Jurnal Ilmu Komputer Agri-Informatika

Tersedia secara daring di:
https://jurnal.ipb.ac.id/index.php/jika

Volume 13 Nomor 1 halaman 56–72
eISSN: 2654-9735, pISSN: 2089-6026

Modifikasi Optimizer ADAM untuk Meningkatkan Kinerja Algoritma LSTM dalam Prediksi Penjualan Sepatu Merek Prodigo

A Modified ADAM Optimizer for Enhancing LSTM Performance in Predicting Prodigo Shoe Sales

YANDI FITRIYANTO[1], KARLISA PRIANDANA[2]*, TOTO HARYANTO[3]

Abstrak

Penelitian ini bertujuan mengevaluasi penggunaan ADAM modifikasi pada model *Long Short-Term Memory* (LSTM) untuk prediksi penjualan sepatu UMKM Prodigo. Permasalahan utama penelitian meliputi ketidakstabilan tren penjualan dan keterbatasan data latih. Modifikasi ADAM dilakukan pada komponen gradien untuk mempertahankan arah pembaruan parameter sehingga diharapkan meningkatkan stabilitas proses optimasi. Data yang digunakan berupa data penjualan mingguan periode Januari 2018 hingga Juni 2025 sebanyak 387 data. Selain itu, dilakukan augmentasi data menggunakan *Time-Series Generative Adversarial Networks* (TimeGAN) untuk menghasilkan data sintetis. Tahapan penelitian meliputi normalisasi Min-Max, pembentukan *sequence* menggunakan metode *sliding window*, serta optimasi *hyperparameter* menggunakan *grid search*. Evaluasi dilakukan melalui 20 kali pelatihan berulang menggunakan metrik *Mean Absolute Error* (MAE) dan *Dynamic Time Warping* (DTW). Hasil penelitian menunjukkan bahwa ADAM modifikasi menghasilkan performa yang relatif sedikit lebih baik dibandingkan ADAM standar. Pada skenario tanpa data sintetis, ADAM modifikasi menghasilkan MAE sebesar 272 ± 6, sedangkan ADAM standar sebesar 278 ± 10. Sementara itu, pada skenario dengan data sintetis, ADAM modifikasi menghasilkan MAE sebesar 270 ± 4 dibandingkan ADAM standar sebesar 272 ± 13. Penggunaan data sintetis juga membantu model dalam mempertahankan pola temporal data, ditunjukkan oleh nilai DTW sebesar 3.5. Selain itu, model telah diimplementasikan dalam bentuk *web service* berbasis Flask.

Kata Kunci**:** Augmentasi data, Long Short-Term Memory (LSTM), Modifikasi ADAM, TimeGAN, UMKM

Abstract

This study aims to evaluate the use of a modified ADAM optimizer in a Long Short-Term Memory *(LSTM) model for predicting sales of Prodigo footwear products. The main challenges addressed in this study are unstable sales trends and limited training data. The ADAM modification was applied to the gradient component to preserve the parameter update direction, thereby improving optimization stability. The dataset consisted of weekly sales data from January 2018 to June 2025, totaling 387 records. In addition, data augmentation was performed using* Time-Series Generative Adversarial Networks *(TimeGAN) to generate synthetic data. The research stages included Min-Max normalization, sequence generation using the sliding window method, and hyperparameter optimization through* grid search*. Model evaluation was conducted through 20 independent experiments using* Mean Absolute Error *(MAE) and* Dynamic Time Warping *(DTW). The results indicate that the modified ADAM achieved comparable performance to the standard ADAM. Without synthetic data, the modified ADAM produced an MAE of 272 ± 6, while the standard ADAM achieved 278 ± 10. With synthetic data, the modified ADAM achieved an MAE of 270 ± 4 compared to 272 ± 13 for the standard ADAM. These results suggest a slight tendency toward lower MAE values with the modified ADAM, although the performance ranges overlap. The use of synthetic data also improved the model's ability to preserve temporal patterns, indicated by a DTW value of 3.5. Furthermore, the model was implemented as a Flask-based web service.*

*Keywords**: Data augmentation, Long Short-Term Memory (LSTM), Modified ADAM, TimeGAN, SMEs*

[1,2,3]Program Studi Ilmu Komputer, Sekolah Sains Data, Matematika, dan Informatika,Institut Pertanian Bogor, Bogor 16680;
*Penulis Korespondensi: Tel/Faks: 0251-8625584, Surel: karlisa@apps.ipb.ac.id

PENDAHULUAN

Trend penjualan yang tidak stabil mendorong perusahaan untuk melakukan prediksi penjualan guna menentukan jumlah produksi secara tepat (Pacella dan Papadia 2021). Salah satu pendekatan yang banyak digunakan untuk memprediksi permintaan konsumen adalah algoritma *deep learning*, khususnya Long Short-Term Memory (LSTM), yang mampu menangkap pola temporal pada data deret waktu. Dalam proses pelatihannya, LSTM umumnya dioptimasi menggunakan algoritma berbasis gradien seperti Gradient Descent, ADAM, dan ADAMax yang bekerja dengan memperbarui parameter model secara iteratif (Yi *et al.* 2020). Metode optimasi memiliki pengaruh yang sangat besar dalam meningkatkan performa model *machine learning*, termasuk pada jaringan saraf dalam (*deep neural network*) (Tekkali dan Natarajan 2024).

ADAM (*Adaptive Moment Estimation*) merupakan salah satu *optimizer* yang banyak digunakan karena memiliki efisiensi komputasi yang tinggi, kebutuhan memori yang relatif kecil, serta kemampuan untuk menyesuaikan laju pembelajaran secara adaptif berdasarkan estimasi momen pertama dan kedua dari gradien (Kingma dan Ba 2015). ADAM juga telah terbukti efektif dalam mengoptimasi LSTM dengan meminimalkan fungsi kerugian, seperti *mean squared error*, untuk memperoleh bobot optimal (Chang *et al.* 2018). Namun demikian, beberapa penelitian menunjukkan bahwa ADAM memiliki keterbatasan, seperti ketidakstabilan akibat gradien stokastik pada periode historis serta variasi dampak pembaruan parameter terhadap fungsi jaringan (Chandriah dan Naraganahalli 2021). Selain itu, ADAM juga berpotensi gagal mencapai konvergensi menuju solusi optimal (Reddi *et al.* 2019).

Di sisi lain, performa model *deep learning* dalam prediksi deret waktu sangat dipengaruhi oleh jumlah dan kualitas data pelatihan yang tersedia (Huang dan Luo 2023). Keterbatasan data sering menjadi kendala, sehingga diperlukan teknik augmentasi data, salah satunya menggunakan *Time-Series Generative Adversarial Networks* (TimeGAN) yang mampu menghasilkan data sintetis dengan karakteristik temporal yang serupa dengan data asli (Yoon *et al.* 2019). Pendekatan ini juga telah digunakan dalam analisis risiko sistemik dan ketahanan pasar energi Eropa untuk memodelkan dinamika pasar secara real-time (Bohórquez Correa *et al.* 2026).

Berdasarkan permasalahan tersebut, penelitian ini mengusulkan modifikasi ADAM *optimizer* untuk meningkatkan kinerja LSTM dalam memprediksi penjualan produk sepatu UMKM Prodigo. Modifikasi dilakukan dengan mempertahankan arah gradien untuk setiap bobot agar pembaruan parameter menjadi lebih stabil dan laju pembelajaran dapat dikendalikan dengan lebih baik (Chandriah dan Naraganahalli 2021). Data yang digunakan berupa data penjualan mingguan UMKM Prodigo dari Januari 2018 hingga minggu pertama Juni 2025 sebanyak 387 data. Selain itu, penelitian ini juga membandingkan kinerja model LSTM menggunakan data asli tanpa augmentasi dan data yang telah ditambahkan data sintetis hasil TimeGAN. Tujuan penelitian ini adalah untuk meningkatkan akurasi prediksi penjualan sepatu merek Prodigo dengan mengoptimalkan algoritma LSTM melalui modifikasi ADAM serta mengevaluasi pengaruh penggunaan data sintetis terhadap kinerja model.

METODE

Pengambilan data

Data yang digunakan dalam penelitian ini merupakan data sekunder yang diperoleh dari database transaksi penjualan UMKM Prodigo. Data yang digunakan berupa data penjualan produk sepatu yang telah terakumulasi secara mingguan, dengan periode pengamatan dimulai dari Januari 2018 hingga minggu pertama Juni 2025, dengan total sebanyak 387 data.

Proses pengumpulan data dilakukan dengan mengekstraksi data transaksi harian dari sistem aplikasi Prodigo, yang kemudian dikonversi menjadi data mingguan. Dalam proses ini, data penjualan harian yang tercatat dari hari Senin hingga Minggu dijumlahkan untuk merepresentasikan total penjualan dalam satu minggu. Transformasi ini dilakukan untuk menyesuaikan kebutuhan pemodelan deret waktu yang digunakan dalam penelitian. Sebagai

contoh, data transaksi harian pada minggu pertama Januari 2018 setelah dikonversi menghasilkan total penjualan mingguan sebesar 149 unit. Contoh data transaksi harian yang diperoleh dari sistem ditunjukkan pada Tabel 1.

Tabel 1 Informasi Transaksi Harian

Tanggal Transaksi	Barang Terjual
01-01-2018	13
02-01-2018	29
03-01-2018	25
04-01-2018	33
05-01-2018	15
06-01-2018	25
07-01-2018	6

Praproses Data

Tahap praproses data dalam penelitian ini dilakukan melalui proses normalisasi untuk menyesuaikan skala data sebelum digunakan pada model. Teknik normalisasi yang digunakan adalah normalisasi Min-Max, yaitu metode penskalaan data yang mentransformasikan nilai data ke dalam rentang tertentu, umumnya antara 0 dan 1 (Ali 2022). Proses ini bertujuan untuk mengurangi perbedaan skala antar data sehingga dapat meningkatkan stabilitas dan kinerja model dalam proses pelatihan.

Penggunaan normalisasi Min-Max dalam penelitian ini dipilih karena telah terbukti mampu meningkatkan akurasi prediksi pada model Long Short-Term Memory (LSTM) dibandingkan dengan teknik normalisasi lainnya, seperti z-score (Pranolo *et al.* 2024). Dengan demikian, data yang telah dinormalisasi diharapkan dapat membantu model LSTM dalam menangkap pola deret waktu secara lebih optimal.

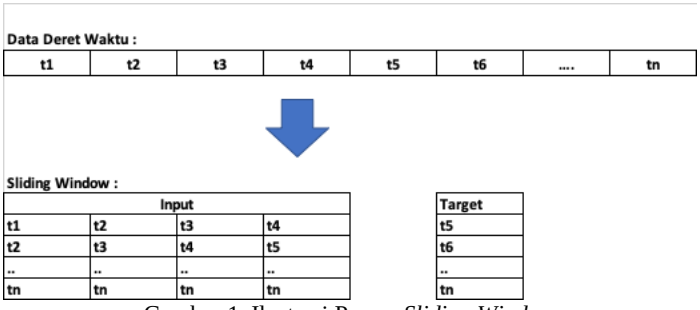
Sliding Window

Model Long Short-Term Memory (LSTM) memerlukan data input dalam bentuk urutan waktu yang direpresentasikan dalam tiga dimensi, yaitu *samples*, *timesteps*, dan *features* (Hochreiter dan Schmidhuber 1997). Sementara itu, data transaksi penjualan yang digunakan dalam penelitian ini masih berupa satu urutan nilai deret waktu satu dimensi. Oleh karena itu, diperlukan proses transformasi data agar sesuai dengan struktur input yang dibutuhkan oleh model LSTM.

Transformasi data dilakukan menggunakan metode *sliding window*, yaitu teknik untuk membentuk sequence dari data historis sehingga setiap sampel input memiliki dimensi *timesteps* dan *features* yang sesuai dengan arsitektur LSTM (Brownlee 2018). Pada penelitian ini, digunakan 4 timestep sebagai data input untuk memprediksi 1 timestep berikutnya sebagai target prediksi. Sebagai ilustrasi, data dengan urutan t_1,t_2, t_3,t_4 digunakan untuk memprediksi target t_5, kemudian t_2, t_3,t_4,t_5 digunakan untuk memprediksi t_6 dan seterusnya. Ilustrasi pembentukan sequence menggunakan metode *sliding window* ditunjukkan pada Tabel 2 dan Gambar 1.

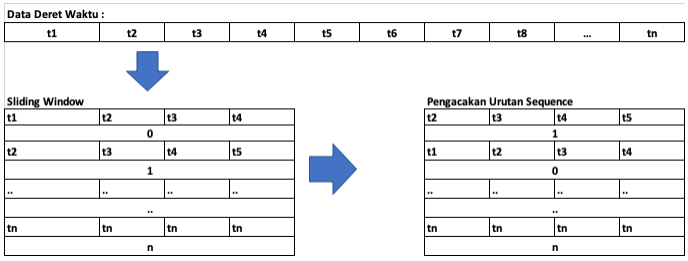
Tabel 2 Ilustrasi Pembentukan *Sequence* Menggunakan *Sliding Window*

Input	*Output*
t_1,t_2, t_3,t_4	t_5
t_2, t_3,t_4,t_5	t_6
t_3,t_4,t_5,t_6	t_7
t_4,t_5,t_6,t_{67}	t_7



Gambar 1 Ilustrasi Proses *Sliding Window*

Pada tahap pembuatan model *Time-Series Generative Adversarial Networks* (TimeGAN), data latih juga diproses menggunakan metode *sliding window* (Yoon *et al.* 2019). Namun, terdapat perbedaan dalam penerapannya dibandingkan dengan proses pada LSTM. Pada model LSTM, *sliding window* menghasilkan pasangan *input* dan *target*, sedangkan pada TimeGAN, proses ini hanya digunakan untuk membentuk sequence tanpa target. Selain itu, urutan *sequence* yang dihasilkan kemudian diacak agar setiap data bersifat independen dan berasal dari distribusi probabilitas yang sama (Yoon *et al.* 2019). Ilustrasi proses sliding window pada TimeGAN ditunjukkan pada Gambar 2.



Gambar 2 Ilustrasi Proses *Sliding Window* untuk Model TimeGAN

Pembagian Data

Pada penelitian ini, data dibagi menjadi dua bagian utama, yaitu data latih dan data uji. Data latih digunakan untuk membangun model serta menyesuaikan parameter selama proses pelatihan, sedangkan data uji digunakan untuk mengevaluasi kinerja model terhadap data yang belum pernah dilihat sebelumnya (Goodfellow *et al.* 2016).

Dari total 387 data yang tersedia, pembagian data dilakukan dengan rasio 80:20, di mana 80% data digunakan sebagai data latih dan 20% sebagai data uji. Pendekatan ini merupakan praktik umum dalam pemodelan *machine learning* untuk menjaga keseimbangan antara proses pelatihan dan evaluasi model (Hastie *et al.* 2009). Setelah dilakukan transformasi data menggunakan metode *sliding window*, jumlah data latih dan data uji yang dihasilkan ditunjukkan pada Tabel 3.

Tabel 3 Jumlah Data Latih dan Data Uji

Jenis Data	Tanpa Data Sintetis	Dengan Data Sintetis
Data Latih Asli	305	305
Data Sintetis	-	768
Total Data Latih	305	1073
Data Uji	74	74

Pada skenario penggunaan data sintetis, data latih dilakukan augmentasi menggunakan model *Time-Series Generative Adversarial Networks* (TimeGAN) untuk menghasilkan tambahan data sintetis sebanyak 768 sampel. Pendekatan augmentasi data ini bertujuan untuk meningkatkan jumlah dan keragaman data latih sehingga dapat membantu meningkatkan kemampuan generalisasi model (Shorten dan Khoshgoftaar 2019).

Penelitian ini belum menerapkan metode validasi khusus deret waktu, seperti *walk-forward validation*, karena proses pelatihan dilakukan secara berulang pada model LSTM dan

TimeGAN sehingga memerlukan waktu lebih banyak lagi. Selain itu, penerapan *walk-forward validation* pada skenario augmentasi data sintetis memerlukan proses pembangkitan data sintetis pada setiap *fold* validasi, sehingga meningkatkan kompleksitas proses pelatihan dan evaluasi model. Untuk mengurangi pengaruh variasi proses pelatihan akibat penggunaan satu kali pembagian data, penelitian ini melakukan 20 kali pelatihan berulang dan melaporkan nilai rata-rata serta standar deviasi performa model. Meskipun demikian, penggunaan *walk-forward validation* dinilai berpotensi memberikan evaluasi yang lebih robust pada data deret waktu dan dapat menjadi pengembangan pada penelitian selanjutnya.

Modifikasi ADAM

Algoritma ADAM (*Adaptive Moment Estimation*) merupakan metode optimasi berbasis gradien yang menghitung estimasi momen pertama (*first moment estimate*) dan momen kedua (*second moment estimate*) untuk memperbarui parameter model secara adaptif (Kingma dan Ba 2015). Proses optimasi dimulai dengan menghitung gradien fungsi objektif terhadap parameter model θ pada iterasi ke-t, sebagaimana ditunjukkan pada Persamaan (1):

$$g_t = \nabla_\theta f_t(\theta_{t-1}), \tag{1}$$

dengan g_t merupakan gradien yang diperoleh dengan menghitung turunan parsial fungsi objektif f_t terhadap parameter model θ. Selanjutnya, estimasi momen pertama m_t dan momen kedua v_t dihitung menggunakan Persamaan (2) dan (3):

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t, \tag{2}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2, \tag{3}$$

dengan m_t merupakan estimasi momen pertama, v_t merupakan estimasi momen kedua, serta β_1 dan β_2 adalah koefisien peluruhan eksponensial. Karena inisialisasi awal menyebabkan bias terhadap nol, maka dilakukan koreksi bias terhadap kedua estimasi tersebut menggunakan Persamaan (4) dan (5):

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \tag{4}$$

$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \tag{5}$$

dengan $\hat{m}_t$ merupakan momen pertama yang sudah terkoreksi biasnya dan $\hat{v}_t$ momen kedua yang sudah terkoreksi biasnya. Dengan menggunakan koreksi bias momen pertama dan momen kedua seperti persamaan (4) dan (5), parameter model diperbarui menggunakan Persamaan (6):

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{6}$$

dengan α sebagai *learning rate*, ϵ sebagai konstanta kecil untuk menghindari pembagian dengan nol, dan θ_t sebagai parameter yang diperbarui.

Meskipun ADAM standar memiliki keunggulan dalam efisiensi komputasi dan kemampuan adaptasi terhadap perubahan gradien, algoritma ini masih memiliki keterbatasan, seperti ketidakstabilan akibat gradien stokastik historis (Chandriah dan Naraganahalli 2021). Untuk mengatasi permasalahan tersebut, penelitian ini mengadopsi modifikasi yang diperkenalkan oleh Chandriah dan Naraganahalli (2021), yaitu dengan mempertimbangkan arah gradien pada iterasi sebelumnya dalam proses pembaruan parameter sehingga arah optimasi menjadi lebih stabil. Modifikasi tersebut dilakukan pada gradien g_t sebelum digunakan dalam perhitungan estimasi momen pertama dan kedua, sebagaimana ditunjukkan pada Persamaan (7):

$$\hat{g}_t = g_t - (g_t \bullet \theta_{t-1})\theta_{t-1}, \tag{7}$$

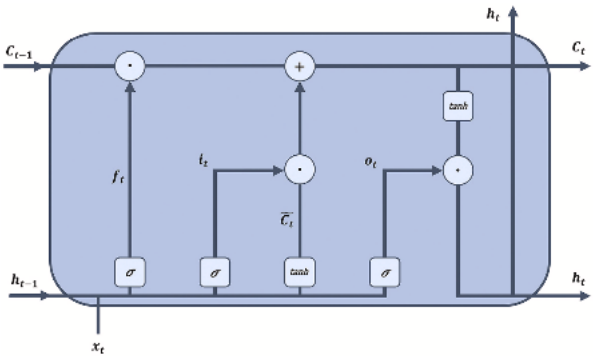
dengan $\hat{g}_t$ merupakan gradien hasil modifikasi, g_t merupakan nilai gradien asli, θ_{t-1} merupakan parameter model sebelumnya dan simbol titik (*dot product*) merupakan perkalian dua vektor antara gradien dengan parameter model sebelumnya. Gradien hasil modifikasi $\hat{g}_t$ diharapkan mampu meningkatkan kestabilan pembaruan bobot dan modifikasi gradien ini

hanya dilakukan pada *layer* LSTM. Selanjutnya, nilai gradien hasil modifikasi digunakan dalam perhitungan estimasi momen pertama, estimasi momen kedua, koreksi bias, dan pembaruan parameter dengan tetap mengikuti Persamaan (2) hingga Persamaan (6).

Long Short-Term Memory

a. Arsitektur Long Short-Term Memory

Secara arsitektur, pada waktu ke-t, sel LSTM terdiri dari beberapa komponen utama, yaitu *forget gate* (f_t), *input gate* (i_t), *memory cell state* (C_t), dan *output gate* (o_t). Mekanisme dalam sel ini menggunakan fungsi aktivasi *sigmoid* (σ) serta fungsi *hyperbolic tangent* ($tanh$) untuk mengatur aliran informasi (Jörges *et al.* 2021). Arsitektur LSTM dapat dilihat pada Gambar 3.

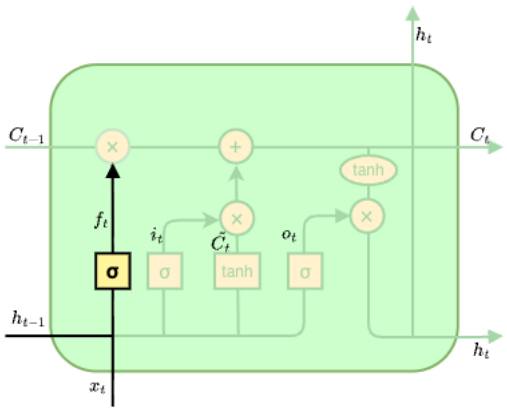


Gambar 3 Arsitektur LSTM (Jörges *et al.* 2021)

Forget gate berfungsi untuk menentukan seberapa banyak informasi masa lalu dari *cell state* sebelumnya (C_{t-1}) yang ingin dibuang atau dipertahankan, seperti pada Gambar 4. Persamaan *forget gate* adalah sebagai berikut:

$$f_t = \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big), \tag{8}$$

dengan f_t merupakan *forget gate*, σ fungsi sigmoid, W_f merupakan nilai *weight* untuk *forget gate*, h_{t-1} nilai *output* sebelum orde ke-t, x_t merupakan nilai *input* pada orde ke-t, dan b_f merupakan nilai bias pada *forget gate*.

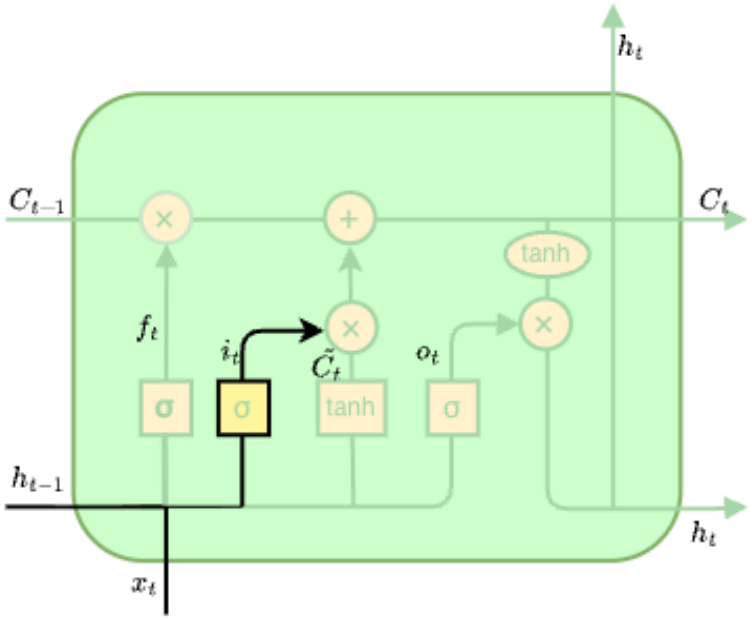


Gambar 4 Arsitektur LSTM *Forget Gate*

Input gate berfungsi untuk mengatur seberapa besar informasi baru yang masuk akan disimpan ke dalam *cell state*, seperti ditunjukkan pada Gambar 5. Persamaan *input gate* adalah sebagai berikut:

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i), \tag{9}$$

dengan i_t merupakan *input gate*, σ fungsi sigmoid, W_i merupakan nilai *weight* untuk *input gate*, h_{t-1} nilai *output* sebelum orde ke-t, x_t merupakan nilai input pada orde ke-t, dan b_i merupakan nilai bias pada *input gate*.

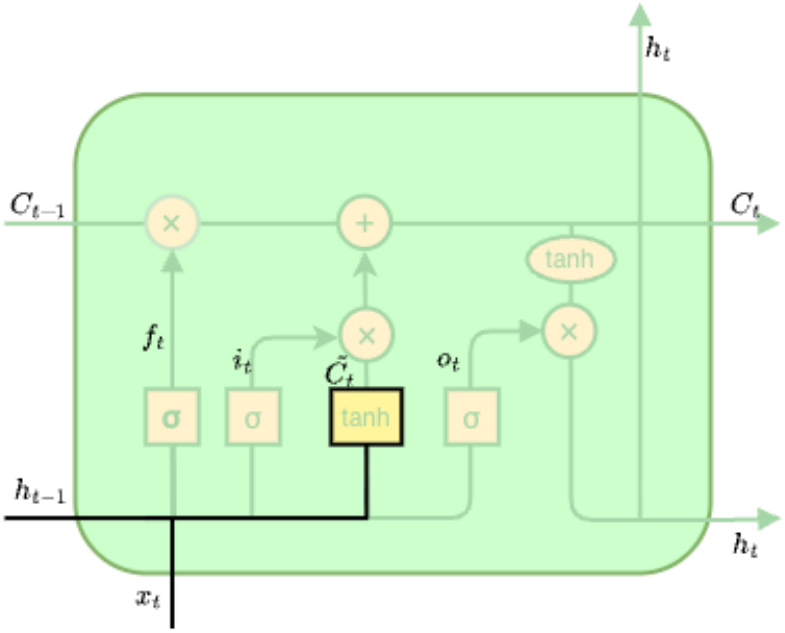


Gambar 5 Arsitektur LSTM *Input Gate*

Kandidat *cell state* berfungsi untuk membuat vektor nilai kandidat baru menggunakan fungsi aktivasi $tanh$ yang berpotensi ditambahkan ke dalam sel, yang ditunjukkan pada Gambar 6. Persamaan kandidat *cell state* adalah sebagai berikut:

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c), \tag{10}$$

dengan $\tilde{C}_t$ merupakan nilai baru yang dapat ditambahkan ke *cell state*, $tanh$ merupakan fungsi tanh, W_c merupakan nilai *weight* untuk *cell state*, h_{t-1} nilai *output* sebelum orde ke$-t$, x_t merupakan nilai *input* pada orde ke-t, dan b_c merupakan nilai bias pada *cell state*.

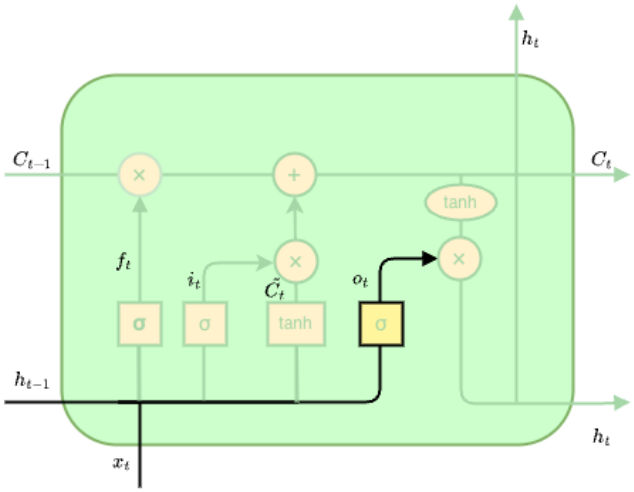


Gambar 6 Arsitektur LSTM Kandidat *Cell State*

Output gate berfungsi untuk menentukan bagian mana dari *cell state* yang akan dikeluarkan sebagai output, seperti yang ditunjukkan pada Gambar 7. Persamaan *output gate* adalah sebagai berikut:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o), \tag{11}$$

dengan o_t merupakan *output gate*, σ fungsi sigmoid, W_o merupakan nilai *weight* untuk *output gate*, h_{t-1} nilai *output* sebelum orde ke t, x_t merupakan nilai input pada orde ke t, dan b_o merupakan nilai bias pada *output gate*. Pada penelitian ini untuk algoritma LSTM menggunakan *library* TensorFlow. Pada *library* TensorFlow W_f, W_i, W_c, W_o masuk ke dalam komponen *lstm/kernel*, h_{t-1} masuk ke dalam komponen *lstm/recurrent_kernel*, dan b_f, b_i, b_c, b_o masuk ke dalam komponen *lstm/bias*.



Gambar 7 Arsitektur LSTM *Output Gate*

b. Pembuatan Model Prediksi

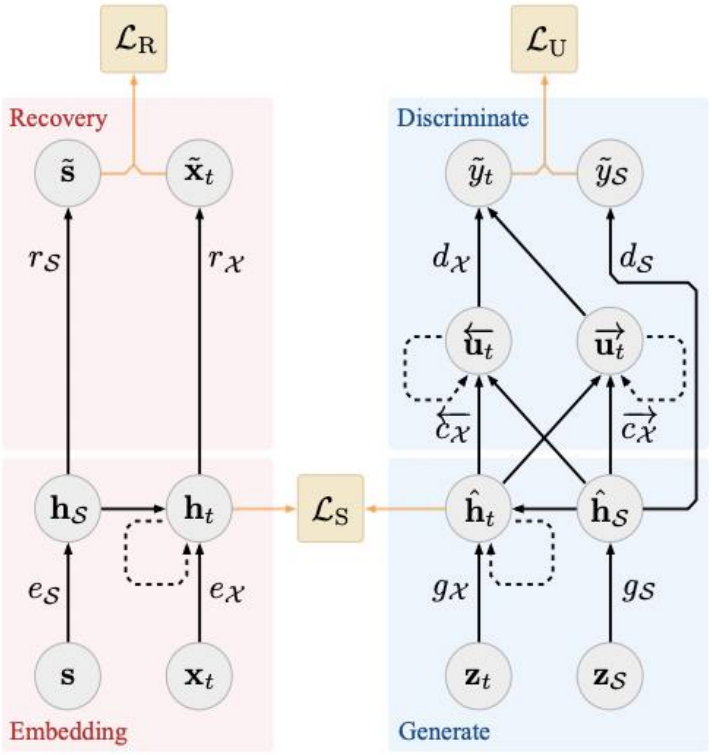
Data yang telah melalui tahap praproses, yaitu normalisasi menggunakan metode Min-Max, selanjutnya ditransformasikan ke dalam bentuk *sequence* dengan dimensi *timesteps* dan *features* menggunakan metode *sliding window*. Setelah itu, data dibagi menjadi data latih dan data uji, sehingga siap digunakan dalam proses pemodelan. Model yang digunakan dalam penelitian ini adalah Long Short-Term Memory (LSTM), yang dioptimasi menggunakan dua pendekatan, yaitu ADAM *non-modified* dan ADAM *modified*. Penggunaan dua pendekatan ini bertujuan untuk membandingkan kinerja model dalam memprediksi penjualan. Kinerja model prediksi sangat dipengaruhi oleh konfigurasi *hyperparameter* yang digunakan (Ilemobayo *et al.* 2024). Oleh karena itu, dalam penelitian ini dilakukan proses optimasi *hyperparameter* menggunakan metode *grid search*, yaitu metode pencarian sistematis dengan menguji seluruh kombinasi nilai *hyperparameter* dalam rentang tertentu untuk mendapatkan konfigurasi terbaik (Bergstra dan Bengio 2012). Adapun *hyperparameter* yang digunakan pada optimizer ADAM dalam penelitian ini meliputi *learning rate*, beta 1, beta 2, epsilon, dan *batch size*. Nilai *hyperparameter* yang diuji dapat dilihat pada Tabel 4. Melalui proses ini, model LSTM diharapkan dapat memperoleh konfigurasi optimal yang mampu meningkatkan akurasi prediksi penjualan.

Tabel 4 *Hyperparameter* ADAM

Hyperparameter	Nilai	Referensi
Learning Rate	0.001	(Kingma dan Ba 2015)
Beta 1	0.9	(Kingma dan Ba 2015)
Beta 2	0.99; 0.999; 0.9999	(Kingma dan Ba 2015; Chandriah dan Naraganahalli 2021)
Epsilon	1e-8	(Kingma dan Ba 2015)
Batch Size	16; 32	

Pembuatan Data Sintetis Menggunakan TimeGAN

Model *Time-Series Generative Adversarial Networks* (TimeGAN) merupakan pengembangan dari *Generative Adversarial Network* (GAN) yang dirancang khusus untuk menghasilkan data deret waktu dengan tetap mempertahankan karakteristik temporal dan distribusi statistik data asli melalui kombinasi *adversarial learning* dan *supervised learning* (Yoon et al., 2019). Arsitektur TimeGAN terdiri dari empat komponen utama, yaitu *Embedder*, *Recovery*, *Generator*, dan *Discriminator*, seperti yang ditunjukkan pada Gambar 8.



Gambar 8 Arsitektur TimeGAN (Yoon *et al.* 2019)

Proses pembuatan data sintetis menggunakan TimeGAN dilakukan melalui tiga tahap utama. Tahap pertama adalah *pre-training autoencoder*, yang melibatkan komponen *Embedder* dan *Recovery* untuk mempelajari representasi laten dari data sehingga mampu menjaga dependensi temporal. Tahap kedua adalah pelatihan *adversarial*, di mana *Generator* (dibantu oleh *Supervisor*) menghasilkan data sintetis di ruang laten, sementara *Discriminator* bertugas membedakan antara data laten asli dan sintetis. Tahap ketiga adalah *joint training*, di mana seluruh komponen dilatih secara bersamaan untuk memastikan bahwa data sintetis yang dihasilkan memiliki distribusi statistik dan pola temporal yang mendekati data asli (Yoon *et al.* 2019).

Dalam penelitian ini, performa model TimeGAN dipengaruhi oleh konfigurasi *hyperparameter* yang digunakan. Oleh karena itu, dilakukan pengujian beberapa kombinasi nilai *hyperparameter* untuk memperoleh hasil yang optimal. Adapun *hyperparameter* yang digunakan meliputi *learning rate*, *noise dimension*, dan *layer dimension*, sebagaimana ditunjukkan pada Tabel 5. Melalui proses ini, data sintetis yang dihasilkan diharapkan mampu memperkaya variasi data latih serta meningkatkan kemampuan model dalam menangkap pola temporal pada data deret waktu.

Tabel 5 *Hyperparameter* TimeGAN

Hyperparameter	Nilai
Learning Rate	0.001;0.0005
Noise Dimension	16;32
Layer Dimension	64;128

Evaluasi Model

Evaluasi model merupakan tahapan penting untuk mengukur tingkat ketepatan dan keakuratan kinerja model yang dihasilkan. Pada penelitian ini, evaluasi dilakukan terhadap dua aspek utama, yaitu kualitas data sintetis yang dihasilkan serta kinerja model prediksi.

Evaluasi data sintetis yang dihasilkan oleh model *Time-Series Generative Adversarial Networks* (TimeGAN) dilakukan menggunakan *temporal similarity* untuk mengukur tingkat kemiripan pola temporal dan karakteristik statistik antara data sintetis dan data asli. Evaluasi dilakukan menggunakan metrik *Mean Absolute Error* (MAE) dan *Dynamic Time Warping* (DTW). MAE digunakan untuk mengukur rata-rata perbedaan numerik antara sequence data sintetis dan data asli, sedangkan DTW digunakan untuk mengukur tingkat kemiripan pola temporal antar deret waktu meskipun terdapat pergeseran waktu atau perbedaan skala temporal.

Selanjutnya, evaluasi kinerja model prediksi dilakukan menggunakan metrik *Mean Absolute Error* (MAE). MAE digunakan untuk mengukur rata-rata kesalahan absolut antara nilai prediksi dan nilai aktual, serta memberikan interpretasi kesalahan yang lebih alami tanpa memberikan bobot berlebih pada kesalahan yang besar (Willmott dan Matsuura 2005). Nilai MAE dihitung menggunakan Persamaan (12):

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |x_i - \hat{x}_i|\,, \tag{12}$$

dengan x_i merupakan nilai aktual, $\hat{x}_i$ merupakan nilai prediksi, dan n adalah jumlah data.

Selain itu, penelitian ini juga menggunakan metode *Dynamic Time Warping* (DTW) untuk mengukur tingkat kemiripan pola antara dua deret waktu, yaitu antara data asli dengan hasil prediksi. DTW mampu mengukur kesamaan dua deret waktu yang mungkin memiliki perbedaan dalam skala waktu atau kecepatan (Berndt dan Clifford 1994). Nilai DTW dihitung menggunakan Persamaan (13):

$$\mathrm{DTW}(S,T) = \min_{W} \textstyle\sum_{k=1}^{P} \delta(w_k)\,, \tag{13}$$

dengan S dan T merupakan deret waktu yang dibandingkan, W adalah jalur lengkungan yang memetakan kedua deret waktu sehingga jaraknya minimum, dan $\delta(w_k)$ merupakan fungsi jarak, yaitu $\delta(i,j) = |s_i - t_j|$, di mana w_k adalah titik ke-k pada jalur tersebut.

Melalui kombinasi metrik MAE dan DTW, evaluasi dalam penelitian ini tidak hanya mengukur tingkat akurasi numerik, tetapi juga kemampuan model dalam menangkap pola temporal pada data deret waktu.

Pembuatan *Web Service* Model Prediksi

Pembuatan *web service* prediksi dilakukan untuk mengintegrasikan model prediksi yang telah dikembangkan dengan aplikasi Prodigo. Integrasi ini memungkinkan sistem untuk melakukan prediksi penjualan secara otomatis melalui mekanisme komunikasi berbasis jaringan. Web service dikembangkan menggunakan *framework* Flask, yaitu *micro web framework* berbasis Python yang ringan dan fleksibel untuk membangun layanan berbasis *Application Programming Interface* (API) (Mufid *et al.* 2019). Flask memungkinkan pengelolaan *request* dan *response* dari pengguna serta mempermudah integrasi model *deep learning* ke dalam sistem berbasis web (Bonney *et al.* 2021). Selain itu, penggunaan API berbasis HTTP merupakan pendekatan umum dalam menghubungkan model *machine learning* dengan aplikasi eksternal secara *real-time* (Fielding 2000).

Dalam implementasinya, aplikasi Prodigo akan mengirimkan permintaan (*request*) ke *web service* melalui protokol HTTP dengan parameter *input* berupa data historis penjualan (misalnya beberapa minggu terakhir) serta jumlah periode prediksi yang diinginkan. Selanjutnya, *web service* akan memproses *input* tersebut menggunakan model LSTM yang telah dilatih dan mengembalikan hasil prediksi dalam bentuk *response* yang dapat digunakan oleh aplikasi. Dengan adanya *web service* ini, model prediksi dapat diakses secara fleksibel dan

terintegrasi dengan sistem operasional, sehingga mendukung pengambilan keputusan berbasis data secara lebih efektif.

HASIL DAN PEMBAHASAN

Hasil Penelitian Tanpa Data Sintetis

Pada tahap ini, dilakukan pembangunan model Long Short-Term Memory (LSTM) menggunakan dua pendekatan optimizer, yaitu ADAM standar dan ADAM modifikasi. Untuk memperoleh konfigurasi terbaik, dilakukan proses *grid search* terhadap *hyperparameter* ADAM. Hasil konfigurasi *hyperparameter* optimal untuk kedua pendekatan ditunjukkan pada Tabel 6.

Tabel 6 *Hyperparameter* ADAM Terbaik Tanpa Data Sintetis

Hyperparameter	ADAM Standar	ADAM Modifikasi
Learning Rate	0.001	0.001
Beta 1	0.9	0.9
Beta 2	0.999	0.999
Epsilon	1e-8	1e-8
Batch Size	16	16

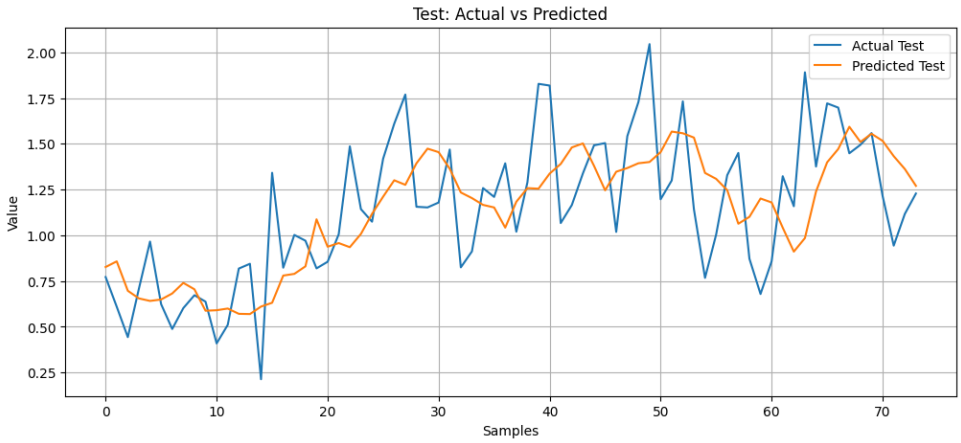
Berdasarkan konfigurasi tersebut, dilakukan pelatihan model sebanyak 20 kali pelatihan berulang untuk mengevaluasi stabilitas performa model. Evaluasi dilakukan menggunakan metrik *Mean Absolute Error* (MAE), sedangkan konsistensi pola temporal dievaluasi menggunakan *Dynamic Time Warping* (DTW). Hasil perbandingan kinerja model ditunjukkan pada Tabel 7.

Tabel 7 Rata-Rata Perbandingan Kinerja Antara ADAM Standar dan ADAM Modifikasi Tanpa Data Sintetis

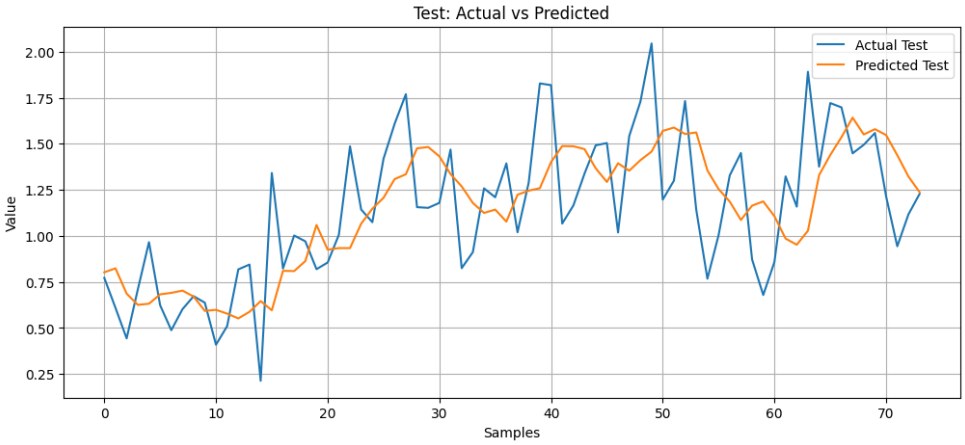
Komponen	ADAM Standar	ADAM Modifikasi
Mean MAE	278 ± 10	272 ± 6
Mean Training Time (detik)	7	7

Hasil evaluasi berdasarkan 20 kali pelatihan berulang menunjukkan bahwa ADAM modifikasi menghasilkan rata-rata MAE sebesar 272 ± 6, sedangkan ADAM standar menghasilkan 278 ± 10. Meskipun nilai rata-rata MAE ADAM modifikasi lebih rendah dan variasi hasilnya lebih kecil, perbedaan yang diperoleh masih relatif kecil sehingga kedua metode dapat dianggap memberikan performa yang sebanding. Namun demikian, berdasarkan evaluasi menggunakan *Dynamic Time Warping* (DTW), kedua model masih belum mampu menangkap pola temporal antara data aktual dan hasil prediksi secara optimal. Hal ini ditunjukkan oleh nilai rata-rata DTW sebesar 4, yang mengindikasikan bahwa masih terdapat perbedaan pola temporal yang cukup signifikan antara data aktual dan hasil prediksi. Visualisasi hasil prediksi terhadap data uji menggunakan ADAM standar ditunjukkan pada Gambar 9, sedangkan hasil prediksi menggunakan ADAM modifikasi ditunjukkan pada Gambar 10.

Temuan ini menunjukkan bahwa meskipun modifikasi ADAM mampu meningkatkan akurasi numerik dan kestabilan proses pelatihan. Namun, model LSTM yang dibangun tanpa dukungan data sintetis masih memiliki keterbatasan dalam mempelajari karakteristik pola temporal data deret waktu secara menyeluruh. Oleh karena itu, diperlukan pendekatan tambahan, seperti augmentasi data, untuk membantu model dalam menangkap dinamika temporal data dengan lebih baik.



Gambar 9 Hasil Prediksi Menggunakan Data Uji ADAM Standar Tanpa Data Sintetis



Gambar 10 Hasil Prediksi Menggunakan Data Uji ADAM Modifikasi Tanpa Data Sintetis

Hasil Penelitian Menggunakan Data Sintetis

a. Pembuatan model dan evaluasi model TimeGAN

Pada tahap ini, dilakukan pembangunan model *Time-Series Generative Adversarial Networks* (TimeGAN) untuk menghasilkan data sintetis melalui proses augmentasi data. Untuk memperoleh konfigurasi terbaik, dilakukan *grid search* terhadap *hyperparameter* TimeGAN. Dalam proses ini, data latih diperluas dengan membangkitkan 618 data sintetis. Hasil konfigurasi hyperparameter optimal ditunjukkan pada Tabel 8 dan hasil evaluasi distribusi statistik antara data asli dengan data sintetis Tabel 9.

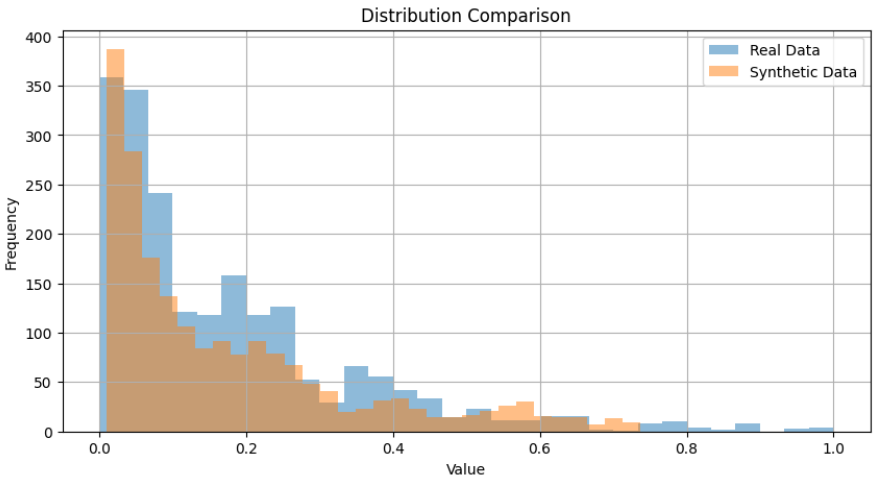
Tabel 8 *Hyperparameter* TimeGAN Terbaik

Hyperparameter	Nilai
Learning Rate	0.0005
Noise Dimension	32
Layer Dimension	64

Tabel 9 Hasil Evaluasi Distribusi Statistik

Statistik	Data Asli	Data Sintetis
Mean	0.174736	0.174553
Std Dev	0.174072	0.171731

Berdasarkan Tabel 9, data sintetis yang dihasilkan oleh TimeGAN memiliki karakteristik distribusi yang mendekati data asli. Hal ini ditunjukkan oleh nilai *mean* dan standar deviasi yang relatif serupa antara data asli dan data sintetis. Hasil evaluasi *temporal similarity* menunjukkan nilai MAE sebesar 0.1588 dan DTW sebesar 1.1616. Nilai tersebut menunjukkan bahwa data sintetis mampu mempertahankan kemiripan numerik dan pola temporal data asli dengan cukup baik. Visualisasi grafik histogram sebaran data antara data asli dan data sintetis dapat dilihat pada Gambar 11.



Gambar 11 Visualisai Grafik Histogram Sebaran Data Asli dan Data Sintetis

b. Pembuatan model dan evaluasi model LSTM

Setelah proses augmentasi data, dilakukan pembangunan model *Long Short-Term Memory* (LSTM) menggunakan dua pendekatan optimizer, yaitu ADAM standar dan ADAM modifikasi. Proses optimasi *hyperparameter* dilakukan menggunakan metode *grid search*, dan hasil konfigurasi terbaik ditunjukkan pada Tabel 10. Selanjutnya, model dilatih sebanyak 20 kali pelatihan berulang untuk mengevaluasi stabilitas performa model. Evaluasi dilakukan menggunakan metrik *Mean Absolute Error* (MAE), sedangkan konsistensi pola temporal dievaluasi menggunakan *Dynamic Time Warping* (DTW). Hasil perbandingan kinerja kedua pendekatan ditunjukkan pada Tabel 11.

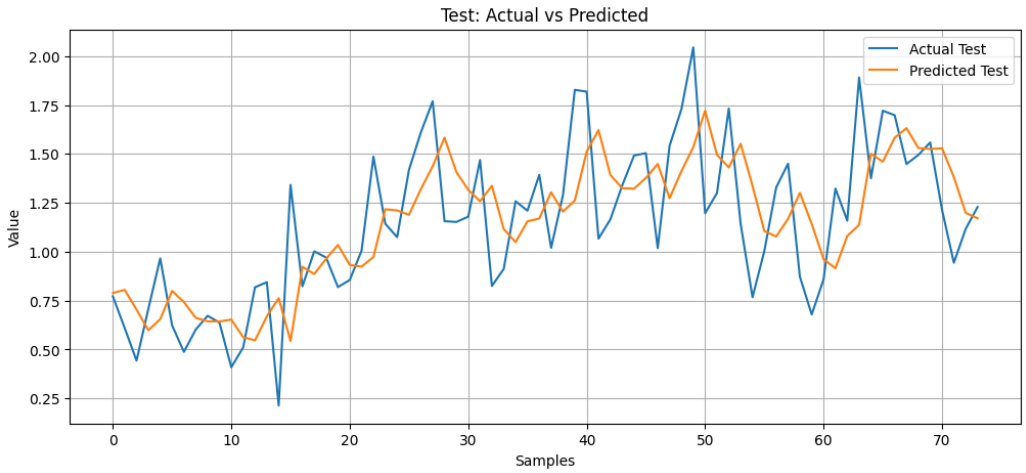
Tabel 10 *Hyperparameter* ADAM Terbaik Menggunakan Data Sintetis

Hyperparameter	ADAM Standar	ADAM Modifikasi
Learning Rate	0.001	0.001
Beta 1	0.9	0.9
Beta 2	0.9999	0.9999
Epsilon	1e-8	1e-8
Batch Size	32	32

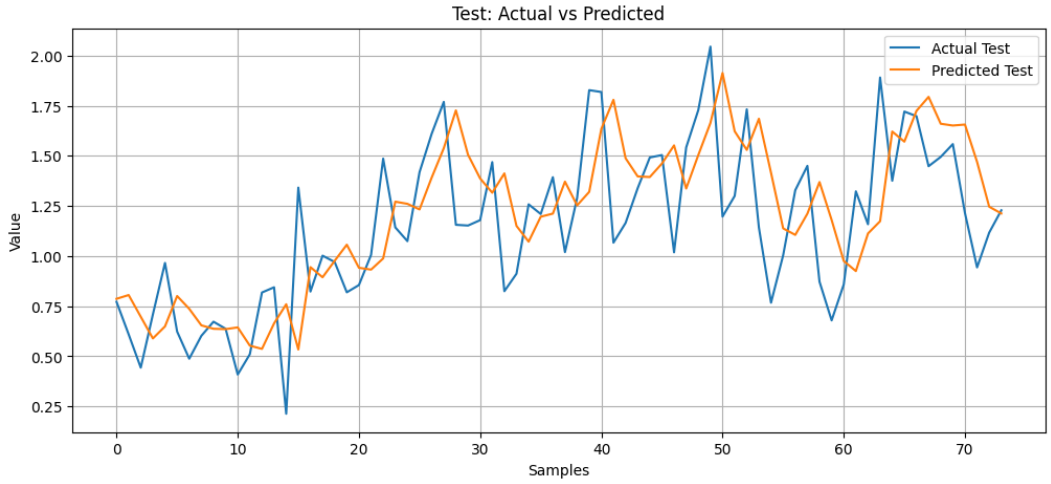
Tabel 11 Perbandingan Kinerja Antara ADAM Standar dan ADAM Modifikasi Menggunakan Data Sintetis

Komponen	ADAM Standar	ADAM Modifikasi
Mean MAE	272±13	270±4
Mean Training Time (detik)	12	12

Hasil evaluasi berdasarkan 20 kali pelatihan berulang menunjukkan bahwa ADAM modifikasi menghasilkan rata-rata MAE yang sedikit lebih rendah dibandingkan ADAM standar, yaitu sebesar 270 ± 4 dibandingkan 272 ± 13. Hasil ini menunjukkan bahwa ADAM modifikasi mampu memberikan performa yang relatif lebih baik dan lebih stabil dibandingkan ADAM standar, meskipun selisih performa yang diperoleh tidak terlalu besar. Selain itu, evaluasi menggunakan *Dynamic Time Warping* (DTW) menunjukkan bahwa model yang menggunakan data sintetis mampu mengikuti pola temporal data aktual dengan cukup baik, ditunjukkan oleh nilai rata-rata DTW sebesar 3.5. Hasil ini mengindikasikan bahwa penambahan data sintetis membantu model dalam mempelajari karakteristik pola temporal data deret waktu, meskipun peningkatan akurasi numerik berdasarkan MAE belum terlalu signifikan. Visualisasi hasil prediksi terhadap data uji menggunakan ADAM standar ditunjukkan pada Gambar 12, sedangkan hasil prediksi menggunakan ADAM modifikasi ditunjukkan pada Gambar 13.



Gambar 12 Hasil Prediksi Menggunakan Data Uji ADAM Standar Menggunakan Data Sintetis



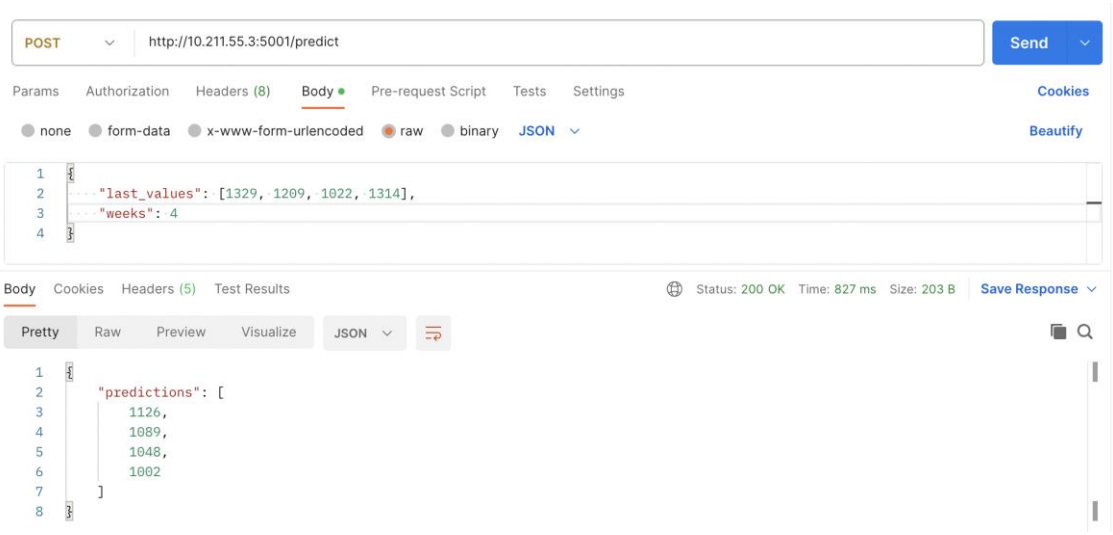
Gambar 13 Hasil Prediksi Menggunakan Data Uji ADAM Modifikasi Menggunakan Data Sintetis

Kemampuan model dalam mempertahankan pola temporal menjadi aspek penting dalam mendukung pengambilan keputusan bisnis berbasis data (Ma dan Zhang 2025). Berdasarkan hasil evaluasi secara keseluruhan, penggunaan data sintetis hasil TimeGAN dan optimasi menggunakan ADAM modifikasi menunjukkan kemampuan yang lebih baik dalam mempertahankan pola temporal data serta menghasilkan performa prediksi yang relatif stabil.

Hasil Penelitian Pembuatan *Web Service* Prediksi

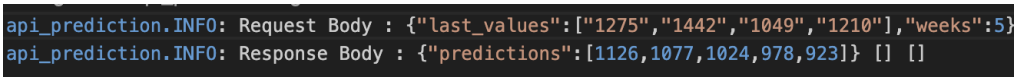
Pada tahap ini, model prediksi yang telah dikembangkan diimplementasikan dalam bentuk *web service* menggunakan *framework* Python Flask. Implementasi ini bertujuan untuk mengintegrasikan model prediksi ke dalam aplikasi Prodigo sehingga dapat digunakan secara operasional.

Web service yang dibangun menerima parameter input berupa *last value*, yaitu data penjualan dalam beberapa periode terakhir (minimal 4 minggu), serta parameter *weeks* yang menunjukkan jumlah periode ke depan yang akan diprediksi. Berdasarkan input tersebut, sistem akan memproses data menggunakan model LSTM yang telah dilatih dan menghasilkan *response* berupa nilai prediksi sesuai dengan jumlah periode yang diminta. Contoh implementasi *web service* prediksi ditunjukkan pada Gambar 13.

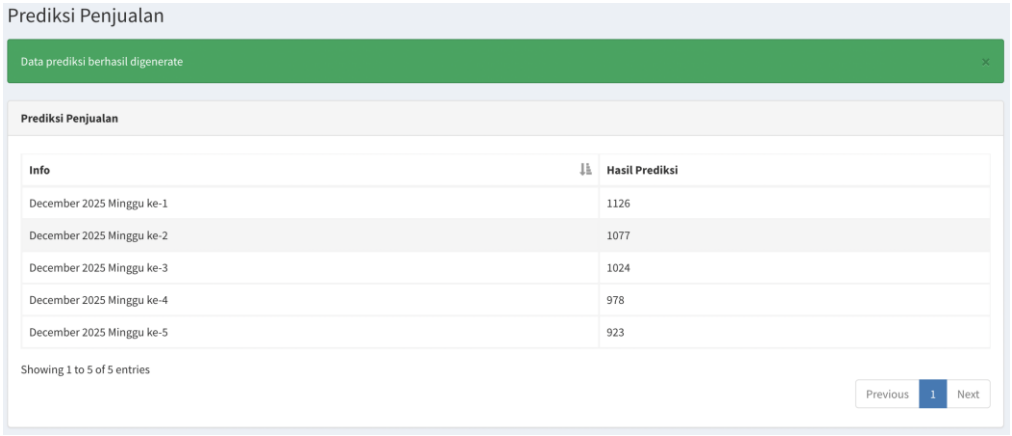


Gambar 13 Web Service Prediksi

Dalam implementasinya, setiap permintaan (*request*) yang dikirimkan oleh aplikasi Prodigo ke *web service* akan dicatat oleh sistem dalam bentuk *logging*. Informasi yang dicatat meliputi parameter input dan hasil prediksi yang dihasilkan (*response*), sehingga memungkinkan proses monitoring dan evaluasi kinerja sistem secara berkelanjutan. Ilustrasi pencatatan *logging web service* ditunjukkan pada Gambar 14. Selain itu, hasil prediksi yang dihasilkan oleh web service juga disimpan dan ditampilkan pada aplikasi Prodigo untuk mendukung kebutuhan analisis dan pengambilan keputusan. Tampilan hasil prediksi pada aplikasi ditunjukkan pada Gambar 15.



Gambar 14 *Logging Web Service* Prediksi



Gambar 15 Halaman Hasil Prediksi

Implementasi *web service* ini menunjukkan bahwa model prediksi tidak hanya mampu memberikan hasil prediksi, tetapi juga dapat diintegrasikan ke dalam sistem aplikasi secara praktis. Dengan demikian, hasil prediksi dapat dimanfaatkan secara langsung oleh pengguna dalam mendukung perencanaan produksi dan pengambilan keputusan berbasis data.

SIMPULAN

Penelitian ini menunjukkan bahwa ADAM modifikasi dan ADAM standar memberikan performa prediksi yang relatif sebanding pada model LSTM untuk prediksi penjualan produk sepatu merek Prodigo. Pada skenario tanpa data sintetis, ADAM modifikasi menghasilkan nilai MAE sebesar 272 ± 6, sedangkan ADAM standar menghasilkan MAE sebesar 278 ± 10.

Sementara itu, pada skenario dengan data sintetis, ADAM modifikasi menghasilkan MAE sebesar 270 ± 4 dibandingkan ADAM standar sebesar 272 ± 13. Meskipun ADAM modifikasi cenderung menghasilkan nilai MAE rata-rata yang sedikit lebih rendah pada kedua skenario pengujian, perbedaan yang diperoleh masih relatif kecil dan rentang hasil kedua metode masih beririsan. Temuan ini mengindikasikan potensi manfaat ADAM modifikasi dalam meningkatkan akurasi dan konsistensi pelatihan, namun diperlukan analisis lebih lanjut untuk mengonfirmasi signifikansi perbedaannya.

Hasil penelitian juga menunjukkan bahwa penggunaan data sintetis hasil *Time-Series Generative Adversarial Networks* (TimeGAN) membantu model dalam mempertahankan pola temporal data deret waktu. Hal ini ditunjukkan oleh nilai Dynamic Time Warping (DTW) rata-rata sebesar 3.5 pada skenario penggunaan data sintetis, yang lebih baik dibandingkan scenario tanpa data sintetis dengan nilai rata-rata DTW sebesar 4. Selain itu, evaluasi distribusi statistik menunjukkan bahwa data sintetis memiliki karakteristik distribusi yang mendekati data asli, ditunjukkan oleh nilai *mean* dan standar deviasi yang relatif serupa.

DAFTAR PUSTAKA

Ali PJM. 2022. Investigating the Impact of Min-Max Data Normalization on the Regression Performance of K-Nearest Neighbor with Different Similarity Measurements. *ARO-The Scientific Journal of Koya University*. 10(1):85–91. doi:10.14500/aro.10955.

Bergstra J, Bengio Y. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*. 13:281–305. http://scikit-learn.sourceforge.net.

Berndt DJ, Clifford J. 1994. Using dynamic time warping to find patterns in time series. Di dalam: *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*. Seattle, USA. hlm 359–370. www.aaai.org.

Bohórquez Correa S, Mosquera-López S, Uribe JM. 2026. Time-varying systemic risk in electricity markets using generative adversarial networks: Market resilience and policy. *Energy Policy*. 210. doi:10.1016/j.enpol.2025.115034.

Bonney MS, Angelis M De, Wagg D, Borgo MD. 2021. Digital twin operational platform for connectivity and accessibility using Flask Python. Di dalam: *Companion Proceedings - 24th International Conference on Model-Driven Engineering Languages and Systems, MODELS-C 2021*. Institute of Electrical and Electronics Engineers Inc. hlm 237–241.

Brownlee J. 2018. *Deep Learning for Time Series Forecasting: Predict the Future with MLPs, CNNs and LSTMs in Python*. Melbourne, Australia: Machine Learning Mastery.

Chandriah KK, Naraganahalli R V. 2021. RNN / LSTM with modified Adam optimizer in deep learning approach for automobile spare parts demand forecasting. *Multimed Tools Appl*. 80(17):26145–26159. doi:10.1007/s11042-021-10913-0.

Chang Z, Zhang Y, Chen W. 2018. Effective Adam-optimized LSTM Neural Network for Electricity Price Forecasting. Di dalam: *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*. Chengdu, China: IEEE. hlm 1678.

Fielding RT. 2000. Architectural styles and the design of network-based software architectures [Doctoral dissertation]. Irvine, USA: University of California.

Goodfellow I, Bengio Y, Courville A. 2016. *Deep Learning*. Cambridge, MA, USA: MIT Press.

Hastie T, Tibshirani R, Friedman J. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Ed ke-2. New York, USA: Springer.

Hochreiter S, Schmidhuber J. 1997. Long Short-Term Memory. *Neural Comput*. 9(8):1735–1780.

Huang K, Luo N. 2023. Improved TimeGAN Based on Attention for Time Series Prediction Method with Few Shot. *Huadong Ligong Daxue Xuebao/Journal of East China University of Science and Technology*. 49(6):890–899. doi:10.14135/j.cnki.1006-3080.20220723001.

Ilemobayo JA, Durodola O, Alade O, Awotunde OJ, Olanrewaju AT, Falana O, Ogungbire A, Osinuga A, Ogunbiyi D, Ifeanyi A, *et al.* 2024. Hyperparameter Tuning in Machine Learning: A Comprehensive Review. *Journal of Engineering Research and Reports*. 26(6):388–395. doi:10.9734/jerr/2024/v26i61188.

Jörges C, Berkenbrink C, Stumpe B. 2021. Prediction and reconstruction of ocean wave heights based on bathymetric data using LSTM neural networks. *Ocean Engineering*. 232. doi:10.1016/j.oceaneng.2021.109046.

Kingma DP, Ba JL. 2015. Adam: A Method for Stochastic Optimization. Di dalam: International Conference on Learning Representations (ICLR 2015). San Diego, USA. http://arxiv.org/abs/1412.6980.

Ma X, Zhang H. 2025. Time Series Forecasting Method Based on Multi-Scale Feature Fusion and Autoformer. Applied Sciences (Switzerland). 15(7). doi:10.3390/app15073768.

Mufid MR, Basofi A, Rasyid MUH Al, Rochimansyah IF, Rokhim A. 2019. Design an MVC model using Python for Flask framework development. Di dalam: *2019 International Electronics Symposium (IES)*. Surabaya, Indonesia: IEEE. hlm 1.

Pacella M, Papadia G. 2021. Evaluation of deep learning with long short-term memory networks for time series forecasting in supply chain management. Di dalam: *Procedia CIRP*. Volume ke-99. Elsevier B.V. hlm 604–609.

Pranolo A, Setyaputri FU, Paramarta AKI, Triono APP, Fadhilla AF, Akbari AKG, Utama ABP, Wibawa AP, Uriu W. 2024. Enhanced Multivariate Time Series Analysis Using LSTM: A Comparative Study of Min-Max and Z-Score Normalization Techniques. *ILKOM Jurnal Ilmiah*. 16(2):210–220. doi:10.33096/ilkom.v16i2.2333.210-220.

Reddi SJ, Kale S, Kumar S. 2019 Apr 19. On the Convergence of Adam and Beyond. http://arxiv.org/abs/1904.09237.

Shorten C, Khoshgoftaar TM. 2019. A survey on image data augmentation for deep learning. *J Big Data*. 6(1):60. doi:10.1186/s40537-019-0197-0.

Tekkali CG, Natarajan K. 2024. Assessing CNN's Performance with Multiple Optimization Functions for Credit Card Fraud Detection. Di dalam: *Procedia Computer Science*. Volume ke-235. Elsevier B.V. hlm 2035–2042.

Willmott CJ, Matsuura K. 2005. Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance. Clim Res. 30:79–82. doi:10.3354/cr030079.

Yi D, Ahn J, Ji S. 2020. An effective optimization method for machine learning based on ADAM. *Applied Sciences (Switzerland)*. 10(3). doi:10.3390/app10031073.

Yoon J, Jarrett D, Van Der Schaar M. 2019. Time-series Generative Adversarial Networks. Di dalam: *Advances in Neural Information Processing Systems 32 (NeurIPS 2019)*. Vancouver, Canada: Curran Associates Inc. hlm 5508–5518.