Prediction of Oxygen Content in Boiler Flue Gas via Neural Networks

Mohammed Aldakhil, et. al.

*Abstract*— Oxygen content in flue gas is an important parameter for monitoring the efficiency of combustion processes and NOx emissions. The aim of this study is to develop an Artificial Intelligence (AI)-based model to predict oxygen content in the flue gas of a 200-MW gas-fired industrial water tube boiler using three machine learning algorithms: Temporal Convolutional Network (TCN), Long Short-Term Memory (LSTM), and Feed-forward Neural Network (FNN). The model was trained on a dataset of flue gas measurements from a single boiler. To improve performance, feature engineering was conducted, including removing weakly correlated features and deriving new features based on domain knowledge. Feature scaling was also applied to enhance model stability. The results showed that the LSTM model achieved the best prediction accuracy with an RMSE score of 0.0056, outperforming the TCN and FNN models. While the TCN model demonstrated competitive performance with slightly higher error margins, the FNN model struggled due to its inability to capture long-term dependencies in time-series data. The developed model can be used to optimize combustion efficiency and NOx emissions by continuously monitoring the oxygen content and adjusting the combustion process accordingly.

Keywords: oxygen content, flue gas, machine learning, combustion efficiency, Temporal Conventional Network, Long-Short Time Memory, Feedforward Neural Network, feature engineering

# Introduction

Neural networks (NN) have been a game-changer in the field of machine learning, mimicking human intelligence to revolutionize how we approach technology and its applications [1]. Series forecasting is the process of predicting future values of a variable based on historical data. One area where NN have shown limitations is using statistical time series forecasting methods to predict non-linear complex data. Statistical time series forecasting methods have limitations that stem from several factors, such as their lack of generalization abilities and their inability to analyze complex relationships in data [2]. These limitations often result in inaccurate predictions, particularly when dealing with non-linear data [3]. Newer generation models for time series forecasting, such as those based on artificial neural networks, offer significant advantages in this regard [4]. These models can learn from complex relationships and analyze past patterns in data to make accurate predictions [5]. Time series forecasting is crucial in many industries, from finance [6] to healthcare [7], where accurate predictions of future trends and patterns are required to make informed decisions [8]. There are several techniques for time series forecasting, including traditional statistical methods such as Auto Regression (AR), Moving Average (MA), and Autoregressive Integrated Moving Average (ARIMA), as well as more advanced and complex methods such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, Gated Recurrent Units (GRUs), encoder-decoder models, Convolutional Neural Network (CNN), and Temporal Convolutional Networks (TCNs). AR models use past observations of a variable to predict future values. They also use the current value of a variable to model it as a linear combination of past values, with the weights determined by the model parameters [9]. MA models, on the other hand, use past forecast errors to predict future values [10]. Additionally, in an MA model, the current value of a variable is modeled as a linear combination of past forecast errors, with the weights determined by the model parameters [10]. Both AR and MA models can be combined into an ARIMA model, which uses both past values and past errors to make predictions [3]. RNNs are a type of neural network that can process sequences of inputs, making them well-suited for time series forecasting but they suffer significantly from the gradient explosion/vanishing problem [11]. Thus, LSTM was developed to solve this issue, LSTM networks are a type of RNN that can learn long-term dependencies and can identify which information to keep and which to forget [11]. GRUs are similar to LSTMs, but have fewer parameters, making them faster to train [12]. Encoder-decoder models are a type of deep learning architecture that can be used for sequence-to-sequence learning. This model consists of two components: an encoder that processes the input sequence and a decoder that generates the output sequence [12]. TCNs are a type of convolutional neural network CNN that can take an arbitrary sequence-to-sequence. In addition, TCNs are considered to be the state-of-the-art for time series forecasting as recent papers suggest that TCNs are robust for time series prediction and more efficient and easier to deploy [13]. Research studies related to boilers’ state and forecasting are always focused on using RNN models such as LSTM, GRU, Encoder-Decoder, and Vanilla RNN to forecast the performance of boilers [14]. Additionally, this approach is also used with coal-fired boilers and it showed high performance in optimizing and controlling the boiler[15]–[17]. However, very few studies have focused on benchmarking a TCN model [18]. In this study, we compare the performance of FNN, LSTM, and TCN in predicting the Oxygen content in the flue gas of a 200-MW gas-firedindustrial water tube boiler using a real dataset. FNNs are well-suited for simple forecasting tasks, while LSTMs can learn long-term dependencies and work well with sequential data. TCNs are a well-established model designed for forecasting tasks, and they can also learn long-range dependencies in data. Therefore, these are the selected models as they all work well with time series data. The study investigates the impact of various factors such as data preprocessing, feature selection, and model selection on the accuracy of forecasting the oxygen content if the flue gas of an industrial boiler. The results of this research can provide insights into the effectiveness of the methods mentioned earlier and help operators make informed decisions to enhance the performance of the boiler.

# Models overview

This section discusses the structures of FNN, LSTM, and TCN models, along with the functionality of each model.

## Feed-forward neural network (FNN)

Feedforward neural networks (FNNs) are commonly used as the basic method for machine learning (ML) problems. They are composed of multiple layers of neurons with an activation function that adds up the inputs received and maps them to a certain range to facilitate the learning process of other neurons inside the network. Fig. 1 depicts the FNN model general structure, while Fig. 2 includes the internal components of each neuron and the calculations performed inside it. Generally, neural network models follow the concept of passes, which involves a forward propagation pass used to calculate the desired output, and a backward propagation pass which adjusts the weights in the neuron to improve the correctness of the predictions made in each iteration. The forward propagation pass and the ReLU Activation function are mathematically described by the following equations:

|  |  |
| --- | --- |
|  | (1) |
|  | (2) |

where *r* denotes the layer number of the model. *k* denotes the current layer. b represents the bias of the specified neuron. w denotes the weight and h is the input of the current neuron. When r = 0, this represents the input layer as . z is the output of each neuron in the hidden layer. The neural network uses equation (1) to start the learning of the model and then feeds the result to equation (2) which represents the activation function used in the hidden layers only. Backpropagation uses the concept of gradient descent (GD) as an optimization function for adjusting the weights. GD is a function that captures how much the model should adjust the weights by differentiating equations (1) and (2) regarding the weights. This is performed in the hope of finding the global minimum or at least a local minimum. Other optimization functions can be used, such as stochastic gradient descent (SGD).

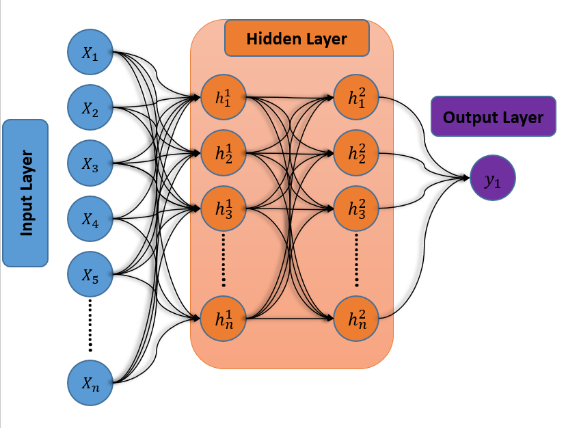


Fig. 1. General structure of a FNN model

A picture containing circle, diagram, line, design

Description automatically generated

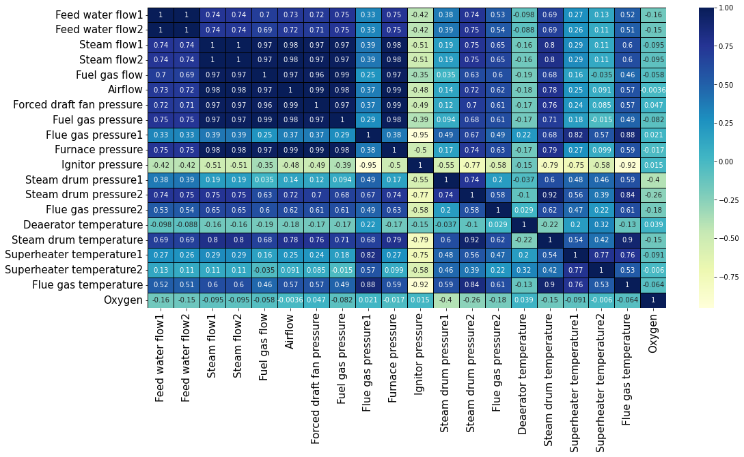
Fig. 2. The internal components of a neuron in FNN model

## Long-Short Time memory model (LSTM)

LSTMs are a special type of recurrent neural network (RNNs) designed for sequential data and handling Long-short dependencies on sequential data. LSTMs have several advantages over other types of RNNs. They are able to learn long-term dependencies in sequential data, which makes them well-suited for time series forecasting[19].In addition, LSTMs are designed to overcome the vanishing gradient problem. They are also relatively easy to train, which makes LSTM a popular choice for many machine-learning applications. LSTMs use a memory cell to store information about previous inputs. This allows them to learn long-term dependencies in the data. LSTMs have three gates that control the flow of information into and out of the memory cell: the forget gate, the input gate, and the output gate. Fig. 3 shows the internal components of a single LSTM unit. The calculations done within a single unit are as follows[19]:

|  |  |
| --- | --- |
|  | (3) |
|  | (4) |
|  | (5) |
|  | (6) |
|  | (7) |
|  | (8) |

Where and are input gate, forget gate, and output gate respectively. is the cell state and as the filter for the cell state. The parameters for the LSTM unit are the W’s and B’s corresponding to each gate. Allowing the units to train on the sequence {} will provide us with this sequence of hidden states {} as outputs. The mean value of the sequence is then used in the future input sequences.

A picture containing diagram, line, parallel, screenshot

Description automatically generated

Fig. 3. The structure of an LSTM unit

## Temporal conventional neural network model (TCN)

Temporal convolutional networks (TCNs) are a type of convolutional neural network (CNN) specifically designed to handle sequential data. TCNs can learn long-term dependencies in sequential data by using dilated convolutions. Dilated convolutions are a type of convolution that allows the network to learn dependencies over a wider range of inputs, as shown in Fig. 4 This makes TCNs well-suited for tasks such as time series forecasting, where the network needs to learn dependencies over long periods.

Fig. 5. Correlation heatmap of the raw data

TCNs have several advantages over other types of neural networks for time series forecasting. First, TCNs can learn long-term dependencies in sequential data. This is a key advantage for time series forecasting, where the network needs to learn dependencies over long periods of time[20]. Second, TCNs are relatively easy to train. This is because TCNs do not require backpropagation through time, which can be difficult to train. Third, TCNs can handle variable-length inputs. This is a key advantage for tasks such as time series forecasting, where the input data can vary in length.

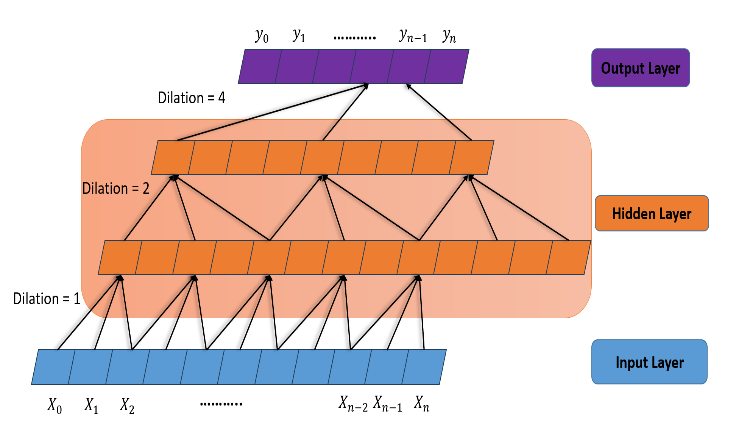
In summary, FNN are simple and efficient for processing static data, while LSTM networks excel in capturing long-term dependencies in sequential data. Further, TCNs are suitable for tasks that require capturing temporal patterns in sequential data. The choice of model depends on the specific requirements of the task at hand, such as the need for memory, the presence of long-term dependencies, and the nature of the input data.

Fig. 4. Structure of a TCN model with 1,2, and 4 dilations

# Problem Statement

The goal of the current study is to develop an AI-based model that can correctly capture O2 contents in flue gas and make valid and reliable predictions. The developed model must also have a reasonable training time and an efficient architecture size. To achieve these goals, a relatively big dataset from a 200-MW gas-fired industrial water tube boiler was utilized. The boiler is used for production of superheated steam for an industrial process at Saudi Aramco company [21]. Three models were used to predict O2 in the flue gas, namely FNN, LSTM, and TCN. The performance of the models was also compared by benchmarking four different error metrics, which are Root Mean Square Error (RMSE), Maximum Absolute Percentage Error (MaxAPE), Mean Absolute Percentage Error (MAPE), and Maximum Root Square Error (MaxRSE). These errors are calculated using Equations 9 – 12.

|  |  |
| --- | --- |
|  | (9) |
|  | (10) |
|  | (11) |
|  | (12) |

where is the actual value, i as the predicted value, and n is the number of test list. To add, the max function is just taking the highest difference between all test points, just to give an insight about how poorly the model could be doing.

# Data preparation

The original dataset, collected over a period of 6 hours and a timestep of 1 second, consists of 21000 datapoints and 20 features, with oxygen (O2) as the desired output feature, and the remaining 19 as input features. Table 1 presents the complete features list and explains their physics. To explore the linear dependency of the input labels on the output feature, the dataset was visualized using the correlation heatmap shown in Fig. 5. It can be concluded from Fig. 5 that most of the input features are linearly correlated, which can result in poor model performance. Thus, in an attempt to reduce such linear dependency, physics-based domain knowledge has been implemented. The data transformation included dropping similar features (i.e. measurements taken very close to each other). Additionally, some of the features are physically correlated and, therefore, combined into a single feature by taking their ratios. These features include {} obtaining new features as A/F, FW/SF, and PFuel/Pfurnace, respectively. The ideal gas equation (p v = n R T) was used to reduce the remaining correlated features, as it seemed convenient to do so since the equation's variables are present, assuming n=1 (i.e., number of moles). The resultant feature has units of the molar basis specific volume (m3/kmol). Hence, the new features (vexh, and vsd) are calculated by Equations 13 and 14.

|  |  |
| --- | --- |
|  | (13) |
|  | (14) |

Where R is the universal gas constant. The remaining unused features (features 2,4,7,9,11,13,15,17-18 in Table 1), were dropped since they did not provide much information about the boiler’s state as can be seen in Fig. 5, nor did they increase the models’ performance when trained. Table 2 summarizes the final list of input features and their physics-based reduction methodology. Moreover, our training inputs are the 5 features, and our output is the Oxygen column. The

correlation between the input variables is no longer present, as shown in Fig. 6. This implies that the input variables are no longer related to each other and can be used independently to train the models. Fig. 7 shows histogram plots of the modified data and how they are distributed. We can see that seems to be constant, so we expect it to not give much information to the model. We will test this in the analysis and results section. Other features seem to follow normal distribution, as can be seen in Fig. 8, with most of them having distributed points, except for vexh which again seems to be constant. Furthermore, vsd and PFuel/Pfurnace appear to be discreet and not chaotic, as they have a path for their values. However, both have an acceptable distribution,

and we expect them to make a decent contribution to the models' training phase.



Fig. 6. Heatmap correlation values of the updated features

TABLE I The original features before any modification

|  |  |  |
| --- | --- | --- |
| **Parameter name** | **Description** | **Unit** |
| Feed water flow1 | Pounds of water fed to boiling tank per hour: 356.2088687 | 1000 lb/h |
| Feed water flow2 | Pounds of water fed to boiling tank per hour: 355.8996857 | 1000 lb/h |
| Steam flow1 | Pounds of steam running the turbine per hour: 350.0534074 | 1000 lb/h |
| Steam flow2 | Pounds of steam running the turbine per hour: 350.0534074 | 1000 lb/h |
| Fuel gas flow | 400.8499634 | SCFH  60F, 14psig |
| Airflow | Mass flow rate of the air in the inlet of the boiler: 1.08702353 | Inch water |
| Forced draft fan pressure | 9.71657224 | Inch water |
| Fuel gas pressure | Pressure of the fuel being fed: 7.576266895 | Psig |
| Flue gas pressure1 | Pressure of the exhaust gas stream:  0.008246493 | Psig |
| Furnace pressure | The pressure of the furnace burning the fuel gas: 3.716080412 | Inch water |
| Ignitor pressure | Pressure in the gas igniter: 7.780428486 | Psig |
| Steam drum pressure1 | Pressure inside the steam drum: 628.2707081 | Psig |
| Steam drum pressure2 | Pressure at the steam drum exit: 649.4017829 | Psig |
| Flue gas pressure2 | Pressure of the exhaust gas stream: 0.310983728 | Inch water |
| Deaerator temperature | Temperature in the deaerator: 298.2767765 | Fahrenheit |
| Steam drum temperature | Temperature inside the steam drum: 597.7321536 | Fahrenheit |
| Superheater temperature1 | Temperature inside the super heater at location 1: 1760.4008775 | Fahrenheit |
| Superheater temperature2 | Temperature inside the super heater at location 2: 765.367391 | Fahrenheit |
| Flue gas temperature | Exhaust gas stream temperature: 335.9888114 | Fahrenheit |
| Oxygen percentage | Oxygen Concentration in the exhaust stream: 2.010391552 | % |

TABLE II new features after modification

|  |  |  |
| --- | --- | --- |
| **New features** | **Equations** | **Units** |
| A/F |  | 1 |
| FW/SF |  | 1 |
|  |  | m3/kmol |
| Pfuel/Pfurnace |  | 1 |
|  |  | m3/kmol |

# Analysis and results

In this section, we begin by benchmarking the different models, next we would choose the best-performing model from the previous subsection and alter the input features to increase the model’s performance.

## choosing the best performing model

To start, we will develop 3 variations that differ in the number of layers of each of the 3 models so in total we have 9 models. All models have ReLU as an activation function in

their hidden layers, learning rate of 0.0001, ADAM function as an optimizer, and a linear activation function in the output layer since we have only one single output. Linear activation functions are simple to implement and understand, easy to compute, and are efficient. The models were trained for 100 epochs with a batch size of 128, except for TCNs which were trained with a batch size of 64. Four-layer models had 128/32 neurons in their hidden layers, five-layer models had 128/64/32 neurons, and six-layer models had 128/64/32/16 neurons. After running the models through multiple trails, we got the results shown in TABLE 3. LSTM is performing well according to the RMSE, MAPE, and MaxRSE, but TCN model is seen to perform better according to MaxAPE. We may conclude that both LSTM and TCN perform well, but we will choose the 6 layered LSTM model since it dominates most of the metrics and is overall the most consistent model. We can’t choose FNN as our final model since it seems to underperform because our data is time dependent (i.e., time series data).

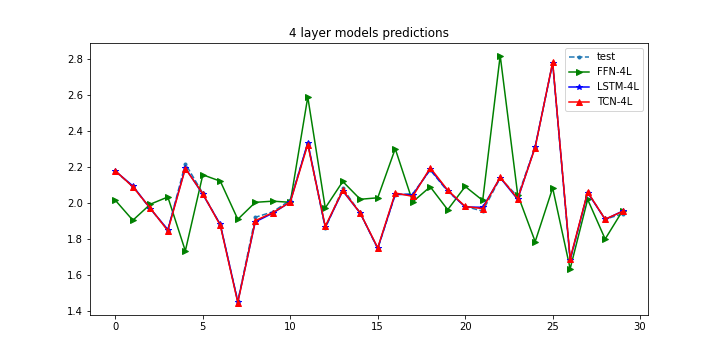
|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |

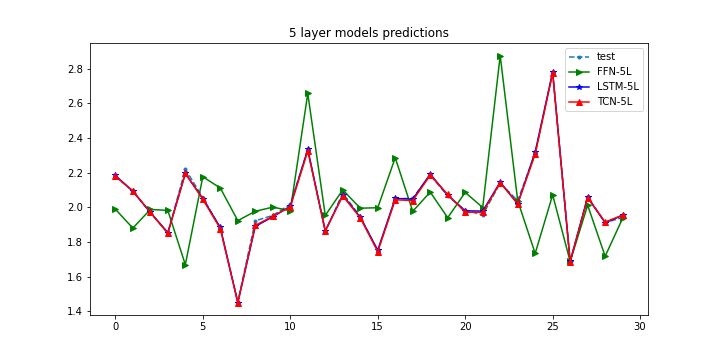
Fig. 7. Histogram plots for all features

Moreover, in real life models we should allow a window of error since AI models can be erroneous. By allowing a margin of error, we can mitigate the risk of a mistake that could have serious consequences[22], [23]. Hence, we chose to allow some window of error in the MaxAPE metric since it represents the largest forecasted value in the test. This grants an advantage of gauging the uncertainty in the forecast. Fig. 9 shows a side by side comparison of the models performance, as can be seen from the graph that FNN are not doing well in predicting O2 percentages, while LSTM and TCN both have similar acceptable predictions.

|  |  |
| --- | --- |
|  |  |
|  |  |
|  | |

Fig. 8. Scatter plot for all input features vs Oxygen (i.e., output feature)





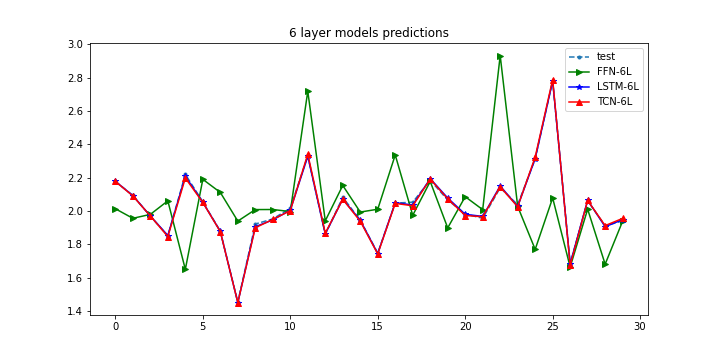


Fig. 9. Predictions of all models

TABLE III all model results after testing

| Models | Metrics | | | |
| --- | --- | --- | --- | --- |
| RMSE | MaxAPE | MaxRSE | MAPE |
| FNN-4 layers | 0.0908 | 40.4% | 0.5441 | 3.41% |
| LSTM-4 layers | 0.0071 | 1.6% | 0.0349 | 0.25% |
| TCN-4 layers | 0.0073 | 1.5% | 0.0337 | 0.27% |
| FNN-5 layers | 0.0864 | 37.9% | 0.5082 | 3.23% |
| LSTM-5 layers | 0.0074 | 1.8% | 0.0373 | 0.26% |
| TCN-5 layers | 0.0074 | 1.8% | 0.0370 | 0.27% |
| FNN-6 layers | 0.0836 | 39.5% | 0.5314 | 3.13% |
| LSTM-6 layers | **0.0056** | 1.7% | 0.0312 | 0.21% |
| TCN-6 layers | 0.0078 | 2.1% | 0.0443 | 0.29% |

## Altering LSTM’s input features

To enhance the performance of the LSTM model, refining the selection of input features is crucial. We evaluated different feature subsets by systematically removing features and observing the impact on model accuracy. The correlation heatmap in Fig. 6 provided an initial perspective on linear dependencies between features, but the final selection was guided by empirical testing rather than correlation alone.  
  
Through iterative experiments, we observed that removing the A/F feature had minimal effect on prediction accuracy. By progressively eliminating less impactful features and retraining the model, we determined that the best-performing LSTM configuration utilized a subset of five key features Table IV. This approach ensured that the model retained essential information while reducing complexity, leading to improved performance.

TABLE IV  
 LSTM result after testing with most correlated feature with Oxygen

| LSTM 6 layers | Metrics | | | |
| --- | --- | --- | --- | --- |
| RMSE | MaxAPE | MaxRSE | MAPE |
| All 5 Features | **0.0056** | 1.7% | 0.0312 | 0.21% |
| , FW/SF, Pfuel/Pfurnace, | 0.0077 | 1.7% | 0.0390 | 0.26% |
| , FW/SF, Pfuel/Pfurnace | 0.0069 | 1.4% | 0.0391 | 0.24% |
| , FW/SF | 0.0067 | 1.3% | 0.0305 | 0.25% |
|  | 0.0067 | 1.4% | 0.0397 | 0.23% |

# Conclusion

In conclusion, we analyzed the boiler's states using Long-short Time Memory (LSTM) models and showed that they are the best performing models. We used different techniques of dimensionality reduction to improve the model's performance, and we concluded that it is always better to reduce the number of input features. We also used correlations as an estimator for the importance of features, and we concluded that the 6-layer LSTM with the most 5 correlated input variables model is performing better than other models with Root Mean Square Error (RMSE), Maximum Absolute Percentage Error (MaxAPE), Maximum Root Square Error (MaxRSE), and Mean Absolute Percentage Error (MAPE) scores 0.0056, 1.7%, 0.0312, and 0.21% respectively. A new method for improving the performance of LSTM models by reducing the number of input features was adopted, and the results indicated that the LSTM model with all 5 features is the best performing model for boilers’ state forecasting having the lowest RMSE and MAPE scores 0.0056, and 0.21% respectively. Moreover, two structures of LSTM were performing well, the 5 input-based model, and 2 input-based model. However, 5 input-based model is preferred more since it gives more freedom to control the boiler. Some complexity in data is needed when developing complex models such as LSTMs. For future research, we suggest developing a reinforcement learning model and further analysis of the current model. Reinforcement learning (RL) models can learn from experience and improve their performance over time, which could lead to better results in boiler state analysis. An integrated control model using RL and our selected model as an agent can result in more accurate boilers’ state forecasting.

# ACKNOWLEDGEMENT

The authors would like to acknowledge the support provided by Saudi Data & AI Authority (SDAIA) and King Fahd University of Petroleum & Minerals (KFUPM) under SDAIA-KFUPM Joint Research Center for Artificial Intelligence (JRC-AI) grant No. JRCAI-RG-06. The support

received from King Abdullah City for Atomic and Renewable Energy (K. A. CARE) is also acknowledged.

##### References

[1] K. Gurney, *An introduction to neural networks*. CRC press, 1997.

[2] K. Lang, M. Zhang, Y. Yuan, and X. Yue, “Short-term load forecasting based on multivariate time series prediction and weighted neural network with random weights and kernels,” *Cluster Comput*, vol. 22, pp. 12589–12597, Sep. 2019, doi: 10.1007/s10586-017-1685-7.

[3] K. E. ArunKumar, D. V. Kalaga, C. Mohan Sai Kumar, M. Kawaji, and T. M. Brenza, “Comparative analysis of Gated Recurrent Units (GRU), long Short-Term memory (LSTM) cells, autoregressive Integrated moving average (ARIMA), seasonal autoregressive Integrated moving average (SARIMA) for forecasting COVID-19 trends,” *Alexandria Engineering Journal*, vol. 61, no. 10, pp. 7585–7603, Oct. 2022, doi: 10.1016/j.aej.2022.01.011.

[4] A. O. Aseeri, “Effective RNN-Based Forecasting Methodology Design for Improving Short-Term Power Load Forecasts: Application to Large-Scale Power-Grid Time Series,” *J Comput Sci*, vol. 68, Apr. 2023, doi: 10.1016/j.jocs.2023.101984.

[5] Y. Zhang, M. Zhong, N. Geng, and Y. Jiang, “Forecasting electric vehicles sales with univariate and multivariate time series models: The case of China,” *PLoS One*, vol. 12, no. 5, pp. e0176729-, May 2017, [Online]. Available: https://doi.org/10.1371/journal.pone.0176729

[6] A. H. Bukhari, M. A. Z. Raja, M. Sulaiman, S. Islam, M. Shoaib, and P. Kumam, “Fractional neuro-sequential ARFIMA-LSTM for financial market forecasting,” *IEEE Access*, vol. 8, pp. 71326–71338, 2020, doi: 10.1109/ACCESS.2020.2985763.

[7] N. Zheng *et al.*, “Predicting COVID-19 in China Using Hybrid AI Model,” *IEEE Trans Cybern*, vol. 50, no. 7, pp. 2891–2904, Jul. 2020, doi: 10.1109/TCYB.2020.2990162.

[8] Y. Hu, C. Hu, S. Fu, P. Shi, and B. Ning, “Predicting the popularity of viral topics based on time series forecasting,” *Neurocomputing*, vol. 210, pp. 55–65, Oct. 2016, doi: 10.1016/j.neucom.2015.10.143.

[9] B. Dhivya Bharathi, B. Anil Kumar, A. Achar, and L. Vanajakshi, “Bus travel time prediction: a log-normal auto-regressive (AR) modelling approach,” *Transportmetrica A Transport Science*, vol. 16, no. 3, pp. 807–839, Jan. 2020, doi: 10.1080/23249935.2020.1720864.

[10] S. R. Srivastava, Y. K. Meena, and G. Singh, “Forecasting on Covid-19 infection waves using a rough set filter driven moving average models,” *Appl Soft Comput*, vol. 131, Dec. 2022, doi: 10.1016/j.asoc.2022.109750.

[11] X. Zhang, C. Zhong, J. Zhang, T. Wang, and W. W. Y. Ng, “Robust recurrent neural networks for time series forecasting,” *Neurocomputing*, vol. 526, pp. 143–157, Mar. 2023, doi: 10.1016/j.neucom.2023.01.037.

[12] K. Cho *et al.*, “Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation,” Jun. 2014, [Online]. Available: http://arxiv.org/abs/1406.1078

[13] S. Bai, J. Z. Kolter, and V. Koltun, “An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling,” Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.01271

[14] X. Jia, Y. Sang, Y. Li, W. Du, and G. Zhang, “Short-term forecasting for supercharged boiler safety performance based on advanced data-driven modelling framework,” *Energy*, vol. 239, p. 122449, Jan. 2022, doi: 10.1016/J.ENERGY.2021.122449.

[15] P.-H. Wang, L.-L. Li, Q. Chen, and Y.-H. Dong, “Research on applications of artificial intelligence to combustion optimization in a coal-fired boiler,” vol. 24, pp. 184–188, Oct. 2004.

[16] T. Yang, K. Ma, Y. Lv, and Y. Bai, “Real-time dynamic prediction model of NOx emission of coal-fired boilers under variable load conditions,” *Fuel*, vol. 274, p. 117811, 2020, doi: https://doi.org/10.1016/j.fuel.2020.117811.

[17] Z. Yuan, L. Meng, X. Gu, Y. Bai, H. Cui, and C. Jiang, “Prediction of NOx emissions for coal-fired power plants with stacked-generalization ensemble method,” *Fuel*, vol. 289, p. 119748, 2021, doi: https://doi.org/10.1016/j.fuel.2020.119748.

[18] J. Hu, H. Zhou, Y. Zhou, H. Zhang, L. Nordströmd, and G. Yang, “Flexibility Prediction of Aggregated Electric Vehicles and Domestic Hot Water Systems in Smart Grids,” *Engineering*, vol. 7, no. 8, pp. 1101–1114, Aug. 2021, doi: 10.1016/J.ENG.2021.06.008.

[19] W. He, “Load Forecasting via Deep Neural Networks,” in *Procedia Computer Science*, Elsevier B.V., 2017, pp. 308–314. doi: 10.1016/j.procs.2017.11.374.

[20] R. Wan, S. Mei, J. Wang, M. Liu, and F. Yang, “Multivariate Temporal Convolutional Network: A Deep Neural Networks Approach for Multivariate Time Series Forecasting,” *Electronics (Basel)*, vol. 8, no. 8, 2019, doi: 10.3390/electronics8080876.

[21] M. Nemitallah, “Boilers Optimal Control for Maximum Load Change Rate,” *J Energy Resour Technol*, vol. 136, p. 31301, Oct. 2014.

[22] F. Prado, M. C. Minutolo, and W. Kristjanpoller, “Forecasting based on an ensemble Autoregressive Moving Average - Adaptive neuro - Fuzzy inference system – Neural network - Genetic Algorithm Framework,” *Energy*, vol. 197, Apr. 2020, doi: 10.1016/j.energy.2020.117159.

[23] S. Selvin, R. Vinayakumar, E. A. Gopalakrishnan, V. K. Menon, and K. P. Soman, “Stock price prediction using LSTM, RNN and CNN-sliding window model,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1643–1647. doi: 10.1109/ICACCI.2017.8126078.