JTIM : Jurnal Teknologi Informasi dan Multimedia

p-ISSN : 2715-2529
e-ISSN : 2684-9151
https://journal.sekawan-org.id/index.php/jtim

Prediksi Beban Kerja Server Secara *Real-Time* pada Pusat Data *Cloud* dengan Pendekatan Gabungan *Long Short-Term Memory (LSTM)* dan *Fuzzy Logic*

Naufal Hanif [1,*], Dadang Priyanto [1] dan Neny Sulistianingsih [1]

[1] Magister Ilmu Komputer, Universitas Bumigora, Indonesia

* Korespondensi: naufal@universitasbumigora.ac.id

Sitasi: Hanif, N.; Priyanto, D.; Sulistianingsih, N. (**2025**). Prediksi Beban Kerja Server Secara *Real-Time* pada Pusat Data *Cloud* dengan Pendekatan Gabungan *Long Short-Term Memory* (*LSTM*) dan *Fuzzy Logic*. JTIM: Jurnal Teknologi Informasi Dan Multimedia, 7(3), 420-432. https://doi.org/10.35746/jtim.v7i3.731

Diterima: 26-04-2025
Direvisi: 05-06-2025
Disetujui: 11-06-2025

Abstract: Efficient resource management in Cloud Data Centers is essential to reduce energy waste and maintain optimal system performance. This study aims to predict server workload in real time using a hybrid approach that combines Long Short-Term Memory (LSTM) and Fuzzy Logic. CPU and RAM usage data were collected every second from a Proxmox Cluster using its API, then normalized and processed using an LSTM model to forecast future workloads. The predicted results were then classified using Fuzzy Logic into three workload categories: light, medium, and heavy. The model was evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE), where the results showed an MAE of 2.48 on the training data and 3.09 on the testing data, as well as RMSE values of 5.15 and 5.57, respectively. Based on these evaluation results, the prediction system achieved an accuracy of 97.52% on the training data and 96.91% on the testing data, indicating that the model can generate accurate and stable predictions. This method enables automated decision-making such as workload-based power management, thereby improving energy efficiency and overall system performance.

Keywords: Cloud Data Center; Energy Efficiency; Fuzzy Logic; LSTM; Workload Prediction.

Abstrak: Pengelolaan sumber daya yang efisien pada *Cloud Data Center* sangat penting untuk mengurangi pemborosan energi dan memastikan performa sistem tetap optimal. Penelitian ini bertujuan untuk memprediksi beban kerja server secara *real-time* menggunakan pendekatan gabungan *Long Short-Term Memory* (*LSTM*) dan Logika *Fuzzy*. Data penggunaan *CPU* dan *RAM* dikumpulkan setiap detik dari *Cluster Proxmox* menggunakan *API*, kemudian dinormalisasi dan diolah menggunakan model *LSTM* untuk memprediksi beban kerja di waktu mendatang. Hasil prediksi selanjutnya diklasifikasikan menggunakan Logika *Fuzzy* menjadi kategori beban kerja ringan, sedang, dan berat. Evaluasi model dilakukan dengan metrik *Mean Absolute Error* (*MAE*) dan *Root Mean Squared Error* (*RMSE*), di mana hasil menunjukkan nilai *MAE* sebesar 2.48 pada data pelatihan dan 3.09 pada data pengujian, serta *RMSE* sebesar 5.15 dan 5.57. Berdasarkan hasil evaluasi yang dilakukan, sistem prediksi memiliki tingkat keberhasilan mencapai 96,91% pada data pengujian dan 97,52% pada data pelatihan, yang menunjukkan bahwa model mampu menghasilkan prediksi yang akurat dan stabil. Kombinasi metode ini memungkinkan pengambilan keputusan otomatis seperti manajemen daya berbasis beban kerja, sehingga meningkatkan efisiensi energi dan kinerja sistem secara keseluruhan.

Kata kunci: Pusat Data *Cloud*; Efisiensi Energi; Logika *Fuzzy*; *LSTM*; Prediksi Beban Kerja.

1. Pendahuluan

Pusat data awan memiliki peran penting dalam menyediakan layanan komputasi awan yang mendukung berbagai sektor, seperti industri dan bisnis elektronik[1]. Menurut [2] dalam [3] *VM* (*Virtual Machine*) mencakup seluruh komponen yang dibutuhkan untuk menjalankan aplikasi, seperti komputasi, penyimpanan, dan lainnya. Baik di mesin virtual atau kontainer Docker, *Cloud Engineer* harus mengalokasikan sumber daya yang cukup untuk aplikasi agar berjalan lancar, namun dalam kebanyakan kasus, aplikasi ini tidak berjalan pada beban terberat, sumber daya yang disediakan sebelumnya menganggur pada sebagian besar waktu, ini menyebabkan pemborosan sumber daya. Dilaporkan bahwa tinggi konsumsi energi disebabkan oleh penggunaan sumber daya komputasi yang tidak efisien di pusat data, yaitu sekitar 93% dari energi yang dikonsumsi adalah oleh sumber daya komputasi awan[4]. Selain itu, ketika beban kerja berat dan harus bersaing dengan aplikasi lain secara bersamaan, sumber daya yang dialokasikan sebelumnya mungkin tidak cukup. Untuk mengatasi masalah tersebut, algoritma prediksi khusus biasanya digunakan di komputasi awan untuk memprediksi kebutuhan sumber daya[5]. Menurut [6] perkiraan beban kerja sangat penting dalam pengelolaan sumber daya di pusat data. Pada pusat data yang heterogen, tantangan untuk memprediksi dengan akurat menjadi lebih besar karena adanya variasi beban kerja.

Prediksi beban kerja memiliki peran penting dalam mendukung penskalaan layanan secara efisien dan mengoptimalkan pemanfaatan sumber daya pusat data awan[7]. Menurut [5] Penyedia layanan *cloud* menggunakan prediktif analisis untuk menghindari berbagai jenis kerugian seperti ketidaksediaan layanan, konsumsi energi maksimum, dan kehilangan pelanggan. Salah satu metode untuk melakukan analisis prediktif menggunakan data deret waktu adalah *Long Short-Term Memory* (*LSTM*). *Long Short-Term Memory* (*LSTM*) adalah jenis jaringan saraf tiruan yang dirancang khusus untuk menangani dan memproses data sekuensial, sehingga sangat cocok digunakan dalam analisis deret waktu[8]. Menurut [9] *LSTM* adalah jenis jaringan saraf yang memiliki kemampuan untuk menyimpan informasi, sehingga cocok digunakan untuk memproses dan memprediksi kejadian penting dengan interval yang cukup panjang dan adanya penundaan dalam data deret waktu. Penelitian yang dilakukan oleh [10] menyajikan analisis prediktif perkiraan deret waktu menggunakan pembelajaran mendalam metode *LSTM* untuk memprediksi beban server di masa depan. Akurasi prediksi *LSTM* telah diukur menggunakan tiga metrik *RMSE*, *MSE*, dan *MAE*, hasilnya menunjukan bahwa model *LSTM* mencapai *MAE* 0,043, *RMSE* 0,075, dan *MSE* 0,066.

Menurut [11] dalam [12] Model *ANFIS* mengintegrasikan dua sistem kecerdasan, yaitu jaringan saraf (*NN*) dan sistem inferensi *fuzzy* (*FIS*). Dalam model ini, algoritma pembelajaran *NN* digunakan untuk mengatur parameter pada *FIS*. *NN* berfungsi sebagai alat pemodelan data non linier yang mampu mengidentifikasi dan mempelajari hubungan antara *input* dan *output*, serta mendeteksi pola-pola kompleks dalam data, sedangkan *FIS* adalah metode yang memanfaatkan logika *fuzzy* untuk memetakan *input* menjadi *output*. Pemetaan ini menjadi dasar dalam pengambilan keputusan atau pengenalan pola.

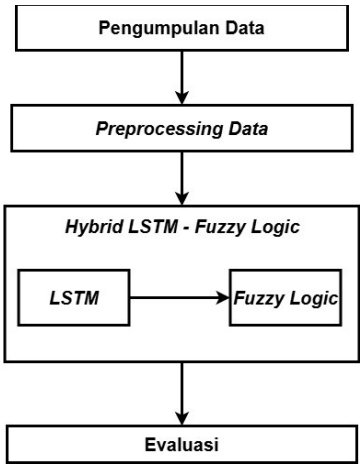
Untuk mengatasi masalah tersebut, penulis merekomendasikan gabungan metode prediksi beban kerja yaitu *Long Sort-Term Memory* (*LSTM*) dan *Fuzzy Logic*, dimana hasil dari prediksi ini nantinya akan dijadikan kebijakan untuk melakukan penjadwalan *power management* agar penggunaan sumber daya server pada *cloud data center* menjadi lebih efisien dan dapat menghemat biaya listrik dan dampak lingkungan yang ditimbulkan oleh *cloud data center*.

2. Penelitian Terdahulu

Penelitian terkait prediksi beban kerja di server *cloud* telah banyak dilakukan. [10] menggunakan data *log* layanan web dari Universitas Complutense Madrid untuk memprediksi beban kerja di server *cloud* menggunakan model *Long Short-Term Memory* (*LSTM*), menghasilkan *MAE* 0,043, *RMSE* 0,075, dan *MSE* 0,066. Penelitian lain oleh [13] menggunakan data lalu lintas *Internet Protocol* dari *Cisco Systems* untuk memprediksi kebutuhan energi pusat data di masa depan, dengan hasil prediksi kebutuhan listrik meningkat dari 292 TWh pada 2016 menjadi 353 TWh pada 2030. Penelitian yang dilakukan oleh [6] mengembangkan metode *Three-Way Ensemble Prediction* (*TWD-RCPM*) yang menunjukkan pengurangan kesalahan prediksi relatif kumulatif hingga 72,63% dibandingkan model *Neural Network* (*NN*). Penelitian yang dilakukan oleh [5] mengusulkan model prediksi hibrida berbasis *ARIMA* dan *Triple Exponential Smoothing* untuk memprediksi penggunaan sumber daya dalam kontainer Docker, meningkatkan akurasi prediksi hingga 203,72% dibandingkan model sebelumnya dengan *overhead* waktu minimal. Berdasarkan beberapa penelitian terdahulu yang telah dipaparkan, dapat disimpulkan bahwa prediksi beban kerja pada lingkungan *Cloud Data Center* memiliki peran yang signifikan dalam meningkatkan efisiensi pengelolaan sumber daya. Penelitian yang dilakukan oleh [10] menunjukkan bahwa metode *LSTM* mampu memberikan akurasi tinggi dalam memprediksi beban kerja dengan nilai *MAE* 0,043, sedangkan penelitian yang dilakukan oleh [6] membuktikan bahwa pendekatan *Three-Way Ensemble Prediction* mampu mengurangi kesalahan prediksi hingga lebih dari 69% dibandingkan model lainnya. Selain itu, pendekatan hibrida seperti *ARIMA* dan *Triple Exponential Smoothing* yang diusulkan oleh [5] berhasil meningkatkan akurasi prediksi sumber daya kontainer secara signifikan. Namun, sebagian besar studi sebelumnya masih terbatas pada akurasi prediksi tanpa integrasi ke sistem pengambilan keputusan otomatis. Penelitian ini mengusulkan pendekatan gabungan *LSTM* dan Logika *Fuzzy* untuk menghasilkan sistem yang adaptif dan efisien dalam memprediksi serta mengelola beban kerja *Cloud Data Center* secara *real-time*.

3. Metodologi Penelitian

Tahapan penelitian menjelaskan tahapan – tahapan yang dilakukan dalam penelitian ini, adapun tahapan pada penelitian ini dimulai dari pengumpulan data, preprocessing, pendekatan gabungan *LSTM* dan Logika *Fuzzy*, dan evaluasi. Alur pada penelitian ini dapat dilihat pada Gambar 1 sebagai berikut:



Gambar 1. Alur Penelitian

Gambar 1 menunjukkan alur penelitian yang terdiri dari empat tahapan utama, yaitu pengumpulan data *CPU* dan *RAM* dari *Cluster* Proxmox melalui *API*, *preprocessing* data (normalisasi dan pemisahan data), pemrosesan dengan model *hybrid LSTM–Fuzzy Logic*

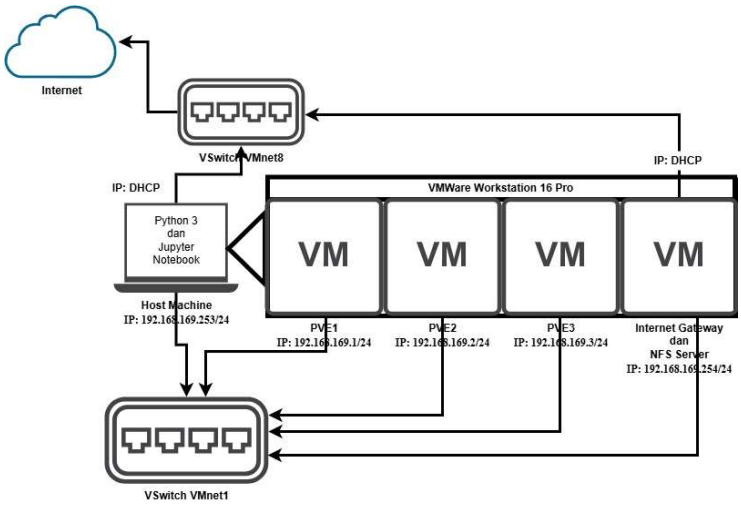
untuk prediksi dan klasifikasi beban kerja, serta evaluasi akurasi menggunakan metrik *MAE* dan *RMSE*.

3.1. *Pengumpulan Data*

Peneliti melakukan pengumpulan data penggunaan *CPU* dan *RAM* atau *Memory* pada *cluster Proxmox* secara langsung menggunakan *API Proxmox*. Data dikumpulkan setiap detik selama rentang waktu tertentu untuk mendapatkan pola beban kerja yang akurat. Pengumpulan data dilakukan dengan menggunakan skrip *Python* yang dijalankan pada *Jupyter Notebook* di sistem operasi *Windows 10* dengan memanfaatkan pustaka proxmoxer untuk mengakses *API Proxmox*. Skrip ini dirancang untuk membaca data penggunaan *CPU* dan *RAM* atau *Memory* dari semua *node server* dalam *cluster*.

Algoritma 1. Proses Mengambil Data *CPU* dan *RAM* pada *Cluster Proxmox Virtual Environment* Menggunakan *Proxmox API*
Masukan: *Proxmox API endpoint* (proxmox_host, proxmox_user, proxmox_password).
Keluaran: Persentase penggunaan *CPU* dan *RAM* total dalam *cluster*.
Proses:
1 Hubungkan ke *Proxmox API* menggunakan proxmox_host, proxmox_user, dan proxmox_password.
2 Ambil daftar *resource* dari *cluster* menggunakan proxmox.cluster.resources.get.
3 Inisialisasi:
  total_cpu_usage = 0
  total_cpu_capacity = 0
  total_ram_usage = 0
  total_ram_capacity = 0
4 For setiap resource dalam daftar resource:
5 If tipe resource adalah node:
6 Tambahkan penggunaan CPU node ke total_cpu_usage:
  total_cpu_usage += res['cpu'] * res['maxcpu'].
7 Tambahkan kapasitas CPU node ke total_cpu_capacity:
  total_cpu_capacity += res['maxcpu'].
8 Tambahkan penggunaan RAM node ke total_ram_usage:
  total_ram_usage += res['mem'].
9 Tambahkan kapasitas RAM node ke total_ram_capacity:
  total_ram_capacity += res['maxmem'].
10 End If
11 End For
12 Hitung persentase total penggunaan CPU:
  total_cpu_percentage = (total_cpu_usage / total_cpu_capacity) * 100.
13 Hitung persentase total penggunaan RAM:
  total_ram_percentage = (total_ram_usage / total_ram_capacity) * 100.
14 Kembalikan hasil:
  [total_cpu_percentage, total_ram_percentage].

Untuk mendukung implementasi sistem prediksi berbasis *LSTM* dan Logika *Fuzzy*, digunakan topologi virtual di VMware Workstation 16 Pro yang mensimulasikan *Cloud Data Center* skala kecil. Lingkungan ini terdiri dari beberapa *node Proxmox VE* dan satu *VM* Debian sebagai *gateway* internet dan *NFS Server*, yang terhubung melalui VMnet1 (jaringan internal) dan VMnet8 (internet). Rancangan ini memungkinkan proses pengumpulan data dan pengujian model dilakukan secara fleksibel dan terisolasi, seperti ditunjukkan pada Gambar 2.



Gambar 2. Rancangan Jaringan Pusat Data Awan *Proxmox Cluster*

Gambar 2 memperlihatkan bahwa *host machine* berperan sebagai pusat pengembangan sistem dengan Python 3 dan Jupyter Notebook sebagai lingkungan kerja. *Host* terhubung ke VMnet1 dan VMnet8, sedangkan tiga *VM* berfungsi sebagai *node Proxmox* (PVE1–PVE3) dan satu *VM* Debian sebagai *Internet Gateway* dan *NFS Server*. Komunikasi antar *node* berlangsung melalui VMnet1, sementara akses internet diperoleh melalui *gateway* atau *NAT*. Pengambilan data *CPU* dan *RAM* dilakukan melalui *API* dan diproses secara lokal menggunakan model *LSTM* dan Logika *Fuzzy*.

3.2. *Preprocessing Data*

Preprocessing merupakan tahap yang sangat penting dalam pengolahan data, karena tahap ini tidak hanya berperan dalam membersihkan dan mempersiapkan data mentah, tetapi juga secara signifikan dapat meningkatkan kinerja model dengan memastikan data yang digunakan lebih terstruktur, relevan, dan bebas dari anomali yang dapat mengganggu proses analisis atau pelatihan model[14]. Pada penelitian ini tahap *preprocessing* dilakukan mulai dari pengumpulan data, normalisasi data, pemisahan fitur dan label, *split data training* dan *testing*, *reshaping data* untuk *LSTM*, dan *inverse transform* hasil prediksi. Min-Max normalisasi adalah teknik normalisasi yang menerapkan transformasi linier pada data asli untuk memastikan keseimbangan perbandingan nilai antara sebelum dan sesudah proses normalisasi [16] dalam[17]. Metode ini dapat dihitung menggunakan rumus berikut:

$$X_{new} = \frac{X - min(X)}{\max(X) - \min(X)} \quad (1)$$

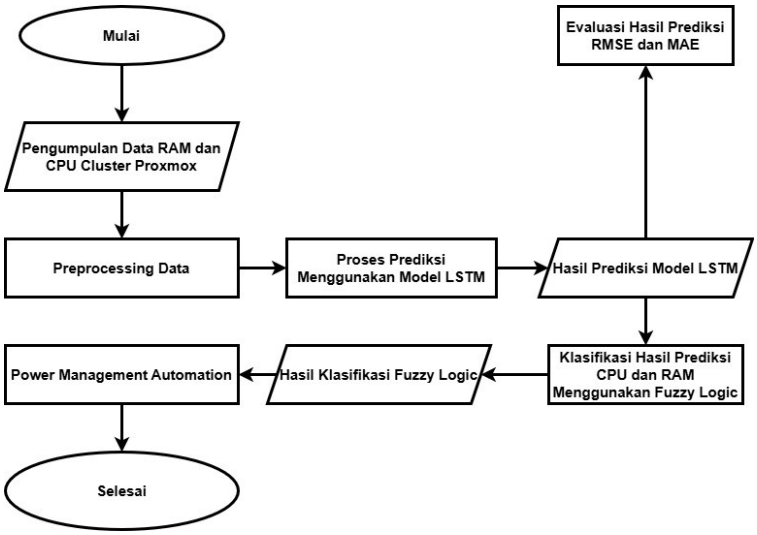
Proses normalisasi dilakukan dengan menggunakan rumus Min-Max di mana nilai baru hasil normalisasi $Xnew$ diperoleh dari nilai awal X yang telah disesuaikan berdasarkan nilai maksimum $\max(X)$ dan minimum $\min(X)$ dalam dataset. Metode ini mengubah skala data ke dalam rentang tertentu, biasanya [0, 1], sehingga seluruh nilai berada dalam skala yang seragam tanpa mengubah pola distribusinya.

Data penggunaan *CPU* dan *RAM* yang telah dikumpulkan kemudian dinormalisasi ke dalam rentang nilai [0, 1] menggunakan MinMaxScaler dari pustaka scikit-learn. Proses normalisasi ini bertujuan untuk mengubah data mentah menjadi skala yang lebih kecil dan seragam, sehingga setiap fitur memiliki bobot yang setara dalam proses pelatihan model. Normalisasi sangat penting dalam konteks algoritma pembelajaran mesin, terutama pada model seperti *LSTM* (*Long Short-Term Memory*), yang lebih sensitif terhadap nilai skala data. Dengan data yang dinormalisasi, *LSTM* dapat bekerja lebih efisien

dan stabil, mengurangi risiko masalah seperti konvergensi lambat atau kesalahan prediksi akibat perbedaan skala antar fitur.

3.3. *Hybrid LSTM – Fuzzy Logic*

Seletah melakukan proses *Preprocessing* data, tahap selanjutnya adalah melakukan penerapan model. Pada tahap penerapan model, data pelatihan dan data pengujian diproses atau dianalisis menggunakan algoritma atau model yang diterapkan yaitu pendekatan gabungan *Long Short-Term Memory* (*LSTM*) dan *Fuzzy Logic*. *Flowchart* atau alur kerja dari metode gabungan *LSTM* dan *Fuzzy Logic* seperti pada Gambar 3.



Gambar 3. *Flowchart* Model Prediksi Gabungan *LSTM* dan Logika *Fuzzy*

Long Short-Term Memory (*LSTM*) merupakan salah satu jenis arsitektur jaringan saraf berulang (*Recurrent Neural Network* atau *RNN*). *LSTM* sangat cocok digunakan untuk pemodelan data berurutan karena kemampuannya dalam mempertahankan informasi dalam jangka waktu yang panjang serta memberikan hasil prediksi yang lebih tinggi dibandingkan dengan model-model lainnya [18]. Operasi dalam sebuah sel *Long Short-Term Memory* (*LSTM*) dapat dinyatakan melalui persamaan berikut:

$$f_t = \sigma_g(W_f . [h_{t-1,} x_t] + b_f \quad (2)$$

$$i_t = \sigma_g(W_i . [h_{t-1,} x_t] + b_i \quad (3)$$

$$\bar{C}_t = \tanh (W_c . [h_{t-1,} x_t] + b_C \quad (4)$$

$$C_t = f_t * C_{t-1} + i_t * \bar{C}_t \quad (5)$$

$$o_t = \sigma_g(W_o . [h_{t-1}, x_t] + b_o \quad (6)$$

$$h_t = o_t * \tanh (C_T) \quad (7)$$

Di mana f_t, i_t, dan o_t masing-masing merupakan vektor gerbang lupa (*forget gate*), gerbang *input* (*input gate*), dan gerbang *output* (*output gate*), $\bar{C}_t$ adalah kandidat keadaan sel (*candidate cell state*), C_t adalah keadaan sel (*cell state*), dan h_t adalah *hidden state* atau *output*. σ_g merupakan fungsi aktivasi sigmoid, sedangkan W_f, W_i, W_c, dan W_o adalah matriks bobot, serta b_f, b_i, b_C, dan b_o adalah vektor bias. Notasi $[h_{t-1,} x_t]$ menunjukkan proses konkatenasi antara *hidden state* sebelumnya dan *input* saat ini.

Hasil prediksi *LSTM* diklasifikasikan menggunakan Logika *Fuzzy* ke dalam tiga kategori beban kerja: rendah, sedang, dan berat. Klasifikasi didasarkan pada nilai prediksi *CPU* dan *RAM* yang dibagi ke dalam tiga tingkat keanggotaan, yaitu *low*, *medium*, dan *high* menggunakan fungsi segitiga. Kombinasi nilai *fuzzy* ini dievaluasi melalui *rule base* untuk menentukan kategori beban kerja akhir.

3.4. *Evaluasi*

Root-Mean-Squared Error (*RMSE*) digunakan untuk mengukur seberapa besar penyimpangan prediksi terhadap nilai aktual dan *Mean Absolute Error* (*MAE*) digunakan untuk mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual. *RMSE* dan *MAE* merupakan dua metrik umum yang digunakan untuk menilai kinerja suatu model[15]. Untuk suatu sampel yang terdiri dari n observasi y (y_i, dengan $i = 1,2, \dots, n$) dan n prediksi model yang bersesuaian $\hat{y}$, *MAE* dan *RMSE* dihitung menggunakan rumus berikut:

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad (8)$$

Dalam evaluasi model prediksi, digunakan notasi berikut: y_i menyatakan nilai aktual pada data ke-i, sedangkan $\hat{y}_i$ merupakan nilai prediksi dari model pada data ke-i. Simbol $\sum$ digunakan untuk menyatakan penjumlahan dari i = 1 hingga n, di mana n merupakan jumlah total data dalam dataset. Notasi ini umumnya digunakan dalam perhitungan metrik evaluasi seperti *MAE* dan *RMSE* untuk mengukur selisih antara prediksi dan nilai aktual secara menyeluruh.

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

Dalam perhitungan evaluasi model, digunakan beberapa notasi matematis, antara lain y_i yang merepresentasikan nilai aktual pada data ke-i, dan $\hat{y}_i$ yang merupakan nilai prediksi dari model pada data ke-i. Jumlah total data dalam dataset dilambangkan dengan n, sementara simbol $\sum$ menunjukkan proses penjumlahan dari i = 1 hingga n. Selain itu, notasi $|y_i - \hat{y}_i|$ digunakan untuk menyatakan nilai absolut dari selisih antara nilai aktual dan nilai prediksi, yang umumnya digunakan dalam perhitungan metrik evaluasi seperti *Mean Absolute Error* (*MAE*).

4. Hasil dan Diskusi

4.1. *Hasil*

Hasil dari penelitian ini digunakan untuk membuktikan bahwa model yang telah dibuat sesuai dengan hasil yang diharapkan dan dapat dimanfaatkan oleh *Cloud Data Center*. Evaluasi model pada penelitian ini dilakukan dengan menggunakan metode *RMSE* dan *MAE* yang terdapat pada Tabel 3. Hasil evaluasi menggunakan metode *RMSE* dan *MAE* dengan mengimplementasikan model gabungan *LSTM* dan *Fuzzy Logic*. Implementasi model gabungan *LSTM* dan *Fuzzy Logic* pada penelitian ini bertujuan untuk mengoptimalkan penggunaan *CPU* dan *RAM* pada *Cluster* Proxmox dengan menggabungkan metode *LSTM* (*Long Short-Term Memory*) untuk prediksi dan *Fuzzy Logic* untuk klasifikasi serta manajemen daya otomatis. Dari hasil pengujian diperoleh hasil dan kesimpulan bahwa model gabungan *LSTM* dan *Fuzzy Logic* yang telah dirancang tidak memiliki kesalahan atau eror dari tahap evaluasi yang telah dilakukan. setiap tahapan akan disertai dengan hasil *capture* dan penjelasan hasil tahapan yang telah dilakukan pada penelitian ini.

4.1.1. Pengumpulan Data

Pengumpulan data penggunaan *CPU* dan *RAM* atau *Memory* pada *Cluster Proxmox* secara langsung menggunakan *API Proxmox*. Data dikumpulkan setiap detik selama rentang waktu tertentu untuk mendapatkan pola beban kerja yang akurat. Pengumpulan data dilakukan dengan menggunakan skrip *Python* yang dijalankan pada *Jupyter Notebook* di sistem operasi Windows 10 dengan memanfaatkan pustaka proxmoxer untuk mengakses *API Proxmox*. Skrip ini dirancang untuk membaca data penggunaan *CPU* dan

RAM atau *Memory* dari semua *node server* dalam *cluster*. Berikut adalah hasil pengumpulan data penggunaan *CPU* dan *RAM* atau *Memory* yang diperoleh dari *Cluster Proxmox*.

Tabel 1. Hasil Pengumpulan Data *CPU* dan *RAM* pada *Cluster* Proxmox

Waktu (Detik)	CPU Usage (%)	RAM Usage (%)
1	100.00%	73.11%
2	100.00%	73.11%
3	100.00%	73.11%
...	...	...
598	36.01%	62.97%
599	36.01%	62.97%
600	36.01%	62.97%

Proses pengumpulan data dilakukan secara berkelanjutan selama rentang waktu 600 detik untuk memastikan data yang dihasilkan mencerminkan pola beban kerja yang sebenarnya. Data ini kemudian digunakan sebagai masukan untuk model pendekatan gabungan *LSTM* dan Logika *Fuzzy* yang dikembangkan dalam penelitian ini.

4.1.2. Preprocessing Data

Min-Max normalisasi adalah teknik normalisasi yang menerapkan transformasi linier pada data asli untuk memastikan keseimbangan perbandingan nilai antara sebelum dan sesudah proses normalisasi [16] dalam[17]. Berikut adalah hasil data penggunaan *CPU* dan *RAM* atau *Memory* setelah dilakukan proses normalisasi.

Tabel 2. Data *CPU* dan *RAM* pada *Cluster* Proxmox Setelah Proses Normalisasi

Waktu (Detik)	CPU Usage (%)	RAM Usage (%)
1	1.000	0.695
2	1.000	0.695
3	1.000	0.695
...	...	...
598	0.335	0.325
599	0.335	0.325
600	0.335	0.325

Data penggunaan *CPU* dan *RAM* dinormalisasi ke rentang [0, 1] menggunakan MinMaxScaler dari pustaka scikit-learn untuk memastikan setiap fitur memiliki skala yang setara. Normalisasi ini penting agar model *LSTM* dapat belajar secara efisien dan stabil, serta menghindari masalah konvergensi lambat atau kesalahan akibat perbedaan skala antar fitur.

4.1.3. Hybrid LSTM-Fuzzy Logic

Seletah melakukan proses *Preprocessing* data, tahap selanjutnya adalah melakukan penerapan model. Pada tahap penerapan model, data pelatihan dan data pengujian diproses atau dianalisis menggunakan algoritma atau model yang diterapkan yaitu pendekatan gabungan *Long Short-Term Memory* (*LSTM*) dan *Fuzzy Logic*. Berikut adalah hasil prediksi beban kerja *Cloud Data Center* secara *real-time* menggunakan pendekatan gabungan *LSTM* dan *Fuzzy Logic*.

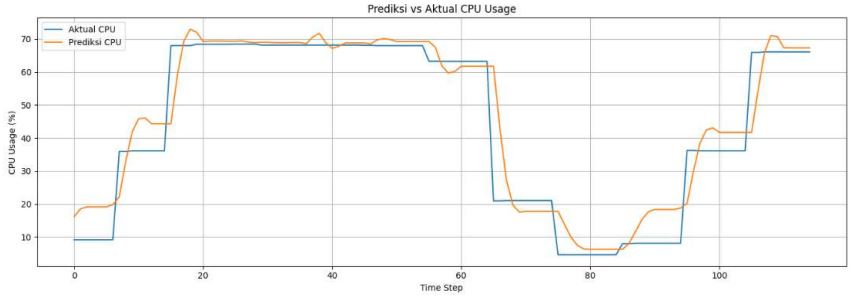
Tabel 3. Hasil prediksi beban kerja *Cloud Data Center*

Waktu (Detik)	Aktual CPU	Prediksi CPU	Aktual RAM	Prediksi RAM	Beban Kerja
1	9.11%	16.15%	63.76%	63.03%	Sedang
2	9.11%	18.51%	63.76%	63.72%	Sedang
3	9.11%	19.10%	63.76%	63.64%	Sedang
...	...	...	...	...	...
113	68.18%	68.96%	73.69%	68.30%	Sedang
114	68.18%	68.62%	73.69%	71.96%	Tinggi
115	68.18%	70.64%	68.99%	72.84%	Tinggi

Dengan pendekatan gabungan *LSTM* dan *Fuzzy Logic*, sistem dapat secara dinamis menyesuaikan penggunaan sumber daya berdasarkan prediksi dan klasifikasi beban kerja, sehingga meningkatkan efisiensi energi serta kinerja sistem secara keseluruhan. Integrasi antara model *LSTM* dan *Fuzzy Logic* memberikan solusi yang lebih cerdas dalam pengelolaan daya dan alokasi sumber daya pada cluster virtualisasi, yang diharapkan dapat meningkatkan efisiensi operasional serta mengurangi konsumsi daya yang tidak perlu.

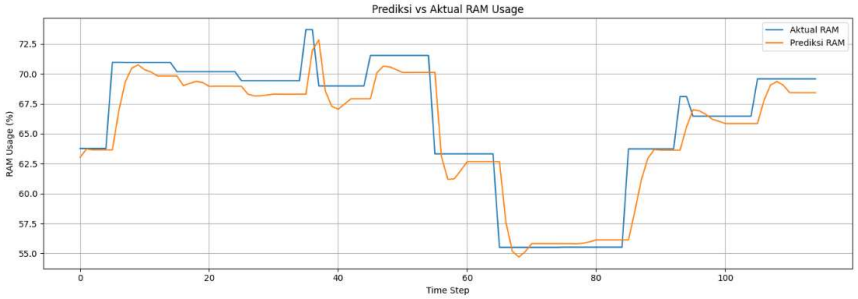
4.1.4. Evaluasi

Evaluasi performa model menggunakan metrik *MAE* dan *RMSE* menunjukkan hasil yang cukup baik untuk prediksi beban kerja *Cloud Data Center* secara *real-time*. Nilai *MAE* sebesar 2.48 (pelatihan) dan 3.09 (pengujian) menunjukkan rata-rata kesalahan prediksi hanya sekitar 3% dalam skala 0–100%. Nilai *RMSE* sebesar 5.15 dan 5.57 mengindikasikan kesalahan prediksi relatif kecil dan tidak terlalu terpengaruh oleh *outlier*. Model cenderung mengalami *underestimation* pada awal prediksi, namun secara umum mampu mengikuti tren beban kerja dengan baik. Gambar 4 memperlihatkan perbandingan antara nilai aktual dan prediksi penggunaan *CPU*, yang menunjukkan kemampuan model dalam mengenali pola variatif beban kerja dari waktu ke waktu.



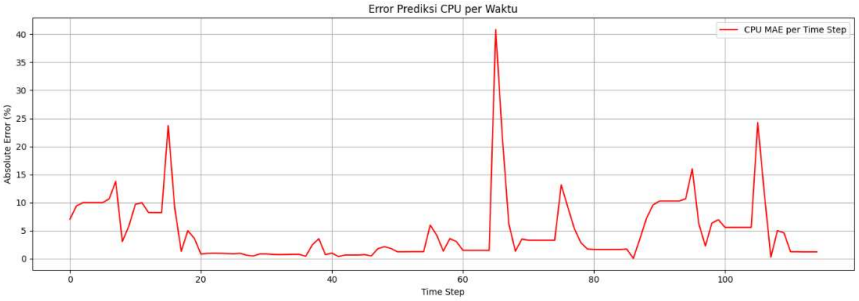
Gambar 4. Prediksi VS Aktual *CPU Usage*

Pada Gambar 4, terlihat bahwa model menunjukkan kemampuan yang baik dalam mengikuti tren penggunaan *CPU*, dengan respons yang konsisten terhadap perubahan beban, meskipun terdapat sedikit keterlambatan saat terjadi fluktuasi tajam. Korelasi antara nilai aktual dan prediksi cukup erat, sejalan dengan hasil evaluasi numerik yang menunjukkan nilai *MAE* dan *RMSE* rendah. Visualisasi prediksi penggunaan *RAM* memperlihatkan kemampuan model dalam menangkap fluktuasi beban kerja. Garis prediksi mendekati garis aktual, menunjukkan performa model yang cukup stabil untuk mendukung monitoring dan pengelolaan sumber daya secara prediktif, seperti terlihat pada Gambar 5.



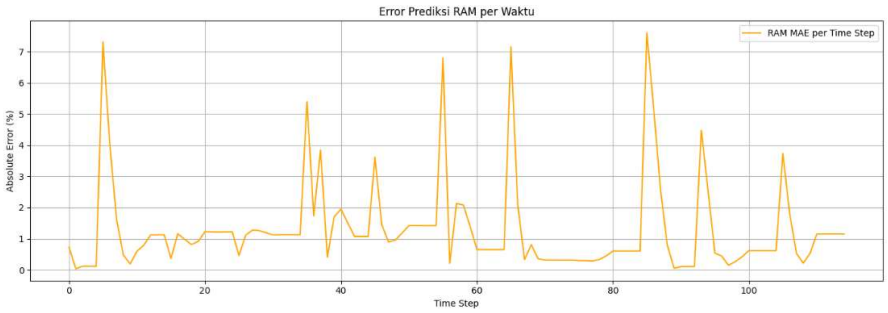
Gambar 5. Prediksi VS Aktual *RAM Usage*

Pada Gambar 5, terlihat bahwa model mampu mengikuti pola penggunaan *RAM* dengan cukup baik, di mana garis prediksi mendekati garis aktual pada sebagian besar *time step*. Meskipun terdapat deviasi kecil saat terjadi perubahan tajam, model tetap menunjukkan sensitivitas yang memadai terhadap fluktuasi beban kerja. Hasil ini sejalan dengan evaluasi numerik yang menunjukkan nilai *MAE* dan *RMSE* yang rendah. Visualisasi grafik yang menunjukan fluktuasi kesalahan prediksi *CPU* dalam satuan *MAE* terhadap waktu memperlihatkan variasi *error* antar *time step* yang berguna untuk mengevaluasi stabilitas model dalam merespons perubahan beban kerja *CPU* secara dinamis, seperti terlihat pada Gambar 6.



Gambar 6. *Error* Prediksi *CPU* per Waktu

Pada Gambar 6, terlihat bahwa sebagian besar *error* prediksi *CPU* berada di bawah 5%, menunjukkan akurasi yang baik. Namun, terdapat lonjakan *error* signifikan pada beberapa *time step*, seperti di sekitar langkah ke-18 dan ke-65, akibat perubahan mendadak yang tidak terantisipasi oleh model. Visualisasi grafik yang menunjukan fluktuasi *MAE* terhadap penggunaan *RAM* per waktu dapat dilihat pada Gambar 7.



Gambar 7. *Error* Prediksi *RAM* per Waktu

Pada Gambar 7, terlihat bahwa sebagian besar *error* prediksi *RAM* berada di bawah 4%, menunjukkan performa model yang stabil dan akurat. Beberapa lonjakan *error* terdeteksi pada *time step* ke-5, 49, 62, dan 82 akibat perubahan drastis yang tidak terprediksi. Grafik ini membantu mengidentifikasi titik kelemahan model dan mengonfirmasi bahwa performa keseluruhan masih dalam batas yang dapat diterima.

4.2. *Diskusi*

Penelitian ini menunjukkan bahwa kombinasi model *LSTM* dan Logika *Fuzzy* efektif dalam memprediksi dan mengklasifikasikan beban kerja *Cloud Data Center* secara *real-time*. Model mampu mengikuti pola penggunaan *CPU* dan *RAM* dengan akurasi yang baik, sebagaimana dibuktikan oleh nilai *MAE* dan *RMSE* yang rendah. Meskipun demikian, terdapat beberapa lonjakan *error* pada saat terjadi perubahan beban kerja yang drastis, menunjukkan bahwa model masih perlu penyempurnaan. Integrasi fitur tambahan seperti *I/O disk*, aktivitas jaringan, atau pendekatan *adaptive learning* dapat meningkatkan akurasi. Secara keseluruhan, model ini berpotensi mendukung sistem manajemen beban kerja *Cloud Data Center* yang lebih efisien dan adaptif.

Tabel 4. Perbandingan Dengan Penelitian Terdahulu

Peneliti	Data	Metode	Evaluasi	Hasil
Yadav et al.	Log layanan web Universitas Complutense Madrid	Long Short-Term Memory (LSTM)	Model LSTM mencapai MAE 0,043, RMSE 0,075, dan MSE 0,066	Analisis
Koot & Wijnhoven	Lalu lintas Internet Protocol dari Cisco Systems 2017-2022	System Dynamic Models dan Scenario Development Methods	Model memprediksi kebutuhan listrik pusat data meningkat signif-ikan pada 2030, di-pengaruhi oleh perilaku pengguna, akhir hukum Moore, dan pertum-buhan IoT. Estimasi konsumsi bervariasi lu-as 343–1031 TWh.	Analisis
Shi & Jiang	Beban CPU dari Google	Three-Way Ensemble Prediction (TWD-RCPM), Simulated Annealing Algorithm, dan Long Short-Term Memory	Model menunjukkan pengurangan kesalahan prediksi relatif kumulatif sebesar 69,05% dari DMASVR-3WD, 68,67% dari ARIMA, dan 72,63% dari NN berdasarkan RMSE, MAE, SMAPE, dan QR.	Analisis
Xie et al.	Sumber daya kontainer (RAM dan CPU) yang dikirim melalui akuisisi sumber daya kontainer dan disimpan dalam database InfluxDB	Model prediksi hibrida yang meng-gabungkan ARIMA dan Triple Exponential Smoothing	Model meningkatkan akurasi prediksi secara signifikan dibanding ARIMA (52,64%), triple exponential (20,15%), dan ANN+SADE (203,72%) berdasarkan MAPE dan MSE, dengan overhead waktu yang minimal	Implementasi
Penelitian ini	Sumber daya Cluster Proxmox (RAM dan CPU) Virtual Environment yang dikirim melalui Proxmox API	Pendekatan gabungan Long Short-Term Memory (LSTM) dan Fuzzy Logic	Model menunjukkan tingkat kesalahan rendah dengan MAE sekitar 2.48 (pelatihan) dan 3.09 (pengujian), serta RMSE kecil yaitu 5.15 dan 5.57, menandakan prediksi penggunaan CPU dan RAM cukup akurat pada skala 0–100%.	Implementasi

5. Kesimpulan

Berdasarkan hasil evaluasi, model prediksi menunjukkan kemampuan yang baik dalam mengikuti pola penggunaan *CPU* dan *RAM*, khususnya saat beban kerja berada dalam kondisi stabil. Hasil perbandingan antara nilai prediksi dan nilai aktual memperlihatkan bahwa model mampu memberikan estimasi yang mendekati kenyataan. Mayoritas kesalahan prediksi berada pada kisaran rendah, baik untuk penggunaan *CPU* maupun *RAM*, yang menunjukkan bahwa model cukup andal dalam merespons pola perubahan beban kerja secara umum.

Model menghasilkan nilai *MAE* sebesar 2.48 pada data pelatihan dan 3.09 pada data pengujian, serta *RMSE* sebesar 5.15 dan 5.57. Angka-angka ini menunjukkan bahwa rata-rata kesalahan prediksi berada di bawah 3%, dan prediksi yang dihasilkan tetap stabil. Meskipun demikian, terdapat beberapa lonjakan *error* pada titik-titik tertentu yang kemungkinan disebabkan oleh perubahan beban kerja yang tiba-tiba dan tidak dapat di antisipasi oleh model, ini menunjukkan bahwa masih terdapat ruang untuk peningkatan, terutama pada bagian-bagian yang sensitif terhadap perubahan mendadak. Secara keseluruhan, model ini dapat dimanfaatkan sebagai dasar dalam sistem pengambilan keputusan otomatis, seperti penjadwalan beban kerja dan manajemen daya di lingkungan virtualisasi atau cloud computing. Sebagai saran untuk pengembangan penelitian selanjutnya, model dapat ditingkatkan dengan menambahkan fitur tambahan seperti penggunaan *disk I/O*, aktivitas jaringan, dan suhu prosesor sebagai variabel input. Selain itu, penerapan teknik *fine-tuning parameter* serta eksplorasi arsitektur *deep learning* lain seperti *GRU* atau *hybrid CNN-LSTM* juga dapat menjadi pendekatan alternatif untuk meningkatkan akurasi dan adaptivitas model dalam menghadapi dinamika beban kerja yang kompleks dan tidak terduga.

Referensi

[1] C. Guo, F. Luo, Z. Cai, and Z. Y. Dong, "Integrated energy systems of data centers and smart grids: State-of-the-art and future opportunities," *Applied Energy*, vol. 301, 2021, pp. 117474, https://doi.org/10.1016/j.apenergy.2021.117474

[2] K.-T. Seo, H.-S. Hwang, I.-Y. Moon, O.-Y. Kwon, and B.-J. Kim, "Performance comparison analysis of Linux container and virtual machine for building cloud," *Advanced Science and Technology Letters*, vol. 66, pp. 105–111, 2014, https://doi.org/10.14257/astl.2014.66.25.

[3] A. Alarifi, et al., "Energy-efficient hybrid framework for green cloud computing," IEEE Access, vol. 8, pp. 115356–115369, 2020, https://doi.org/10.1109/ACCESS.2020.3002184

[4] M. Ibrahim, M. Imran, F. Jamil, Y. J. Lee, and D. H. Kim, "EAMA: Efficient adaptive migration algorithm for cloud data centers (CDCs)," *Symmetry*, vol. 13, no. 4, 2021, https://doi.org/10.3390/sym13040690

[5] Y. Xie *et al*., "Real-time prediction of Docker container resource load based on a hybrid model of ARIMA and triple exponential smoothing," *IEEE Trans. Cloud Comput*, vol. 10, no. 2, pp. 1386–1401, 2022, https://doi.org/10.1109/TCC.2020.2989631

[6] R. Shi and C. Jiang, "Three-way ensemble prediction for workload in the data center," *IEEE Access*, vol. 10, pp. 10021–10030, 2022, https://doi.org/10.1109/ACCESS.2022.3145426

[7] S. Ştefan and V. Niculescu, "Microservice-oriented workload prediction using deep learning," *E-Inform. Softw. Eng. J.*, vol. 16, no. 1, 2022, https://doi.org/10.37190/e-Inf220107

[8] B. Gülmez, "Stock price prediction with optimized deep LSTM network with artificial rabbits optimization algorithm," *Expert Syst. Appl.*, vol. 227, Art. no. 120346, 2023, https://doi.org/10.1016/j.eswa.2023.120346

[9] Y. Lin, Y. Yan, J. Xu, Y. Liao, and F. Ma, "Forecasting stock index price using the CEEMDAN-LSTM model," *N. Am. J. Econ. Finance*, vol. 57, 2021, Art. no. 101421, https://doi.org/10.1016/j.najef.2021.101421

[10] M. P. Yadav, N. Pal, and D. K. Yadav, "Workload prediction over cloud server using time series data," in *Proc. Confluence 2021: 11th Int. Conf. Cloud Comput., Data Sci. Eng.*, pp. 267–272, 2021, https://doi.org/10.1109/Confluence51648.2021.9377032

[11] J.-S. R. Jang, "ANFIS: Adaptive-network-based fuzzy inference system," *IEEE Trans. Syst., Man, Cybern.*, vol. 23, no. 3, 1993, https://doi.org/10.1109/21.256541

[12] Z. Amekraz and M. Y. Hadi, "A cluster workload forecasting strategy using a higher order statistics based ARMA model for IaaS cloud services," *Int. J. Netw. Virtual Organ.*, vol. 26, no. 1/2, p. 3, 2022, https://doi.org/10.1504/IJNVO.2022.121844

[13] M. Koot and F. Wijnhoven, "Usage impact on data center electricity needs: A system dynamic forecasting model," *Appl. Energy*, vol. 291, 2021, Art. no. 116798, https://doi.org/10.1016/j.apenergy.2021.116798

[14] Z. Yang and H. Cao, "Preprocessing is not needed: An end-to-end solution for physiological signals based emotion recognition," 2024, https://openreview.net/pdf?id=egwH8GLlMN

[15] T. O. Hodson, "Root-mean-square error (RMSE) or mean absolute error (MAE): When to use them or not," *Geosci. Model Dev.*, vol. 15, no. 14, pp. 5481–5487, 2022, https://doi.org/10.5194/gmd-15-5481-2022

[16] A. Salamh, M. Al-Rawahnaa, A. Y. Bader, and A. Hadid, "Data mining for education sector, a proposed concept," *J. Appl. Data Sci*., vol. 1, no. 1, pp. 1–10, 2020. https://bright-journal.org/Journal/index.php/JADS/article/view/6/6

[17] Hendri, T. Wahyuningsih, and E. Rahwanto, "Comparison of Min-Max normalization and Z-score normalization in the K-nearest neighbor (KNN) algorithm to test the accuracy of types of breast cancer," *UCI Machine Learning Repository*, 2021, https://ijiis.org/index.php/IJIIS/article/view/73

[18] N. Sulistianingsih and G. H. Martono, "Comparative study on stock movement prediction using hybrid deep learning model," *ECTI Trans. Comput. Inf. Technol*., vol. 18, no. 4, pp. 531–542, 2024, https://doi.org/10.37936/ecti-cit.2024184.256303