BAB II

TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Tabel berikut merangkum beberapa penelitian relevan yang menjadi acuan dalam studi ini.

Tabel 2.1 Ringkasan Penelitian Terkait Model Prediksi Harga Bitcoin

Judul Penelitian	**Peneliti, Tahun**	**Metode**	**Hasil**
Enhancing Bitcoin Price Prediction with Deep Learning: Integrating Social Media Sentiment and Historical Data [7]	Hla Soe Htay, Mani Ghahremani, Stavros Shiaeles, 2025	*Multivariate* LSTM + Sentimen Twitter	Model dengan sentimen Twitter terbukti paling unggul dari lima model yang diuji, mencapai MAE = 0.00196 dan RMSE = 0.00304.
Optimizing Bitcoin Price Prediction: Multivariate LSTM Triumphs [8]	Brian Scanlon, Keith Quille, Rajesh Jaiswal, 2025	*Multivariate* LSTM	Dengan input sentimen (Google Trends, frekuensi tweet) dan data ekonomi (USD, Emas, VIX), model terbukti mengungguli RNN, SVR, ANN, dan ARIMA dengan RMSE = 268.83.
A Multivariate LSTM-Based Deep Learning Model for Stock Market Prediction [12]	Samuel Ibukun Olotu, 2023	*Multivariate* LSTM	Dalam konteks prediksi pasar saham, model *Multivariate* LSTM yang menggunakan data harga historis menunjukkan kinerja yang lebih baik dibandingkan metode tradisional dan DL lainnya.

2.2 Bitcoin

Bitcoin merupakan aset kripto pertama yang diperkenalkan oleh Satoshi Nakamoto pada tahun 2008 sebagai sistem mata uang digital terdesentralisasi berbasis *peer-to-peer*, yang memungkinkan transaksi tanpa perantara [1]. Sistem ini berjalan di atas teknologi *blockchain* dan menggunakan mekanisme konsensus *Proof-of-Work* untuk memverifikasi dan mencatat transaksi secara permanen. Karakteristik

seperti desentralisasi, transparansi, dan keterbatasan suplai hingga 21 juta unit menjadikan Bitcoin menarik sebagai aset alternatif dan sering disebut sebagai "emas digital". Namun, sifat pasar kripto yang tidak terpusat, beroperasi 24/7, dan sangat dipengaruhi oleh faktor spekulatif serta sentimen publik menyebabkan volatilitas harga Bitcoin sangat tinggi, dapat mencapai sepuluh kali lipat dibanding mata uang fiat [13].

2.3 *Crypto Fear and Greed Index* (FGI)

Crypto Fear and Greed Index (FGI) adalah indikator sentimen pasar kripto yang mengukur emosi kolektif investor, dari ketakutan hingga keserakahan, dalam skala 0–100. FGI tersusun atas enam komponen utama, yaitu volatilitas (25%), volume (25%), media sosial (15%), survei (15%), dominasi Bitcoin (10%), dan tren pencarian (10%) [14]. Indeks ini mencerminkan peran emosi dalam volatilitas pasar. Nilai FGI dikelompokkan ke dalam lima kategori utama, yaitu *Extreme Fear* (0–24), *Fear* (25–49), *Neutral* (50), *Greed* (51–74), dan *Extreme Greed* (75–100) [15].

2.4 Korelasi Pearson

Korelasi Pearson mengukur kekuatan dan arah hubungan linier antara dua variabel numerik. Nilai koefisien korelasi r berada dalam rentang -1 hingga +1. Nilai r positif menunjukkan hubungan searah, sedangkan nilai negatif menunjukkan hubungan berlawanan arah. Korelasi dianggap signifikan secara statistik apabila nilai p-value < 0,05 [16]. Koefisien korelasi Pearson dihitung menggunakan Persamaan 2.1 berikut:

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.1}$$

Keterangan:

r : Koefisien korelasi Pearson,
x_i, y_i : Data ke-i dari variabel x dan y,
$\bar{x}, \bar{y}$: Rata-rata dari variabel x dan y,

n : Jumlah pasangan data.

Untuk menguji signifikansi statistik dari nilai r, digunakan uji t dengan derajat kebebasan $(n-2)$ menggunakan Persamaan 2.2:

$$t = \frac{r\sqrt{n-2}}{\sqrt{1-r^2}} \tag{2.2}$$

Keterangan:

t : Nilai statistik t untuk uji signifikansi,

r : Koefisien korelasi Pearson,

n : Jumlah pasangan data.

Nilai t kemudian digunakan untuk menghitung p-value berdasarkan distribusi t-Student dua sisi dengan derajat kebebasan $(n-2)$. Jika p-value $< 0{,}05$, maka hubungan antara kedua variabel dianggap signifikan secara statistik.

2.5 *Min-Max Scaler*

Min-Max Scaler merupakan metode normalisasi yang digunakan untuk mengubah skala data ke dalam rentang tertentu, biasanya antara 0 dan 1. Normalisasi dilakukan agar skala data pada setiap fitur sama sehingga model dapat belajar secara optimal [17]. Proses normalisasi dilakukan menggunakan Persamaan 2.3 berikut:

$$X_{\text{norm}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \tag{2.3}$$

Keterangan:

X_{norm} : Nilai hasil normalisasi,

X : Nilai asli dari data,

X_{min} : Nilai minimum dalam data,

X_{max} : Nilai maksimum dalam data.

Inverse transform adalah metode untuk mengembalikan data yang telah dinormalisasi ke skala aslinya agar hasil prediksi dapat diterjemahkan ke dalam skala aslinya [17]. Rumusnya ditunjukkan pada Persamaan 2.4 berikut:

$$X = X_{\text{norm}} \cdot (X_{\max} - X_{\min}) + X_{\min} \tag{2.4}$$

Keterangan:

X : Nilai asli dari data, hasil *inverse transform*,

X_{norm} : Nilai hasil prediksi yang telah dinormalisasi,

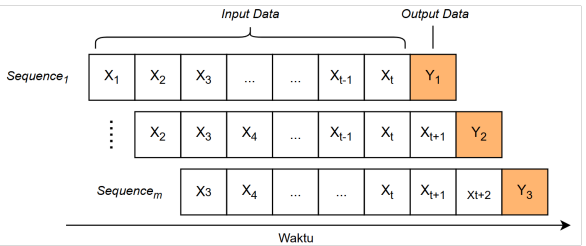
$X_{\min}$, $X_{\max}$: Nilai minimum dan maksimum.

2.6 *Bobot*

Bobot (*weight*) adalah parameter internal pada neuron yang menentukan seberapa besar pengaruh setiap input terhadap output. Dalam konteks pembelajaran mesin, proses pelatihan bertujuan menemukan nilai bobot yang optimal agar model dapat memetakan input ke output secara akurat. Bobot diperbarui secara bertahap menggunakan algoritma *optimizer* berdasarkan nilai gradien dari fungsi kerugian [18].

2.7 *Sliding Window*

Sliding window menyusun data deret waktu menjadi pasangan *input-output* untuk pelatihan model. Setiap sampel terdiri dari sejumlah data historis sepanjang t hari sebagai *input* (X), dan nilai pada hari berikutnya sebagai target (Y). Seperti ditunjukkan pada Gambar 2.1, jendela digeser satu langkah untuk menghasilkan sampel berikutnya secara berurutan hingga seluruh data terpenuhi [19].



Gambar 2.1 Proses *sliding window*

2.8 *Hyperparameter*

Hyperparameter merupakan parameter konfigurasi eksternal yang nilainya ditetapkan sebelum proses pelatihan model dimulai. Parameter ini tidak dipelajari secara langsung oleh model dari data, melainkan mengontrol bagaimana model belajar. Pemilihan *hyperparameter* yang tepat sangat krusial untuk mencapai performa model yang optimal [18]. Dalam pengembangan model *deep learning*, terdapat sejumlah hyperparameter utama yang secara umum memengaruhi performa model. Beberapa di antaranya adalah sebagai berikut:

1. *Sequence Length*
   Sequence length adalah jumlah langkah waktu historis yang digunakan model untuk memprediksi nilai di masa depan. Pemilihannya memengaruhi kemampuan model dalam menangkap pola jangka pendek maupun panjang [18].
2. Jumlah Lapisan LSTM
   Jumlah lapisan LSTM mengacu pada banyaknya lapisan LSTM yang ditumpuk secara vertikal. Menambah jumlah lapisan dapat memungkinkan model untuk mempelajari representasi data yang lebih kompleks dan hierarkis, namun juga meningkatkan risiko *overfitting* dan beban komputasi [18].
3. Neuron
   Neuron merupakan unit komputasi dasar dalam jaringan saraf tiruan, termasuk pada arsitektur *LSTM*. Setiap *neuron* menerima satu atau lebih sinyal masukan, mengalikannya dengan *weight*, menambahkan *bias*, lalu menerapkan fungsi aktivasi non-linear untuk menghasilkan keluaran. Sekumpulan *neuron* membentuk lapisan tersembunyi yang mampu merepresentasikan pola kompleks secara hierarkis, sehingga memungkinkan jaringan mempelajari hubungan non-linier dari data [18].
4. *Dropout Rate*
   Dropout adalah teknik regularisasi yang secara acak menonaktifkan sebagian *neuron* dan koneksinya saat pelatihan,

guna mencegah *overfitting*. Pendekatan ini memaksa model belajar representasi yang lebih umum dan tidak bergantung pada jalur aktivasi tertentu. Dengan demikian, *dropout* meningkatkan kemampuan generalisasi jaringan terhadap data yang belum pernah dilihat [20].

5. *Kernel Regularization* (L_2)

   Regularisasi L_2 adalah teknik yang menambahkan penalti terhadap besarnya bobot (*weights*) ke dalam fungsi kerugian, dengan tujuan mencegah model memiliki parameter yang terlalu besar dan kompleks. Penalti ini berbentuk jumlah kuadrat bobot, yang mendorong nilai bobot menuju nol tanpa benar-benar mengeliminasi fitur. Dengan demikian, L_2 regularisasi membantu menekan varians, meningkatkan kemampuan generalisasi, dan mengurangi risiko *overfitting*, terutama saat model dilatih pada data yang terbatas [18]. Rumus regularisasi L_2 untuk fungsi kerugian *Mean Squared Error (MSE)* ditunjukkan pada Persamaan 2.5.

$$L_{total} = \text{MSE} + \lambda \sum_i w_i^2 \tag{2.5}$$

   Keterangan:

   L_{total} : Fungsi kerugian total setelah regularisasi,
   λ : Koefisien regularisasi,
   w_i : Bobot model yang dikenai penalti.

6. *Optimizer*

   Optimizer adalah algoritma yang memperbarui bobot jaringan saraf untuk meminimalkan fungsi kerugian [21]. Pemilihan optimizer yang tepat berpengaruh besar terhadap efisiensi dan hasil pelatihan model.

   (a) RMSprop

   Root Mean Square Propagation (RMSprop) menyesuaikan *learning rate* untuk setiap parameter menggunakan rata-rata pergerakan eksponensial dari gradien kuadrat. Pendekatan ini membantu mempercepat konvergensi,

terutama pada data yang bersifat non-stasioner [22].

(b) Adam

Adaptive Moment Estimation (Adam) menggabungkan prinsip RMSprop dan momentum dengan menghitung dua jenis rata-rata pergerakan, yaitu gradien (momen pertama) dan gradien kuadrat (momen kedua). Adam dikenal efisien secara komputasi dan sering memberikan hasil baik dengan sedikit penyetelan *hyperparameter* [21].

7. *Learning Rate*

   Learning rate adalah *hyperparameter* yang menentukan besarnya langkah dalam pembaruan bobot pada setiap iterasi pelatihan. Nilai *learning rate* yang terlalu kecil akan memperlambat proses pelatihan, sedangkan nilai yang terlalu besar dapat menyebabkan ketidakstabilan model dan menghambat konvergensi [23].

8. *Batch Size*, Iterasi, dan *Epoch*

   Batch size adalah jumlah sampel data yang diproses sekaligus sebelum bobot jaringan diperbarui. Satu iterasi mengacu pada satu kali pembaruan bobot berdasarkan satu *batch*. Sementara itu, satu *epoch* adalah satu siklus penuh di mana seluruh dataset pelatihan telah digunakan sekali untuk melatih model. Pelatihan umumnya dilakukan dalam beberapa epoch untuk memungkinkan model menyerap pola dari data secara bertahap [18]. Epoch maksimum yang digunakan dalam pencarian hyperparameter ini adalah 150.

9. *Early Stopping*

   Early stopping adalah teknik untuk mencegah *overfitting* dengan menghentikan pelatihan saat performa pada data validasi tidak lagi membaik setelah beberapa *epoch* (*patience*). Bobot terbaik dari *epoch* sebelumnya akan disimpan sebagai model akhir, sehingga model tidak belajar berlebihan dari *noise* [24].

10. Fungsi Aktivasi

    Fungsi aktivasi memungkinkan jaringan saraf untuk mempelajari pola *non-linier* yang kompleks. Tanpa fungsi aktivasi, sebuah

model hanya akan mampu membentuk pemetaan linier, terlepas dari jumlah lapisan yang dimilikinya, sehingga membatasi kemampuannya secara signifikan [23]. Dalam arsitektur *Long Short-Term Memory* (LSTM), dua fungsi aktivasi utama yang digunakan adalah Sigmoid dan Tanh (*Hyperbolic Tangent*).

(a) Fungsi Sigmoid

Fungsi aktivasi Sigmoid, yang juga dikenal sebagai fungsi logistik, memetakan nilai input numerik ke dalam rentang antara 0 dan 1. Karena outputnya dapat diinterpretasikan sebagai probabilitas, fungsi ini secara historis populer digunakan pada lapisan output untuk masalah klasifikasi biner [25]. Rumus matematis untuk fungsi Sigmoid dinyatakan dalam Persamaan 2.6.

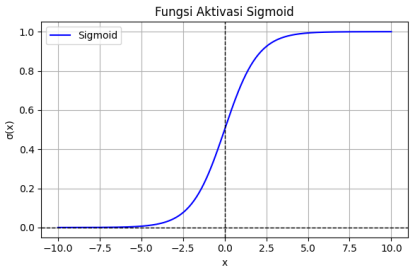
$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

Keterangan:

$\sigma(x)$: *Output* fungsi Sigmoid,

x : *Input linear* dari neuron.

Kurva fungsi Sigmoid ditampilkan pada Gambar 2.2, yang memperlihatkan bahwa *output* fungsi ini selalu berada pada rentang (0, 1), dengan transisi yang halus.



Gambar 2.2 Kurva Fungsi Aktivasi Sigmoid

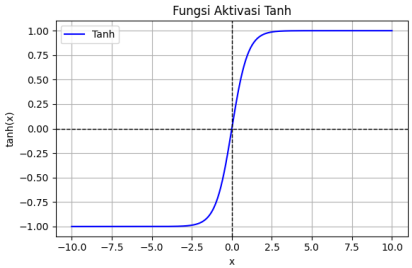
(b) Fungsi Tanh (*Hyperbolic Tangent*)

Fungsi aktivasi Tanh (*hyperbolic tangent*) adalah fungsi non-linear yang memetakan nilai input ke dalam rentang antara -1 dan 1. Fungsi ini pada dasarnya adalah versi dari

fungsi Sigmoid yang telah diskalakan dan digeser sehingga berpusat di nol (*zero-centered*). Sifat *zero-centered* ini sering kali membuat konvergensi model lebih cepat dibandingkan Sigmoid [26]. Persamaan matematis fungsi Tanh diberikan pada Persamaan 2.7.

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = 2\sigma(2x) - 1 \tag{2.7}$$

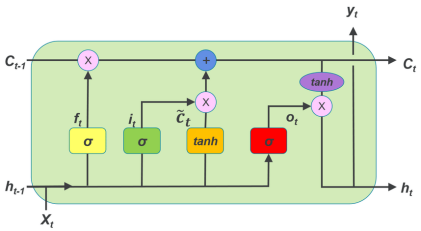
Keterangan:
$\tanh(x)$: *Output fungsi* Tanh,
$\sigma(x)$: Fungsi Sigmoid.
Gambar 2.3 memperlihatkan bentuk kurva fungsi Tanh, yang simetris terhadap titik origin dan memungkinkan gradien yang lebih seimbang selama proses *backpropagation*, terutama pada lapisan-lapisan tersembunyi dari jaringan saraf.



Gambar 2.3 Kurva Fungsi Aktivasi Tanh

2.9 *Long Short-Term Memory* (LSTM)

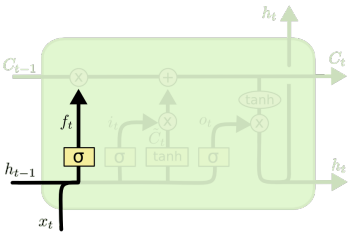
Long Short-Term Memory (LSTM) adalah arsitektur dalam *Recurrent Neural Network* (RNN) yang dikembangkan oleh Hochreiter dan Schmidhuber pada 1997 untuk mengatasi masalah *vanishing gradient* dalam RNN klasik [27]. LSTM dirancang untuk memproses data sekuensial dengan memanfaatkan *cell state* dan sejumlah *gate* sebagai pengendali aliran informasi [28]. Struktur internal sel LSTM ditunjukkan pada Gambar 2.4.



Gambar 2.4 Struktur sel LSTM

Masing-masing komponen utama dalam sel LSTM dijelaskan sebagai berikut [12].

2.9.1 *Forget Gate*



Gambar 2.5 *Forget Gate*

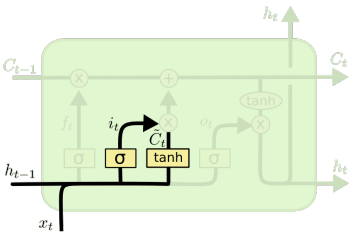
Gambar 2.5 menunjukkan bagaimana *Forget Gate* menentukan informasi dari memori sebelumnya (C_{t-1}) yang perlu dilupakan. Nilai *gate* ini dihitung dengan fungsi sigmoid terhadap gabungan *input* saat ini (x_t) dan *hidden state* sebelumnya (h_{t-1}), sebagaimana ditunjukkan pada Persamaan 2.8.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$

Keterangan:

f_t : Nilai *Forget Gate* pada waktu t,
x_t : *Input* pada waktu t,
h_{t-1} : *Hidden state* sebelumnya,
W_f : Bobot *input Forget Gate*,
b_f : Bias *Forget Gate*,
σ : Fungsi aktivasi sigmoid.

2.9.2 *Input Gate*



Gambar 2.6 *Input Gate*

Gambar 2.6 menggambarkan bagaimana *Input Gate* menentukan informasi baru yang akan disimpan dalam memori sel. Proses ini terdiri dari dua tahap: menghitung nilai *gate* i_t dan menghasilkan kandidat nilai memori $\tilde{C}_t$, ditunjukkan pada Persamaan 2.9 dan 2.10.

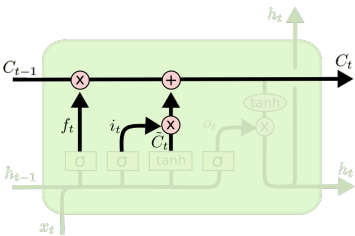
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \tag{2.10}$$

Keterangan:

i_t : Nilai *Input Gate* pada waktu t,
$\tilde{C}_t$: Kandidat nilai memori baru,
W_i, W_c : Bobot untuk *Input Gate* dan kandidat memori,
b_i, b_c : Bias untuk masing-masing,
tanh : Fungsi aktivasi *Hyperbolic Tangent*.

2.9.3 *Cell State Update*



Gambar 2.7 *cell state*

Gambar 2.7 memperlihatkan bagaimana memori sel diperbarui dengan menggabungkan informasi lama dan baru melalui hasil dari *Forget Gate* dan *Input Gate*. Hal ini dirumuskan pada Persamaan 2.11.

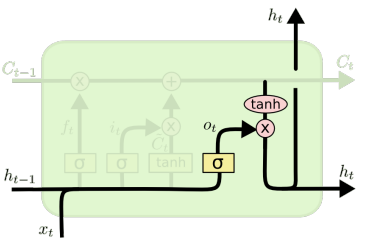
$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{2.11}$$

Keterangan:

C_t : Nilai memori sel pada waktu t,

$\odot$: Perkalian elemen-per-elemen.

2.9.4 *Output Gate*



Gambar 2.8 *Output Gate*

Gambar 2.8 menunjukkan bagaimana *Output Gate* menentukan informasi apa yang akan dikeluarkan dari sel memori sebagai *hidden state* (h_t). Proses ini melibatkan dua tahap, yaitu aktivasi dari *gate* o_t dan pembobotan terhadap memori terkini C_t menggunakan fungsi aktivasi *tanh*. Persamaan matematisnya ditunjukkan pada Persamaan 2.12 dan 2.13.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.12}$$

$$h_t = o_t \odot \tanh(C_t) \tag{2.13}$$

Keterangan:

o_t : Nilai *Output Gate* pada waktu t,

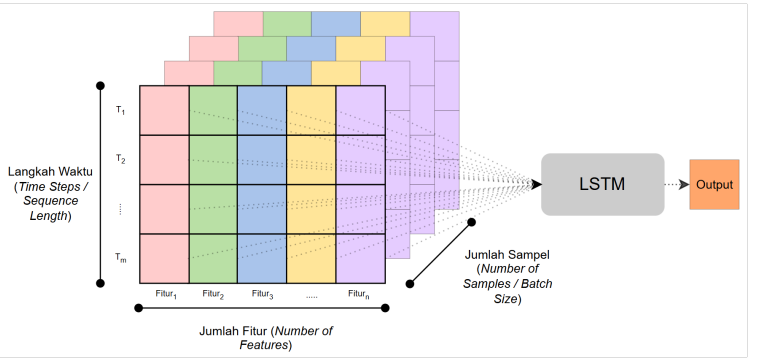
h_t : *Hidden State* pada waktu t,

W_o, b_o : Bobot dan bias untuk *Output Gate*.

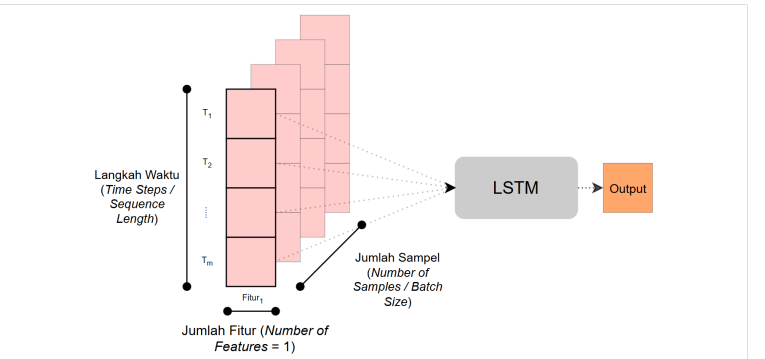
2.10 *Multivariate Long Short-Term Memory* (*Multivariate* LSTM)

Multivariate LSTM merupakan perluasan dari arsitektur LSTM yang dirancang untuk memproses data deret waktu dengan lebih dari satu fitur pada setiap langkah waktu. Dengan menerima vektor *input* berdimensi lebih dari satu ($n > 1$), arsitektur ini mampu menangkap

pola temporal serta interaksi antar fitur secara simultan [29]. Struktur internal selnya identik dengan LSTM standar, namun perbedaannya terletak pada bentuk tensor *input* yang digunakan [30]. Ilustrasi perbandingan antara *input Multivariate* dan *Univariate* LSTM ditunjukkan pada Gambar 2.9 dan 2.10.



Gambar 2.9 *Input Multivariate* LSTM (*many to one*)



Gambar 2.10 *Input Univariate* LSTM (*many to one*)

2.11 Hyperband *Hyperparameter Tuning*

Beberapa *hyperparameter* penting yang umum disetel dalam model *Long Short-Term Memory* (LSTM) meliputi *sequence length*, jumlah lapisan LSTM, jumlah neuron per lapisan, *dropout rate*, *kernel regularization*, jenis *optimizer*, *learning rate*, *batch size*, dan jumlah *epoch* maksimum. Penyetelan nilai-nilai tersebut berperan besar dalam menentukan kapasitas memori jangka panjang, kemampuan generalisasi, serta efisiensi pembelajaran model. Untuk

mengoptimalkan kombinasi parameter ini secara efisien, salah satu metode yang dapat digunakan adalah Hyperband, yakni algoritma yang menggabungkan pemilihan konfigurasi secara acak dan eliminasi bertahap berdasarkan kinerja awal melalui mekanisme *successive halving* [31].

Algorithm 1: *Pseudocode* Algoritma Hyperband

Input: R (sumber daya maksimum), η (faktor eliminasi, default: 3)

$s_{max} \leftarrow \lfloor \log_{\eta}(R) \rfloor, \quad B \leftarrow (s_{max} + 1) \cdot R;$

for $s = s_{max}, s_{max} - 1, ..., 0$ **do**

  $n \leftarrow \lceil \frac{B}{R} \cdot \frac{\eta^s}{s+1} \rceil;$

  $r \leftarrow R \cdot \eta^{-s};$

  $T \leftarrow$ Ambil n konfigurasi *hyperparameter* acak;

  for $i = 0$ *to* s **do**

    $n_i \leftarrow \lfloor n \cdot \eta^{-i} \rfloor;$

    $r_i \leftarrow r \cdot \eta^{i};$

    Latih semua n_i konfigurasi selama r_i sumber daya;

    Pilih $\lfloor n_i/\eta \rfloor$ konfigurasi terbaik untuk iterasi berikutnya;

  end

end

return konfigurasi dengan *validation loss* terendah;

Keterangan:

R	: Total sumber daya maksimum (misalnya, jumlah *epoch*),
η	: Faktor eliminasi konfigurasi (default = 3),
s	: Indeks *bracket*,
B	: Total anggaran sumber daya dalam satu *bracket*,
n	: Konfigurasi awal pada *bracket* ke-s,
r	: Sumber daya awal per konfigurasi,
n_i	: Jumlah konfigurasi pada iterasi-i,
r_i	: Sumber daya per konfigurasi pada iterasi-i.

2.12 Evaluasi Kinerja Model

2.12.1 *Mean Absolute Percentage Error* (MAPE)

Evaluasi kinerja dalam studi prediksi dilakukan untuk mengukur tingkat akurasi model dalam merepresentasikan pola data aktual. Salah satu metrik evaluasi yang paling umum digunakan adalah *Mean Absolute Percentage Error* (MAPE), karena memberikan interpretasi kesalahan dalam bentuk persentase yang bersifat skala-independen dan mudah dipahami. Model dengan nilai MAPE di bawah 10% umumnya dianggap memiliki akurasi yang sangat baik [32]. Rumus perhitungan MAPE ditunjukkan pada Persamaan 2.14:

$$\text{MAPE} = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{Y_t - \hat{Y}_t}{Y_t} \right| \times 100\% \tag{2.14}$$

Keterangan:

MAPE : Persentase kesalahan rata-rata,
n : Jumlah observasi,
Y_t : Nilai aktual,
$\hat{Y}_t$: Nilai prediksi.