The 9th European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS Congress 2024
3–7 June 2024, Lisboa, Portugal

DATA SCIENCE APPROACH TO MODELLING AND OPTIMISING DATA CENTER ENERGY EFFICIENCY

Yibrah Gebreyesus[1], Damian Dalton[2], D.De Chiara[3] and M.Chinnici[4]

[1] School of Computer Science
University College of Dublin (UCD), D04 V1W8, Ireland
e-mail: yibrah.gebreyesus@ucdconnect.ie

[2] School of Computer Science
University College of Dublin, Do4 V1W8, Ireland.
damian.dalton@ucd.ie

[3] ENEA-ICT Division
C.R. Portici Piazzale Enrico Fermi, Portici NA 80055, Italy
davide.dechiara@enea.it

[3] ENEA- ICT Division
C.R Casaccia Via Anguillarese 301, ROMA 00123, Italy
marta.chinnici@enea.it

Key words: Data Center, Energy Efficiency, Fan speed, Ambient temperature, CNN, LSTM, CNN-LSTM and Sensitivity Analysis.

Abstract. The rapid digitisation has led to a surge in DCs energy consumption, projected to reach 21% of global demand by 2030. This poses significant operational and environmental impacts. The integration of IoT and sensors has generated vast amounts of operational data, which, if effectively modelled, can enhance energy efficiency and reduce costs. Traditional solutions are inadequate due to the complexity and non-linear system interactions of DCs and the growing volume of monitored operational data. This paper develops a comprehensive hybrid CNN-LSTM deep learning model that combines Convolutional Neural Networks' (CNNs') ability to recognise spatial and complex operational patterns with Long Short-Term Memory (LSTMs') strength in handling temporal dependencies to predict DC ambient temperature, thereby enhancing energy efficiency while maintaining thermal compliance. Trained on operational data from the Enea HPC CRESCO6 cluster and compared its effectiveness with CNN and LSTM, the model outperformed CNN and LSTM models, achieving a Mean Absolute Error (MAE) of 0.0401 in predicting ambient temperature every 15 minutes. A sensitivity analysis with cooling fan speed set-points ($\pm$10%, $\pm$25%, and $\pm$50%) showed the model's adaptability, with a $\pm$50% set-points demonstrates for monitoring and optimising cooling settings, ultimately enhance energy efficiency.

1 INTRODUCTION

Data Centres (DCs) are the core digital infrastructure of the modern economy. With the increasing demand for data-driven applications, DC services are expanding rapidly, leading to considerable energy consumption due to their high demand for cooling and power IT resources. Currently they consume for approximately 3% of global electricity demand, and with the rapid pace of digitisation, this figure is projected to increase to 21% by 2030 [1], equating 23% of the total CO2 emissions. This could significantly impact the carbon-neutral economy by 2050.

According to a recent report, the energy efficiency of DCs, measured by the Power Usage Effectiveness (PUE) metric, was found to be 1.58 [2]. The ideal PUE value is 1.0, indicating that currently, 58-60% of the energy is dissipated as heat or energy overhead. Beyond the environmental impact, if not effectively regulated, this excess heat can lead to critical issues, including IT equipment failure, performance degradation, shortened equipment lifespan, and increased energy consumption due to inefficient cooling management. These can be attributed to over 20% of total DC operational costs. This underscores the need for effective approaches to optimise DC management, enhancing both energy and cooling efficiency. Effectively regulating fan speed is essential for expelling exhaust heat generated from IT equipment while maintaining thermal compliance and ensuring the safe operation of IT systems. This approach ultimately enhances energy efficiency through improved cooling management, which largely relies on operational optimisation.

The most commonly used methods for optimising DC energy and cooling efficiency include heuristic approaches, such as those by [4], [5], [6] and [7], and Computational Fluid Dynamics (CFD) approaches, as demonstrated in research by [8], [9], [10] and [11]. These engineering formula-based solutions model, simulate, and regulate DC airflow. Additionally, statistical and mathematical analytical approaches are employed to quantify and analyse operational behaviours, enabling informed decision-making. However, these solutions have struggled to yield optimal results due to the growing physical complexity of DCs, their dynamically changing behaviours, non-linear system interactions, and the increasing volume of operational management data. Traditional approaches are computationally expensive, lack adaptability, and often fail to transform the vast amount of data into actionable insights. Furthermore, the sheer number of possible configurations and fluctuating set points over time make it challenging to identify the optimal solution. Given constraints such as time, IT load variability, weather conditions, and the need for a stable DC environment, evaluating every possible feature combination to optimise efficiency becomes impractical with these rule-based and reactive methods.

More recently, data-driven Artificial Intelligence (AI) and Machine Learning (ML) approaches have been introduced to effectively model and optimise DC operations. These methods excel in learning from complex and vast amounts of data, enabling them to produce more accurate predictions without relying on prior knowledge of the underlying physical infrastructure. For instance, Google implemented a simple Neural Network (NN) machine learning model, claiming a 40% reduction in cooling costs [12]. Other research, such as that by [13], applied various ML models, including Random Forest (RF), eXtream Gradient Boost (XGB), and NN, demonstrating promising results in improving energy efficiency. Additionally, [15] utilised ML for thermal characterisation, thereby enhancing energy efficiency, while [16] showcased ML-based inlet temperature characterisation and other like [14]. These studies and others in literature highlight

the potential of ML in optimising various aspects of DC operations.

However, these methods are still in their early stages and often focus on optimising cooling and IT systems individually. This is problematic because these systems are strongly interdependent—optimising one can lead to efficiency trade-offs in the other. Additionally, environmental conditions significantly impact overall energy efficiency, underscoring the need for a more integrated optimisation approach that considers these complex interactions.

This research aims to develop a comprehensive hybrid CNN-LSTM deep learning prediction framework to model plant DC performance and accurately predict DC room temperature (ambient temperature) every 15 minutes, thereby regulating cooling fan speed. This will ultimately enhance energy efficiency and maintain thermal compliance. The model was selected for its ability to capture spatial hotspots and recognise complex operating feature interactions through its CNN component, as well as its capability to capture temporal dependencies and predict future temperature values using its LSTM component. It trained using various operational features of DC encompassing IT, cooling and environmental features obtained from ENEA HPC CRESCO6 cluster, consists of 55 normalised features mapped to one normalised output (i.e, ambient temperature). To evaluate the model's robustness, adaptability, and responsiveness to the dynamic operations of a DC, a sensitivity analysis was conducted. This analysis utilised various cooling system fan speed set-points ($\pm$10%, $\pm$25%, and $\pm$50%) selected in consultation with domain experts and tested through extensive experimentation using historical data, all while ensuring compliance with ASHRAE operating standards. Sensitivity with larger values indicates that the room for monitoring and optimising fan settings, ultimately enhance energy efficiency and ensure thermal compliance. Hence, this paper will contribute a valuable approach in the field, enabling operators to effectively plan and manage energy demand, enhance operational efficiency, reduce energy and environmental costs through informed decision-making.

The rest of this paper is organised as follows: Section 2 details the methodology, including model development processes and optimisation strategies, Section 3 presents the experimental results and discussion, and Section 3.1 presents sensitivity analysis , and Section 4 provides conclusions and outlines future work.

2 Research Methodology

As illustrated in Fig. 1, this research develops a holistic AI-driven framework specifically designed to model and predict DC ambient temperature every 15 minutes, thereby optimising cooling fan speed settings to enhance energy efficiency while maintaining thermal compliance. Given the complex system interactions, dynamically changing DC operational behaviour, and the increasing volume of operational monitoring data, this research focuses on a hybrid deep learning method, CNN-LSTM, which combines CNN and Long LSTM networks. CNN-LSTM uses the capability of CNN for capturing the complex operating features interaction and spatial pattern recognition of the DC behaviour while LSTM uses for capturing the temporal dependencies of the time series data to predict the next value. The model were trained using diverse physical and operational data (see Section 2.1 for data source and descriptions), capturing the complex and non-linear interactions among features that influence ambient temperature. 70% of the data was used for training, while the remaining 30% was reserved for model validation, preserving time order. By learning from this comprehensive dataset, the model able to predict

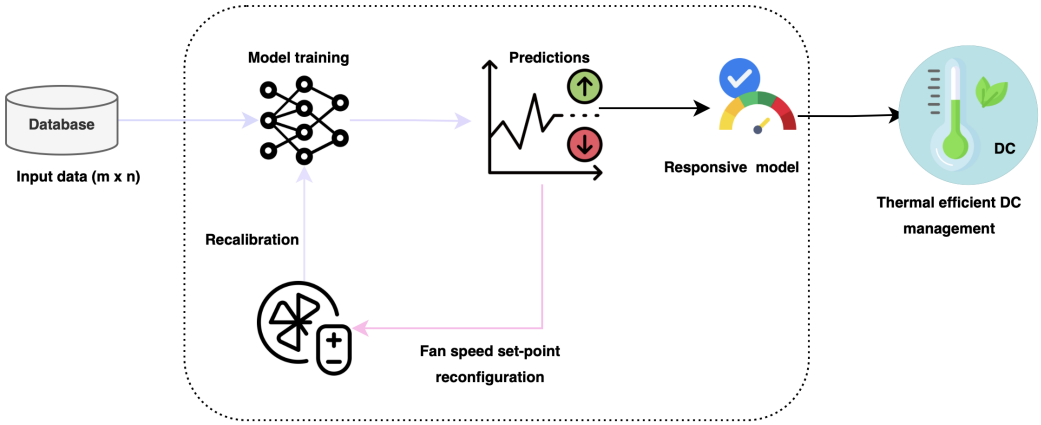

Figure 1: AI-driven Ambient temperature prediction and optimisation within the recommended thermal condition.

DC ambient temperature accurately. The proposed model's effectiveness was evaluated against CNN and LSTM's most effective NN models at solving complex challenges analogs to DC operating behaviours by training with the same dataset and operating platform. Their performance evaluated using Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) error metrics. The model with the lowest error is considered the most suitable for accurately modelling and predicting ambient temperature within a DC.

Additionally, we simulates sensitivity analysis to evaluate the model's adaptability and responsiveness to evolving conditions by adjusting various cooling system fan speed set-points ($\pm 10\%$, $\pm 25\%$, and $\pm 50\%$). Fig. 1 illustrates that the framework development process involves several key steps: Input data provided in Section 2.1, model development and evaluation detailed in Section 2.2, and a sensitivity analysis optimisation strategy by reconfiguring fan speed detailed in Section3.1.

2.1 Data Source and Descriptions

This study uses training data from the ENEA HPC CRESCO6 cluster, consisting of 434 computing nodes and 20,832 cores. The data includes IT parameters (e.g., power, system utilisation), cooling system parameters (e.g., air supply, fan speed), and environmental conditions (e.g., temperature, humidity). These data streams are monitored by sensors and accessed via an Intelligent Platform Management Interface (IPMI). The data is stored in MySQL across three tables—IT, cooling system, and environmental data—collected at intervals ranging from seconds to minutes.

After data collection, the necessary data preprocess performed for making it suitable for AI and ML models. Irrelevant features were removed, and missing values were interpolated. The data was resampled into 15-minute intervals and aggregated into a compact dataset of 34553 instances with 50 features, covering approximately one year. We then performed feature engineering to extract temporal features, adding 5 key predictors. The final dataset was structured

as 34553 instances with 55 features (see Section A.1 for details).

The features normalised into the same scale, and the values of a feature vector z are mapped to the range [-1, 1], ready to fed the ML models, computed as:

$$z_{norm} = \frac{z - mean(z)}{max(z) - min(z)}$$

where z_{min} and z_{max} are the minimum and maximum values of the feature vector z, respectively.

2.2 Deep Learning Model Buildings

Before delving into the proposed CNN-LSTM model for ambient temperature prediction, the descriptions and technical implementations of CNN and LSTM models are provided in the following sections.

2.3 Convolutional Neural Network (CNN)

CNNs, introduced by Yann LeCun et al. in 1998 [17], excel in spatial data tasks like image processing and video recognition [18], [19] and [20]. They learn spatial hierarchies through convolutional layers and reduce dimensionality with pooling layers. CNNs come in three forms: 1D for sequential data, 2D for images, and 3D for volumetric data. Given the time-series nature of DC operating behaviours, we use 1D-CNNs to predict DC ambient temperature. As shown in Fig.2, our 1D-CNN model consists of five layers, including:

- **Input Layer:** The CNN processes a matrix of DC operating time-series data with 55 normalised features (i.e., f_1, f_2, f_3... f_{55}) over 24 time steps to predict ambient temperature. A.1 provided feature lists and their descriptions.
- **1D-convolution (Cov1D) Layer**: The Conv1D layer detects spatial patterns and seasonality in the reshaped time-series data through the following computations:
  - Cov1D: The CNN uses convolutional filters to scan the time-series data, detecting local patterns. For instance, with a filter size of 3 for processing 3 consecutive time steps, makes the CNN more faster.
  - Mathematically:Each filter performs element-wise multiplication with a segment of the input data, sums the results, and slides across the entire data. If x is the input data, k is the filter or kernel size, and b is the bias, the convolution output y will be calculated as: $y = (x * w) + b$, where $*$ denotes the convolution operation.
  - After applying the filters, $ReLU$ activation function is used to introduce non-linearity into the model. This helps the model learn more complex patterns from the input data x. It is calculated as : $ReLU(x) = max(0, x)$

    $$y_t = ReLU\left(x_t * k_{wt} + b_t\right), \tag{1}$$

    where y_t is the output value after convolution, which depends on the hidden layer or it is the final predicted output, $ReLU$ is the activation function, x_t is the input feature, k_t is the weight of the kernel, and b_t is the kernel bias. The result y_t is part of

the output feature map generated by the convolutional layer. The process is repeated for all valid time steps, sliding the filter across the time-series data.

- **Pooling Layer:** This layer performs down-sampling by reducing the dimensionality of the data while retaining essential features, helping to reduce computation and prevent overfitting. Reduce the size of the feature map from Cov1D by applying a pooling operation (e.g., max pooling): $y_t^{pooled} = max(y_t, y_{t+1}, y_{t+p-1})$. Where p is the pooling size and y_t^{pooled} is the pooled output for time t. The output of the last pooling or convolutional layer is a 3D tensor of shape as (batch_size, length, channels). Where **batch_size** is the number of samples in the batch, **length** is the number of time steps or sequence length after applying convolution and pooling, and **channels** is the number of filters used in the convolutional layer.

- **Flattening Layer:** This layer converts the pooled feature maps into a one-dimensional vector (suitable for time series prediction), preparing the data input into fully connected layers. The flatten layer takes the 3D tensor and converts it into a 2D tensor by collapsing the **length** and **channels** dimensions into a single dimension. The flattened output shape will be $(batch\_size, length * channels)$, which is $Flatten(F_t) =$ flatten (y_t^{pooled}). For instance, the output from the previous Cov1D layer has shaped as (32, 24, 55), then: The flatten layer will convert each sample's output into a 1D vector of size $24 \times 55 = 1320$. The resulting shape after flattening will be (32, 1320), where 1320 is the total number of features for each sample.

  Hence, the flattening is crucial because the subsequent fully connected layers expect a 1D input, where each node in the fully connected layer is connected to every feature in the flattened vector. This step is necessary for the model to learn from the spatial patterns detected by the convolutional layers. This flatten layer reshapes the 3D tensor output from Cov1D layers into a 2D tensor, where each sample's data is represented as a one-dimensional vector. This operation is critical for transitioning from convolutional layers to fully connected layers in a neural network.

- **Fully Connected Layers:** These layers are used to perform the final prediction result. The input to the fully connected layer is a 2D tensor, typically obtained after flattening the output of the last convolutional or pooling layer. Let's denote this input as a vector x of size n, where n is the number of features (e.g., n could be 1320 as calculated in the flattening step above). Hence the input of the full connected layer represented as: $x = [x_1, x_2, \ldots, x_n]$. The fully connected layer has weights and biases that need to be learned during training. Suppose the fully connected layer has m neurons. The weight matrix w will be of size $m \times n$, and the bias vector b will be of size m. Therefore, The output of the fully connected layer is computed by performing a matrix multiplication of the input vector x with the weight matrix w, and then adding the bias vector b. As illustrated in Fig. 2, the Fully Connected Layer (FCL) is shaped as Dense (64, 1). Here, 64 refers to the number of units in the FCL, and 1 represents the predicted output, which is the DC ambient temperature prediction.

$$\textbf{Ambient temperature prediction} = y_t = w_t * x_t + b_t. \tag{2}$$

This approach utilises CNNs to capture spatial patterns and correlations among various operational parameters, providing accurate predictions of DC ambient temperature within the data center environment. The model's granularity in predicting temperature at every time t allows for precise and timely forecasts. This enables optimal thermal performance by making decisions based on the predictive outputs.

Following the detailed technical implementation procedures, as illustrated in Fig.2, we developed a 5-layer 1D-CNN using the Keras library. Keras is a user-friendly interface written in python for defining neural network architectures, compiling models, and fitting them to data. The optimal hyperparameters, as shown in Table 3, were determined through extensive experimentation. We adjusted parameters such as kernel size and learning rate, among others, to identify the configurations that yielded the best performance. The parameters listed in the table represent those that produced the most effective model.

2.4 Long-Short-Term-Memory (LSTM)

LSTM (Long Short-Term Memory), proposed by Schmidhuber et al. in 1997 [21], is designed to analyse sequential data by addressing gradient issues in traditional RNNs. Unlike RNNs, which struggle with long-term dependencies due to vanishing or exploding gradients, LSTMs use a complex architecture with input, forget, output, and cell state gates. This structure effectively maintains and updates memory, making LSTMs well-suited for time-series predictions. The LSTM architecture, shown in Fig. 3, includes internal memory c_t and three gates: the forget gate f_t, the input gate i_t, and the output gate o_t. The memory c_t tracks input sequence dependencies, while the gates manage information flow. The forget gate f_t determines what information to remove from the cell state c_t, allowing LSTM to maintain relevant long-term dependencies and make accurate time-series predictions.

- Processing through LSTM Cells: The LSTM processes the input sequence (which may include various features related to data center operations) through its network of cells. Each cell computes intermediate states based on the forget gate, input gate, and cell state updates. Here, how each component of the LSTM cell works is given as follows:
  - Forget Gate (f_t):
  $$f_t = \sigma\left(w_f * [h_{t-1}, x_t] + b_f\right), \tag{3}$$
  Where:
    - $f_t$is the forget gate activation, ranging between 0 and 1.
    - w_f represents the weight matrix for the forget gate.
    - b_f is the bias term for the forget gate.
    - x_t is the input feature at the current time step t.
    - h_{t-1} is the output from the previous time step.
  - Input Gate (i_t) and Candidate Cell State $\widetilde{c}_t$:
  $$i_t = \sigma\left(w_{i^*}\left[h_{t-1}, x_t\right] + b_i\right), \tag{4}$$
  $$\widetilde{c}_t = tanh\left(w_c * [h_{t-1}, x_t]\, b_c\right), \tag{5}$$
  Where:

* i_t is the input gate activation, also ranging between 0 and 1.
* w_i is the weight matrix for the input gate.
* b_i is the bias term for the input gate.
* $\widetilde{c}_t$ is the candidate cell state, calculated using weights w_c and bias b_c.

– Cell State Update (c_t):

$$c_t = (f_{t^*} c_{t-1} + i_t * \widetilde{c}_t) \tag{6}$$

Where:

* c_t represents the updated cell state. This state is a combination of the previous cell state.
* c_{t-1} modulated by the forget gate f_t and and new candidate values $\widetilde{c}_t$ modulated by the input gate i_t.

– Output Gate (o_t) and and Hidden State (h_t):

$$o_t = \sigma\left(w_o * [h_{t-1}, x_t] + b_o\right), \tag{7}$$

$$h_t = o_{t^*} \tan h\left(C_t\right). \tag{8}$$

where:

* o_t is the output gate activation, ranging between 0 and 1.
* w_o is the weight matrix for the output gate.
* b_o is the bias term for the output gate.
* h_t is the hidden state that will be passed to the next time step and used for the final prediction. It is calculated by applying the output gate 0_t to the cell state's $tanh$ function.

The output h_t represents the DC ambient temperature prediction at time step t. We built a five-layer LSTM model using 55 normalised features and a 24-minute time step, mapped to one normalised output (ambient temperature). Optimal hyper-parameters for the LSTM model were identified through parameter tuning, as detailed in Table 4.

2.5 Proposed Hybrid CNN-LSTM Prediction Model

Following the CNN and LSTM descriptions, the proposed CNN-LSTM model integrates CNN for spatial feature extraction with LSTM for capturing temporal dependencies, predicts DC ambient temperature every 15 minutes and optimises cooling settings. Figure 4 shows the CNN-LSTM architecture, which includes input, Conv1D, pooling, LSTM, and Dense layers. The main steps for CNN-LSTM-based DC ambient temperature training and prediction procedures are as follows:

1. Input data: The CNN-LSTM model is trained using data on IT loads, cooling system parameters and environmental conditions factors, capturing key aspects influencing ambient temperature.

2. Data standardisation: The operational data is normalised and structured into 24-minute time series sequences. The model uses 55 normalised features per sequence to predict the DC ambient temperature.
3. Initialise weights and biases by setting each CNN-LSTM layer's weights and biases to their starting values.
4. The input data were sequentially transmitted via a convolutional process of the time-series data, capturing local spatial patterns within the input sequences. Pooling layers reduced the dimensionality of the feature maps generated by the convolutional layers, retaining the most relevant features while minimising computational complexity.
5. The output from the CNN layers is passed to LSTM layers, which capture the temporal dependencies within the time series data. The LSTM layers maintain memory states that allow the model to learn and retain information over long sequences.
6. The output from the LSTM layers is flattened into a one-dimensional vector. This flattened vector is then passed through fully connected (dense) layers that combine the learned features and predict the ambient temperature.
7. The output value computed by the output layer was compared with the true value of the data collection, and the related error was calculated.
8. The predicted result was compared with the actual values, the hyper-tuning parameters were configured, and the model was trained again iterative until produce good result.
9. Finally, retained the forecasting ready trained model for production data.

   Note: The technical details for CNN and LSTM are outlined in Sections 2.3 and 2.4.

. Following the description of the proposed CNN-LSTM architecture, the optimal hyper-parameters used for implementing the model are presented in Table5.

2.6 Model performance Evaluation Metrics

To evaluate the model performance, we used MAE, MSE, and RMSE error metrics computed as given in equations (9-11), respectively. The formulas calculated the deviation between predicted ($\hat{y}_i$) and actual values y_i. The computation is performed on the 20% of testing data.

$$MAE(y, \hat{y}) = \frac{\sum_{i=1}^{n} |\hat{y}_i - y_i|}{n}, \tag{9}$$

$$MSE(y, \hat{y}) = \frac{\sum_{i=0}^{n-1} (y_i - \hat{y}_i)^2}{n} \tag{10}$$

$$RMSE(y, \hat{y}) = \sqrt{\frac{\sum_{i=0}^{N=n-1} (y_i - \hat{y}_i)^2}{n}} \tag{11}$$

where $\hat{y}_i$ is the predictive value at time point i, y_i is the actual value n is the total testing sample. The lower the MAE, MSE, and RMSE values, the better the model performed at predicting DC energy consumption. y is nothing but the target variable (energy consumption).

3 Experimental Results and Discussions

This section presents the performance analysis and computational time complexity of the CNN, LSTM, and CNN-LSTM models for ambient temperature prediction. The models were trained on 70% of the data and validated on 30%, under consistent operating conditions. Their performance is evaluated using MAE, MSE, and RMSE, key error metrics for time series model performance analysis. Based on the experiments, Table 1 presents the performance analysis and computational time metrics of the models on the testing data. According to the experimental results, the hybrid CNN-LSTM model outperformed CNN and LSTM, achieving a lower MAE of 0.0331, MSE of 0.00251, and RMSE of 0.0501 while maintaining reasonable computational time. When compared to CNN-LSTM and LSTM, CNN is faster but performs less effectively due to its limitations in handling temporal dependencies. Specifically, CNN has an MAE of 0.0812, an MSE of 0.00352, and an RMSE of 0.0593, which are higher error than those of CNN-LSTM and LSTM. On the other hand, LSTM outperforms CNN in handling temporal dependencies, achieving an MAE of 0.0352, an MSE of 0.00341, and an RMSE of 0.00251. However, LSTM is slower than CNN due to its lack of parallel processing and its reliance on sequential feature extraction.

Table 1: Comparison of DC ambient temperature prediction methods evaluation indexes.

Methods	optimal feature set	MAE	MSE	RMSE	execution_time(sec)
CNN	55	0.0812	0.00352	0.0593	328
LSTM	55	0.0352	0.00341	0.0584	340
CNN-LSTM	**55**	**0.0331**	**0.00251**	**0.0501**	**330**

According to the experimental results, the CNN-LSTM model performed best by leveraging CNN's ability to capture spatial features and automate feature extraction, combined with LSTM's strength in modelling temporal dependencies, leading to the most accurate prediction of DC ambient temperature. Hence, compared to the other neural networks (NNs), the CNN-LSTM model is identified as the most suitable for modelling DC ambient temperature. Following this, a sensitivity analysis is conducted using various cooling system fan speeds to assess the CNN-LSTM model's adaptability and responsiveness to the evolving conditions (see section 3.1).

3.1 Optimising Cooling Efficiency and Ambient Temperature Regulation: Sensitivity Analysis and What-if Scenarios

Following the optimal model identification, a sensitivity analysis was conducted using varying cooling system fan speed set-points ($\pm$10%, $\pm$25%, and $\pm$50%). These set points were selected based on intensive experimentations and domain expert consultation. This analysis assessed how these changes impact ambient temperature predictions, aiming to identify optimal fan settings for thermal compliance and energy efficiency. The sensitivity analysis was applied to the last 24 time steps of the testing dataset to predict the next value and observe model performance under varying fan speeds. The sensitivity analysis, conducted using historical operational data, involves the following procedures:

1. Model Setup: Set a baseline DC ambient temperature prediction using historical data

where the fan speed is recorded along with other operational variables. This serves as the reference scenario for comparison.

2. Adjust Fan Speed Set-Points: Modify the fan speed set-points by ±10%, ±25%, and ±50%. These adjustments are designed to explore the effect of varying fan speeds on ambient temperature predictions.

3. Simulate with Adjusted Set-Points: Perform simulations using the CNN-LSTM model with the adjusted fan speed settings. Maintain other operational variables constant to isolate the impact of fan speed changes on ambient temperature predictions.

4. Analyse Results: Compare the simulation results with the baseline predictions to assess how different fan speed adjustments influence ambient temperature. Analyse the impact to determine the most effective settings.

5. Optimisation of Fan Speed Settings: Based on the sensitivity analysis results, optimal fan speed settings are identified. These settings aim to balance thermal management and enhance energy efficiency, providing recommendations for effective cooling system operation.

6. Finally, the findings from the sensitivity analysis enables to monitor, control and optimise fan settings, ultimately enhance energy efficiency while ensure thermal compliance.

The sensitivity analysis for each cooling system fan speed set point can be calculated based on the following assumptions:

1. Calculation Method: If a 10% increase in fan speed results in a 5% increase in the predicted ambient temperature, the sensitivity is calculated as:

$$Senstivity = \frac{Percentage\ Change\ in\ Prediction\ Result}{Percentage\ Change\ in\ Fan\ Speed}$$

2. Interpretation: A smaller sensitivity value indicates a more stable model, with less impact from variations in fan speed. Conversely, a larger sensitivity value implies that greater attention is needed to monitor, control, and optimize fan speed to effectively regulate thermal performance within the DC.

3. Optimal Model Selection: The model with greater responsiveness to fan speed variations proves to be the most effective for modelling and predicting DC ambient temperature. Its ability to adapt to changes in cooling system fan speeds demonstrates enhanced stability and robustness, making it well-suited for maintaining optimal thermal conditions in the DC.

Table 2 illustrates sensitivity analysis results simulated from historical data using ± 10%, ± 25%, and ± 50% cooling system fan speed controllable variable set-point. These set-points represent six different scenarios: the (+) sign indicates an increase in fan speed by the specified percentage, while the (-) sign indicates a decrease by the given percentage. These set points are applied to the last 24 time steps of the testing dataset to predict the subsequent values and

analyse how variations in fan speed influence the predictions. To assess the impact of fan speed variations on the CNN-LSTM-based DC ambient temperature predictions, we first establish a baseline by evaluating the CNN-LSTM model's predictions on unseen data with measured fan speeds and other features. This baseline serves as a reference for detecting the effects of changes in fan speed on the prediction results.

Refer to Table 2 where **column 2** presents the baseline predicted results and **column 5** shows predicted result after fan speed adjustments. By comparing these with the results from different fan speed scenarios, we can observe how variations in fan speed affect the predictions. This comparison allows us to identify the optimal fan settings that minimise error indices and improve thermal performance.

Table 2: What-if sensitivity analysis results.

Method	Baseline(°C)	Scenarios	fan_speed	After(°C)
CNN-LSTM	16.0564	1	+10%	16.021
-	16.0564	2	−10%	16.0918
-	16.0564	3	+25%	15.6856
-	16.0564	4	−25%	16.447
-	**16.0564**	**5**	**+ 50%**	**15.215**
-	**16.0564**	**6**	**− 50%**	**17.08**

To calculate the sensitivity when the fan speed scenarios are applied, we can use the following formula:

$$Senstivity = \frac{Percentage\ Change\ in\ Prediction\ Result}{Percentage\ Change\ in\ Fan\ Speed}$$

The empirical results presented in Table 2 can be interpreted as follows:

- Scenario (1): Increasing the fan speed by 10% results in a 0.220% decrease in the CNN-LSTM predicted ambient temperature. Sensitivity is calculated by dividing the percentage change in the predicted temperature by the percentage change in fan speed.

  $$Senstivity = \frac{0.220\%}{10\%} = 0.022$$

  Thus, a sensitivity value of 0.022 indicates the CNN-LSTM model's responsiveness to fan speed changes.

- In Scenario (2), a 10% decrease in fan speed leads to a 0.221% increase in the CNN-LSTM predicted ambient temperature.

  $$Senstivity = \frac{0.221\%}{10\%} = 0.0221$$

  Thus, a sensitivity value of 0.0221 indicates the CNN-LSTM model's responsiveness to fan speed changes.

- In Scenario (3), where the fan speed is increased by 25%, the observed change in the CNN-LSTM predicted result is a 2.3% decrease. Therefore, the sensitivity is calculated as is in Scenario (1). Hence, Sensitivity =

$$Senstivity = \frac{2.3\%}{25\%} = 0.0923$$

  Thus, a sensitivity value of 0.0923 indicates larger sensitivity to CNN-LSTM model's responsiveness compared the previous set-points.

- In Scenario (4), where the fan speed is decreased by 25%, the CNN-LSTM model's predicted result shows a 2.43% increase. The sensitivity analysis is calculated as follows:

  Sensitivity =

$$Senstivity = \frac{2.43\%}{25\%} = 0.097$$

  This sensitivity value of 0.097 indicates a high responsiveness of the CNN-LSTM model's predictions to decreases in fan speed.

- In Scenario (5), where the fan speed is increased by 50%, the CNN-LSTM model's predicted result shows a 5.24% decrease.

$$Senstivity = \frac{25.24\%}{50\%} = 0.105$$

  This sensitivity value of 0.105 indicates a high responsiveness of the CNN-LSTM model's predictions to decreases in fan speed.

- In Scenario (6), where the fan speed is decreased by 50%, the CNN-LSTM model's predicted result shows a 6.38% increase. The sensitivity analysis is calculated as follows: Sensitivity =

$$Senstivity = \frac{6.38\%}{50\%} = 0.128$$

  This sensitivity value of 0.128 indicates a high responsiveness of the CNN-LSTM model's predictions.

To summarise, sensitivity analysis is crucial for optimising DC operations by revealing how changes in key variables affect critical outputs. It enables better decision-making for performance optimisation and risk mitigation. The experimental results show that the AI model is adaptable and responsive to changing conditions. Larger sensitivity values highlight scenarios that need close monitoring and optimisation—such as the ±50% fan speed changes observed, which significantly impact ambient temperature and are thus prioritised for optimisation. Smaller sensitivity values indicate stable operations.

4 Conclusions and Future Works

In conclusion, the integration of AI-driven DC management can significantly improve energy efficiency, yield significant operational cost savings, and reduce footprints. For instance, the

proposed hybrid CNN-LSTM deep learning-driven framework is suitable for modelling plant performance learning from data and accurately predicting ambient temperature every 15 minutes with an average MAE of 0.0331. Its responsiveness and adaptability to evolving conditions demonstrate the model's suitability for dynamically predicting ambient temperature in rapidly changing scenarios within the DC. The sensitivity analysis reveals that a ±50% adjustment in fan speed showed greater sensitivity, highlighting areas that require closer monitoring, control, and optimisation. This approach ultimately enhances energy efficiency while maintaining thermal compliance align with the ASHRAE guideline's temperature range of 18°C to 27°C, allowable to 15°C to 32°C. Integrating this AI-driven approach can address the complex energy efficiency and optimisation challenges, yet traditional engineering tools are challenging to achieve. By integrating this technology, DC operators can effectively plan, manage and optimise DC operations while reducing energy and environmental costs.

Although our predictive model has shown promising results, we acknowledge the potential challenges and limitations of our study. For instance, accessing enough data quality and real-time operation requires extensive DC experts. Therefore, continuous monitoring, model recalibration, and real-time data integration are essential for maintaining accuracy over time. Our future research will primarily focus on expanding the model using larger datasets and modelling sensitivity analysis approach with different time length and seasonal variations.

Acknowledgements

This work was partly funded by the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No 778196. Marta Chinnici & Davide De Chiara were supported for this research by Project ECS 0000024 Rome Technopole, - CUP B83C22002820006, National Recovery and Resilience Plan (NRRP), Mission 4, Component 2 Investment 1.5", funded by the European Union – NextGenerationEU.

REFERENCES

[1] Anders SG Andrae and Tomas Edler2015. On global electricity usage of communication technology: trends to 2030. Challenges, 6(1):117–157.

[2] Uptime.2023. "Uptime institute global data centre survey 2023". https://www.brighttalk.com/resource/core/436687/uptime-global-data-center-survey-2023-webinar-presentation 931881.pdf. Accessed: (28-05-2024).

[3] Hessam Lavi.2022. "Measuring greenhouse gas emissions in data centres": the environmental impact of cloud computing. https://www.climatiq.io/blog/measure-greenhouse-gas-emissions-carbon-data-centres-cloud-computing. Accessed: (10-07-2024).

[4] Foudil Abdessamia et al. 2017. "An improved particle swarm optimisation for energy-efficiency virtual machine placement". In 2017 International Conference on Cloud Computing Research and Innovation (ICCCRI), IEEE, pages 7–13.

[5] Alangaram S, Balakannan SP.2023. "Optimisation of cloud data centre resources using meta-heuristic approaches". Soft Computing. Apr 28:1-1.

[6] Mohammadzadeh A et al., 2021. "Energy and cost-aware workflow scheduling in cloud computing data centers using a multi-objective optimisation algorithm". Journal of Network and Systems Management. 2021 Jul;29(3):31.

[7] Rasoul Rahmani et al.2020. "Modelling and optimisation of microgrid configuration for green data centres: A metaheuristic approach". Future Generation Computer Systems, 108:742–750.

[8] Ni,BowenJin et al., 2017. "Simulation of thermal distribution and airflow for efficient energy consumption in a small data centers". Sustainability, 9(4):664.

[9] Pogorelskiy S, Kocsis I 2023. "BIM and Computational Fluid Dynamics Analysis for Thermal Management Improvement in Data Centres". Buildings. Oct 19;13(10):2636.

[10] Han X et al., 2021. An open source fast fluid dynamics model for data center thermal management. Energy and Buildings. Jan 1;230:110599.

[11] melie Wibron et al.,2018. "Computational fluid dynamics modeling and validating experiments of airflow in a data center". Energies, 11(3):644.

[12] Gao,J.andJamidar,R.,2014."Machine learning applications for data centre optimisation".GoogleWhitePaper,21.

[13] Yang, Z. et al.(2022) "Increasing the energy efficiency of a data centre based on machine learning". Journal of Industrial Ecology 26(1), 323–335.

[14] Kumar R, Khatri SK, Diván MJ.2022. "Optimization of power consumption in data centers using machine learning based approaches: a review". International Journal of Electrical and Computer Engineering. Jun 1;12(3):3192.

[15] Grishina,A et al.(2020). "A machine learning solution for data centre thermal characteristics analysis". Energies 13(17), 4378.

[16] Lloyd, R., Rebow, M.(2018) "Data driven prediction model (ddpm) for server inlet temperature prediction in raised-floor data centers". In: 2018 17th IEEE Intersociety Conference on Thermal and Thermo mechanical Phenomena in Electronic Systems (ITherm). pp. 716–725.

[17] Yann LeCun et al.1998. "Gradient-based learning applied to document recognition". Proceedings of the IEEE, 86(11):2278–2324.

[18] Rikiya Yamashita et al.2018. "Convolutional neural networks: an overview and application in radiology". Insights into imaging, 9:611–629.

[19] Szegedy, C. et al.,2015."Going deeper with convolutions," Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1-9.

[20] He, K. et al., 2016."Deep residual learning for image recognition." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 770-778.

[21] Sepp Hochreiter and Jurgen Schmidhuber.1997. "Long short-term memory". Neural computation, 9(8):1735–1780.

Appendices

This section provides detailed feature lists and descriptions of the model architectures, operational processes, and hyperparameters used to implement the models with the Keras Python library.

A part-1: Features

A.1 List of features and descriptions

1. **Timestamp_measure:** is datetime for reading data streams from sensors (in min).
2. **sys_power:** is the total instantaneous Power of each node (watt).
3. **cpu_power:** is the CPU instantaneous Power of each node (watt).
4. **mem_power:** is each node's RAM memory instantaneous Power (watt).
5. **fan1a:** is the node fan speed expressed in RPM (revs per minute)
6. **fan1b:** is the fan Fan1b installed in the node expressed in RPM
7. **fan2a:** is the fan Fan2a installed in the node expressed in RPM
8. **fan2b:** is the fan Fan2b installed in the node expressed in RPM
9. **fan3a:** is the fan Fan3a installed in the node expressed in RPM
10. **fan3b:** is the fan Fan3b installed in the node expressed in RPM
11. **fan4a:** is the fan Fan4a installed in the node expressed in RPM
12. **fan4b:** is the fan Fan4b installed in the node expressed in RPM
13. **fan5a:** is the fan Fan5a installed in the node expressed in RPM
14. **fan5b:** is the fan Fan5b installed in the node expressed in RPM
15. **sys_util:** is the percentage of use of the system (%)
16. **cpu_util:** is the percentage of use of the CPU's(%).
17. **mem_util:** is the percentage use of the RAM (%).
18. **io_util:** is the node I/O traffic
19. **cpu1_Temp:** is the CPU1 temperature (°C)

20. **cpu2_Temp:** is the CPU2 temperature (°C)
21. **sysairflow:** is the airflow of the node in CFM (cubic feet to minute) .
22. **exh_temp:** is the exhaust temperature (air exit of the node) in (°C)
23. **amb_temp:** is the ambient temperature/room temperature in (°C)
24. **dcenergy:** is the DC energy demand (target variable) (Kwh) (Target variable)
25. **supply_air:** is a cold air or inlet temperature (°C)
26. **return_air:** is the ejected heat or warm air to the outside (°C)
27. **relative_umidity:** is the working humidity of the CRAC in (°C)
28. **fan_speed:** is the speed of the CRAC cooling system (RPM)
29. **cooling:** is the working intensity of the CRAC (%)
30. **free_cooling:** Not applicable, it is 0 values
31. **hot103_temp:** is the hot temperature monitored by hot103 sensor.
32. **hot103_hum:** hot_humidity monitored by hot103 sensor.
33. **hot101_temp:** hot temperature (°C) monitored by hot101 sensor.
34. **hot101_hum:** hot_humidity (%) monitored by hot101
35. **hot111_temp:** hot temperature (°C) monitored by hot111 sensor.
36. **hot111_hum:** hot_humidity (%) monitored by hot111 sensor.
37. **hot117_temp:** hot temperature (°C) monitored by hot117 sensor.
38. **hot117_hum:** hot_humidity (%) monitored by hot117 sensor.
39. **hot109_temp:** temperature (°C) monitored by hot109 sensor
40. **hot109_hum:** hot_humidity (%) monitored by hot109 sensor.
41. **hot119_temp:** hot_temperature (°C) monitored by hot119 sensor.
42. **hot119_hum:** hot_humidity (%) monitored by hot119 sensor.
43. **cold107_temp:** cold_temperature (°C) monitored by cold107 sensor.
44. **cold107_hum:** cold_humidity (%) monitored by cold107 sensor.
45. **cold105_temp:** cold_temperature (°C) monitored by cold105 sensor.
46. **cold105_hum:** cold_humidity (%) monitored by cold105 sensor.

47. **cold115_temp:** cold_temperature (°C) monitored by cold115.
48. **cold115_hum:** cold_humidity (%) monitored by cold115 sensor.
49. **cold113_temp:** cold_temperature (°C) monitored by cold113 sensor.
50. **cold113_hum:** cold_humidity (%) monitored by cold113 sensor.
51. **hour:** hours of the day
52. **day:** days of the week
53. **month:** months of the year
54. **quarter:** quarter of the year
55. **Holidays:**holidays of the year.

B part-2: Hyperparameter settings

B.1 Setting Models hyper-parametter

Table 3: Hyperparameters setting of a CNN with 55 normalised input features in 24 minutes window_size.

parameters	Values
input tensor	64*24*55
kernel	3
Activation function	*relu*
Fully Connected Layer	64*3
Time steps	24
Output layer	64*1
Optimiser	*adam*
Loss function	*mse*
epochs	100
batch_sizes	32

C part-3: Architectural view of the models

C.1 Models architectural overview

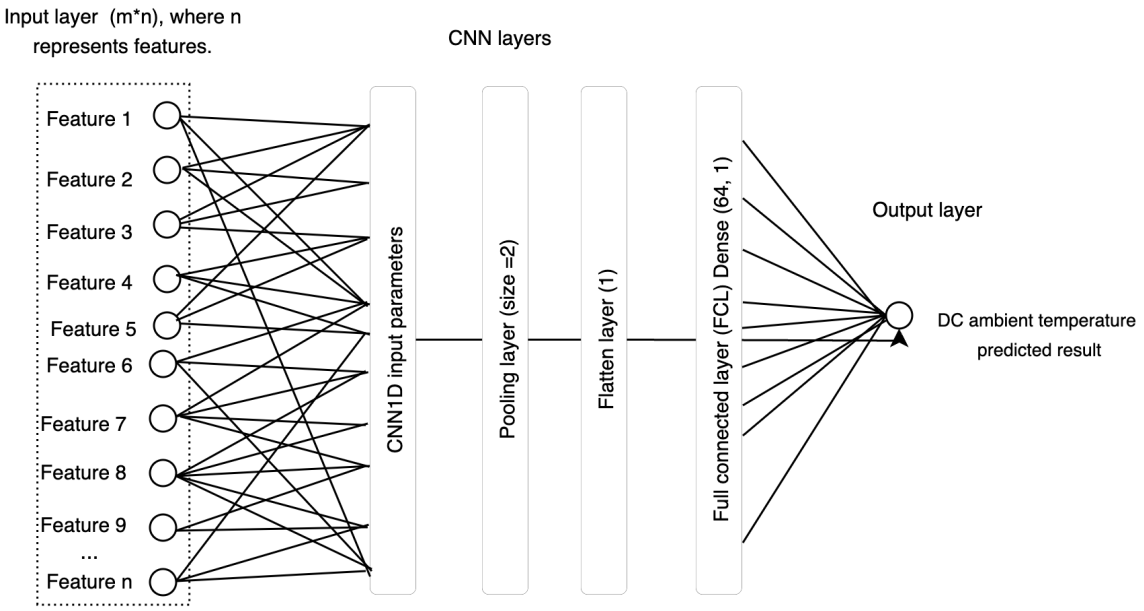

Figure 2: Architecture of CNN-based ambient temperature prediction.

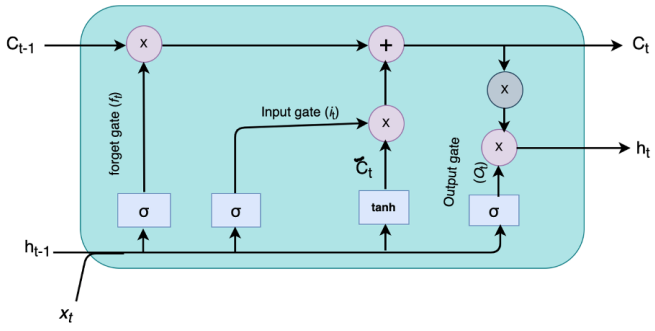

Figure 3: Basic LSTM Cell structure diagram [21].

Table 4: Hyperparameters setting of a LSTM with 55 normalised input features in 24 minutes window_size.

parameters	Values
input tensor	64*24*55
LSTM layers	4
Units	64
Time steps	24
Activation layer	Relu
Dense layer	64*1
Optimiser	*adam*
Loss function	*mse*
epochs	100
batch_sizes	32

Table 5: Hyperparameters setting of a CNN-LSTM with 55 normalised input features in 10 minutes window_size.

parameters	Values
input tensor	64*24*55
Con1D layer filter	32
kernel	3
Activation function	*relu*
Pooling layer_size	2
pooling activation function	*relu*
Number of LSTM layers	3
LSTM layers units	64
LSTM layer activation function	*relu*
Time steps	24
Dense layer	64*1
Output layer	64*1
Optimiser	*adam*
Loss function	*mse*
epochs	100
batch_sizes	32

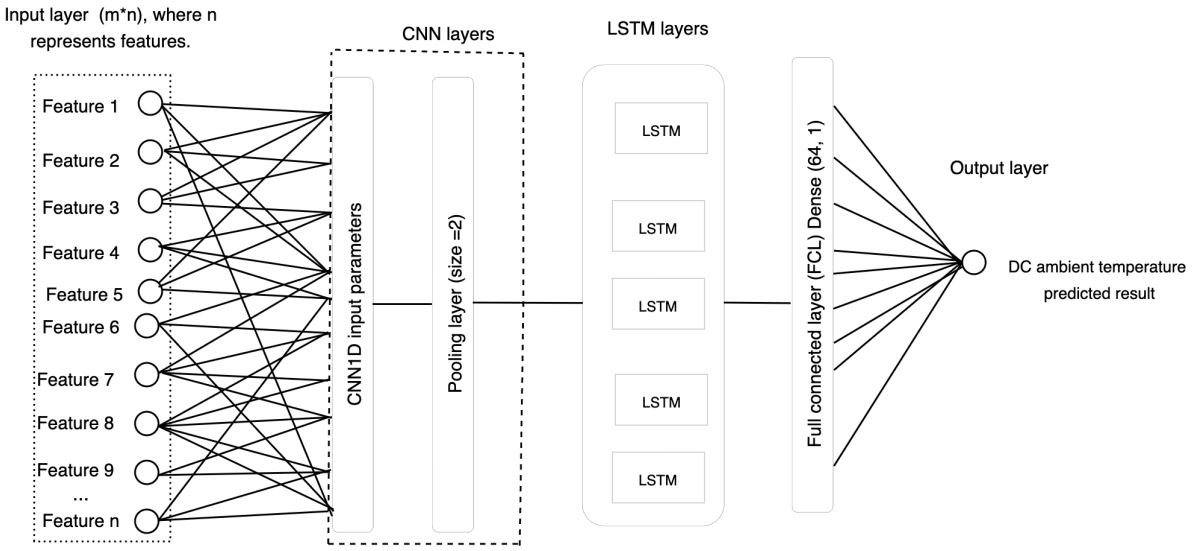

Figure 4: Architecture of CNN-LSTM Regressors for DC ambient temperature prediction.