Fog-Based Deep Learning for Real-Time Cold Chain Temperature Prediction Using IoT Data

Jeremiah Taguta[1 [0000−0001−8199−4307]], Jean Frederic Isingizwe Nturambirwe[2 [0000−0002−1794−7343]], and Clement Nthambazale Nyirenda[1,2 [0000−0002−4181−0478]]

[1] Department of Computer Science, University of the Western Cape, Robert Sobukwe Rd, Cape Town, 7535, South Africa
tagutaj@gmail.com
[2] eResearch Office, University of the Western Cape, Robert Sobukwe Rd, Cape Town, 7535, South Africa
fisingizwe@uwc.ac.za✉; cnyirenda@uwc.ac.za

Abstract. A third of the food produced globally and in South Africa is lost or wasted annually. Fresh fruits and vegetables (FFVs) contribute 44% of South Africa's wastage, with temperature abuse as the main cause due to their high sensitivity and perishability. Real-time cold chain management with predictive analytics is necessary to control parameters and minimise temperature breaks proactively. While Machine Learning (ML) can predict temperature, cloud deployment causes latency, bandwidth demands, and internet dependency, hindering real-time operations. Fog computing mitigates this by localising ML predictions, an area underexplored for the FFV cold chain. This study investigates fog-based Deep Neural Networks for predicting cold room temperatures in FFV cold chains, utilising IoT data from a South African apple cold room laboratory. SimPy, MinMax scaling, 75%/25% train/test splitting and a sequence length of 2 were used. The fog-based LSTM and GRU ensemble outperformed the cloud-based model with R^2 [0.9081, 0.9245] vs [0.5477, 0.5992], MAE [0.9999 °C, 1.6353 °C] vs [4.2983 °C, 4.6029 °C], MSE [10.0084,12.0685] vs [50.6492, 56.4827], processing time [0.0840, 0.0847] vs [0.3375, 0.3378], and Pearson R [0.9533, 0.9618] vs [0.7670, 0.7930] (95% confidence). Paired t-test and Wilcoxon test confirm fog's significant superiority. Both had high data utilisation of 99.8%, guaranteeing analysis for transmitted data. High accuracy and low processing time make fog ideal for real-time cold chains, reducing FFV wastage while improving sustainability, affordability, profitability, and food security. Future work will add sensors, fog nodes, compare models, consider different datasets and implement asynchronous sensor fusion and temperature break and cause predictions.

Keywords: Fog Computing · Temperature Prediction · Deep Neural Network · Machine Learning · Cold Chain · Food Wastage · Fresh Fruits and Vegetables

1 Introduction

Annually, 1.3 billion tonnes, equivalent to 1/3 of food produced for human consumption, are lost or wasted across the supply chain globally. Fresh fruits and vegetables (FFVs) make up 38% of this waste [7, 16, 31]. In South Africa, 50% of food is lost or wasted after harvest, with FFVs accounting for 44% despite their high nutritional value, while many people still lack access to a balanced diet. The horticultural industry is an important source of jobs and a contributor to the country's economy overall, whereby deciduous fruits such as apples, other pome as well as stone fruits account for a significant part of agricultural exports and local consumption. However, these products are highly perishable and considerable losses occur along their supply chain [7, 13, 31]. Cold storage is responsible for a significant chunk of the country's FFVs wastage (fruits, 20%; vegetables, 15%), hence, the focus of this study. FFVs are very sensitive to temperature deviations, so even minor deviations have a significant impact. Therefore, effective cold chain management is essential to reduce food and economic losses, greenhouse gas emissions, and prevent food-borne illness or deaths. This is especially pertinent since some markets reject consignments when cold chain temperature breaches occur [5, 7, 11, 12, 15].

In an attempt to predict the temperature of apples in a pallet stored in a cold room, a 3-layer Multi-layer Perceptron (MLP) was used [6]. In another study, a peephole Long Short-Term Memory (LSTM) and Backpropagation (BP)-MLP were used to predict a vegetable's cold room internal temperature using 1500 data points [20]. A 3-5 layer-MLP, Random Forest (RF), AdaBoost, Linear Regression (LR), Lasso and Support Vector Machines (SVM) were used to predict apples' core temperature [22]. Another study used a shallow and deep Multi-layer Feedforward Neural Network and a 3-layer LSTM to predict the temperature of citrus fruits and bananas with 22750 records collected from a multi-delivery truck[32]. A 1D-Convolutional Neural Network was used to predict sensor-free locations' temperature in a shipment transporting strawberries [4] while in another study, Extreme Learning Machines (ELM), Decision Trees, LR, Naive Bayes, RF and SVM were used to predict the internal temperature of a refrigerated transport carrying tomatoes with 28433 records [1]. A 4-layer LSTM was utilised to predict the internal temperature and when the cold chain breaks will occur in a navel orange's cold storage [17]. Another study predicted the internal temperature of an apple's cold room using traditional GRU, MLP, ELM and eXtreme Gradient Boosting [28].However, none of these studies [1, 4, 6, 17, 20, 22, 28, 32] explored ML deployment using Fog computing. Cloud-enabled ML deployment for real-time data processing and prediction has been addressed by Jiang et al. [20]. However, deploying ML models in the cloud introduces latency, which may be unsuitable for real-time applications. It requires high bandwidth, which may not be available in developing countries like South Africa. The constant need for connectivity may also hinder low-connectivity sites. Fog computing, an extension of the cloud but located closer to the edge, becomes a suitable solution. It promotes reduced latency and local data analysis, even using ML, making it suitable for real-time applications. Since aggregated data will be uploaded to

the cloud and not all the sensor data, reduced bandwidth will be used, even with increased security, as data will be anonymised [24, 26]. Fog needs no internet connection for local processing, hence it may suit poor bandwidth sites, including remote areas where cold rooms may be located or refrigerated transport may travel through. Aggregated data may be uploaded once strong connectivity becomes available. Fog becomes more necessary considering that IoT sensors generate big data, which requires more bandwidth and time to upload [24, 26, 30]. Moreover, the use of LSTM and GRU (LSGR) ensemble is novel for FFVs cold chain temperature prediction, given how it combines the strengths of these two types of Recurrent Neural Networks (RNNs): LSTM's capacity for complex long-term dependency modelling and GRU's computational efficiency and fast training [17, 20, 28, 32]. This work demonstrates how the integration of machine learning with fog computing and IoT creates a novel approach for FFV cold chain temperature prediction. Rather than focusing solely on algorithmic novelty, the primary contribution is a system-level validation showing how this integrated architecture mitigates network bottlenecks to enable robust, real-time prediction, a crucial insight for practical ML deployment in IoT systems. By facilitating real-time decision-making that reduces temperature break severity, this approach substantially minimises FFV wastage. The key contributions are:

- An investigation into the deployment of DNNs on fog nodes using IoT data,
- A comparison of temperature prediction performance between fog-based and cloud-based DNNs using R^2, MAE, and MSE,
- A comparison of processing times and data utilisation for both deployment approaches.

The remainder of this paper is structured as follows. Section 2 outlines the methodology, including data collection, pre-processing, deep learning architecture, and simulation setup. Section 3 presents the experimental results, comparing the performance of fog and cloud models. Section 4 concludes with key findings and future research directions.

2 Methodology

This section covers data collection and preprocessing; development of machine learning models, including architectural design, hyperparameter tuning, and evaluation using performance metrics.

2.1 Data Collection

Data was gathered from July 2024 to December 2024 from Apple's cold room laboratory in the Western Cape, South Africa, storing 40 boxes, each with 100 apples. The temperatures ranged from -0.2°C to 30.2°C with many simulated deviations. The Synetica enLink Zone Plus (eZone) captured data every 30 minutes. The sensor data was of interest for timestamps, temperature, pressure, humidity, CO_2 estimate equivalent, and ambient light. The sensor was initially

placed outside, on the door and then later inside, so this study did not use its temperature recordings, as they could lead to data leakage, especially when the sensor is inside the cold room. The Netvox_R718E (Netvox) sensor was also used, capturing timestamps, internal temperature, and acceleration and velocity in 3D, hourly. These parameters were used in previous studies [4, 6, 22, 28, 32]. Every 2 rows in the Netvox dataset, separated by 10 seconds, represented a sample, with the first row for accelerations and the second for velocities and cold room internal temperature. The sensors were connected to an iFemtoCell gateway using LoRaWAN, which uploaded data to the cloud.

2.2 Data Preprocessing

Data cleaning, splitting, scaling and sequencing were done as well as feature engineering.

Data Cleaning: The eZone dataset had 9010 rows after de-duplication. Netvox had 3852 rows after de-duplication and sample merging. After the sample merge, 0.42% of the data was missing, which was filled with forward filling [25, 28]. Data was standardised by removing units, converting percentages to decimals, and normalising number formats to integers and floats. Velocities were converted from mm/s to m/s to align with acceleration units (m/s^2). To align the data from the 2 sensors, the data was resampled hourly using the mean, resulting in 3967 rows, which were sorted in ascending order as required for time series analysis. This resulted in 3% and 0.25% missing sensor data for Netvox and eZone, respectively. The datasets were fused, with 2.08% missing after the process, filled using forward filling [25, 28]. Data from both sensors was merged for uniform splitting.

Data Splitting: After cleaning the data, it was split using Python's TimeSeriesSplit class to maintain temporal time-based ordering, suitable for time series analysis [10, 28]. To the class, 3 splits were passed, resulting in an initial 75% / 25% train/test data. This splitting strategy was selected after an optimisation process ranging from 2 to 20 splits, as it offered the best balance between supplying sufficient data for model training and preserving a robust test set to reliably assess generalisability and mitigate overfitting. The training data was used to train the model, while the test data was used to evaluate the cloud-based Deep Learning (cloudDL) and fog-based Deep Learning (fogDL) performances. Scaling and sequencing were done separately for training and test data to avoid data leakage.

Feature Engineering and Selection: Above measured parameters, additional features were created using rolling, lagging and interactions on the training and testing data separately, with due care to prevent data leakage. Based on Mutual

Information Scores shown in Fig. 1 [19, 28], which shows that past temperatures have a huge impact on current temperature, feature groups were created, and the best one for the model was selected. The optimum group was Temperature_CO2e_Interaction, Ambient_Light_Trend, Humidity, Ambient Light, Humidity_Temperature_Interaction, CO2e_Trend, CO2e_Lagged, Acceleration_Magnitude, Humidity_Lagged, CO2e Estimate Equivalent, Temperature_Lagged, Average_Humidity, Humidity_CO2e_Interaction. It was used to create features, with the cold room's internal temperature as the target. Creating new features on the training and test data introduced 0.01% and 0.02% missing data, respectively, filled with backfilling due to their small magnitude as well as positioning (top rows due to lagging) [28].

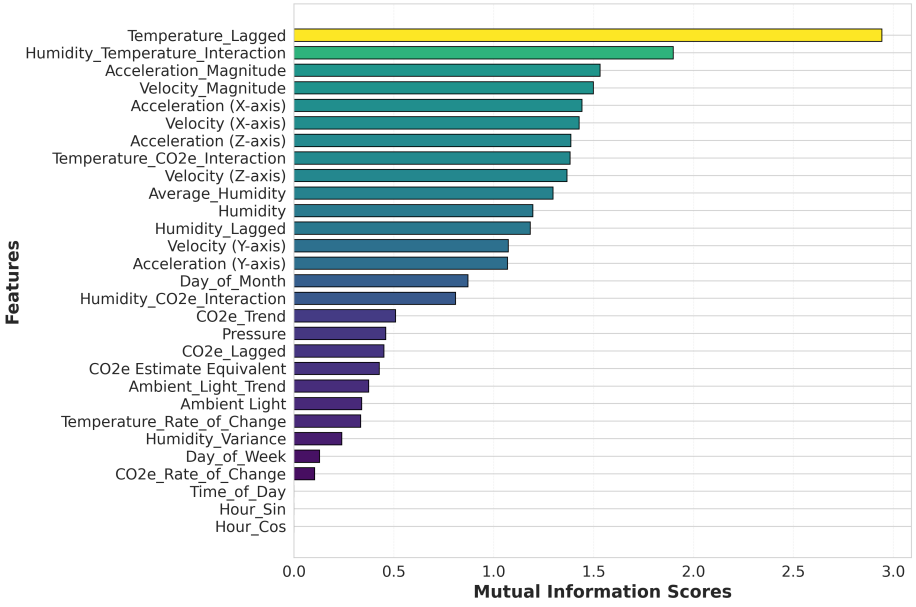

Fig. 1: MI Scores: Feature Importance vs. Temperature

Data Transformation: To speed up the training of the ML model, only the features (excluding the target, temperature) were normalised using MinMaxScaler, which scaled both training and test data to values between 0 to 1 [21, 28].

Data Sequencing: Since RNNs were used, the data was sequenced, with a window length optimised from 2 - 19 and 24 past hours. The optimum was 2, hence its usage in this study for training and testing data [20, 28].

2.3 The ML Model and its Hyperparameter Tuning

An ensemble model, combining the Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) layers, was developed. The LSTM is known for effective long-term dependency learning, sequential data proficiency, robustness to noise and inherent support for time series analysis, which makes it suitable for this study. The GRU is also preferred for long-term dependencies, simpler and quicker convergence, as well as computational efficiency. Both are types of recurrent neural networks which can handle vanishing gradient challenges during back-propagation as well as modelling complex and non-linear relationships [17, 20, 28, 32]. The combination thus brings the benefits of both RNNs, creating a quick training and long-term dependency learning model. The order of the LSTM and GRU layers was optimised using trial and error. Hyperparameter (units, activation functions, drop-out, use of batch normalisation, optimisers, learning rate, patience and batch size) optimisation was performed using random search, evaluating 100 randomly sampled parameter combinations to identify the best-performing model configuration. Early stopping was used to avoid overfitting [2, 28].

DNN configuration: 128-unit LSTM, Batch Normalisation, 64-unit GRU, 0.4 dropout and output layers. Both LSTM and GRU layers were activated by the hyperbolic tangent and reactivated by the sigmoid functions. The RMSProp was the cost function.

2.4 Performance Metrics

For cloudDL and fogDL predictions, regression metrics were used. Pipeline timing and data utilisation tracking were done for both fog and cloud ML simulated deployments. The simulators were run 10 times, for each iteration, Pearson's R was used to determine the relationship between the iteration's prediction against the actual values. For each metric (coefficient of determination (R^2), Mean Absolute Error (MAE) and Mean Squared Error (MSE), and Processing Time), results from the iterations were compared between the two ML deployments using a paired t-test and a Wilcoxon signed-rank test. Confidence intervals (CI) for the differences were computed at 95% confidence. The statistical tests and CI were implemented using Python's scipy.stats package.

Model's Regression Metrics: R^2, MAE and MSE were used for both cloudDL and fogDL predictions. The nearer to 1 the (R^2), the better, whereas the nearer to 0 the MAE and MSE, the better [14, 28].

Processing Time: Processing time was measured using Python's SimPy simulation clock to evaluate real-time performance constraints in fog and cloud computing environments. Timing captured the complete pipeline from sensor data transmission to model prediction, as this end-to-end measurement reflects actual

system responsiveness. The average processing time, T_{avg}, was calculated as the arithmetic mean of all individual cycle times to account for system variability, as is represented by

$$T_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} T_{\text{processing,i}} \tag{1}$$

where N is the total number of predictions; $T_{\text{processing,i}}$ is the processing time for the i^{th} pipeline, which is defined by

$$T_{\text{processing,i}} = t_{\text{end,i}} - t_{\text{start,i}} \tag{2}$$

where $t_{\text{start,i}}$ and $t_{\text{end,i}}$ are the timestamps recorded before data transmission and after prediction, respectively.

Data Utilization: This metric measures the fraction of incoming sensor data that results in a prediction and is defined by

$$\text{Data Utilization} = \frac{P_{\text{made}}}{\lfloor D_{\text{arrivals}}/|S| \rfloor} \tag{3}$$

where P_{made} is the number of predictions made; D_{arrivals} is the total number of data arrivals; $|S|$ is the number of sensors and $\lfloor \cdot \rfloor$ denotes integer division. The data utilisation equation 3 shows how efficiently the fog node turns raw measurements into predictions. This ensures the utilisation ratio reflects only full synchronisation opportunities, providing a realistic gauge of the node's data-to-prediction throughput, guiding decisions on compute allocation and network bandwidth.

2.5 Fog-Based and Cloud-Based Deployment Framework and Simulation (SimPy)

Fog computing is an extension of cloud computing, placed closer to the data source, with storage, computing and networking capabilities[23, 24, 26, 30]. Python's SimPy package was used to create a Fog Computing Cold Chain Simulator whose architecture is shown in Fig. 2, as well as a cloud computing simulator.

Simulation provides a cheap yet realistic platform (sandbox) for testing fog-based and cloud-based applications to identify and solve emerging issues before real-world deployment [18, 29]. SimPy is an event-driven simulation library which provides tools for modelling processes as generators that yield events, including algorithms that generate or respond to events such as transmitting sensor data. It is highly scalable and modular (flexible). A timeout event can be yielded to control the simulation timeline, while shared resources like the Central Processing Unit (CPU) are modelled as Resource objects. When a process yields a resource request, it suspends execution until the resource becomes available. If it yields to a timeout, it suspends until the waiting time is complete [29].

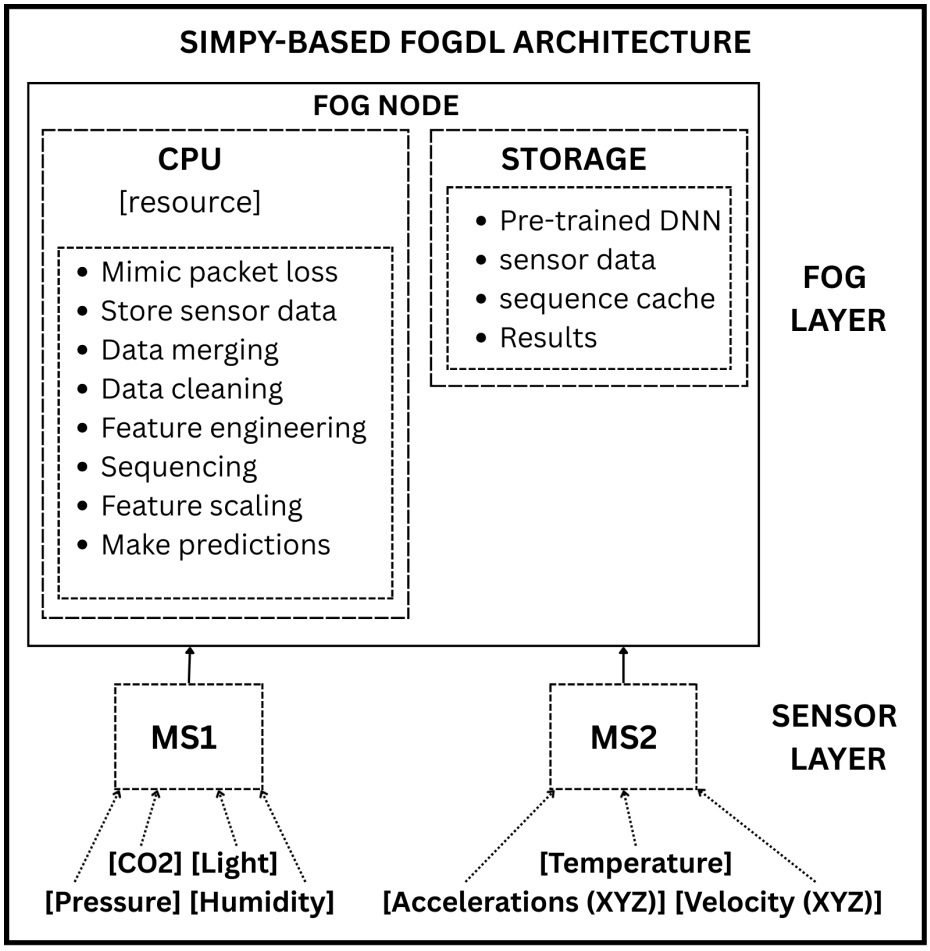

Fig. 2: Simpy-based FogDL Architecture Diagram

SimPy has been used to study Cloud-Fog RAN (CF-RAN) architectures in 5G networks, thus simulating both cloud and fog computing, running Integer Linear Program and graph-based heuristics to allocate resources for large networks for cities (Brazilian city) [29]. In another study, SimPy was used for simulation to optimise task scheduling between cloud and fog nodes using reinforcement learning, against baselines (LSTM, Deep Q-learning Network) using real-world Google Cloud Jobs data [9]. These deployments show the applicability of SimPy, hence its usage in this study.

For this study, the fog and cloud simulators had a single-core and 4-core CPU, respectively, modelled as finite resources to emulate task queuing and processing incoming data. The experiment covered a single cold room with 2 sensors (multi-sensors since each measured many parameters) in close proximity, so a single

fog node would be sufficient for fog computing. The fog node and cloud server stored the pre-trained LSGR model, results (including the actual temperature), 6 past readings per multi-sensor, as up to 5 past readings were used for feature engineering, a sequence cache and performance data as shown in Fig. 2.

The model would only predict cold room temperature when sufficient data is available, that is, for creating sequences (2 readings) as discussed in Section 2.2. For every timestamp, both sensors transmitted feature testing data, which was merged on the timestamp, with additional features created and scaled. For 10 times, the simulator was run for the duration of the length of the testing data, that is, 991 times.

The only difference with fog simulation is that for cloud, a packet loss of 30%, bit smaller than 40% experienced in another study, and 15% for fog, greater than the minimum achieved latency in the same study, were used in this study [8]. This was done so that fog experiences higher-than-perfect delays while the cloud experiences a double delay. The packet loss was implemented at a parameter sensor level or sensor, which was part of the multi-sensor. Losing data from a multi-sensor would mean losing data for all parameters which are measured by a multi-sensor. For conservative system-level testing, end-to-end latency values reported in [3], namely 9.30 ms (fog path) and 292.6 ms (cloud path), were used. These values include network transmission, processing and queuing delays and therefore act as conservative upper bounds on network delay; using them ensures the evaluation demonstrates correctness and timeliness even under heavy compute and network contention.

2.6 Hardware and Software

Training parameter and hyperparameter optimisation were carried out on the CHPC platform to optimise the model's performance. Cloud and Fog simulations were done on an Asus VivoBook Core i7 with 16 GB RAM and Ubuntu Linux. Visual Studio Code was used for programming, which was done in the Python language, mostly using Jupyter notebooks. LSGR was based on Keras. Matplotlib and Seaborn were used for visualisation, while Pandas managed data processing.

3 Results and Discussions

This section presents and discusses the deep learning prediction results.

3.1 Model Perfomance

Table 1 shows that fogDL has better temperature prediction performance compared to cloudDL, in terms of R^2(0.9163 vs 0.5734), MAE (1.3176 °C vs 4.4506 °C) and MSE (11.0384 vs 53.566).

The 95% CI for fogDL's R^2 are very high, ranging from 0.9081 to 0.9245, implying a good fit and generalisation. Thus, fogDL's predicted temperatures

Table 1: Temperature Prediction Metrics (Mean) and 95% CI: Fog vs. Cloud

Metrics	Fog	Cloud	Fog CI	Cloud CI
R^2	**0.9163**	0.5734	**[0.9081, 0.9245]**	[0.5477, 0.5992]
MAE	**1.3176**	4.4506	**[0.9999, 1.6353]**	[4.2983, 4.6029]
MSE	**11.0384**	53.5660	**[10.0084, 12.0685]**	[50.6492, 56.4827]
Pearson R	**0.95754**	0.78	**[0.9533, 0.9618]**	[0.7670, 0.7930]

generally follow the trend of the actual temperatures. This is not the case for cloudDL, whose values range from 0.5477 to 0.5992, implying a struggle with explaining the temperature data. By looking at MAE, fogDL ranges from 0.9999 °C to 1.6353 °C, while cloudDL ranges from 4.2983 °C to 4.6029 °C. fogDL predictions are thus off by 1.3^0C, with a smaller maximum deviation, compared to cloudDL's 4.5^0C, with a higher maximum deviation, on average. For MSE, fogDL ranges from 10.3633 to 12.0640, whereas cloudDL ranges from 50.6492 to 56.4827. This implies that cloudDL's predictions are off by a huge margin compared to fogDL. These results show that fogDL is more accurate than cloudDL. The good fit and lower prediction errors for fogDL are supported by a strong positive correlation between the actual and predicted temperatures, with a Pearson correlation coefficient of 0.95754 (95% CI: 0.9533 - 0.9618). This is notably higher than the correlation of 0.78 (95% CI: 0.7670 - 0.7930) observed for cloudDL. The correlations indicate a huge accuracy for fogDL, considering the 15% packet loss that was introduced, hence resilient, data loss-tolerant and robust. This is crucial for the cold chain of FFVs, implying that fog computing does improve the model's temperature prediction performance and is suitable for real-world deployment.

Table 2 shows the differences between the fogDL and cloudDL based on the T-stat and W-stat tests. T-stat and W-stat values in Table 2 indicate that: (i) for R^2, there is an extremely strong, highly significant difference between fogDL and cloudDL, with fogDL always getting higher values; (ii) for MAE and MSE, it can be argued that fogDL significantly does better than MSE, and it's not by chance, and (iii) fogDL's predictions are substantially significantly correlated to the actual temperature values compared to cloudDL.

Table 2: Temperature Prediction: T-Test and W-Test Statistics (Fog vs. Cloud)

Metrics	T-stat	T-stat p-val	W-stat	W-stat p-val
R^2	31.55	0.0	0.0	0.002
MAE	-28.77	0.0	0.0	0.002
MSE	-35.31	0.0	0.0	0.002
Pearson R	33.27	0.0	0.0	0.002

The architectural advantage of the fogDL model in reducing packet loss is the direct cause of its higher accuracy. The local processing of the fog model main-

tains the integrity of the data, whereas the cloud model loses more data due to lengthy transmission, which destroys the temporal sequences that are essential for prediction. Their performance difference is easily explained by the cloud architecture's 30% packet loss rate compared to the fog's 15%. Because of the fog paradigm's innate resistance to network deterioration by being located close to the data source, this illustrates that the performance advantage, MSE included (11.03 vs 53.57), is infrastructural rather than algorithmic.

3.2 Processing Time and System Performance

Table 3 shows fogDL achieves a lower average processing time per complete pipeline (84.31 milliseconds (ms), with a 95% CI of 84 to 84.7 ms) compared to cloudDL (337.65 ms, with a 95% CI of 337.5 to 337.8 ms). The average processing time and CIs show that fogDL has a better response time.

Table 3: Average Processing Time (APT) and 95% CI: Fog vs. Cloud

Fog APT	**Cloud APT**	**Fog APT CI**	**Fog APT CI**
84.31	337.65	**[84, 84.7]**	[337.5, 337.8]

The small time it takes from the moment data is transmitted until a prediction is made using fog computing is reasonable since it takes less than a second for the system to complete the data pipeline, which is almost instantaneous, thus without lag. This implies that many transmit-to-predict pipelines per second can be made, which may not be needed since the cold room's temperature changes slowly. This is essential in real-time systems where the stakeholders don't have to wait many minutes for a model's prediction, in some cases where the prediction happens after the predicted event has already occurred, resulting in control lags, and that defeats the essence of real-time prediction and proactive control. This implies that the single fog node is sufficient, the model is lightweight, and the computing power required is affordable. For the average processing time, the fogDL to cloudDL t-stat is -1410.35, and the W-stat is 0, both with p-values of 0, less than 0.05, confirming that fogDL has a significantly lower processing time than cloudDL. This result indicates that fogDL achieves lower response times than cloudDL, making fog a more suitable deployment option for real-time systems such as FFVs' cold chains. This finding aligns with previous studies [24, 26] and with evidence from other domains, including workflow scheduling [27]. The smaller processing time for fog deployment is due to the smaller distance between the data source, that is, sensors, and the fog server, while in cloud deployment, the sensor data is transmitted over long distances, introducing significant network latency, particularly in the case of simulated network congestion. This highlights the benefits of fog computing for real-time temperature prediction.

As shown in Figure 3, each sensor sent data 991 times, and the models deployed on fog and the cloud did 989 predictions, resulting in 99.8% data utilisation. Only two transmissions had no predictions associated with them. This

is mainly due to insufficient data required to create a sequence of 2 readings. These metrics imply a huge conversion rate for the fog-based and cloud-based ML framework, providing an assurance that if data is transmitted from sensors, it will surely be analysed.

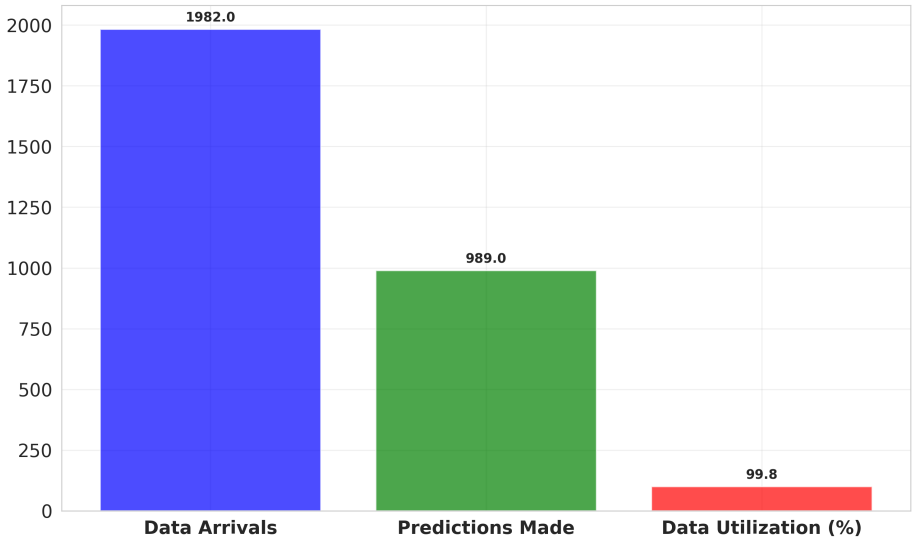

Fig. 3: Cloud vs. Fog Performance: System Performance

4 Conclusion

This study evaluated the deployment of a DNN on the fog for IoT data, comparing its predictive performance to a cloud-based approach in the context of cold chain monitoring. The findings demonstrated that fogDL consistently outperformed cloudDL, achieving substantially lower errors (MAE: 1.3°C vs. 4.5°C; MSE: 11 vs. 53.5) and higher accuracy (R^2: 0.9163 vs. 0.5734). Importantly, even with 15% data loss during transmission, predictions remained within a margin of 1.3°C, showing resilience to imperfect network conditions. In addition, processing time was significantly reduced (84.3 ms vs. 337.7 ms), confirming that fog computing provides the responsiveness needed for real-time cold chain systems. Both deployments maintained a high prediction success rate of 99.8%, ensuring that almost every transmitted data point was reliably processed. Taken together, these results indicate that fog-based deployment not only improves predictive accuracy and efficiency but also empowers stakeholders to act proactively, controlling cold chain parameters before deviations occur. This ability to anticipate rather than react minimises the risk of temperature abuse, reduces food and

fresh vegetable wastage, lowers operational costs, and mitigates environmental impacts associated with supply chain inefficiencies.

Although this study highlights the potential of fog computing to reduce latency, processing time and packet loss while improving prediction accuracy, its generalisability is limited by reliance on simulation-based abstraction and a generic network model, both of which affect latency and reliability. Future work will address these limitations by incorporating real-world failures, such as sensor breakdowns or multi-sensor outages, through sensor redundancy, as well as by modelling networking protocols such as 3G, 4G, and 5G. Scalability and resource constraints also limit the study, as the setup used a single fog node without considering its power or memory requirements, two sensors, one DNN model and a dataset to provide a clear baseline comparison. While suitable for small-scale scenarios, this design limits scalability. In larger deployments with many sensors, the computational and network load on a single fog node could become a bottleneck, reducing its latency advantage over the cloud. Thus, the results apply mainly to small-scale use cases and require further validation by exploring multi-node collaborative processing, more sensors, alternative models, and more diverse datasets and hardware scenarios. Beyond prediction, systems could also provide actionable insights by indicating whether conditions are within acceptable ranges and identifying the cause of deviations. Finally, extending the framework to asynchronous sensor fusion under packet loss remains an important direction for advancing robustness and scalability. Overall, the results contribute to the growing evidence that fog computing can serve as a practical, scalable, and generalizable paradigm for real-time IoT applications well beyond cold chain monitoring, including healthcare, smart cities, and industrial automation.

Acknowledgments. This work is based on the research supported in part by the National Research Foundation (NRF) of South Africa (Ref Numbers SRUG22051310444, PMDS230703126166). The opinions, findings and conclusions or recommendations expressed in any publication generated by the NRF-supported research are those of the author(s) alone, and the NRF accepts no liability whatsoever in this regard.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Ali, H., Rafaqat, K., Teg, A., Rab Nawaz, B., Haitham, N., Amjad Rehman, K., Aqsa: IoT- Enabled Firmness Grades of Tomato in Cold Supply Chain Using Fusion of Whale Optimization Algorithm and Extreme Learning Machine. IEEE access **12**, 52744–52758 (2024). https://doi.org/10.1109/ACCESS.2024.3379327

[2] Anggoro, D., Mukti, S.: Performance Comparison of Grid Search and Random Search Methods for Hyperparameter Tuning in Extreme Gradient Boosting Algorithm to Predict Chronic Kidney Failure. International Journal of Intelligent Engineering and Systems **14**(6), 198–207 (2021). https://doi.org/10.22266/ijies2021.1231.19

[3] Awaisi, K.S., Abbas, A., Zareei, M., Khattak, H.A., Shahid Khan, M.U., Ali, M., Ud Din, I., Shah, S.: Towards a Fog Enabled Efficient Car Parking Architecture. IEEE Access **7**, 159100–159111 (2019). https://doi.org/10.1109/ACCESS.2019.2950950

[4] Ayanoglu, M., Uysal, I.: ML Approach to Improve the Costs and Reliability of a Wireless Sensor Network. Sensors **23**(9) (2023). https://doi.org/10.3390/s23094303

[5] Badia-Melis, R., Mc Carthy, U., Ruiz-Garcia, L., Garcia-Hierro, J., Robla Villalba, J.I.: New trends in cold chain monitoring applications - A review. Food Control **86**, 170–182 (2018). https://doi.org/10.1016/j.foodcont.2017.11.022

[6] Badia-Melis, R., Qian, J.P., Fan, B.L., Hoyos-Echevarria, P., Ruiz-García, L., Yang, X.T.: Artificial Neural Networks and Thermal Image for Temperature Prediction in Apples. Food and Bioprocess Technology **9**(7), 1089–1099 (2016). https://doi.org/10.1007/s11947-016-1700-7

[7] Bai, L., Liu, M., Sun, Y.: Overview of Food Preservation and Traceability Technology in the Smart Cold Chain System. Foods **12**(15) (2023). https://doi.org/10.3390/foods12152881

[8] Boukhali, Y., Kabbaj, M.N., Benbrahim, M.: Fog-IoPM: Fog computing for Internet of Plants data management. Scientific African **28**, e02682 (2025). https://doi.org/10.1016/j.sciaf.2025.e02682

[9] Choppara, P., Mangalampalli, S.S.: Reliability and Trust Aware Task Scheduler for Cloud-Fog Computing Using Advantage Actor Critic (A2C) Algorithm. IEEE Access **12**, 102126–102145 (2024). https://doi.org/10.1109/ACCESS.2024.3432642

[10] Duarte, M.S., Martins, G., Oliveira, P., Fernandes, B., Ferreira, E.C., Alves, M.M., Lopes, F., Pereira, M.A., Novais, P.: A Review of Computational Modeling in Wastewater Treatment Processes. ACS EST Water **4**(3), 784–804 (2024). https://doi.org/10.1021/acsestwater.3c00117

[11] FAO: The State of Food and Agriculture 2019. Moving forward on food loss and waste reduction. The State of the World, Food and Agriculture Organization of the United Nations, Rome

(2019), https://openknowledge.fao.org/server/api/core/bitstreams/11f9288f-dc78-4171-8d02-92235b8d7dc7/content, accessed: 2024-02-08

[12] Findlay, J.S., Combrink, J.C.: South African controlled atmosphere storage operator's manual. Hortgro Pome, Paarl, South Africa (2013)

[13] Fruit South Africa: Key fruit statistics 2021/22 (2022), https://fruitsa.co.za/wp-content/uploads/2023/09/A5-Fruit-SA-Booklet_2023_Digital.pdf, accessed: 2024-02-08

[14] Gideon, K., Nyirenda, C., Temaneh-Nyah, C.: Echo state network-based radio signal strength prediction for wireless communication in Northern Namibia. IET Communications **11**(12), 1920–1926 (2017). https://doi.org/10.1049/iet-com.2016.1290

[15] Goedhals-Gerber, L.L., Khumalo, G.: Identifying temperature breaks in the export cold chain of navel oranges: A Western Cape case. Food Control **110**, 107013 (2020). https://doi.org/https://doi.org/10.1016/j.foodcont.2019.107013

[16] Goedhals-Gerber, L.L., van Dyk, E., Getor, R.Y., Louw, B., Mishra, N.: Identifying container hotspots for table grape exports from South Africa to the UK: A case study. Transportation Research Interdisciplinary Perspectives **24** (2024). https://doi.org/10.1016/j.trip.2024.101054

[17] Guo, J., Liu, D., Lin, S., Lin, J., Zhen, W.: Temperature Prediction of a Temperature-Controlled Container with Cold Energy Storage System Based on Long Short-Term Memory Neural Network. Applied Sciences (Switzerland) **14**(2) (2024). https://doi.org/10.3390/app14020854

[18] Gupta, P., Sharma, R., Gupta, S.: Simulators for Fog Computing and Information Processing. PROCEEDINGS OF THE NATIONAL ACADEMY OF SCIENCES INDIA SECTION A-PHYSICAL SCIENCES **94**(4), 437–447 (2024). https://doi.org/10.1007/s40010-024-00891-x

[19] Huang, L., Zhou, X., Shi, L., Gong, L.: Time Series Feature Selection Method Based on Mutual Information. Applied Sciences **14**(5) (2024). https://doi.org/10.3390/app14051960

[20] Jiang, J., Peng, C., Liu, W., Liu, S., Luo, Z., Chen, N.: Environmental Prediction in Cold Chain Transportation of Agricultural Products Based on K-Means++ and LSTM Neural Network. Processes **11**(3) (2023). https://doi.org/10.3390/pr11030776

[21] Kowsher, M., Islam Sanjid, M.Z., Das, A., Ahmed, M., Hossain Sarker, M.M.: Machine Learning and Deep Learning based Information Extraction from Bangla Names. Procedia Computer Science **178**, 224–233 (2020). https://doi.org/10.1016/j.procs.2020.11.024

[22] Loisel, J., Cornuéjols, A., Laguerre, O., Tardet, M., Cagnon, D., Duchesne de Lamotte, O., Duret, S.: Machine learning for temperature prediction in food pallet along a cold chain: Comparison between synthetic and experimental training dataset. Journal of food engineering **335**, 111156 (2022). https://doi.org/10.1016/j.jfoodeng.2022.111156

[23] Ma, K., Bagula, A., Nyirenda, C., Ajayi, O.: An IoT-Based Fog Computing Model. Sensors **19**(12) (2019). https://doi.org/10.3390/s19122783

[24] Musa, Z., Vidyasankar, K.: A Fog Computing Framework for Blackberry Supply Chain Management. vol. 113, pp. 178–185 (2017). https://doi.org/10.1016/j.procs.2017.08.338

[25] Oluwaseye, J.L., Doorsamy, W., Sena, P.B.: A Review of Missing Data Handling Techniques for Machine Learning. IJITIS **5**(3), 971–1005 (2022). https://doi.org/10.15157/IJITIS.2022.5.3.971-1005

[26] Ribeiro Junior, F.M., Bianchi, R.A.C., Prati, R.C., Kolehmainen, K., Soininen, J.P., Kamienski, C.A.: Data reduction based on machine learning algorithms for fog computing in IoT smart agriculture. Biosystems engineering **223**, 142–158 (2022). https://doi.org/10.1016/j.biosystemseng.2021.12.021

[27] Subramoney D., Nyirenda C. N.: Multi-Swarm PSO Algorithm for Static Workflow Scheduling in Cloud-Fog Environments. IEEE Access **10**, 117199–117214 (2022). https://doi.org/10.1109/ACCESS.2022.3220239

[28] Taguta, J., Isingizwe Nturambirwe, J.F., Nyirenda, C.N.: Comparative Evaluation of Machine Learning Models for Predicting Fresh Produce Cold Chain Temperature: A Case of South African Apples. In: 2025 IST-Africa Conference (IST-Africa). pp. 1–10 (2025). https://doi.org/10.23919/IST-Africa67297.2025.11060484

[29] Tinini, R.I., Santos, M.R.P.d., Figueiredo, G.B., Batista, D.M.: 5GPy: A SimPy-based simulator for performance evaluations in 5G hybrid Cloud-Fog RAN architectures. Simulation Modelling Practice and Theory **101**, 102030 (2020). https://doi.org/10.1016/j.simpat.2019.102030

[30] Tzounis, A., Katsoulas, N., Bartzanas, T., Kittas, C.: Internet of Things in agriculture, recent advances and future challenges. Biosystems Engineering **164**, 31–48 (2017). https://doi.org/10.1016/j.biosystemseng.2017.09.007

[31] WWF South Africa: Food loss and waste: Facts and futures (2017), www.wwf.org.za/food_loss_and_waste_facts_and_futures, accessed: 2024-02-08

[32] Zou, Y., Wu, J., Wang, X., Morales, K., Liu, G., Manzardo, A.: An improved artificial neural network using multi-source data to estimate food temperature during multi-temperature delivery. Journal of Food Engineering **351** (2023). https://doi.org/10.1016/j.jfoodeng.2023.111518