BAB II

LANDASAN TEORI

2.1 Sampah dan Dampak Emisi Gas Metana

Sampah menjadi permasalahan lingkungan yang kompleks dengan dampak signifikan terhadap peningkatan emisi gas rumah kaca, terutama gas metana (CH_4). Gas ini terbentuk melalui proses dekomposisi anaerobik sampah organik di tempat pembuangan akhir (TPA) dan memiliki potensi pemanasan global jauh lebih besar dibandingkan karbon dioksida (CO_2). Emisi dari TPA menyumbang 15–18% dari total emisi metana, sehingga pengelolaan sampah berperan penting dalam upaya mitigasi perubahan iklim (Ferrari dkk., 2024).

2.2 Keterkaitan dengan *Sustainable Development Goals* (SDGs)

Sustainable Development Goals (SDGs) terdiri atas 17 tujuan global yang dicanangkan oleh Perserikatan Bangsa-Bangsa (PBB) dalam Agenda 2030 untuk pembangunan berkelanjutan (Senger, 2017). Penelitian ini memiliki keterkaitan dengan dua tujuan utama, yaitu SDG 11: Kota dan Pemukiman yang Berkelanjutan, serta SDG 13: Penanganan Perubahan Iklim.

SDG 11 menekankan pentingnya pengurangan dampak lingkungan di wilayah perkotaan, termasuk pengelolaan sampah yang aman dan berkelanjutan. Prediksi jumlah sampah yang akurat memungkinkan penelitian ini berkontribusi pada perencanaan sistem pengelolaan sampah yang lebih adaptif terhadap pertumbuhan kota (Ntemngweh, 2023).

SDG 13 mendorong tindakan terhadap perubahan iklim. Prediksi jumlah sampah dan estimasi emisi gas metana yang dihasilkan memberikan data penting untuk mendukung kebijakan pengurangan emisi gas rumah kaca. Hasil dari penelitian ini dapat digunakan sebagai dasar pengambilan keputusan dalam strategi mitigasi perubahan iklim di tingkat lokal maupun nasional (Fernandes, 2024).

2.3 *Time series Forecasting* dalam Prediksi Data

Time series forecasting digunakan untuk memprediksi nilai di masa depan berdasarkan pola data historis (Sushanth dkk., 2024). Metode ini diterapkan dalam berbagai bidang seperti ekonomi, cuaca, kesehatan, dan lingkungan untuk memahami tren serta mendukung pengambilan keputusan yang lebih akurat. Secara umum, *time series forecasting* terbagi menjadi tiga kategori utama, yaitu metode statistik, metode *machine learning*, dan metode *deep learning* (Spiliotis, 2023).

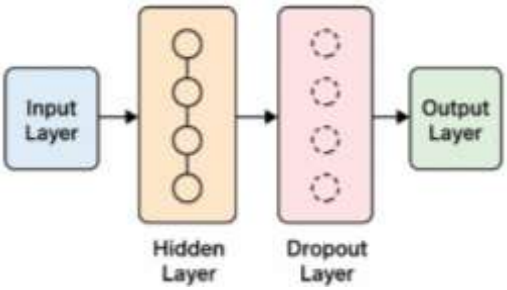
Time series forecasting sangat bermanfaat dalam konteks prediksi jumlah sampah karena mampu menangkap pola pertumbuhan dan tren jangka panjang yang muncul akibat peningkatan aktivitas manusia dan urbanisasi. *Long Short-Term Memory* (LSTM) merupakan salah satu pendekatan *deep learning* yang paling umum digunakan untuk prediksi deret waktu.

2.4 *Long Short-Term Memory* (LSTM)

Long Short-Term Memory (LSTM) termasuk jenis *Recurrent Neural Network* (RNN) yang dirancang untuk mengatasi permasalahan *long-term dependency* pada data deret waktu. Arsitektur LSTM terdiri atas sel memori dengan tiga gerbang utama (*input*, *forget*, dan *output gate*) yang memungkinkan model mempertahankan informasi penting dalam jangka waktu panjang (Fukai dkk., 2024). Beberapa varian

LSTM, seperti *Bidirectional* LSTM (BiLSTM) dan *Enhanced* LSTM (e-LSTM), dikembangkan untuk meningkatkan akurasi prediksi serta menangkap pola yang lebih kompleks (Pokkuluri & Khang, 2025).

Model *Long Short-Term Memory* (LSTM) dirancang menggunakan arsitektur berlapis (*layer*) yang bertujuan untuk menangkap pola kompleks dalam data *time series*, gambaran arsitektur (*layer*) model LSTM dapat dilihat pada Gambar 2.1.



Gambar 2. 1 LSTM *Layer*

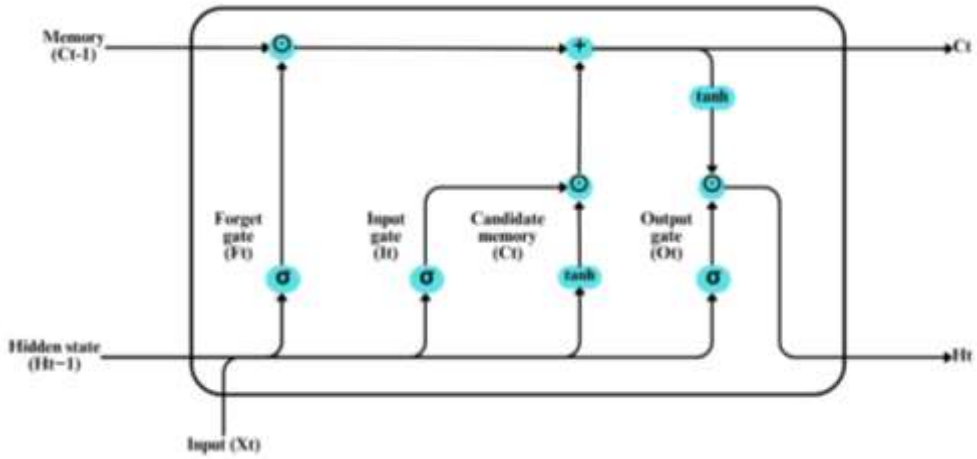
2.4.1 *Input Layer*

Input layer memproses data *time series* hasil *preprocessing*, yang telah melalui tahapan penanganan *missing values*, normalisasi, deteksi *outlier*, dan *sliding window*. Lapisan ini bertanggung jawab untuk menginterpretasikan urutan data dan meneruskannya ke *hidden layer* (Kartini dkk., 2021). Tahap ini memproses input berupa urutan data X_t yang dikombinasikan dengan *hidden state* sebelumnya (H_{t-1}) untuk diproses lebih lanjut.

2.4.2 *Hidden Layer*

Terdiri dari beberapa lapisan LSTM dengan jumlah unit (*neurons*) yang disesuaikan berdasarkan kompleksitas pola data. Lapisan ini menganalisis ketergantungan temporal dan menyimpan informasi jangka panjang maupun

jangka pendek menggunakan tiga *gate* utama (Muhammad dkk., 2024) dengan arsitektur yang digambarkan pada Gambar 2.2.



Gambar 2. 2 Arsitektur LSTM *cell/unit*

a. *Forget Gate* (F_t) untuk memutuskan informasi mana dari *cell state* sebelumnya (C_{t-1}) yang perlu dilupakan (Fanta dkk., 2021). Nilai F_t dihitung dengan rumus pada Persamaan 2.1.

$$F_t = \sigma\left(W_f . [H_{t-1}, X_t] + b_f\right) \quad (2.1)$$

Forget Gate menggunakan fungsi sigmoid (σ) untuk menghasilkan nilai antara 0 (lupakan sepenuhnya) dan 1 (pertahankan sepenuhnya). *Gate* ini memproses gabungan *hidden state* sebelumnya (H_{t-1}) dan *input* saat ini (X_t) melalui matriks bobot (W_f) dan menambahkan bias (b_f). Hasil ini menentukan proporsi informasi dari *cell state* sebelumnya yang akan dipertahankan.

b. *Input Gate* (I_t) untuk menentukan informasi baru yang akan ditambahkan ke *cell state* (Sen & Raghunathan, 2018). Dihitung dengan rumus pada Persamaan 2.2 serta rumus kandidat memori baru pada Persamaan 2.3.

$$I_t = \sigma(W_i.[H_{t-1}, X_t] + b_i) \quad (2.2)$$

Input Gate menentukan nilai mana yang akan diperbarui menggunakan fungsi sigmoid. Sama seperti *Forget Gate*, *gate* ini memproses H_{t-1} dan X_t melalui matriks bobot W_i dan bias b_i untuk menghasilkan nilai antara 0 dan 1 yang menentukan seberapa besar informasi baru akan ditambahkan.

$$\hat{C}_t = tanh(W_c.[H_{t-1}, X_t] + b_c) \quad (2.3)$$

Cell state kandidat ($\hat{C}_t$) adalah informasi baru yang potensial untuk ditambahkan ke *cell state*. Fungsi $tanh$ menghasilkan nilai antara -1 dan 1, memberikan bobot pada informasi baru. Kandidat ini kemudian akan difilter oleh *Input Gate* (I_t) untuk menentukan berapa banyak dari informasi baru ini yang benar-benar akan ditambahkan ke *cell state*.

c. *Output gate* (O_t) untuk mengontrol informasi apa yang akan keluar sebagai *hidden state* baru H_t (Cheng dkk., 2020). Rumusnya ada pada Persamaan 2.4.

$$O_t = \sigma(W_o.[H_{t-1}, X_t] + b_o) \quad (2.4)$$

Output Gate menentukan bagian mana dari *cell state* yang akan dikeluarkan sebagai *output* (*hidden state*). Menggunakan fungsi sigmoid untuk menghasilkan nilai antara 0 dan 1, *gate* ini memproses H_{t-1} dan X_t melalui matriks bobot W_o dan bias b_o. Hasil ini akan mengontrol aliran informasi dari *cell state* ke *output*.

Cell state diperbarui setiap waktu t dengan Persamaan 2.5.

$$C_t = F_t \times C_{t-1} + I_t \times \hat{C}_t \quad (2.5)$$

Pembaruan *cell state* terjadi melalui dua proses yang saling melengkapi. *Forget Gate* (F_t) mengalikan *cell state* sebelumnya (C_{t-1}) untuk menghapus informasi yang tidak relevan (operasi $F_t \times C_{t-1}$), sedangkan *Input Gate* (I_t) mengalikan *cell state* kandidat ($\hat{C}_t$) untuk menambahkan informasi baru (operasi $I_t \times \hat{C}_t$). Kedua hasil tersebut kemudian dijumlahkan untuk menghasilkan *cell state* yang diperbarui. Mekanisme ini memungkinkan LSTM mempertahankan informasi penting jangka panjang sambil tetap memasukkan informasi baru yang relevan.

Dan *hidden state baru* dihitung menggunakan rumus pada Persamaan 2.6.

$$H_t = O_t \times tanh(C_t) \tag{2.6}$$

Hidden state (H_t) adalah *output* akhir dari LSTM *cell* yang akan diteruskan ke *layer* berikutnya atau menjadi *output* model. *Hidden state* dihitung dengan mengalikan *Output Gate* (O_t) dengan fungsi $tanh$ dari *cell state* yang telah diperbarui (C_t). Fungsi $tanh$ menormalisasi nilai *cell state* ke rentang -1 hingga 1, kemudian *Output Gate* memfilter informasi mana yang akan dikeluarkan. *Hidden state* ini mengandung informasi yang relevan dari seluruh *sequence* yang telah diproses hingga *timestep* saat ini.

Berikut adalah penjelasan lengkap untuk setiap notasi yang digunakan dalam persamaan-persamaan LSTM.

F_t: *Output* dari *Forget Gate* pada *timestep* t (nilai antara 0 dan 1 yang menentukan informasi mana yang akan dilupakan)

I_t: *Output* dari *Input Gate* pada *timestep* t (nilai antara 0 dan 1 yang menentukan informasi baru yang akan disimpan)

O_t: *Output* dari *Output Gate* pada *timestep* t (nilai antara 0 dan 1 yang menentukan informasi yang akan dikeluarkan)

σ: Fungsi aktivasi sigmoid yang menghasilkan nilai antara 0 dan 1

$tanh$: Fungsi aktivasi tangent hiperbolik yang menghasilkan nilai antara -1 dan 1

W_f, W_i, W_o, W_c: Matriks bobot (*weight matrices*) untuk *Forget Gate*, *Input Gate*, *Output Gate*, dan *cell state* kandidat

H_{t-1}: *Hidden state* dari *timestep* sebelumnya (*output* dari LSTM *cell* sebelumnya)

X_t: *Input* data pada *timestep* saat ini

$[H_{t-1}, X_t]$: Konkatenasi (penggabungan) antara *hidden state* sebelumnya dan *input* saat ini

bf, bi, bo, bc: Bias untuk *Forget Gate*, *Input Gate*, *Output Gate*, dan *cell state* kandidat

$\hat{C}_t$: *Cell state* kandidat (calon informasi baru yang akan ditambahkan ke *cell state*)

C_{t-1}: *Cell state* dari *timestep* sebelumnya (memori jangka panjang)

C_t: *Cell state* yang diperbarui pada *timestep* saat ini

H_t: *Hidden state* baru pada *timestep* saat ini (*output* dari LSTM *cell* yang akan diteruskan ke *layer* berikutnya)

Ketiga *gate* dalam LSTM, *Forget*, *Input*, dan *Output*, bekerja secara bersamaan untuk mengelola aliran informasi dengan cara mengatur *cell state* melalui mekanisme penghapusan dan penambahan informasi yang relevan,

sekaligus mencegah terjadinya *vanishing* atau *exploding gradient*. Sinergi antar *gate* ini memungkinkan model mempelajari dependensi jangka panjang pada data *time series* serta menghasilkan *output* yang lebih informatif melalui pembaruan *hidden state*.

2.4.3 *Dropout Layer*

Diimplementasikan untuk mengurangi risiko *overfitting* dengan secara acak menonaktifkan sejumlah unit selama proses pelatihan. *Dropout layer* membantu memastikan model tidak terlalu bergantung pada pola tertentu dalam data latih dan meningkatkan kemampuan generalisasi (Pansambal & Nandgaokar, 2023).

2.4.4 *Output Layer*

Menghasilkan prediksi jumlah sampah berdasarkan pola yang telah dipelajari oleh lapisan tersembunyi. Biasanya lapisan ini menggunakan fungsi aktivasi linear untuk menghasilkan keluaran berupa nilai kontinu (Charoenthamanont dkk., 2023), seperti prediksi jumlah sampah atau emisi gas metana.

2.5 *Optimizer* dalam Model *Deep Learning*

Pemilihan *optimizer* sangat penting karena memengaruhi kecepatan konvergensi dan kemampuan model menemukan solusi optimal. Secara umum, *optimizer* bekerja dengan memperbarui bobot jaringan saraf untuk meminimalkan

nilai *loss* selama proses pelatihan. Tabel 2.1 menampilkan beberapa *optimizer* yang umum digunakan dalam pengembangan model *deep learning*.

Tabel 2. 1 *Optimizer* dalam *Deep Learning*

Optimizer	**Deskripsi**
RMSprop (*Root Mean Square Propagation*)	Mengatur *Learning Rate* adaptif per parameter dengan rata-rata kuadrat gradien, sehingga lebih stabil pada gradien fluktuatif dan cocok untuk data *non-stationary* (Mukkamala & Hein, 2017).
Adam	Menggabungkan Momentum dan RMSprop dengan momen pertama dan kedua, membuatnya stabil di banyak arsitektur dan umum menjadi *optimizer default* (Jin dkk., 2024).
RAdam (*Rectified Adam*)	Varian Adam yang menstabilkan awal pelatihan dengan koreksi varians adaptif, sehingga mengurangi *over-adaptiveness* dan membantu mencegah *overshooting* (Chakrabarti & Chopra, 2022).

RMSprop dipilih sebagai *optimizer* utama karena kemampuannya menyesuaikan *learning rate* secara adaptif terhadap gradien yang berfluktuasi, kondisi yang umum terjadi pada data deret waktu. *Optimizer* ini memiliki beberapa keunggulan, antara lain kemampuan menyesuaikan pembaruan bobot berdasarkan rata-rata kuadrat gradien sebelumnya (*adaptive learning rate*), stabilitas dalam menangani variasi musiman pada data *time series*, efisiensi konvergensi yang mempercepat proses pembelajaran tanpa mengorbankan stabilitas, serta keseimbangan generalisasi yang lebih baik karena tidak seagresif Adam sehingga mampu mengurangi risiko *overfitting* (Mukkamala & Hein, 2017).

Secara matematis, pembaruan RMSprop dinyatakan dengan rumus pada Persamaan 2.7 dan 2.8 , dengan $\boldsymbol{\alpha}$ = 0.001 sebagai *Learning Rate* dan $\boldsymbol{\epsilon}$ konstanta kecil untuk stabilitas numerik (Yun, 2023).

$$E[g^2]_t = 0.9E[g^2]_{t-1} + 0.1g_t^2 \tag{2.7}$$

$$\theta_{t+1} = \theta_t - \frac{\alpha}{\sqrt{E[g^2]_t + \epsilon}} \cdot g_t \tag{2.8}$$

2.6 Regularisasi

Regularisasi berperan mencegah *overfitting,* kondisi ketika model terlalu menyesuaikan diri pada data pelatihan sehingga kehilangan kemampuan generalisasi. Model LSTM menggunakan teknik regularisasi berupa *dropout* sebagai salah satu metode untuk meningkatkan kemampuan generalisasi. *Dropout* bekerja dengan cara menonaktifkan secara acak sejumlah unit neuron selama proses pelatihan, sehingga jaringan tidak terlalu bergantung pada fitur tertentu (Diukarev & Starukhin, 2024).

2.7 *Grid Search* untuk *Tuning Hyperparameter*

Grid Search berfungsi sebagai metode pencarian *hyperparameter* secara sistematis dengan mencoba seluruh kombinasi parameter yang telah ditentukan (Anggreani, 2024). Metode ini sangat efektif untuk menemukan konfigurasi terbaik dalam model LSTM, seperti jumlah unit neuron, *dropout rate*, *batch size*, dan *learning rate,* meskipun memerlukan komputasi yang intensif. Teknik ini memungkinkan eksplorasi parameter secara menyeluruh untuk mencapai performa model terbaik (Khullar, 2024).

2.8 Evaluasi Kinerja Model Prediksi

2.8.1 *Mean Absolute Error* (MAE)

Mean Absolute Error (MAE) digunakan sebagai metrik evaluasi untuk mengukur rata-rata perbedaan absolut antara nilai prediksi dan nilai aktual (Dumre dkk., 2024). MAE dihitung dengan membagi total nilai absolut dari selisih antara nilai aktual y_i dan nilai prediksi $\hat{y}_i$ oleh jumlah data n, dengan rumus pada Persamaan 2.9.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.9}$$

MAE memberikan gambaran seberapa jauh prediksi model menyimpang dari data sebenarnya, tanpa memperhitungkan arah *error*, karena semua selisih dihitung dalam nilai absolut. Semakin kecil nilai MAE, semakin akurat model dalam melakukan prediksi. Metrik ini bersifat linear, yang berarti semua *error* memiliki bobot yang sama, sehingga perbedaan kecil maupun besar berkontribusi secara proporsional terhadap hasil akhir (Hosamo & Mazzetto, 2025).

2.8.2 *Mean Squared Error* (MSE)

Mean Squared Error (MSE) digunakan sebagai metrik evaluasi untuk mengukur rata-rata kuadrat selisih antara nilai aktual dan nilai prediksi (Rauf dkk., 2024). MSE dihitung dengan membagi total kuadrat selisih antara nilai aktual y_i dan nilai prediksi $\hat{y}_i$ oleh jumlah data n, menggunakan rumus pada Persamaan 2.10.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.10}$$

Metrik ini tidak hanya menunjukkan seberapa jauh prediksi model dari data sebenarnya, tetapi juga memberikan bobot lebih besar pada *error* yang besar, karena selisih dikuadratkan. Hal ini membuat MSE lebih sensitif terhadap *outlier* dibandingkan MAE. Semakin kecil nilai MSE, semakin baik performa model, sedangkan nilai MSE yang tinggi menunjukkan adanya prediksi yang jauh meleset dari nilai aktual (Dumre dkk., 2024).

2.8.3 *Root Mean Squared Error* (RMSE)

Root Mean Squared Error (RMSE) dihitung sebagai akar kuadrat dari Mean Squared *Error* (MSE) dan bertujuan mengembalikan skala *error* ke unit yang sama dengan data aslinya (Achmadin dkk., 2024). RMSE dihitung menggunakan rumus pada Persamaan 2.11.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{2.11}$$

RMSE memberikan gambaran seberapa besar rata-rata *error* antara nilai aktual y_i dan nilai prediksi $\hat{y}_i$, dengan mempertimbangkan bobot yang lebih besar untuk *error* yang besar, karena selisih dikuadratkan sebelum diakarkan. Metrik ini berguna ketika kita ingin memahami besarnya *error* dalam satuan asli data, memudahkan interpretasi hasil, dan tetap sensitif terhadap *outlier*, seperti MSE. Semakin kecil nilai RMSE, semakin baik performa model, sedangkan RMSE yang tinggi menunjukkan adanya prediksi yang meleset secara signifikan dari nilai aktual (Dumre dkk., 2024).

2.8.4 *R-Squared* (R^2)

R-Squared (R^2) atau Koefisien Determinasi digunakan sebagai metrik evaluasi untuk mengukur seberapa baik model mampu menjelaskan variansi dalam data aktual (Gupta dkk., 2024). R^2 menunjukkan proporsi variansi dalam variabel dependen y yang dapat dijelaskan oleh variabel independen dalam model. Nilainya berkisar antara 0 hingga 1, dan dihitung menggunakan rumus pada Persamaan 2.12.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \quad (2.12)$$

Di sini, y_i adalah nilai aktual, $\hat{y}_i$ adalah nilai prediksi, dan $\bar{y}$ adalah rata-rata dari nilai aktual. Interpretasi R^2 cukup intuitif:

a. $R^2 = 1$ berarti model mampu menjelaskan 100% variansi data, atau model memiliki performa sempurna.
b. $R^2 = 0$ menunjukkan bahwa model sama sekali tidak mampu menjelaskan variansi dalam data.
c. R^2 negatif bisa terjadi jika model memiliki performa lebih buruk daripada menggunakan rata-rata nilai sebagai prediksi (*underfitting*).

Semakin tinggi nilai R^2, semakin baik model menjelaskan hubungan antara variabel, namun penting untuk memperhatikan bahwa R^2 saja tidak cukup untuk mengevaluasi model secara keseluruhan, terutama jika ada *overfitting* atau data memiliki *outlier* (Gao, 2024). R^2 biasanya dikombinasikan dengan metrik lain seperti RMSE atau MAE untuk analisis yang lebih akurat.

2.9 Konversi Prediksi Jumlah Sampah ke Emisi Gas Metana

Berbagai penelitian telah dilakukan terkait prediksi jumlah sampah, namun sebagian besar studi cenderung berfokus pada estimasi kuantitas sampah semata tanpa mengintegrasikan hasil prediksi tersebut ke dalam perhitungan dampak lingkungan, khususnya emisi gas metana. Analisis yang dihadirkan umumnya hanya mengevaluasi akurasi model prediksi jumlah sampah, tanpa mengeksplorasi pemanfaatan data tersebut untuk memproyeksikan konsekuensi ekologis.

Penelitian ini mengadopsi pendekatan konversi emisi gas metana yang didasarkan pada faktor emisi dari *Intergovernmental Panel on Climate Change* (IPCC) dan telah divalidasi dalam konteks Indonesia oleh (Rarastry, 2016) dalam bukunya Kontribusi Sampah Terhadap Pemanasan Global. Menurut pedoman IPCC 2006 untuk inventarisasi gas rumah kaca nasional, timbulan sampah organik yang mengalami dekomposisi anaerobik di TPA menghasilkan gas metana dengan rasio konversi spesifik. (Rarastry, 2016) menerapkan faktor konversi 1 ton sampah menghasilkan 50 kg gas metana (CH_4) berdasarkan karakteristik sampah Indonesia dengan komposisi organik yang dominan (sekitar 60-70%) dan kondisi TPA yang umumnya menggunakan sistem *open dumping* dengan kondisi anaerobik. Formula konversi ini dinyatakan dalam Persamaan 2.13.

$$1\ ton\ sampah\ =\ 50\ kg\ gas\ metana \tag{2.13}$$

Data Kementerian Lingkungan Hidup mengatakan bahwa pada tahun 1995 rata-rata orang di perkotaan di Indonesia menghasilkan sampah 0,8 kg per hari dan terus meningkat hingga 1 kg per orang per hari pada tahun 2000. Diperkirakan timbunan sampah pada tahun 2020 untuk tiap orang per hari adalah sebesar 2,1 kg. Sampah sendiri turut menghasilkan emisi GRK berupa gas metana, walaupun dalam jumlah yang cukup kecil dibandingkan emisi GRK yang dihasilkan dari sektor kehutanan dan energi. Diperkirakan 1 ton sampah padat menghasilkan sekitar 50 kg gas metana (Rarastry, 2016).

Formula Persamaan 2.13 telah digunakan dalam berbagai studi emisi gas rumah kaca dari sektor persampahan di Indonesia dan konsisten dengan Pedoman Umum Penyelenggaraan Inventarisasi Gas Rumah Kaca Nasional yang dikeluarkan oleh

Kementerian Lingkungan Hidup dan Kehutanan. Pendekatan yang disederhanakan ini tetap mampu memberikan estimasi yang cukup akurat untuk perencanaan st*Rate*gis pengelolaan sampah serta kebijakan mitigasi emisi di tingkat nasional dan daerah.

Pendekatan ini tidak hanya memberikan dasar perhitungan awal, tetapi juga membuka peluang bagi pengembangan model yang lebih kompleks di masa depan. Model tersebut dapat mengakomodasi berbagai variabel lain, seperti jenis sampah, laju dekomposisi, tingkat kelembaban, suhu, pH, serta kondisi lingkungan TPA (ada/tidaknya sistem penangkapan gas), sehingga menghasilkan proyeksi emisi gas rumah kaca yang lebih akurat dan komprehensif untuk mendukung kebijakan mitigasi perubahan iklim yang efektif.

2.10 *State of The Art*

Penelitian terkini mengenai prediksi data *time series*, khususnya menggunakan metode LSTM, menunjukkan berbagai upaya untuk meningkatkan akurasi prediksi dalam berbagai bidang seperti volume sampah, kualitas udara, dan produksi susu. *State of The Art* yang disajikan pada Tabel 2.2 memperlihatkan berbagai metode yang telah diterapkan untuk memprediksi jumlah sampah, termasuk normalisasi data, *sliding window*, dan perbandingan model seperti LSTM, BiLSTM, GRU, dan *Prophet Model*.

Tabel 2. 2 *State of The Art*

Author	***Research Topic***	***Methodology***	***Findings***	***Limitations***
(Loni Manikari dkk., 2024)	Prediksi volume sampah Cirebon menggunakan LSTM berdasarkan data 2021-2023.	LSTM. Normalisasi data dan teknik *sliding window*.	Normalisasi dan *sliding window* menghasilkan MSE LSTM sebesar 0,02207.	Jumlah lapisan LSTM mempengaruhi variasi dan performa model.
(Santoso dkk., 2023)	Prediksi volume sampah TPSA Banyuurip dengan *Backpropagation Neural Network*.	*Backpropagation Neural Network*.	Nilai MSE terbaik: 0.018870.	Data, parameter, belum ada validasi *real-time*, dan ketidakmampuan memprediksi beberapa *outlier*.
(Karyadi & Santoso, 2022)	Prediksi parameter kualitas udara (suhu, kelembaban, PM10, ISPU) menggunakan LSTM, BiLSTM, dan GRU pada data *time series*.	LSTM, BiLSTM, dan GRU.	Hasil prediksi LSTM dan BiLSTM lebih baik dibandingkan GRU.	Keterbatasan utama meliputi data, parameter, validasi *real-time*, dan kinerja GRU.
(Primawati dkk., 2023)	Prediksi produksi susu sapi perah dengan LSTM dan *Prophet*.	LSTM dan *Prophet Model*.	LSTM mengungguli *Prophet* pada data terbatas, meski keduanya meningkatkan R^2 (0.2 ke 0.3).	Kinerja LSTM dan *Prophet* tidak signifikan berbeda pada data terbatas.

Author	***Research Topic***	***Methodology***	***Findings***	***Limitations***
(Almira dkk., 2024)	Prediksi variabel menggunakan Regresi Linier, SVR, dan *Random Forest*.	Regresi Linier, *Support Vector Regression*, dan *Random Forest Regressor*.	Dalam hal akurasi (MSE/RMSE), *Random Forest unggul*; SVR optimal pada *hyperparameter*.	Data, parameter, validasi *real-time*, dan kinerja model pada data multi-periodik yang tidak optimal.
(Simbolon dkk., 2023)	Prediksi timbulan sampah dengan pendekatan kuantitatif deskriptif di Tanjungpinang lima tahun ke depan.	Observasi dan studi literatur.	Prediksi jumlah penduduk tahun 2027: 274.883 jiwa. Timbulan sampah diperkirakan 29.283 ton/tahun.	Rentang waktu prediksi terbatas.
(Rong dkk., 2022)	Pengoptimalan LSTM untuk prediksi deret waktu, dengan fokus khusus pada penggunaan listrik rumah tangga.	LSTM, ASSOMA, dan SOMA.	ASSOMA mengoptimalkan struktur LSTM dan *hyperparameter* secara efektif.	Tantangan LSTM dalam menentukan struktur jaringan dan *hyperparameter*.
(Yadav dkk., 2023)	Penggunaan jaringan LSTM untuk *time series forecasting*, dengan fokus pada kemampuannya dalam mempelajari *long-term dependencies*.	*Peephole connections*. Mengimplementasikan *early stop* untuk jaringan LSTM.	LSTM secara efektif meramalkan data *time series*. LSTM mempelajari *long-term dependencies* yang kompleks.	-
(Dey & Chatterjee, 2024)	Prediksi tingkat timbulan sampah kota dengan LSTM dan *Prophet model*.	LSTM, *Prophet model*, dan SARIMAX.	Model *Prophet* mengungguli LSTM dan SARIMAX dalam hal metrik kesalahan.	Dataset terbatas. Pengabaian efek musiman dan hari libur.
(Voipan dkk., 2024)	Peningkatan prediksi pengolahan air limbah menggunakan LSTM.	LSTM *networks* di BSM2. Analisis simulasi historis.	LSTM meningkatkan prediksi proses pengolahan air limbah.	-

State of The Art yang disajikan memperlihatkan bahwa meskipun LSTM secara umum memberikan hasil yang baik, beberapa studi membandingkannya dengan model lain seperti *Prophet* dan SARIMAX untuk mengatasi keterbatasan tertentu, seperti data yang terbatas dan validasi *real-time* yang belum optimal. Keterbatasan umum yang ditemukan meliputi pengaruh jumlah lapisan LSTM terhadap performa model, keterbatasan parameter, dan kesulitan dalam memprediksi *outlier*. Berdasarkan penelitian-penelitian yang telah diulas pada Tabel 2.2, penelitian ini mengusulkan implementasi model LSTM untuk memprediksi emisi gas metana dari jumlah sampah. Model ini akan mengintegrasikan pendekatan *time series forecasting* dengan pengembangan parameter guna meningkatkan akurasi prediksi. Penelitian ini juga bertujuan menghitung emisi gas metana berdasarkan hasil prediksi jumlah sampah menggunakan persamaan tertentu, sehingga memberikan gambaran lebih jelas mengenai potensi dampak lingkungan.

2.11 Matriks Penelitian

Tabel 2.3 memaparkan matriks penelitian prediksi *time series* dengan metode LSTM dan variannya. Informasi yang tercakup meliputi penulis, model, *optimizer*, serta metrik evaluasi seperti akurasi, MAE, MSE, RMSE, dan R^2. Penelitian sebelumnya umumnya menggunakan model LSTM dengan *optimizer* seperti Adam, RAdam, serta kombinasi AdaGrad dan RMSprop, namun hasil performanya bervariasi dan beberapa studi hanya melaporkan sebagian metrik evaluasi. Tidak ada penelitian yang mengonversi hasil

prediksi menjadi estimasi emisi metana. Penelitian ini mengusulkan penggunaan LSTM dengan *optimizer* RMSprop yang terintegrasi dengan konversi prediksi sampah menjadi estimasi emisi metana untuk menghasilkan *output* yang lebih aplikatif dan komprehensif.

Tabel 2. 3 Matriks Penelitian

Author	**Model**	***Optimizer***	**MAE**	**MSE**	**RMSE**	**R^2**	**Konversi Estimasi Gas Metana**
(Fadhel dkk., 2024)	e-LSTM	Radam	0.0409	0.0028	0.065	0.90	Tidak
(Loni Manikari dkk., 2024)	LSTM	Adam	-	0.02207	-	-	Tidak
(Pei dkk., 2022)	LSTM-RNN	Adam	0.1695	-	0.2102	-	Tidak
(Huang dkk., 2020)	LSTM	AdaGrad dan RMSProp	6.79; 5.48	-	10.14; 5.95	-	Tidak
(Yuniar dkk., 2025)	LSTM	Adam	2,633,536	54,711,498	7,396,722	-	Tidak
Penelitian ini	LSTM	RMSProp	?	?	?	?	Ya