BAB II

TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Penelitian yang membahas mengenai model hidrologi menggunakan algoritma pembelajaran mendalam untuk pengelolaan air yang lebih baik. Model *Single-Output Long Short-Term Memory* (LSTM-SO) dan *Encoder-Decoder Long Short-Term Memory* (LSTM-ED) dikembangkan, dan kinerjanya dibandingkan dengan menggunakan variabel input yang berbeda (Tsumugu dkk, 2022). Peneliti menggunakan data ketinggian air dan curah hujan dari 2018 hingga 2020 di Waduk Takayama (Prefektur Nara, Jepang) untuk melatih, menguji dan menilai kedua model. Kesalahan *Root-Mean-Squared* dan *Efisiensi Nash-Sutcliffe* diperkirakan untuk membandingkan kinerja model. Curah hujan masa lalu dan perubahan ketinggian air dimasukkan ke dalam lapisan input dan ketinggian air yang di prediksi pada waktu setelah nilai input terakhir dimasukkan ke dalam lapisan output. Struktur dasar yang digunaka menghasilkan output diperoleh sebagai input selanjutnya. Proses ini diulangi terus menerus dan melakukan prediksi hingga 24 jam. Hasil penelitian menunjukkan bahwa model LSTM-ED memiliki akurasi yang lebih baik. Namun, akurasi model secara signifikan lebih rendah saat memprediksi ketinggian air diluar kisaran dataset pelatihan.

Penelitian tentang analisis teknik berbasis AI untuk memperkirakan ketinggian air menurut curah hujan mengusulkan SVM dan teknik peningkatan *gradien* digunakan untuk AI pembelajaran mesin (Chorong dan Chung-Soo, 2021). AI *deep learning*, prakiraan ketinggian air dilakukan menggunakan jaringan LSTM di antara RNN yang digunakan untuk analisis deret waktu atau rentan waktu. Data yang dipakai berupa pengamatan curah hujan di Sabang-gyo dan ketinggian air observatorium di jembatan *Gloucester Valley Battle Monumenttahun* 2009-2020. Hasil dari penelitian menyatakan bahwa LSTM memiliki kesalahan terkecil dan kinerja analisis terbaik dibandingkan dengan

DNN dan SVM dalam memprediksi muka air tertinggi dan waktu terjadinya. Tinjaun pustaka yang lain dapat dilihat pada Tabel 2.1.

Tabel 2.1 Tinjauan pustaka

Referensi	Algoritma	Keterangan
(Venkata dkk, 2022)	CNN dan LSTM (deep learning) dan Automated DL (AutoDL)	Mengeksplorasi kesesuaian didalam mengadopsi AutoDL untuk Penilaian Kualitas Air dengan menggambarkan perbandingan antara AutoDL dan model konvensional dan analisis untuk meramalkan kualitas air.
(Jin dkk, 2020)	Cost-Sensitive Long Short-Term Memory (CSLSTM)	Mengusulkan mekanisme bersahabat dengan biaya yang harus dikeluarkan untuk memfasilitasi model dalam mempelajari representasi fitur contoh dan fluktuasi. (CSLSTM) untuk memprediksi PWL (tingkat air pressurizer) dari PWR (reaktor air bertekanan) reaktor nuklir.
(Tao dkk, 2020)	Multilayer Perceptron (MLP) dan Recurrent Neural Network (RNN)	Untuk mengatasi masalah yang disebabkan oleh penyebaran fitur hidrodinamik melalui aliran air MLP memanfaatkan fitur hidrodinamik dengan melatih *spasial* dan *temporal* serta mencari korelasi saluran yang berdekata langkah-langkah *time-steps* dan RNN untuk memprediksi ketinggian air berdasarkan data historis. .

2.2. Dasar-Dasar Teori

2.2.1 Deep Learning

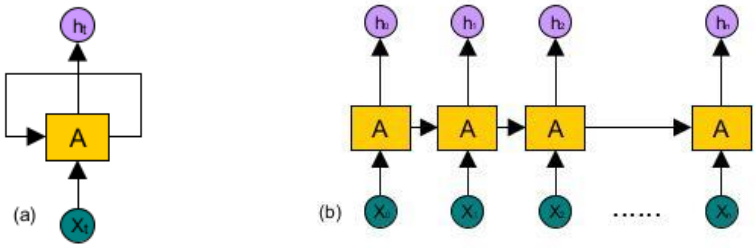
Deep leraning atau pembelajaran mendalam masih terhitung merupakan area baru didalam penelitian ML yang diperkenalkan dengan tujuan menggerakkan pembelajaran mesin lebih dekat dengan salah satu tujuan aslinya yaitu kecerdasan buatan. Pembelajaran mendalam mengimplementasikan *deep*

architecture yang dapat mempresentasikan data secara lebih efisien dengan unit komputasi lebih sedikit untuk fungsi sama dengan cara meningkatkan pembelajaran, aksesibilitas dan data latihnya (Venkata dkk, 2022). Pembelajaran mendalam sebagian besar metodenya menggunakan arsitektur jaringan saraf, itulah sebabnya model pembelajaran mendalam atau *Deep learning* (DL) sering disebut sebagai *Deep Neural Networks*. Pembelajaran mendalam mengacu kepada DNN yang mengadaptasi sistem kerja otak manusia yang tersusun dari sejumlah *neuron* membentuk jaringan. *Deep learning* melakukan pelatihan baik *supervised* maupun *unsupervised* (Shuai dkk, 2021). Pembelajaran mendalam mencakup banyak jaringan seperti diantaranya *Convulution Neural Network* (CNN), RNN, RNN dengan mengembangkan LSTM dan *Deep Belife Network* (DBN).

Mengatasi masalah *exploding vanishing gradients* dalam RNN, telah banyak variasi dikembangkan, salah satu paling terkenal adalah model LSTM. Dalam konsep dasarnya *recurrent unit* LSTM mencoba untuk "mengingat" semua pengetahuan dimasa lalu serta untuk "melupakan" data tidak relevan (Shuai dkk, 2021). Hal ini dilakukan dengan memperkenalkan lapisan fungsi aktivasi berbeda yang disebut "*gate*" untuk tujuan berbeda. Setiap unit *recurrent* LSTM juga memelihara vektor disebut *Cell state* yang secara konseptual menggambarkan informasi yang dipilih untuk disimpan oleh *recurrent unit* LSTM sebelumnya.

2.2.2 Recurrent Neural Networks (RNN)

RNN adalah salah satu metode *deep neural network learning* yang mampu memproses data *sequensial*. Manusia didalam membuat dan mengambil keputusan tidak selalu berdasarkan informasi pada saat itu, namun cenderung juga memperhitungkan informasi sebelumnya atau masa lalu. RNN meniru manusia didalam membuat kaputusan dengan cara mencari informasi dari masa lalu dengan melakukan *looping* didalam arsitekturnya seperti pada Gambar 2.1 (a) di bawah, secara otomatis informasi tersebut tersimpan dan dapat digunakan jika diperlukan sewaktu-waktu (Tao dkk, 2020).



Gambar 2.1 (a) Proses perulangan RNN (b) Multi salinan jaringan RNN

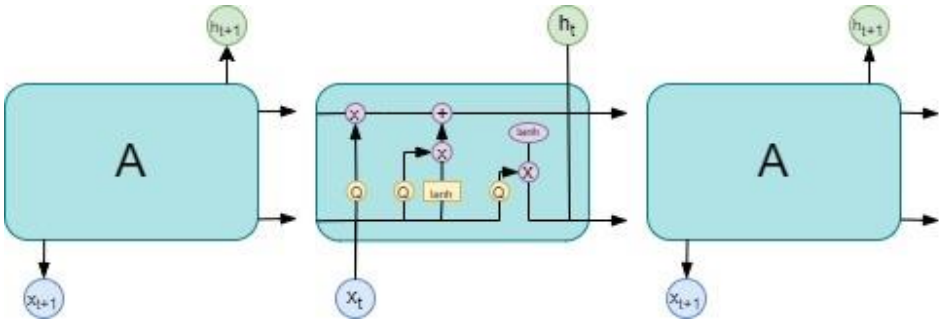
x_t sebagai input, h_t sebagai output dan terdapat *looping* atau perulangan yang memungkinkan informasi atau data diproses dari satu langkah ke langkah berikutnya. RNN dapat diasumsikan memiliki salinan berlapis dari jaringan yang sama dan kemudian selanjutnya mengirimkan pesan atau informasi tersebut kepada salinan berikutnya seperti pada Gambar 2.1 (b) di atas.

2.2.3 Long Short-Term Memory (LSTM)

LSTM merupakan salah satu jenis dari arsitektur RNN yang memiliki kelebihan untuk memprediksi dan mempelajari data sekuensial. LSTM memiliki struktur memori yang dapat mengingat informasi untuk jangka waktu yang lebih lama dan memiliki tiga gerbang *input gate*, *forget gate* dan *output gate* untuk mengatur aliran informasi masuk dan keluar dari sel. Komponen utama lapisan LSTM disebut sebagai sel memori (Yongping dkk, 2020), yang membedakan RNN dengan LSTM terletak pada penggunaan jumlah *layer*, yaitu *layer* aktivasi *tanh*, sedangkan untuk LSTM memiliki empat layer pada perulangan modelnya seperti pada Gambar 2.2 di bawah (Kamilya dan Alex, 2019).

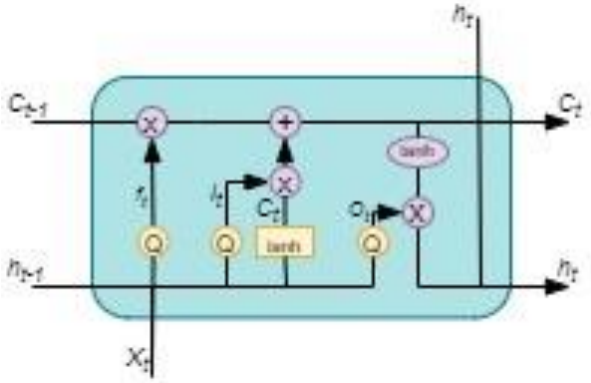
Exploding vanishing gradients atau masalah di mana kesalahan besar *gradien* menumpuk dan menghasilkan pembaruan yang sangat besar untuk bobot model jaringan saraf selama pelatihan model RNN, banyak variasi metode yang sudah dikembangkan salah satunya paling terkenal adalah LSTM. Konsep dasarnya *recurrent unit* LSTM mencoba untuk "mengingat" semua pengetahuan atau informasi dimasa lalu serta agar bisa "melupakan" data tidak relevan, hal ini dilakukan dengan memperkenalkan lapisan fungsi aktivasi berbeda disebut "*Gate*" untuk tujuan yang berbedap pula. *Unit recurrent* LSTM juga memelihara *vektor*

yang disebut *cell state* secara konseptual menggambarkan informasi yang dipilih untuk disimpan oleh *recurrent unit* LSTM sebelumnya.



Gambar 2.2 Perulangan empat layar LSTM

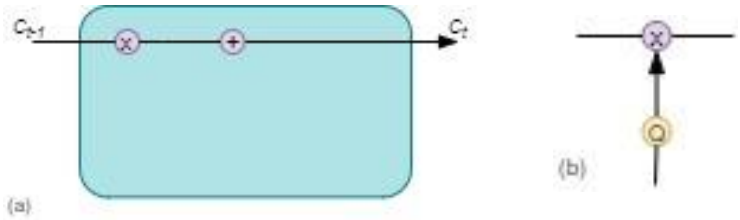
Gambar 2.2 di atas merupakan perulangan empat layer metode LSTM jika dilakukan proses perulangan sebanyak *time-stape* yang ada, setiap *time-stape* tersebut mengandung selnya dan gerbangnya masing-masing berdasarkan *cell* LSTM yang dilakukan dalam pemerosesan.



Gambar 2.3 Arsitektur model LSTM

Cell pada LSTM dapat dilihat pada Gambar 2.3 terdiri dari *layer neuron* dilambangkan dengan persegi panjang seperti pada Gambar 2.4, operasi *element-wise* dilambangkan dengan lingkaran. Panah hitam melambangkan aliran informasi didalam *cell* dan antar *cell* maupun keluaran dari *cell* (*output h*). *Cell* LSTM mempunyai 2 hasil keluaran, pertama adalah informasi yang sebenarnya *hidden state* (h_t) diteruskan ke *cell* selanjutnya menjadi *input* dari *cell* selanjutnya, kedua yaitu *cell state* (C_t), C_t merupakan kunci utama dari LSTM. *cell state* merupakan garis horizontal yang menghubungkan semua *output layar* pada LSTM seperti pada Gambar 2.4 bagian (a) di bawah. LSTM memiliki

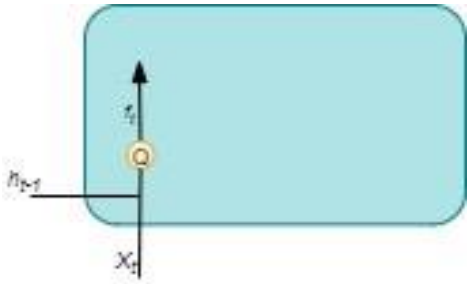
kemampuan untuk menambah atau menghapus informasi dari *cell state* yang disebut *gates*, *gate* ini berfungsi mengatur informasi masuk secara opsional dengan menggunakan *sigmoid layer* yang ada pada Gambar 2.4 bagian (b). Keluaran dari *sigmoid layer* menunjukkan informasi diteruskan dengan simbol angka 1 atau dihentikan dengan simbol angka 0.



Gambar 2.4 (a) *Cell state* pada LSTM (b) *Sigmoid* layar pada LSTM

LSTM memiliki satu *cell state* dan 3 langkah *gates* yaitu, *forget gates* (i_t), *input gates* (f_t) dan *output gates* (o_t). *Cell state* berfungsi sebagai memori dari jaringan, *forget gates* berfungsi untung memutuskan informasi yang dihapus dari *cell*, *input gates* memutuskan nilai dari *input* untuk diperbarui pada *state* memori dan *output gates* berfungsi untuk memutuskan apakah yang dihasilkan *output* sesuai dengan *input* dan memori pada *cell* atau tidak. Proses berjalannya metode LSTM berdasarkan langkah-langkahnya sebagai berikut (Kamilya dan Alex, 2019).

A. Langkah pertama adalah memutuskan informasi apa yang dihapus dari *cell state*, keputusan ini dibuat oleh *sigmoid gate layer*. *Forget gate layer* memproses h (t_{-1}) dan x_t sebagai *input* dan menghasilkan *output* berupa angka 0 atau 1 *cell state* C_{t-1} seperti pada Gambar 2.5 di bawah ini.



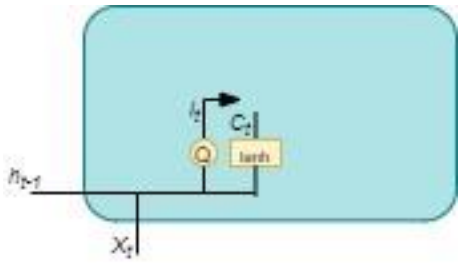
Gambar 2.5 *Forget gate* layer LSTM

Formula *forget gate* dinyatakan dengan Formula 2.1.

$$f_t = \sigma\left(w_f[h_{t-1}, x_t] + b_f\right) \qquad (2.1)$$

di mana f_t nilai dari *forget gate*, w_f adalah bobot untuk nilai *input* pada waktu ke t, h_{t-1} adalah nilai *output* dari waktu ke t_{-1}, x_t adalah nilai *input* pada waktu ke *t*, b_f adalah bias pada *forget gate* dan σ adalah aktivasi *fungsi sigmoid*.

B. Langkah kedua bertujuan untuk memutuskan informasi yang disimpan *cell state.* Langkah kedua ini memiliki 2 bagian, pertama aktivasi *sigmoid layer* bernama *input gate layer* untuk memutuskan nilai yang diperbarui, bagian kedua menggunakan aktivasi *tanh layer* untuk membuat satu kandidat dengan nilai baru ($\tilde{C}_t$) yang dapat ditambahkan ke *cell state*. Tahapan selanjutnya hasil dari aktivasi *sigmoid* dan *tanh* digunakan untuk memperbarui *cell state* yang baru seperti pada Gambar 2.6 di bawah ini.



Gambar 2.6 *Input gate* layer dan *tanh* layer LSTM

Formula *input gate* dinyatakan dengan Formula 2.2 dan formula kandidat baru dinyatakan dengan Formula 2.3.

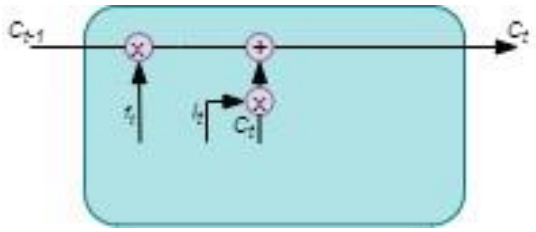
$$i_t = \sigma(w_i[h_{t-1}, x_t] + b_i) \qquad (2.2)$$

di mana i_t nilai dari *input gate*, w_i adalah bobot untuk nilai *input* pada waktu ke t, h_{t-1} adalah nilai *output* dari waktu ke t_{-1}, x_t adalah nilai *input* pada waktu ke *t*, b_i adalah bias pada *input gate* dan σ adalah *fungsi sigmoid*.

$$\tilde{C}_t = tanh(w_c[h_{t-1}, x_t] + b_c) \qquad (2.3)$$

di mana $\tilde{C}_t$ nilai kandidat *cell state*, w_c adalah bobot untuk nilai *input* pada *cell* ke c, h_{t-1} adalah nilai *output* dari waktu ke t_{-1}, x_t adalah nilai *input* pada waktu ke *t*, b_c adalah bias pada *cell* ke c dan tanh adalah *fungsi hyperbolic tangent*.

C. Langkah ketiga bertujuan untuk memperbarui *cell state* lama (C_{t-1}) menjadi *cell state* baru C_t seperti pada Gambar 2.7 di bawah. Perkalian *state* lama dengan f_t bertujuan untuk menghapus informasi yang sudah ditentukan sebelumnya pada langkah *forget gate*, selanjutnya ditambahkan dengan $i_t * \tilde{C}_t$ yang merupakan nilai baru dan digunakan untuk memperbarui *state* berikutnya.



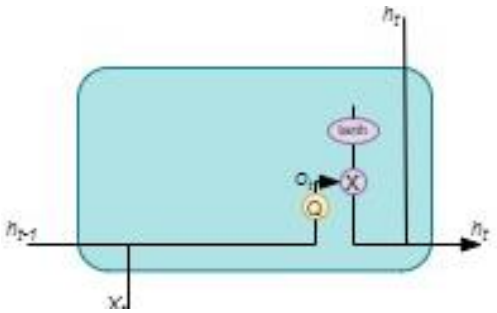
Gambar 2.7 Pembuatan *cell state* baru

Formula *cell state* dinyatakan dengan Formula 2.4.

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \qquad (2.4)$$

di mana C_t nilai *memory cell state*, i_t adalah nilai dari *input gate*, $\tilde{C}_t$ adalah nilai kandidat *memory cell state*, f_t adalah nilai *forget gate* dan C_{t-1} adalah nilai memori *cell state* pada *cell* sebelumnya.

D. Langkah terahir adalah bertujuan untuk memutuskan hasil *output* seperti Gambar 2.8 di bawah, output harus sesuai dengan *cell state* yang telah diproses terlebih dahulu. Pertama, hasil aktivasi *sigmoid* memutuskan bagian *cell state* yang menjadi *output*, kedua, hasil aktivasi *sigmoid* dikalikan dengan hasil penjumlahan aktivasi *tanh* dengan *cell state* baru (untuk mengganti nilai menjadi diantara -1 dan 1) dan menghasilkan *output* yang sesuai dengan apa yang diputuskan sebelumnya.



Gambar 2.8 Pembuatan *output gate*

Formula *output gate* dinyatakan dengan Formula 2.5 dan Formula 2.6.

$$0_t = \sigma(w_0[h_{t-1}, x_t] + b_0) \qquad (2.5)$$

di mana 0_t nilai dari *output gate*, w_0 adalah bobot untuk nilai input pada waktu ke t, h_{t-1} adalah nilai *output* dari waktu ke t_{-1} , x_t adalah nilai input pada waktu ke t, b_0 adalah bias pada *output gate* dan σ adalah *fungsi sigmoid*.

$$h_t = 0_t * tanh(C_t) \qquad (2.6)$$

di mana h_t *output* final, 0_t adalah nilai *output gate*, C_t adalah nilai memory *cell state* yang baru dan *tanh* adalah *fungsi tanh*.

2.2.4 Pengoptimalan Adam

Pengomptimalan Adam berasal dari kalimat *adaptive moment estimation* merupakan metode optimasi berbasis stokastik orde pertama dari fungsi objektif stokastik untuk memperbarui bobot jaringan berulang berdasarkan data pelatihan (Zihan dkk, 2019). Optimasi Adam bertujuan menghitung *adaptive learning rate* untuk setiap parameter, learning rate tersebut menentukan seberapa banyak perubahan bobot pada jaringan. LSTM adalah salah satu modifikasi atau pengembangan dari metode RNN dengan menggunakan banyak sekali data dan memanfaatkan pemerosesan yang saling terhubung dengan *gate* yang ada, oleh karena itu optimasi ini dibutuhkan untuk menhilangkan *noise* pada model LSTM yang dibuat. Cara kerja Adam dapat digambarkan sebagai kombinasi dari dua *stochastic gradient descent*, yaitu *AdaGrad* dan *RMSProp*. Dengan kombinasi ini, Adam mampu mengoptimasi algoritma LSTM yang dapat menangani masalah

noise atau data tidak berguna yang tidak dapat diinterpretasikan oleh *tools* yang di pakai (Zihan dkk, 2019), formula dari optimasi Adam sebagai berikut (Balaji dkk, 2021).

a. Menhitung *gradien* pada g_t waktu t
b. Memperbarui bias momen pertama

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \quad (2.7)$$

c. Memperbarui bias momen kedua

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t \quad (2.8)$$

d. Menghitung nilai koreksi momen pertama

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \quad (2.9)$$

e. Menghitung nilai koreksi momen kedua

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \quad (2.10)$$

f. Memperbarui parameter dengan menggunakan *learning rate*

$$\theta_t = \theta_{t-1} - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}} \quad (2.11)$$

Keterangan seluruh notasi langkah optimasi Adam dijelaskan pada Tabel 2.2 di bawah ini.

Tabel 2.2 Keterangan seluruh notasi pada formula optimasi Adam

Notasi	Keterangan
$\boldsymbol{g_t}$	Nilai gradien
$\boldsymbol{m_t}$	Nilai first momment ke t
$\boldsymbol{v_t}$	Nilai second momment ke t
$\boldsymbol{\beta_1}$	Nilai beta pertama
$\boldsymbol{\beta_2}$	Nilai beta kedua
$\boldsymbol{\hat{m}_t}$	Jumlah Nilai first momment ke t
$\boldsymbol{\hat{v}_t}$	Jumlah Nilai second momment ke t
$\boldsymbol{\theta_t}$	Nilai prameter yang baru
$\boldsymbol{\varepsilon}$	Nilai epsilon
$\boldsymbol{n}$	Jumlah data observasi
α	Nilai *learning rate*

2.2.5 Evaluasi Akurasi Model Prediksi

Penelitian ini akan melewati proses evaluasi untuk menentukan hasil yang didapatkan, evaluasi yang dilakukan menggunakan MAPE untuk mencari nilai akurasi. MAPE atau *Mean Absolute Percentage Error* adalah salah satu metode metrik akurasi prediksi model paling umum digunakan dengan menjadikan hasil nilainya berbentuk persentase (Aishah dan Zeyar, 2022) (Brenon dkk, 2021). MAPE mengukur besarnya rata-rata kesalahan yang dihasilkan oleh model atau seberapa jauh prediksi rata-rata yang dihasilkan. MAPE dapat dianggap sebagai *loss function* untuk menentukan kesalahan yang disebut dengan evaluasi model. Dengan menggunakan model MAPE dapat memperkirakan akurasi dalam hal perbedaan nilai prediksi atau $\hat{y}_i$ dengan nilai sebenarnya atau y_i. Metode ini populer digunakan karena meyajikan kesalahan berbentuk persentase dan membuatnya mudah dipahami oleh pengguna serta mudah untuk membandingkan hasil akurasi model di seluruh kasus penggunaan serta terhadap kumpulan data (Pengfei dkk, 2023). Formula evaluasi akurasi menggunakan MAPE dapat dilihat pada Formula 2.12 di bawah ini (Shuai dkk, 2021).

$$MAPE = \frac{1}{n}\sum_{i=1}^{n}\frac{|y_i - \hat{y}_i|}{y_i} * 100 \qquad (2.12)$$

di mana nilai y_i adalah nilai sebenarnya atau aktual ke *i* dari data observasi, $\hat{y}_i$ adalah nilai prediksi ke *i* yang dihasilkan dari data observasi yang dipakai dan n adalah jumlah data observasi yang digunakan pada pembuatan model.

2.2.6 Ketinggian Permukaan Air

Ketinggian air biasanya digunakan sebagai acuan untuk mengukur ketinggian dan dalamannya suatu tempat, sungai, kali, waduk teluk dan lainnya. Studi terbaru tentang prediksi ketinggian air telah berkonsentrasi pada penurunan permukaan danau, serta berkurangnya debit ke aliran sungai (Jin dkk, 2020). Alat atau metode yang mampu dalam melakukan prediksi status ketinggian air sangat penting demi dapat mengelola sungai, kali, danau, konsumsi air, persediaan air, rekreasi, navigasi, irigasi, pembangkit listrik tenaga air dan banyak lagi dengan baik. Pengukuran fisik dan pemodelan digunakan untuk memetakan prediksi

ketinggian air, sedangkan dengan menggunakan metode pemodelan menjadi alternatif yang lebih murah dan efektif (Viet-Ha dkk, 2020).

Ketinggian air diukur dengan metode prediksi mulai populer dilakukan karena dapat memberikan efisiensi dan efektifitas yang lebih baik (Venkata dkk, 2022). Sungai Ciliwung hampir meliputi seluruh DKI Jakarta yang akhirnya menyebabkan banyaknya pintu air yang harus diawasi secara bersamaan sehingga membutuhkan sebuah solusi metode kumputasi dalam mempermudah melakukan prediksi ketinggian pemukaan air sungai. Penanggulangan terjadinya peluapan air sungai sangat memungkinkan jika sudah memiliki hasil prediksi lebih awal sehingga meminimalisir segala kemungkinan buruk yang mungkin terjadi seperti banjir dan lainnya (HirokI dkk, 2016).

2.2.7 Sungai Ciliwung

Sungai Ciliwung merupakan salah satu sungai terbesar dan terpanjang yang mengalir di wilayah Provinsi DKI Jakarta. Mata air Sungai Ciliwung terdapat di Gunung Pangrango Jawa Barat. Sungai Ciliwung mengalir ke arah Jakarta melalui Puncak, Kabupaten Bogor, Kota Bogor, Kota Depok dan bermuara ke Teluk Jakarta. Panjang Sungai Ciliwung dari hulu hingga ke muara kurang lebih 117 km dengan luas Daerah Aliran Sungai (DAS) sekitar 347 km^2. Saat ini sungai Ciliwung masih digunakan sebagai salah satu sumber air utama oleh masyarakat dan industri yang berada di bantaran sungai (Syifa, 2019). Nama-nama pintu air yang ada di sungai Ciliwung seperti yang ada pada Tabel 2.3 di bawah ini:

Tabel 2.3 Nama-nama pintu air sungai Ciliwung

No	Nama Pintu Air	7	Pintu Air Flusing Ancol
1	Bendung Cibalok Gadong	8	Pintu Air Cibalok Gadong
2	Bendung Katulampa (Hulu)	9	Pintu Sungai Katulama (Hulu)
3	Pintu Sungai Depok		
4	Pintu Air Mangarai		
5	Pintu Air Istiqlal		
6	Pintu Air Jembatan Merah		