LAPORAN KEMAJUAN
PENELITIAN INTERNAL UPY
SKEMA PENELITIAN DASAR

Pengembangan Model Prediksi Konsumsi Energi Berbasis Data Time Series Menggunakan Integrasi Long Short-Term Memory (LSTM) dan Fuzzy Logic

Oleh :

Muhammad Fairuzabadi (Dosen Prodi Informatika)
Puji Handayani Putri (Dosen Prodi Informatika)
Gema Kharismajati (Dosen Prodi Sistem Informasi).
Anindya Elga Frinayanti (Mahasiswa Prodi Informatika)
Riswan Dwi Cahyono (Mashasiswa Prodi Informatika)

Program Studi Informatika
Fakultas Sains dan Teknologi

Penelitian ini Terlaksana Dengan Dana Bantuan Penelitian dari
Universitas PGRI Yogyakarta Melalui Anggaran LPPM Tahun 2025/2026

UNIVERSITAS PGRI YOGYAKARTA
TAHUN 2025

UNIVERSITAS PGRI YOGYAKARTA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. PGRI I No. 117 Sonosewu, Yogyakarta, 55182 Telp/Fax: (0274) 376808
Web: http://lppm.upy.ac.id Email: lppm@upy.ac.id

HALAMAN PENGESAHAN

1. Judul Penelitian	:	Pengembangan Model Prediksi Konsumsi Energi Berbasis Data Time Series Menggunakan Integrasi Long Short-Term Memory (LSTM) dan Fuzzy Logic
2. Skema Penelitian	:	Penelitian Dasar
3. Identitas Ketua Peneliti		
Nama Pangkat/Golongan NIS Fakultas/Program Studi Telp/Email	: : : : :	Muhammad Fairuzabadi, S.Si., M.Kom. Lektor 1999022 220160 1 2001 FST/Informatika 089699870225/fairuz@upy.ac.id
4. Identitas Anggota 1 Nama Pangkat/Golongan NIS Fakultas/Program Studi Telp/Email	 : : : : :	 Puji Handayani Putri, S.T., M.Kom. Lektor 19900222 201601 2 001 FST/Informatika 082323565460/ pujihp@upy.ac.id
5. Identitas Anggota 2 Nama Pangkat/Golongan NIS Fakultas/Program Studi Telp/Email	 : : : : :	 Puji Handayani Putri, S.T., M.Kom. Lektor 19900222 201601 2 001 FST/Informatika 082323565460/ pujihp@upy.ac.id
6. Identitas Mahasiswa Nama NPM Fakultas/Program Studi Telp/Email	 : : : :	 Anindya Elga Frinayanti 22111100015 FST/Informatika 085741195821/anindyaelga03@gmail.com
7. Identitas Mahasiswa Nama NPM Fakultas/Program Studi Telp/Email	 : : : :	 Riswan Dwi Cahyono 24111100057 FST/Informatika 087812567711/ riswandwicahyono@gmail.com
8. Luaran Yang Dihasilkan	:	1. Artikel ilmiah pada jurnal terindeks Scopus. 2. Model/prototipe prediksi konsumsi energi berbasis Fuzzy-Gated Attention LSTM (FGA-LSTM). 3. Buku referensi berjudul Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series. 4. Hak Kekayaan Intelektual (HKI) Program Komputer Fuzzy-Gated Attention LSTM (FGA-LSTM).
9. Waktu Pelaksanaan	:	12 Bulan
10. Total Pendanan	:	Rp. 15.000.000,-
11. Sumber Dana	:	LPPM UPY

UNIVERSITAS PGRI YOGYAKARTA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. PGRI I No. 117 Sonosewu, Yogyakarta, 55182 Telp/Fax: (0274) 376808
Web: http://lppm.upy.ac.id Email: lppm@upy.ac.id

Yogyakarta, 23 Juni 2026

Mengetahui,
Ketua Program Studi

Aditya Wahana, S.Pd.T., M.Kom.
NIS. 19850424 201604 1 005

Ketua Penelitian

Muhammad Fairuzabadi, S.Si., M.Kom.
NIS. 1999022 220160 1 2001

Menyetujui,
Kepala Pusat Penelitian UPY

Wahyu Sugianto, M.Si.
NIS. 19950801 202010 1 003

UNIVERSITAS PGRI YOGYAKARTA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. PGRI I No. 117 Sonosewu, Yogyakarta, 55182 Telp/Fax: (0274) 376808
Web: http://lppm.upy.ac.id Email: lppm@upy.ac.id

Isian Substansi Proposal

LAPORAN PENELITIAN INTERNAL UNIVERSITAS PGRI YOGYAKARTA

Petunjuk:Pengusul hanya diperkenankan mengisi di tempat yang telah disediakan sesuai dengan petunjuk pengisian dan tidak diperkenankan melakukan modifikasi template atau penghapusan di setiap bagian.

RINGKASAN

Ringkasan penelitian tidak lebih dari 500 kata yang berisi latar belakang penelitian, tujuan dan tahapan metode penelitian, hasil penelitian dan luaran yang ditargetkan.

Penelitian ini dilatarbelakangi oleh meningkatnya kebutuhan energi listrik serta ketersediaan data konsumsi energi dalam bentuk deret waktu yang bersumber dari smart meter dan perangkat Internet of Things (IoT). Data konsumsi energi memiliki karakteristik nonlinier, fluktuatif, musiman, serta rentan terhadap noise dan anomali sensor, sehingga sulit diprediksi secara akurat menggunakan pendekatan statistik konvensional. Model deep learning, khususnya Long Short-Term Memory (LSTM), memiliki kemampuan dalam mempelajari pola temporal dan dependensi jangka panjang. Namun, LSTM dan mekanisme attention konvensional masih memiliki keterbatasan dalam membedakan pola historis yang valid dengan gangguan data, serta cenderung bersifat black box. Oleh karena itu, diperlukan pengembangan model hibrida yang tidak hanya akurat, tetapi juga mampu menangani ketidakpastian dan memberikan interpretasi yang lebih mudah dipahami.

Tujuan penelitian ini adalah mengembangkan dan menganalisis model prediksi konsumsi energi berbasis data time series melalui integrasi LSTM, mekanisme attention, dan Fuzzy Logic. Model yang dikembangkan diberi nama Fuzzy-Gated Attention LSTM (FGA-LSTM), yaitu arsitektur hibrida yang memanfaatkan Interval Type-2 Fuzzy Logic untuk menghasilkan bobot perhatian yang adaptif terhadap ketidakpastian dan noise pada data IoT.

Tahapan metode penelitian meliputi studi literatur, pengumpulan dataset konsumsi energi berbasis time series, pra-pemrosesan data, pembentukan window data, pengembangan model LSTM dan Fuzzy-Gated Attention, pelatihan model, evaluasi performa menggunakan metrik MAE, RMSE, MAPE, R^2, serta analisis komparatif dengan model baseline seperti Standard LSTM, LSTM dengan standard attention, TCN, dan Transformer. Selain itu, penelitian juga melakukan analisis explainability melalui visualisasi fuzzy attention untuk menunjukkan pola waktu yang dianggap penting oleh model.

Hasil penelitian menunjukkan bahwa model FGA-LSTM mampu meningkatkan akurasi prediksi dan interpretabilitas model. Pada pengujian terhadap dataset London Smart Meter dan UCI Electricity, model menunjukkan penurunan RMSE dibandingkan Standard LSTM dan LSTM-SA, memperoleh nilai R^2 yang baik, serta menghasilkan visualisasi perhatian yang dapat menjelaskan pengaruh pola musiman dan anomali terhadap hasil prediksi.

Luaran penelitian meliputi artikel ilmiah berjudul “Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting” yang telah submitted pada Jurnal RESTI, buku referensi berjudul “Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series” yang telah mencapai draft 50% atau 6 bab, serta HKI program komputer “Fuzzy-Gated Attention LSTM (FGA-LSTM)” yang telah terdaftar dengan Nomor Pencatatan 001303048.

KATA KUNCI

Kata kunci maksimal 5 kata

FGA-LSTM; LSTM; Fuzzy Logic; Prediksi Time Series; Smart Meter

LATAR BELAKANG

Latar belakang penelitian tidak lebih dari 500 kata yang berisi latar belakang dan **permasalahan** yang akan diteliti, **tujuan**, **rumusan permsalahan** dan **pendekatan pemecahan masalah**. Pada bagian ini perlu dijelaskan uraian tentang keterkaitan penelitian yang diajukan dengan **road map penelitian UPY**.

Peningkatan kebutuhan energi listrik pada bangunan, fasilitas publik, dan lingkungan berbasis Internet of Things (IoT) mendorong pentingnya sistem prediksi konsumsi energi yang akurat dan adaptif. Perangkat smart meter dan sensor IoT mampu menghasilkan data konsumsi energi secara berkala dalam bentuk deret waktu atau time series. Namun, data tersebut umumnya memiliki karakteristik nonlinier, fluktuatif, musiman, serta rentan terhadap noise dan anomali sensor. Kondisi ini menyebabkan prediksi konsumsi energi menjadi tantangan penting, terutama ketika data digunakan untuk mendukung efisiensi energi, pengelolaan beban puncak, dan pengambilan keputusan pada sistem smart grid maupun smart building.

Permasalahan utama dalam penelitian ini adalah masih terbatasnya model prediksi yang mampu menggabungkan akurasi prediksi numerik dengan kemampuan menangani ketidakpastian dan interpretabilitas model. Long Short-Term Memory (LSTM) dikenal mampu mempelajari pola temporal dan dependensi jangka panjang pada data time series. Namun, LSTM dan mekanisme attention standar cenderung bersifat deterministik serta belum optimal dalam membedakan pola historis yang valid dengan gangguan data atau anomali IoT. Akibatnya, model dapat memberikan bobot perhatian yang kurang tepat pada data yang mengandung noise, sehingga berpotensi meningkatkan kesalahan prediksi.

Tujuan penelitian ini adalah mengembangkan model prediksi konsumsi energi berbasis data time series dengan mengintegrasikan LSTM, mekanisme attention, dan Fuzzy Logic. Model yang dikembangkan diberi nama Fuzzy-Gated Attention LSTM (FGA-LSTM), yaitu model hibrida yang memanfaatkan Interval Type-2 Fuzzy Logic untuk menghasilkan bobot perhatian yang lebih adaptif terhadap ketidakpastian data.

Rumusan masalah dalam penelitian ini meliputi: (1) bagaimana merancang model FGA-LSTM untuk prediksi konsumsi energi berbasis data time series; (2) bagaimana mengintegrasikan Fuzzy Logic ke dalam mekanisme attention agar model mampu merespons noise dan anomali data; dan (3) bagaimana kinerja model FGA-LSTM dibandingkan dengan model baseline seperti Standard LSTM, LSTM dengan standard attention, TCN, dan Transformer.

Pendekatan pemecahan masalah dilakukan melalui pengumpulan dan pra-pemrosesan dataset konsumsi energi, pembentukan window data time series, pengembangan arsitektur FGA-LSTM, pelatihan model, evaluasi menggunakan metrik MAE, RMSE, MAPE, dan R^2, serta analisis explainability melalui visualisasi fuzzy attention. Penelitian ini sejalan dengan Roadmap Penelitian Universitas PGRI Yogyakarta 2025–2028, khususnya pada pengembangan rekayasa dan teknologi berbasis data, kecerdasan buatan, data science, sistem cerdas, serta pemanfaatan teknologi untuk mendukung efisiensi sumber daya dan ketahanan sistem.

TINJAUAN PUSTAKA

Tinjauan pustaka tidak lebih dari 1000 kata dengan mengemukakan ***state of the art*** dalam bidang yang diteliti/teknologi yang dikembangkan. Penyajian dalam bagan dapat dibuat dalam bentuk JPG/PNG yang kemudian disisipkan dalam isian ini. Sertakan peta jalan (***road map***) penelitian anda setidaknya selama 5 tahun. Gunakan sumber pustaka/referensi primer yang relevan dan dengan mengutamakan hasil penelitian pada jurnal ilmiah dan/atau paten 10 tahun terakhir. Sitasi disusun dan ditulis berdasarkan sistem nomor sesuai dengan urutan pengutipan (**IEEE Style**).

Perkembangan smart grid dan smart building mendorong pemanfaatan Internet of Things (IoT) untuk memantau konsumsi energi secara real-time melalui smart meter, sensor arus, sensor tegangan, sensor okupansi, dan gateway komunikasi. Data yang dihasilkan umumnya berbentuk deret waktu atau time series dengan resolusi tinggi, misalnya per 15 menit atau 30 menit. Data tersebut penting untuk mendukung prediksi beban, efisiensi energi, demand response, pengendalian beban puncak, dan pengambilan keputusan operasional [1], [2]. Namun, karakteristik data konsumsi energi tidak sederhana karena bersifat nonlinier, fluktuatif, musiman, tidak stasioner, serta rentan terhadap missing value, noise, dan anomali sensor [3], [4]. Kondisi ini menyebabkan metode statistik konvensional sering kurang memadai untuk menangkap pola temporal jangka panjang dan perubahan konsumsi energi yang kompleks.

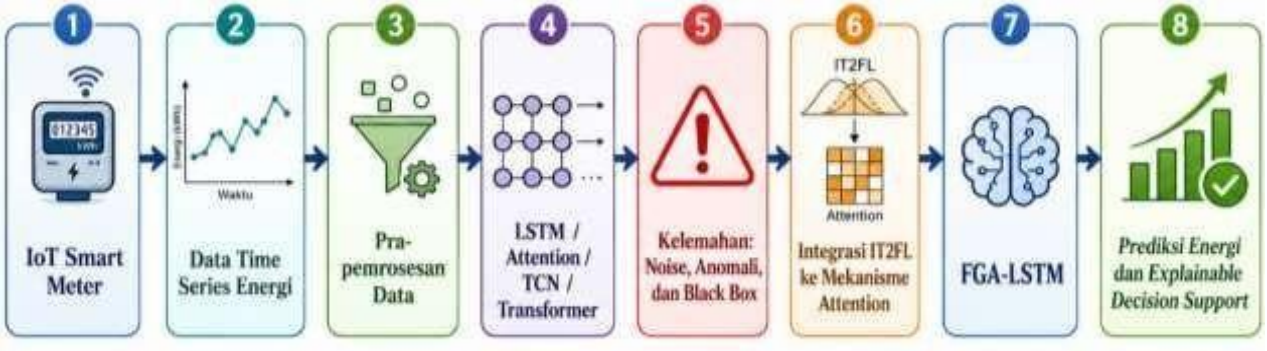
State of the art dalam prediksi konsumsi energi menunjukkan pergeseran dari pendekatan statistik menuju machine learning dan deep learning. Model seperti Artificial Neural Network, Support Vector Regression, Random Forest, dan Gradient Boosting telah digunakan dalam prediksi beban listrik, tetapi masih membutuhkan rekayasa fitur yang cukup besar dan sering kurang stabil pada data sekuensial yang panjang [5], [6]. Deep learning kemudian menjadi pendekatan dominan karena mampu mengekstraksi pola data secara otomatis. Dalam konteks data time series, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Temporal Convolutional Network (TCN), dan Transformer banyak digunakan untuk memodelkan dependensi temporal [7], [8].

LSTM merupakan salah satu arsitektur yang banyak digunakan dalam short-term load forecasting karena mampu mengatasi masalah vanishing gradient dan mempelajari dependensi jangka panjang pada data deret waktu [9], [10]. Pengembangan berikutnya banyak memanfaatkan mekanisme attention agar model dapat memberikan bobot lebih besar pada time-step historis yang dianggap penting [11]. Attention-LSTM, multi-head attention, self-attention, dan Transformer telah menunjukkan peningkatan kinerja prediksi pada berbagai dataset energi [12], [13]. Akan tetapi, mekanisme attention standar umumnya menggunakan pembobotan deterministik, seperti dot-product attention dan softmax. Mekanisme ini belum secara eksplisit membedakan apakah nilai historis yang diberi bobot tinggi berasal dari pola konsumsi yang valid atau dari gangguan sensor. Pada data IoT, kondisi ini berisiko membuat model memperhatikan noise atau anomali sebagai pola penting.

Fuzzy Logic menjadi relevan karena mampu merepresentasikan ketidakpastian dan ambiguitas melalui aturan linguistik yang mudah dipahami manusia [14]. Dalam sistem prediksi energi, Fuzzy Logic banyak digunakan untuk klasifikasi beban, pengambilan keputusan, dan interpretasi hasil prediksi. Namun, sebagian besar pendekatan fuzzy masih ditempatkan pada level output, misalnya mengubah hasil prediksi numerik menjadi kategori rendah, sedang, tinggi, atau kritis. Integrasi Fuzzy Logic ke dalam inti mekanisme pembelajaran deep learning, khususnya attention, masih relatif terbatas. Interval Type-2 Fuzzy Logic (IT2FL) memiliki keunggulan dibanding Type-1 Fuzzy Logic karena mampu memodelkan ketidakpastian fungsi keanggotaan melalui Footprint of Uncertainty (FOU) [15]. Dengan demikian, IT2FL berpotensi digunakan untuk menyaring pengaruh noise pada data time series sebelum model memberikan bobot perhatian.

Research gap penelitian ini terletak pada tiga aspek. Pertama, model LSTM dan Attention-LSTM yang umum digunakan masih berfokus pada akurasi numerik, tetapi belum optimal dalam menangani noise dan anomali IoT secara internal. Kedua, Fuzzy Logic pada penelitian prediksi energi umumnya digunakan sebagai komponen interpretasi output, belum dimasukkan langsung ke dalam mekanisme attention. Ketiga, model deep learning untuk prediksi energi masih sering bersifat black box, sehingga sulit menjelaskan alasan model

memilih time-step tertentu sebagai dasar prediksi. Untuk menjawab kesenjangan tersebut, penelitian ini mengembangkan Fuzzy-Gated Attention LSTM (FGA-LSTM), yaitu arsitektur hibrida yang mengintegrasikan LSTM, attention, dan IT2FL. Model ini menggunakan FOU sebagai filter ketidakpastian untuk menghasilkan bobot perhatian yang lebih adaptif terhadap pola historis, variansi lokal, dan konteks temporal [16].



Gambar 1. Alur pengembangan model FGA-LSTM untuk prediksi energi berbasis data time series IoT.

Peta jalan penelitian lima tahun disusun sebagai kesinambungan riset kecerdasan buatan, deep learning, fuzzy logic, dan prediksi data time series. Roadmap penelitian adalah sebagai berikut:

Tabel 1. Roadmap Penelitian

Tahun	Fokus Roadmap	Target Capaian
2025	Perancangan metodologi LSTM–Fuzzy Logic	Kajian literatur, rancangan dataset, dan rancangan model
2026	Implementasi dan pengujian FGA-LSTM	Model hibrida, evaluasi MAE, RMSE, MAPE, R^2, artikel, buku, dan HKI
2027	Pengembangan prototipe sistem	Dashboard/API prediksi energi berbasis model FGA-LSTM
2028	Validasi lanjutan dan hilirisasi	Publikasi lanjutan, penguatan HKI, dan uji coba pada data real-time
2029	Pengembangan model adaptif dan explainable AI	Model prediksi energi lintas domain yang lebih adaptif, interpretatif, dan siap dikembangkan untuk sistem pendukung keputusan

METODE

Isian metode atau cara untuk mencapai tujuan yang telah ditetapkan tidak lebih dari 1000 kata. Pada bagian metoda wajib dilengkapi dengan diagram alir penelitian yang menggambarkan apa yang sudah dilaksanakan dan yang akan dikerjakan selama waktu yang diusulkan. Format diagram alir dapat berupa file JPG/PNG. Metode penelitian harus memuat sekurang-kurangnya **prosedur penelitian**, **hasil yang diharapkan**, **indikator capaian** yang ditargetkan, serta tugas anggota tim/mitra pada kegiatan penelitian. Tuliskan dengan detail hasil yang diharapkan/luaran yang dijanjikan luaran yang dijanjikan.

Metode penelitian dirancang untuk mencapai tujuan utama penelitian, yaitu mengembangkan model prediksi konsumsi energi berbasis data time series dengan mengintegrasikan Long Short-Term Memory (LSTM), mekanisme attention, dan Interval Type-2 Fuzzy Logic (IT2FL). Model yang dikembangkan diberi nama Fuzzy-Gated Attention LSTM (FGA-LSTM). Penelitian ini menggunakan pendekatan eksperimen

komputasional dengan tahapan sistematis mulai dari studi literatur, pengumpulan data, pra-pemrosesan data, pengembangan model, pengujian, analisis hasil, hingga penyusunan luaran penelitian.

Prosedur penelitian terdiri atas beberapa tahap. Tahap pertama adalah studi literatur dan analisis state of the art. Pada tahap ini dilakukan kajian terhadap penelitian terkait prediksi konsumsi energi, data time series, smart meter, LSTM, attention mechanism, Temporal Convolutional Network (TCN), Transformer, Fuzzy Logic, dan Interval Type-2 Fuzzy Logic. Hasil yang diharapkan dari tahap ini adalah diperolehnya dasar teoritis, research gap, dan kerangka konseptual pengembangan FGA-LSTM. Indikator capaiannya adalah tersusunnya tinjauan pustaka, peta state of the art, dan rancangan awal model.

Tahap kedua adalah pengumpulan dataset. Data yang digunakan berupa dataset konsumsi energi berbasis time series, yaitu London Smart Meter dan UCI Electricity. Dataset tersebut dipilih karena merepresentasikan pola konsumsi energi yang fluktuatif, periodik, dan relevan dengan permasalahan smart grid atau smart building. Hasil yang diharapkan adalah tersedianya dataset yang dapat digunakan untuk pelatihan dan pengujian model. Indikator capaiannya adalah dataset berhasil dikumpulkan, dipahami strukturnya, dan siap diproses lebih lanjut.

Tahap ketiga adalah pra-pemrosesan data. Kegiatan pada tahap ini meliputi pembersihan data, penanganan missing value, normalisasi, pembentukan sliding window, serta pembagian data menjadi data latih dan data uji berdasarkan urutan waktu. Pembagian data tidak dilakukan secara acak untuk menjaga karakteristik deret waktu dan mencegah data leakage. Hasil yang diharapkan adalah terbentuknya data input dan target yang sesuai untuk model prediksi time series. Indikator capaiannya adalah tersedianya data latih dan data uji dalam format sekuensial.

Tahap keempat adalah pengembangan model FGA-LSTM. Model ini diawali dengan pemrosesan input time series menggunakan LSTM untuk menangkap pola temporal dan dependensi jangka panjang. Hidden states yang dihasilkan LSTM kemudian diproses oleh mekanisme Fuzzy-Gated Attention berbasis IT2FL. Pada bagian ini digunakan beberapa komponen, yaitu similarity score, local variance/noise indicator, temporal context, Footprint of Uncertainty (FOU), type-reduction, dan defuzzification. Komponen tersebut digunakan untuk menghasilkan bobot attention yang lebih adaptif terhadap ketidakpastian dan noise pada data. Hasil yang diharapkan adalah terbentuknya arsitektur FGA-LSTM yang mampu menghasilkan prediksi energi sekaligus memberikan penjelasan melalui bobot fuzzy attention. Indikator capaiannya adalah tersusunnya kode program, arsitektur model, dan dokumen deskripsi program.

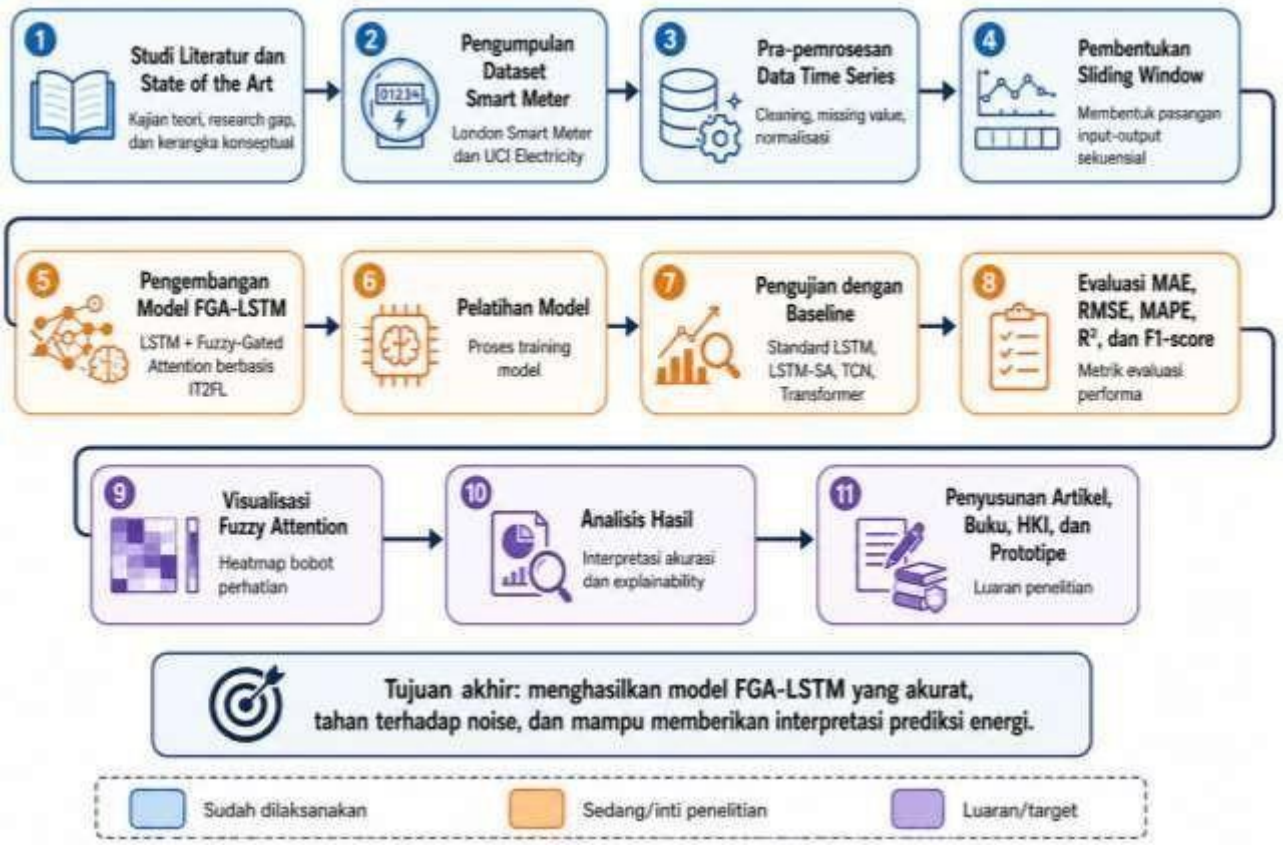
Tahap kelima adalah pelatihan dan pengujian model. Model FGA-LSTM dilatih menggunakan data latih, kemudian diuji menggunakan data uji. Kinerja model dibandingkan dengan beberapa baseline, yaitu Standard LSTM, LSTM dengan standard attention, TCN, dan Transformer. Metrik evaluasi yang digunakan meliputi MAE, RMSE, MAPE, R^2, dan F1-score untuk klasifikasi beban kritis. Hasil yang diharapkan adalah diperolehnya nilai evaluasi yang menunjukkan kinerja FGA-LSTM dibandingkan model baseline. Indikator capaiannya adalah tersusunnya tabel hasil eksperimen, grafik performa, dan analisis perbandingan model.

Tahap keenam adalah analisis explainability. Pada tahap ini bobot fuzzy attention divisualisasikan dalam bentuk heatmap untuk menunjukkan time-step historis yang paling berpengaruh terhadap hasil prediksi. Analisis ini digunakan untuk membuktikan bahwa FGA-LSTM tidak hanya menghasilkan prediksi numerik, tetapi juga memberikan interpretasi terhadap proses prediksi. Hasil yang diharapkan adalah tersedianya visualisasi fuzzy attention dan analisis interpretatif. Indikator capaiannya adalah terbentuknya heatmap attention dan penjelasan kontribusi time-step historis terhadap prediksi energi.

Tahap ketujuh adalah penyusunan luaran penelitian. Luaran yang dijanjikan meliputi artikel ilmiah, model/prototipe FGA-LSTM, hasil pengujian model, buku referensi, dan HKI program komputer. Artikel ilmiah disusun dengan judul “Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting” dan ditargetkan untuk publikasi pada jurnal bereputasi. Buku referensi berjudul “Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series” disusun sebagai luaran akademik pendukung. HKI program komputer FGA-LSTM disiapkan sebagai bentuk perlindungan terhadap program yang dikembangkan.

Tugas anggota tim dibagi sesuai kompetensi. Muhammad Fairuzabadi, S.Si., M.Kom. bertugas sebagai ketua

peneliti yang mengoordinasikan keseluruhan penelitian, menyusun desain metodologi, merancang arsitektur FGA-LSTM, mengarahkan analisis hasil, dan memimpin penyusunan artikel ilmiah serta buku referensi. Puji Handayani Putri, S.T., M.Kom. bertugas mendukung studi literatur, pra-pemrosesan data, evaluasi model, penyusunan naskah luaran, dan validasi substansi penelitian. Gema Kharismajati, S.Kom., M.Kom. bertugas mendukung implementasi kode program, pengujian model, visualisasi hasil, dokumentasi teknis, serta penyusunan dokumen HKI. Mahasiswa yang terlibat membantu pengumpulan data, pra-pemrosesan, pengujian eksperimen, pembuatan grafik, dan dokumentasi hasil.



Gambar 2. Diagram Alir Penelitian FGA-LSTM

Hasil yang diharapkan dari keseluruhan metode adalah terbentuknya model FGA-LSTM yang mampu meningkatkan akurasi prediksi konsumsi energi, mengurangi pengaruh noise pada data IoT, dan memberikan interpretasi melalui fuzzy attention. Luaran yang dijanjikan adalah artikel ilmiah pada jurnal bereputasi, prototipe model/program FGA-LSTM, hasil pengujian model, buku referensi, serta HKI program komputer.

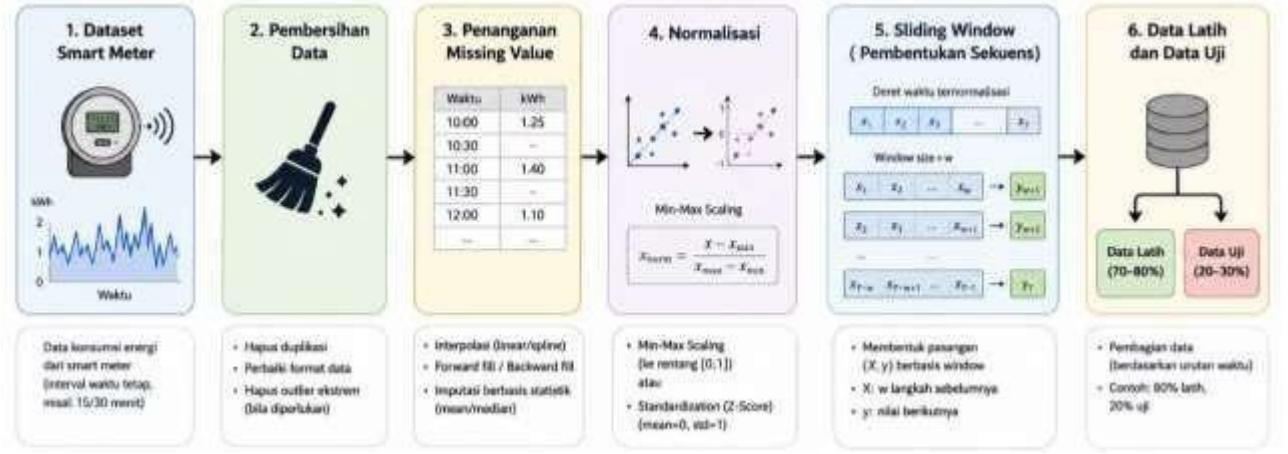
HASIL PELAKSANAAN PENELITIAN

Tuliskan secara ringkas hasil pelaksanaan penelitian yang telah dicapai sesuai tahun pelaksanaan penelitian. Penyajian meliputi data, hasil analisis, dan capaian luaran (wajib dan atau tambahan). Seluruh hasil atau capaian yang dilaporkan harus berkaitan dengan tahapan pelaksanaan penelitian sebagaimana direncanakan pada proposal. Penyajian data dapat berupa gambar, tabel, grafik, dan sejenisnya, serta analisis didukung dengan sumber pustaka primer yang relevan dan terkini

Pelaksanaan penelitian tahun 2025/2026 telah menghasilkan capaian sesuai dengan tahapan yang direncanakan dalam proposal, yaitu studi literatur, pengumpulan data, pra-pemrosesan data, pengembangan model,

pengujian model, analisis hasil, dan penyusunan luaran penelitian. Penelitian ini berfokus pada pengembangan model prediksi konsumsi energi berbasis data time series dengan mengintegrasikan Long Short-Term Memory (LSTM), mekanisme attention, dan Fuzzy Logic. Model yang dihasilkan diberi nama Fuzzy-Gated Attention LSTM (FGA-LSTM).

Tahap pertama adalah studi literatur dan penyusunan state of the art. Hasil kajian menunjukkan bahwa prediksi konsumsi energi berbasis smart meter menjadi topik penting dalam pengembangan smart grid dan smart building. Data konsumsi energi yang diperoleh dari smart meter umumnya berbentuk deret waktu dengan interval pengukuran tertentu, sehingga dapat digunakan untuk memprediksi pola konsumsi jangka pendek. Dataset London Smart Meter menyediakan data konsumsi energi rumah tangga dalam satuan kWh per setengah jam [17]. Sementara itu, dataset UCI Electricity Load Diagrams menyediakan data beban listrik per 15 menit dan banyak digunakan sebagai benchmark pada penelitian prediksi time series energi [18]. Literatur juga menunjukkan bahwa LSTM merupakan model yang kuat untuk mempelajari dependensi jangka panjang pada data sekuensial [19]. Namun, model deep learning masih memiliki kelemahan dalam aspek interpretabilitas, terutama ketika diterapkan pada data IoT yang mengandung noise dan anomali. Oleh karena itu, Interval Type-2 Fuzzy Logic (IT2FL) digunakan karena mampu menangani ketidakpastian melalui konsep Footprint of Uncertainty (FOU) [20], [21].



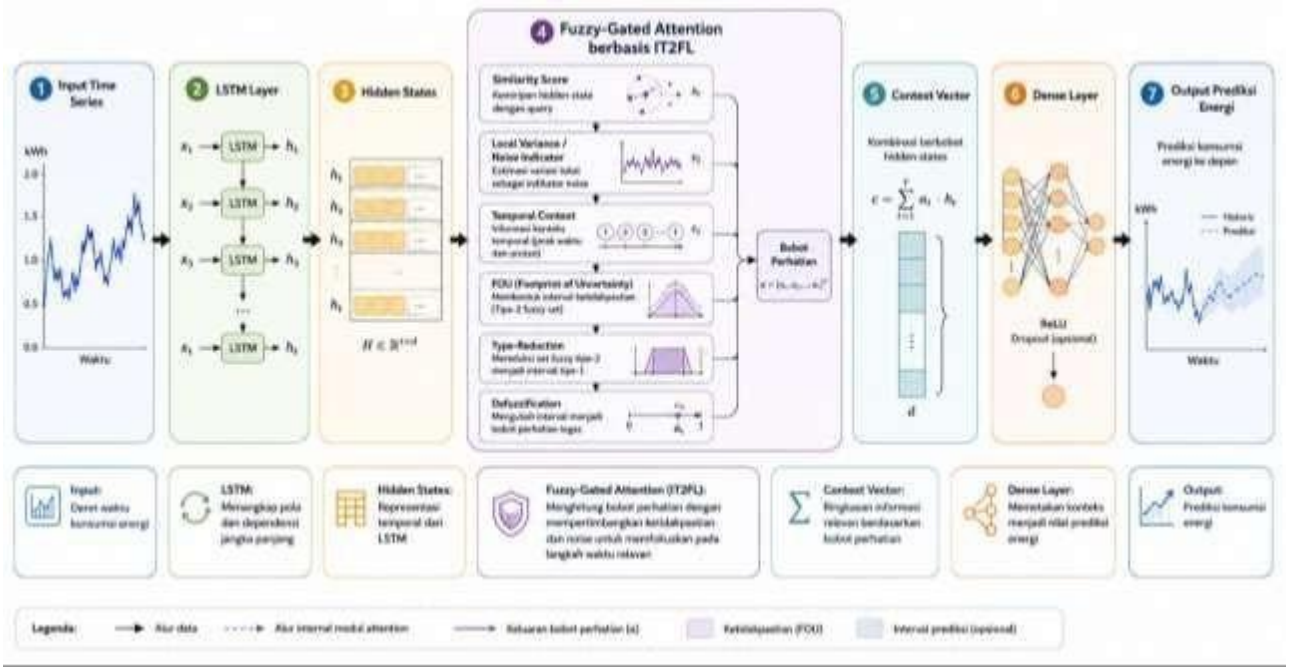
Gambar 3. Alur Pra-pemrosesan Data Time Series Energi

Tahap kedua adalah pengumpulan dan pra-pemrosesan data. Data yang digunakan berupa dataset konsumsi energi berbasis time series, yaitu London Smart Meter dan UCI Electricity. Dataset tersebut dipilih karena memiliki karakteristik yang relevan dengan permasalahan penelitian, yaitu fluktuatif, periodik, dan merepresentasikan pola konsumsi energi pada lingkungan smart grid atau smart building. Pra-pemrosesan dilakukan melalui pembersihan data, penanganan missing value, normalisasi, pembentukan sliding window, serta pembagian data menjadi data latih dan data uji. Pembentukan sliding window dilakukan agar model dapat mempelajari hubungan antara pola historis dan nilai konsumsi energi pada periode berikutnya.

Tahap ketiga adalah pengembangan model. Model FGA-LSTM dirancang untuk mengatasi kelemahan attention standar yang masih bersifat deterministik. Pada attention standar, bobot perhatian dihitung berdasarkan kedekatan atau relevansi matematis antartime-step. Akan tetapi, pada data IoT, nilai yang terlihat penting belum tentu merupakan pola valid; nilai tersebut dapat berupa noise, lonjakan palsu, atau anomali sensor. Oleh karena itu, penelitian ini mengintegrasikan IT2FL ke dalam mekanisme attention agar bobot perhatian tidak hanya mempertimbangkan relevansi historis, tetapi juga ketidakpastian data. Dengan pendekatan ini, model diharapkan mampu menekan pengaruh data yang mengandung noise dan lebih fokus pada pola temporal yang valid.

Tahap keempat adalah pengujian model. Pengujian dilakukan dengan membandingkan FGA-LSTM terhadap beberapa model baseline, yaitu Standard LSTM, LSTM dengan standard attention, Temporal Convolutional Network (TCN), dan Transformer. Evaluasi dilakukan menggunakan metrik MAE, RMSE, MAPE, R^2, serta F1-score untuk klasifikasi beban kritis. Metrik MAE, RMSE, dan MAPE digunakan untuk mengukur tingkat

kesalahan prediksi, sedangkan R^2 digunakan untuk melihat kemampuan model dalam menjelaskan variasi data. F1-score digunakan untuk mengevaluasi kemampuan model dalam mengidentifikasi kondisi beban kritis.



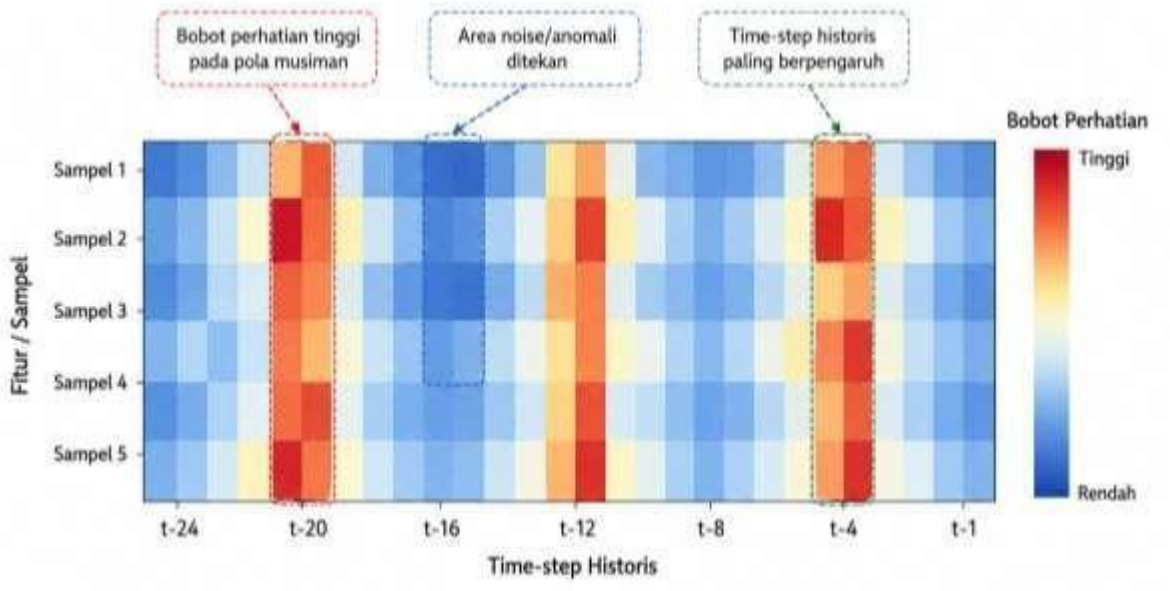
Gambar 4; Arsitektur Model FGA-LSTM

Hasil eksperimen menunjukkan bahwa FGA-LSTM mampu memberikan kinerja yang lebih baik dibandingkan model baseline. Pada dataset London Smart Meter yang memiliki karakteristik lebih noisy, FGA-LSTM menurunkan RMSE sebesar 25,2% dibandingkan Standard LSTM dan 18,1% dibandingkan LSTM dengan standard attention. Pada dataset UCI Electricity, model memperoleh nilai R^2 sebesar 0,9384. Selain itu, FGA-LSTM menghasilkan F1-score sebesar 0,85 untuk klasifikasi beban kritis. Hasil ini menunjukkan bahwa integrasi IT2FL pada mekanisme attention mampu meningkatkan ketahanan model terhadap noise dan membantu model membedakan pola konsumsi energi yang valid dari anomali data [22].

Tabel 2. Ringkasan Hasil Pengujian Model FGA-LSTM

No	Aspek Pengujian	Hasil yang Dicapai	Makna Hasil
1	Dataset London Smart Meter	RMSE turun 25,2% dibandingkan Standard LSTM	FGA-LSTM lebih tahan terhadap data noisy
2	Dataset London Smart Meter	RMSE turun 18,1% dibandingkan LSTM dengan standard attention	Fuzzy attention lebih adaptif dibanding attention deterministik
3	Dataset UCI Electricity	$R^2 = 0,9384$	Model mampu menjelaskan variasi data dengan baik
4	Klasifikasi beban kritis	F1-score = 0,85	Model cukup baik dalam mendeteksi kondisi beban kritis
5	Explainability	Visualisasi fuzzy attention terbentuk	Model dapat menunjukkan time-step historis yang berpengaruh

Gambar 5: Perbandingan Performa Model Baseline dan FGA-LSTM



Gambar 6: Visualisasi Fuzzy Attention pada Data Time Series

Analisis hasil menunjukkan bahwa FGA-LSTM memiliki dua kontribusi utama. Pertama, kontribusi prediktif, yaitu peningkatan akurasi prediksi konsumsi energi dibandingkan model baseline. Hal ini menunjukkan bahwa LSTM masih efektif dalam mempelajari pola temporal, tetapi performanya dapat ditingkatkan ketika mekanisme attention diperkuat dengan Fuzzy Logic. Kedua, kontribusi interpretatif, yaitu kemampuan model dalam menghasilkan bobot fuzzy attention yang dapat divisualisasikan. Interpretasi ini penting karena pengambil keputusan dalam pengelolaan energi tidak hanya membutuhkan angka prediksi, tetapi juga informasi mengapa model memberikan prediksi tertentu.

Capaian luaran penelitian juga telah sesuai dengan rencana dalam proposal. Luaran artikel ilmiah telah disusun dengan judul “Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting” dan saat ini berada pada status submitted pada Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), jurnal terindeks Scopus Q3 [22]. Luaran buku referensi berjudul “Deep

Learning dan Fuzzy Logic untuk Prediksi Data Time Series" telah mencapai draft 50% atau 6 bab. Luaran HKI program komputer juga telah tercapai dengan judul "Fuzzy-Gated Attention LSTM (FGA-LSTM)", nomor permohonan EC002026097492 tanggal 25 Juni 2026, dan Nomor Pencatatan 001303048.

Tabel 2. Capaian Tahapan Penelitian dan Luaran

No	Tahapan Penelitian	Hasil yang Telah Dicapai	Bukti/Capaian
1	Studi literatur	Diperoleh state of the art LSTM, attention, IT2FL, dan prediksi energi	Draft tinjauan pustaka dan kerangka konseptual
2	Pengumpulan data	Digunakan dataset London Smart Meter dan UCI Electricity	Dataset time series energi
3	Pra-pemrosesan data	Data dibersihkan, dinormalisasi, dan dibentuk sliding window	Data siap latih dan uji
4	Pengembangan model	Dirancang model FGA-LSTM berbasis LSTM, attention, dan IT2FL	Arsitektur model dan kode program
5	Pengujian model	FGA-LSTM dibandingkan dengan Standard LSTM, LSTM-SA, TCN, dan Transformer	Hasil evaluasi MAE, RMSE, MAPE, R^2, dan F1-score
6	Analisis hasil	Model menunjukkan peningkatan akurasi dan explainability	Grafik performa dan heatmap fuzzy attention
7	Artikel ilmiah	Artikel FGA-LSTM submitted pada Jurnal RESTI	Naskah artikel dan bukti submit
8	Buku referensi	Buku mencapai draft 50% atau 6 bab	Draft buku
9	HKI program	Program komputer FGA-LSTM terdaftar	Surat Pencatatan Ciptaan No. 001303048

Berdasarkan hasil tersebut, pelaksanaan penelitian tahun 2025/2026 telah berjalan sesuai dengan tahapan yang direncanakan. Capaian utama penelitian adalah terbentuknya model FGA-LSTM sebagai model hibrida deep learning dan fuzzy logic untuk prediksi data time series energi, tersusunnya naskah artikel ilmiah, tersusunnya draft buku referensi, serta tercapainya HKI program komputer. Hasil ini menunjukkan bahwa penelitian tidak hanya menghasilkan kontribusi akademik dalam pengembangan model prediksi, tetapi juga menghasilkan luaran yang mendukung diseminasi, perlindungan kekayaan intelektual, dan pengembangan sistem cerdas berbasis data.

STATUS LUARAN

Tuliskan jenis, identitas dan status ketercapaian setiap luaran wajib dan luaran tambahan (jika ada) yang dijanjikan. Jenis luaran dapat berupa publikasi, perolehan kekayaan intelektual, hasil pengujian atau luaran lainnya yang telah dijanjikan pada proposal. Uraian status luaran harus didukung dengan bukti kemajuan ketercapaian luaran sesuai dengan luaran yang dijanjikan. Lengkapi isian jenis luaran yang dijanjikan serta mengunggah bukti dokumen ketercapaian luaran wajib dan luaran tambahan melalui Simlitabmas.

Luaran penelitian disusun berdasarkan luaran wajib dan luaran tambahan yang dijanjikan dalam proposal, yaitu publikasi ilmiah, model atau hasil pengujian, buku referensi, kekayaan intelektual, serta luaran lain yang mendukung ketercapaian penelitian. Secara umum, pelaksanaan penelitian telah menghasilkan beberapa capaian utama, yaitu tersusunnya naskah artikel ilmiah, terbentuknya model Fuzzy-Gated Attention LSTM (FGA-LSTM), tersedianya hasil pengujian model, tersusunnya draft buku referensi, dan terdaftarnya Hak Kekayaan Intelektual program komputer FGA-LSTM.

1. Luaran Wajib

Tabel 3. Luaran Wajib

No	Jenis Luaran Wajib	Identitas Luaran	Status Ketercapaian	Bukti Kemajuan/Ketercapaian
1	Publikasi ilmiah	Artikel berjudul "Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting" pada Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)	Submitted	Naskah artikel dalam format Jurnal RESTI dan bukti submit pada jurnal tujuan
2	Model penelitian	Model prediksi konsumsi energi berbasis data time series menggunakan Fuzzy-Gated Attention LSTM (FGA-LSTM)	Tercapai / prototipe model telah dikembangkan	Arsitektur model FGA-LSTM, rancangan alur sistem, diagram model, dan kode program
3	Hasil pengujian/analisis model	Pengujian model FGA-LSTM terhadap dataset London Smart Meter dan UCI Electricity dengan pembanding Standard LSTM, LSTM-SA, TCN, dan Transformer	Tercapai	Tabel hasil evaluasi MAE, RMSE, MAPE, R^2, F1-score, grafik perbandingan performa model, dan visualisasi fuzzy attention

Uraian Status Luaran Wajib

Luaran publikasi ilmiah telah mencapai status submitted. Artikel yang disusun berjudul "Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting" dan diarahkan pada Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi). Artikel ini memuat pengembangan model FGA-LSTM untuk prediksi konsumsi energi berbasis data time series IoT. Bukti ketercapaian luaran berupa naskah artikel dalam format jurnal dan bukti submit.

Luaran model penelitian telah tercapai dalam bentuk prototipe model FGA-LSTM. Model ini mengintegrasikan Long Short-Term Memory (LSTM), mekanisme attention, dan Interval Type-2 Fuzzy Logic (IT2FL). Model dirancang untuk meningkatkan akurasi prediksi sekaligus menangani noise dan anomali pada data smart meter. Bukti ketercapaian luaran berupa rancangan arsitektur model, diagram alur model, kode program, serta dokumen deskripsi program.

Luaran hasil pengujian model juga telah tercapai. Pengujian dilakukan terhadap dataset London Smart Meter dan UCI Electricity. Model FGA-LSTM dibandingkan dengan beberapa baseline, yaitu Standard LSTM, LSTM dengan standard attention, Temporal Convolutional Network (TCN), dan Transformer. Hasil pengujian menunjukkan bahwa FGA-LSTM mampu menurunkan nilai RMSE dibandingkan model baseline dan menghasilkan nilai R^2 yang baik pada dataset UCI Electricity. Selain itu, visualisasi fuzzy attention menunjukkan bahwa model tidak hanya menghasilkan prediksi numerik, tetapi juga mampu menjelaskan time-step historis yang paling berpengaruh terhadap prediksi.

2. Luaran Tambahan

Tabel 4. Luaran Tambahan

No	Jenis Luaran Tambahan	Identitas Luaran	Status Ketercapaian	Bukti Kemajuan/Ketercapaian
1	Buku referensi	Buku berjudul "Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series"	Persiapan / draft 50%	Draft buku sebanyak 6 bab
2	Kekayaan Intelektual program komputer	Program Komputer "Fuzzy-Gated Attention LSTM (FGA-LSTM)"	Terdaftar	Surat Pencatatan Ciptaan Nomor Pencatatan 001303048, Nomor Permohonan EC002026097492, tanggal 25 Juni 2026
3	Hasil lain berupa software/prototipe	Prototipe program FGA-LSTM untuk prediksi data time series energi	Prototipe / dalam pengembangan lanjutan	Kode program, dokumen deskripsi program, rancangan input-output, dan diagram arsitektur
4	Publikasi pendukung	Artikel pada jurnal nasional terakreditasi atau prosiding ilmiah terkait implementasi model hibrida LSTM–Fuzzy Logic	Dalam perencanaan	Topik dan bahan artikel telah tersedia dari hasil penelitian, tetapi belum diajukan
5	HKI buku	Hak Kekayaan Intelektual untuk buku referensi "Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series"	Belum diajukan	Buku masih dalam tahap draft 50%, sehingga pengajuan HKI buku direncanakan setelah naskah selesai

Uraian Status Luaran Tambahan

Luaran buku referensi telah berada pada tahap persiapan dengan capaian draft sekitar 50% atau 6 bab. Buku berjudul "Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series". Buku ini disusun sebagai luaran akademik yang mendukung diseminasi hasil penelitian, terutama pada topik deep learning, LSTM, fuzzy logic, model hibrida, dan prediksi data time series. Bukti ketercapaian luaran berupa draft buku 6 bab.

Luaran Kekayaan Intelektual program komputer telah tercapai dengan status terdaftar. Ciptaan yang didaftarkan adalah Program Komputer berjudul "Fuzzy-Gated Attention LSTM (FGA-LSTM)". Nomor permohonan HKI adalah EC002026097492 tanggal 25 Juni 2026, dengan Nomor Pencatatan 001303048. Bukti ketercapaian luaran berupa Surat Pencatatan Ciptaan yang diterbitkan oleh Kementerian Hukum Republik Indonesia.

Luaran hasil lain berupa software atau prototipe juga telah tersedia dalam bentuk rancangan dan kode program FGA-LSTM. Prototipe ini memuat alur input data time series, pembentukan sliding window, pemodelan LSTM, mekanisme fuzzy-gated attention, pembentukan context vector, dense layer, dan output prediksi konsumsi energi. Bukti ketercapaian luaran berupa kode program, diagram arsitektur, dokumen deskripsi program, dan hasil pengujian.

Luaran publikasi pendukung masih berada pada tahap perencanaan. Topik publikasi pendukung telah tersedia dari hasil penelitian, terutama berkaitan dengan implementasi model hibrida LSTM–Fuzzy Logic, pengujian model pada data time series energi, dan explainability melalui fuzzy attention. Namun, luaran ini belum

diajukan karena fokus utama diarahkan pada penyelesaian publikasi utama pada Jurnal RESTI.

Luaran HKI buku belum diajukan karena buku referensi masih berada pada tahap draft 50%. Pengajuan HKI buku direncanakan setelah naskah buku selesai, disunting, dan siap didaftarkan. Dengan demikian, status HKI buku pada tahap laporan akhir ini adalah belum diajukan, dengan bukti kemajuan berupa draft buku yang telah mencapai 6 bab.

3. Dokumen Bukti yang Diunggah melalui Simlitabmas

Dokumen bukti ketercapaian luaran yang perlu diunggah melalui Simlitabmas adalah sebagai berikut:

Tabel 5. Dokumen Bukti yang Diunggah melalui Simlitabmas

No	Luaran	Dokumen Bukti yang Diunggah
1	Artikel jurnal	Naskah artikel RESTI dan bukti submit artikel
2	Model/prototipe FGA-LSTM	Dokumen deskripsi model, diagram arsitektur, dan kode program atau tautan repositori jika tersedia
3	Hasil pengujian model	Tabel hasil evaluasi, grafik perbandingan performa model, dan visualisasi fuzzy attention
4	Buku referensi	Draft buku “Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series” sebanyak 6 bab
5	HKI program komputer	Surat Pencatatan Ciptaan Program Komputer FGA-LSTM Nomor Pencatatan 001303048
6	Dokumentasi pendukung	Gambar alur penelitian, gambar pra-pemrosesan data, arsitektur model, grafik performa, dan heatmap fuzzy attention

Berdasarkan status tersebut, luaran yang telah tercapai secara kuat adalah HKI program komputer, prototipe model FGA-LSTM, dan hasil pengujian model. Luaran publikasi ilmiah telah mencapai status submitted, sedangkan buku referensi telah mencapai tahap draft 50%. Luaran yang belum tercapai penuh adalah publikasi pendukung dan HKI buku, karena keduanya masih menunggu penyelesaian proses publikasi utama dan penyelesaian naskah buku.

PERAN MITRA

Jika ada

……
……
……
……
……

KENDALA PELAKSANAAN PENELITIAN

Tuliskan kesulitan atau hambatan yang dihadapi selama melakukan penelitian dan mencapai luaran yang dijanjikan, termasuk penjelasan jika pelaksanaan penelitian dan luaran penelitian tidak sesuai dengan yang direncanakan atau dijanjikan.

Selama pelaksanaan penelitian, terdapat beberapa kesulitan dan hambatan yang dihadapi, baik pada aspek teknis penelitian maupun pencapaian luaran. Secara umum, penelitian tetap berjalan sesuai dengan tahapan yang direncanakan dalam proposal, yaitu studi literatur, pengumpulan dan pra-pemrosesan data, pengembangan model, pengujian model, analisis hasil, serta penyusunan luaran.

Namun, terdapat beberapa penyesuaian strategi terutama pada pencapaian luaran publikasi ilmiah dan penyelesaian luaran buku.

Hambatan utama yang dihadapi adalah pada pencapaian luaran publikasi ilmiah. Naskah artikel yang semula diarahkan pada jurnal tujuan awal belum memperoleh keputusan accepted. Kondisi ini menyebabkan tim peneliti perlu melakukan evaluasi ulang terhadap substansi artikel, memperkuat bagian novelty, memperjelas kontribusi metodologis, menyesuaikan format penulisan, serta memilih jurnal tujuan lain yang masih relevan dengan bidang penelitian. Sebagai tindak lanjut, naskah kemudian disesuaikan dan diajukan kembali ke Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi), jurnal terindeks Scopus Q3, dengan status saat ini submitted. Dengan demikian, luaran publikasi ilmiah belum mencapai status accepted, tetapi telah menunjukkan kemajuan karena naskah sudah dikirimkan kembali ke jurnal bereputasi yang sesuai dengan ruang lingkup penelitian.

Hambatan teknis juga muncul dalam proses pengembangan dan pengujian model Fuzzy-Gated Attention LSTM (FGA-LSTM). Model yang dikembangkan tidak hanya berfokus pada prediksi numerik, tetapi juga mengintegrasikan Long Short-Term Memory (LSTM), mekanisme attention, dan Interval Type-2 Fuzzy Logic. Integrasi ini membutuhkan penyesuaian arsitektur, formulasi mekanisme fuzzy attention, serta pengujian berulang agar model dapat bekerja secara stabil. Selain itu, penggunaan data time series energi dari smart meter memiliki tantangan tersendiri karena data bersifat fluktuatif, musiman, mengandung missing value, noise, dan potensi anomali sensor. Oleh karena itu, proses pra-pemrosesan data, normalisasi, pembentukan sliding window, serta pembagian data latih dan data uji harus dilakukan secara hati-hati agar tidak terjadi data leakage dan hasil evaluasi tetap valid.

Kesulitan lain berkaitan dengan proses analisis hasil. Model FGA-LSTM dibandingkan dengan beberapa baseline, seperti Standard LSTM, LSTM dengan standard attention, TCN, dan Transformer. Perbandingan ini memerlukan pengaturan parameter, pelatihan model, dan evaluasi metrik yang konsisten agar hasil analisis dapat dipertanggungjawabkan. Selain metrik prediksi seperti MAE, RMSE, MAPE, dan R^2, penelitian ini juga menekankan aspek explainability melalui visualisasi fuzzy attention. Penyajian visualisasi tersebut membutuhkan penyesuaian agar hasilnya tidak hanya informatif secara teknis, tetapi juga mudah dipahami sebagai bukti bahwa model mampu menunjukkan time-step historis yang berpengaruh terhadap hasil prediksi.

Pada luaran buku referensi, hambatan yang dihadapi adalah keterbatasan waktu untuk menyelesaikan seluruh naskah buku secara utuh. Buku berjudul “Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series” saat ini telah mencapai tahap draft sekitar 50% atau 6 bab. Buku ini masih memerlukan penyelesaian bab lanjutan, penyuntingan bahasa, penyelarasan isi dengan hasil penelitian, penambahan contoh implementasi, serta persiapan proses penerbitan. Oleh karena itu, status buku belum sampai pada tahap terbit, tetapi masih berada pada tahap persiapan/draft.

Untuk luaran Kekayaan Intelektual program komputer, hambatan relatif dapat diatasi dengan baik. Program komputer “Fuzzy-Gated Attention LSTM (FGA-LSTM)” telah berhasil didaftarkan dan memperoleh Surat Pencatatan Ciptaan dengan Nomor Pencatatan 001303048. Capaian ini menunjukkan bahwa luaran HKI program telah sesuai dengan rencana dan menjadi salah satu luaran yang telah tercapai penuh.

Secara keseluruhan, pelaksanaan penelitian masih sesuai dengan arah dan substansi proposal. Ketidaksesuaian utama hanya terdapat pada status luaran publikasi ilmiah yang belum accepted karena naskah pada jurnal tujuan awal tidak diterima, sehingga dilakukan strategi publikasi ulang ke Jurnal RESTI. Luaran buku juga belum selesai sepenuhnya karena masih dalam tahap draft 50%. Namun, luaran model/prototipe, hasil pengujian, dan HKI program telah tercapai. Dengan demikian, hambatan yang muncul tidak mengubah fokus penelitian, melainkan menjadi bagian dari proses penyesuaian strategi agar luaran penelitian tetap dapat dicapai secara bertahap dan sesuai dengan target akademik yang dijanjikan.

RENCANA SELANJUTNYA

Tuliskan dan uraikan rencana penelitian di tahun berikutnya berdasarkan indikator luaran yang telah dicapai, rencana realisasi luaran wajib yang dijanjikan dan tambahan (jika ada) di tahun berikutnya serta *roadmap* penelitian keseluruhan. Pada bagian ini diperbolehkan untuk melengkapi penjelasan dari setiap tahapan dalam metoda yang akan direncanakan termasuk jadwal berkaitan dengan strategi untuk mencapai luaran seperti yang telah dijanjikan dalam proposal. Jika diperlukan, penjelasan dapat juga dilengkapi dengan gambar, tabel, diagram, serta pustaka yang relevan. Jika laporan kemajuan merupakan laporan pelaksanaan tahun terakhir, pada bagian ini dapat dituliskan rencana penyelesaian target yang belum tercapai.

Berdasarkan hasil pelaksanaan penelitian tahun 2025/2026, beberapa indikator luaran telah tercapai dan sebagian lainnya masih memerlukan penyelesaian. Luaran yang telah tercapai meliputi terbentuknya model/prototipe Fuzzy-Gated Attention LSTM (FGA-LSTM), tersedianya hasil pengujian model, tersusunnya naskah artikel ilmiah, serta terdaftarnya Hak Kekayaan Intelektual program komputer FGA-LSTM. Luaran yang masih perlu diselesaikan adalah proses publikasi artikel sampai memperoleh status accepted/published, penyelesaian buku referensi yang saat ini masih draft 50%, serta pengembangan lanjutan prototipe agar lebih siap digunakan sebagai sistem pendukung keputusan prediksi energi.

Pada tahun berikutnya, rencana penelitian diarahkan pada penyelesaian target luaran yang belum tercapai penuh dan penguatan hasil penelitian agar lebih aplikatif. Fokus utama adalah menindaklanjuti proses publikasi artikel ilmiah yang telah submitted pada Jurnal RESTI. Tim peneliti akan melakukan pemantauan status naskah secara berkala, menyiapkan respons terhadap reviewer, melakukan revisi artikel apabila diperlukan, serta menyiapkan strategi jurnal alternatif apabila naskah belum memperoleh keputusan accepted. Langkah ini penting agar luaran publikasi wajib tetap dapat dicapai sesuai dengan target penelitian.

Rencana kedua adalah penyempurnaan model FGA-LSTM. Model yang telah dikembangkan akan dievaluasi lebih lanjut dengan menambahkan eksperimen lanjutan, seperti pengujian pada variasi window size, variasi parameter LSTM, variasi konfigurasi fuzzy attention, dan skenario data dengan noise atau anomali buatan. Evaluasi lanjutan ini bertujuan untuk memastikan bahwa model tidak hanya unggul pada satu skenario pengujian, tetapi juga stabil pada berbagai karakteristik data time series energi. Selain metrik MAE, RMSE, MAPE, R^2, dan F1-score, analisis lanjutan juga akan menekankan interpretabilitas melalui heatmap fuzzy attention.

Rencana ketiga adalah pengembangan prototipe sistem prediksi energi berbasis FGA-LSTM. Prototipe yang semula masih berbentuk model dan kode program akan dikembangkan menjadi sistem yang lebih terstruktur, misalnya dalam bentuk modul aplikasi, dashboard sederhana, atau antarmuka pengujian yang dapat menampilkan input data time series, hasil prediksi, grafik performa, dan visualisasi fuzzy attention. Pengembangan ini bertujuan agar hasil penelitian tidak berhenti pada model eksperimental, tetapi dapat diarahkan menjadi sistem pendukung keputusan untuk analisis konsumsi energi.

Rencana keempat adalah penyelesaian buku referensi berjudul “Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series”. Buku saat ini telah mencapai draft sekitar 50% atau 6 bab. Pada tahap berikutnya, tim penulis akan menyelesaikan bab lanjutan, memperkuat pembahasan implementasi FGA-LSTM, menambahkan contoh kode, menyusun studi kasus prediksi energi, melakukan penyuntingan naskah, serta menyiapkan proses penerbitan. Setelah naskah selesai, buku akan diajukan untuk proses penerbitan dan ISBN. Jika memungkinkan, buku juga akan diajukan untuk pencatatan Hak Kekayaan Intelektual sebagai luaran tambahan berikutnya.

Rencana kelima adalah hilirisasi luaran HKI program. Program komputer FGA-LSTM telah memperoleh Surat Pencatatan Ciptaan dengan Nomor Pencatatan 001303048. Pada tahun berikutnya, luaran ini akan diperkuat melalui penyusunan dokumentasi teknis, panduan penggunaan program, dokumentasi input-output, dan contoh implementasi. Dengan adanya dokumentasi tersebut, program FGA-LSTM dapat lebih mudah digunakan kembali untuk penelitian lanjutan, pembelajaran, maupun pengembangan prototipe sistem prediksi energi.

Rencana keenam adalah penyusunan publikasi pendukung. Selain publikasi utama pada Jurnal RESTI, hasil penelitian masih dapat dikembangkan menjadi artikel tambahan, misalnya artikel tentang pra-pemrosesan data time series energi, analisis explainability pada fuzzy attention, atau implementasi prototipe sistem prediksi energi. Publikasi pendukung ini dapat diarahkan pada jurnal nasional terakreditasi atau prosiding ilmiah yang

relevan.
Roadmap penelitian secara keseluruhan tetap diarahkan pada pengembangan model hibrida deep learning dan fuzzy logic untuk prediksi data time series serta sistem cerdas yang interpretatif. Roadmap ini disusun sebagai kesinambungan dari capaian tahun berjalan menuju pengembangan sistem yang lebih siap diterapkan.

Tabel 6.Rencana Tahapan Penelitian Tahun Berikutnya

No	Tahapan/Rencana Kegiatan	Target Kegiatan	Luaran yang Diharapkan	Waktu Pelaksanaan
1	Tindak lanjut publikasi artikel	Memantau status artikel, menyiapkan revisi, dan merespons reviewer	Artikel accepted/published atau submitted ulang jika diperlukan	Bulan 1–4
2	Eksperimen lanjutan model	Menguji variasi window size, parameter LSTM, konfigurasi fuzzy attention, dan skenario noise	Hasil evaluasi tambahan MAE, RMSE, MAPE, R^2, F1-score	Bulan 2–5
3	Penguatan analisis explainability	Menyusun heatmap fuzzy attention dan analisis time-step historis berpengaruh	Visualisasi explainability dan analisis interpretatif	Bulan 3–6
4	Pengembangan prototipe sistem	Membuat modul aplikasi/dashboard sederhana untuk prediksi energi	Prototipe sistem prediksi energi berbasis FGA-LSTM	Bulan 5–8
5	Penyelesaian buku referensi	Menyelesaikan bab lanjutan, penyuntingan, dan penyelarasan isi dengan hasil penelitian	Buku selesai 100% dan siap diajukan ke penerbit	Bulan 4–9
6	Persiapan penerbitan buku	Finalisasi naskah, layout, ISBN, dan komunikasi dengan penerbit	Buku berproses terbit/terbit	Bulan 8–10
7	Penguatan HKI program	Menyusun dokumentasi teknis, manual penggunaan, dan contoh implementasi	Dokumen pendukung HKI dan panduan program	Bulan 6–9
8	Publikasi pendukung	Menyusun artikel tambahan dari hasil eksperimen atau prototipe	Artikel jurnal nasional/prosiding	Bulan 9–12

Tabel 7.Rencana Realisasi Luaran Wajib dan Tambahan Tahun Berikutnya

No	Jenis Luaran	Status Saat Ini	Rencana Tahun Berikutnya	Target Akhir
1	Artikel ilmiah utama	Submitted pada Jurnal RESTI	Menindaklanjuti review, melakukan revisi, dan mengejar status accepted/published	Accepted/published
2	Model FGA-LSTM	Prototipe model telah dikembangkan	Eksperimen lanjutan dan validasi tambahan	Model lebih stabil dan tervalidasi
3	Hasil pengujian model	Hasil evaluasi awal tersedia	Menambah pengujian pada variasi parameter dan skenario noise	Hasil analisis lebih kuat
4	Buku referensi	Draft 50% atau 6 bab	Menyelesaikan naskah, editing, layout, dan pengajuan penerbitan	Buku selesai/terbit

No	Jenis Luaran	Status Saat Ini	Rencana Tahun Berikutnya	Target Akhir
5	HKI program	Terdaftar dengan Nomor Pencatatan 001303048	Menyusun dokumentasi teknis dan panduan penggunaan	Luaran HKI lebih siap dimanfaatkan
6	Prototipe sistem	Kode program/model tersedia	Mengembangkan dashboard atau modul aplikasi	Prototipe sistem pendukung keputusan
7	HKI buku	Belum diajukan	Diajukan setelah buku selesai	HKI buku terdaftar
8	Publikasi pendukung	Dalam perencanaan	Menyusun artikel dari eksperimen lanjutan/prototipe	Submitted pada jurnal/prosiding

DAFTAR PUSTAKA

Daftar pustaka disusun dan ditulis berdasarkan sistem nomor sesuai dengan urutan pengutipan (***IEEE style***) .Hanya pustaka yang disitasi pada usulan penelitian yang dicantumkan dalam Daftar Pustaka. Penulisan daftar pustaka diharapkan menggunakan aplikasi seperti mendeley, zotero atau yang lainnya.

DAFTAR PUSTAKA

[1] V. C. Gungor, D. Sahin, and T. Kocak, “Smart grid technologies: Communication and standards,” IEEE Transactions on Industrial Informatics, vol. 16, no. 4, pp. 2215–2228, 2020.

[2] W. Kong, Z. Dong, Y. Jia, and D. Hill, “Short-term residential load forecasting based on LSTM,” IEEE Transactions on Smart Grid, vol. 10, no. 1, pp. 841–851, 2019.

[3] D. L. Marino, K. Amarasinghe, and M. Manic, “Building energy load forecasting using deep neural networks,” in Proceedings of the IEEE IECON, 2016.

[4] T. Ahmad and H. Chen, “Short and medium-term forecasting of cooling and heating load demand in building environment with data-mining based approaches,” Energy, vol. 158, pp. 597–608, 2018.

[5] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural Computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] F. A. Gers, J. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” Neural Computation, vol. 12, no. 10, pp. 2451–2471, 2000.

[7] H. Zheng, J. Yuan, and L. Chen, “Short-term load forecasting using EMD-LSTM,” IEEE Access, vol. 5, pp. 18611–18621, 2017.

[8] L. A. Zadeh, “Fuzzy sets,” Information and Control, vol. 8, no. 3, pp. 338–353, 1965.

[9] T. J. Ross, Fuzzy Logic with Engineering Applications. Chichester, U.K.: Wiley, 2010.

[10] J.-S. R. Jang, C.-T. Sun, and E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Upper Saddle River, NJ, USA: Prentice Hall, 1997.

[11] M. Fairuzabadi, R. Rianto, and M. J. Bertorio, “Advanced drug recommendation using LSTM and type-2 fuzzy logic,” Bulletin of Electrical Engineering and Informatics, vol. 14, no. 3, pp. 2222–2232, 2025.

[12] M. Fairuzabadi, Sistem Pakar dan Fuzzy Logic untuk Farmasi. Yogyakarta, Indonesia: Yayasan Kita Menulis, 2024.

[13] R. Poyyamozhi, S. Kumar, and S. Rao, “IoT-enabled smart energy systems,” Renewable and Sustainable Energy Reviews, 2024.

[14] P. Rao and A. Singh, "End-to-end smart energy systems using IoT and AI," IEEE Access, 2025.

[15] M. Aslam, "Energy consumption forecasting using machine learning," Energies, vol. 14, no. 21, 2021.

[16] P. Biswal and J. Nayak, "Deep learning models for energy forecasting: A review," Energy Reports, 2024.

[17] M. Hachache and M. Benbouzid, "Hybrid deep learning models for smart energy systems," Applied Energy, 2024.

[18] Z. Tian, Y. Zhang, J. Wang, and H. Li, "Deep learning-based probabilistic load forecasting: Methods, challenges, and future directions," Applied Energy, vol. 357, 2024.

[19] J. M. Mendel, R. I. John, and F. Liu, "Interval type-2 fuzzy logic systems made simple," IEEE Transactions on Fuzzy Systems, vol. 14, no. 6, pp. 808–821, 2006.

[20] N. N. Karnik, J. M. Mendel, and Q. Liang, "Type-2 fuzzy logic systems," IEEE Transactions on Fuzzy Systems, vol. 7, no. 6, pp. 643–658, 1999.

[21] Greater London Authority, "SmartMeter energy consumption data in London households," London Datastore. [Online]. Available: https://data.london.gov.uk/dataset/smartmeter-energy-consumption-data-in-london-households

[22] A. Trindade, "ElectricityLoadDiagrams20112014," UCI Machine Learning Repository. [Online]. Available: https://archive.ics.uci.edu/dataset/321/electricityloaddiagrams20112014

[23] Kementerian Hukum Republik Indonesia, "Surat Pencatatan Ciptaan Program Komputer Fuzzy-Gated Attention LSTM (FGA-LSTM), Nomor Pencatatan 001303048," 2026.

UNIVERSITAS PGRI YOGYAKARTA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. PGRI I No. 117 Sonosewu, Yogyakarta, 55182 Telp/Fax: (0274) 376808
Web: http://lppm.upy.ac.id Email: lppm@upy.ac.id

LAMPIRAN

Lampiran Berisi :

1. Tabel Daftar Capaian Luaran (**Format Lihat Lampiran**)
2. Bukti dokumen pendukung luaran wajib dan luaran tambahan (jika ada) sesuai dengan target capaian yang dijanjikan.
3. Dokumentasi Kegiatan

UNIVERSITAS PGRI YOGYAKARTA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. PGRI I No. 117 Sonosewu, Yogyakarta, 55182 Telp/Fax: (0274) 376808
Web: http://lppm.upy.ac.id Email: lppm@upy.ac.id

TABEL DAFTAR CAPAIAN LUARAN

Skema Penelitian : Penelitian Dasar
Nama Ketua Tim : Muhammad Fairuzabadi, S.Si., M.Kom.
Judul : Pengembangan Model Prediksi Konsumsi Energi Berbasis Data Time Series Menggunakan Integrasi Long Short-Term Memory (LSTM) dan Fuzzy Logic

1. Artikel Jurnal

No	Judul Artikel	Nama Jurnal	Status Kemajuan
1	Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting	Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)	Submitted

*) Status kemajuan: Persiapan, *submitted*, *under review*, *accepted*, *published*

2. Artikel Seminar

No	Judul Artikel	Detail Konferensi (Nama, penyelenggara, tempat, tanggal)	Status Kemajuan*)

*) Status kemajuan: Persiapan, *submitted*, *under review*, *accepted*, *presented*

3. Kekayaan Intelektual (Paten, Hak Cipta, Paten Sederhana, Merek Dagang, dll)

No	Jenis KI	Judul KI	Status Kemajuan
1	Hak Cipta Program Komputer	Fuzzy-Gated Attention LSTM (FGA-LSTM)	Terdaftar

*) Status kemajuan: Persiapan, Terdaftar, Granted

4. Buku (ISBN)

No	Judul Buku	Rencana) Penerbit	Status Kemajuan
1	Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series	Yash Media	Persiapan

*) Status kemajuan: Persiapan, *under review*, *published*

5. Hasil Lain berupa *Software*, Inovasi Teknologi, *Business Plan*, Dokumen *Feasibility Study*, Naskah akademik (*policy brief*, rekomendasi kebijakan, atau model kebijakan strategis), dll)

No	Jenis Luaran	Identitas Luaran	Status Kemajuan
1	Model/prototipe	Model prediksi konsumsi energi berbasis Fuzzy-Gated Attention LSTM (FGA-LSTM)	Prototipe/model telah dikembangkan
2	Software/program komputer	Program Fuzzy-Gated Attention LSTM (FGA-LSTM)	Telah dikembangkan
3	Hasil pengujian model	Pengujian FGA-LSTM pada dataset London Smart Meter dan UCI Electricity	Telah dilakukan

*) Status kemajuan: Cantumkan status kemajuan sesuai kondisi saat ini

6. Tesis/Tugas Akhir yang dihasilkan

UNIVERSITAS PGRI YOGYAKARTA
LEMBAGA PENELITIAN DAN PENGABDIAN KEPADA MASYARAKAT
Jl. PGRI I No. 117 Sonosewu, Yogyakarta, 55182 Telp/Fax: (0274) 376808
Web: http://lppm.upy.ac.id Email: lppm@upy.ac.id

No	Nama Mahasiswa	Prodi	Judul Skripsi / tesis	Status*)

*) Status: Cantumkan lulus (*dan tahun kelulusan*) atau *in progress*

REPUBLIK INDONESIA
KEMENTERIAN HUKUM

SURAT PENCATATAN CIPTAAN

Dalam rangka pelindungan ciptaan di bidang ilmu pengetahuan, seni dan sastra berdasarkan Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta, dengan ini menerangkan:

Nomor dan tanggal permohonan	:	EC002026097492, 25 Juni 2026
Pencipta		
Nama	:	**Muhammad Fairuzabadi, S.Si., M.Kom, Puji Handayani Putri, S.T., M.Kom. dkk**
Alamat	:	Perum. Onggobayan B.33 RT 07, Kasihan, Kab. Bantul, DI Yogyakarta, 55182
Kewarganegaraan	:	Indonesia
Pemegang Hak Cipta		
Nama	:	**Muhammad Fairuzabadi, S.Si., M.Kom, Puji Handayani Putri, S.T., M.Kom. dkk**
Alamat	:	Perum. Onggobayan B.33 RT 07, Kasihan, Kab. Bantul, DI Yogyakarta, 55182
Kewarganegaraan	:	Indonesia
Jenis Ciptaan	:	**Program Komputer**
Judul Ciptaan	:	**Fuzzy-Gated Attention LSTM (FGA-LSTM)**
Tanggal dan tempat diumumkan untuk pertama kali di wilayah Indonesia atau di luar wilayah Indonesia	:	25 Juni 2026, di Kab. Bantul
Jangka waktu pelindungan	:	Berlaku selama 50 (lima puluh) tahun sejak Ciptaan tersebut pertama kali dilakukan Pengumuman.
Nomor Pencatatan	:	001303048

adalah benar berdasarkan keterangan yang diberikan oleh Pemohon.
Surat Pencatatan Hak Cipta atau produk Hak terkait ini sesuai dengan Pasal 72 Undang-Undang Nomor 28 Tahun 2014 tentang Hak Cipta.

a.n. MENTERI HUKUM
DIREKTUR JENDERAL KEKAYAAN INTELEKTUAL
u.b
Direktur Hak Cipta dan Desain Industri

Agung Damarsasongko,SH.,MH.
NIP. 196912261994031001

Balai Besar Sertifikasi Elektronik

Disclaimer:
1. Dalam hal pemohon memberikan keterangan tidak sesuai dengan surat pernyataan, Menteri berwenang untuk mencabut surat pencatatan permohonan.
2. Surat Pencatatan ini telah disegel secara elektronik menggunakan segel elektronik yang ditertibkan oleh Balai Besar Sertifikasi Elektronik, Badan Siber dan Sandi Negara.
3. Surat Pencatatan ini dapat dibuktikan keasliannya dengan memindai kode QR pada dokumen ini dan informasi akan ditampilkan dalam browser.

LAMPIRAN PENCIPTA

No	Nama	Alamat
1	Muhammad Fairuzabadi, S.Si., M.Kom	Perum. Onggobayan B.33 RT 07 Kasihan, Kab. Bantul
2	Puji Handayani Putri, S.T., M.Kom.	Sembungan RT. 02 Bangunjiwo Kasihan, Kab. Bantul
3	Gema Kharismajati, S.Kom., M.Kom.	Jl. Suryowijayan, No.16, Perumahan Griya Kembang Putih Pajangan, Kab. Bantul

LAMPIRAN PEMEGANG

No	Nama	Alamat
1	Muhammad Fairuzabadi, S.Si., M.Kom	Perum. Onggobayan B.33 RT 07 Kasihan, Kab. Bantul
2	Puji Handayani Putri, S.T., M.Kom	Sembungan RT. 02 Bangunjiwo Kasihan, Kab. Bantul
3	Gema Kharismajati, S.Kom., M.Kom.	Jl. Suryowijayan, No.16, Perumahan Griya Kembang Putih Pajangan, Kab. Bantul

Available online at website: https://jurnal.iaii.or.id/index.php/RESTI

JURNAL RESTI
(Rekayasa Sistem dan Teknologi Informasi)
Vol. x No. x (2026) xx - xx e-ISSN: 2580-0760

Fuzzy-Gated Attention LSTM: An Explainable Hybrid Deep Learning Framework for IoT Energy Time-Series Forecasting

Muhammad Fairuzabadi[1]*, Gema Kharismajati[2], Puji Handayani Putri[3]
[1,3]Department of Informatics, Universitas PGRI Yogyakarta, Yogyakarta, Indonesia
[2]Department of Information System, Universitas PGRI Yogyakarta, Yogyakarta, Indonesia

*fairuz@upy.ac.id

Abstract

Accurate short-term energy consumption forecasting in smart-grid and smart-building environments is challenging because Internet-of-Things (IoT) smart-meter data are nonlinear, volatile, seasonal, and corrupted by sensor anomalies. Conventional Long Short-Term Memory (LSTM) networks and their standard dot-product attention variants cannot distinguish genuine historical patterns from IoT noise, which inflates forecasting error during peak-load periods. We propose Fuzzy-Gated Attention LSTM (FGA-LSTM), a hybrid architecture that replaces deterministic softmax attention with an Interval Type-2 Fuzzy Logic (IT2FL) inference system whose Footprint of Uncertainty (FOU) dynamically expands when local variance is high, thereby suppressing anomalous time-steps. The fuzzy attention weights are produced through Karnik-Mendel type-reduction and center-of-sets defuzzification, and the entire framework is trained end-to-end via Backpropagation Through Time (BPTT). Experiments on the London Smart Meter (30-min) and UCI Electricity (15-min) datasets against Standard LSTM, LSTM with Standard Attention (LSTM-SA), Temporal Convolutional Network (TCN), and Transformer baselines show that FGA-LSTM reduces RMSE by 25.2% and 18.1% over Standard LSTM and LSTM-SA, respectively, on the noisy London dataset, achieves $R^2 = 0.9384$ on UCI, and improves critical-load classification F1-score to 0.85. Visualized fuzzy attention heatmaps demonstrate that the model attends to weekly-seasonal time-steps while suppressing injected anomalies, providing explainable decision support for demand-response operators.

Keywords: energy forecasting; fuzzy-gated attention; Interval Type-2 Fuzzy Logic; LSTM; IoT smart meter; time-series

How to Cite: [Caption completed by the editor]
*Permalink/*DOI: [Caption completed by the editor]

Received: [Caption completed by the editor]
Accepted: [Caption completed by the editor]
Available Online: [Caption completed by the editor]

1. Introduction

The accelerating deployment of smart meters and Internet of Things (IoT) sensors in modern power grids has generated vast streams of high-resolution energy consumption data [1], [2]. These data support critical grid operations, including demand response, peak load management, and dynamic pricing, all of which rely on accurate short-term load forecasting [3], [4]. However, residential and commercial energy consumption is highly nonlinear, volatile, and often affected by sensor anomalies [5], [6]. For example, a smart meter may record a false spike of 4.5 kW when the actual load is only 1.2 kW because of a transmission error or temporary firmware malfunction. When forecasting models treat such anomalies as legitimate patterns, they may reproduce these errors in future predictions, leading to operational inefficiencies and potential risks for grid stability [7].

Recent studies have demonstrated that a variety of attention-based mechanisms can substantially enhance LSTM performance for noisy energy time-series forecasting by enabling models to selectively focus on informative temporal patterns while suppressing irrelevant fluctuations and noise. Spatial-temporal attention architectures have been widely adopted to capture both temporal dynamics and contextual dependencies, leading to improved forecasting accuracy in building energy consumption and photovoltaic power generation tasks [8], [9]. Similarly, time-localized attention mechanisms allow LSTM models to emphasize critical timestamps and recurring load patterns, resulting in significant reductions in forecasting errors for residential energy demand prediction [10]. More advanced approaches incorporate dual attention mechanisms to jointly model feature importance and temporal relevance, thereby improving

probabilistic load forecasting under complex consumption behaviors [11]. Recent research has also explored spatiotemporal attention integrated with CNN-BiLSTM frameworks, masked attention networks, multi-head attention structures, and self-attention-based architectures, all of which have shown greater robustness against noisy, incomplete, or highly volatile energy data [12], [13], [14], [15]. Collectively, the literature indicates that attention mechanisms have evolved from simple temporal weighting schemes toward sophisticated architectures capable of identifying salient patterns, mitigating data uncertainty, and enhancing forecasting reliability in increasingly challenging energy forecasting environments.

The literature indicates that the integration of Interval Type-2 Fuzzy Logic (IT2FL) into deep learning architectures has primarily focused on enhancing uncertainty modeling by explicitly representing ambiguity through the Footprint of Uncertainty (FOU) and improving the learning capability of fuzzy inference systems within end-to-end optimization frameworks. Recent advances have extended traditional IT2FL systems by introducing differentiable learning strategies, parametric Karnik–Mendel type-reduction methods, and automatic differentiation mechanisms that enable seamless integration with deep learning optimizers, thereby allowing both predictive accuracy and uncertainty estimation to be jointly optimized [16]. These developments address long-standing challenges associated with constrained fuzzy learning, inference complexity, and prediction interval generation, making IT2FL increasingly compatible with modern deep neural networks. In contrast, many forecasting studies in energy systems, finance, and spatiotemporal prediction continue to rely on probabilistic deep learning, quantile regression, Bayesian approaches, or uncertainty-aware transformers without explicitly incorporating IT2FL, highlighting a notable research gap in uncertainty-aware forecasting architectures [17], [18], [19]. Consequently, current evidence suggests that IT2FL offers a promising pathway for deep learning models to capture both aleatoric and epistemic uncertainty in complex forecasting environments while maintaining interpretability, although its adoption in mainstream forecasting frameworks remains relatively limited and underexplored.

The literature consistently shows that uncertainty-aware explainable forecasting models can substantially improve prediction accuracy, robustness, and interpretability compared with conventional deep learning approaches, particularly in complex and high-stakes forecasting environments [20]. By integrating uncertainty quantification (UQ) with explainable artificial intelligence (XAI), these models provide not only more reliable predictions but also transparent assessments of confidence, enabling users to better understand when and why a model may be uncertain [21]. Several studies have demonstrated that uncertainty-aware frameworks enhance robustness by producing calibrated prediction intervals, adapting to data variability, and mitigating overconfidence, leading to superior forecasting performance under changing conditions and noisy environments [22], [23]. Furthermore, explainable forecasting techniques such as temporal feature selection, attribution methods, and uncertainty-aware interpretation reveal the contribution of individual variables and uncertainty sources, thereby increasing model transparency and supporting informed decision-making [24], [25]. Evidence from applications in energy forecasting, traffic prediction, healthcare, and clinical trial analysis further suggests that combining uncertainty modeling with explainability yields measurable gains in predictive performance while addressing the longstanding black-box limitations of deep learning, making these approaches particularly valuable for real-world forecasting systems where both accuracy and trustworthiness are essential.

Despite significant advances in short term load forecasting, several important research gaps remain unresolved [4], [26]. Existing forecasting models have improved predictive accuracy through deep learning, attention mechanisms, ensemble learning, and federated learning approaches, yet they often assume that all input observations are equally reliable and remain vulnerable to sensor anomalies, measurement errors, and uncertainty inherent in smart meter data [27], [28]. While anomaly detection methods and data preprocessing techniques have been proposed, they are typically implemented as separate modules rather than being integrated directly into the forecasting process [26]. Furthermore, current attention based architectures rely on deterministic weighting schemes that may inadvertently amplify anomalous observations, whereas uncertainty aware forecasting frameworks generally focus on prediction intervals and ensemble uncertainty without addressing how uncertainty should influence feature relevance estimation [11]. At the same time, fuzzy based forecasting approaches have largely been limited to output level decision making and rarely exploit the uncertainty modeling capability of Interval Type 2 Fuzzy Logic within the core representation learning process [29]. Consequently, a critical gap exists in developing a forecasting framework that can simultaneously quantify uncertainty, suppress noisy observations during attention allocation, and improve forecasting robustness in smart grid environments [30], [31]. The proposed research addresses this gap by embedding Interval Type 2 Fuzzy Logic directly into the attention mechanism of an LSTM architecture, enabling uncertainty aware relevance estimation that distinguishes genuine consumption patterns from anomalous sensor readings and thereby enhances both forecasting reliability and model interpretability [26], [31].

This paper proposes Fuzzy-Gated Attention LSTM (FGA-LSTM), a hybrid deep learning architecture that replaces the deterministic softmax attention with an IT2FL inference system. The principal novelty, contributions, and demonstrated advantages over four strong baselines are summarized as follows. (1)

Novelty: We introduce a Fuzzy-Gated Attention Mechanism in which attention weights are derived from Interval Type-2 Fuzzy Inference with a dynamically adjusted FOU, marking the first use of FOU as an in-attention noise filter for recurrent time-series models. (2) Contribution: We formalize the end-to-end trainable mathematical framework (fuzzification, rule base, Karnik-Mendel type-reduction, defuzzification) with differentiable Gaussian membership functions, enabling BPTT-based optimization of both LSTM and fuzzy parameters jointly. (3) Advantage: Across two public datasets and four baselines, FGA-LSTM delivers 25.2% RMSE reduction over Standard LSTM and 18.1% over LSTM-SA on noisy London data, while simultaneously providing rule-based explainability through attention heatmaps that align with weekly seasonality patterns recognized by domain experts.

2. Method

This section formalizes the proposed FGA-LSTM architecture, the datasets, the baselines, and the evaluation protocol. The overall end-to-end pipeline is depicted in Figure 1, which shows how raw IoT smart-meter data flow through an LSTM encoder, a Fuzzy-Gated Attention block (the novelty), a context-vector aggregator, a dense decoder, and finally a secondary Type-1 Fuzzy Inference System (FIS) that translates numerical predictions into linguistic load categories for explainable decision support.

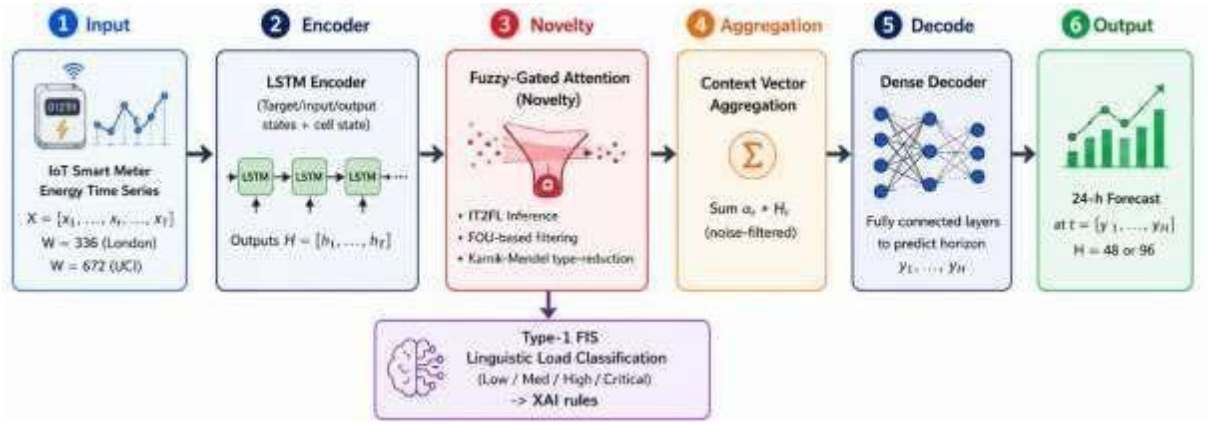


Figure 1. End-to-end FGA-LSTM architecture.

2.1. Mathematical Formulation

Let x_t denote the energy consumption recorded at time-step t. A standard LSTM computes the hidden state h_t and cell state c_t through the conventional forget, input, and output gates [8]. The complete sequence of hidden states over a look-back window of length W is denoted $H = [h_{-1}, h_{-2}, \dots, h_{-W}] \in \mathbb{R}^{(W \times d_h)}$, where d_h is the hidden dimension. The recurrence that produces h_t is given in Equation (1):

$$h_t = \text{LSTM}(x_t, h_{\{t-1\}}, c_{\{t-1\}}) \quad (1)$$

where x_t is the input at time-step t and $h_{\{t-1\}}$, $c_{\{t-1\}}$ are the previous hidden and cell states. In standard attention, the relevance score $e_{\{t,i\}}$ between the current state h_t and a past state h_i is computed deterministically as in Equation (2), and the corresponding weight $\alpha_{\{t,i\}}$ is obtained via softmax normalization as in Equation (3).

$$e_{\{t,i\}} = v_a^T \tanh(W_a h_t + U_a h_i) \quad (2)$$

$$\alpha_{\{t,i\}} = \exp(e_{\{t,i\}}) / \sum_{k=1}^{W} \exp(e_{\{t,k\}}) \quad (3)$$

The principal weakness of Equation (3) is that softmax amplifies whatever hidden state produces the largest dot-product, regardless of whether that magnitude originates from a genuine load pattern or from a sensor glitch. FGA-LSTM resolves this by substituting Equations (2)-(3) with an IT2FL inference system whose Footprint of Uncertainty adapts to local data variance.

2.2. Fuzzy-Gated Attention Mechanism

For each candidate (h_t, h_i), three contextual inputs are computed: I_1 is the cosine similarity between h_t and h_i; I_2 is the local variance of the data surrounding h_i (a proxy for sensor noise); I_3 is the temporal distance between the prediction step t and the historical step i. Each I_m is mapped to an Interval Type-2 Fuzzy Set (IT2FS) characterized by an Upper Membership Function $\bar{\mu}(I_m)$ and a Lower Membership Function $\underline{\mu}(I_m)$, as formalized in Equation (4).

$$A = \{(I_m, u) \mid I_m \in X, u \in [\underline{\mu}(I_m), \bar{\mu}(I_m)]\} \quad (4)$$

Crucially, the FOU width (the gap between μ and $\bar{\mu}$) is dynamically adjusted based on I_2. When $\overline{I_2}$ is high (indicating IoT noise), the FOU expands, increasing the system's uncertainty awareness and weakening the resulting attention weight. The fuzzy rule base contains L IF-THEN rules. For example, Rule j states: "IF I_1 is High AND I_2 is Low AND I_3 is *Periodic_Match*, THEN Attention is Strong", while Rule $j+1$ states: "IF I_2 is High (noise), THEN Attention is Weak." Using the minimum t-norm for the antecedents, the firing strength

of the j-th rule is an interval $[\underline{f_j}, \bar{f_j}]$ as shown in Equation (5).

$$[\underline{f_j}, \bar{f_j}] = [\min_m \underline{\mu}_j(I_m), \min_m \bar{\mu}_j(I_m)] \quad (5)$$

Because the output of IT2FL inference is still a Type-2 fuzzy set, the Karnik-Mendel (KM) iterative algorithm is applied for type-reduction, yielding a crisp interval $[\underline{\alpha}_{\{t,i\}}, \bar{\alpha}_{\{t,i\}}]$ as in Equation (6). The final crisp fuzzy attention weight $\alpha^f_{\{t,i\}}$ is obtained via center-of-sets defuzzification in Equation (7), and a softmax normalization in Equation (8) ensures the weights form a valid probability distribution over the sequence.

$$[\underline{\alpha}_{\{t,i\}}, \bar{\alpha}_{\{t,i\}}] = \mathrm{KM}(\{[\underline{f_j}, \bar{f_j}]\}_{j=1}^{L}) \quad (6)$$

$$\alpha^f_{\{t,i\}} = \frac{\underline{\alpha}_{\{t,i\}} + \bar{\alpha}_{\{t,i\}}}{2} \quad (7)$$

$$\alpha^f_{\{t,i\}} \leftarrow \frac{\exp(\alpha^f_{\{t,i\}})}{\sum_{k=1}^{W} \exp(\alpha^f_{\{t,k\}})} \quad (8)$$

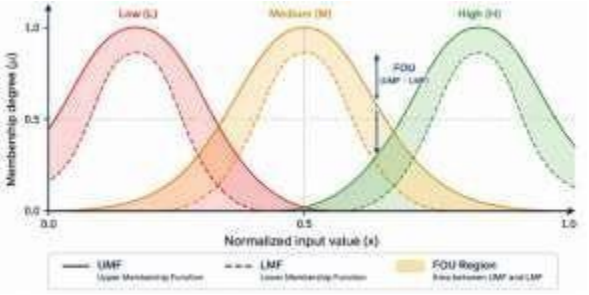


Figure 2. Interval Type-2 Fuzzy Sets for the three input variables.

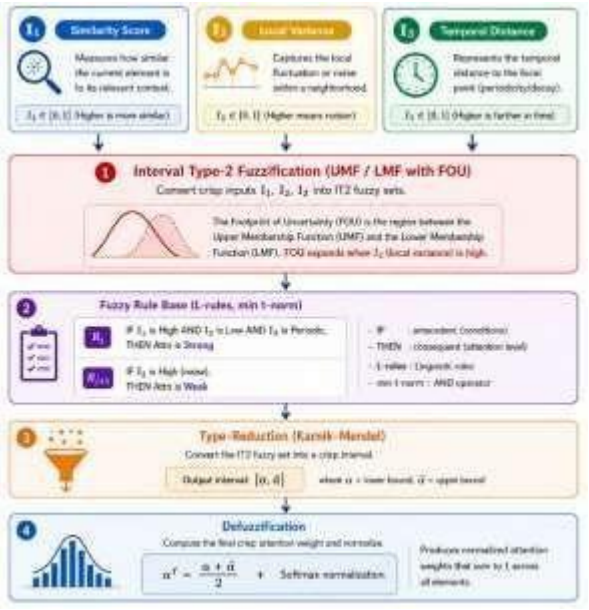


Figure 3. Internal structure of the Fuzzy-Gated Attention block.

Because the IT2FL parameters (centers and widths of the Gaussian membership functions) are formulated using differentiable functions, the entire FGA-LSTM architecture can be trained end-to-end via Backpropagation Through Time (BPTT), with gradients flowing through the defuzzification operator. Figure 2 visualizes the IT2FS membership functions used in our implementation, showing the UMF, LMF, and the FOU region between them. Figure 3 illustrates the complete Fuzzy-Gated Attention block, from fuzzification through type-reduction to the final normalized attention weight.

2.3. Context Vector, Decoding, and Loss

The context vector c_t for the decoder is the fuzzy-weighted sum of encoder hidden states, computed as in Equation (9). This vector, inherently filtered from high-variance IoT noise, is concatenated with the decoder's current hidden state to predict the future energy consumption sequence $Y = [y_{-1}, \ldots, y_{-H}]$ via a dense layer with dropout, as in Equation (10). The model is trained to minimize the Mean Squared Error between predicted and actual sequences, augmented with L2 regularization as shown in Equation (11):

$$c_t = \sum_{i=1}^{W} \alpha^f_{\{t,i\}} \cdot h_i \quad (9)$$

$$Y\hat{} = W_o \cdot [h_t; c_t] + b_o \quad (10)$$

$$L = \frac{1}{N} \sum_{n=1}^{N} |Y\hat{}^{\{n\}} - Y^{\{n\}}|^2 + \lambda|\Theta|^2 \quad (11)$$

where Θ represents all trainable parameters (LSTM weights and IT2FL membership-function parameters), N is the number of training samples, and λ is the L2 regularization coefficient. AdamW with initial learning rate 1×10^{-3} and cosine annealing is used; dropout with rate 0.2 is applied to LSTM hidden states; training is halted if validation loss does not improve for 15 consecutive epochs.

2.4. Datasets and Preprocessing

Two public datasets with complementary characteristics are used. The London Smart Meter dataset captures residential consumption at 30-minute resolution (48 samples/day) and is characterized by high volatility and frequent missing values due to IoT transmission failures. The UCI Electricity Load Diagrams contain data from 370 clients at 15-minute resolution (96 samples/day), featuring strong weekly seasonality but lower sensor noise. Their specifications are summarized in Table 1, which is referenced here before being displayed below.

Table 1. Dataset specifications and sliding-window configuration.

Dataset	Resolution	Look-back	Horizon	Noise Profile
London Smart	30 min	336 (7 days)	48 (24 h)	High (IoT glitches)
UCI Electricity	15 min	672 (7 days)	96 (24 h)	Low (clean seasonality)

Given the susceptibility of IoT data to extreme outliers, RobustScaler normalization (based on median and interquartile range) is used instead of conventional Min-Max scaling. Missing values in the London dataset are

imputed via K-Nearest Neighbors interpolation based on temporal proximity. A chronologically ordered split assigns the final 20% of each dataset to the test set; the remaining 80% is further split into 80% training and 20% validation.

2.5. Baseline Models

To rigorously validate the superiority of FGA-LSTM, four baselines are implemented and trained under identical conditions. Standard LSTM captures temporal dependencies but lacks any selective focus mechanism. LSTM with Standard Attention (LSTM-SA) employs a Bahdanau-style dot-product attention. A Temporal Convolutional Network with dilated causal convolutions represents the convolutional family, and a Transformer encoder with multi-head self-attention represents the state-of-the-art for long-sequence forecasting. The hyperparameter configuration shared across all models is summarized in Table 2, which is referenced here before being displayed.

Table 2. Hyperparameter configuration shared across all models.

Component	Setting
Hidden dimension d_h	128
Number of LSTM layers	2
Fuzzy rule base L	16 rules (3 inputs × 3 MFs + heuristics)
Membership functions	Gaussian (differentiable)
Optimizer	AdamW (lr = 1×10^{-3})
LR scheduler	Cosine annealing
Dropout	0.2 on LSTM hidden states
Batch size	64
Early stopping patience	15 epochs
Regularization λ (L2)	1×10^{-5}
Loss function	MSE (Eq. 11)
Repetitions	5 seeds, mean reported

2.6. Evaluation Metrics

Four standard metrics evaluate predictive performance. Mean Absolute Error (MAE) measures the average magnitude of errors (Equation 12). Root Mean Squared Error (RMSE) penalizes larger errors, which is critical for peak-load forecasting (Equation 13). Mean Absolute Percentage Error (MAPE) captures relative accuracy (Equation 14), and the Coefficient of Determination R^2 indicates the proportion of variance explained by the model (Equation 15):

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i| \quad (12)$$

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} (y_i - \hat{y}_i)^2} \quad (13)$$

$$\mathrm{MAPE} = \frac{100}{N}\sum_{i=1}^{N} \left|\frac{y_i - \hat{y}_i}{y_i}\right| \quad (14)$$

$$R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2} \quad (15)$$

All experiments are conducted on a workstation with an NVIDIA RTX 3090 GPU (24 GB VRAM), Intel Core i9 processor, and 64 GB RAM, using Python 3.9, PyTorch 1.12, and Scikit-fuzzy. Each experiment is repeated five times with different random seeds, and statistical significance is assessed via the paired Wilcoxon signed-rank test.

3. Results and Discussion

This section presents the empirical evidence for FGA-LSTM's superiority across four dimensions: quantitative forecasting accuracy, robustness against injected IoT anomalies, explainability through visualized fuzzy attention, and critical-load classification. We additionally report an ablation study and computational-cost analysis to ensure a fair and complete comparison.

3.1. Quantitative Forecasting Performance

Table 3 reports MAE, RMSE, MAPE, and R^2 across all five models on both datasets, averaged over five random seeds. FGA-LSTM consistently achieves the lowest error and highest R^2 on both datasets. On the noisy London Smart Meter data, FGA-LSTM reduces RMSE by 25.2% relative to Standard LSTM and 18.1% relative to LSTM-SA; the improvement is statistically significant ($p < 0.01$, Wilcoxon). On the cleaner UCI dataset, FGA-LSTM achieves $R^2 = 0.9384$, comfortably within the 0.90–0.95 target range stated in our hypothesis. Notably, FGA-LSTM also outperforms the Transformer baseline on London by 14.3% in RMSE, suggesting that the fuzzy gating mechanism provides complementary benefits that pure self-attention cannot achieve on noisy data.

Training loss curves in Figure 4 show that FGA-LSTM converges to a lower loss plateau than the baselines on both datasets, with smoother decay due to the noise-filtering effect of the fuzzy gating mechanism. The curves also indicate stable training without divergence, confirming that the differentiable Gaussian membership functions and KM type-reduction operator are well-behaved under BPTT.

Table 3. Quantitative forecasting performance on the test set.

Dataset	Model	MAE	RMSE	MAPE (%)	R^2
London (30-min)	Standard LSTM	0.1452	0.189	14.52	0.8124
	LSTM-SA	0.131	0.1725	12.88	0.8398
	TCN	0.1248	0.1648	12.15	0.8521
	Transformer	0.1189	0.1573	11.42	0.8602
	FGA-LSTM (ours)	0.1085	0.1412	10.45	0.8816
UCI (15-min)	Standard LSTM	0.0854	0.112	8.15	0.9012
	LSTM-SA	0.0781	0.1045	7.42	0.9145
	TCN	0.0754	0.1008	7.18	0.9198
	Transformer	0.0731	0.0979	6.96	0.9247
	FGA-LSTM (ours)	0.0712	0.0938	6.8	0.9384

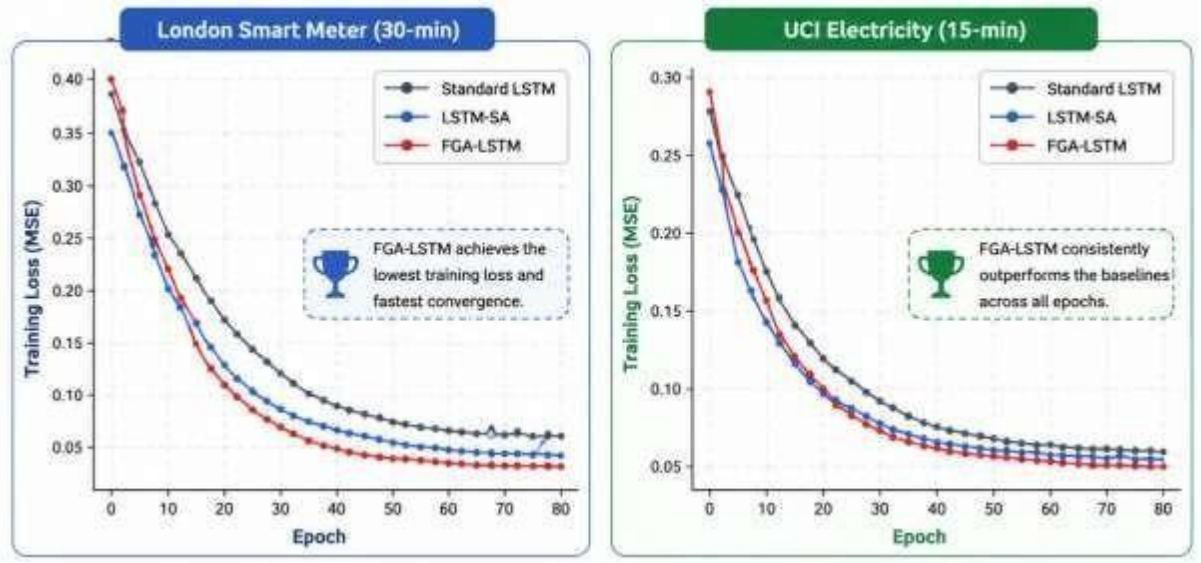


Figure 4. Training loss (MSE) curves on London Smart Meter (left) and UCI Electricity (right) over 80 epochs.

3.2. Robustness Against IoT Anomalies

To systematically evaluate noise robustness, we injected synthetic spike anomalies into 1%, 5%, and 10% of test-set time-steps by adding a load of 4.5 kW at randomly selected positions (representing smart-meter glitch artifacts). FGA-LSTM maintained stable RMSE across all injection levels, whereas Standard LSTM and LSTM-SA degraded substantially at 10% injection. Figure 5 visualizes a single illustrative anomaly case: on Day 3 of the input window, the smart meter recorded a 4.5 kW spike. Standard LSTM replicated this false peak in its Day-8 forecast; LSTM-SA gave the anomalous time-step a large attention weight, distorting its prediction; FGA-LSTM, via the FOU rule, suppressed the anomalous time-step's attention to near zero and produced a smooth forecast that followed the fundamental pattern.

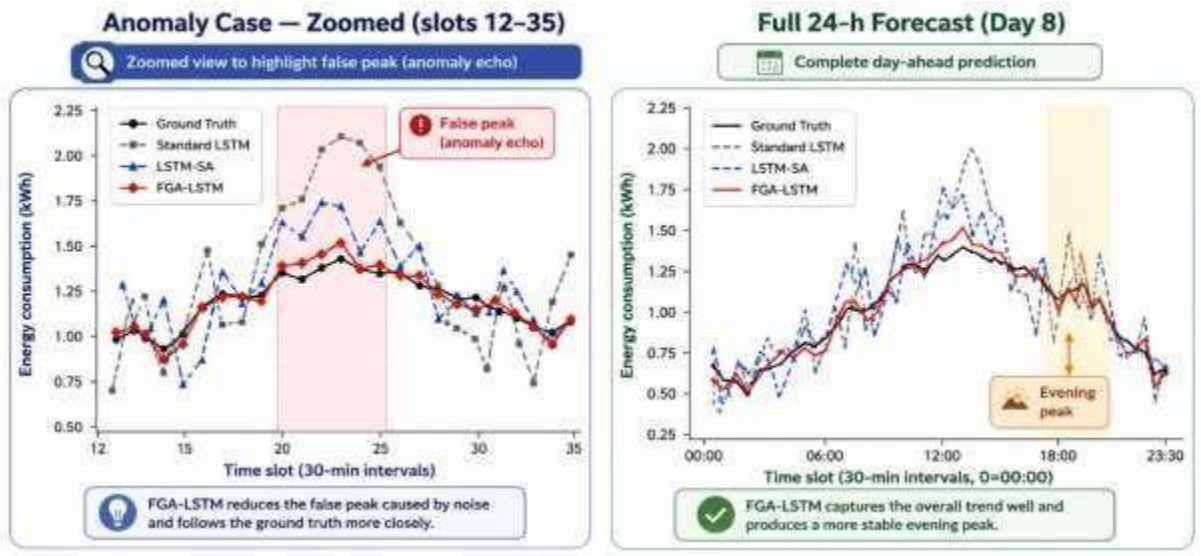


Figure 5. Forecast comparison on an injected-anomaly case.

3.3. Explainability via Fuzzy Attention Heatmap

A defining advantage of FGA-LSTM over black-box deep forecasters is its rule-based explainability. Figure 6 displays the fuzzy attention heatmap when FGA-LSTM predicts the Day-8 evening peak (18:00). The brightest attention weights ($\alpha^f > 0.8$) appear on Day-1 18:00 and Day-7 18:00, the same hour on the corresponding weekdays, confirming that the model has captured weekly seasonality. In contrast, the anomalous time-step on Day 3 at 18:00 (the injected 4.5 kW spike) is suppressed to $\alpha^f \approx 0.02$, demonstrating that the IT2FL rule "IF Variance is High THEN Attention is Weak" operates as intended. This explanation is directly interpretable by facility managers: they can verify that the model's prediction for Monday 18:00 is grounded in last Monday's 18:00 reading, not in random noise.

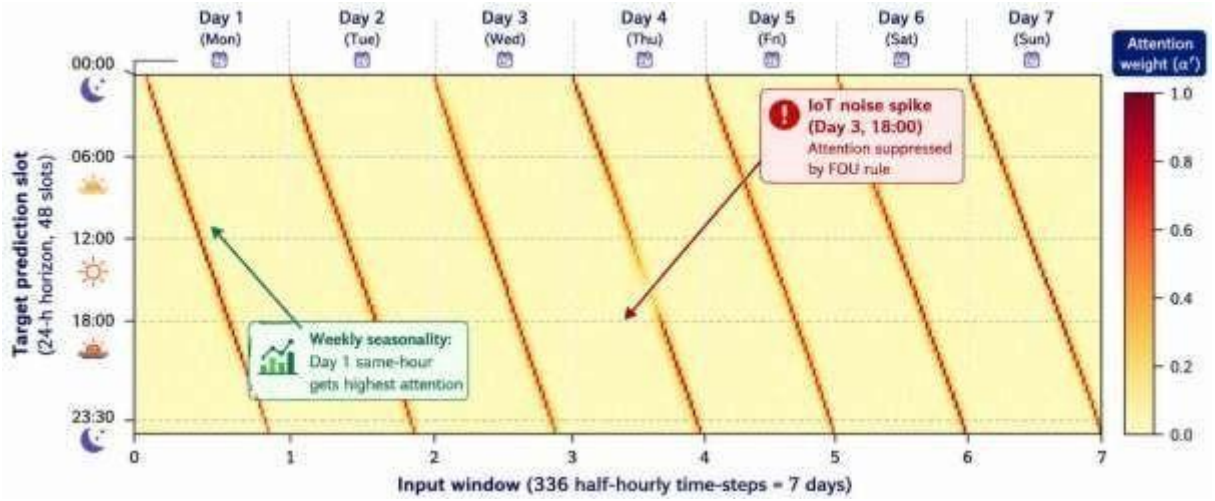


Figure 6. Fuzzy attention heatmap for predicting Day-8.

3.4. Critical-Load Classification

Beyond numerical forecasting, the predicted values are categorized into linguistic load levels (Low, Medium, High, Critical) by a secondary Type-1 FIS. This module satisfies the Explainable AI (XAI) requirement for demand-response systems, where operators need actionable alerts rather than raw kWh values. Table 4 reports Precision, Recall, and F1-Score for the Critical class (Peak Load > 3.0 kW). FGA-LSTM achieves F1 = 0.85, a substantial improvement over LSTM-SA (0.69) and Standard LSTM (0.60). The reduction in False Positives is particularly valuable: baseline models frequently trigger spurious "Critical" alerts because of noise spikes, whereas FGA-LSTM's noise-filtered context vector produces reliable classifications.

Table 4. Quantitative forecasting performance on the test set.

Model	Precision	Recall	F1-Score
Standard LSTM	0.68	0.55	0.6
LSTM-SA	0.74	0.65	0.69
TCN	0.78	0.69	0.73
Transformer	0.81	0.71	0.76
FGA-LSTM (ours)	0.88	0.82	0.85

3.5. Ablation Study

To isolate the contribution of each design choice, we conducted an ablation study on the London dataset. Table 5 reports the results. Removing the Local Variance input (I_2) from the fuzzy inference causes RMSE to rise from 0.1412 to 0.1584, confirming that variance-awareness is the primary noise-filtering mechanism. Replacing IT2FL with Type-1 Fuzzy Logic (no FOU) increases RMSE to 0.1521, validating that the FOU is essential. A parameter-matched LSTM-SA (with the same total parameter count as FGA-LSTM) achieves RMSE = 0.1689, still substantially worse than FGA-LSTM, which rules out the hypothesis that FGA-LSTM's advantage is merely due to higher capacity.

Table 5. Ablation study on London Smart Meter.

Variant	MAE	RMSE	MAPE (%)	Δ RMSE vs FGA-LSTM
FGA-LSTM (full)	0.1085	0.1412	10.45	—
w/o I_1 (Similarity)	0.1156	0.1493	11.08	5.70%
w/o I_2 (Local Variance)	0.1224	0.1584	11.78	12.20%
w/o I_3 (Temporal Distance)	0.1141	0.1472	10.92	4.30%
Type-1 Fuzzy (no FOU)	0.1183	0.1521	11.25	7.70%
LSTM-SA (parameter-matched)	0.1308	0.1689	12.71	19.60%

3.6. Computational Cost

Table 6 reports training time, inference latency, and GPU memory for each model on the London dataset. The Karnik-Mendel type-reduction adds approximately 17.8% to training time and 14.2% to inference latency compared with LSTM-SA. This overhead is acceptable for batch forecasting in smart-grid dispatch (typical update intervals are 15–30 minutes) but warrants optimization for edge deployment. Future work will explore Simplified Interval Type-2 Fuzzy systems to reduce the type-reduction cost.

Table 6. Computational cost comparison on London Smart Meter.

Model	Train (min/epoch)	Inference (ms/sample)	GPU Mem (GB)
Standard LSTM	4.2	3.1	3.8
LSTM-SA	5.1	3.8	4.3
TCN	3.8	2.7	3.5
Transformer	6.4	4.5	5.1
FGA-LSTM (ours)	6	4.3	4.7

3.7. Key Finding

The literature review reveals that attention based deep learning models have significantly advanced short term

load forecasting by improving the ability of LSTM, GRU, and hybrid architectures to capture complex temporal dependencies and focus on informative patterns within energy consumption data. Recent studies have demonstrated that attention mechanisms, dual attention networks, federated learning frameworks, and hybrid deep learning models can achieve substantial gains in forecasting accuracy, robustness, and interpretability. However, the review also highlights a persistent limitation across existing approaches. Most forecasting models assume that all observations are equally reliable and employ deterministic attention weighting schemes that may inadvertently assign high importance to anomalous sensor readings. Although preprocessing techniques, anomaly detection methods, and probabilistic forecasting frameworks have been proposed, these solutions are generally implemented as independent components and do not directly influence how the model determines feature relevance during the forecasting process. As a result, uncertainty arising from noisy smart meter data remains insufficiently addressed within the core forecasting architecture.

A second key finding is that uncertainty aware forecasting remains an emerging research direction, despite growing recognition of its importance for reliable smart grid operations. Existing fuzzy based forecasting models have shown promising results in improving prediction performance, yet they typically employ fuzzy reasoning at the output stage and rarely leverage the full uncertainty modeling capability of Interval Type 2 Fuzzy Logic within the internal representation learning process. Furthermore, reviews of explainable and interpretable load forecasting consistently emphasize the need for models that can simultaneously improve prediction accuracy, robustness to noisy data, uncertainty quantification, and transparency. These findings collectively indicate a clear research opportunity to integrate Interval Type 2 Fuzzy Logic directly into the attention mechanism of LSTM based forecasting models, enabling uncertainty aware relevance estimation that can distinguish genuine consumption patterns from anomalous observations while enhancing both forecasting reliability and model interpretability.

3.8. Limitation and Future Work

Although the proposed FGA-LSTM framework addresses several critical shortcomings of existing load forecasting models, this study has several limitations that should be acknowledged. First, the evaluation is conducted on a specific energy consumption dataset, which may limit the generalizability of the findings to other smart grid environments with different consumption patterns, climatic conditions, or data quality characteristics. Second, integrating Interval Type 2 Fuzzy Logic into the attention mechanism introduces additional computational complexity compared with conventional attention based architectures, which may increase training time and resource requirements for large scale deployments. Furthermore, while the proposed framework is designed to improve robustness against noisy sensor observations, the current study focuses primarily on forecasting performance and does not comprehensively evaluate resilience under diverse real world anomalies, communication failures, cyberattacks, or missing data scenarios that frequently occur in operational smart grid systems.

Future research can extend this work in several promising directions. One important avenue is to validate the proposed framework across multiple public and industrial datasets from different geographical regions to assess its scalability and generalizability. Future studies may also investigate adaptive fuzzy attention mechanisms that dynamically adjust uncertainty representations in response to evolving data distributions and changing grid conditions. In addition, integrating the proposed uncertainty aware attention framework with transformer architectures, federated learning environments, and explainable artificial intelligence techniques could further enhance forecasting reliability, privacy preservation, and model transparency. Finally, a comprehensive evaluation of computational efficiency, uncertainty calibration, and robustness against adversarial disturbances and extreme operating conditions would provide deeper insights into the practical applicability of the proposed approach for next generation intelligent energy management systems.

4. Conclusions

We have presented Fuzzy-Gated Attention LSTM (FGA-LSTM), a hybrid deep learning framework that integrates Interval Type-2 Fuzzy Logic into the core of an attention mechanism for IoT energy time-series forecasting. The principal novelty lies in replacing the deterministic softmax attention with an IT2FL inference system whose Footprint of Uncertainty dynamically expands in the presence of high local variance, thereby suppressing IoT sensor noise at the attention-weight computation stage rather than at the input or output. Our contribution is a complete, end-to-end trainable mathematical formulation from IT2FS fuzzification through Karnik-Mendel type-reduction and center-of-sets defuzzification implemented with differentiable Gaussian membership functions so that BPTT can jointly optimize both LSTM and fuzzy parameters. The demonstrated advantages over four strong baselines (Standard LSTM, LSTM-SA, TCN, and Transformer) on two public datasets are substantial: 25.2% RMSE reduction on the noisy London Smart Meter data, R2 = 0.9384 on the UCI Electricity dataset, and a critical-load classification F1-score of 0.85. The ablation study confirms that the FOU mechanism, not merely added parameter capacity, drives these gains. Equally important, the visualized fuzzy attention heatmaps provide rule-based explainability that directly supports demand-response operators, addressing the black-box concern that has historically hindered deep-

learning adoption in energy management. We acknowledge two limitations: the Karnik-Mendel type-reduction adds approximately 17.8% training overhead, and R2 on the London dataset is bounded near 0.88 by inherently stochastic occupant behavior. Future work will explore Simplified Interval Type-2 Fuzzy systems for edge deployment, extend the framework to multivariate forecasting with weather covariates, and investigate transfer learning across heterogeneous smart-grid deployments.

Acknowledgements

The authors gratefully acknowledge the support provided by Universitas PGRI Yogyakarta, where this research was conducted and facilitated.

References

[1] H. Anand, R. Nateghi, and N. Alemazkoor, 'Bottom-up forecasting: Applications and limitations in load forecasting using smart-meter data', *Data-Centric Engineering*, 2023, doi: 10.1017/dce.2023.10.

[2] N. Abdulla, M. Demirci, and S. Ozdemir, 'Smart meter-based energy consumption forecasting for smart cities using adaptive federated learning', *Sustainable Energy, Grids and Networks*, 2024, doi: 10.1016/j.segan.2024.101342.

[3] X. Wen, J. Liao, Q. Niu, N. Shen, and Y. Bao, 'Deep learning-driven hybrid model for short-term load forecasting and smart grid information management', *Sci Rep*, vol. 14, no. 1, Jun. 2024, doi: 10.1038/s41598-024-63262-x.

[4] S. Akhtar *et al.*, 'Short-Term Load Forecasting Models: A Review of Challenges, Progress, and the Road Ahead', *Energies*, vol. 16, no. 10, p. 4060, May 2023, doi: 10.3390/en16104060.

[5] C. Li, D. Liu, M. Wang, H. Wang, and S. Xu, 'Detection of Outliers in Time Series Power Data Based on Prediction Errors', *Energies*, vol. 16, no. 2, p. 582, Jan. 2023, doi: 10.3390/en16020582.

[6] A. Zaboli, S. R. Kasimalla, K. Park, Y. Hong, and J. Hong, 'A Comprehensive Review of Behind-the-Meter Distributed Energy Resources Load Forecasting: Models, Challenges, and Emerging Technologies', *Energies*, vol. 17, no. 11, p. 2534, May 2024, doi: 10.3390/en17112534.

[7] M. Iqbal, M. Adnan, S. E. G. Mohamed, and M. Tariq, 'A Hybrid Deep Learning Framework for Short-Term Load Forecasting with Improved Data Cleansing and Preprocessing Techniques', *Results in Engineering*, 2024, doi: 10.1016/j.rineng.2024.103560.

[8] M. Awais *et al.*, 'Short-term photovoltaic energy generation for solar powered high efficiency irrigation systems using LSTM with Spatio-temporal attention mechanism', *Scientific Reports*, 2024, doi: 10.1038/s41598-024-60672-9.

[9] M. Diqi, A. Sahal, and F. N. Aini, 'Multi-Step Vector Output Prediction of Time Series Using EMA LSTM', *Jurnal Online Informatika*, vol. 8, no. 1, pp. 107–114, 2023, doi: 10.15575/join.v8i1.1037.

[10] W. Cao, H. Liu, X. Zhang, and Y. Zeng, 'Residential Load Forecasting Based on Long Short-Term Memory, Considering Temporal Local Attention', *Sustainability*, 2024, doi: 10.3390/su162411252.

[11] S. Z. H. Shah *et al.*, 'Improved electric load forecasting using quantile long short-term memory network with dual attention mechanism', *Energy Reports*, 2025, doi: 10.1016/j.egyr.2025.01.058.

[12] Md. S. Abid, R. Ahshan, M. Al-Abri, and R. Al Abri, 'Spatiotemporal Forecasting of Solar and Wind Energy Production: A Robust Deep Learning Model with Attention Framework', *Energy Conversion and Management: X*, vol. 26, p. 100919, Apr. 2025, doi: 10.1016/j.ecmx.2025.100919.

[13] U. Shivani *et al.*, 'MSMVAN: Multi Step Multi Variate Deep Attention Network for Renewable Energy Forecast', *IEEE transactions on industry applications*, 2024, doi: 10.1109/TIA.2023.3347174.

[14] J. Bi, H. Ma, H. Yuan, and J. Zhang, 'Accurate Prediction of Workloads and Resources With Multi-Head Attention and Hybrid LSTM for Cloud Data Centers', *IEEE Trans. Sustain. Comput.*, vol. 8, no. 3, pp. 375–384, Jul. 2023, doi: 10.1109/tsusc.2023.3259522.

[15] X. Dai, G. Liu, and W. Hu, 'An online-learning-enabled self-attention-based model for ultra-short-term wind power forecasting', *Energy*, 2023, doi: 10.1016/j.energy.2023.127173.

[16] A. Köklü, Y. Güven, and T. Kumbasar, 'Odyssey of Interval Type-2 Fuzzy Logic Systems: Learning Strategies for Uncertainty Quantification', *IEEE transactions on fuzzy systems*, 2025, doi: 10.1109/TFUZZ.2024.3482393.

[17] Z. Ni, C. Zhang, M. Karlsson, and S. Gong, 'A study of deep learning-based multi-horizon building energy forecasting', *Energy and Buildings*, vol. 303, p. 113810, Jan. 2024, doi: 10.1016/j.enbuild.2023.113810.

[18] J. Wang, K. Wang, Z. Li, H. Lu, H. Jiang, and Q. Xing, 'A Multitask Integrated Deep-Learning Probabilistic Prediction for Load Forecasting', *IEEE Transactions on Power Systems*, 2024, doi: 10.1109/TPWRS.2023.3257353.

[19] T. Blasco, J. S. Sánchez, and V. García, 'A survey on uncertainty quantification in deep learning for financial time series prediction', *Neurocomputing*, 2024, doi: 10.1016/j.neucom.2024.127339.

[20] M. Diqi, E. Utami, Kusrini, and F. W. Wibowo, 'DQVAE: leveraging adaptive modulated sigmoid, attention, and quantization for high-fidelity synthetic stock market data', *International Journal of Information Technology*, Jun. 2025, doi: 10.1007/s41870-025-02575-0.

[21] M. Salvi *et al.*, 'Explainability and uncertainty: Two sides of the same coin for enhancing the interpretability of deep learning models in healthcare', *International Journal of Medical Informatics*, vol. 197, p. 105846, May 2025, doi: 10.1016/j.ijmedinf.2025.105846.

[22] A. Lebedev, A. Das, S. Pappert, and S. Schlüter, 'Analyzing Uncertainty Quantification in Statistical and Deep Learning Models for Probabilistic Electricity Price Forecasting', *IEEE Access*, vol. 14, pp. 52162–52189, 2026, doi: 10.1109/access.2026.3672716.

[23] Y. Wu, Y. Ye, A. Zeb, J. J. Q. Yu, and Z. Wang, 'Adaptive Modeling of Uncertainties for Traffic Forecasting', *IEEE transactions on intelligent transportation systems (Print)*, 2023, doi: 10.1109/TITS.2023.3327100.

[24] M. Jiménez-Navarro, M. Lovrić, S. Kecorius, E. Nyarko, and M. Martínez-Ballesteros, 'Explainable deep learning on multi-target time series forecasting: an air pollution use case', *Results in Engineering*, 2024, doi: 10.1016/j.rineng.2024.103290.

[25] D. Wood, T. Papamarkou, M. Benatan, and R. Allmendinger, 'Model-agnostic variable importance for predictive uncertainty: an entropy-based approach', *Data mining and knowledge discovery*, 2023, doi: 10.1007/s10618-024-01070-7.

[26] Y. Eren and İ. Küçükdemiral, 'A comprehensive review on deep learning approaches for short-term load forecasting', *Renewable & Sustainable Energy Reviews*, 2024, doi: 10.1016/j.rser.2023.114031.

[27] M. A. A. Sarker, B. Shanmugam, S. Azam, and S. Thennadil, 'Enhancing smart grid load forecasting: An attention-based deep learning model integrated with federated learning and XAI for security and interpretability', *Intelligent Systems with Applications*, 2024, doi: 10.1016/j.iswa.2024.200422.

[28] C. Huang, S. Bu, W. Chen, H. Wang, and Y. Zhang, 'Deep Reinforcement Learning-Assisted Federated Learning for Robust Short-Term Load Forecasting in Electricity Wholesale Markets', *IEEE Transactions on Network Science and Engineering*, 2024, doi: 10.1109/TNSE.2024.3427672.

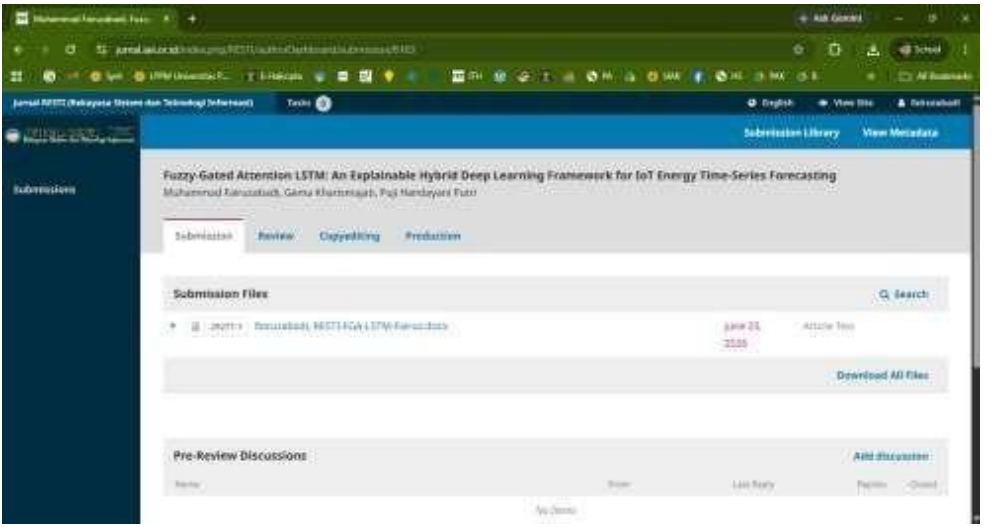
[29] M. J. Mokarram, R. Rashiditabar, M. Gitizadeh, and J. Aghaei, 'Net-load forecasting of renewable energy systems

using multi-input LSTM fuzzy and discrete wavelet transform', *Energy*, vol. 275, p. 127425, Jul. 2023, doi: 10.1016/j.energy.2023.127425.

[30] H. Habbak, M. Mahmoud, K. Metwally, M. M. Fouda, and M. I. Ibrahem, 'Load Forecasting Techniques and Their Applications in Smart Grids', *Energies*, 2023, doi: 10.3390/en16031480.

[31] L. Baur, K. Ditschuneit, M. Schambach, C. Kaymakci, T. Wollmann, and A. Sauer, 'Explainability and Interpretability in Electric Load Forecasting Using Machine Learning Techniques – A Review', *Energy and AI*, 2024, doi: 10.1016/j.egyai.2024.100358.

Bukti Submit



BUKU REFERENSI

Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series

Penulis

Muhammad Fairuzabadi, S.Si., M.Kom.
Puji Handayani Putri, S.T., M.Kom.
Gema Kharismajati, S.Kom., M.Kom.

Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series

Penulis

Muhammad Fairuzabadi, S.Si., M.Kom.

Puji Handayani Putri, S.T., M.Kom.

Gema Kharismajati, S.Kom., M.Kom.

Kata Pengantar

Puji syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas rahmat dan karunia-Nya, buku **Deep Learning dan Fuzzy Logic untuk Prediksi Data Time Series** ini dapat disusun sebagai bahan bacaan akademik yang membahas integrasi antara pendekatan *deep learning* dan *fuzzy logic* dalam pemodelan prediktif berbasis data deret waktu.

Perkembangan kecerdasan buatan telah membawa perubahan besar dalam cara data dianalisis dan digunakan untuk mendukung pengambilan keputusan. Salah satu jenis data yang banyak ditemukan dalam kehidupan nyata adalah data *time series*, yaitu data yang dicatat berdasarkan urutan waktu. Data konsumsi energi, suhu, penjualan, trafik jaringan, harga saham, data sensor IoT, hingga data kesehatan merupakan contoh data *time series* yang memiliki pola temporal dan dapat dimanfaatkan untuk membangun sistem prediksi.

Buku ini disusun untuk memberikan pemahaman bertahap mengenai konsep dasar data *time series*, pra-pemrosesan data, jaringan saraf tiruan, RNN, LSTM, GRU, serta *fuzzy logic*. Pembahasan kemudian diarahkan pada bagaimana LSTM dapat digunakan sebagai model prediksi numerik, sedangkan *fuzzy logic* dapat digunakan untuk membantu interpretasi hasil prediksi melalui kategori linguistik seperti rendah, sedang, dan tinggi. Dengan pendekatan tersebut, sistem prediksi tidak hanya menghasilkan angka, tetapi juga memberikan makna yang lebih mudah dipahami dan digunakan.

Fokus utama buku ini adalah integrasi **LSTM–Fuzzy Logic** untuk prediksi data *time series*, khususnya pada studi kasus konsumsi energi. LSTM digunakan karena memiliki kemampuan dalam mempelajari pola data berurutan dan dependensi jangka panjang, sedangkan *fuzzy logic* digunakan karena mampu menangani ketidakpastian serta menyajikan hasil prediksi dalam bentuk penalaran yang lebih interpretatif. Kombinasi keduanya diharapkan dapat memperkaya pemahaman pembaca mengenai model hibrida dalam kecerdasan buatan.

Materi dalam buku ini disusun dengan bahasa yang mudah dipahami, tetapi tetap mempertahankan ketepatan konsep akademik. Setiap bab dirancang

agar pembaca dapat memahami konsep secara bertahap, mulai dari fondasi data *time series*, konsep *deep learning*, mekanisme LSTM dan GRU, dasar-dasar *fuzzy logic*, perancangan model hibrida, evaluasi model, hingga studi kasus prediksi konsumsi energi. Beberapa bagian juga dilengkapi dengan rumus, tabel, serta rekomendasi gambar untuk memperjelas konsep yang dibahas.

Penulis menyadari bahwa buku ini masih memiliki keterbatasan, baik dari sisi kedalaman pembahasan, kelengkapan contoh, maupun cakupan implementasi. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan untuk penyempurnaan edisi berikutnya. Semoga buku ini dapat memberikan manfaat bagi pembaca yang ingin mempelajari *deep learning*, *fuzzy logic*, dan pemodelan prediksi data *time series*, serta menjadi referensi dalam pengembangan penelitian dan implementasi sistem cerdas berbasis data.

Yogyakarta, Juni 2026

Penulis

Daftar Isi

Daftar Gambar

No table of figures entries found.

Daftar Tabel

Bab 1
Pengantar Deep Learning, Fuzzy Logic, dan Data Time Series

1.1 Perkembangan Deep Learning dan Fuzzy Logic dalam Prediksi Data Time Series

Prediksi data *time series* menjadi salah satu topik penting dalam kecerdasan buatan karena banyak data di dunia nyata tersusun berdasarkan urutan waktu. Contohnya adalah konsumsi energi listrik harian, suhu udara, jumlah transaksi penjualan, kunjungan pasien, harga saham, trafik jaringan, dan data sensor Internet of Things (IoT). Pada data seperti ini, nilai pada waktu tertentu sering kali memiliki hubungan dengan nilai sebelumnya. Artinya, untuk memperkirakan kondisi masa depan, sistem perlu memahami pola masa lalu.

Dalam konteks penelitian ini, data konsumsi energi termasuk data *time series* yang bersifat fluktuatif, nonlinier, dan dipengaruhi oleh pola temporal jangka pendek maupun jangka panjang. Oleh karena itu, pendekatan berbasis *deep learning*, khususnya Long Short-Term Memory (LSTM), dinilai relevan untuk mempelajari pola berurutan, sedangkan *fuzzy logic* digunakan untuk membantu menjelaskan hasil prediksi dalam bentuk kategori linguistik seperti rendah, sedang, dan tinggi.

1.1.1 Perkembangan Prediksi Data dari Statistik ke Kecerdasan Buatan

Pada awalnya, prediksi data *time series* banyak menggunakan pendekatan statistik, seperti *moving average*, *exponential smoothing*, ARIMA, dan regresi. Metode tersebut masih banyak digunakan karena relatif sederhana dan mudah dijelaskan. Namun, metode statistik klasik sering memiliki keterbatasan ketika data memiliki pola yang kompleks, tidak linier, banyak

noise, atau dipengaruhi oleh banyak variabel lain (Box et al., 2015; Hyndman & Athanasopoulos, 2021).

Seiring berkembangnya kecerdasan buatan, metode *machine learning* mulai digunakan untuk melakukan prediksi berbasis data historis. Algoritma seperti *decision tree*, *random forest*, *support vector machine*, dan *neural network* mampu mengenali pola yang lebih kompleks dibandingkan metode statistik sederhana. Namun, ketika data memiliki struktur berurutan, seperti data konsumsi listrik per jam atau per hari, model perlu memahami ketergantungan antarwaktu. Kebutuhan inilah yang mendorong penggunaan *deep learning* untuk pemodelan data sekuensial.

Deep learning berkembang pesat karena kemampuannya mempelajari representasi data secara otomatis melalui banyak lapisan jaringan saraf tiruan. Model ini tidak hanya menerima data sebagai input, tetapi juga mampu menemukan pola tersembunyi yang sulit dirumuskan secara manual (Goodfellow et al., 2016; LeCun et al., 2015). Pada data *time series*, kemampuan ini sangat penting karena pola data sering kali tidak terlihat secara langsung, misalnya pola beban puncak pada jam tertentu atau peningkatan konsumsi energi pada hari kerja.

1.1.2 Peran Deep Learning dalam Prediksi Data Time Series

Salah satu kelompok model *deep learning* yang banyak digunakan untuk data berurutan adalah Recurrent Neural Network (RNN). RNN dirancang untuk mengolah data yang memiliki urutan, sehingga cocok digunakan pada teks, sinyal, suara, dan data *time series*. Namun, RNN dasar memiliki kelemahan ketika harus mengingat informasi dalam rentang waktu yang panjang karena muncul masalah *vanishing gradient*.

Untuk mengatasi keterbatasan tersebut, dikembangkan Long Short-Term Memory (LSTM). LSTM memiliki mekanisme *gate* yang mengatur informasi mana yang perlu disimpan, dilupakan, atau digunakan untuk prediksi. Mekanisme ini membuat LSTM lebih mampu mempelajari hubungan jangka panjang pada data berurutan (Hochreiter & Schmidhuber, 1997). Dalam prediksi konsumsi energi, LSTM dapat digunakan untuk membaca pola konsumsi masa lalu dan menghasilkan nilai prediksi konsumsi energi pada periode berikutnya.

Selain LSTM, terdapat pula Gated Recurrent Unit (GRU) yang memiliki struktur lebih sederhana tetapi tetap kuat untuk pemodelan data sekuensial. GRU sering digunakan sebagai alternatif LSTM karena jumlah parameternya lebih sedikit sehingga proses pelatihan dapat lebih ringan. Meskipun demikian, pemilihan LSTM atau GRU tetap bergantung pada karakteristik data, ukuran dataset, kompleksitas pola, dan kebutuhan akurasi model.

1.1.3 Peran Fuzzy Logic dalam Interpretasi Hasil Prediksi

Meskipun *deep learning* memiliki kemampuan prediksi yang baik, model ini sering dianggap sebagai *black box*. Artinya, pengguna dapat melihat hasil prediksi, tetapi sulit memahami alasan model menghasilkan nilai tersebut. Pada kasus prediksi konsumsi energi, angka prediksi saja kadang belum cukup. Pengambil keputusan biasanya membutuhkan interpretasi yang lebih mudah dipahami, misalnya apakah konsumsi energi termasuk rendah, sedang, atau tinggi.

Di sinilah *fuzzy logic* memiliki peran penting. *Fuzzy logic* diperkenalkan sebagai pendekatan untuk menangani ketidakpastian dan nilai yang tidak bersifat tegas. Berbeda dengan logika biner yang hanya mengenal benar atau salah, *fuzzy logic* memungkinkan suatu nilai memiliki derajat keanggotaan tertentu dalam beberapa kategori sekaligus (Zadeh, 1965). Misalnya, konsumsi energi 480 kWh dapat dianggap sebagian termasuk kategori "sedang" dan sebagian mendekati kategori "tinggi", bergantung pada batas dan fungsi keanggotaan yang digunakan.

Pendekatan ini membuat hasil prediksi lebih mudah diterjemahkan ke dalam bahasa manusia. Jika LSTM menghasilkan nilai prediksi konsumsi energi, maka *fuzzy logic* dapat mengubah nilai tersebut menjadi informasi keputusan. Contohnya, sistem dapat memberikan status "beban energi tinggi" dan menyarankan pengurangan penggunaan perangkat tertentu. Dengan cara ini, *fuzzy logic* tidak menggantikan fungsi prediksi LSTM, tetapi melengkapi LSTM dari sisi interpretasi dan penalaran.

1.1.4 Integrasi Deep Learning dan Fuzzy Logic

Integrasi *deep learning* dan *fuzzy logic* bertujuan menggabungkan dua kelebihan utama. *Deep learning* kuat dalam mengenali pola data yang

kompleks, sedangkan *fuzzy logic* kuat dalam menjelaskan hasil prediksi melalui aturan dan kategori linguistik. Dalam model hibrida LSTM–Fuzzy Logic, LSTM dapat ditempatkan sebagai model prediktif, sedangkan *fuzzy logic* berperan sebagai sistem interpretasi atau sistem pendukung keputusan.

Secara umum, alur kerjanya dapat dijelaskan sebagai berikut. Pertama, data *time series* dikumpulkan dan diproses agar siap digunakan oleh model. Kedua, LSTM dilatih menggunakan data historis untuk memprediksi nilai masa depan. Ketiga, hasil prediksi LSTM dimasukkan ke dalam sistem *fuzzy*. Keempat, sistem *fuzzy* mengklasifikasikan hasil prediksi ke dalam kategori tertentu, misalnya konsumsi rendah, sedang, atau tinggi. Pendekatan seperti ini sesuai dengan rancangan penelitian yang mengintegrasikan LSTM untuk pemodelan data deret waktu dan Fuzzy Logic untuk penalaran serta klasifikasi keputusan yang toleran terhadap ketidakpastian.

Cara paling sederhana memahami integrasi ini adalah dengan membedakan antara "prediksi angka" dan "makna dari angka". LSTM bertugas menjawab pertanyaan "berapa nilai yang diprediksi?", sedangkan *fuzzy logic* membantu menjawab "apa arti nilai tersebut bagi pengguna?". Perbedaan ini penting karena sistem cerdas tidak hanya dituntut akurat, tetapi juga perlu memberikan informasi yang dapat dipahami dan digunakan.

1.1.5 Contoh Penerapan pada Prediksi Konsumsi Energi

Pada prediksi konsumsi energi, data yang digunakan dapat berupa konsumsi listrik per jam, per hari, atau per bulan. Data tersebut kemudian dipelajari oleh model LSTM untuk menemukan pola tertentu. Misalnya, konsumsi energi dapat meningkat pada jam kerja, menurun pada malam hari, atau naik pada periode tertentu karena penggunaan perangkat pendingin ruangan.

Setelah model menghasilkan prediksi, *fuzzy logic* dapat digunakan untuk mengelompokkan hasilnya. Jika prediksi konsumsi energi berada pada rentang tertentu, sistem dapat mengategorikannya sebagai "rendah", "sedang", atau "tinggi". Kategori ini lebih mudah dipahami oleh pengelola gedung, teknisi, atau pengguna umum dibandingkan hanya menampilkan angka mentah. Dalam proposal penelitian, integrasi ini memang diarahkan untuk menghasilkan model prediksi konsumsi energi berbasis LSTM–Fuzzy Logic sekaligus mendukung interpretasi hasil prediksi pada data *time series*.

1.2 Peran Deep Learning Dalam Pemodelan Sekuensial

Data sekuensial adalah data yang urutan kemunculannya memiliki arti penting. Pada data jenis ini, nilai saat ini tidak berdiri sendiri, tetapi sering dipengaruhi oleh nilai sebelumnya. Contohnya dapat ditemukan pada data konsumsi energi per jam, suhu harian, trafik internet, harga saham, jumlah transaksi, sinyal kesehatan, dan data sensor IoT. Jika urutan waktunya diacak, maka makna pola datanya dapat berubah. Oleh karena itu, pemodelan sekuensial tidak hanya mempelajari "berapa besar nilainya", tetapi juga "bagaimana pola nilainya berubah dari waktu ke waktu".

Dalam prediksi data *time series*, *deep learning* berperan sebagai pendekatan yang mampu mempelajari pola temporal secara otomatis dari data historis. Model ini dapat menangkap hubungan nonlinier, pola berulang, tren jangka pendek, dan ketergantungan jangka panjang yang sering sulit ditangkap oleh metode statistik sederhana. Pada konteks konsumsi energi, data biasanya bersifat fluktuatif, nonlinier, dan dipengaruhi oleh pola waktu jangka pendek maupun jangka panjang. Hal ini menjadikan model berbasis *deep learning*, khususnya RNN, LSTM, dan GRU, relevan untuk pemodelan data sekuensial.

1.2.1 Konsep Dasar Pemodelan Sekuensial

Pemodelan sekuensial bertujuan mempelajari hubungan antar-data yang tersusun dalam urutan tertentu. Pada data *time series*, urutan tersebut biasanya berdasarkan waktu. Misalnya, konsumsi listrik pukul 08.00 dapat dipengaruhi oleh konsumsi listrik pukul 07.00, 06.00, bahkan pola konsumsi pada hari sebelumnya. Hubungan seperti ini disebut dependensi temporal.

Secara sederhana, pemodelan sekuensial dapat dipahami sebagai proses menggunakan data masa lalu untuk memperkirakan kondisi berikutnya. Jika tersedia data konsumsi energi selama beberapa jam sebelumnya, model dapat mempelajari pola tersebut untuk memprediksi konsumsi energi pada jam berikutnya. Dalam praktiknya, data sering dibentuk menjadi *window* atau jendela waktu. Misalnya, data 24 jam terakhir digunakan untuk memprediksi konsumsi energi 1 jam berikutnya.

Konsep tersebut dapat dituliskan secara sederhana sebagai berikut:

$$X_t = [x_{t-n}, x_{t-n+1}, \dots, x_{t-1}]$$

$$\hat{y_t} = f(X_t)$$

Keterangan:

- X_t adalah kumpulan data historis dalam jendela waktu tertentu.
- x_{t-n} sampai x_{t-1}adalah nilai-nilai sebelumnya.
- y_t adalah nilai prediksi pada waktu ke-ttt.
- $f(X_t)$ adalah fungsi model yang mempelajari pola dari data historis.

Rumus ini hanya menggambarkan ide umum bahwa model menggunakan sekumpulan nilai masa lalu untuk menghasilkan prediksi masa depan. Pada *deep learning*, fungsi fff tidak dirumuskan secara manual, tetapi dipelajari oleh jaringan saraf melalui proses pelatihan.

1.2.2 Mengapa Deep Learning Digunakan Untuk Data Sekuensial

Metode statistik klasik seperti ARIMA, *moving average*, dan *exponential smoothing* masih berguna untuk prediksi data *time series*. Namun, ketika data memiliki pola yang kompleks, banyak variabel, tidak linier, atau mengandung perubahan mendadak, model statistik sederhana sering membutuhkan asumsi yang cukup ketat. *Deep learning* menawarkan pendekatan yang lebih fleksibel karena mampu mempelajari pola dari data secara langsung melalui banyak lapisan pemrosesan (Goodfellow et al., 2016).

Keunggulan utama *deep learning* terletak pada kemampuan *representation learning*, yaitu kemampuan model untuk membentuk representasi atau pola penting dari data tanpa harus seluruhnya ditentukan secara manual oleh manusia. Pada data konsumsi energi, misalnya, model dapat belajar bahwa konsumsi cenderung meningkat pada jam kerja, menurun pada malam hari, atau berubah karena pola penggunaan perangkat tertentu. Model tidak hanya membaca nilai satu per satu, tetapi mencoba menemukan keteraturan dari urutan nilai tersebut.

Selain itu, *deep learning* dapat digunakan pada data dalam jumlah besar. Hal ini penting karena perkembangan IoT dan *smart meter* memungkinkan pencatatan data secara terus-menerus. Data yang awalnya hanya dicatat harian atau bulanan kini dapat direkam dalam interval menit atau jam.

Semakin sering data dicatat, semakin besar peluang model untuk mengenali pola temporal secara lebih detail.

1.2.3 Recurrent Neural Network Sebagai Dasar Model Sekuensial

Recurrent Neural Network atau RNN merupakan salah satu arsitektur jaringan saraf yang dirancang untuk mengolah data berurutan. Berbeda dengan jaringan saraf biasa yang menganggap setiap input berdiri sendiri, RNN memiliki mekanisme memori sederhana yang memungkinkan informasi dari langkah sebelumnya digunakan pada langkah berikutnya. Dengan cara ini, RNN dapat memproses data secara berurutan.

Pada data teks, RNN dapat membaca kata demi kata. Pada data *time series*, RNN dapat membaca nilai dari waktu ke waktu. Misalnya, model membaca konsumsi energi pukul 01.00, lalu pukul 02.00, kemudian pukul 03.00, dan seterusnya. Informasi dari waktu sebelumnya ikut memengaruhi pemahaman model terhadap waktu berikutnya.

Namun, RNN dasar memiliki kelemahan dalam menangani dependensi jangka panjang. Ketika urutan data terlalu panjang, informasi dari waktu yang jauh sebelumnya dapat melemah selama proses pelatihan. Masalah ini dikenal sebagai *vanishing gradient*. Akibatnya, model dapat kesulitan mengingat pola penting yang terjadi jauh sebelum waktu prediksi.

1.2.4 LSTM Untuk Menangkap Dependensi Jangka Panjang

Long Short-Term Memory atau LSTM dikembangkan untuk mengatasi kelemahan RNN dasar, terutama masalah *vanishing gradient*. LSTM memiliki struktur khusus berupa *gate* yang mengatur aliran informasi di dalam model. Melalui mekanisme ini, LSTM dapat menentukan informasi mana yang perlu disimpan, dilupakan, atau digunakan untuk menghasilkan output (Hochreiter & Schmidhuber, 1997).

Secara konseptual, LSTM bekerja seperti sistem memori yang lebih selektif. Jika suatu informasi dianggap penting untuk prediksi, maka informasi tersebut dapat dipertahankan lebih lama. Jika informasi dianggap tidak relevan, maka model dapat mengabaikannya. Pada prediksi konsumsi energi, kemampuan ini bermanfaat karena pola konsumsi tidak hanya

dipengaruhi oleh waktu terdekat, tetapi juga pola sebelumnya, seperti pola harian, mingguan, atau musiman.

Sebagai contoh, konsumsi energi pada hari Senin pagi mungkin memiliki pola yang berbeda dengan Minggu malam. LSTM dapat mempelajari hubungan semacam ini dari data historis. Inilah alasan LSTM banyak digunakan dalam prediksi beban listrik, prediksi cuaca, analisis sinyal, pemrosesan bahasa alami, dan berbagai bentuk data sekuensial lainnya.

1.2.5 GRU Sebagai Alternatif LSTM

Gated Recurrent Unit atau GRU merupakan arsitektur lain yang juga dikembangkan untuk mengolah data sekuensial. GRU memiliki prinsip kerja yang mirip dengan LSTM, tetapi strukturnya lebih sederhana. Jika LSTM memiliki beberapa mekanisme *gate*, GRU menggunakan jumlah *gate* yang lebih sedikit sehingga proses komputasinya dapat lebih ringan (Cho et al., 2014).

GRU sering dipilih ketika peneliti membutuhkan model yang lebih sederhana, lebih cepat dilatih, atau ketika ukuran dataset tidak terlalu besar. Meskipun demikian, tidak selalu berarti GRU lebih baik daripada LSTM. Pada beberapa kasus, LSTM dapat memberikan hasil lebih stabil, terutama ketika data memiliki pola jangka panjang yang kompleks. Pada kasus lain, GRU dapat menghasilkan performa yang sebanding dengan waktu pelatihan lebih singkat.

Pemilihan antara LSTM dan GRU perlu mempertimbangkan karakteristik data, ukuran dataset, tujuan penelitian, sumber daya komputasi, serta metrik evaluasi yang digunakan. Oleh karena itu, dalam penelitian prediksi *time series*, kedua model ini sering dibandingkan sebagai bagian dari analisis eksperimental.

1.2.6 Alur Deep Learning Dalam Prediksi Data Time Series

Pemodelan *time series* menggunakan *deep learning* biasanya melalui beberapa tahapan. Tahap pertama adalah pengumpulan data historis. Data dapat berasal dari sensor, *smart meter*, sistem transaksi, rekam medis, atau sumber data lain yang tersusun berdasarkan waktu. Tahap kedua adalah pra-pemrosesan data, seperti membersihkan data, menangani nilai hilang, melakukan normalisasi, dan membentuk *window time series*.

Tahap ketiga adalah membangun model. Pada tahap ini, peneliti menentukan arsitektur yang digunakan, misalnya RNN, LSTM, GRU, atau variasi lainnya. Tahap keempat adalah pelatihan model, yaitu proses ketika model belajar dari data latih. Setelah itu dilakukan validasi dan pengujian untuk melihat kemampuan model dalam memprediksi data baru.

Tahap terakhir adalah interpretasi hasil. Pada tahap ini, hasil prediksi tidak cukup hanya dilihat dari angka. Model perlu dievaluasi menggunakan metrik seperti MAE, RMSE, atau MAPE. Untuk sistem yang membutuhkan pengambilan keputusan, hasil prediksi juga dapat diterjemahkan menjadi kategori yang lebih mudah dipahami. Pada pendekatan LSTM–Fuzzy Logic, LSTM digunakan untuk menghasilkan prediksi numerik, sedangkan *fuzzy logic* digunakan untuk membantu interpretasi keputusan. Proposal penelitian juga menempatkan LSTM sebagai model prediksi deret waktu dan Fuzzy Logic sebagai mekanisme penalaran serta klasifikasi keputusan yang toleran terhadap ketidakpastian.

1.2.7 Keterbatasan Deep Learning Dalam Pemodelan Sekuensial

Meskipun *deep learning* memiliki kemampuan yang kuat, pendekatan ini tetap memiliki keterbatasan. Pertama, model membutuhkan data yang cukup agar dapat belajar secara baik. Jika data terlalu sedikit atau tidak representatif, hasil prediksi dapat menjadi kurang akurat. Kedua, model sensitif terhadap kualitas data. Data yang banyak mengandung kesalahan, nilai hilang, atau pencatatan tidak konsisten dapat menurunkan performa model.

Ketiga, *deep learning* membutuhkan proses pelatihan yang lebih kompleks dibandingkan metode statistik sederhana. Peneliti perlu menentukan jumlah lapisan, jumlah neuron, fungsi aktivasi, *optimizer*, ukuran *batch*, jumlah *epoch*, dan parameter lainnya. Keputusan tersebut dapat memengaruhi hasil akhir model.

Keempat, model *deep learning* sering dianggap sulit dijelaskan. Pengguna dapat mengetahui hasil prediksinya, tetapi tidak selalu mudah memahami alasan model menghasilkan prediksi tersebut. Keterbatasan ini menjadi salah satu alasan integrasi dengan *fuzzy logic* menjadi penting. *Fuzzy logic* dapat

membantu menerjemahkan hasil prediksi menjadi kategori linguistik dan aturan keputusan yang lebih mudah dipahami.

1.3 Peran Fuzzy Logic Dalam Penalaran Ketidakpastian

Pada banyak kasus prediksi, hasil yang diperoleh tidak selalu dapat dijelaskan secara tegas. Misalnya, ketika sistem memprediksi konsumsi energi sebesar 480 kWh, pertanyaan berikutnya adalah: apakah nilai tersebut termasuk rendah, sedang, atau tinggi? Jika batas kategori dibuat terlalu kaku, maka nilai yang berada dekat dengan batas kategori dapat menimbulkan masalah interpretasi. Konsumsi energi 499 kWh mungkin dikategorikan "sedang", sedangkan 501 kWh dikategorikan "tinggi", padahal perbedaannya hanya 2 kWh. Kondisi seperti ini menunjukkan bahwa pengambilan keputusan dalam sistem cerdas sering berhadapan dengan ketidakpastian.

Fuzzy logic atau logika fuzzy dikembangkan untuk menangani kondisi yang tidak sepenuhnya benar atau salah. Berbeda dengan logika klasik yang hanya mengenal dua nilai, yaitu 0 dan 1, *fuzzy logic* memungkinkan suatu nilai memiliki derajat keanggotaan di antara 0 sampai 1. Konsep ini diperkenalkan oleh Lotfi A. Zadeh melalui teori himpunan fuzzy, yang menjelaskan bahwa suatu objek dapat menjadi anggota suatu himpunan dengan tingkat keanggotaan tertentu, bukan hanya sebagai anggota atau bukan anggota (Zadeh, 1965).

1.3.1 Konsep Ketidakpastian Dalam Data

Ketidakpastian dalam data dapat muncul karena berbagai faktor. Pada data *time series*, ketidakpastian dapat berasal dari perubahan pola waktu, noise, kesalahan pencatatan, data hilang, atau faktor eksternal yang tidak sepenuhnya tercatat. Dalam prediksi konsumsi energi, misalnya, perubahan cuaca, aktivitas pengguna, jam operasional, dan penggunaan perangkat listrik dapat menyebabkan pola konsumsi menjadi tidak stabil.

Ketidakpastian juga muncul ketika data numerik perlu diterjemahkan menjadi keputusan. Nilai prediksi yang dihasilkan model *deep learning* biasanya berbentuk angka. Angka tersebut memang penting, tetapi tidak selalu langsung bermakna bagi pengguna. Pengguna sering membutuhkan interpretasi yang lebih mudah dipahami, seperti "beban energi rendah",

"beban energi sedang", atau "beban energi tinggi". *Fuzzy logic* membantu menjembatani angka prediksi dengan makna keputusan yang lebih manusiawi.

Dalam konteks sistem cerdas, ketidakpastian bukan berarti data tidak berguna. Ketidakpastian justru perlu dimodelkan agar sistem tidak membuat keputusan secara terlalu kaku. Pendekatan fuzzy memungkinkan sistem memberikan penilaian yang lebih lentur, terutama ketika data berada pada area abu-abu atau mendekati batas antar-kategori (Klir & Yuan, 1995).

1.3.2 Himpunan Fuzzy Dan Derajat Keanggotaan

Konsep utama dalam *fuzzy logic* adalah himpunan fuzzy. Dalam himpunan tegas, suatu nilai hanya memiliki dua kemungkinan: masuk anggota himpunan atau tidak masuk anggota himpunan. Misalnya, jika batas konsumsi tinggi ditentukan mulai dari 500 kWh, maka nilai 501 kWh dianggap tinggi, sedangkan 499 kWh dianggap tidak tinggi.

Pada himpunan fuzzy, nilai 499 kWh dapat memiliki derajat keanggotaan tertentu terhadap kategori "tinggi", misalnya 0,6. Artinya, nilai tersebut belum sepenuhnya tinggi, tetapi sudah mendekati kategori tinggi. Derajat keanggotaan ini biasanya dinyatakan dengan simbol μ(x)\mu(x)μ(x), dengan rentang nilai antara 0 sampai 1.

Secara sederhana, konsep derajat keanggotaan dapat ditulis sebagai berikut:

$$0 \leq \mu(x) \leq 1$$

Keterangan:

- $\mu(x)$ adalah derajat keanggotaan suatu nilai xxx terhadap kategori tertentu.
- Nilai 0 berarti tidak menjadi anggota kategori.
- Nilai 1 berarti sepenuhnya menjadi anggota kategori.
- Nilai di antara 0 dan 1 menunjukkan tingkat keanggotaan sebagian.

Sebagai contoh, konsumsi energi 300 kWh dapat memiliki derajat keanggotaan 0,8 pada kategori "sedang" dan 0,2 pada kategori "rendah". Dengan cara ini, sistem dapat menggambarkan kondisi yang lebih realistis dibandingkan pembagian kategori yang terlalu tegas.

1.3.3 Fungsi Keanggotaan Dalam Fuzzy Logic

Fungsi keanggotaan digunakan untuk menentukan seberapa besar suatu nilai masuk ke dalam kategori fuzzy tertentu. Bentuk fungsi keanggotaan yang umum digunakan antara lain segitiga, trapesium, Gaussian, dan sigmoid. Pemilihan bentuk fungsi keanggotaan bergantung pada karakteristik data dan kebutuhan sistem (Ross, 2010).

Dalam prediksi data *time series*, fungsi keanggotaan dapat digunakan untuk mengelompokkan hasil prediksi ke dalam kategori linguistik. Misalnya, hasil prediksi konsumsi energi dapat dikelompokkan menjadi rendah, sedang, dan tinggi. Masing-masing kategori memiliki rentang nilai dan fungsi keanggotaan sendiri.

Contoh sederhana:

Tabel 1.1: Tabel 1.3 Contoh Interpretasi Fuzzy Pada Prediksi Konsumsi Energi

Nilai Prediksi Konsumsi Energi	Kategori Dominan	Makna Keputusan
150 kWh	Rendah	Konsumsi masih aman
420 kWh	Sedang	Konsumsi perlu dipantau
780 kWh	Tinggi	Perlu pengendalian beban

Tabel tersebut menunjukkan bahwa *fuzzy logic* tidak hanya berfungsi untuk mengelompokkan nilai, tetapi juga membantu memberikan makna keputusan. Pada sistem nyata, kategori tersebut dapat dilengkapi dengan aturan, misalnya jika beban tinggi maka sistem memberikan peringatan atau rekomendasi penghematan.

1.3.4 Aturan Fuzzy IF–THEN

Selain himpunan fuzzy dan fungsi keanggotaan, komponen penting lain dalam *fuzzy logic* adalah aturan fuzzy. Aturan ini biasanya ditulis dalam bentuk IF–THEN. Aturan fuzzy memungkinkan sistem melakukan penalaran berdasarkan kondisi tertentu.

Contoh aturan fuzzy dalam prediksi konsumsi energi:

- **IF** konsumsi energi rendah **THEN** status beban aman.
- **IF** konsumsi energi sedang **THEN** status beban perlu dipantau.
- **IF** konsumsi energi tinggi **THEN** status beban perlu dikendalikan.

Aturan tersebut sederhana, tetapi mudah dipahami oleh pengguna. Inilah salah satu kelebihan utama *fuzzy logic*. Sistem tidak hanya menghasilkan keluaran numerik, tetapi juga memberikan dasar penalaran yang dapat dijelaskan. Dalam sistem berbasis kecerdasan buatan, aspek keterjelasan ini penting karena pengguna tidak hanya membutuhkan hasil, tetapi juga alasan di balik hasil tersebut.

Model fuzzy juga dapat menggunakan lebih dari satu variabel input. Misalnya, kategori beban energi tidak hanya ditentukan oleh nilai konsumsi energi, tetapi juga waktu penggunaan, suhu ruangan, atau tingkat okupansi. Contohnya:

```
IF konsumsi energi tinggi AND suhu ruangan tinggi
THEN risiko beban puncak tinggi.
```

Aturan seperti ini membuat sistem lebih fleksibel dalam menangani kondisi nyata yang melibatkan banyak faktor.

1.3.5 Fuzzy Logic Sebagai Pelengkap Deep Learning

Dalam prediksi data *time series*, *deep learning* seperti LSTM atau GRU digunakan untuk mempelajari pola historis dan menghasilkan prediksi numerik. Namun, model *deep learning* sering sulit dijelaskan karena proses internalnya kompleks. *Fuzzy logic* dapat digunakan sebagai pelengkap untuk membuat hasil prediksi lebih mudah ditafsirkan.

Integrasi tersebut dapat dipahami dalam dua peran. Pertama, *deep learning* berperan sebagai mesin prediksi. Model ini membaca data masa lalu dan memperkirakan nilai masa depan. Kedua, *fuzzy logic* berperan sebagai mesin interpretasi. Model ini mengubah angka prediksi menjadi kategori dan aturan keputusan.

Misalnya, LSTM memprediksi konsumsi energi satu jam ke depan sebesar 650 kWh. Nilai tersebut kemudian masuk ke sistem fuzzy. Sistem fuzzy membaca nilai tersebut dan menghasilkan keputusan bahwa beban energi berada pada kategori “tinggi” dengan derajat keanggotaan tertentu. Hasil ini lebih mudah digunakan dibandingkan hanya menampilkan angka 650 kWh tanpa penjelasan.

Pendekatan hibrida seperti ini sejalan dengan gagasan *neuro-fuzzy system*, yaitu integrasi antara jaringan saraf tiruan dan logika fuzzy. Jaringan saraf kuat dalam pembelajaran pola, sedangkan logika fuzzy kuat dalam representasi pengetahuan berbasis aturan (Jang et al., 1997; Nauck et al., 1997). Kombinasi keduanya dapat menghasilkan sistem yang tidak hanya prediktif, tetapi juga lebih mudah dipahami.

1.3.6 Kelebihan Dan Keterbatasan Fuzzy Logic

Fuzzy logic memiliki beberapa kelebihan. Pertama, pendekatan ini mudah dipahami karena menggunakan istilah linguistik seperti rendah, sedang, tinggi, cepat, lambat, besar, atau kecil. Kedua, *fuzzy logic* mampu menangani ketidakpastian dan nilai yang tidak sepenuhnya tegas. Ketiga, aturan fuzzy dapat disusun berdasarkan pengetahuan pakar atau hasil analisis data.

Namun, *fuzzy logic* juga memiliki keterbatasan. Sistem fuzzy membutuhkan rancangan fungsi keanggotaan dan aturan yang tepat. Jika fungsi keanggotaan dibuat sembarangan, hasil keputusan dapat menjadi kurang akurat. Selain itu, ketika jumlah variabel input terlalu banyak, jumlah aturan fuzzy dapat meningkat secara signifikan. Hal ini membuat sistem menjadi lebih kompleks dan sulit dikelola.

Oleh karena itu, penggunaan *fuzzy logic* dalam prediksi data *time series* perlu dirancang secara hati-hati. Fungsi keanggotaan harus disesuaikan dengan karakteristik data, sedangkan aturan fuzzy harus relevan dengan tujuan pengambilan keputusan. Dalam model hibrida LSTM–Fuzzy Logic, proses evaluasi tidak hanya menilai akurasi prediksi numerik, tetapi juga menilai apakah kategori fuzzy yang dihasilkan masuk akal dan bermanfaat.

1.4 Hubungan Deep Learning, Fuzzy Logic, Dan Time Series

Deep learning, *fuzzy logic*, dan data *time series* memiliki peran yang saling melengkapi dalam sistem prediksi modern. Data *time series* menyediakan informasi historis yang tersusun berdasarkan waktu. *Deep learning* digunakan untuk mempelajari pola dari urutan data tersebut, sedangkan *fuzzy logic* membantu menerjemahkan hasil prediksi menjadi informasi yang lebih mudah dipahami. Hubungan ketiganya penting karena sistem prediksi tidak

hanya perlu menghasilkan angka yang akurat, tetapi juga perlu memberikan makna keputusan yang jelas.

Dalam prediksi data *time series*, model perlu memahami pola waktu seperti tren, musiman, siklus, dan perubahan acak. Pola tersebut menjadi dasar penting dalam memilih metode prediksi yang sesuai (Hyndman & Athanasopoulos, 2021). Pada kasus yang sederhana, metode statistik dapat digunakan. Namun, ketika data bersifat nonlinier, besar, fluktuatif, dan memiliki dependensi jangka panjang, pendekatan *deep learning* menjadi lebih relevan karena mampu mempelajari representasi data secara otomatis (Goodfellow et al., 2016).

1.4.1 Time Series Sebagai Dasar Data Prediksi

Data *time series* adalah data yang dikumpulkan berdasarkan urutan waktu. Setiap nilai dalam data tidak hanya penting sebagai angka, tetapi juga penting karena posisinya dalam urutan. Misalnya, konsumsi energi pukul 08.00 dapat berkaitan dengan konsumsi energi pukul 07.00, pola konsumsi hari sebelumnya, bahkan pola mingguan. Jika urutan waktu diabaikan, sebagian informasi penting dapat hilang.

Karakteristik utama data *time series* adalah adanya ketergantungan temporal. Artinya, nilai masa kini dan masa depan sering dipengaruhi oleh nilai masa lalu. Dalam buku *Forecasting: Principles and Practice*, pola *time series* dijelaskan dapat mengandung tren, musiman, dan siklus. Pengenalan pola-pola ini diperlukan agar metode prediksi dapat menangkap struktur data secara tepat (Hyndman & Athanasopoulos, 2021).

Pada konteks konsumsi energi, pola *time series* dapat muncul dalam bentuk peningkatan beban pada jam kerja, penurunan beban pada malam hari, atau perubahan konsumsi pada akhir pekan. Pola seperti ini tidak selalu linier. Oleh karena itu, model prediksi perlu memiliki kemampuan untuk membaca perubahan pola dari waktu ke waktu.

1.4.2 Deep Learning Sebagai Mesin Pembelajaran Pola

Deep learning berperan sebagai mesin pembelajaran pola. Model ini mempelajari hubungan antara data historis dan nilai yang akan diprediksi melalui lapisan-lapisan jaringan saraf tiruan. Pada data *time series*, model seperti Recurrent Neural Network (RNN), Long Short-Term Memory

(LSTM), dan Gated Recurrent Unit (GRU) sering digunakan karena dirancang untuk memproses data berurutan.

LSTM menjadi salah satu arsitektur penting karena mampu mengatasi kelemahan RNN dasar dalam menyimpan informasi jangka panjang. Melalui mekanisme *gate*, LSTM dapat mengatur informasi mana yang perlu disimpan, dilupakan, atau digunakan dalam prediksi (Hochreiter & Schmidhuber, 1997). Kemampuan ini membuat LSTM sesuai untuk data *time series* yang memiliki pola berulang dan dependensi temporal yang panjang.

Dalam hubungan dengan *time series*, *deep learning* berfungsi untuk menemukan pola tersembunyi dalam data. Misalnya, model dapat mempelajari bahwa konsumsi energi cenderung naik pada jam tertentu, turun pada periode lain, atau berubah karena kombinasi faktor waktu dan perilaku pengguna. Hasil dari proses ini biasanya berupa prediksi numerik, misalnya nilai konsumsi energi satu jam atau satu hari ke depan.

1.4.3 Fuzzy Logic Sebagai Mesin Interpretasi

Jika *deep learning* berperan sebagai mesin pembelajaran pola, maka *fuzzy logic* dapat dipahami sebagai mesin interpretasi. *Fuzzy logic* membantu mengubah hasil prediksi numerik menjadi kategori linguistik, seperti rendah, sedang, tinggi, aman, waspada, atau kritis. Pendekatan ini berguna karena pengguna sering membutuhkan penjelasan yang lebih mudah dipahami daripada angka prediksi saja.

Dasar dari *fuzzy logic* adalah konsep himpunan fuzzy yang memungkinkan suatu nilai memiliki derajat keanggotaan antara 0 dan 1. Konsep ini berbeda dari logika biner yang hanya mengenal benar atau salah. Zadeh (1965) memperkenalkan himpunan fuzzy untuk menangani batas kategori yang tidak tegas, sedangkan Ross (2010) menjelaskan penerapan *fuzzy logic* secara luas dalam bidang rekayasa dan sistem pengambilan keputusan.

Contohnya, prediksi konsumsi energi sebesar 520 kWh tidak harus langsung dikategorikan secara kaku sebagai “tinggi”. Dalam sistem fuzzy, nilai tersebut dapat memiliki derajat keanggotaan tertentu pada kategori “sedang” dan “tinggi”. Dengan demikian, sistem dapat memberikan keputusan yang lebih fleksibel, terutama ketika nilai prediksi berada di sekitar batas kategori.

1.4.4 Integrasi Deep Learning Dan Fuzzy Logic Pada Time Series

Hubungan antara *deep learning*, *fuzzy logic*, dan *time series* dapat dijelaskan melalui alur kerja model hibrida. Pertama, data *time series* dikumpulkan dan diproses menjadi bentuk yang dapat dibaca oleh model. Kedua, model *deep learning* seperti LSTM digunakan untuk mempelajari pola historis dan menghasilkan prediksi numerik. Ketiga, hasil prediksi tersebut dimasukkan ke dalam sistem *fuzzy logic*. Keempat, sistem fuzzy menghasilkan kategori dan rekomendasi keputusan.

Alur tersebut menunjukkan bahwa *deep learning* dan *fuzzy logic* tidak saling menggantikan, tetapi saling melengkapi. *Deep learning* kuat dalam pembelajaran pola dari data besar dan kompleks, sedangkan *fuzzy logic* kuat dalam penalaran berbasis aturan dan interpretasi linguistik. Buku tentang *neuro-fuzzy* menjelaskan bahwa integrasi jaringan saraf dan logika fuzzy merupakan bagian dari *computational intelligence* yang menggabungkan kemampuan belajar dari data dengan representasi pengetahuan yang lebih dapat dijelaskan (Jang et al., 1997).

Dalam model LSTM–Fuzzy Logic, LSTM menjawab pertanyaan "berapa nilai yang diprediksi?", sedangkan *fuzzy logic* menjawab "apa arti nilai tersebut?". Misalnya, LSTM memprediksi konsumsi energi sebesar 700 kWh. Sistem fuzzy kemudian mengolah nilai tersebut dan menghasilkan kategori "beban tinggi" dengan rekomendasi pengendalian penggunaan energi. Dengan cara ini, sistem prediksi menjadi lebih mudah digunakan dalam pengambilan keputusan.

1.4.5 Manfaat Integrasi Dalam Sistem Prediksi

Integrasi *deep learning* dan *fuzzy logic* memberikan beberapa manfaat. Pertama, sistem dapat memanfaatkan kekuatan *deep learning* dalam menangkap pola temporal yang kompleks. Kedua, sistem dapat menghasilkan interpretasi yang lebih mudah dipahami melalui kategori fuzzy. Ketiga, hasil prediksi dapat dikaitkan dengan aturan keputusan yang lebih praktis.

Pada prediksi konsumsi energi, manfaat tersebut dapat terlihat dari keluaran sistem. Model tidak hanya menampilkan angka prediksi, tetapi juga

memberikan status beban energi. Misalnya, prediksi konsumsi dapat dikategorikan sebagai "rendah", "sedang", atau "tinggi". Kategori ini dapat dilengkapi dengan rekomendasi, seperti memantau beban, menunda penggunaan perangkat tertentu, atau memberikan peringatan beban puncak.

Meskipun demikian, integrasi ini tetap membutuhkan perancangan yang teliti. Model *deep learning* perlu dilatih dan dievaluasi dengan data yang representatif. Sistem fuzzy juga perlu memiliki fungsi keanggotaan dan aturan yang sesuai dengan konteks masalah. Jika fungsi keanggotaan atau aturan fuzzy tidak tepat, interpretasi yang dihasilkan dapat menyesatkan meskipun prediksi numeriknya cukup baik.

1.4.6 Contoh Hubungan Ketiganya Dalam Prediksi Konsumsi Energi

Sebagai contoh, sebuah sistem memiliki data konsumsi energi per jam selama beberapa bulan. Data tersebut diproses menjadi *window time series*, misalnya 24 jam terakhir digunakan untuk memprediksi konsumsi energi satu jam berikutnya. Model LSTM kemudian mempelajari pola dari data tersebut dan menghasilkan nilai prediksi.

Nilai prediksi selanjutnya masuk ke sistem *fuzzy logic*. Jika hasil prediksi berada pada rentang rendah, sistem memberikan status "aman". Jika berada pada rentang sedang, sistem memberikan status "perlu dipantau". Jika berada pada rentang tinggi, sistem memberikan status "perlu dikendalikan". Alur ini membuat hasil prediksi lebih dekat dengan kebutuhan pengguna karena angka prediksi diubah menjadi informasi keputusan.

Contoh tersebut menunjukkan hubungan yang jelas: *time series* adalah sumber data, *deep learning* adalah metode pembelajaran pola, dan *fuzzy logic* adalah metode interpretasi hasil. Ketiganya dapat membentuk sistem prediksi yang tidak hanya menghitung nilai masa depan, tetapi juga membantu memahami makna dari nilai tersebut.

1.5 Studi Kasus Umum Prediksi Data Time Series

Prediksi data *time series* banyak digunakan untuk membantu organisasi memahami pola masa lalu dan memperkirakan kondisi masa depan. Data *time series* memiliki ciri utamaberupa urutan waktu, sehingga nilai pada satu

periode sering berkaitan dengan periode sebelumnya. Dalam praktiknya, prediksi jenis ini digunakan pada berbagai bidang, seperti energi, ekonomi, kesehatan, transportasi, lingkungan, industri, dan teknologi informasi. Tujuannya bukan hanya memperkirakan angka, tetapi juga mendukung pengambilan keputusan yang lebih tepat.

Salah satu studi kasus yang sering digunakan adalah **prediksi konsumsi energi listrik**. Data konsumsi energi biasanya dicatat per jam, per hari, atau per bulan. Polanya dapat dipengaruhi oleh aktivitas pengguna, jam kerja, cuaca, penggunaan perangkat elektronik, dan perubahan musim. Model *deep learning*, khususnya LSTM, dapat digunakan untuk mempelajari pola konsumsi dari data historis dan memprediksi kebutuhan energi pada periode berikutnya. Hasil prediksi ini bermanfaat untuk mengatur beban listrik, mengurangi pemborosan, dan memberikan peringatan ketika konsumsi energi diperkirakan tinggi. LSTM banyak digunakan pada data sekuensial karena mampu mempelajari dependensi jangka panjang yang sulit ditangkap oleh RNN dasar (Hochreiter & Schmidhuber, 1997).

Studi kasus lain adalah **prediksi penjualan produk**. Perusahaan dapat menggunakan data penjualan harian atau bulanan untuk memperkirakan permintaan pada periode berikutnya. Prediksi ini membantu perusahaan menentukan jumlah stok, jadwal produksi, strategi promosi, dan kebutuhan distribusi. Misalnya, toko ritel dapat memprediksi peningkatan penjualan menjelang hari libur atau akhir bulan. Jika prediksi menunjukkan permintaan tinggi, perusahaan dapat menambah stok agar tidak terjadi kekurangan barang. Sebaliknya, jika permintaan diprediksi rendah, perusahaan dapat mengurangi pembelian stok agar tidak terjadi penumpukan barang.

Dalam bidang kesehatan, prediksi data *time series* dapat digunakan untuk memperkirakan **jumlah pasien**, **kebutuhan obat**, atau **pola penyebaran penyakit**. Rumah sakit dapat memanfaatkan data kunjungan pasien dari waktu ke waktu untuk memperkirakan kebutuhan tenaga medis dan fasilitas layanan. Pada kasus penyakit menular, data historis dapat digunakan untuk melihat kecenderungan peningkatan atau penurunan kasus. Prediksi seperti ini tidak selalu sempurna, tetapi dapat menjadi dasar awal untuk perencanaan sumber daya kesehatan.

Pada bidang transportasi, data *time series* digunakan untuk memprediksi **kepadatan lalu lintas**, **jumlah penumpang**, atau **waktu tempuh**

perjalanan. Sistem transportasi modern sering mengumpulkan data dari sensor jalan, GPS, kamera lalu lintas, dan aplikasi navigasi. Data tersebut dapat dianalisis untuk memperkirakan jam sibuk, titik kemacetan, atau kebutuhan armada. Prediksi ini membantu pengelola transportasi membuat keputusan yang lebih cepat, misalnya mengatur jadwal kendaraan, mengalihkan rute, atau memberikan informasi perjalanan kepada pengguna.

Selain *deep learning*, *fuzzy logic* dapat digunakan untuk membantu interpretasi hasil prediksi. Misalnya, hasil prediksi konsumsi energi sebesar 700 kWh dapat diterjemahkan menjadi kategori "beban tinggi". Prediksi jumlah pasien dapat dikategorikan menjadi "normal", "padat", atau "sangat padat". Kategori seperti ini lebih mudah dipahami oleh pengguna dibandingkan hanya menampilkan angka. *Fuzzy logic* berguna karena mampu menangani batas kategori yang tidak tegas dan memberikan penalaran berbasis istilah linguistik (Zadeh, 1965; Ross, 2010).

Tabel 1.2: Contoh Studi Kasus Prediksi Data Time Series

Bidang	Contoh Data	Tujuan Prediksi	Manfaat
Energi	Konsumsi listrik per jam	Memperkirakan beban energi	Efisiensi dan pengendalian beban
Penjualan	Transaksi harian	Memperkirakan permintaan produk	Pengelolaan stok
Kesehatan	Kunjungan pasien	Memperkirakan kebutuhan layanan	Perencanaan tenaga dan fasilitas
Transportasi	Kepadatan lalu lintas	Memperkirakan kemacetan	Pengaturan rute dan jadwal
Lingkungan	Suhu dan curah hujan	Memperkirakan kondisi cuaca	Mitigasi risiko lingkungan

Studi kasus tersebut menunjukkan bahwa prediksi data *time series* memiliki cakupan luas. Perbedaan bidang hanya memengaruhi jenis data dan tujuan penggunaannya, sedangkan prinsip dasarnya tetap sama: data masa lalu digunakan untuk memperkirakan kondisi masa depan. Integrasi *deep learning* dan *fuzzy logic* dapat memperkuat proses tersebut karena *deep learning* berperan dalam mempelajari pola data, sedangkan *fuzzy logic* membantu menerjemahkan hasil prediksi menjadi informasi yang lebih mudah digunakan.

Bab 2
Konsep Dasar Data Time Series

2.1 Pengertian Data Time Series

Data *time series* adalah data yang dikumpulkan, dicatat, atau diamati berdasarkan urutan waktu. Setiap nilai dalam data memiliki penanda waktu tertentu, misalnya detik, menit, jam, hari, bulan, atau tahun. Karena tersusun berdasarkan waktu, urutan data menjadi bagian yang sangat penting. Jika urutan waktunya diubah atau diacak, maka pola yang terkandung di dalam data dapat hilang atau berubah maknanya.

Contoh sederhana data *time series* adalah konsumsi listrik rumah setiap jam, suhu udara harian, jumlah transaksi penjualan per bulan, jumlah pengunjung situs web per hari, harga saham per menit, atau jumlah pasien rumah sakit per minggu. Semua contoh tersebut memiliki kesamaan, yaitu nilai data dicatat mengikuti urutan waktu. Dalam analisis *time series*, waktu bukan sekadar keterangan tambahan, tetapi menjadi unsur utama dalam memahami pola data (Hyndman & Athanasopoulos, 2021).

Secara sederhana, data *time series* dapat dituliskan sebagai berikut:

$$y_1, y_2, y_3, \dots, y_t$$

Keterangan:

y_t menunjukkan nilai data pada waktu ke-t.

t menunjukkan urutan waktu pengamatan.

Nilai y_t dapat dipengaruhi oleh nilai-nilai sebelumnya, seperti y_{t-1} , y_{t-2} , dan seterusnya.

Sebagai contoh, jika y_t adalah konsumsi listrik pada pukul 10.00, maka yt−1y_{t-1}yt−1 dapat menunjukkan konsumsi listrik pada pukul 09.00. Hubungan antarwaktu seperti ini disebut dependensi temporal. Dependensi temporal berarti nilai saat ini sering memiliki keterkaitan dengan nilai masa lalu. Inilah yang membedakan data *time series* dari data biasa yang tidak memperhatikan urutan waktu.

Data *time series* berbeda dengan data *cross-sectional*. Data *cross-sectional* dikumpulkan pada satu waktu tertentu dari banyak objek, misalnya data tinggi badan 100 mahasiswa pada hari yang sama. Pada data tersebut, urutan pencatatan biasanya tidak terlalu penting. Sebaliknya, pada data *time series*, urutan waktu sangat menentukan. Data konsumsi listrik hari Senin, Selasa, dan Rabu tidak dapat ditukar begitu saja karena setiap hari dapat memiliki pola penggunaan yang berbeda.

Dalam praktiknya, data *time series* sering memiliki pola tertentu. Pola tersebut dapat berupa tren, musiman, siklus, dan variasi acak. Tren menunjukkan kecenderungan naik atau turun dalam jangka panjang. Musiman menunjukkan pola yang berulang dalam periode tertentu, misalnya peningkatan penjualan menjelang hari raya. Siklus menunjukkan perubahan yang berulang tetapi tidak selalu memiliki periode tetap. Sementara itu, variasi acak atau *noise* merupakan perubahan yang tidak mudah diprediksi karena dipengaruhi oleh faktor yang tidak selalu terlihat dalam data (Box et al., 2015).

Pemahaman terhadap data *time series* penting karena banyak keputusan di dunia nyata bergantung pada kemampuan memperkirakan masa depan. Perusahaan membutuhkan prediksi penjualan untuk mengatur stok. Rumah sakit membutuhkan prediksi jumlah pasien untuk menyiapkan tenaga medis. Pengelola energi membutuhkan prediksi konsumsi listrik untuk menghindari beban berlebih. Pemerintah membutuhkan prediksi cuaca, inflasi, atau kebutuhan transportasi untuk menyusun kebijakan.

Pada bidang kecerdasan buatan, data *time series* menjadi objek penting karena dapat dipelajari menggunakan model prediktif. Model seperti LSTM, GRU, dan arsitektur *deep learning* lainnya dapat digunakan untuk mengenali pola historis dan memperkirakan nilai pada waktu berikutnya. Namun, sebelum membangun model prediksi, data perlu dipahami terlebih dahulu. Kesalahan dalam memahami struktur waktu, pola data, atau interval pencatatan dapat menyebabkan hasil prediksi menjadi tidak akurat.

Dengan demikian, data *time series* dapat dipahami sebagai data yang tidak hanya memiliki nilai, tetapi juga memiliki urutan waktu yang bermakna. Analisis terhadap data ini bertujuan menemukan pola masa lalu, memahami perubahan yang terjadi, dan memperkirakan kemungkinan nilai di masa depan. Pemahaman dasar ini menjadi fondasi penting sebelum membahas

pra-pemrosesan, pemodelan, evaluasi, dan integrasi *deep learning* dengan *fuzzy logic* pada bab-bab berikutnya.

2.2 Komponen Time Series: Tren, Musiman, Siklus, Dan Noise

Data *time series* biasanya tidak bergerak secara acak sepenuhnya. Di dalamnya sering terdapat pola tertentu yang dapat diamati dari waktu ke waktu. Pola tersebut dapat berupa kecenderungan naik atau turun, pengulangan pada periode tertentu, perubahan jangka panjang yang tidak selalu tetap, serta gangguan acak yang sulit diprediksi. Komponen-komponen ini penting dipahami karena menjadi dasar dalam analisis dan pemodelan prediksi data *time series*.

Secara umum, data *time series* dapat dipahami sebagai gabungan dari beberapa komponen utama, yaitu **tren**, **musiman**, **siklus**, dan **noise**. Pemahaman terhadap komponen ini membantu peneliti menentukan metode analisis yang tepat. Misalnya, data yang memiliki pola musiman kuat perlu diperlakukan berbeda dari data yang hanya memiliki tren naik secara perlahan. Dalam konteks prediksi menggunakan *deep learning*, pemahaman terhadap komponen *time series* juga membantu dalam proses pra-pemrosesan, pemilihan panjang *window*, evaluasi model, dan interpretasi hasil prediksi (Hyndman & Athanasopoulos, 2021).

2.2.1 Tren

Tren adalah kecenderungan umum data dalam jangka panjang. Tren dapat menunjukkan pola meningkat, menurun, atau relatif stabil. Misalnya, konsumsi energi listrik pada suatu gedung dapat meningkat dari tahun ke tahun karena bertambahnya jumlah perangkat elektronik. Sebaliknya, konsumsi energi dapat menurun apabila gedung menerapkan sistem hemat energi atau menggunakan perangkat yang lebih efisien.

Tren tidak harus selalu berbentuk garis lurus. Pada beberapa kasus, tren dapat bergerak secara perlahan, cepat, atau berubah arah pada periode tertentu. Contohnya, jumlah pengguna aplikasi dapat meningkat tajam pada awal peluncuran, kemudian melambat setelah pasar mulai jenuh. Dalam analisis *time series*, tren membantu menunjukkan arah umum perkembangan data.

Pada prediksi konsumsi energi, tren dapat digunakan untuk melihat apakah kebutuhan energi cenderung meningkat dari waktu ke waktu. Jika tren meningkat, maka pengelola energi perlu menyiapkan strategi pengendalian beban. Jika tren menurun, perlu dianalisis apakah penurunan tersebut disebabkan oleh efisiensi energi, perubahan aktivitas pengguna, atau faktor lain.

2.2.2 Musiman

Musiman atau *seasonality* adalah pola yang berulang secara teratur dalam periode tertentu. Pola ini dapat muncul harian, mingguan, bulanan, kuartalan, atau tahunan. Contohnya, konsumsi listrik kantor biasanya meningkat pada jam kerja dan menurun pada malam hari. Penjualan produk tertentu dapat meningkat menjelang hari raya. Jumlah penumpang transportasi umum dapat naik pada jam berangkat dan pulang kerja.

Ciri utama pola musiman adalah adanya pengulangan dengan interval yang relatif tetap. Jika suatu pola selalu muncul setiap hari, setiap minggu, atau setiap bulan, maka pola tersebut dapat disebut sebagai pola musiman. Dalam buku *Forecasting: Principles and Practice*, pola musiman dijelaskan sebagai variasi yang muncul pada periode tetap dan biasanya berkaitan dengan kalender, cuaca, atau kebiasaan manusia (Hyndman & Athanasopoulos, 2021).

Pada data konsumsi energi, pola musiman sangat sering ditemukan. Misalnya, penggunaan pendingin ruangan dapat meningkat pada siang hari, penggunaan lampu meningkat pada malam hari, dan konsumsi listrik gedung perkantoran lebih rendah pada akhir pekan. Jika pola musiman tidak diperhatikan, model prediksi dapat menghasilkan perkiraan yang kurang akurat karena gagal menangkap pengulangan perilaku penggunaan energi.

2.2.3 Siklus

Siklus adalah pola naik turun yang terjadi dalam jangka waktu lebih panjang, tetapi tidak selalu memiliki periode yang tetap. Siklus berbeda dari musiman. Musiman memiliki pola pengulangan yang relatif teratur, sedangkan siklus lebih fleksibel dan sering dipengaruhi oleh faktor ekonomi, sosial, kebijakan, atau kondisi lingkungan.

Contoh siklus dapat ditemukan pada data ekonomi. Permintaan barang dapat meningkat ketika kondisi ekonomi membaik dan menurun ketika terjadi perlambatan ekonomi. Pada sektor energi, siklus dapat terjadi karena perubahan aktivitas industri, kebijakan tarif listrik, perubahan pola kerja masyarakat, atau perubahan penggunaan teknologi.

Perbedaan antara musiman dan siklus sering membingungkan. Cara sederhananya, musiman biasanya mengikuti kalender yang lebih jelas, seperti harian, mingguan, atau tahunan. Siklus tidak selalu mengikuti kalender tetap dan durasinya bisa berubah. Misalnya, peningkatan konsumsi listrik setiap siang hari termasuk pola musiman harian. Namun, peningkatan kebutuhan energi selama beberapa tahun akibat pertumbuhan kawasan industri dapat dianggap sebagai siklus atau perubahan jangka panjang yang lebih kompleks.

2.2.4 Noise

Noise adalah variasi acak dalam data yang tidak dapat dijelaskan secara langsung oleh tren, musiman, atau siklus. *Noise* dapat muncul karena kesalahan pencatatan, gangguan sensor, kejadian mendadak, perubahan perilaku pengguna, atau faktor eksternal yang tidak tercatat dalam dataset. Dalam data nyata, *noise* hampir selalu ada.

Pada data konsumsi energi, *noise* dapat muncul ketika terjadi pemadaman listrik sementara, gangguan alat ukur, perubahan aktivitas mendadak, atau penggunaan perangkat listrik di luar pola normal. Misalnya, sebuah gedung biasanya memiliki konsumsi listrik stabil pada malam hari, tetapi tiba-tiba meningkat karena ada kegiatan tambahan. Jika kejadian tersebut tidak tercatat sebagai informasi tambahan, maka model hanya melihatnya sebagai perubahan acak.

Noise menjadi tantangan penting dalam prediksi data *time series*. Jika model terlalu mengikuti *noise*, maka model dapat mengalami *overfitting*. Artinya, model terlihat sangat baik pada data latih, tetapi buruk ketika digunakan untuk data baru. Oleh karena itu, proses pra-pemrosesan, normalisasi, deteksi pencilan, dan pemilihan model perlu dilakukan dengan hati-hati (Box et al., 2015).

2.2.5 Model Dekomposisi Time Series

Untuk memahami hubungan antar-komponen, data *time series* sering dijelaskan menggunakan konsep dekomposisi. Dekomposisi berarti memecah data menjadi beberapa bagian agar pola di dalamnya lebih mudah dianalisis. Dua bentuk dekomposisi yang umum digunakan adalah model aditif dan model multiplikatif.

Model aditif dapat ditulis sebagai berikut:

$$Y_t = T_t + S_t + C_t + R_t$$

Keterangan:

Y_t = nilai data pada waktu ke-ttt

T_t = komponen tren

S_t = komponen musiman

C_t = komponen siklus

R_t = komponen acak atau *noise*

Model aditif digunakan ketika pengaruh masing-masing komponen dianggap saling menjumlah. Misalnya, peningkatan musiman relatif sama besar dari waktu ke waktu. Sementara itu, model multiplikatif dapat ditulis sebagai berikut:

$$Y_t = T_t \times S_t \times C_t \times R_t$$

Model multiplikatif digunakan ketika besar variasi musiman atau fluktuasi berubah mengikuti level data. Misalnya, ketika nilai penjualan semakin besar, variasi musiman juga semakin besar. Pemilihan model aditif atau multiplikatif bergantung pada karakteristik data yang dianalisis (Chatfield, 2003).

Dalam pemodelan berbasis *deep learning*, dekomposisi tidak selalu dilakukan secara manual. Model seperti LSTM dan GRU dapat belajar dari pola historis secara langsung. Namun, pemahaman terhadap komponen *time series* tetap penting karena membantu menjelaskan mengapa data berubah, mengapa model menghasilkan prediksi tertentu, dan mengapa kesalahan prediksi dapat terjadi pada periode tertentu.

2.3 Karakteristik Data Time Series

Data *time series* memiliki karakteristik yang berbeda dari data biasa karena setiap nilai terikat pada urutan waktu. Pada data tabular biasa, seperti daftar nama mahasiswa, alamat, atau nilai ujian, urutan baris sering tidak terlalu berpengaruh. Namun, pada data *time series*, urutan menjadi bagian penting dari makna data. Nilai hari ini dapat berkaitan dengan nilai kemarin, minggu lalu, atau periode sebelumnya. Oleh karena itu, analisis *time series* harus memperhatikan waktu sebagai struktur utama data, bukan sekadar informasi tambahan.

Karakteristik pertama dari data *time series* adalah **ketergantungan temporal**. Ketergantungan temporal berarti nilai pada waktu tertentu dapat dipengaruhi oleh nilai sebelumnya. Misalnya, konsumsi listrik pukul 10.00 mungkin berkaitan dengan konsumsi listrik pukul 09.00 dan 08.00. Begitu pula harga saham hari ini dapat dipengaruhi oleh harga pada hari sebelumnya. Dalam analisis *time series*, hubungan seperti ini sering disebut sebagai *autocorrelation*, yaitu korelasi antara suatu nilai dengan nilai pada periode sebelumnya (Box et al., 2015).

Secara sederhana, hubungan antarwaktu dapat digambarkan dengan konsep *lag*. Jika y_t adalah nilai pada waktu ke-ttt, maka y_{t-1} adalah nilai satu periode sebelumnya, y_{t-2} adalah nilai dua periode sebelumnya, dan seterusnya. Konsep ini penting karena banyak model prediksi menggunakan nilai masa lalu untuk memperkirakan nilai masa depan.

$$y_t \rightarrow y_{t-1}, y_{t-2}, y_{t-3}, \dots$$

Notasi tersebut menunjukkan bahwa nilai saat ini dapat dianalisis dengan melihat nilai-nilai sebelumnya. Pada model *deep learning* seperti LSTM atau GRU, nilai-nilai masa lalu biasanya disusun dalam bentuk *sequence* atau *window* agar model dapat mempelajari pola temporal.

Karakteristik kedua adalah **interval waktu**. Data *time series* dapat dicatat dalam interval yang tetap atau tidak tetap. Interval tetap berarti data dicatat secara konsisten, misalnya setiap jam, setiap hari, atau setiap bulan. Contohnya adalah data suhu harian yang selalu dicatat pukul 07.00. Sebaliknya, interval tidak tetap berarti pencatatan data tidak selalu terjadi pada jarak waktu yang sama. Data transaksi pelanggan, misalnya, dapat terjadi kapan saja sesuai aktivitas pembelian. Interval waktu yang tidak

konsisten perlu diperhatikan karena dapat memengaruhi proses analisis dan pemodelan.

Karakteristik ketiga adalah **adanya pola waktu**. Data *time series* sering mengandung tren, musiman, siklus, dan *noise*. Tren menunjukkan arah perubahan jangka panjang. Musiman menunjukkan pola yang berulang secara tetap, seperti peningkatan konsumsi listrik pada siang hari atau peningkatan penjualan menjelang hari raya. Siklus menunjukkan perubahan naik turun dalam jangka panjang yang tidak selalu memiliki periode tetap. Sementara itu, *noise* adalah gangguan acak yang sulit dijelaskan secara langsung. Pemahaman terhadap pola ini penting karena model prediksi yang baik harus mampu menangkap struktur utama data, bukan hanya mengikuti perubahan acak (Hyndman & Athanasopoulos, 2021).

Karakteristik keempat adalah **stasioneritas**. Data disebut stasioner apabila pola statistiknya relatif stabil dari waktu ke waktu, seperti rata-rata dan varians yang tidak banyak berubah. Sebaliknya, data tidak stasioner memiliki pola yang berubah, misalnya rata-rata terus meningkat atau fluktuasi semakin besar. Banyak metode statistik klasik membutuhkan data yang stasioner agar dapat bekerja dengan baik. Namun, dalam praktiknya, banyak data nyata justru tidak stasioner. Data konsumsi energi, harga barang, dan jumlah pengguna aplikasi sering mengalami perubahan pola karena dipengaruhi oleh kebiasaan pengguna, kondisi lingkungan, atau perubahan sistem (Chatfield, 2003).

Karakteristik kelima adalah **adanya nilai ekstrem dan data hilang**. Data *time series* yang berasal dari sensor, transaksi, atau sistem digital sering mengandung nilai yang tidak normal. Nilai ekstrem dapat terjadi karena kejadian khusus, kesalahan pencatatan, gangguan alat, atau perubahan mendadak. Misalnya, konsumsi energi tiba-tiba melonjak karena ada acara besar di sebuah gedung. Data hilang juga sering terjadi, terutama ketika sensor tidak aktif, jaringan terganggu, atau sistem gagal mencatat data. Jika tidak ditangani, nilai ekstrem dan data hilang dapat mengganggu proses pelatihan model.

Karakteristik keenam adalah **urutan data tidak boleh diacak sembarangan**. Pada banyak kasus *machine learning*, data dapat dibagi menjadi data latih dan data uji secara acak. Namun, pada data *time series*, pembagian data perlu mempertahankan urutan waktu. Data masa lalu digunakan untuk pelatihan, sedangkan data periode berikutnya digunakan

untuk pengujian. Jika data diacak, model dapat "melihat" informasi dari masa depan saat pelatihan, sehingga hasil evaluasi menjadi tidak valid. Kesalahan ini sering disebut sebagai *data leakage* dalam pemodelan prediktif.

Karakteristik ketujuh adalah **dapat berbentuk univariat atau multivariat**. Data *time series* univariat hanya memiliki satu variabel utama, misalnya konsumsi listrik per jam. Data *time series* multivariat memiliki lebih dari satu variabel yang diamati dalam waktu yang sama, misalnya konsumsi listrik, suhu ruangan, kelembapan, jumlah penghuni, dan waktu operasional gedung. Data multivariat dapat memberikan informasi yang lebih kaya, tetapi juga membutuhkan pemrosesan dan pemodelan yang lebih hati-hati.

Pemahaman terhadap karakteristik data *time series* sangat penting sebelum membangun model prediksi. Model yang baik tidak hanya bergantung pada algoritma, tetapi juga pada pemahaman terhadap struktur data. Jika karakteristik data diabaikan, model dapat menghasilkan prediksi yang tampak baik pada data latih, tetapi lemah ketika digunakan pada data baru. Oleh karena itu, analisis awal terhadap ketergantungan temporal, interval waktu, pola data, stasioneritas, *noise*, nilai hilang, dan bentuk variabel menjadi langkah penting dalam pemodelan *time series*.

2.4 Data Stasioner Dan Tidak Stasioner

Dalam analisis *time series*, salah satu konsep penting yang perlu dipahami adalah **stasioneritas**. Stasioneritas berkaitan dengan kestabilan pola statistik data dari waktu ke waktu. Data disebut **stasioner** apabila karakteristik utamanya, seperti rata-rata, varians, dan pola fluktuasi, relatif tidak berubah sepanjang waktu. Sebaliknya, data disebut **tidak stasioner** apabila karakteristik tersebut berubah, misalnya rata-rata semakin naik, fluktuasi semakin besar, atau terdapat pola musiman yang kuat.

Konsep ini penting karena banyak metode prediksi klasik, seperti ARIMA, bekerja lebih baik ketika data berada dalam kondisi stasioner. Jika data tidak stasioner, model dapat kesulitan mengenali pola yang stabil karena struktur data terus berubah dari waktu ke waktu. Oleh karena itu, sebelum membangun model prediksi, analis perlu memeriksa apakah data yang digunakan bersifat stasioner atau tidak stasioner (Box et al., 2015).

2.4.1 Pengertian Data Stasioner

Data stasioner adalah data *time series* yang pola statistiknya relatif konstan. Artinya, nilai rata-rata data tidak menunjukkan kenaikan atau penurunan yang jelas dalam jangka panjang. Selain itu, tingkat penyebaran data atau varians juga tidak berubah secara ekstrem dari waktu ke waktu. Jika data memiliki fluktuasi, fluktuasi tersebut masih berada di sekitar rata-rata yang stabil.

Contoh sederhana data stasioner adalah suhu ruangan laboratorium yang dikendalikan oleh pendingin ruangan, sehingga nilainya hanya berfluktuasi kecil di sekitar suhu tertentu. Misalnya, suhu bergerak di sekitar 24–26°C sepanjang hari. Walaupun nilainya berubah, perubahan tersebut tidak menunjukkan tren naik atau turun yang kuat.

Secara sederhana, data stasioner dapat dipahami melalui dua ciri utama:

$$E(y_t) =$$

$$Var(y_t) = \sigma^2$$

Keterangan:

- $E(y_t)$ menunjukkan rata-rata data pada waktu ke-ttt.
- μ menunjukkan rata-rata yang relatif tetap.
- $Var(y_t)$ menunjukkan varians data pada waktu ke-ttt.
- σ2\sigma^2σ2 menunjukkan varians yang relatif tetap.

Rumus tersebut tidak perlu dipahami secara rumit. Intinya, data stasioner memiliki rata-rata dan variasi yang tidak berubah secara besar dari waktu ke waktu. Dalam grafik, data stasioner biasanya terlihat berfluktuasi di sekitar satu garis tengah yang relatif stabil.

2.4.2 Pengertian Data Tidak Stasioner

Data tidak stasioner adalah data *time series* yang pola statistiknya berubah seiring waktu. Perubahan ini dapat muncul dalam bentuk tren naik, tren turun, pola musiman, perubahan varians, atau kombinasi dari beberapa pola tersebut. Data nyata lebih sering bersifat tidak stasioner karena dipengaruhi oleh banyak faktor eksternal.

Contoh data tidak stasioner adalah konsumsi listrik suatu kota yang terus meningkat dari tahun ke tahun karena pertumbuhan penduduk dan aktivitas industri. Contoh lain adalah harga barang yang meningkat akibat inflasi, jumlah pengguna aplikasi yang naik setelah promosi besar, atau penjualan produk yang meningkat menjelang hari raya.

Pada grafik, data tidak stasioner sering tampak memiliki arah tertentu. Jika garis data cenderung naik, berarti terdapat tren meningkat. Jika pola naik turun selalu berulang pada periode tertentu, berarti terdapat unsur musiman. Jika fluktuasi data semakin besar dari waktu ke waktu, berarti varians data juga berubah. Pola-pola ini perlu dikenali karena dapat memengaruhi hasil prediksi.

2.4.3 Perbedaan Data Stasioner Dan Tidak Stasioner

Perbedaan antara data stasioner dan tidak stasioner dapat dilihat dari kestabilan pola data. Data stasioner lebih mudah dimodelkan karena perilakunya relatif konsisten. Data tidak stasioner lebih menantang karena model harus memahami perubahan pola yang terjadi dari waktu ke waktu.

Tabel 2.1: Perbedaan Data Stasioner Dan Tidak Stasioner

Aspek	Data Stasioner	Data Tidak Stasioner
Rata-rata	Relatif tetap	Berubah dari waktu ke waktu
Varians	Relatif stabil	Dapat meningkat atau menurun
Tren	Tidak memiliki tren kuat	Sering memiliki tren naik atau turun
Musiman	Tidak dominan	Dapat memiliki pola musiman
Contoh	Suhu ruangan yang stabil	Konsumsi energi yang meningkat tiap tahun
Dampak pada model	Lebih mudah dimodelkan	Perlu pra-pemrosesan tambahan

Tabel tersebut menunjukkan bahwa stasioneritas bukan sekadar istilah statistik, tetapi berkaitan langsung dengan cara data dipahami dan diproses. Jika data tidak stasioner langsung digunakan tanpa analisis awal, model dapat menghasilkan prediksi yang bias atau kurang stabil.

2.4.4 Penyebab Data Menjadi Tidak Stasioner

Data *time series* dapat menjadi tidak stasioner karena beberapa faktor. Pertama, adanya **tren jangka panjang**. Misalnya, konsumsi energi terus meningkat karena jumlah pengguna bertambah. Kedua, adanya **pola musiman**, seperti peningkatan penjualan pada akhir tahun atau peningkatan konsumsi listrik pada siang hari. Ketiga, adanya **perubahan kebijakan atau kondisi eksternal**, misalnya perubahan tarif listrik, perubahan jam kerja, pandemi, atau pergeseran perilaku pengguna.

Selain itu, data juga dapat menjadi tidak stasioner karena perubahan varians. Misalnya, pada awal periode data bergerak stabil, tetapi pada periode berikutnya fluktuasi menjadi semakin besar. Kondisi ini sering ditemukan pada data ekonomi, keuangan, dan energi. Dalam situasi seperti ini, model prediksi perlu memperhatikan bukan hanya arah perubahan data, tetapi juga tingkat ketidakpastiannya.

2.4.5 Cara Menangani Data Tidak Stasioner

Data tidak stasioner dapat diolah agar lebih mudah dianalisis. Salah satu cara yang umum digunakan adalah **differencing**, yaitu menghitung selisih antara nilai saat ini dan nilai sebelumnya. Tujuannya adalah mengurangi pengaruh tren sehingga data menjadi lebih stabil. Secara sederhana, *differencing* dapat ditulis sebagai:

$$z_t = y_t - y_{t-1}$$

Keterangan:

- $z_t$adalah hasil selisih pada waktu ke-ttt.
- y_t adalah nilai data pada waktu ke-ttt.
- y_{t-1} adalah nilai data satu periode sebelumnya.

Selain *differencing*, data tidak stasioner juga dapat ditangani melalui transformasi, seperti transformasi logaritma untuk mengurangi fluktuasi yang terlalu besar. Pada data musiman, dapat digunakan *seasonal differencing*, yaitu menghitung selisih antara nilai saat ini dan nilai pada periode musiman sebelumnya. Misalnya, pada data bulanan, nilai Januari tahun ini dibandingkan dengan Januari tahun sebelumnya.

Pada pendekatan *deep learning*, seperti LSTM dan GRU, data tidak selalu harus dibuat stasioner secara ketat seperti pada metode statistik klasik. Model *deep learning* dapat mempelajari pola nonlinier dan pola temporal yang kompleks secara langsung dari data. Namun, pra-pemrosesan tetap penting. Normalisasi, pembersihan data, pembentukan *window*, dan pemilihan fitur dapat membantumodel belajar denganlebih stabil (Goodfellow et al., 2016).

2.4.6 Pentingnya Stasioneritas Dalam Prediksi Time Series

Stasioneritas membantu analis memahami apakah pola masa lalu dapat digunakan untuk memperkirakan masa depan. Jika pola data relatif stabil, model dapat belajar dengan lebih mudah. Jika pola data berubah terlalu cepat, prediksi menjadi lebih sulit karena hubungan antarwaktu tidak selalu konsisten.

Dalam prediksi konsumsi energi, misalnya, data mungkin memiliki tren meningkat, pola harian, pola mingguan, dan *noise*. Jika semua pola tersebut diabaikan, model dapat salah membaca perubahan data. Oleh karena itu, pemeriksaan stasioneritas menjadi bagian penting dalam analisis awal. Pemeriksaan ini tidak selalu bertujuan membuat semua data menjadi stasioner, tetapi untuk memahami sifat data sebelum menentukan metode pemodelan yang tepat.

Pemahaman tentang data stasioner dan tidak stasioner juga membantu dalam evaluasi model. Kesalahan prediksi yang tinggi pada periode tertentu mungkin bukan hanya disebabkan oleh model yang buruk, tetapi karena data mengalami perubahan pola yang tidak muncul pada data latih. Misalnya, konsumsi energi berubah drastis karena perubahan jam operasional gedung. Situasi seperti ini perlu dianalisis agar perbaikan model dapat dilakukan secara tepat.

2.5 Contoh Data Time Series Dalam Energi, Ekonomi, Kesehatan, Dan IoT

Data *time series* dapat ditemukan pada banyak bidang karena hampir semua aktivitas yang dicatat secara berkala akan menghasilkan data berbasis waktu. Perbedaan bidang hanya memengaruhi jenis data, interval pencatatan, dan tujuan analisisnya. Prinsip dasarnya tetap sama, yaitu data masa lalu

digunakan untuk memahami pola dan memperkirakan kondisi pada periode berikutnya. Dalam pengembangan model prediksi, pemahaman terhadap contoh penerapan ini penting agar pemilihan metode, pra-pemrosesan, dan evaluasi model sesuai dengan karakteristik masalah.

Pada bidang **energi**, data *time series* banyak digunakan untuk memprediksi konsumsi listrik, beban puncak, dan kebutuhan energi pada periode tertentu. Data dapat dicatat per menit, per jam, per hari, atau per bulan melalui *smart meter* dan sensor energi. Misalnya, data konsumsi listrik gedung selama 24 jam dapat digunakan untuk memprediksi konsumsi listrik pada jam berikutnya. Prediksi ini membantu pengelola energi mengatur penggunaan perangkat, menghindari beban berlebih, serta meningkatkan efisiensi operasional. Pada bidang ini, pola harian dan mingguan sering sangat terlihat, seperti kenaikan konsumsi pada jam kerja dan penurunan konsumsi pada malam hari.

Pada bidang **ekonomi dan bisnis**, data *time series* digunakan untuk menganalisis harga saham, inflasi, nilai tukar, penjualan, permintaan produk, dan arus kas. Data penjualan bulanan, misalnya, dapat membantu perusahaan memperkirakan permintaan pada bulan berikutnya. Jika hasil prediksi menunjukkan peningkatan permintaan, perusahaan dapat menambah stok atau memperkuat distribusi. Sebaliknya, jika permintaan diprediksi menurun, perusahaan dapat mengurangi produksi untuk menghindari kelebihan stok. Analisis deret waktu dalam ekonomi penting karena banyak keputusan bisnis bergantung pada kemampuan membaca tren dan perubahan pasar (Box et al., 2015; Hyndman & Athanasopoulos, 2021).

Pada bidang **kesehatan**, data *time series* dapat berupa jumlah pasien harian, tekanan darah berkala, detak jantung, kadar gula darah, penggunaan tempat tidur rumah sakit, atau jumlah kasus penyakit menular dari waktu ke waktu. Data tersebut dapat digunakan untuk memprediksi kebutuhan tenaga medis, obat, ruang rawat, atau alat kesehatan. Misalnya, rumah sakit dapat menggunakan data kunjungan pasien untuk memperkirakan periode padat layanan. Pada sistem pemantauan kesehatan digital, sensor tubuh juga dapat menghasilkan data *time series* yang dianalisis untuk mendeteksi perubahan kondisi pasien secara lebih cepat.

Pada bidang **Internet of Things (IoT)**, data *time series* berasal dari sensor dan perangkat yang mencatat kondisi lingkungan atau aktivitas mesin secara terus-menerus. Contohnya adalah data suhu ruangan, kelembapan, getaran

mesin, kualitas udara, tekanan air, konsumsi bahan bakar, dan status perangkat. Karena IoT menghasilkan data dalam jumlah besar dan interval waktu yang rapat, pendekatan *deep learning* seperti LSTM dan GRU sering digunakan untuk mempelajari pola sekuensial. Data sensor mesin, misalnya, dapat digunakan untuk memprediksi potensi kerusakan sebelum mesin benar-benar berhenti bekerja. Pendekatan ini dikenal sebagai *predictive maintenance*.

Beberapa contoh data *time series* pada berbagai bidang dapat diringkas sebagai berikut.

Tabel 2.2: Contoh Data Time Series Pada Berbagai Bidang

Bidang	Contoh Data	Interval Umum	Tujuan Prediksi
Energi	Konsumsi listrik gedung	Per jam/per hari	Memprediksi beban energi
Ekonomi	Harga saham atau penjualan	Per hari/per bulan	Memprediksi harga atau permintaan
Kesehatan	Jumlah pasien atau detak jantung	Per menit/per hari	Memprediksi kebutuhan layanan atau kondisi pasien
IoT	Suhu, kelembapan, getaran mesin	Per detik/per menit	Memantau kondisi dan mendeteksi anomali

Contoh-contoh tersebut menunjukkan bahwa data *time series* tidak hanya digunakan untuk menghitung nilai masa depan, tetapi juga untuk mendukung pengambilan keputusan. Pada prediksi energi, hasilnya dapat digunakan untuk pengendalian beban. Pada bisnis, hasilnya membantu perencanaan stok. Pada kesehatan, hasilnya membantu kesiapan layanan. Pada IoT, hasilnya membantu pemantauan sistem secara otomatis. Oleh karena itu, pemodelan *time series* menjadi bagian penting dalam pengembangan sistem cerdas berbasis data.

Bab 3
Pra-Pemrosesan Data Time Series

3.1 Pembersihan Data

Pembersihan data merupakan tahap awal yang sangat penting dalam pemodelan data *time series*. Data yang diperoleh dari sistem nyata jarang berada dalam kondisi sempurna. Data dapat mengandung nilai kosong, nilai ganda, kesalahan pencatatan, format waktu yang tidak konsisten, atau nilai ekstrem yang tidak sesuai dengan pola umum. Jika data seperti ini langsung digunakan untuk membangun model, hasil prediksi dapat menjadi tidak akurat karena model belajar dari pola yang keliru.

Pada data *time series*, pembersihan data menjadi lebih sensitif karena setiap nilai terikat pada urutan waktu. Kesalahan pada satu titik waktu dapat memengaruhi pembacaan pola pada periode berikutnya. Misalnya, data konsumsi energi yang seharusnya tercatat setiap jam tiba-tiba memiliki beberapa jam yang kosong. Jika bagian ini tidak diperbaiki, model LSTM atau GRU dapat kesulitan memahami pola konsumsi yang sebenarnya. Data yang tidak bersih juga dapat menyebabkan hasil evaluasi model tampak buruk, bukan karena algoritmanya lemah, tetapi karena kualitas data yang digunakan tidak memadai.

Tahap pembersihan data biasanya dimulai dengan memeriksa struktur dataset. Pemeriksaan ini mencakup nama kolom, tipe data, jumlah baris, format waktu, dan keberadaan nilai kosong. Pada data *time series*, kolom waktu harus dipastikan memiliki format tanggal atau waktu yang benar. Misalnya, data yang dicatat dalam format “01/02/2025” perlu dipastikan apakah berarti 1 Februari 2025 atau 2 Januari 2025. Kesalahan membaca format tanggal dapat mengubah urutan data dan menghasilkan analisis yang keliru.

Masalah lain yang sering muncul adalah **missing value** atau nilai hilang. Nilai hilang dapat terjadi karena sensor tidak aktif, gangguan jaringan, kesalahan input, atau sistem gagal menyimpan data. Pada data konsumsi

energi, nilai kosong dapat muncul ketika *smart meter* tidak mengirim data pada jam tertentu. Penanganannya dapat dilakukan dengan beberapa cara, seperti menghapus baris data, mengisi dengan nilai rata-rata, menggunakan nilai sebelumnya, atau menggunakan teknik interpolasi. Pemilihan metode harus disesuaikan dengan jumlah data hilang dan karakteristik data yang dianalisis (Hyndman & Athanasopoulos, 2021).

Selain nilai hilang, data juga dapat mengandung **duplikasi**. Duplikasi terjadi ketika satu waktu pencatatan memiliki lebih dari satu nilai yang sama atau sangat mirip. Pada data *time series*, duplikasi waktu dapat membingungkan model karena satu titik waktu seharusnya mewakili satu pengamatan tertentu. Jika ditemukan data ganda, peneliti perlu menentukan apakah data tersebut benar-benar duplikat atau merupakan pencatatan dari sumber yang berbeda.

Masalah berikutnya adalah **outlier** atau nilai ekstrem. Outlier adalah nilai yang sangat berbeda dari pola umum data. Misalnya, konsumsi listrik sebuah ruangan biasanya berada pada rentang 100–150 kWh, tetapi pada satu titik tercatat 2.000 kWh. Nilai tersebut perlu diperiksa. Bisa jadi nilai itu benar karena ada aktivitas besar, tetapi bisa juga salah karena gangguan alat ukur. Outlier tidak selalu harus dihapus. Jika outlier merepresentasikan kejadian nyata, maka data tersebut tetap penting. Namun, jika outlier disebabkan oleh kesalahan pencatatan, maka perlu diperbaiki atau dikeluarkan dari dataset (Chandola et al., 2009).

Beberapa masalah umum dalam pembersihan data *time series* dapat diringkas pada tabel berikut.

Tabel 3.1 Masalah Umum Dalam Pembersihan Data Time Series

Masalah Data	Contoh	Penanganan Umum
Nilai hilang	Data konsumsi energi pukul 10.00 tidak tercatat	Interpolasi, nilai sebelumnya, atau rata-rata
Format waktu salah	Tanggal terbaca tidak sesuai format	Standarisasi format tanggal dan waktu
Data duplikat	Dua data pada waktu yang sama	Hapus duplikasi atau ambil nilai rata-rata
Outlier	Lonjakan konsumsi yang tidak wajar	Verifikasi, koreksi, atau pisahkan sebagai kejadian khusus

Masalah Data	Contoh	Penanganan Umum
Tipe data tidak sesuai	Angka terbaca sebagai teks	Konversi tipe data

Pembersihan data tidak boleh dilakukan secara sembarangan. Setiap perubahan pada data perlu memiliki alasan yang jelas. Misalnya, menghapus outlier tanpa pemeriksaan dapat menghilangkan informasi penting tentang kejadian tidak biasa. Sebaliknya, membiarkan nilai salah tetap masuk ke model dapat merusak proses pelatihan. Oleh karena itu, pembersihan data harus dilakukan dengan prinsip hati-hati, terdokumentasi, dan sesuai konteks masalah.

Pada pemodelan berbasis *deep learning*, kualitas data sangat menentukan kualitas model. Model seperti LSTM dapat mempelajari pola temporal yang kompleks, tetapi tetap bergantung pada data yang diberikan. Jika data bersih, konsisten, dan tersusun dengan benar, model memiliki peluang lebih besar untuk menghasilkan prediksi yang baik. Sebaliknya, jika data masih mengandung banyak kesalahan, peningkatan arsitektur model tidak selalu mampu memperbaiki hasil prediksi secara signifikan.

3.2 Penanganan Missing Value

Missing value adalah kondisi ketika sebagian nilai dalam dataset tidak tersedia, kosong, tidak tercatat, atau tidak dapat dibaca oleh sistem. Pada data *time series*, masalah ini sering muncul karena gangguan sensor, koneksi jaringan yang terputus, kesalahan input, kegagalan perangkat, atau proses pencatatan yang tidak berjalan secara konsisten. Dalam konteks data konsumsi energi, misalnya, *smart meter* dapat gagal mengirim data pada jam tertentu sehingga terdapat celah pada urutan waktu.

Penanganan *missing value* sangat penting karena data *time series* bergantung pada urutan waktu. Jika satu atau beberapa titik data hilang, pola temporal dapat terganggu. Model seperti LSTM dan GRU membutuhkan urutan data yang rapi agar dapat mempelajari hubungan antarwaktu dengan baik. Jika nilai hilang dibiarkan tanpa penanganan, model dapat gagal membaca pola, menghasilkan prediksi bias, atau bahkan tidak dapat memproses data karena terdapat nilai kosong pada input.

3.2.1 Penyebab Missing Value Pada Data Time Series

Nilai hilang dapat terjadi karena beberapa penyebab. Pertama, gangguan teknis pada alat pencatat data. Pada sistem berbasis IoT, sensor dapat berhenti bekerja sementara karena kerusakan perangkat, kehabisan daya, atau koneksi internet yang tidak stabil. Kedua, kesalahan sistem penyimpanan data, misalnya database gagal menyimpan data pada waktu tertentu. Ketiga, kesalahan manusia, seperti data yang belum diinput, format data yang salah, atau file yang tidak lengkap.

Pada data *time series*, *missing value* dapat berbentuk satu titik data yang hilang, beberapa titik berurutan yang hilang, atau periode panjang yang tidak memiliki data sama sekali. Kehilangan satu data mungkin masih mudah ditangani. Namun, jika data hilang dalam jumlah besar atau terjadi pada periode penting, proses imputasi menjadi lebih sulit. Oleh karena itu, sebelum memilih metode penanganan, perlu dianalisis terlebih dahulu pola hilangnya data.

3.2.2 Pemeriksaan Missing Value

Langkah pertama dalam menangani *missing value* adalah melakukan pemeriksaan terhadap dataset. Pemeriksaan ini bertujuan mengetahui jumlah nilai hilang, lokasi nilai hilang, dan apakah nilai hilang terjadi secara acak atau membentuk pola tertentu. Pada data konsumsi energi, misalnya, perlu diperiksa apakah data hilang hanya pada beberapa jam tertentu atau selalu terjadi pada waktu yang sama setiap hari.

Pemeriksaan *missing value* tidak cukup hanya menghitung jumlah data kosong. Analis juga perlu melihat posisi waktunya. Jika data hilang terjadi secara acak dan jumlahnya sedikit, metode sederhana seperti interpolasi dapat digunakan. Namun, jika data hilang terjadi dalam periode panjang, maka imputasi menjadi lebih berisiko karena nilai pengganti yang dibuat dapat terlalu jauh dari kondisi sebenarnya.

Secara umum, rasio *missing value* dapat dihitung dengan rumus sederhana berikut:

$$\text{Rasio Missing Value} = \frac{\text{Jumlah Data Hilang}}{\text{Jumlah Seluruh Data}} \times 100\%$$

Rumus tersebut membantu mengetahui seberapa besar masalah data hilang dalam dataset. Jika rasio nilai hilang kecil, proses perbaikan biasanya lebih mudah. Jika rasio nilai hilang besar, perlu dipertimbangkan apakah dataset masih layak digunakan untuk membangun model prediksi.

3.2.3 Metode Penanganan Missing Value

Ada beberapa metode yang dapat digunakan untuk menangani *missing value*. Metode paling sederhana adalah **menghapus data yang hilang**. Cara ini dapat digunakan jika jumlah nilai hilang sangat sedikit dan penghapusan tidak merusak pola data. Namun, pada data *time series*, penghapusan data perlu dilakukan dengan hati-hati karena dapat mengubah interval waktu. Jika data seharusnya dicatat per jam, menghapus satu baris dapat membuat urutan waktu menjadi tidak lengkap.

Metode kedua adalah **mengisi dengan nilai sebelumnya** atau *forward fill*. Pada metode ini, nilai yang hilang diisi menggunakan nilai terakhir yang tersedia. Misalnya, jika data konsumsi energi pukul 10.00 hilang, maka nilainya dapat diisi dengan data pukul 09.00. Metode ini cocok ketika perubahan antarwaktu relatif lambat. Namun, metode ini kurang tepat jika data sangat fluktuatif karena dapat membuat pola terlihat terlalu datar.

Metode ketiga adalah **mengisi dengan nilai setelahnya** atau *backward fill*. Cara ini menggunakan nilai setelah data yang hilang sebagai pengganti. Metode ini sederhana, tetapi perlu berhati-hati ketika digunakan untuk pelatihan model prediksi karena dapat menyebabkan informasi dari masa depan masuk ke data masa lalu. Kondisi ini dikenal sebagai *data leakage* dan dapat membuat hasil evaluasi terlihat lebih baik daripada kondisi sebenarnya.

Metode keempat adalah **mengisi dengan rata-rata, median, atau modus**. Cara ini umum digunakan pada data tabular, tetapi kurang ideal untuk data *time series* karena tidak memperhatikan urutan waktu. Mengisi konsumsi energi yang hilang dengan rata-rata keseluruhan dapat menghilangkan pola jam tertentu, misalnya perbedaan antara konsumsi siang dan malam. Jika metode ini digunakan, sebaiknya rata-rata dihitung berdasarkan konteks waktu, misalnya rata-rata pada jam yang sama atau hari yang sama.

Metode kelima adalah **interpolasi**. Interpolasi mengisi nilai hilang dengan memperkirakan nilai di antara dua titik data yang tersedia. Metode ini cukup

sering digunakan pada data *time series* karena mempertimbangkan nilai sebelum dan sesudah data yang hilang. Interpolasi linear sederhana dapat ditulis sebagai berikut:

$$y_t = y_{t-1} + \frac{(y_{t+1} - y_{t-1})}{2}$$

Keterangan:

- y_t adalah nilai yang hilang pada waktu ke-t.
- y_{t-1} adalah nilai sebelum data hilang.
- y_{t+1} adalah nilai setelah data hilang.

Rumus tersebut digunakan ketika hanya satu titik data yang hilang di antara dua nilai yang tersedia. Untuk kasus yang lebih kompleks, interpolasi dapat dilakukan dengan metode linear, polinomial, spline, atau metode berbasis pola musiman. Interpolasi cukup baik jika data berubah secara bertahap, tetapi kurang tepat jika data mengalami lonjakan mendadak.

Metode keenam adalah **imputasi berbasis model**. Pada pendekatan ini, nilai hilang diperkirakan menggunakan model statistik atau *machine learning*. Misalnya, nilai konsumsi energi yang hilang dapat diprediksi berdasarkan data sebelum, data sesudah, suhu, hari, atau jam operasional. Metode ini lebih fleksibel, tetapi membutuhkan proses yang lebih kompleks dan tetap harus dievaluasi agar tidak menghasilkan nilai pengganti yang menyesatkan (Little & Rubin, 2019; Moritz & Bartz-Beielstein, 2017).

3.2.4 Pemilihan Metode Yang Tepat

Pemilihan metode penanganan *missing value* harus memperhatikan karakteristik data. Jika data hilang sangat sedikit, interpolasi sederhana mungkin sudah cukup. Jika data hilang terjadi pada interval pendek dan pola data relatif stabil, *forward fill* dapat digunakan. Jika data memiliki pola musiman kuat, imputasi berdasarkan periode yang sama dapat lebih sesuai. Misalnya, data konsumsi energi yang hilang pada pukul 13.00 dapat diperkirakan dari rata-rata konsumsi pukul 13.00 pada hari-hari sebelumnya.

Hal yang perlu dihindari adalah memilih metode hanya karena mudah diterapkan. Metode yang sederhana belum tentu sesuai untuk semua kasus. Mengisi semua nilai hilang dengan rata-rata keseluruhan, misalnya, dapat

membuat data kehilangan variasi penting. Begitu pula penggunaan data masa depan dalam proses imputasi dapat menyebabkan evaluasi model menjadi tidak objektif. Pada pemodelan prediktif, data latih sebaiknya hanya menggunakan informasi yang memang tersedia pada masa itu.

Ringkasan metode penanganan *missing value* dapat dilihat pada tabel berikut.

Tabel 3.1: Metode Penanganan Missing Value Pada Data Time Series

Metode	Cara Kerja	Kelebihan	Keterbatasan
Menghapus data	Baris atau periode yang hilang dikeluarkan	Sederhana	Dapat merusak urutan waktu
Forward fill	Mengisi dengan nilai sebelumnya	Mudah dan cocok untuk data stabil	Kurang cocok untuk data fluktuatif
Backward fill	Mengisi dengan nilai setelahnya	Sederhana	Berisiko menyebabkan *data leakage*
Rata-rata/median	Mengisi dengan nilai pusat data	Mudah diterapkan	Kurang memperhatikan pola waktu
Interpolasi	Mengestimasi nilai di antara dua titik	Cocok untuk celah pendek	Kurang tepat untuk lonjakan mendadak
Imputasi berbasis model	Menggunakan model untuk menaksir nilai hilang	Lebih fleksibel	Lebih kompleks dan perlu validasi

3.2.5 Dampak Missing Value Pada Model Deep Learning

Model *deep learning* sangat bergantung pada kualitas input. Pada LSTM dan GRU, data biasanya disusun dalam bentuk *sequence* atau *window*. Jika terdapat nilai hilang di dalam *window*, model dapat menerima informasi yang tidak lengkap. Hal ini dapat mengganggu proses pembelajaran pola temporal.

Selain itu, nilai hilang yang ditangani secara tidak tepat dapat menciptakan pola palsu. Misalnya, jika banyak nilai hilang diisi dengan angka nol, model dapat menganggap nol sebagai pola konsumsi sebenarnya. Padahal, angka nol tersebut hanya hasil pengisian data. Kesalahan seperti ini dapat menurunkan akurasi prediksi dan menyebabkan interpretasi yang keliru.

Penanganan *missing value* sebaiknya dilakukan sebelum normalisasi dan pembentukan *window time series*. Urutannya biasanya dimulai dari pemeriksaan data, penanganan nilai hilang, validasi hasil imputasi, normalisasi, lalu pembentukan *sequence.* Dengan alur ini, data yang masuk ke model sudah lebih bersih dan konsisten.

3.3 Normalisasi Dan Standardisasi Data

Setelah data *time series* dibersihkan dan nilai hilang ditangani, tahap berikutnya adalah menyesuaikan skala data melalui **normalisasi** atau **standardisasi**. Tahap ini penting karena banyak model *machine learning* dan *deep learning* sensitif terhadap skala nilai input. Jika satu variabel memiliki rentang nilai yang sangat besar, sedangkan variabel lain memiliki rentang kecil, model dapat lebih "memperhatikan" variabel berskala besar meskipun belum tentu variabel tersebut paling penting.

Pada data konsumsi energi, misalnya, nilai konsumsi listrik dapat berada pada rentang ratusan hingga ribuan kWh. Jika data tersebut digabungkan dengan variabel lain seperti suhu ruangan yang hanya berada pada rentang 20–35°C, maka perbedaan skala dapat memengaruhi proses pelatihan model. Model seperti LSTM dan GRU menggunakan proses optimisasi berbasis gradien. Skala data yang terlalu besar atau tidak seimbang dapat membuat proses pelatihan menjadi lebih lambat, tidak stabil, atau sulit mencapai hasil optimal (Goodfellow et al., 2016).

3.3.1 Pengertian Normalisasi

Normalisasi adalah proses mengubah nilai data ke dalam rentang tertentu, biasanya antara 0 sampai 1 atau -1 sampai 1. Tujuannya adalah membuat semua nilai berada dalam skala yang sebanding. Salah satu teknik normalisasi yang paling umum digunakan adalah **Min-Max Scaling**.

Rumus Min-Max Scaling adalah sebagai berikut:

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}}$$

dengan:

- x' adalah nilai hasil normalisasi.

- x adalah nilai asli.
- x_{min}alah nilai minimum pada data.
- x_{max}adalah nilai maksimum pada data.

Sebagai contoh, jika konsumsi energi minimum adalah 100 kWh dan maksimum adalah 900 kWh, maka nilai 500 kWh dapat dinormalisasi menjadi:

$$x' = \frac{500 - 100}{900 - 100} = \frac{400}{800} = 0{,}5$$

Artinya, nilai 500 kWh berada di tengah rentang data. Normalisasi seperti ini banyak digunakan pada model *deep learning* karena membantu proses pelatihan menjadi lebih stabil. Pada model LSTM, data yang sudah dinormalisasi dapat membantu model mempelajari pola temporal tanpa terganggu oleh skala nilai yang terlalu besar.

3.3.2 Pengertian Standardisasi

Standardisasi adalah proses mengubah data agar memiliki rata-rata 0 dan standar deviasi 1. Teknik ini sering disebut **Z-Score Standardization**. Standardisasi tidak membatasi nilai dalam rentang 0 sampai 1, tetapi mengubah data berdasarkan posisi nilai terhadap rata-rata dan sebaran data.

Rumus standardisasi adalah sebagai berikut:

$$z = \frac{x - \mu}{\sigma}$$

Keterangan:

- z adalah nilai hasil standardisasi.
- x adalah nilai asli.
- μ adalah rata-rata data.
- σ adalah standar deviasi data.

Jika suatu nilai memiliki $z = 0$, berarti nilai tersebut sama dengan rata-rata. Jika $z > 0$, berarti nilai berada di atas rata-rata. Jika $z < 0$, berarti nilai berada di bawah rata-rata. Standardisasi sering digunakan ketika data memiliki distribusi yang cukup menyebar atau ketika model membutuhkan data dengan pusat nilai yang seimbang.

Dalam pemodelan *time series*, standardisasi dapat membantu ketika data memiliki variasi yang besar dari waktu ke waktu. Misalnya, data suhu, tekanan, konsumsi energi, dan kelembapan memiliki satuan serta rentang nilai yang berbeda. Dengan standardisasi, semua variabel dapat ditempatkan dalam skala yang lebih sebanding.

3.3.3 Perbedaan Normalisasi Dan Standardisasi

Normalisasi dan standardisasi sama-sama bertujuan menyesuaikan skala data, tetapi cara kerjanya berbeda. Normalisasi mengubah data ke rentang tertentu, sedangkan standardisasi mengubah data berdasarkan rata-rata dan standar deviasi. Pemilihan metode bergantung pada karakteristik data dan model yang digunakan.

Tabel 3.3 Perbandingan Normalisasi Dan Standardisasi

Aspek	Normalisasi	Standardisasi
Tujuan	Mengubah data ke rentang tertentu	Mengubah data berdasarkan rata-rata dan standar deviasi
Rumus umum	Min-Max Scaling	Z-Score
Rentang hasil	Biasanya 0 sampai 1	Tidak memiliki batas tetap
Cocok digunakan ketika	Data memiliki batas minimum dan maksimum yang jelas	Data memiliki variasi besar dan perlu dipusatkan
Kelemahan	Sensitif terhadap nilai ekstrem	Hasil tidak selalu mudah dibaca secara langsung
Contoh penggunaan	LSTM untuk prediksi konsumsi energi	Data multivariat dengan skala berbeda

Tabel tersebut menunjukkan bahwatidakadametodeyang selalu paling baik untuk semua kasus. Normalisasi sering dipilih pada pemodelan LSTM

karena menghasilkan nilai dalam rentang kecil dan stabil. Namun, jika data memiliki banyak nilai ekstrem, standardisasi dapat menjadi pilihan yang lebih sesuai karena tidak terlalu bergantung pada nilai minimum dan maksimum.

3.3.4 Pentingnya Scaling Pada Model Deep Learning

Model *deep learning* belajar melalui proses pembaruan bobot secara bertahap. Jika nilai input terlalu besar, proses pembaruan bobot dapat menjadi tidak stabil. Jika nilai input memiliki skala yang sangat berbeda antarvariabel, model dapat mengalami kesulitan dalam menemukan pola yang seimbang. Oleh karena itu, normalisasi atau standardisasi membantu model belajar dengan lebih efisien.

Pada LSTM dan GRU, data biasanya disusun dalam bentuk *sequence*. Setiap *sequence* berisi beberapa nilai historis yang digunakan untuk memprediksi nilai berikutnya. Jika data belum dinormalisasi, model dapat membutuhkan waktu pelatihan lebih lama. Bahkan, dalam beberapa kasus, model dapat menghasilkan prediksi yang buruk karena proses optimisasi tidak berjalan dengan baik.

Scaling juga penting ketika data memiliki lebih dari satu variabel. Misalnya, model prediksi konsumsi energi menggunakan input berupa konsumsi listrik, suhu, kelembapan, dan jumlah penghuni. Karena setiap variabel memiliki satuan dan rentang nilai berbeda, scaling diperlukan agar model dapat membaca semua variabel secara lebih proporsional.

3.3.5 Risiko Data Leakage Dalam Normalisasi

Salah satu kesalahan yang sering terjadi dalam normalisasi data *time series* adalah menggunakan seluruh dataset untuk menghitung nilai minimum, maksimum, rata-rata, atau standar deviasi sebelum data dibagi menjadi data latih dan data uji. Cara ini berisiko menyebabkan **data leakage**, yaitu kondisi ketika informasi dari data uji masuk ke proses pelatihan model.

Pada data *time series*, data uji biasanya merepresentasikan masa depan. Jika nilai minimum dan maksimum dihitung dari seluruh data, maka model secara tidak langsung telah menggunakan informasi dari masa depan. Hal ini dapat membuat hasil evaluasi terlihat lebih baik daripada kondisi sebenarnya.

Praktik yang lebih tepat adalah menghitung parameter scaling hanya dari data latih. Setelah itu, parameter yang sama digunakan untuk mentransformasi data validasi dan data uji. Dengan cara ini, proses evaluasi lebih adil karena model hanya menggunakan informasi yang tersedia pada periode pelatihan (Hyndman & Athanasopoulos, 2021).

Alur yang disarankan adalah sebagai berikut:

Membagi data berdasarkan urutan waktu menjadi data latih, validasi, dan uji.

Menghitung parameter normalisasi atau standardisasi dari data latih.

Menerapkan parameter tersebut pada data latih, validasi, dan uji.

Melatih model menggunakan data yang sudah diskalakan.

Mengembalikan hasil prediksi ke skala asli jika diperlukan.

3.3.6 Inverse Transform Pada Hasil Prediksi

Setelah model menghasilkan prediksi, hasil tersebut biasanya masih berada dalam skala normalisasi atau standardisasi. Agar hasil prediksi mudah dipahami, nilai tersebut perlu dikembalikan ke skala asli. Proses ini disebut **inverse transform**.

Misalnya, model LSTM menghasilkan prediksi konsumsi energi sebesar 0,65 dalam skala normalisasi. Nilai ini belum langsung bermakna bagi pengguna karena bukan satuan kWh. Dengan *inverse transform*, nilai 0,65 dapat dikembalikan menjadi nilai konsumsi energi sebenarnya, misalnya 620 kWh. Nilai asli inilah yang kemudian dapat digunakan untuk evaluasi, pelaporan, atau proses interpretasi menggunakan *fuzzy logic*.

Inverse transform penting karena metrik evaluasi seperti MAE, RMSE, dan MAPE lebih mudah dipahami jika dihitung pada skala asli data. Selain itu, sistem fuzzy juga biasanya membutuhkan nilai yang sesuai dengan konteks nyata. Misalnya, kategori “rendah”, “sedang”, dan “tinggi” lebih mudah dirancang berdasarkan satuan kWh daripada skala 0 sampai 1.

3.3.7 Pemilihan Metode Scaling Yang Tepat

Pemilihan normalisasi atau standardisasi harus disesuaikan dengan karakteristik data. Jika data memiliki batas minimum dan maksimum yang cukup jelas, normalisasi dapat digunakan. Jika data memiliki distribusi yang menyebar dan terdapat variasi besar, standardisasi dapat dipertimbangkan.

Jika data mengandung outlier yang kuat, perlu dilakukan pemeriksaan lebih lanjut karena normalisasi Min-Max sangat dipengaruhi oleh nilai ekstrem.

Dalam praktik pemodelan *time series*, normalisasi Min-Max sering digunakan untuk LSTM karena sederhana dan membantu menstabilkan input. Namun, keputusan akhir tetap perlu didukung oleh eksperimen. Model dapat diuji dengan beberapa skenario scaling, kemudian dibandingkan berdasarkan metrik evaluasi seperti MAE, RMSE, dan MAPE. Dengan cara ini, pemilihan metode tidak hanya berdasarkan teori, tetapi juga berdasarkan hasil empiris.

3.4 Pembentukan Window Atau Sequence Data

Pembentukan *window* atau *sequence data* merupakan tahap penting dalam pemodelan data *time series*, terutama ketika menggunakan model *deep learning* seperti RNN, LSTM, dan GRU. Data *time series* pada awalnya biasanya berbentuk satu deret nilai berdasarkan waktu, misalnya konsumsi energi per jam. Agar dapat dipelajari oleh model, deret tersebut perlu diubah menjadi pasangan input dan target. Input berisi sejumlah nilai historis, sedangkan target berisi nilai yang ingin diprediksi.

Konsep *window* dapat dipahami sebagai jendela waktu yang digunakan untuk melihat pola masa lalu. Misalnya, sistem menggunakan data konsumsi energi selama 24 jam terakhir untuk memprediksi konsumsi energi 1 jam berikutnya. Dalam contoh tersebut, panjang *window* adalah 24, sedangkan target prediksinya adalah satu periode setelahnya. Teknik ini membuat model dapat belajar bahwa nilai masa depan tidak muncul secara terpisah, tetapi berkaitan dengan pola pada periode sebelumnya.

Secara sederhana, jika data *time series* ditulis sebagai:

y1,y2,y3,…,yty_1, y_2, y_3, \ldots, y_ty1,y2,y3,…,yt

maka pembentukan *window* dapat dibuat menjadi:

Xt=[yt−n,yt−n+1,…,yt−1]X_t = [y_{t-n}, y_{t-n+1}, \ldots, y_{t-1}]Xt =[yt−n,yt−n+1,…,yt−1] y^t=f(Xt)\hat{y}_t = f(X_t)y^t=f(Xt)

Keterangan:

XtX_tXt adalah input model berupa kumpulan nilai historis.

nnn adalah panjang *window* atau jumlah periode masa lalu yang digunakan.

yt−ny_{t-n}yt−n sampai yt−1y_{t-1}yt−1 adalah data sebelum waktu ke-ttt.

y^t\hat{y}_ty^t adalah hasil prediksi pada waktu ke-ttt.

f(Xt)f(X_t)f(Xt) adalah fungsi model yang mempelajari pola dari data historis.

Rumus tersebut menunjukkan bahwa model tidak hanya melihat satu nilai terakhir, tetapi melihat sekumpulan nilai sebelumnya. Pada model LSTM dan GRU, bentuk input seperti ini sangat penting karena model dirancang untuk membaca urutan data. Semakin baik susunan *sequence* yang diberikan, semakin besar peluang model untuk memahami pola temporal dengan benar.

3.4.1 Sliding Window Pada Data Time Series

Metode yang paling umum digunakan dalam pembentukan *sequence* adalah **sliding window**. Pada metode ini, jendela data bergerak secara bertahap dari awal sampai akhir deret waktu. Setiap pergeseran menghasilkan satu pasangan input dan target baru. Misalnya, jika panjang *window* adalah 3, maka tiga nilai pertama digunakan untuk memprediksi nilai keempat. Kemudian jendela bergeser satu langkah, sehingga nilai kedua, ketiga, dan keempat digunakan untuk memprediksi nilai kelima.

Contoh sederhana:

[10,12,13]→15[10, 12, 13] \rightarrow 15[10,12,13]→15
[12,13,15]→16[12, 13, 15] \rightarrow 16[12,13,15]→16
[13,15,16]→18[13, 15, 16] \rightarrow 18[13,15,16]→18

Pada contoh tersebut, bagian kiri adalah input, sedangkan bagian kanan adalah target. Dengan cara ini, satu deret data panjang dapat diubah menjadi banyak sampel pelatihan. Teknik *sliding window* banyak digunakan dalam prediksi *time series* karena dapat mempertahankan urutan waktu sekaligus memperbanyak pasangan data latih (Hyndman & Athanasopoulos, 2021).

3.4.2 Menentukan Panjang Window

Pemilihan panjang *window* perlu disesuaikan dengan karakteristik data dan tujuan prediksi. Jika *window* terlalu pendek, model mungkin tidak memiliki

informasi historis yang cukup untuk mengenali pola. Misalnya, menggunakan data 2 jam terakhir mungkin belum cukup untuk memprediksi konsumsi energi jika pola hariannya sangat kuat. Sebaliknya, jika *window* terlalu panjang, model dapat menerima terlalu banyak informasi yang tidak semuanya relevan. Hal ini dapat memperberat proses pelatihan dan meningkatkan risiko *overfitting*.

Pada data konsumsi energi per jam, panjang *window* 24 sering digunakan untuk menangkap pola harian. Jika data memiliki pola mingguan, panjang *window* dapat diperluas menjadi 168 jam atau 7 × 24 jam. Namun, semakin panjang *window*, semakin besar kebutuhan memori dan waktu pelatihan. Oleh karena itu, panjang *window* sebaiknya ditentukan melalui kombinasi pemahaman domain dan eksperimen model.

Dalam praktik penelitian, beberapa skenario *window* dapat diuji, misalnya 6 jam, 12 jam, 24 jam, atau 48 jam. Hasilnya kemudian dibandingkan menggunakan metrik evaluasi seperti MAE, RMSE, dan MAPE. Pendekatan ini membantu menentukan panjang *window* yang paling sesuai berdasarkan performa model, bukan hanya asumsi awal.

3.4.3 Sequence Untuk Prediksi Satu Langkah Dan Banyak Langkah

Berdasarkan target prediksinya, pembentukan *sequence* dapat digunakan untuk dua jenis prediksi, yaitu **single-step forecasting** dan **multi-step forecasting**. Pada *single-step forecasting*, model memprediksi satu nilai berikutnya. Misalnya, data 24 jam terakhir digunakan untuk memprediksi konsumsi energi 1 jam ke depan. Pendekatan ini lebih sederhana dan sering digunakan sebagai tahap awal dalam penelitian prediksi *time series*.

Pada *multi-step forecasting*, model memprediksi beberapa nilai ke depan sekaligus. Misalnya, data 24 jam terakhir digunakan untuk memprediksi konsumsi energi 6 jam berikutnya. Pendekatan ini lebih menantang karena kesalahan prediksi dapat bertambah pada setiap langkah. Namun, *multi-step forecasting* berguna ketika sistem membutuhkan perencanaan beberapa periode ke depan, misalnya untuk pengaturan beban energi harian.

Perbedaan keduanya dapat dilihat dari bentuk target. Pada *single-step*, target berupa satu nilai:

$$X_t = [y_{t-24}, \dots, y_{t-1}] \rightarrow y_t$$

Pada *multi-step*, target berupa beberapa nilai:

$$X_t = [y_{t-24}, \dots, y_{t-1}] \rightarrow [y_t, y_{t+1}, \dots, y_{t+k}]$$

Keterangan kkk menunjukkan jumlah langkah prediksi ke depan. Pemilihan *single-step* atau *multi-step* perlu disesuaikan dengan kebutuhan sistem. Untuk studi awal, *single-step forecasting* biasanya lebih mudah diterapkan dan dievaluasi.

3.4.4 Bentuk Input Untuk LSTM Dan GRU

Model LSTM dan GRU memiliki format input khusus. Pada umumnya, input untuk model sekuensial berbentuk tiga dimensi, yaitu:

$$(\text{jumlah sampel}, \text{time steps}, \text{jumlah fitur})$$

Jumlah sampel menunjukkan banyaknya pasangan input yang terbentuk dari proses *sliding window*. *Time steps* menunjukkan panjang *window*. Jumlah fitur menunjukkan banyaknya variabel yang digunakan dalam setiap waktu pengamatan.

Jika data hanya menggunakan satu variabel, misalnya konsumsi energi, maka jumlah fiturnya adalah 1. Jika data menggunakan beberapa variabel, seperti konsumsi energi, suhu, kelembapan, dan jumlah penghuni, maka jumlah fiturnya lebih dari 1. Data dengan satu variabel disebut *univariate time series*, sedangkan data dengan banyak variabel disebut *multivariate time series*.

Sebagai contoh, jika tersedia 1.000 sampel, panjang *window* 24 jam, dan jumlah fitur 1, maka bentuk input model adalah:

$$(1000, 24, 1)$$

Format ini penting dipahami karena kesalahan dalam membentuk dimensi input dapat menyebabkan model gagal dijalankan atau belajar dari pola yang salah. Chollet (2021) menjelaskan bahwa representasi tensor menjadi aspek penting dalam penerapan *deep learning*, termasuk pada data sekuensial.

3.4.5 Risiko Kesalahan Dalam Pembentukan Sequence

Kesalahan dalam pembentukan *window* dapat menyebabkan hasil model menjadi tidak valid. Salah satu kesalahan yang sering terjadi adalah memasukkan data target ke dalam input. Misalnya, model diminta memprediksi konsumsi pukul 10.00, tetapi data pukul 10.00 sudah ikut masuk dalam input. Kondisi ini termasuk *data leakage* karena model memperoleh informasi yang seharusnya belum tersedia saat prediksi dilakukan.

Kesalahan lain adalah mengacak data sebelum membentuk *sequence*. Pada data *time series*, urutan waktu harus dijaga. Jika data diacak sebelum *window* dibentuk, maka pola temporal menjadi rusak. Data latih dan data uji juga sebaiknya dibagi berdasarkan urutan waktu, bukan secara acak seperti pada banyak kasus data tabular.

Selain itu, panjang *window* yang tidak sesuai juga dapat memengaruhi kualitas model. *Window* yang terlalu pendek dapat membuat model kehilangan pola penting, sedangkan *window* yang terlalu panjang dapat membuat model terlalu kompleks. Karena itu, pembentukan *sequence* harus dilakukan secara hati-hati dan terdokumentasi.

3.4.6 Peran Window Dalam Model LSTM–Fuzzy Logic

Dalam model hibrida LSTM–Fuzzy Logic, pembentukan *window* berperan pada tahap prediksi numerik. Data historis dibentuk menjadi *sequence*, kemudian diberikan kepada LSTM untuk menghasilkan nilai prediksi. Setelah nilai prediksi diperoleh, hasil tersebut dapat diteruskan ke sistem *fuzzy logic* untuk diklasifikasikan ke dalam kategori linguistik, seperti rendah, sedang, atau tinggi.

Dengan alur tersebut, *window* menjadi jembatan antara data mentah dan model prediksi. Jika *window* dibentuk dengan benar, LSTM dapat membaca pola temporal dengan lebih baik. Jika hasil prediksi sudah cukup stabil, *fuzzy logic* dapat memberikan interpretasi keputusan yang lebih mudah dipahami. Oleh karena itu, kualitas pembentukan *sequence* berpengaruh langsung terhadap kualitas prediksi dan interpretasi sistem.

3.5 Pembagian Data Latih, Validasi, Dan Uji

Pembagian data merupakan tahap penting sebelum model *time series* dilatih. Tujuannya adalah memisahkan data menjadi beberapa bagian agar proses pembelajaran, pemilihan model, dan pengujian performa dapat dilakukan secara adil. Secara umum, dataset dibagi menjadi **data latih**, **data validasi**, dan **data uji**. Data latih digunakan untuk melatih model, data validasi digunakan untuk memilih konfigurasi terbaik, sedangkan data uji digunakan untuk menilai kemampuan model pada data yang belum pernah dilihat sebelumnya.

Pada data *time series*, pembagian data tidak boleh dilakukan secara acak seperti pada banyak kasus data tabular. Hal ini karena data *time series* memiliki urutan waktu yang harus dipertahankan. Data masa lalu digunakan untuk memprediksi masa depan. Jika data diacak, maka informasi dari masa depan dapat masuk ke proses pelatihan. Kesalahan ini disebut *data leakage*, yaitu kondisi ketika model secara tidak langsung memperoleh informasi yang seharusnya belum tersedia pada saat prediksi dilakukan (Hyndman & Athanasopoulos, 2021).

Contoh pembagian yang tepat adalah menggunakan data Januari sampai Agustus sebagai data latih, data September sampai Oktober sebagai data validasi, dan data November sampai Desember sebagai data uji. Dengan cara ini, model belajar dari data masa lalu, kemudian diuji pada periode setelahnya. Pola ini lebih sesuai dengan kondisi nyata karena dalam praktik prediksi, model memang digunakan untuk memperkirakan nilai masa depan berdasarkan data historis.

Data latih berfungsi sebagai sumber utama pembelajaran model. Pada tahap ini, model seperti LSTM atau GRU mempelajari pola temporal, tren, musiman, dan hubungan antarwaktu. Semakin representatif data latih, semakin besar peluang model memahami pola yang benar. Namun, jumlah data latih yang besar tidak selalu menjamin hasil baik jika data tersebut kotor, tidak konsisten, atau tidak mewakili kondisi yang akan diprediksi.

Data validasi digunakan untuk membantu memilih model terbaik. Misalnya, peneliti dapat mencoba beberapa variasi panjang *window*, jumlah neuron, jumlah lapisan LSTM, *learning rate*, atau ukuran *batch*. Setiap konfigurasi diuji menggunakan data validasi. Model dengan hasil terbaik pada data

validasi kemudian dipilih untuk diuji pada data uji. Dengan adanya data validasi, data uji tetap dapat dijaga sebagai data yang benar-benar independen.

Data uji digunakan pada tahap akhir untuk mengukur performa model. Data ini tidak boleh digunakan saat pelatihan maupun pemilihan konfigurasi model. Jika data uji terlalu sering digunakan untuk menyesuaikan model, maka hasil evaluasi dapat menjadi terlalu optimistis. Artinya, model tampak baik pada laporan eksperimen, tetapi belum tentu baik ketika digunakan pada data baru.

Pembagian data dapat dilakukan dengan beberapa rasio, misalnya 70:15:15 atau 80:10:10. Angka tersebut bukan aturan mutlak. Pemilihan rasio perlu disesuaikan dengan jumlah data dan tujuan analisis. Jika dataset sangat besar, porsi data uji yang lebih kecil mungkin sudah cukup. Jika dataset kecil, pembagian harus dilakukan lebih hati-hati agar setiap bagian tetap memiliki data yang memadai.

Tabel 3.2: Contoh Pembagian Data Time Series

Bagian Data	Fungsi Utama	Contoh Periode	Catatan Penting
Data latih	Melatih model	Januari–Agustus	Digunakan untuk mempelajari pola historis
Data validasi	Memilih konfigurasi model	September–Oktober	Digunakan untuk tuning parameter
Data uji	Menguji performa akhir	November–Desember	Tidak digunakan saat pelatihan

Selain pembagian sederhana, terdapat juga pendekatan **walk-forward validation**. Pada pendekatan ini, model diuji secara bertahap mengikuti urutan waktu. Misalnya, model dilatih pada bulan Januari sampai Maret, lalu diuji pada April. Setelah itu, data April ikut dimasukkan ke data latih, kemudian model diuji pada Mei, dan seterusnya. Pendekatan ini sering digunakan ketika peneliti ingin menilai stabilitas model pada beberapa periode berbeda (Brownlee, 2018).

Pembagian data juga perlu dilakukan sebelum proses normalisasi atau standardisasi. Parameter normalisasi, seperti nilai minimum dan maksimum, sebaiknya dihitung hanya dari data latih. Setelah itu, parameter tersebut

digunakan untuk mentransformasi data validasi dan data uji. Cara ini mencegah informasi dari data masa depan masuk ke proses pelatihan.

Pada model LSTM–Fuzzy Logic, pembagian data berpengaruh terhadap dua hal. Pertama, kualitas model LSTM dalam mempelajari pola prediksi numerik. Kedua, kualitas interpretasi fuzzy yang bergantung pada hasil prediksi model. Jika pembagian data tidak benar, maka performa model dapat terlihat tinggi secara semu, tetapi lemah ketika diterapkan pada kondisi nyata. Oleh karena itu, pembagian data harus mempertahankan urutan waktu, menghindari *data leakage*, dan mendukung evaluasi yang objektif.

Bab 4
Dasar-Dasar Deep Learning untuk Data Sekuensial

4.1 Konsep Neural Network

Neural network atau jaringan saraf tiruan adalah model komputasi yang terinspirasi dari cara kerja neuron biologis dalam memproses informasi. Dalam bidang kecerdasan buatan, *neural network* digunakan untuk mempelajari pola dari data dan menghasilkan keluaran tertentu, seperti klasifikasi, prediksi, pengenalan gambar, pengenalan suara, atau prediksi data *time series*. Model ini menjadi dasar penting dalam *deep learning* karena arsitektur *deep learning* pada dasarnya tersusun dari banyak lapisan jaringan saraf tiruan (Goodfellow et al., 2016).

Secara umum, *neural network* terdiri atas tiga bagian utama, yaitu **input layer**, **hidden layer**, dan **output layer**. *Input layer* menerima data masukan, misalnya nilai konsumsi energi, suhu, kelembapan, atau data historis lainnya. *Hidden layer* memproses data melalui sejumlah neuron buatan. Pada bagian inilah model mempelajari pola dan hubungan antarvariabel. *Output layer* menghasilkan keluaran akhir, misalnya nilai prediksi konsumsi energi atau kategori tertentu.

Neuron buatan bekerja dengan menerima beberapa nilai input, mengalikannya dengan bobot, menambahkan bias, lalu meneruskannya ke fungsi aktivasi. Secara sederhana, proses tersebut dapat ditulis sebagai berikut:

$$z = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Keterangan:

- $x_1, x_2, \ldots, x_n$ adalah nilai input.
- $w_1, w_2, \ldots, w_n$ adalah bobot.
- b adalah bias.
- z adalah hasil kombinasi linear.

- f(z) adalah fungsi aktivasi.
- a adalah output neuron.

Bobot menentukan seberapa besar pengaruh suatu input terhadap output. Bias membantu model menyesuaikan hasil agar lebih fleksibel. Fungsi aktivasi berperan memberikan kemampuan nonlinier pada model. Tanpa fungsi aktivasi, jaringan saraf hanya akan bekerja seperti model linear biasa, sehingga kurang mampu menangkap pola kompleks pada data (Haykin, 2009).

Dalam proses pelatihan, *neural network* belajar dengan cara memperbarui bobot dan bias. Pada awalnya, bobot biasanya bernilai acak. Model kemudian menghasilkan prediksi, membandingkannya dengan nilai sebenarnya, lalu menghitung kesalahan menggunakan *loss function*. Kesalahan tersebut digunakan untuk memperbaiki bobot melalui proses *backpropagation* dan optimisasi. Proses ini dilakukan berulang-ulang sampai model menghasilkan prediksi yang semakin baik (Rumelhart et al., 1986).

Pada prediksi data *time series*, *neural network* dapat digunakan untuk mempelajari hubungan antara data masa lalu dan nilai masa depan. Misalnya, data konsumsi energi selama beberapa jam sebelumnya dapat digunakan sebagai input untuk memprediksi konsumsi energi pada jam berikutnya. Namun, jaringan saraf biasa atau *feedforward neural network* belum secara khusus dirancang untuk mengingat urutan waktu. Karena itu, untuk data sekuensial, dikembangkan arsitektur lanjutan seperti Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), dan Gated Recurrent Unit (GRU).

Kelebihan utama *neural network* adalah kemampuannya mempelajari pola yang kompleks dan nonlinier. Model ini tidak selalu membutuhkan rumus manual untuk menjelaskan hubungan antarvariabel karena pola tersebut dapat dipelajari langsung dari data. Namun, *neural network* juga memiliki keterbatasan. Model ini membutuhkan data yang cukup, proses pelatihan yang tepat, serta pemilihan parameter yang hati-hati. Selain itu, model jaringan saraf sering sulit dijelaskan secara langsung, terutama jika memiliki banyak lapisan dan neuron.

Dalam buku ini, pemahaman tentang *neural network* menjadi dasar sebelum membahas LSTM, GRU, dan integrasinya dengan *fuzzy logic*. LSTM dan GRU merupakan pengembangan dari jaringan saraf yang dirancang khusus untuk data berurutan. Sementara itu, *fuzzy logic* dapat membantu memberikan interpretasi terhadap hasil prediksi agar keluaran model lebih mudah dipahami dan digunakan dalam pengambilan keputusan.

4.2 Fungsi Aktivasi Dan Loss Function

Dalam *neural network*, dua komponen penting yang menentukan proses pembelajaran model adalah **fungsi aktivasi** dan **loss function**. Fungsi aktivasi digunakan untuk membantu neuron mempelajari pola nonlinier, sedangkan *loss function* digunakan untuk mengukur seberapa besar kesalahan model dalam menghasilkan prediksi. Keduanya saling berkaitan karena model tidak hanya perlu menghasilkan output, tetapi juga perlu mengetahui apakah output tersebut sudah mendekati nilai yang benar.

Fungsi aktivasi bekerja setelah neuron menerima input, bobot, dan bias. Pada dasarnya, neuron menghitung kombinasi linear dari input, kemudian hasilnya diproses oleh fungsi aktivasi. Tanpa fungsi aktivasi, jaringan saraf tiruan hanya akan bekerja seperti model linear biasa. Akibatnya, model sulit menangkap pola kompleks yang banyak ditemukan pada data nyata, termasuk data *time series* seperti konsumsi energi, harga saham, suhu, atau trafik jaringan. Fungsi aktivasi memungkinkan jaringan saraf mempelajari hubungan yang tidak selalu berbentuk garis lurus (Goodfellow et al., 2016).

Secara umum, proses pada neuron dapat dituliskan sebagai berikut:

$$z = wx + b$$

$$a = f(z)$$

Keterangan:

- z adalah hasil kombinasi linear.
- w adalah bobot.
- x adalah input.
- b adalah bias.
- f(z)f(z)f(z) adalah fungsi aktivasi.
- a adalah output setelah aktivasi.

Salah satu fungsi aktivasi yang dikenal luas adalah **sigmoid**. Fungsi sigmoid menghasilkan nilai antara 0 dan 1. Fungsi ini cocok digunakan ketika keluaran model perlu ditafsirkan sebagai probabilitas. Rumus sigmoid adalah:

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Kelebihan sigmoid adalah mudah dipahami dan menghasilkan output dalam rentang terbatas. Namun, sigmoid memiliki kelemahan, terutama pada jaringan yang dalam, karena dapat menyebabkan masalah *vanishing gradient*. Masalah ini terjadi ketika nilai gradien menjadi sangat kecil sehingga proses pembelajaran model menjadi lambat.

Fungsi aktivasi lain adalah **tanh** atau *hyperbolic tangent*. Fungsi ini menghasilkan nilai antara -1 dan 1. Dibandingkan sigmoid, *tanh* sering lebih baik karena output-nya berpusat di sekitar nol. Namun, *tanh* juga masih dapat mengalami masalah *vanishing gradient* pada jaringan yang dalam.

Pada banyak model *deep learning* modern, fungsi aktivasi yang sering digunakan adalah **Rectified Linear Unit** atau **ReLU**. ReLU memiliki bentuk sederhana, yaitu menghasilkan nilai 0 jika input negatif dan menghasilkan nilai input itu sendiri jika input positif. Rumus ReLU dapat ditulis sebagai berikut:

$$f(x) = \max(0, x)$$

ReLU banyak digunakan karena sederhana, cepat dihitung, dan membantu mengurangi masalah *vanishing gradient* pada banyak kasus. Namun, ReLU juga memiliki kelemahan, yaitu kemungkinan munculnya *dead neuron*. Kondisi ini terjadi ketika neuron terus menghasilkan nilai 0 dan tidak lagi aktif dalam proses pembelajaran. Untuk mengatasi hal tersebut, terdapat variasi ReLU seperti *Leaky ReLU* dan *Parametric ReLU* (Nair & Hinton, 2010).

Selain fungsi aktivasi, komponen penting lain adalah **loss function**. *Loss function* digunakan untuk menghitung selisih antara hasil prediksi model dan nilai sebenarnya. Semakin kecil nilai *loss*, semakin baik model dalam melakukan prediksi. Selama proses pelatihan, model berusaha memperkecil nilai *loss* melalui pembaruan bobot.

Pada kasus prediksi numerik atau regresi, *loss function* yang sering digunakan adalah **Mean Squared Error** atau **MSE**. MSE menghitung rata-rata kuadrat selisih antara nilai aktual dan nilai prediksi. Rumusnya adalah:

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

Keterangan:

- y_i adalah nilai sebenarnya.
- $\hat{y}_i$ adalah nilai prediksi.
- n adalah jumlah data.

MSE memberikan penalti lebih besar pada kesalahan yang besar karena selisihnya dikuadratkan.

Selain MSE, terdapat **Mean Absolute Error** atau **MAE**. MAE menghitung rata-rata nilai absolut dari selisih antara nilai aktual dan nilai prediksi. Rumusnya adalah:

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

MAE lebih mudah dipahami karena satuannya sama dengan data asli. Misalnya, jika data konsumsi energi menggunakan satuan kWh, maka nilai MAE juga dapat ditafsirkan dalam kWh. Pada prediksi data *time series*, MAE dan MSE sering digunakan untuk mengukur seberapa jauh prediksi model dari nilai sebenarnya.

Untuk kasus klasifikasi, *loss function* yang umum digunakan adalah **cross-entropy loss**. Fungsi ini digunakan ketika model menghasilkan probabilitas kelas. Misalnya, sistem ingin mengklasifikasikan hasil prediksi ke dalam kategori rendah, sedang, atau tinggi. *Cross-entropy* mengukur seberapa jauh distribusi probabilitas hasil prediksi dari label yang sebenarnya. Semakin besar perbedaan antara prediksi dan label, semakin besar nilai *loss* yang dihasilkan (Bishop, 2006).

Pemilihan fungsi aktivasi dan *loss function* harus disesuaikan dengan jenis masalah. Jika model digunakan untuk prediksi nilai konsumsi energi, maka output model biasanya berupa angka, sehingga *loss function* seperti MSE

atau MAE dapat digunakan. Jika model digunakan untuk klasifikasi kategori, seperti rendah, sedang, atau tinggi, maka *cross-entropy* lebih sesuai. Dalam model LSTM–Fuzzy Logic, LSTM dapat menggunakan MSE atau MAE untuk mempelajari prediksi numerik, sedangkan hasil prediksi tersebut kemudian dapat diteruskan ke sistem *fuzzy logic* untuk menghasilkan interpretasi linguistik.

Tabel 4.2 Contoh Fungsi Aktivasi Dan Loss Function

Komponen	Contoh	Fungsi Utama	Penggunaan Umum
Fungsi Aktivasi	Sigmoid	Menghasilkan nilai 0 sampai 1	Output probabilitas
Fungsi Aktivasi	Tanh	Menghasilkan nilai -1 sampai 1	Jaringan sekuensial tertentu
Fungsi Aktivasi	ReLU	Mengaktifkan nilai positif	Hidden layer deep learning
Loss Function	MSE	Mengukur kesalahan kuadrat	Prediksi numerik
Loss Function	MAE	Mengukur rata-rata kesalahan absolut	Prediksi time series
Loss Function	Cross-Entropy	Mengukur kesalahan klasifikasi	Klasifikasi kelas

Fungsi aktivasi dan *loss function* memiliki peran yang berbeda tetapi sama-sama menentukan kualitas model. Fungsi aktivasi membantu model mempelajari pola kompleks, sedangkan *loss function* memberikan ukuran kesalahan yang digunakan untuk memperbaiki model. Pemahaman terhadap keduanya penting sebelum membangun model *deep learning* yang lebih kompleks seperti LSTM dan GRU.

Rekomendasi Gambar

Gambar 4.2 Hubungan Fungsi Aktivasi Dan Loss Function Dalam Neural Network

Bentuk gambar yang disarankan adalah diagram alur:

Input → Bobot Dan Bias → Fungsi Aktivasi → Output Prediksi → Loss Function → Pembaruan Bobot

Keterangan gambar: fungsi aktivasi bekerja pada proses menghasilkan output neuron, sedangkan *loss function* bekerja setelah output dibandingkan dengan nilai sebenarnya.

4.3 Optimisasi Model Deep Learning

Optimisasi model *deep learning* adalah proses memperbaiki bobot dan bias dalam jaringan saraf agar model dapat menghasilkan prediksi yang semakin mendekati nilai sebenarnya. Pada awal pelatihan, bobot model biasanya diinisialisasi secara acak. Karena itu, hasil prediksi awal sering masih jauh dari target. Melalui proses optimisasi, model menghitung kesalahan prediksi, lalu memperbarui bobot secara bertahap agar nilai kesalahan semakin kecil.

Dalam *neural network*, proses optimisasi berkaitan erat dengan *loss function*. *Loss function* mengukur seberapa besar kesalahan model, sedangkan algoritma optimisasi menentukan bagaimana bobot model diperbaiki berdasarkan kesalahan tersebut. Jika *loss function* menunjukkan bahwa prediksi masih jauh dari nilai sebenarnya, maka optimizer akan mengubah bobot agar prediksi berikutnya menjadi lebih baik. Proses ini dilakukan berulang-ulang selama pelatihan model (Goodfellow et al., 2016).

Secara umum, tujuan optimisasi dapat ditulis sebagai upaya mencari parameter model terbaik:

$$\theta^* = \arg\min_{\theta} L\,(\theta)$$

Keterangan:

- θ^* adalah parameter model terbaik.
- θ adalah kumpulan bobot dan bias model.
- $L(\theta)$ adalah nilai *loss* yang ingin diminimalkan.
- $argmin$ berarti mencari nilai parameter yang membuat *loss* menjadi paling kecil.

Rumus tersebut menunjukkan bahwa pelatihan model pada dasarnya adalah proses mencari kombinasi bobot dan bias yang menghasilkan kesalahan sekecil mungkin. Pada model prediksi *time series*, seperti LSTM atau GRU, parameter yang baik akan membantu model memahami pola temporal dan menghasilkan prediksi yang lebih akurat.

4.3.1 Gradient Descent

Salah satu konsep dasar dalam optimisasi *deep learning* adalah **gradient descent**. Metode ini memperbarui bobot model berdasarkan arah penurunan nilai *loss*. Jika *loss* dianggap seperti permukaan bukit dan lembah, maka *gradient descent* berusaha membawa model menuju titik lembah, yaitu posisi dengan nilai kesalahan paling rendah.

Secara sederhana, pembaruan parameter dapat ditulis sebagai berikut:

$$(\theta)\theta_{baru} = \theta_{lama} - \alpha\nabla L(\theta)$$

Keterangan:

- θ_{baru} adalah parameter setelah diperbarui.
- θ_{lama} adalah parameter sebelum diperbarui.
- α adalah *learning rate*.
- $\nabla L(\theta)$ adalah gradien dari *loss function* terhadap parameter.

Bagian penting dari rumus tersebut adalah *learning rate*. *Learning rate* menentukan seberapa besar langkah perubahan bobot pada setiap proses pembaruan. Jika *learning rate* terlalu kecil, proses pelatihan menjadi sangat lambat. Jika terlalu besar, model dapat melewati titik optimal dan sulit mencapai hasil yang stabil.

4.3.2 Learning Rate

Learning rate merupakan salah satu parameter paling penting dalam pelatihan model *deep learning*. Nilai ini menentukan kecepatan model dalam memperbarui bobot. Pemilihan *learning rate* yang tepat dapat membantu model belajar dengan stabil. Sebaliknya, *learning rate* yang tidak sesuai dapat menyebabkan pelatihan gagal.

Jika *learning rate* terlalu kecil, nilai *loss* memang dapat turun, tetapi prosesnya membutuhkan waktu lama. Model mungkin memerlukan banyak *epoch* untuk mencapai performa yang baik. Jika *learning rate* terlalu besar, nilai *loss* dapat naik turun tidak stabil atau bahkan semakin membesar. Kondisi ini menunjukkan bahwa model tidak berhasil menemukan arah pembelajaran yang baik.

Dalam praktiknya, *learning rate* sering diuji melalui beberapa percobaan. Nilai seperti 0,1; 0,01; 0,001; atau 0,0001 dapat digunakan sebagai kandidat awal, tergantung pada arsitektur dan dataset. Pada banyak implementasi modern, optimizer seperti Adam dapat menyesuaikan proses pembaruan bobot secara lebih adaptif sehingga pelatihan menjadi lebih stabil (Kingma & Ba, 2015).

4.3.3 Stochastic Gradient Descent Dan Mini-Batch

Pada dataset berukuran besar, menghitung gradien menggunakan seluruh data sekaligus dapat memerlukan waktu dan memori yang besar. Untuk mengatasi hal tersebut, digunakan variasi *gradient descent*, seperti **Stochastic Gradient Descent** dan **mini-batch gradient descent**.

Pada **batch gradient descent**, seluruh data latih digunakan untuk menghitung satu kali pembaruan bobot. Cara ini stabil, tetapi dapat lambat jika dataset besar. Pada **Stochastic Gradient Descent** atau SGD, pembaruan bobot dilakukan menggunakan satu sampel data pada setiap langkah. Cara ini lebih cepat, tetapi arah pembaruan dapat lebih berisik. Pada **mini-batch gradient descent**, data dibagi menjadi beberapa kelompok kecil atau *batch*. Setiap *batch* digunakan untuk memperbarui bobot. Pendekatan ini banyak digunakan karena menjadi kompromi antara efisiensi dan stabilitas.

Dalam pelatihan LSTM untuk prediksi *time series*, ukuran *batch* perlu dipilih dengan hati-hati. *Batch size* yang terlalu kecil dapat membuat pelatihan tidak stabil, sedangkan *batch size* yang terlalu besar dapat membutuhkan memori besar dan membuat model kurang fleksibel dalam belajar dari variasi data.

4.3.4 Optimizer Modern

Selain SGD, terdapat beberapa optimizer modern yang sering digunakan dalam *deep learning*. Salah satunya adalah **Adam**, singkatan dari *Adaptive*

Moment Estimation. Adam menggabungkan gagasan momentum dan penyesuaian *learning rate* adaptif untuk setiap parameter. Karena stabil dan mudah digunakan, Adam sering menjadi pilihan awal dalam banyak eksperimen *deep learning* (Kingma & Ba, 2015).

Optimizer lain yang juga dikenal adalah **RMSProp**. RMSProp dirancang untuk menyesuaikan besar pembaruan bobot berdasarkan rata-rata kuadrat gradien. Optimizer ini sering digunakan pada model sekuensial dan RNN karena dapat membantu mengatasi masalah perubahan gradien yang tidak stabil. Ada pula **Adagrad**, yang memberikan pembaruan lebih besar pada parameter yang jarang diperbarui dan pembaruan lebih kecil pada parameter yang sering diperbarui.

Meskipun optimizer modern membantu proses pelatihan, tidak ada optimizer yang selalu terbaik untuk semua kasus. Pemilihan optimizer perlu disesuaikan dengan dataset, arsitektur model, ukuran data, dan tujuan eksperimen. Pada beberapa kasus, Adam memberikan hasil cepat dan stabil. Pada kasus lain, SGD dengan pengaturan *learning rate* yang baik dapat memberikan generalisasi yang lebih kuat.

4.3.5 Epoch, Batch Size, Dan Iterasi

Dalam proses pelatihan model, terdapat tiga istilah penting, yaitu **epoch**, **batch size**, dan **iterasi**. *Epoch* adalah satu putaran penuh ketika seluruh data latih telah digunakan untuk melatih model. *Batch size* adalah jumlah sampel yang diproses sebelum model memperbarui bobot. Iterasi adalah jumlah pembaruan bobot yang terjadi dalam satu *epoch*.

Sebagai contoh, jika terdapat 1.000 sampel data latih dan *batch size* yang digunakan adalah 100, maka dalam satu *epoch* terdapat 10 iterasi. Jika model dilatih selama 50 *epoch*, maka proses pembaruan bobot dilakukan sebanyak 500 kali. Jumlah *epoch* yang terlalu sedikit dapat menyebabkan model belum belajar dengan baik. Sebaliknya, jumlah *epoch* yang terlalu banyak dapat menyebabkan model terlalu menyesuaikan diri dengan data latih atau mengalami *overfitting*.

Pada prediksi *time series*, pemantauan nilai *loss* pada data latih dan validasi penting dilakukan. Jika *loss* data latih terus menurun, tetapi *loss* data validasi mulai meningkat, maka model kemungkinan mulai mengalami *overfitting*. Salah satu teknik yang dapat digunakan untuk mengatasi hal ini adalah **early**

stopping, yaitu menghentikan pelatihan ketika performa pada data validasi tidak lagi membaik.

4.3.6 Tantangan Optimisasi Pada Model Sekuensial

Model sekuensial seperti RNN, LSTM, dan GRU memiliki tantangan optimisasi yang lebih kompleks dibandingkan jaringan saraf biasa. Salah satu tantangan utamanya adalah masalah **vanishing gradient** dan **exploding gradient**. *Vanishing gradient* terjadi ketika gradien menjadi sangat kecil sehingga bobot sulit diperbarui. Sebaliknya, *exploding gradient* terjadi ketika gradien menjadi terlalu besar sehingga pembaruan bobot menjadi tidak stabil (Pascanu et al., 2013).

LSTM dikembangkan untuk mengurangi masalah *vanishing gradient* pada data sekuensial, tetapi proses pelatihannya tetap membutuhkan pengaturan yang hati-hati. Teknik seperti normalisasi data, pemilihan *learning rate*, penggunaan optimizer yang sesuai, *gradient clipping*, dan regularisasi dapat membantu meningkatkan stabilitas pelatihan.

Dalam model LSTM–Fuzzy Logic, optimisasi terutama dilakukan pada bagian LSTM sebagai model prediksi numerik. Jika optimisasi LSTM berjalan baik, hasil prediksi numerik akan lebih stabil. Setelah itu, hasil prediksi dapat diteruskan ke sistem *fuzzy logic* untuk diinterpretasikan menjadi kategori linguistik seperti rendah, sedang, atau tinggi.

Tabel 4.3 Komponen Penting Dalam Optimisasi Deep Learning

Komponen	Fungsi	Catatan Penting
Loss Function	Mengukur kesalahan prediksi	Semakin kecil nilainya, semakin baik model
Gradient Descent	Memperbarui bobot berdasarkan gradien	Dasar dari banyak optimizer
Learning Rate	Menentukan besar langkah pembaruan bobot	Terlalu besar atau kecil dapat mengganggu pelatihan
Batch Size	Menentukan jumlah sampel per pembaruan	Memengaruhi stabilitas dan penggunaan memori
Epoch	Jumlah putaran pelatihan penuh	Terlalu banyak dapat menyebabkan overfitting
Optimizer	Mengatur strategi pembaruan bobot	Contoh: SGD, RMSProp, Adam

4.4 Overfitting Dan Regularisasi

Dalam pelatihan model *deep learning*, salah satu masalah yang sering muncul adalah **overfitting**. Overfitting terjadi ketika model terlalu menyesuaikan diri dengan data latih, tetapi kurang mampu bekerja dengan baik pada data baru. Model yang mengalami overfitting biasanya memiliki nilai kesalahan rendah pada data latih, tetapi nilai kesalahan tinggi pada data validasi atau data uji. Kondisi ini menunjukkan bahwa model bukan hanya mempelajari pola utama data, tetapi juga ikut menghafal noise atau detail yang terlalu spesifik pada data latih (Goodfellow et al., 2016).

Pada prediksi data *time series*, overfitting dapat terjadi ketika model LSTM atau GRU terlalu kompleks dibandingkan jumlah dan kualitas data yang tersedia. Misalnya, model memiliki banyak lapisan dan neuron, tetapi dataset konsumsi energi yang digunakan relatif sedikit. Akibatnya, model mampu mengikuti pola data latih dengan sangat baik, tetapi gagal memprediksi periode berikutnya karena pola pada data baru sedikit berbeda.

Overfitting dapat dikenali melalui perbandingan antara *training loss* dan *validation loss*. Jika *training loss* terus menurun, tetapi *validation loss* mulai meningkat, maka model kemungkinan mulai mengalami overfitting. Dalam kondisi ideal, keduanya sama-sama menurun dan berada pada selisih yang tidak terlalu jauh. Namun, jika selisihnya semakin besar, model perlu dikendalikan agar tidak terlalu menghafal data latih.

Salah satu cara mengatasi overfitting adalah **regularisasi**. Regularisasi adalah teknik untuk membatasi kompleksitas model agar tidak terlalu bebas menyesuaikan diri dengan data latih. Tujuannya bukan membuat model sesederhana mungkin, tetapi membuat model mampu belajar pola yang lebih umum dan tetap baik ketika digunakan pada data baru (Bishop, 2006).

Teknik regularisasi yang umum digunakan adalah **L2 regularization** atau *weight decay*. Teknik ini menambahkan penalti terhadap bobot model yang terlalu besar. Secara sederhana, bentuk regularisasi L2 dapat ditulis sebagai berikut:

$$L_{reg} = L + \lambda \sum w_i^2$$

Keterangan:

- L_{reg} adalah nilai *loss* setelah ditambahkan regularisasi.
- L adalah nilai *loss* asli.
- λ adalah parameter regularisasi.
- w_i adalah bobot model.

Rumus tersebut menunjukkan bahwa model tidak hanya diminta memperkecil kesalahan prediksi, tetapi juga menjaga agar bobot tidak terlalu besar. Bobot yang terlalu besar dapat membuat model terlalu sensitif terhadap perubahan kecil pada input. Dalam data *time series*, sensitivitas berlebihan ini dapat membuat model mengikuti noise, bukan pola utama.

Teknik lain yang banyak digunakan adalah **dropout**. Dropout bekerja dengan menonaktifkan sebagian neuron secara acak selama proses pelatihan. Tujuannya agar model tidak terlalu bergantung pada neuron tertentu. Dengan cara ini, jaringan belajar membangun representasi yang lebih kuat dan tidak mudah menghafal data latih. Dropout diperkenalkan sebagai teknik regularisasi yang efektif untuk mengurangi overfitting pada jaringan saraf tiruan (Srivastava et al., 2014).

Selain dropout, teknik **early stopping** juga sering digunakan. Early stopping menghentikan proses pelatihan ketika performa pada data validasi tidak lagi membaik. Misalnya, jika *validation loss* tidak menurun selama beberapa *epoch*, pelatihan dihentikan meskipun jumlah *epoch* maksimum belum tercapai. Teknik ini sederhana, tetapi efektif karena mencegah model terus belajar sampai terlalu menyesuaikan diri dengan data latih (Prechelt, 1998).

Cara lain untuk mengurangi overfitting adalah mengurangi kompleksitas model. Jika model terlalu besar, jumlah lapisan atau neuron dapat dikurangi. Pada LSTM, misalnya, jumlah unit LSTM tidak harus selalu banyak. Model yang lebih kecil kadang menghasilkan generalisasi lebih baik, terutama jika dataset terbatas. Selain itu, penggunaan data yang lebih banyak dan lebih representatif juga dapat membantu model belajar pola yang lebih stabil.

Pada pemodelan *time series*, regularisasi harus dilakukan dengan hati-hati. Data tidak boleh diacak sembarangan karena urutan waktu harus tetap dijaga. Pembagian data latih, validasi, dan uji harus mengikuti urutan waktu agar evaluasi tidak bias. Jika proses evaluasi salah, model dapat terlihat baik

padahal sebenarnya hanya memanfaatkan kebocoran informasi dari masa depan.

Dalam model LSTM–Fuzzy Logic, overfitting terutama perlu diperhatikan pada bagian LSTM sebagai model prediksi numerik. Jika LSTM mengalami overfitting, hasil prediksi yang masuk ke sistem *fuzzy logic* juga dapat menjadi kurang stabil. Akibatnya, kategori fuzzy seperti rendah, sedang, atau tinggi dapat menjadi kurang tepat. Oleh karena itu, regularisasi tidak hanya berpengaruh terhadap angka prediksi, tetapi juga terhadap kualitas interpretasi keputusan.

Tabel 4.1: Teknik Mengurangi Overfitting Pada Deep Learning

Teknik	Cara Kerja	Manfaat
Dropout	Menonaktifkan sebagian neuron secara acak saat pelatihan	Mengurangi ketergantungan pada neuron tertentu
L2 Regularization	Memberi penalti pada bobot yang terlalu besar	Membuat model lebih stabil
Early Stopping	Menghentikan pelatihan saat validasi tidak membaik	Mencegah model terlalu lama belajar data latih
Mengurangi Kompleksitas Model	Mengurangi jumlah layer atau neuron	Menurunkan risiko model terlalu rumit
Menambah Data	Menggunakan data yang lebih banyak dan representatif	Membantu model belajar pola yang lebih umum

4.5 Perbedaan ANN, CNN, RNN, LSTM, dan GRU

Dalam *deep learning*, terdapat beberapa arsitektur jaringan saraf tiruan yang digunakan untuk jenis data dan tujuan yang berbeda. Beberapa arsitektur yang sering dibahas adalah **Artificial Neural Network (ANN)**, **Convolutional Neural Network (CNN)**, **Recurrent Neural Network (RNN)**, **Long Short-Term Memory (LSTM)**, dan **Gated Recurrent Unit (GRU)**. Meskipun semuanya termasuk keluarga jaringan saraf tiruan, cara kerja dan penggunaannya tidak sama.

ANN merupakan bentuk dasar jaringan saraf tiruan. Model ini biasanya terdiri atas input layer, hidden layer, dan output layer. ANN cocok digunakan untuk data tabular atau data yang setiap fiturnya tidak memiliki urutan waktu khusus. Misalnya, ANN dapat digunakan untuk memprediksi harga rumah berdasarkan luas tanah, jumlah kamar, lokasi, dan usia bangunan. Namun,

ANN kurang ideal untuk data sekuensial karena tidak memiliki mekanisme khusus untuk menyimpan informasi urutan waktu (Haykin, 2009).

CNN adalah arsitektur yang banyak digunakan untuk data berbentuk grid, terutama gambar. CNN menggunakan operasi konvolusi untuk mengenali pola lokal, seperti tepi, bentuk, tekstur, dan objek. Pada gambar, CNN dapat mempelajari pola dari piksel yang berdekatan. Selain untuk gambar, CNN juga dapat digunakan pada data deret waktu tertentu untuk menangkap pola lokal dalam urutan data. Namun, CNN tidak secara alami dirancang untuk mengingat dependensi jangka panjang seperti LSTM atau GRU (LeCun et al., 2015).

RNN dikembangkan untuk menangani data berurutan. Model ini memiliki mekanisme umpan balik yang memungkinkan informasi dari waktu sebelumnya memengaruhi proses pada waktu berikutnya. Karena itu, RNN cocok untuk teks, suara, sinyal, dan data *time series*. Namun, RNN dasar memiliki kelemahan dalam menangkap dependensi jangka panjang karena rentan terhadap masalah *vanishing gradient* (Goodfellow et al., 2016).

LSTM merupakan pengembangan dari RNN yang dirancang untuk mengatasi kelemahan tersebut. LSTM memiliki mekanisme *gate* yang mengatur informasi mana yang perlu disimpan, dilupakan, dan digunakan untuk menghasilkan output. Dengan mekanisme ini, LSTM lebih mampu menangkap pola jangka panjang dalam data sekuensial. Pada prediksi konsumsi energi, LSTM dapat mempelajari pola harian, mingguan, atau perubahan jangka panjang dari data historis (Hochreiter & Schmidhuber, 1997).

GRU juga merupakan pengembangan dari RNN. GRU memiliki tujuan yang mirip dengan LSTM, tetapi strukturnya lebih sederhana. Jumlah *gate* pada GRU lebih sedikit, sehingga proses pelatihan dapat lebih ringan. GRU sering digunakan ketika peneliti membutuhkan model sekuensial yang efisien, terutama pada dataset yang tidak terlalu besar atau ketika sumber daya komputasi terbatas (Cho et al., 2014).

Perbedaan utama kelima arsitektur tersebut dapat dilihat dari jenis data yang paling sesuai. ANN cocok untuk data umum atau tabular. CNN kuat untuk gambar dan pola lokal. RNN cocok untuk urutan pendek. LSTM dan GRU lebih sesuai untuk data *time series* karena mampu menangkap hubungan antarwaktu. Dalam buku ini, fokus utama diarahkan pada LSTM dan GRU

karena keduanya relevan untuk prediksi data *time series*, terutama ketika digabungkan dengan *fuzzy logic* untuk menghasilkan interpretasi keputusan.

Tabel 4.2: Perbandingan ANN, CNN, RNN, LSTM, Dan GRU

Arsitektur	Karakteristik Utama	Cocok Untuk	Keterbatasan
ANN	Jaringan saraf dasar tanpa memori urutan	Data tabular dan prediksi umum	Kurang cocok untuk data berurutan
CNN	Mengenali pola lokal melalui konvolusi	Gambar, sinyal, pola lokal	Tidak fokus pada dependensi jangka panjang
RNN	Memproses data berurutan dengan memori sederhana	Teks dan time series pendek	Rentan vanishing gradient
LSTM	RNN dengan mekanisme gate dan memori jangka panjang	Time series kompleks dan dependensi panjang	Struktur lebih kompleks
GRU	Versi lebih sederhana dari LSTM	Time series dengan komputasi lebih ringan	Tidak selalu lebih baik dari LSTM

Bab 5
Recurrent Neural Network, LSTM, dan GRU

5.1 Konsep Recurrent Neural Network

Recurrent Neural Network atau RNN adalah arsitektur jaringan saraf tiruan yang dirancang untuk memproses data berurutan. Berbeda dengan *Artificial Neural Network* biasa yang menganggap setiap input berdiri sendiri, RNN memiliki mekanisme untuk menyimpan informasi dari langkah sebelumnya. Mekanisme ini membuat RNN cocok digunakan untuk data yang memiliki urutan, seperti teks, suara, sinyal, dan data *time series*.

Pada data *time series*, urutan data sangat penting karena nilai pada suatu waktu sering berkaitan dengan nilai sebelumnya. Misalnya, konsumsi energi pukul 10.00 dapat dipengaruhi oleh konsumsi pukul 09.00, 08.00, atau pola konsumsi pada hari sebelumnya. RNN mencoba mempelajari hubungan tersebut dengan cara membawa informasi dari waktu sebelumnya ke waktu berikutnya. Konsep ini membuat RNN menjadi salah satu dasar penting dalam pemodelan data sekuensial (Elman, 1990).

Secara sederhana, RNN memiliki dua masukan pada setiap langkah waktu. Masukan pertama adalah data pada waktu saat ini, sedangkan masukan kedua adalah informasi tersembunyi dari waktu sebelumnya. Informasi tersembunyi ini disebut *hidden state*. *Hidden state* berperan sebagai memori sementara yang membantu model memahami konteks urutan data.

Secara matematis, proses sederhana RNN dapat ditulis sebagai berikut:

$$h_t = f(W_x x_t + W_h h_{t-1} + b)$$

$$y^t = g(Wyht + c)\, y\hat{}_t = g(W_y h_t + c)$$

Keterangan:

- x_t adalah input pada waktu ke-ttt.
- h_t adalah *hidden state* pada waktu ke-ttt.

- h_{t-1} adalah *hidden state* dari waktu sebelumnya.
- W_x, W_h, dan W_y adalah bobot model.
- b dan c adalah bias.
- $\hat{y}_t$ adalah output atau hasil prediksi.
- f dan g adalah fungsi aktivasi.

Rumus tersebut menunjukkan bahwa output RNN tidak hanya dipengaruhi oleh input saat ini, tetapi juga oleh informasi dari waktu sebelumnya. Inilah yang membedakan RNN dari jaringan saraf biasa. Dengan cara ini, RNN dapat membaca data secara bertahap dari awal sampai akhir urutan.

Dalam prediksi data *time series*, RNN dapat digunakan untuk mempelajari pola masa lalu dan menghasilkan prediksi masa depan. Misalnya, data konsumsi energi selama beberapa jam terakhir dapat dimasukkan ke RNN untuk memprediksi konsumsi energi satu jam berikutnya. RNN akan membaca nilai konsumsi energi secara berurutan dan memperbarui *hidden state* pada setiap langkah waktu.

Namun, RNN dasar memiliki keterbatasan. Model ini cukup baik untuk urutan pendek, tetapi sering mengalami kesulitan ketika harus mengingat informasi dari waktu yang jauh sebelumnya. Masalah ini muncul karena informasi lama dapat semakin melemah selama proses pelatihan. Keterbatasan tersebut kemudian mendorong pengembangan model lanjutan seperti Long Short-Term Memory (LSTM) dan Gated Recurrent Unit (GRU), yang dirancang agar lebih mampu menangani dependensi jangka panjang (Goodfellow et al., 2016).

Meskipun memiliki keterbatasan, RNN tetap penting dipelajari karena menjadi fondasi dari banyak arsitektur sekuensial modern. LSTM dan GRU pada dasarnya merupakan pengembangan dari RNN. Dengan memahami RNN, pembahasan tentang LSTM, GRU, dan model hibrida LSTM–Fuzzy Logic akan lebih mudah dipahami.

5.2 Masalah Vanishing Gradient

Salah satu kelemahan utama pada RNN dasar adalah masalah **vanishing gradient**. Masalah ini terjadi ketika nilai gradien menjadi sangat kecil selama proses pelatihan, terutama ketika model mempelajari data dengan

urutan panjang. Akibatnya, bobot pada bagian awal urutan sulit diperbarui, sehingga model kesulitan mengingat informasi dari waktu yang jauh sebelumnya (Bengio et al., 1994).

Untuk memahami masalah ini, perlu dipahami bahwa pelatihan jaringan saraf dilakukan melalui proses *backpropagation*. Pada RNN, proses ini disebut *Backpropagation Through Time* atau BPTT karena kesalahan model dihitung dan disebarkan kembali melalui banyak langkah waktu. Semakin panjang urutan data, semakin panjang pula jalur yang harus dilalui gradien.

Secara sederhana, pembaruan bobot dalam jaringan saraf bergantung pada nilai gradien. Jika gradien cukup besar dan stabil, model dapat memperbarui bobot dengan baik. Namun, jika gradien semakin kecil mendekati nol, pembaruan bobot menjadi sangat lambat. Dalam kondisi ini, model hampir tidak belajar dari informasi yang berada jauh di awal urutan.

Masalah tersebut dapat dijelaskan secara sederhana melalui perkalian berulang. Pada RNN, gradien dari waktu akhir ke waktu awal melibatkan banyak perkalian. Jika nilai yang dikalikan lebih kecil dari 1, maka hasil perkalian akan semakin kecil.

Contoh sederhana:

$$0{,}5 \times 0{,}5 \times 0{,}5 \times 0{,}5 = 0{,}0625$$

Jika proses perkalian terjadi puluhan atau ratusan kali, nilainya dapat menjadi sangat kecil. Kondisi inilah yang disebut *vanishing gradient*. Gradien "menghilang" sebelum mencapai bagian awal urutan, sehingga model gagal memperbarui bobot yang berhubungan dengan informasi lama.

Dalam data *time series*, masalah ini cukup serius. Banyak pola penting tidak hanya muncul pada waktu terdekat, tetapi juga pada periode yang lebih jauh. Misalnya, konsumsi energi hari ini dapat dipengaruhi oleh pola konsumsi minggu sebelumnya. Jika RNN tidak mampu menyimpan informasi jangka panjang, maka prediksi yang dihasilkan dapat kurang akurat.

Selain *vanishing gradient*, terdapat juga masalah **exploding gradient**. Masalah ini terjadi ketika nilai gradien menjadi terlalu besar selama proses pelatihan. Jika *vanishing gradient* membuat model sulit belajar, maka *exploding gradient* membuat pelatihan menjadi tidak stabil. Nilai *loss* dapat melonjak dan bobot model berubah terlalu ekstrem (Pascanu et al., 2013).

Dampak *vanishing gradient* paling terlihat ketika RNN digunakan untuk data dengan dependensi panjang. Misalnya, dalam prediksi konsumsi energi, model mungkin mampu menangkap hubungan antara jam saat ini dan jam sebelumnya, tetapi gagal menangkap pola harian atau mingguan. Pada pemrosesan bahasa alami, RNN mungkin mampu memahami kata yang berdekatan, tetapi kesulitan memahami hubungan antara kata di awal dan akhir kalimat panjang.

Untuk mengatasi masalah ini, dikembangkan arsitektur LSTM. LSTM memiliki mekanisme *cell state* dan *gate* yang membantu mengatur aliran informasi. Dengan mekanisme tersebut, informasi penting dapat dipertahankan lebih lama, sedangkan informasi yang tidak relevan dapat dilupakan. Inilah alasan LSTM sering digunakan untuk data *time series* yang memiliki pola jangka panjang (Hochreiter & Schmidhuber, 1997).

GRU juga dikembangkan sebagai alternatif yang lebih sederhana dari LSTM. GRU menggunakan mekanisme *gate* yang lebih ringkas, tetapi tetap bertujuan membantu model menyimpan informasi penting dalam urutan data. Baik LSTM maupun GRU merupakan jawaban terhadap keterbatasan RNN dasar dalam mempelajari dependensi jangka panjang.

Beberapa cara yang dapat digunakan untuk mengurangi masalah *vanishing gradient* dan *exploding gradient* antara lain penggunaan LSTM atau GRU, pemilihan fungsi aktivasi yang sesuai, normalisasi data, pengaturan *learning rate*, serta teknik *gradient clipping*. Pada praktiknya, penggunaan LSTM atau GRU menjadi pilihan yang paling umum ketika data memiliki urutan panjang dan pola temporal yang kompleks.

Tabel 5.1: Masalah Gradien Pada RNN

Masalah	Penyebab Utama	Dampak Pada Model	Solusi Umum
Vanishing Gradient	Gradien semakin kecil saat disebarkan ke waktu sebelumnya	Model sulit belajar dependensi jangka panjang	LSTM, GRU, normalisasi, pengaturan learning rate
Exploding Gradient	Gradien menjadi terlalu besar	Pelatihan tidak stabil dan loss melonjak	Gradient clipping dan learning rate lebih kecil

Pemahaman tentang *vanishing gradient* penting sebelum membahas LSTM dan GRU. Kedua arsitektur tersebut tidak muncul secara tiba-tiba, tetapi dikembangkan untuk mengatasi kelemahan RNN dasar. Dengan demikian, LSTM dan GRU dapat dipahami sebagai pengembangan RNN yang lebih kuat untuk menangani data sekuensial, terutama data *time series* dengan pola jangka panjang.

5.3 Arsitektur Long Short-Term Memory

Long Short-Term Memory atau LSTM adalah pengembangan dari Recurrent Neural Network (RNN) yang dirancang untuk mengatasi kesulitan dalam mempelajari dependensi jangka panjang. Pada RNN dasar, informasi dari waktu yang jauh sebelumnya sering melemah ketika diproses melalui banyak langkah waktu. Akibatnya, model sulit mengingat pola lama yang sebenarnya penting untuk prediksi. LSTM diperkenalkan oleh Hochreiter dan Schmidhuber untuk membantu model mempertahankan informasi penting dalam urutan data yang panjang (Hochreiter & Schmidhuber, 1997).

Perbedaan utama LSTM dibandingkan RNN biasa terletak pada adanya **cell state** dan beberapa **gate**. *Cell state* dapat dipahami sebagai jalur memori utama yang membawa informasi dari satu waktu ke waktu berikutnya. Sementara itu, *gate* berfungsi sebagai pengatur informasi. Melalui mekanisme ini, LSTM dapat menentukan informasi mana yang perlu disimpan, informasi mana yang perlu dilupakan, dan informasi mana yang digunakan untuk menghasilkan output.

Secara umum, LSTM memiliki tiga komponen *gate* utama, yaitu **forget gate**, **input gate**, dan **output gate**. Ketiga *gate* ini bekerja bersama untuk mengatur aliran informasi di dalam model. Mekanisme tersebut membuat LSTM lebih fleksibel dibandingkan RNN dasar karena model tidak harus membawa seluruh informasi lama, tetapi dapat memilih informasi yang paling relevan untuk proses prediksi.

5.3.1 Cell State Sebagai Memori Utama

Cell state merupakan bagian penting dalam LSTM. Bagian ini berfungsi seperti jalur memori yang membawa informasi sepanjang urutan waktu. Jika suatu informasi dianggap penting, LSTM dapat mempertahankannya dalam

cell state. Jika informasi dianggap tidak relevan, LSTM dapat mengurangi atau menghapus pengaruhnya.

Pada prediksi data *time series*, *cell state* membantu model mengingat pola yang terjadi dalam jangka panjang. Misalnya, data konsumsi energi tidak hanya dipengaruhi oleh satu jam sebelumnya, tetapi juga dapat dipengaruhi oleh pola harian, pola mingguan, atau kebiasaan penggunaan energi pada periode tertentu. Dengan *cell state*, LSTM memiliki kemampuan lebih baik untuk menyimpan informasi semacam itu dibandingkan RNN biasa.

5.3.2 Forget Gate

Forget gate bertugas menentukan informasi lama mana yang perlu dilupakan dari *cell state*. Gate ini menerima input saat ini dan *hidden state* sebelumnya, kemudian menghasilkan nilai antara 0 dan 1. Nilai mendekati 0 berarti informasi cenderung dilupakan, sedangkan nilai mendekati 1 berarti informasi dipertahankan.

Secara sederhana, *forget gate* dapat ditulis sebagai berikut:

ft=σ(Wf[ht−1,xt]+bf)

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Keterangan:

- f_t adalah keluaran *forget gate* pada waktu ke-ttt.
- σ adalah fungsi sigmoid.
- W_f adalah bobot pada *forget gate*.
- h_{t-1} adalah *hidden state* sebelumnya.
- x_t adalah input saat ini.
- b_f adalah bias.

Dalam konteks konsumsi energi, *forget gate* dapat membantu model mengabaikan pola yang sudah tidak relevan. Misalnya, pola konsumsi saat hari libur mungkin tidak selalu relevan untuk memprediksi konsumsi pada hari kerja.

5.3.3 Input Gate

Input gate bertugas menentukan informasi baru yang akan dimasukkan ke dalam *cell state*. Gate ini membantu LSTM memilih informasi dari input saat ini yang dianggap penting untuk disimpan. Proses ini biasanya melibatkan dua bagian, yaitu menentukan nilai informasi yang akan diperbarui dan membentuk kandidat nilai baru untuk *cell state*.

Rumus sederhananya adalah:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

Keterangan:

- i_t adalah keluaran *input gate*.
- $\tilde{C}_t$ adalah kandidat informasi baru untuk *cell state*.
- $tanh$ adalah fungsi aktivasi yang menghasilkan nilai antara -1 dan 1.

Setelah itu, *cell state* diperbarui dengan menggabungkan informasi lama dan informasi baru:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

Rumus tersebut menunjukkan bahwa LSTM tidak langsung mengganti memori lama dengan memori baru. Model terlebih dahulu menentukan bagian lama yang dipertahankan dan bagian baru yang perlu ditambahkan.

5.3.4 Output Gate

Output gate menentukan informasi dari *cell state* yang akan digunakan sebagai keluaran atau *hidden state* pada waktu saat ini. Dengan kata lain, *output gate* memilih bagian informasi yang relevan untuk menghasilkan prediksi atau diteruskan ke langkah waktu berikutnya.

Rumusnya dapat ditulis sebagai berikut:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(C_t)$$

Keterangan:

- o_t adalah keluaran *output gate*.

- h_t adalah *hidden state* pada waktu ke-ttt.
- C_t adalah *cell state* yang sudah diperbarui.

Pada prediksi data *time series*, *hidden state* ini dapat digunakan untuk menghasilkan prediksi nilai berikutnya. Misalnya, setelah membaca data konsumsi energi beberapa jam terakhir, LSTM menghasilkan representasi pola yang kemudian digunakan untuk memprediksi konsumsi energi satu jam ke depan.

5.3.5 Alur Kerja LSTM Dalam Prediksi Time Series

Alur kerja LSTM dalam prediksi *time series* dapat dijelaskan secara bertahap. Pertama, data historis dibentuk menjadi *sequence* atau *window*. Kedua, setiap nilai dalam *sequence* dibaca oleh LSTM secara berurutan. Ketiga, pada setiap langkah waktu, LSTM memperbarui *cell state* dan *hidden state* menggunakan mekanisme *gate*. Keempat, setelah seluruh urutan diproses, model menghasilkan prediksi.

Misalnya, data konsumsi energi selama 24 jam terakhir digunakan untuk memprediksi konsumsi energi pada jam berikutnya. LSTM akan membaca data dari jam pertama sampai jam ke-24, lalu menggunakan informasi yang dianggap penting untuk menghasilkan prediksi. Jika terdapat pola penggunaan energi yang berulang setiap hari, LSTM dapat mempelajari pola tersebut selama proses pelatihan.

Tabel 5.2: Tabel 5.2 Komponen Utama Dalam Arsitektur LSTM

Komponen	Fungsi Utama
Cell State	Menyimpan informasi penting dalam jangka panjang
Hidden State	Membawa informasi keluaran dari satu waktu ke waktu berikutnya
Forget Gate	Menentukan informasi lama yang perlu dilupakan
Input Gate	Menentukan informasi baru yang perlu disimpan
Output Gate	Menentukan informasi yang digunakan sebagai keluaran
Fungsi Sigmoid	Menghasilkan nilai pengontrol antara 0 dan 1
Fungsi Tanh	Membentuk nilai kandidat dan representasi informasi

Arsitektur LSTM menjadi penting karena mampu menangani pola sekuensial yang panjang dan kompleks. Dalam prediksi data *time series*,

kemampuan ini sangat berguna ketika data memiliki pola jangka pendek dan jangka panjang secara bersamaan. LSTM tidak hanya membaca nilai terakhir, tetapi juga mempertimbangkan informasi historis yang relevan. Hal ini menjadikan LSTM sebagai salah satu model utama dalam prediksi konsumsi energi, cuaca, trafik, sinyal, dan berbagai data berbasis waktu lainnya.

5.4 Komponen Gate Pada LSTM

Komponen utama yang membedakan Long Short-Term Memory atau LSTM dari RNN dasar adalah keberadaan **gate**. Gate berfungsi sebagai mekanisme pengatur informasi. Melalui gate, LSTM dapat menentukan informasi mana yang perlu disimpan, informasi mana yang perlu dilupakan, dan informasi mana yang perlu digunakan untuk menghasilkan output. Kemampuan ini membuat LSTM lebih kuat dalam mempelajari data sekuensial yang memiliki dependensi jangka panjang.

Pada RNN biasa, informasi dari waktu sebelumnya diteruskan secara langsung ke waktu berikutnya. Cara ini sederhana, tetapi memiliki kelemahan karena model sulit memilih informasi yang masih relevan dan informasi yang sudah tidak diperlukan. LSTM mengatasi masalah tersebut dengan menambahkan mekanisme kontrol. Gate bekerja seperti "pintu" yang dapat membuka atau menutup aliran informasi. Nilai gate biasanya berada pada rentang 0 sampai 1. Nilai mendekati 0 berarti informasi ditahan atau dilupakan, sedangkan nilai mendekati 1 berarti informasi diteruskan (Hochreiter & Schmidhuber, 1997).

Secara umum, LSTM memiliki tiga gate utama, yaitu **forget gate**, **input gate**, dan **output gate**. Ketiganya bekerja bersama dengan **cell state** dan **hidden state**. Cell state berperan sebagai memori utama, sedangkan hidden state berperan sebagai informasi keluaran yang diteruskan ke langkah waktu berikutnya.

5.4.1 Forget Gate

Forget gate berfungsi menentukan bagian informasi lama yang perlu dilupakan dari cell state. Pada setiap langkah waktu, LSTM menerima input saat ini (xt)(x_t)(xt) dan hidden state sebelumnya (h_{t-1}). Kedua informasi tersebut diproses untuk menghasilkan nilai forget gate (f_t).

Rumus forget gate dapat ditulis sebagai berikut:

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

Keterangan:

- f_t adalah keluaran forget gate.
- σ adalah fungsi sigmoid.
- W_f adalah bobot pada forget gate.
- h_{t-1} adalah hidden state sebelumnya.
- x_t adalah input pada waktu saat ini.
- b_f adalah bias.

Fungsi sigmoid menghasilkan nilai antara 0 dan 1. Jika nilai forget gate mendekati 0, informasi lama cenderung dilupakan. Jika nilainya mendekati 1, informasi lama tetap dipertahankan. Pada prediksi konsumsi energi, forget gate dapat membantu model mengabaikan pola lama yang sudah tidak relevan, misalnya pola konsumsi pada hari libur ketika model sedang memprediksi konsumsi hari kerja.

5.4.2 Input Gate

Input gate berfungsi menentukan informasi baru yang akan dimasukkan ke dalam cell state. Gate ini penting karena tidak semua informasi baru perlu disimpan. Sebagian input mungkin penting untuk prediksi berikutnya, tetapi sebagian lainnya hanya berupa noise atau perubahan sementara.

Input gate bekerja bersama kandidat cell state. Rumusnya dapat ditulis sebagai berikut:

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$$

Keterangan:

- i_t adalah keluaran input gate.
- $\tilde{C}_t$ adalah kandidat informasi baru yang akan dimasukkan ke cell state.
- $tanh$ adalah fungsi aktivasi yang menghasilkan nilai antara -1 dan 1.
- W_i, W_C, b_i, dan b_C adalah parameter model.

Input gate menentukan seberapa besar kandidat informasi baru akan digunakan. Jika informasi baru dianggap penting, nilainya akan lebih besar. Jika kurang penting, nilainya akan lebih kecil. Misalnya, jika terjadi lonjakan konsumsi energi yang benar-benar mencerminkan perubahan aktivitas, input gate dapat menyimpan informasi tersebut. Namun, jika lonjakan hanya disebabkan oleh gangguan pencatatan, model dapat belajar untuk tidak terlalu menekankannya.

5.4.3 Pembaruan Cell State

Setelah forget gate dan input gate bekerja, LSTM memperbarui cell state. Proses ini menggabungkan informasi lama yang masih dipertahankan dengan informasi baru yang dianggap penting. Rumus pembaruan cell state adalah:

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t$$

Keterangan:

- C_t adalah cell state pada waktu saat ini.
- C_{t-1} adalah cell state sebelumnya.
- f_t menentukan bagian informasi lama yang dipertahankan.
- i_t menentukan bagian informasi baru yang dimasukkan.
- $\tilde{C}_t$ adalah kandidat informasi baru.

Rumus ini menunjukkan bahwa LSTM tidak mengganti seluruh memori lama secara langsung. Model memilih bagian mana yang dipertahankan dan bagian mana yang diperbarui. Mekanisme inilah yang membuat LSTM mampu menyimpan informasi penting dalam jangka panjang (Gers et al., 2000).

5.4.4 Output Gate

Output gate berfungsi menentukan informasi dari cell state yang akan digunakan sebagai keluaran. Keluaran ini disebut hidden state (ht)(h_t)(ht). Hidden state kemudian dapat diteruskan ke langkah waktu berikutnya atau digunakan untuk menghasilkan prediksi.

Rumus output gate adalah:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t \times \tanh(C_t)\, ht = ot \times tanh(Ct)$$

Keterangan:

- o_t adalah keluaran output gate.
- h_t adalah hidden state pada waktu saat ini.
- $C_t$adalah cell state yang sudah diperbarui.

Output gate membantu model memilih bagian informasi yang paling relevan untuk digunakan. Pada prediksi time series, hidden state dapat digunakan untuk memperkirakan nilai berikutnya. Misalnya, setelah membaca data konsumsi energi 24 jam terakhir, LSTM menggunakan hidden state terakhir untuk memprediksi konsumsi energi pada jam berikutnya.

Ketiga gate dalam LSTM bekerja secara saling melengkapi. Forget gate mengontrol informasi lama, input gate mengontrol informasi baru, dan output gate mengontrol informasi yang digunakan sebagai keluaran. Dengan mekanisme ini, LSTM dapat mempelajari pola jangka pendek dan jangka panjang secara lebih baik dibandingkan RNN dasar.

Tabel 5.3: Peran Gate Pada LSTM

Komponen	Fungsi Utama	Analogi Sederhana
Forget Gate	Menentukan informasi lama yang dilupakan	Menyaring memori lama
Input Gate	Menentukan informasibaru yang disimpan	Memilih informasi baru yang penting
Cell State	Menyimpan informasi jangka panjang	Jalur memori utama
Output Gate	Menentukan informasi yang menjadi keluaran	Menentukan informasi yang digunakan untuk prediksi
Hidden State	Membawa keluaran ke langkah berikutnya	Ringkasan informasi saat ini

5.5 Gated Recurrent Unit Sebagai Alternatif LSTM

Gated Recurrent Unit atau GRU adalah arsitektur jaringan saraf berulang yang dikembangkan sebagai alternatif dari LSTM. Sama seperti LSTM, GRU dirancang untuk mengatasi kelemahan RNN dasar dalam mempelajari dependensi jangka panjang. Perbedaannya, GRU memiliki struktur yang lebih sederhana karena menggunakan jumlah gate yang lebih sedikit. Struktur yang lebih ringkas membuat GRU sering lebih ringan secara komputasi dan lebih cepat dilatih dibandingkan LSTM (Cho et al., 2014).

GRU tidak memiliki cell state terpisah seperti LSTM. Informasi memori pada GRU dibawa melalui hidden state. Dengan demikian, GRU menggabungkan konsep memori dan keluaran dalam satu jalur utama. Meskipun lebih sederhana, GRU tetap memiliki mekanisme pengendalian informasi melalui dua gate utama, yaitu **update gate** dan **reset gate**.

5.5.1 Update Gate

Update gate berfungsi menentukan seberapa banyak informasi lama yang akan dipertahankan dan seberapa banyak informasi baru yang akan digunakan. Gate ini memiliki fungsi yang mirip dengan gabungan forget gate dan input gate pada LSTM. Jika update gate bernilai besar, informasi lama lebih banyak dipertahankan. Jika nilainya kecil, informasi baru lebih banyak digunakan.

Rumus update gate dapat ditulis sebagai berikut:

$$z_t = \sigma(W_z[h_{t-1}, x_t])$$

Keterangan:

- z_t adalah keluaran update gate.
- σ adalah fungsi sigmoid.
- W_z adalah bobot update gate.
- h_{t-1} adalah hidden state sebelumnya.
- x_t adalah input saat ini.

Dalam prediksi data time series, update gate membantu GRU mempertahankan pola penting dari masa lalu. Misalnya, jika pola konsumsi

energi pada jam sebelumnya masih relevan untuk prediksi berikutnya, update gate dapat mempertahankan informasi tersebut.

5.5.2 Reset Gate

Reset gate berfungsi menentukan seberapa banyak informasi lama yang perlu diabaikan ketika membentuk kandidat informasi baru. Jika reset gate bernilai kecil, maka informasi lama cenderung diabaikan. Jika nilainya besar, informasi lama tetap digunakan dalam pembentukan informasi baru.

Rumus reset gate adalah:

$$r_t = \sigma(W_r[h_{t-1}, x_t])$$

Keterangan:

- r_t adalah keluaran reset gate.
- W_r adalah bobot reset gate.

Setelah reset gate dihitung, GRU membentuk kandidat hidden state baru:

$$\tilde{h}_t = \tanh(W_h[r_t \times h_{t-1}, x_t])$$

Kemudian hidden state akhir diperbarui dengan rumus:

$$h_t = (1 - z_t) \times h_{t-1} + z_t \times \tilde{h}_t$$

Rumus tersebut menunjukkan bahwa GRU menyeimbangkan informasi lama dan informasi baru melalui update gate. Dengan cara ini, GRU tetap dapat mempelajari urutan data meskipun strukturnya lebih sederhana daripada LSTM.

5.5.3 Perbandingan GRU Dan LSTM

GRU dan LSTM sama-sama digunakan untuk data sekuensial. Keduanya mampu menangani masalah vanishing gradient lebih baik dibandingkan RNN dasar. Namun, terdapat beberapa perbedaan penting. LSTM memiliki cell state, hidden state, forget gate, input gate, dan output gate. GRU hanya memiliki hidden state, update gate, dan reset gate. Karena itu, GRU biasanya memiliki jumlah parameter lebih sedikit.

Jumlah parameter yang lebih sedikit dapat membuat GRU lebih cepat dilatih dan lebih hemat sumber daya komputasi. Hal ini bermanfaat ketika dataset

tidak terlalu besar atau ketika perangkat komputasi terbatas. Namun, LSTM dapat lebih unggul pada data yang memiliki pola jangka panjang sangat kompleks karena memiliki struktur memori yang lebih eksplisit (Goodfellow et al., 2016).

Pemilihan antara LSTM dan GRU tidak sebaiknya dilakukan hanya berdasarkan teori. Keduanya perlu diuji pada dataset yang sama menggunakan metrik evaluasi yang sesuai. Pada beberapa kasus, GRU dapat memberikan hasil yang sebanding dengan LSTM. Pada kasus lain, LSTM dapat memberikan prediksi yang lebih stabil. Oleh karena itu, perbandingan eksperimen menjadi langkah penting dalam penelitian prediksi time series.

Tabel 5.4: Tabel 5.4 Perbandingan LSTM Dan GRU

Aspek	LSTM	GRU
Struktur memori	Memiliki cell state dan hidden state	Hanya menggunakan hidden state
Gate utama	Forget gate, input gate, output gate	Update gate dan reset gate
Kompleksitas	Lebih kompleks	Lebih sederhana
Jumlah parameter	Lebih banyak	Lebih sedikit
Kecepatan pelatihan	Cenderung lebih lambat	Cenderung lebih cepat
Kesesuaian	Pola jangka panjang yang kompleks	Model sekuensial yang lebih ringan

Dalam konteks prediksi data time series, GRU dapat dipahami sebagai pilihan yang lebih ringkas dari LSTM. Jika tujuan penelitian adalah memperoleh model yang lebih sederhana dan efisien, GRU layak dipertimbangkan. Namun, jika data memiliki pola jangka panjang yang kompleks, LSTM tetap menjadi pilihan yang kuat. Keduanya dapat menjadi dasar model prediksi numerik sebelum hasilnya diterjemahkan lebih lanjut menggunakan fuzzy logic.

Bab 6
Fuzzy Logic untuk Penalaran Ketidakpastian

6.1 Pengertian Fuzzy Logic

Fuzzy logic atau logika fuzzy adalah pendekatan penalaran yang digunakan untuk menangani kondisi yang tidak sepenuhnya pasti, tidak sepenuhnya benar, atau tidak sepenuhnya salah. Dalam logika klasik, suatu pernyataan biasanya hanya memiliki dua kemungkinan nilai, yaitu benar atau salah. Misalnya, sebuah sistem dapat menyatakan bahwa konsumsi energi "tinggi" jika nilainya lebih dari 500 kWh, dan "tidak tinggi" jika nilainya kurang dari atau sama dengan 500 kWh. Pendekatan seperti ini disebut tegas atau *crisp* karena batas antar-kategorinya dibuat secara mutlak.

Dalam kehidupan nyata, banyak kondisi tidak selalu dapat dibatasi secara tegas. Konsumsi energi sebesar 499 kWh dan 501 kWh sebenarnya tidak terlalu berbeda jauh, tetapi jika menggunakan batas tegas, keduanya dapat masuk ke kategori yang berbeda. Nilai 499 kWh dianggap "tidak tinggi", sedangkan 501 kWh dianggap "tinggi". Kondisi seperti ini menunjukkan perlunya pendekatan yang lebih lentur dalam menilai data. *Fuzzy logic* hadir untuk menangani situasi seperti ini dengan memberi ruang bagi nilai yang berada di antara dua kategori.

Konsep *fuzzy logic* berawal dari gagasan himpunan fuzzy yang diperkenalkan oleh Lotfi A. Zadeh. Zadeh menjelaskan bahwa suatu objek tidak harus hanya menjadi anggota atau bukan anggota dari suatu himpunan, tetapi dapat memiliki derajat keanggotaan tertentu (Zadeh, 1965). Dengan kata lain, suatu nilai dapat sebagian termasuk dalam kategori "rendah", sebagian termasuk "sedang", atau sebagian mendekati "tinggi". Cara berpikir ini lebih dekat dengan cara manusia membuat penilaian sehari-hari.

Sebagai contoh, suhu 30°C dapat dianggap "cukup panas", tetapi belum tentu "sangat panas". Konsumsi energi 480 kWh dapat dianggap mendekati "tinggi", tetapi masih memiliki unsur "sedang". Dalam *fuzzy logic*, kondisi seperti ini dapat dimodelkan menggunakan derajat keanggotaan. Derajat keanggotaan biasanya berada pada rentang 0 sampai 1. Nilai 0 berarti tidak menjadi anggota kategori, nilai 1 berarti sepenuhnya menjadi anggota

kategori, sedangkan nilai di antara 0 dan 1 menunjukkan tingkat keanggotaan sebagian.

Pada sistem prediksi, *fuzzy logic* sering digunakan untuk membantu interpretasi hasil. Model *deep learning* seperti LSTM dapat menghasilkan prediksi numerik, misalnya konsumsi energi sebesar 650 kWh. Angka tersebut kemudian dapat diproses oleh sistem fuzzy untuk menentukan apakah konsumsi energi termasuk rendah, sedang, atau tinggi. Dengan demikian, *fuzzy logic* tidak hanya membantu mengelompokkan angka, tetapi juga membantu mengubah hasil prediksi menjadi informasi yang lebih mudah digunakan dalam pengambilan keputusan.

Kelebihan utama *fuzzy logic* adalah kemampuannya merepresentasikan penalaran linguistik. Istilah seperti "rendah", "sedang", "tinggi", "cepat", "lambat", "dekat", atau "jauh" dapat dimodelkan secara matematis. Hal ini membuat *fuzzy logic* banyak digunakan dalam sistem kendali, sistem pakar, pengambilan keputusan, klasifikasi, dan interpretasi hasil prediksi. Ross (2010) menjelaskan bahwa *fuzzy logic* memiliki manfaat besar dalam bidang rekayasa karena mampu menangani ketidakpastian dan ambiguitas yang sering muncul dalam sistem nyata.

Dalam konteks buku ini, *fuzzy logic* digunakan sebagai pelengkap *deep learning*. *Deep learning* bertugas mempelajari pola data dan menghasilkan prediksi, sedangkan *fuzzy logic* membantu menjelaskan makna dari hasil prediksi tersebut. Kombinasi ini penting karena sistem prediksi yang baik tidak hanya harus akurat, tetapi juga perlu menghasilkan keluaran yang dapat dipahami oleh pengguna.

6.2 Himpunan Fuzzy Dan Derajat Keanggotaan

Himpunan fuzzy adalah dasar utama dalam *fuzzy logic*. Untuk memahami himpunan fuzzy, perlu dibedakan terlebih dahulu antara **himpunan tegas** dan **himpunan fuzzy**. Pada himpunan tegas, suatu nilai hanya memiliki dua kemungkinan, yaitu menjadi anggota atau bukan anggota. Misalnya, jika kategori "beban tinggi" ditentukan untuk konsumsi energi di atas 500 kWh, maka nilai 600 kWh termasuk anggota kategori tinggi, sedangkan 400 kWh tidak termasuk anggota kategori tinggi.

Pendekatan tersebut sederhana, tetapi kurang fleksibel untuk menggambarkan kondisi nyata. Dalam banyak kasus, batas antar-kategori

tidak selalu jelas. Konsumsi energi 490 kWh dan 510 kWh memiliki perbedaan kecil, tetapi dapat masuk ke kategori yang berbeda jika menggunakan batas tegas. Himpunan fuzzy mengatasi masalah ini dengan memperbolehkan suatu nilai memiliki derajat keanggotaan sebagian dalam suatu kategori (Klir & Yuan, 1995).

Derajat keanggotaan menunjukkan seberapa kuat suatu nilai termasuk dalam suatu himpunan fuzzy. Derajat ini biasanya ditulis dengan simbol μ(x)\mu(x)μ(x), yaitu fungsi keanggotaan dari nilai xxx. Nilainya berada dalam rentang 0 sampai 1.

0≤μ(x)≤10 \leq \mu(x) \leq 10≤μ(x)≤1

Keterangan:

μ(x)\mu(x)μ(x) adalah derajat keanggotaan nilai xxx.

Nilai 0 berarti xxx tidak termasuk dalam himpunan fuzzy.

Nilai 1 berarti xxx sepenuhnya termasuk dalam himpunan fuzzy.

Nilai antara 0 dan 1 berarti xxx termasuk sebagian dalam himpunan fuzzy.

Sebagai contoh, dalam sistem prediksi konsumsi energi, kategori beban dapat dibagi menjadi “rendah”, “sedang”, dan “tinggi”. Nilai konsumsi energi 300 kWh mungkin memiliki derajat keanggotaan 0,8 pada kategori “rendah” dan 0,2 pada kategori “sedang”. Nilai 500 kWh mungkin memiliki derajat keanggotaan 0,6 pada kategori “sedang” dan 0,4 pada kategori “tinggi”. Artinya, satu nilai dapat memiliki hubungan dengan lebih dari satu kategori, tergantung fungsi keanggotaan yang digunakan.

Konsep ini berbeda dari klasifikasi biasa yang memaksa satu nilai masuk ke satu kategori saja. Dalam *fuzzy logic*, batas antar-kategori dibuat lebih halus. Hal ini sangat berguna pada data yang memiliki ketidakpastian, perubahan bertahap, atau nilai yang berada di sekitar batas kategori. Pada prediksi data *time series*, pendekatan seperti ini membantu sistem menafsirkan hasil prediksi secara lebih realistis.

Fungsi keanggotaan digunakan untuk menentukan bentuk hubungan antara nilai input dan derajat keanggotaannya. Beberapa bentuk fungsi keanggotaan yang sering digunakan adalah segitiga, trapesium, Gaussian, dan sigmoid. Fungsi segitiga dan trapesium sering digunakan karena sederhana dan

mudah dipahami. Fungsi Gaussian dan sigmoid lebih halus, tetapi biasanya membutuhkan pemahaman matematis yang lebih kuat (Ross, 2010).

Contoh sederhana fungsi keanggotaan segitiga untuk kategori "sedang" dapat digambarkan sebagai berikut: nilai rendah memiliki derajat keanggotaan kecil, nilai mendekati pusat kategori memiliki derajat keanggotaan tinggi, dan nilai yang semakin jauh dari pusat kategori kembali memiliki derajat keanggotaan kecil. Jika pusat kategori "sedang" berada pada 500 kWh, maka nilai 500 kWh dapat memiliki derajat keanggotaan 1, sedangkan nilai 300 kWh dan 700 kWh memiliki derajat keanggotaan lebih kecil.

Dalam penerapan LSTM–Fuzzy Logic, himpunan fuzzy digunakan setelah model LSTM menghasilkan prediksi numerik. Misalnya, LSTM memprediksi konsumsi energi satu jam ke depan sebesar 620 kWh. Nilai ini kemudian dimasukkan ke dalam sistem fuzzy. Sistem fuzzy menghitung derajat keanggotaan nilai tersebut terhadap kategori "rendah", "sedang", dan "tinggi". Jika derajat keanggotaan tertinggi berada pada kategori "tinggi", maka sistem dapat menyimpulkan bahwa beban energi diprediksi tinggi.

Agar lebih mudah dipahami, perbedaan antara himpunan tegas dan himpunan fuzzy dapat dilihat pada tabel berikut.

Tabel 6.1:Tabel 6.1 Perbedaan Himpunan Tegas Dan Himpunan Fuzzy

Aspek	Himpunan Tegas	Himpunan Fuzzy
Keanggotaan	Hanya 0 atau 1	Bernilai antara 0 sampai 1
Batas kategori	Kaku dan tegas	Lentur dan bertahap
Contoh	501 kWh = tinggi, 499 kWh = tidak tinggi	499 kWh dapat sebagian sedang dan sebagian tinggi
Kelebihan	Sederhana	Lebih realistis untuk data tidak pasti
Keterbatasan	Kurang fleksibel	Membutuhkan fungsi keanggotaan

Himpunan fuzzy dan derajat keanggotaan menjadi fondasi penting sebelum membahas fungsi keanggotaan, aturan fuzzy, inferensi fuzzy, dan defuzzifikasi. Tanpa memahami dua konsep ini, sistem fuzzy akan sulit dipahami karena seluruh proses penalaran fuzzy bergantung pada cara nilai numerik diubah menjadi derajat keanggotaan linguistik.

6.3 Fungsi Keanggotaan Dalam Fuzzy Logic

Fungsi keanggotaan adalah fungsi yang digunakan untuk menentukan derajat keanggotaan suatu nilai terhadap himpunan fuzzy tertentu. Jika himpunan fuzzy menjelaskan kategori seperti rendah, sedang, atau tinggi, maka fungsi keanggotaan menentukan seberapa kuat suatu nilai masuk ke dalam kategori tersebut. Nilai derajat keanggotaan berada pada rentang 0 sampai 1. Nilai 0 berarti tidak termasuk dalam kategori, nilai 1 berarti sepenuhnya termasuk, sedangkan nilai di antara 0 dan 1 menunjukkan keanggotaan sebagian.

Dalam sistem prediksi konsumsi energi, fungsi keanggotaan dapat digunakan untuk menafsirkan nilai numerik hasil prediksi. Misalnya, prediksi konsumsi energi sebesar 450 kWh dapat memiliki derajat keanggotaan 0,7 pada kategori "sedang" dan 0,3 pada kategori "tinggi". Artinya, nilai tersebut belum sepenuhnya tinggi, tetapi sudah mulai mendekati kategori tinggi. Cara ini membuat sistem fuzzy lebih fleksibel dibandingkan sistem klasifikasi tegas yang hanya memberi satu label secara mutlak.

Beberapa bentuk fungsi keanggotaan yang umum digunakan adalah **segitiga**, **trapesium**, **Gaussian**, dan **sigmoid**. Fungsi segitiga dan trapesium sering dipakai karena sederhana, mudah divisualisasikan, dan mudah diterapkan dalam sistem pakar. Fungsi Gaussian dan sigmoid memiliki bentuk kurva yang lebih halus sehingga dapat digunakan ketika perubahan derajat keanggotaan ingin dibuat lebih bertahap (Ross, 2010).

Fungsi keanggotaan segitiga dapat ditulis secara sederhana sebagai berikut:

$$\mu(x) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a < x \leq b \\ \frac{c-x}{c-b}, & b < x < c \\ 0, & x \geq c \end{cases}$$

Keterangan:

- x adalah nilai input.
- $\mu(x)$ adalah derajat keanggotaan.
- a adalah batas bawah.
- b adalah titik tengah dengan derajat keanggotaan maksimum.

- ccc adalah batas atas.

Sebagai contoh, kategori "sedang" pada konsumsi energi dapat dibuat dengan batas 300 kWh, titik tengah 500 kWh, dan batas atas 700 kWh. Nilai 500 kWh memiliki derajat keanggotaan tertinggi pada kategori "sedang", sedangkan nilai yang menjauh dari 500 kWh memiliki derajat keanggotaan yang semakin kecil.

Pemilihan bentuk fungsi keanggotaan harus disesuaikan dengan karakteristik data dan kebutuhan sistem. Jika sistem membutuhkan interpretasi sederhana, fungsi segitiga atau trapesium sudah cukup. Namun, jika data memiliki perubahan yang lebih halus, fungsi Gaussian atau sigmoid dapat dipertimbangkan. Dalam model LSTM–Fuzzy Logic, fungsi keanggotaan berperan setelah LSTM menghasilkan prediksi numerik. Nilai prediksi tersebut kemudian dipetakan ke dalam kategori fuzzy agar hasilnya lebih mudah dipahami.

6.4 Variabel Linguistik Dalam Fuzzy Logic

Variabel linguistik adalah variabel yang nilainya dinyatakan dalam bentuk istilah bahasa manusia, bukan hanya angka. Dalam fuzzy logic, istilah seperti rendah, sedang, tinggi, lambat, cepat, dekat, jauh, aman, waspada, dan kritis dapat digunakan sebagai nilai dari suatu variabel. Konsep ini membuat fuzzy logic mudah dipahami karena cara kerjanya mendekati pola penalaran manusia (Zadeh, 1975).

Pada sistem prediksi konsumsi energi, variabel linguistik dapat berupa **beban energi**. Variabel ini dapat memiliki nilai linguistik seperti rendah, sedang, dan tinggi. Jika hasil prediksi LSTM berupa angka 650 kWh, maka sistem fuzzy dapat menafsirkan angka tersebut sebagai "beban tinggi" dengan derajat keanggotaan tertentu. Dengan demikian, variabel linguistik berfungsi sebagai jembatan antara data numerik dan keputusan yang mudah dipahami.

Variabel linguistik biasanya memiliki beberapa komponen utama, yaitu nama variabel, semesta pembicaraan, himpunan fuzzy, dan fungsi keanggotaan. **Nama variabel** menunjukkan aspek yang dianalisis, misalnya konsumsi energi. **Semesta pembicaraan** adalah rentang nilai yang mungkin, misalnya 0 sampai 1000 kWh. **Himpunan fuzzy** adalah kategori

linguistik, misalnya rendah, sedang, dan tinggi. **Fungsi keanggotaan** menentukan derajat keanggotaan setiap nilai terhadap kategori tersebut.

Tabel 6.2: Contoh Variabel Linguistik Pada Prediksi Konsumsi Energi

Komponen	Contoh
Nama variabel	Beban energi
Semesta pembicaraan	0–1000 kWh
Nilai linguistik	Rendah, sedang, tinggi
Input numerik	650 kWh
Output interpretasi	Beban energi tinggi

Variabel linguistik juga dapat digunakan dalam aturan fuzzy. Misalnya, aturan dapat ditulis sebagai berikut: **IF** beban energi tinggi **THEN** status penggunaan perlu dikendalikan. Aturan lain dapat berupa: **IF** beban energi sedang **THEN** status penggunaan perlu dipantau. Aturan seperti ini lebih mudah dipahami dibandingkan aturan yang hanya menggunakan batas angka tegas.

Dalam sistem yang lebih kompleks, variabel linguistik tidak hanya berasal dari satu input. Misalnya, prediksi beban energi dapat dipengaruhi oleh konsumsi listrik, suhu ruangan, waktu penggunaan, dan jumlah penghuni. Setiap variabel dapat memiliki nilai linguistik masing-masing. Contohnya, suhu ruangan dapat bernilai rendah, normal, atau tinggi. Waktu penggunaan dapat bernilai pagi, siang, sore, atau malam. Gabungan variabel tersebut dapat membentuk aturan fuzzy yang lebih kaya.

Penggunaan variabel linguistik membuat sistem fuzzy lebih komunikatif. Pengguna tidak hanya melihat angka hasil prediksi, tetapi juga memperoleh informasi dalam bentuk istilah yang lebih mudah digunakan untuk mengambil keputusan. Dalam konteks model hibrida LSTM–Fuzzy Logic, LSTM menghasilkan angka prediksi, sedangkan variabel linguistik membantu mengubah angka tersebut menjadi kategori keputusan.

6.4.1 Aturan Fuzzy IF–THEN

Aturan fuzzy **IF–THEN** merupakan bagian penting dalam *fuzzy logic* karena digunakan untuk menyusun proses penalaran. Aturan ini menghubungkan kondisi tertentu dengan keluaran atau keputusan tertentu. Bentuk umumnya adalah **IF kondisi terpenuhi THEN keputusan**

dihasilkan. Dalam sistem fuzzy, kondisi dan keputusan biasanya dinyatakan menggunakan variabel linguistik, seperti rendah, sedang, tinggi, aman, waspada, atau kritis.

Berbeda dengan aturan tegas yang hanya bekerja berdasarkan batas angka tertentu, aturan fuzzy dapat menangani kondisi yang tidak sepenuhnya pasti. Misalnya, pada sistem prediksi konsumsi energi, nilai prediksi 650 kWh dapat memiliki derajat keanggotaan pada kategori "sedang" dan "tinggi" sekaligus. Aturan fuzzy kemudian digunakan untuk menentukan bagaimana sistem menafsirkan kondisi tersebut dan menghasilkan keputusan yang lebih mudah dipahami. Pendekatan ini sesuai dengan prinsip dasar fuzzy logic yang dikembangkan untuk menangani ketidakpastian dan penalaran mendekati cara manusia berpikir (Zadeh, 1965; Ross, 2010).

Contoh aturan fuzzy sederhana dalam prediksi konsumsi energi adalah sebagai berikut:

```
IF beban energi rendah THEN status penggunaan aman.
IF beban energi sedang THEN status penggunaan perlu dipantau.
IF beban energi tinggi THEN status penggunaan perlu dikendalikan.
```

Aturan tersebut menunjukkan bahwa hasil prediksi tidak berhenti pada angka, tetapi diterjemahkan menjadi keputusan. Jika model LSTM memprediksi konsumsi energi sebesar 720 kWh, maka sistem fuzzy dapat mengolah nilai tersebut melalui fungsi keanggotaan dan aturan IF–THEN. Apabila nilai tersebut dominan masuk kategori "tinggi", maka sistem dapat memberikan status "perlu dikendalikan".

Aturan fuzzy juga dapat menggunakan lebih dari satu kondisi. Misalnya, prediksi beban energi tidak hanya ditentukan oleh konsumsi listrik, tetapi juga suhu ruangan atau waktu penggunaan. Contoh aturannya adalah:

```
IF beban energi tinggi AND suhu ruangan tinggi THEN risiko beban
puncak tinggi.
```

Pada aturan tersebut, operator **AND** digunakan untuk menggabungkan dua kondisi. Dalam fuzzy logic, operator AND biasanya mengambil nilai minimum dari derajat keanggotaan. Jika beban energi memiliki derajat keanggotaan tinggi sebesar 0,8 dan suhu ruangan memiliki derajat keanggotaan tinggi sebesar 0,6, maka kekuatan aturan dapat dihitung sebagai:

$$\alpha = \min(0{,}8, 0{,}6) = 0{,}6$$

Nilai α\alphaα menunjukkan kekuatan aturan. Semakin besar nilainya, semakin kuat aturan tersebut berpengaruh terhadap keputusan akhir. Selain AND, sistem fuzzy juga dapat menggunakan operator **OR**. Operator OR biasanya mengambil nilai maksimum dari derajat keanggotaan. Misalnya, jika sistem memberi peringatan ketika beban energi tinggi atau suhu ruangan tinggi, maka kondisi dengan derajat keanggotaan terbesar dapat menjadi dasar penalaran.

Aturan fuzzy dapat disusun berdasarkan pengetahuan pakar, hasil pengamatan data, atau kombinasi keduanya. Pada sistem sederhana, aturan dapat dibuat secara manual berdasarkan pemahaman terhadap masalah. Namun, pada sistem yang lebih kompleks, jumlah aturan dapat bertambah banyak, terutama jika variabel input semakin banyak. Oleh karena itu, penyusunan aturan perlu dibuat terstruktur agar sistem tetap mudah dipahami dan tidak saling bertentangan.

Dalam model LSTM–Fuzzy Logic, aturan IF–THEN berperan setelah proses prediksi numerik dilakukan. LSTM bertugas mempelajari pola historis dan menghasilkan nilai prediksi, sedangkan aturan fuzzy bertugas menerjemahkan nilai tersebut menjadi keputusan linguistik. Dengan demikian, sistem tidak hanya menjawab "berapa nilai prediksinya", tetapi juga "apa makna dari nilai tersebut".

Tabel 6.3: Contoh Aturan Fuzzy Pada Prediksi Konsumsi Energi

No	Aturan Fuzzy	Makna Keputusan
1	IF beban energi rendah THEN status aman	Konsumsi masih dalambatas wajar
2	IF beban energi sedang THEN status perlu dipantau	Konsumsi mulai membutuhkan perhatian
3	IF beban energi tinggi THEN status perlu dikendalikan	Diperlukan tindakan pengurangan beban
4	IF beban energi tinggi AND suhu tinggi THEN risiko puncak tinggi	Sistem perlu memberi peringatan

Bab 7
Model Hibrida Deep Learning dan Fuzzy Logic

Bab 8
Perancangan Model Prediksi Time Series Berbasis LSTM–Fuzzy Logic

Bab 9
Evaluasi Model Prediksi Time Series

Bab 10
Studi Kasus Prediksi Konsumsi Energi

Bab 11
Implementasi Prediksi Time Series Menggunakan Python

Bab 12
Arah Pengembangan dan Tantangan Penelitian

Bioadata Penulis

Muhammad Fairuzabadi, S.Si., M.Kom., lahir di Bulukumba, Sulawesi Selatan, pada 26 september 1974. meraih gelar Sarjana Ilmu Komputer dari Fakultas MIPA Universitas Kristen Immanuel (UKRIM) Yogyakarta pada tahun 1998 dan menyelesaikan Program Magister Ilmu Komputer di Universitas Gadjah Mada pada tahun 2006. saat ini, bekerja sebagai dosen pada Program Sarjana Informatika Fakultas Sains dan Teknologi Universitas PGRI Yogyakarta (UPY). telah menulis 25 buku dengan topik sistem informasi, artificial intelligence (AI), data science, dan deep learning. aktif dalam penelitian dan pengabdian kepada masyarakat, dengan fokus pada pengembangan sistem informasi, sistem pakar, dan data science.

email: fairuz@upy.ac.id

Puji Handayani Putri, S.T., M.Kom., lahir di Kulon Progo, Daerah Istimewa Yogyakarta, pada 22 Februari 1990. Meraih gelar Sarjana Teknik Informatika dari Fakultas Teknologi Industri Universitas Ahmad Dahlan (UAD) Yogyakarta pada tahun 2012 dan menyelesaikan program Magister Teknik Informatika di Amikom Yogyakarta pada tahun 2015. Saat ini, bekerja sebagai dosen pada Program Sarjana Informatika Fakultas Sains dan Teknologi Universitas PGRI Yogyakarta (UPY). Selain berperan sebagai pengajar, aktif dalam penelitian dan pengabdian kepada masyarakat, dengan fokus pada Artificial Intelligence. Berbagai publikasi telah dimuat dalam jurnal nasional dan internasional.

Email: pujihp@upy.ac.id

Gema Kharismajati, S.Kom., M.Kom., lahir di Purbalingga, Jawa Tengah, pada 14 Januari 1996. Meraih gelar Sarjana Teknik Informatika dari Fakultas Teknik Universitas PGRI Yogyakarta pada tahun 2018, kemudian melanjutkan pendidikan dan menyelesaikan Program Magister Informatika pada Fakultas Teknologi Industri Universitas Ahmad Dahlan pada tahun 2022. Saat ini berkarier sebagai dosen pada Program Sarjana Sistem Informasi, Fakultas Sains dan Teknologi, Universitas PGRI Yogyakarta (UPY). Dalam aktivitas akademik, aktif melaksanakan kegiatan pendidikan, penelitian, pengabdian kepada masyarakat, serta pengembangan kurikulum dan inovasi pembelajaran berbasis teknologi. Bidang keahlian dan minat yang ditekuni meliputi Digital Technology & Innovation, Intelligent System & Artificial Intelligence, Data Analytics, serta pengembangan Sistem Informasi dan transformasi digital. Selain kegiatan pengajaran, juga terlibat dalam berbagai pengembangan sistem dan implementasi teknologi untuk mendukung peningkatan kualitas pendidikan dan tata kelola berbasis digital. Berbagai karya ilmiah dan aktivitas akademik telah dipublikasikan melalui seminar, jurnal ilmiah, maupun kegiatan pengembangan di bidang teknologi informasi.

Email: gemakharismajati@upy.ac.id

Daftar Pustaka

Ahmad, T., & Chen, H. (2018). Short and medium-term forecasting of cooling and heating load demand in building environment with data-mining based approaches. *Energy and Buildings, 166*, 460–476.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks, 5*(2), 157–166.

Bishop, C. M. (2006). *Pattern recognition and machine learning*. Springer.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time series analysis: Forecasting and control* (5th ed.). Wiley.

Brownlee, J. (2018). *Deep learning for time series forecasting*. Machine Learning Mastery.

Chai, T., & Draxler, R. R. (2014). Root mean square error or mean absolute error? Arguments against avoiding RMSE in the literature. *Geoscientific Model Development, 7*(3), 1247–1250.

Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys, 41*(3), 1–58.

Chatfield, C. (2003). *The analysis of time series: An introduction* (6th ed.). Chapman & Hall/CRC.

Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1724–1734).

Chollet, F. (2021). *Deep learning with Python* (2nd ed.). Manning Publications.

de Myttenaere, A., Golden, B., Le Grand, B., & Rossi, F. (2016). Mean absolute percentage error for regression models. *Neurocomputing, 192*, 38–48.

Elman, J. L. (1990). Findingstructure in time. *Cognitive Science, 14*(2), 179–

211.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with LSTM. *Neural Computation, 12*(10), 2451–2471.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Haykin, S. (2009). *Neural networks and learning machines* (3rd ed.). Pearson.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and practice* (3rd ed.). OTexts.

Jang, J.-S. R., Sun, C.-T., & Mizutani, E. (1997). *Neuro-fuzzy and soft computing: A computational approach to learning and machine intelligence*. Prentice Hall.

Kelly, J., & Knottenbelt, W. (2015). The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. *Scientific Data, 2*, 150007.

Kingma, D. P.,& Ba, J. (2015). Adam: Amethod for stochasticoptimization. *International Conference on Learning Representations*.

Klir, G. J., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic: Theory and applications*. Prentice Hall.

Kong, W., Dong, Z. Y., Jia, Y., Hill, D. J., Xu, Y., & Zhang, Y. (2019). Short-term residential load forecasting based on LSTM recurrent neural network. *IEEE Transactions on Smart Grid, 10*(1), 841–851.

Kotu, V., & Deshpande, B. (2019). *Data science: Concepts and practice* (2nd ed.). Morgan Kaufmann.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Lim, B., & Zohren, S. (2021). Time-series forecasting with deep learning: A survey. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 379*(2194),

20200209.

Little, R. J. A., & Rubin, D. B. (2019). *Statistical analysis with missing data* (3rd ed.). Wiley.

Moritz, S., & Bartz-Beielstein, T. (2017). imputeTS: Time series missing value imputation in R. *The R Journal, 9*(1), 207–218.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning* (pp. 807–814).

Nauck, D., Klawonn, F., & Kruse, R. (1997). *Foundations of neuro-fuzzy systems*. Wiley.

Nielsen, M. A. (2015). *Neural networks and deep learning*. Determination Press.

Olah, C. (2015). Understanding LSTM networks. *Colah's Blog*.

Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning* (pp. 1310–1318).

Prechelt, L. (1998). Early stopping—But when? In G. B. Orr & K.-R. Müller (Eds.), *Neural networks: Tricks of the trade* (pp. 55–69). Springer.

Ross, T. J. (2010). *Fuzzy logic with engineering applications* (3rd ed.). Wiley.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature, 323*(6088), 533–536.

Schafer, J. L., & Graham, J. W. (2002). Missing data: Our view of the state of the art. *Psychological Methods, 7*(2), 147–177.

Shumway, R. H., & Stoffer, D. S. (2017). *Time series analysis and its applications: With R examples* (4th ed.). Springer.

Sola, J., & Sevilla, J. (1997). Importance of input data normalization for the application of neural networks to complex industrial problems. *IEEE*

Transactions on Nuclear Science, 44(3), 1464–1468.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research, 15*, 1929–1958.

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error over the root mean square error in assessing average model performance. *Climate Research, 30*(1), 79–82.

Zadeh, L. A. (1965). Fuzzy sets. *Information and Control, 8*(3), 338–353.

Zadeh, L. A. (1975). The concept of a linguistic variable and its application to approximate reasoning—I. *Information Sciences, 8*(3), 199–249.