictions (in green) and the ground-truth (in blue) for a sample time period in the testing phase of the bidirectional sequence to sequence model trained with scheduled sampling (Bi-S2S(SS)). It must be noted that the predictions are not clearly visible in Fig. (a)-(c) and Fig. (e) as they are superimposed on the actual values indicating that the Bi-S2S(SS) forecasting model is able to yield very good long-term forecasts.**

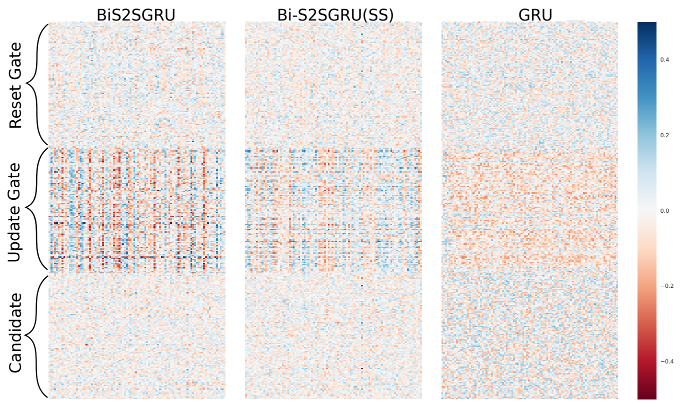


Fig. 8: **Heatmap of GRU weight matrices (reset gate, update gate and candidate). We see that Bi-Seq2Seq models learn sparser weights than simple recurrent models. Also the update gate weight matrix in Bi-Seq2Seq models shows definite patterns, with the scheduled sampling variant (Bi-S2S(SS)) being sparser.**

w	Seq2Seq with Attention	Bi-Seq2Seq(SS) with Attention
10	1.850	-58.000
90	71.162	-67.807
110	72.403	-189.179
130	75.058	-7.226
150	65.175	-25.867

TABLE III: **Percentage improvement of attention based models over their corresponding non-attention counterparts. All models had a single hidden layer and were tested with GRU as the recurrent unit.**

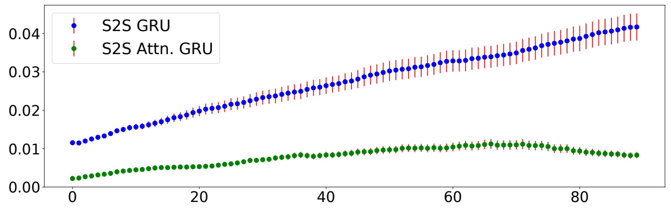


Fig. 9: **Encoder Decoder Attention Model Comparison of mean squared error per decoder unit (w = 90). We observe that the mean squared error increases significantly with each decoder unit in long-term sequence forecasting problems for Seq2Seq models without attention. In the case of Seq2Seq models with attention, the flexibility brought in by the attention mechanism helps control the effect of these additive errors in long decoder sequences.**

on different parts of the encoder sequence and select the subset of encoder hidden states most amenable to model the next sequence output.

- The red error bars indicate the variance in MSE per decoder unit and we notice that the variance increases significantly more in the case of the Seq2Seq model without attention than for the model with attention indicating that the Seq2Seq model without attention becomes more uncertain about its predictions further into the decoder sequence as it is making its predictions with less information about the encoder sequence than its counterpart Seq2Seq model with attention.

VIII. Conclusion

In this paper, we have developed a bidirectional Seq2Seq model **Bi-Seq2Seq** capable of providing long-term forecasts of the operational state of a CPS as well as reconstructing missing data sequences. We have evaluated our models in comparison with simple unidirectional Seq2Seq architectures as well as with regular step ahead forecasting variants of the recurrent model cells and showcased quantitative results. All our results indicate that the **Bi-Seq2Seq** models significantly outperform other models in the forecasting task and hence we are able to conclude that they learn a far better representation of the underlying CPS process relative to the non bidirectional, non Seq2Seq variants. We show qualitatively that the learned representations are sparser for Seq2Seq models than their simpler step-ahead counterparts. We have also shown that incorporating mechanisms like attention does not improve the performance of our **Bi-Seq2Seq** model trained with scheduled sampling whereas attention mechanisms do have a positive effect on the performance of the unidirectional **Seq2Seq** models. Moving forward, we wish to utilize the forecasting models in applications like anomaly (or intrusion) detection in CPS.

References

[1] P. Filonov, A. Lavrentyev, and A. Vorontsov, "Multivariate industrial time series with cyber-attack simulation: Fault detection using an lstm-based predictive data model," *arXiv preprint arXiv:1612.06676*, 2016.

[2] K.-D. Kim and P. Kumar, "An overview and some challenges in cyber-physical systems," *Journal of the Indian Institute of Science*, 2013.

[3] M. Momtazpour, J. Zhang, S. Rahman *et al.*, "Analyzing invariants in cyber-physical systems using latent factor regression," in *ACM SIGKDD*, 2015.

[4] G. S. Babu, P. Zhao, and X.-L. Li, "Deep convolutional neural network based regression approach for estimation of remaining useful life," in *DASFAA*, 2016.

[5] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *ACM/IEEE DAC*, 2010.

[6] L. Zhao, Z. Chen, Z. Yang, Y. Hu, and M. S. Obaidat, "Local similarity imputation based on fast clustering for incomplete data in cyber-physical systems," *IEEE Systems Journal*, 2016.

[7] J. G. De Gooijer and R. J. Hyndman, "25 years of time series forecasting," *International journal of forecasting*, 2006.

[8] K. Cho, B. Van Merriënboer, C. Gulcehre *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *NeurIPS*, 2014, pp. 3104–3112.

[10] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017, pp. 5998–6008.

[12] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[13] J. Lin, X. Sun, S. Ma, and Q. Su, "Global encoding for abstractive summarization," *arXiv preprint arXiv:1805.03989*, 2018.

[14] R. Nallapati, B. Zhou, C. Gulcehre, B. Xiang *et al.*, "Abstractive text summarization using sequence-to-sequence rnns and beyond," *arXiv preprint arXiv:1602.06023*, 2016.

[15] J. Gehring, M. Auli, Grangier *et al.*, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[16] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. v. d. Oord, A. Graves, and K. Kavukcuoglu, "Neural machine translation in linear time," *arXiv preprint arXiv:1610.10099*, 2016.

[17] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 93–98.

[18] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *arXiv preprint arXiv:1507.06947*, 2015.

[19] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskever, D. Sussillo, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," in *Advances in Neural Information Processing Systems*, 2016, pp. 5067–5075.

[20] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE ICASSP*. IEEE, 2013, pp. 6645–6649.

[21] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *NeurIPS*, 2015, pp. 577–585.

[22] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *IEEEE ICASSP, 2017*. IEEE, 2017, pp. 4845–4849.

[23] L. Lu, X. Zhang, and S. Renais, "On training the recurrent neural network encoder-decoder for large vocabulary end-to-end speech recognition," in *ICASSP*. IEEE, 2016, pp. 5060–5064.

[24] P. Filonov, F. Kitashov, and A. Lavrentyev, "Rnn-based early cyber-attack detection for the tennessee eastman process," *arXiv preprint arXiv:1709.02232*, 2017.

[25] P. Malhotra, A. Ramakrishnan, G. Anand *et al.*, "Lstm-based encoder-decoder for multi-sensor anomaly detection," *arXiv preprint arXiv:1607.00148*, 2016.

[26] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE transactions on neural networks*, 1994.

[27] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, 1997.

[29] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[30] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[31] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *NeurIPS*, 2015.

[32] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[33] D. Cheng, M. T. Bahadori, and Y. Liu, "Fblg: a simple and effective approach for temporal dependence discovery from time series data," in *ACM SIGKDD*, 2014.

[34] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, 1997.